Graduate School Introductory Computational Simulation Course Pedagogy

by

Laura L. Proctor

Submitted to the Department of Computation for Design and Optimization
in partial fulfillment of the requirements for the degree of

Master of Science in Computation for Design and Optimization

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June 2009

Author .....
Department of Computation for Design and Optimization
January 9, 2009

Certified by .....
Jacob K. White
Cecil H. Green Professor of Electrical Engineering & Computer Science
Thesis Supervisor

Accepted by .....
Jaime Peraire
Professor of Aeronautics and Astronautics
Co-Director, Department of Computation for Design & Optimization

Graduate School Introductory Computational Simulation Course Pedagogy

by

Laura L. Proctor

Submitted to the Department of Computation for Design and Optimization
on January 12, 2009, in partial fulfillment of the
requirements for the degree of
Master of Science in Computation for Design and Optimization

Abstract

Numerical methods and algorithms have developed and matured vastly over the past three decades now that computational analysis can be performed on almost any personal computer. There is a need to be able to teach and present this material in a manner that is easy for the reader to understand and be able to go forward and use. Three popular course at MIT were without lecture notes; in this thesis the lecture notes are presented. The first chapter covers material taught in Numerical Methods for Partial Differential Equations (2.097/6.339/16.920) specifically the Integral Equation Methods section of this course, chapter two shows the notes for the course Introduction to Numerical Simulation (2.096/6.336/16.910), and chapter three contains the notes for the class Foundations of Algorithms and Computational Techniques in Systems Biology (6.581/20.482). These course notes give a broad overview of many algorithms and numerical methods that one can use to solve many problems that span many fields - from biology to aerospace to electronics to mechanics.

Thesis Supervisor: Jacob K. White
Title: Cecil H. Green Professor

Acknowledgments

I would like to express my unending gratitude to Professor Jacob White for guiding me through this time. He has shown me a compassion that I can only hope to emulate, a patience that is rare, and a massive knowledge for numerical methods and integral equation methods. All of this, and a great sense of humor that puts one at ease.

A great thanks to the other professors with whom I've had the opportunity to work – notedly Professor Luca Daniel for whom I was a teaching assistant in the Introduction to Numerical Simulation course. He is a great teacher, very enthusiastic, and easy to approach. Also, Professor Jaime Peraire, for whom I was a teaching assistant in the Numerical Methods for Partial Differential Equations course. He is also a great teacher, an understanding person, straightforward, and respectable.

I also want to thank my colleagues who have been wonderful in their own rite, and always willing to lend a hand: Bo Kim, JungHoon Lee, Brad Bond, Tarek Moselhy, Lei Zhang, Kin Sou, Dima Vasilyev, Jay Bardhan, Homer Reid, David Joe Willis, and Carlos Pinto Coelho.

In addition, I want to thank my wonderful fiance, Kevin Flaherty, who has been there for me throughout my long graduate student career.

Contents

Introduction

Numerical methods and algorithms have developed and matured vastly over the past three decades now that computational analysis can be performed on almost any personal computer. There is a need to be able to teach and present this material in a manner that is easy for the reader to understand and be able to go forward and use. Three popular course at MIT were without lecture notes; in this thesis the lecture notes are presented. The first chapter covers material taught in Numerical Methods for Partial Differential Equations (2.097/6.339/16.920) specifically the Integral Equation Methods section of this course, chapter two shows the notes for the course Introduction to Numerical Simulation (2.096/6.336/16.910), and chapter three contains the notes for the class Foundations of Algorithms and Computational Techniques in Systems Biology (6.581/20.482). These course notes give a broad overview of many algorithms and numerical methods that one can use to solve many problems that span many fields - from biology to aerospace to electronics to mechanics.

Chapter 1

Integral Equation Methods

Numerical Methods for Partial Differential Equation is a course that covers several techniques for discretizing partial differential equations in order to solve them. Very often partial differential equations do not have analytic solutions and one needs to apply an appropriate method to solve them. In this thesis, the Integral Equation Methods used for solving Partial Differential Equations is covered.

1.1 Discretization of Boundary Integral Equations

Numerical Methods for PDEs

Boundary Element Methods, Lecture 1
Introduction to Discretization of Boundary Integral Equations

L. Proctor, S. De, C. Coelho, D. Willis, X. Wang, & J. White

November 23, 2008

1 Module Outline - 6 Lectures

Overview Integral Equation Methods
Applications: Aero, MEMS, IC Design
Informal Overview:
Formulation & Discretization
Quadrature and Cubature for computing integrals
1-D, 2-D, and dealing with Singularities
1st and 2nd Kind Theory - Part 1
Discretization
Formulating 3-D Integral Equations
Radiation Conditions, Ansatz & Green's Formulations
1st and 2nd Kind Theory - Part 2
Convergence theory
Fast Multipole and FFT-based methods

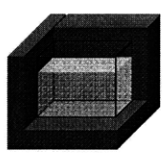
2 Outline for Today

Background
Exterior versus interior problems
Point source approach
Test Function Selection
Collocation Method
Galerkin Method
Some issues in 3D
Singular integrals

3 Background

3.1 Interior vs Exterior Problems

Interior Exterior



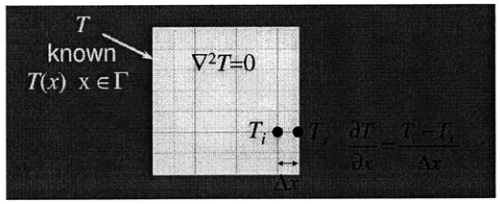
Temperature in a tank *Ice cube in a bath*

<u>What is the heat distribution?</u>

Heat flow = Thermal conductivity $\int_{surface} \frac{\partial T}{\partial n}$

3.1.1 The Interior Problem

Example: Heat Distribution in a Tank



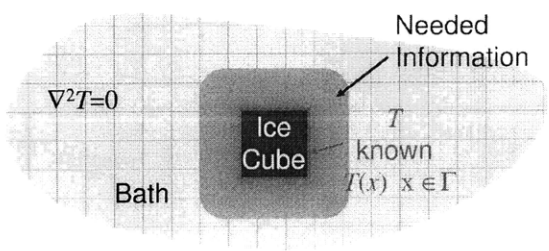
How does one determine the heat distribution using the finite difference method?

Use the above example of heat distribution in a tank, to see what is meant by an interior problem. This is a Dirichlet problem because $T(\vec{x})$, the temperature, is defined on the surface $\vec{x} \in \Gamma$. The steady state two dimensional heat flow equation is defined using the Laplace Equation, $\nabla^2 T(\vec{x}) = 0$. The domain of this problem is clearly the <u>interior</u> of the tank, $\vec{x} \in \Omega$.

3.1.2 The Exterior Problem

Up until this time, all problems that have been studied using finite-element and finite-difference methods have been *interior* problems, but now, we begin to wonder how to form an <u>exterior</u> problem. This poses some problems for these other methods such as generating the grid as well as mesh truncation.

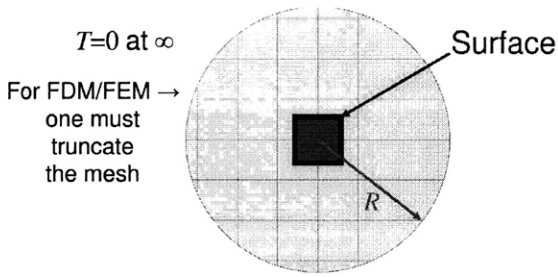
Example: Ice Cube in a Bath



Above is a problem showing an *ice cube in a bath* which is used to indicate an exterior problem. Again, this is a Dirichlet problem because $T(\vec{x})$ is defined on the surface $\vec{x} \in \Gamma$ and described by the steady state two dimensional heat flow equation defined using Laplace's Equation, $\nabla^2 T(\vec{x}) = 0$. The problem domain is the infinitely extending region exterior to the ice cube. A point that we will expand in a later lecture is that with exterior problems, an additional boundary condition is needed to specify what happens at a large distance away from our point source. Assuming there are no heat sources exterior to the cube will impose the following *radiation* boundary condition

$$\lim_{\|\vec{x}\| \to \infty} T(\vec{x}) \to 0.$$

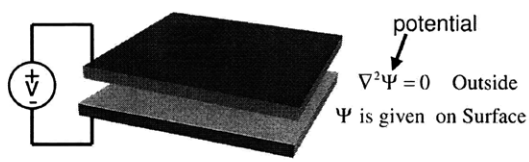
Suppose that for this specific problem, we are only interested in what occurs at the surface of the cube. It seems inefficient to use the finite-difference or finite-element methods where one needs to compute the temperature everywhere in Ω.



3.2 Examples

3.2.1 Computation of Capacitance

What is the capacitance?



$$\text{Capacitance} = \text{Dielectric Permittivity} \int \frac{\partial \Psi}{\partial n}$$

In the example in the slide, the yellow plates form a parallel-plate capacitor with an applied voltage V. In this 3-D electrostatics problem, the electrostatic potential Ψ satisfies Laplace's equation $\nabla^2 \Psi(x) = 0$ in the region exterior to the plates, and the potential is known on the surface of the plates (Dirichlet boundary condition). Furthermore, far from the plates,

$$\lim_{\|\vec{x}\| \to \infty} \Psi(\vec{x}) \to 0.$$

(Exterior Radiation Boundary Condition to be studied further in a future lecture). The value of interest is the capacitance, C, which satisfies

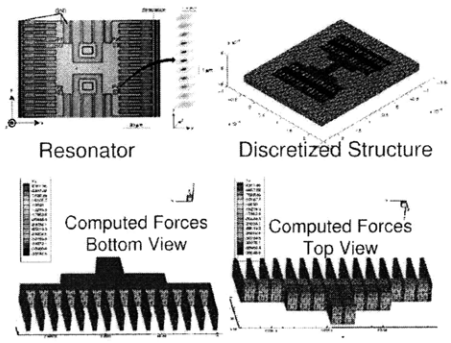
$$q = CV$$

where q, the net charge on one of the plates, is given by the surface normal of the potential integrated over one plate and scaled by a dielectric permittivity.

Note 1 ***Example 1: Capacitance problem***

This is a typical application example, determining the charge density on the surface of conducting plates given an applied voltage. In this particular example, the top plate potential is $\Psi = 0.5V$ and the bottom plate potential is $\Psi = -0.5V$, where V is the voltage noted in the figure.
For this exterior Dirichlet problem, one can write an integral equation that relates the surface charge density on the plates σ to the potential on the plates. This integral equation, $\Psi(\vec{x}) = \int_\Gamma \frac{1}{\|\vec{x}-\vec{x}'\|}\sigma(\vec{x}')dS'$, is often referred to by physicists as the superposition integral. In the integral equation, x is any point on the plate surfaces and the surface being integrated over is the union of the top and bottom plate surfaces. Note that the integration surface is not a connected domain, but this presents no difficulties.

3.2.2 Drag Force in a Microresonator



Note 2 ***Example 2: Drag force in a MEMS device***

The example in the slide is a microresonator, it is a structure that can be made to vibrate using electrostatic forces. The changing character of those vibrations can be used to sense rotation. The particulars of how the microresonator operates is not directly relevant to our discussion of integral equations, except for one point. In order to determine how much energy is needed to keep the microresonator vibrating, it is necessary to determine the fluid drag force on comb structures shown in the bottom part of the slide. The fluid is the air surrounding the structure, and at the micron-scale of these devices, air satisfies the incompressible Stokes equation,

$$\begin{aligned} \nabla^2 u(x) &= \nabla p(x) \\ \nabla \cdot u(x) &= 0 \end{aligned} \tag{1}$$

where u is the fluid velocity and p is the pressure. By specifying the comb velocity, and then computing the surface pressure and the normal derivative of velocities tangent to the surface, one can determine the net drag force on the comb. Once again, this is a problem in which the known quantities (the

comb velocity) and the quantities of interest (the derivative of the tangential components of fluid velocity) are on the surface.

3.2.3 Aircraft Drag

Discretization for F-18 pressure simulation (no lift)

Inviscid, Irrotational, Steady Flow

Potential flow: $\nabla^2 u(x) = 0 \qquad \nabla u = \text{velocity}$

Note 3 ***Example 3: Aircraft Drag***

The potential flow model for aircraft drag computation will be discussed in more detail in subsequent lectures, so we only give a brief description here. In order to compute the drag on the wing of an aircraft, one must determine the difference between the wing velocity and the velocity of the air very close to the wing. If the air can be assumed inviscid, irrotational, and incompressible, the velocity is given by the gradient of a scalar potential which satisfies Laplace's equation. The boundary conditions for the Laplace's equation are given as a velocity boundary condition on the aircraft surface, equivalently a Neumann condition on the potential, and it is usually assumed that the potential approaches zero at infinity. The boundary condition at infinity is more subtle than it may seem, as we shall see in later lectures. Finally, it is common to introduce an artificial boundary in the domain, and specify a condition on that boundary to introduce rotational effects.

3.2.4 Capacitance of Microprocessor Signal Lines

Note 4 ***Example 4: Capacitance of microprocessor signal lines***

This last example in the above slide is a picture of the wiring on a microprocessor integrated circuit. A typical microprocessor has millions of wires, so we are only looking at a small piece of a processor. The critical problem in this example is determining how long signals take to get from the output of a logical gate to the input of the next gate. To compute that delay, one must determine the capacitance on each of the wires given in the slide picture. To do so requires computing charges given electrostatic potentials as noted above.

3.3 Advantages of Integral Equation Method

3.3.1 What is common about these examples?

Exterior Problems

MEMS device - fluid (air) creates drag

Aircraft Design - exterior air flow

Signal Line - Exterior fields.

Quantities of interest are on surface

MEMS device - Just want surface traction force

Aircraft Design - Just want surface tangent velocities

Signal Line - Just want surface charge.

Exterior problem is linear and space-invariant

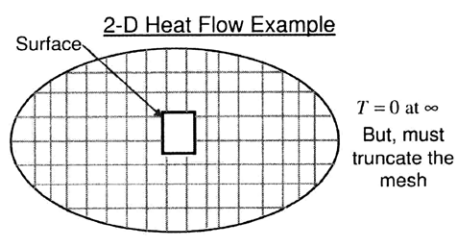
MEMS device - Exterior Stoke's flow equation (linear)

Aircraft Design - Laplace's equation, plus wakes.

Signal line - Laplace's equation in free spce (linear)

But problems are geometrically very complex

3.3.2 Why not use FDM / FEM?



Only need $\frac{\partial T}{\partial n}$ on the surface, but T is computed everywhere.
Must truncate the mesh, $\Rightarrow T(\infty) = 0$ becomes $T(R) = 0$.

Consider the two dimensional exterior heat conduction problem in the above figure in which the temperature is known on the surface of the square. Suppose the quantity of interest is the total heat flow out of the square.
The temperature T satisfies

$$\nabla^2 T(x) = 0 \qquad x \in \Omega$$

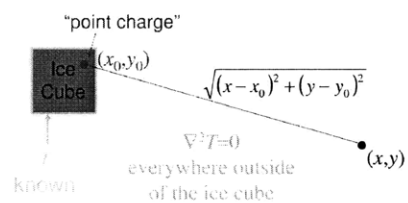
$$T(x) \; given \; x \in \Gamma \tag{2}$$
$$\lim_{\|x\| \to \infty} T(x) = 0$$

where Ω is the infinite domain outside the square and Γ is the region formed by the edges of the square.
Using finite-element or finite-difference methods to solve this problem requires introducing an additional approximation beyond discretization error. It is not possible to discretize all of Ω, as it is infinite, and therefore the domain must be truncated with an artificial finite boundary. In the slide,the artificial boundary is a large ellipse on which we assume the temperature is zero. Clearly, as the radius of the ellipse increases, the truncated problem more accurately represents the domain problem, but the number of unknowns in the discretization increases.

3.4 Point Source Approach

3.4.1 Green's Function

Heat Distribution in 2-D



Green's Function: $T = log\left(\sqrt{(x-x_0)^2+(y-y_0)^2}\right)$

From basic electrostatics, one knows that in 3-D, the potential field produced by a point charge decays inversely with the distance to the point charge. Since, roughly, one can represent any charge distribution using a sum of point charges, one can express the potential due to a charge density as a sum of point charge potentials. Therefore, point charge potentials play a special role, and are often referred to as Greens' functions for the problem.

<u>In 2-D</u> The potential due to a point charge is:

$$u = log\left(\sqrt{(x-x_0)^2+(y-y_0)^2}\right) \tag{3}$$
$$\forall (x,y) \neq (x_0,y_0)$$

<u>In 3D</u> The potential due to a point charge is:

$$u = \frac{1}{\sqrt{(x-x_0)^2+(y-y_0)^2+(z-z_0)^2}} \tag{4}$$
$$\forall (x,y,z) \neq (x_0,y_0,z_0)$$

$$\underline{\textbf{In 2D}} \qquad \text{If } u = log\left(\sqrt{(x-x_0)^2+(y-y_0)^2}\right)$$
$$\text{then } \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = 0 \quad \forall \quad (x,y) \neq (x_0, y_0)$$

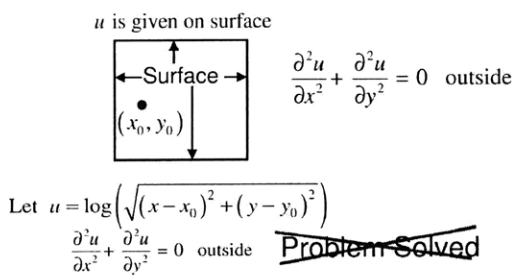
$$\underline{\textbf{In 3D}} \qquad \text{If } u = \frac{1}{\sqrt{(x-x_0)^2+(y-y_0)^2+(z-z_0)^2}}$$
$$\text{then } \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} + \frac{\partial^2 u}{\partial z^2} = 0 \quad \forall \quad (x,y,z) \neq (x_0, y_0, z_0)$$

Proof: Just differentiate and see!

In the next few slides, we will use an informal semi-numerical approach to derive the integral form of Laplace's equation. We do this in part because such a derivation lends insight to the subsequent numerical procedures.
To start, recall from basic physics that the potential due to a point charge is related only to the distance between the point charge and the evaluation point. In 2-D the potential is given by the log of the distance, equation (3), and in 3-D the potential is inversely proportional to the distance, equation (4). These functions are sometimes referred to as Green's functions for Laplace's equation, but have the physical interpretation as the potential due to a point charge. We will be studying Green's functions in more depth later on.

▷ ***Exercise 1*** Show by direct differentiation that the functions in equations (3) and (4) satisfy $\nabla^2 u = 0$, in the appropriate dimension almost everywhere. ■

3.4.2 Scaling Green's Function



Boundary conditions are not satisfied!

A simple idea for computing the solution of Laplace's equation outside the square is to let

$$u(x,y) = \alpha \log \sqrt{(x-x_0)^2+(y-y_0)^2}$$

where (x_0, y_0) is a point inside the square. Clearly u will always satisfy $\nabla^2 u = 0$ outside the square, but u may not match the boundary conditions. By adjusting

α, it is possible to make sure to match the boundary conditions at at least one point.
This concept is applied to a circle as a simple example of how to match the boundary conditions.

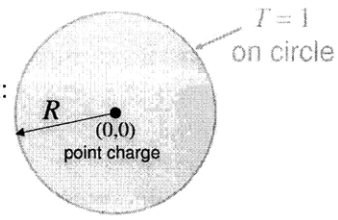
$$T = \log\left(\sqrt{x^2 + y^2}\right)$$

$$T(\vec{x} \in \Gamma) = \log R = 1?$$

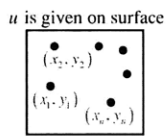
Multiply by a constant:

$$T = \alpha \log\left(\sqrt{x^2 + y^2}\right)$$

charge strength



▷ ***Exercise 2*** Suppose the potential on the surface of the square is a constant. Can you match that constant potential everywhere on the perimeter of the square by judiciously selecting α ? ■

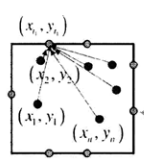


$$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = 0 \quad \text{outside}$$

$$u = \sum_{i=1}^{n} \alpha_i \log\left(\sqrt{(x - x_i)^2 + (y - y_i)^2}\right) = \sum_{i=1}^{n} \alpha_i G(x - x_i, y - y_i)$$

Pick the α_i's to match the boundary conditions!

To construct a potential that satisfies Laplace's equation and matches the boundary conditions at more points, let u be represented by the potential due to a sum of n weighted point charges in the square's interior. As shown in the slide, we can think of the potential due to a sum of charges as a sum of Green's functions. Of course, we have to determine the weights on the n point charges, and the weight on the ith charge is denoted hereby α_i.



Source Strengths selected to give correct potential at **test** points.

$$\begin{bmatrix} G(x_{t_1} - x_1, y_{t_1} - y_1) & \cdots & G(x_{t_1} - x_n, y_{t_1} - y_n) \\ \vdots & \ddots & \vdots \\ G(x_{t_n} - x_1, y_{t_n} - y_1) & \cdots & G(x_{t_n} - x_n, y_{t_n} - y_n) \end{bmatrix} \begin{bmatrix} \alpha_1 \\ \vdots \\ \alpha_n \end{bmatrix} = \begin{bmatrix} \Psi(x_{t_1}, y_{t_1}) \\ \vdots \\ \Psi(x_{t_n}, y_{t_n}) \end{bmatrix}$$

To determine a system of n equations for the n α_i's,consider selecting a set of n test points, as shown in the slide above. Then, by superposition, for each test point (x_{t_i}, y_{t_i}),

$$u(x_{t_i}, y_{t_i}) = \sum_{i=1}^{n} \alpha_i \log \sqrt{(x_{t_i} - x_0)^2 + (y_{t_i} - y_0)^2} = \sum_{i=1}^{n} \alpha_i G(x_{t_i} - x_0, y_{t_i} - y_0). \tag{5}$$

Writing an equation like (5) for each test point yields the matrix equation

$$\begin{bmatrix} G(x_{t_1} - x_1, y_{t_1} - y_1) & \cdots & G(x_{t_1} - x_n, y_{t_1} - y_n) \\ \vdots & \ddots & \vdots \\ G(x_{t_n} - x_1, y_{t_n} - y_1) & \cdots & G(x_{t_n} - x_n, y_{t_n} - y_n) \end{bmatrix} \begin{bmatrix} \alpha_1 \\ \vdots \\ \alpha_n \end{bmatrix} = \begin{bmatrix} \Psi(x_{t_1}, y_{t_1}) \\ \vdots \\ \Psi(x_{t_n}, y_{t_n}) \end{bmatrix} \tag{6}$$

The matrix A in equation (6) has some properties worth noting:

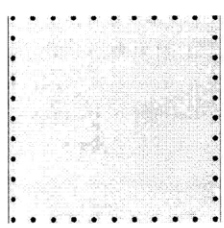
- A is dense, that is $A_{i,j}$ never equals zero. This is because every charge contributes to every potential.
- If the test points and the charge points are ordered so that the ith test point is nearest the ith charge, then $A_{i,i}$ will be larger than $A_{i,j}$ for all j.

The 2nd item above seems to suggest that A is diagonally dominant, but this is *not* the case. Diagonal dominance requires that the absolute *sum* of the off-diagonal entries is smaller than the magnitude of the diagonal. The matrix above easily violates that condition.

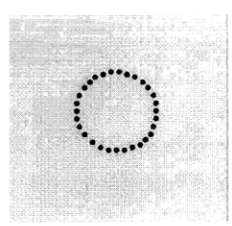
▷ ***Exercise 3*** Determine a set of test points and charge locations for the 2-D square problem that generates an A matrix where the magnitude of the diagonals are bigger than the absolute value of the off-diagonals, but the magnitude of the diagonal is smaller than the absolute sum of the off-diagonals. ■

3.4.3 Source Point Locations

Where should the sources be located?

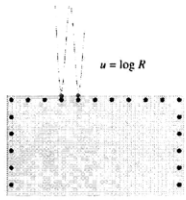


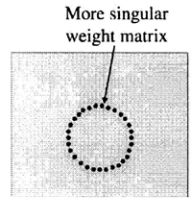
Close to the boundary



Clustered in the center

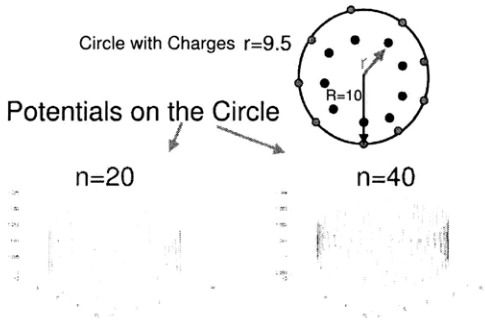
Problems with these placements:





Close to the boundary Clustered in the center

3.4.4 Computational Results



It is possible to construct a numerical scheme for solving exterior Laplace problems by adding progressively more point charges so as to match more boundary conditions. In the above graph, we show an example of using such a method to compute the potential exterior to a circle of radius 10, where the potential on the circle is given to be unity. In the example, charges are placed uniformly on a circle of radius 9.5, and test points are placed uniformly on the radius 10 circle. If 20 point charges are placed in a circle of radius 9.5, then the potential produced will be exactly one only at the 20 test points on the radius 10 circle. The potential produced by the twenty point charges on the radius 10 circle is plotted in the lower left corner of the slide above. As might be expected, the potential produced on the radius 10 circle is exactly one at the 20 test points, but then oscillates between 1 and 1.2 on the radius 10 circle. If 40 charges and test points are used, the situation improves. The potential on the circle still oscillates, as shown in the lower right hand corner, but now the amplitude is only between 1 and 1.004.

3.5 Charge Density

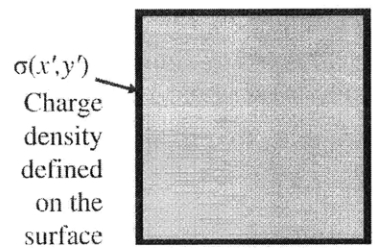
Want to smear point charges to the surface

Results in an integral equation

$$\Psi(\vec{x}) = \int_\Gamma G(\vec{x}, \vec{x}')\sigma(\vec{x}')dS' \tag{7}$$

How do we solve the integral equation?

In equation (7) for which variable are we trying to solve?



$$\Psi(x, y) = \int_\Gamma \sigma(x', y') \log \sqrt{(x - x')^2 + (y - y')^2} dx' dy'$$

Single Layer Potential

The oscillating potential produced by the point charge method is due to the rapid change in potential as the separation between evaluation point and point charge shrinks. If the point charges could be smeared out, so that the produced potential did not rise to infinity with decreasing separation, then the resulting computed potential would not have the oscillation noted on the previous slide. In addition, it makes the most sense to smear the point charges onto the surface, as then the charge density and the known potential have the same associated geometry. The result is the integral equation (7), where now the unknown is a charge *density* on the surface and the potential due to that charge density is given by the well-known superposition integral. In the case of two or three dimensional Laplace problems, $G(\vec{x}, \vec{x}')$ can be written as $\hat{G}(\vec{x} - \vec{x}')$, as the potential is only a function of distance to the charge density and not a function of absolute position. For such a Green's function, this equation is,

$$\Psi(\vec{x}) = \int_\Gamma \hat{G}(\vec{x} - \vec{x}')\sigma(\vec{x}')dS', \tag{8}$$

which one may recognize from system theory as a convolution integral. This connection is quite precise. A space-invariant system has an impulse response, which is usually referred to as a Green's function. The output, in this case, the potential, is a convolution of the impulse response with the input, in this case, the charge density. Such an integral form of the potential is referred to as a *single layer potential.*

The single layer potential is an example of a class of integral equations known as "Fredholm integral equation of the First Kind". A Fredholm integral equation of the Second Kind results when the unknown charge density exists not only under the integral sign but also outside it. An example of such an equation is

$$\Psi(\vec{x}) = \sigma(\vec{x}) + \int_{\Gamma} K(\vec{x} - \vec{x}')\sigma(\vec{x}')dS'. \tag{9}$$

Fredholm integral equations, in which the domain of integration is fixed, usually arise out of boundary value problems. Initial value problems typically give rise to the so-called Volterra integral equations, where the domain of integration depends on the output of interest. For example, consider the initial value problem

$$\frac{dx(t)}{dt} = tx(t); \quad t \in [0, T],\ T > 0.$$

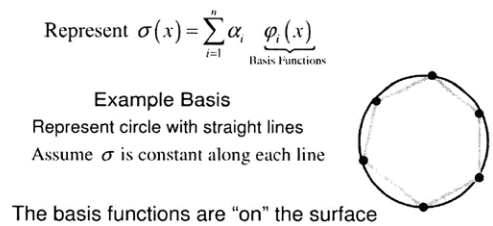
$$x(t = 0) = x_0$$

The "solution" of this equation is the following Volterra integral equation:

$$x(t) = x_0 + \int_0^t \xi x(\xi) d\xi.$$

4 Basis Functions

4.1 Basic Idea



Basis Functions can be used to approximate the surface charge density in a similar way in which they approximate geometry for finite elements.

Numerical solution of the single layer potential

As we have studied extensively in the finite-element section of the course, one approach to numerically computing solutions to partial differential equations is to represent the solution approximately as a weighted sum of basis functions. Then, the original problem is replaced with the problem of determining the basis function weights. In finite-element methods, the basis functions exist in a volume, for integral equations they typically exist on a surface. For 2-D problems that means the basis functions are restricted to curves and in 3-D the

basic functions are on physical surfaces.
As an example, consider the circle in the above figure. One could try to represent the charge density on the circle by breaking the circle into n sub-arcs, and then assume the charge density is a constant on each sub-arc. Such an approach is not commonly used. Instead the geometry is usually approximated along with the charge density. In this example case, shown in the center right of the slide, the sub-arcs of the circle are replaced with straight sections, thus forming a polygon. The charge density is assumed constant on each edge of the polygon. The result is a piecewise constant representation of the charge density on a polygon.

4.2 Geometric Approximation

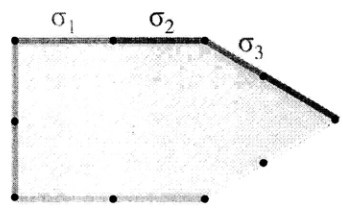
Piecewise Straight surface basis Functions approximate the circle

Triangles for 2-D FEM approximate the circle too!

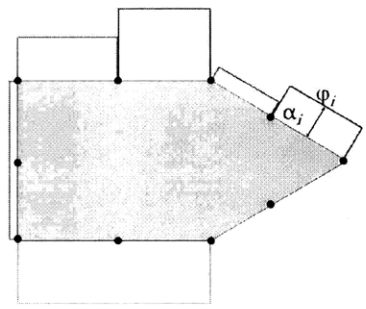
$$\Psi(x) = \int_{\text{approx surface}} G(x,x') \sum_{i=1}^{n} \alpha_i \varphi_i(x') \, dS'$$

The idea that both the geometry and the unknown charge density has been approximated is not actually a new issue. As shown in the figure in the above slide, if FEM methods are used to solve an interior problem, and triangular elements are used, then the circle is approximated to exactly the same degree as when straight sections replace the sub-arcs for the surface integral equation. As shown at the bottom of the above slide, we can substitute the basis function representation into the integral equation, but then we should also note that the integral is now over the approximated geometry. It is common, but not mathematically justified, to ignore the errors generated by the geometry approximation. We will also ignore the error in the geometric approximation in our analyses, just for simplicity. In the case of polygonal geometries, there is no geometric approximation, so there is at least one case where the assumption is precise. It should be noted, however, that there are often analytic results only for smooth geometries, and then before making comparisons to such analytic results, it is necessary to examine the effect of the approximated geometry.

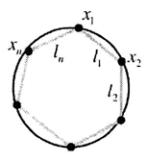
If the original problem is a polygon



there is no geometric approximation



4.2.1 Piecewise Constant Straight Sections



1) Pick a set of n Points on the surface
2) Define a new surface by connecting points with n lines.
3) Define $\varphi_i(x) = 1$ if x is on line l_i otherwise, $\varphi_i(x) = 0$

$$\Psi(\vec{x}) = \int_{\substack{\text{approx}\\ \text{surface}}} G(\vec{x}, \vec{x}') \sum_{i=1}^{n} \alpha_i \varphi_i(\vec{x}') dS' = \sum_{i=1}^{n} \alpha_i \int_{line\ l_i} G(\vec{x}, \vec{x}') dS'$$

How do we determine the α_i's?

We complete the description of using constant charge densities on straight sections as the basis. If we substitute this example basis function into the integral equation, as is done above, the result is to replace the original integration of the product of the Green's function and the density with a weighted sum of integrals over straight lines of just the Green's function. The next step is then to develop an approach for determining the weights, denoted here by α_i's.

5 Test Points

5.1 Residuals

5.1.1 Definition and Minimization

$$R(\vec{x}) = \Psi(\vec{x}) - \int_{\substack{\text{approx}\\ \text{surface}}} G(\vec{x}, \vec{x}') \sum_{i=1}^{n} \alpha_i \varphi_i(\vec{x}') dS' \tag{10}$$

Pick the α_i's to minimize $R(\vec{x})$
General Approach: Pick a set of test functions $\phi_1, \ldots, \phi_n$ and force $R(\vec{x})$ to be orthogonal to the set:

$$\int \phi_i(\vec{x}) R(\vec{x}) dS = 0 \quad \forall i \tag{11}$$

One way of assessing the accuracy of the basis function based approximation of the charge density is to examine how well the approximation satisfies the integral equation. To be more precise, we define the residual associated with the integral equation and an approximate solution, equation (10). Note that $R(\vec{x})$ is just the difference between the given potential on the surface and the potential produced by the approximated charge density. Note also that the equation is now over the approximate geometry and therefore $\vec{x}$ and $\vec{x}'$ are both on the approximated surface.
If the representation satisfies the integral equation exactly, then the residual $R(\vec{x})$ will be zero for all $\vec{x}$ and the approximate solution is equal to the exact solution (provided the integral equation has a unique exact solution ...more on this later). In general, though, this is not possible, and instead we will try to pick the basis function weights, the α_i's, to somehow minimize $R(\vec{x})$. One approach to minimizing $R(\vec{x})$ is to make it orthogonal to a collection of test functions, which may or may not be related to the basis functions, equation (11). Enforcing orthogonality in this case means ensuring that the integral of the product of $R(\vec{x})$ and $\phi(\vec{x})$ over the surface is zero.

5.1.2 Residual Minimization Using Test Functions

$\boxed{\int \phi_i(\vec{x}) R(\vec{x}) dS = 0} \Rightarrow$

$$\int \phi_i(\vec{x}) \Psi(\vec{x}) dS - \iint_{\substack{\text{approx}\\ \text{surface}}} \phi_i(\vec{x}) G(\vec{x}, \vec{x}') \sum_{j=1}^{n} \alpha_j \varphi_j(\vec{x}') dS' dS = 0 \tag{12}$$

We will generate different methods by choosing the $\phi_1, \ldots, \phi_n$
Collocation : $\phi_i(\vec{x}) = \delta(\vec{x} - \vec{x}_{t_i})$ (point matching)
Galerkin Method : $\phi_i(\vec{x}) = \varphi_i(\vec{x})$ (basis = test)
Weighted Residual Method : $\phi_i(\vec{x}) = 1$ if $\varphi_i(\vec{x}) \neq 0$ (averages)

As noted in the equation (12), by substituting the definition of the residual into the equation (11), it is possible to generate n equations, one for each test function. The generated equation has two integrals. The first is a surface integral of the product of the given potential with the test function. The second integral is a double integral over the surface. The integrand of the double integral is a product of the test function, the Green's function, and the charge density representation.
Three different numerical techniques can be derived by altering the test functions.

5.2 Collocation

Collocation: $\phi_i(\vec{x}) = \delta(\vec{x} - \vec{x}_{t_i})$ (point matching)

$\boxed{\int \delta(\vec{x}-\vec{x}_{t_i})R(\vec{x})dS = R(\vec{x}_{t_i}) = 0} \Rightarrow$

$$\sum_{j=1}^{n} \alpha_j \overbrace{\int_{\substack{\text{approx}\\ \text{surface}}} G(\vec{x}_{t_i}, \vec{x}')\varphi_j(\vec{x}')dS'}^{A_{i,j}} = \Psi(\vec{x}_{t_i}) \Rightarrow$$

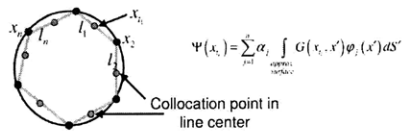
$$\begin{bmatrix} A_{1,1} & \cdots & A_{1,n} \\ \vdots & \ddots & \vdots \\ A_{n,1} & \cdots & A_{n,n} \end{bmatrix} \begin{bmatrix} \alpha_1 \\ \vdots \\ \alpha_n \end{bmatrix} = \begin{bmatrix} \Psi(\vec{x}_{t_1}) \\ \vdots \\ \Psi(\vec{x}_{t_n}) \end{bmatrix}$$

The collocation method, described in the above slide, uses shifted impulse functions as test functions, $\phi_i(\vec{x}) = \delta(\vec{x} - \vec{x}_i)$. Impulse functions, also called "delta" functions, have a *sifting* property when integrated with a smooth function $f(\vec{x})$,

$$\int f(\vec{x})\delta(\vec{x} - \vec{x}_i)dx = f(\vec{x}_i).$$

Impulse functions are also referred to as generalized functions, and they are specified only by their behavior when integrated with a smooth function. In the case of the impulse function, one can think of the function as being zero except for a very narrow interval around $\vec{x}_i$, and then being so large in that narrow interval that $\int \delta(\vec{x} - \vec{x}_i)dx = 1$.
As the summation equation in the middle of the above slide indicates, testing with impulse functions is equivalent to insisting that $R(\vec{x}_i) = 0$, or in words, that the potential produced by the approximated charge density should match the given potential at n test points. That the potentials match at the test points gives rise to the method's name, the point where the potential is exactly matched is "co-located" with a set of test points.
The $n \times n$ matrix equation at the bottom of the above slide has as its right-hand side the potentials at the test points. The unknowns are the basis function weights. The jth matrix element for the ith row is the potential produced at test point x_i by a charge density equal to basis function φ_j.

5.2.1 Centroid Collocation for Piecewise Constant Bases



$$\begin{bmatrix} A_{1,1} & \cdots & A_{1,n} \\ \vdots & \ddots & \vdots \\ A_{n,1} & \cdots & A_{n,n} \end{bmatrix} \begin{bmatrix} \alpha_1 \\ \vdots \\ \alpha_n \end{bmatrix} = \begin{bmatrix} \Psi(\vec{x}_{t_1}) \\ \vdots \\ \Psi(\vec{x}_{t_1}) \end{bmatrix} \Rightarrow \Psi(\vec{x}_{t_i}) = \sum_{j=1}^{n} \alpha_j \underbrace{\int_{line\,j} G(\vec{x}_{t_i}, \vec{x}')dS'}_{A_{i,j}}$$

In the above slide, a specific collocation algorithm is described. First, the basis being used is the constant charge density on n straight sections or lines, as described above. Note that therefore the geometry is being approximated. Second, the collocation points being selected are the centroids of the basis functions, in this case just the center of each straight line. Note that the collocation point is on the approximated geometry, not the original geometry. So, one can think of the problem as having been restated to be on a polygon instead of the original circle. One could also have selected the collocation points on the original circle, but then the replacement interpretation does not hold.

In collocation, or point-matching, the charge densities on each of the straight lines are selected so that the resulting potential at the line centers matches the given potential. As the equations on this slide make clear, the matrix element $A_{i,j}$ is the potential at the center of line i due to a unit charge density along line j.

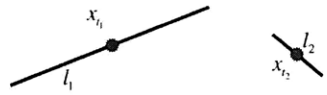
It should be noted that the matrix A is dense, the charge on line j contributes to the potential everywhere. Also note that if line j is far away from line i, then

$$A_{i,j} \approx \text{length}(line_j) \times G(\vec{x}_{t_i}, \vec{x}_{t_j}) \tag{13}$$

▷ ***Exercise 4*** Suppose we are using piecewise constant centroid collocation to solve a 2-D Laplace problem, so $G(x, y, x', y') = \log\sqrt{(x-x')^2 + (y-y')^2}$. Roughly how far apart do line sections i and j have to be for equation (13) to be accurate to within one percent? Assume line j has length of one. Does your answer depend on the orientation of line j? Does your answer depend on the orientation of line i? (You should answer yes to one of these and no to the other, do you see why?) ■

5.2.2 Centroid Collocation Generates Nonsymmetric A

$$\Psi(\vec{x}_{t_i}) = \sum_{j=1}^{n} \alpha_j \overbrace{\int_{\text{line } j} G(\vec{x}_{t_i}, \vec{x}')dS'}^{A_{i,j}}$$



$$A_{1,2} = \int_{\text{line 2}} G(\vec{x}_{t_1}, \vec{x}')dS' \neq \int_{\text{line 1}} G(\vec{x}_{t_2}, \vec{x}')dS' = A_{2,1} \tag{14}$$

Consider the two line sections, l_1 and l_2 given in the above figure. For Laplace problems, $G(\vec{x}, \vec{x}') = G(\vec{x}', \vec{x})$, which suggests a symmetry in the underlying integral equation that is not represented in the collocation discretization. This asymmetry is shown in equation (14) by noting that $A_{1,2} \neq A_{2,1}$. That is, the potential at the center of l_2 due to a unit charge density on l_1 is not equal to the potential at the center of l_1 due to a unit charge on l_2.
It is possible to scale the variables to improve the symmetry, consider a change of variables

$$\hat{\alpha}_i = \alpha_i \times \text{length}(line_j).$$

In this change of variables, the unknowns $\hat{\alpha}_i$ are now the net line charges rather than the line charge densities. In this new system, $\hat{A}\hat{\alpha} = \Psi$, where the elements of the matrix $\hat{A}$ are given by

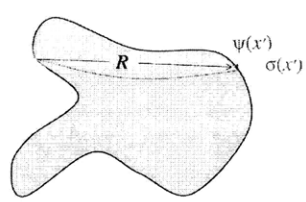
$$\hat{A}_{i,j} = \frac{1}{\text{length}(line_j)} \int_{line_j} G(\vec{x}_{t_i}, \vec{x}')dS'.$$

Under the change of variables, if line j is far away from line i, then

$$\hat{A}_{i,j} \approx G(\vec{x}_{t_i}, \vec{x}_{t_j}) \approx \hat{A}_{j,i}. \tag{15}$$

In other words, the elements of $\hat{A}$ corresponding to distant terms are approximately symmetric.

Is $\Psi(\vec{x})$ due to $\sigma(\vec{x}')$ the same as $\Psi(\vec{x}')$ due to $\sigma(\vec{x})$?



Green's Function is due to $\log R$

▷ ***Exercise 5*** Give an example which shows that the scaled entries of $\hat{A}$ can be far from symmetric. Assume we are using piecewise constant straight sections with centroid collocation and the 2-D Laplace's equation Green's function. ■

5.3 Galerkin

$$\underline{\text{Galerkin: } \phi_i(x) = \varphi_i(x) \text{ (test=basis)}}$$

$$\int \varphi_i(x)\, R(x)\, dS = \int \varphi_i(x)\, \Psi(x)\, dS - \int_{\substack{\text{approx}\\ \text{surface}}} \int_{\substack{\text{approx}\\ \text{surface}}} \varphi_i(x) G(x,x') \sum_{j=1}^{n} \alpha_j \varphi_j(x')\, dS'\, dS = 0$$

$$\underbrace{\int_{\substack{\text{approx}\\ \text{surface}}} \varphi_i(x) \Psi(x)\, dS}_{b_i} = \sum_{j=1}^{n} \alpha_j \underbrace{\int_{\substack{\text{approx}\\ \text{surface}}} \int_{\substack{\text{approx}\\ \text{surface}}} G(x,x') \varphi_i(x) \varphi_j(x')\, dS'\, dS}_{A_{i,j}}$$

$$\begin{bmatrix} A_{1,1} & \cdots & A_{1,n} \\ \vdots & \ddots & \vdots \\ A_{n,1} & \cdots & A_{n,n} \end{bmatrix} \begin{bmatrix} \alpha_1 \\ \vdots \\ \alpha_n \end{bmatrix} = \begin{bmatrix} b_1 \\ \vdots \\ b_n \end{bmatrix}$$

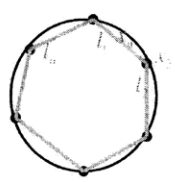
If $G(\vec{x}, \vec{x}') = G(\vec{x}', \vec{x})$ then $A_{i,j} = A_{j,i} \Rightarrow$ **A is symmetric**

In the Galerkin method, the test functions are equal to the basis functions. In particular, one generates n equations for the basis function weights by insisting that $R(\vec{x})$ is orthogonal to each of the basis functions. Enforcing orthogonality corresponds to setting

$$\int \varphi(\vec{x}) R(\vec{x}) dS = 0$$

and substituting the definition of $R(\vec{x})$ into the orthogonality condition yields the equation in the center of the above slide.
Note that the Galerkin method yields a system of n equations, one for each orthogonality condition, and n unknowns, one for each basis function weight. Also, the system does not have the potential explicitly as the right hand side. Instead, the ith right-hand side entry is the average of the product of the potential and the ith basis function.

5.3.1 Galerkin for Piecewise Constant Bases



$$\underbrace{\int_{line_i} \Psi(x)\, dS}_{b_i} = \sum_{j=1}^{n} \alpha_j \underbrace{\int_{line_i} \int_{line_j} G(x,x')\, dS'\, dS}_{A_{i,j}}$$

$$\begin{bmatrix} A_{1,1} & \cdots & A_{1,n} \\ \vdots & \ddots & \vdots \\ A_{n,1} & \cdots & A_{n,n} \end{bmatrix} \begin{bmatrix} \alpha_1 \\ \vdots \\ \alpha_n \end{bmatrix} = \begin{bmatrix} b_1 \\ \vdots \\ b_n \end{bmatrix}$$

In the Galerkin method, the basis has constant charge density on n straight sections or lines. We will think of the problem as having been restated to be on a polygon instead of the original circle. The charge densities on each of the straight lines are selected so that the resulting line averaged potential matches the line averaged given potential. As the equations on the above slide make clear, the matrix element $A_{i,j}$ is the average potential over line i, scaled by the length of line i, due to a unit charge density along line j.

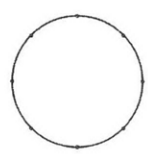
As with the collocation method, the matrix A is dense because the the charge on line j contributes to the averaged potentials everywhere. Also note that if line j is far away from line i, then

$$A_{i,j} \approx \text{length}(line_j) \times \text{length}(line_i) \times G(\vec{x}_{t_i}, \vec{x}_{t_j}) \tag{16}$$

▷ ***Exercise 6*** Suppose we are using piecewise constant centroid collocation to solve a 2-D Laplace problem, so $G(x, y, x', y') = \log \sqrt{(x - x')^2 + (y - y')^2}$. Roughly how far apart do line sections i and j have to be for equation (16) to be accurate to within one percent? Assume line j has length of one. Does your answer depend on the orientation of line j? Does your answer depend on the orientation of line i? (Your answer should be different than the answer you gave for the collocation method. Do you see why?) ■

5.4 Summary

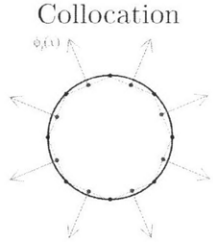
Compare the Collocation and Galerkin methods on a two-dimensional circle.



What do the test functions look like?
What do the Residuals look like?

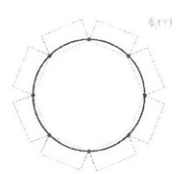
Test Functions

Collocation



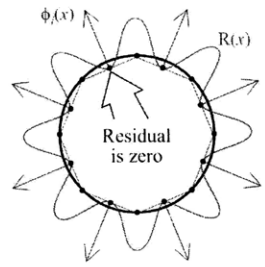
$\phi_i(\vec{x}) = \delta(\vec{x} - \vec{x}_i)$

Galerkin

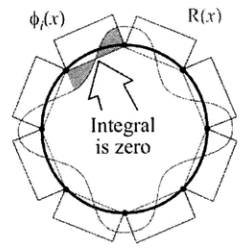


$\phi_i(\vec{x}) = \varphi_i(\vec{x})$

Collocation Method



Galerkin Method

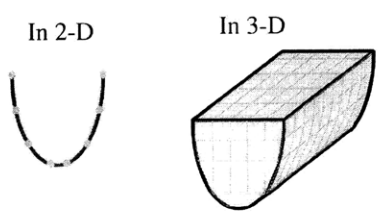


6 Issues in 3D

6.1 Geometric Representation

6.1.1 Introduction

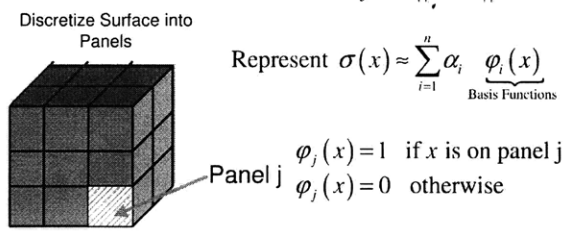
Example: Ship's Hull
More errors are introduced with expansion of dimensions



Note 6 ***"Leaky Panels"***

Many papers in the literature on solving integral equations refer to "panel methods". The name is derived from the idea of breaking a surface into flat panels. In the application area of analyzing ocean wave forces on ship hulls, panel methods are commonly used. However, it is not possible to represent a curved hull with quadrilateral flat panels. Researchers in the area often create a best fit panelled surface in which there are gaps between the edges of the panels. Such a discretization technique is often referred to as using "leaky panels", a very compelling image.

Integral Equation : $\Psi(x) = \int_{surface} \frac{1}{||x-x'||}\sigma(x')dS'$



Consider solving the integral equation where the surface is the surface of the cube shown. The first step, as we have mentioned in previous lectures, is to develop a basis in which to represent the surface charge density σ.
The cube pictured in the slide has had its surface divided into panels, and a basis is derived from the panels. In particular, one can associate a basis function φ_j with each panel j by assigning $\varphi_j(\vec{x})$ the value one when $\vec{x}$ is a point on panel j, and setting $\varphi_j(\vec{x}) = 0$ otherwise. If σ is approximated by a weighted combination of these basis functions, then the approximation is a piecewise constant representation of the charge density on the surface of the cube.
A few aspects of this basis set should be noted.

- The basis functions are orthogonal, that is if $i \neq j$,

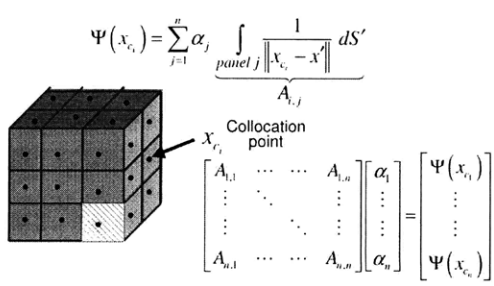
$$\int \varphi_j(\vec{x})\varphi_i(\vec{x})dx = 0.$$

- These basis functions are normalized with respect to l_∞, not l_2. That is, $\|\varphi\|_\infty = 1$ but

$$\|\varphi_j\|_2^2 = \int \varphi_i(\vec{x})\varphi_i(\vec{x})dx = \text{panel area}.$$

6.2 Centroid Collocation

Put collocation points at panel centroids



After one has decided on a basis with which to approximately represent the surface charge density, the next step is to develop a system of equations from which to determine the basis weights, denoted as the α_i's. The most commonly used approach to forming such a system is to use collocation, though Galerkin methods are also quite widely used. Recall that in collocation, the basis function weights are determined by ensuring the the integral equation is exactly satisfied at a collection of "collocation" points. For panel methods, the most common choice for the position of the collocation points are the panel centroids, as shown in the cube diagram above.

The equation in the top of the above slide relates the potential at collocation point $\vec{x}_{c_i}$ to the weights for the panel-based basis functions. To see how the equation was derived, consider evaluting the potential at the ith collocation point using the original integral equation

$$\Phi(\vec{x}_{c_i}) = \int_{surface} \frac{1}{\|\vec{x}_{c_i} - \vec{x}'\|} \sigma(\vec{x}') dS', \tag{17}$$

where Φ is the know potential on the problem surface and σ is the unknown charge density. Substituting the approximate representation for σ,

$$\sigma(\vec{x}) \approx \sum_{j=1}^{n} \alpha_j \varphi_j(\vec{x})$$

into the integral equation results in

$$\Phi(\vec{x}_{c_i}) = \int_{surface} G(\vec{x}_{c_i}, \vec{x}') \sum_{j=1}^{n} \alpha_j \varphi_j(\vec{x}') dS', \tag{18}$$

where $G(\vec{x}, \vec{x}') \equiv \frac{1}{\|\vec{x}-\vec{x}'\|}$ is used to simplify the formula. Exploiting the fact that $\varphi_j(\vec{x}) = 1$ if $\vec{x}$ is on panel j, and zero otherwise, results in the formula at the top of the above slide.

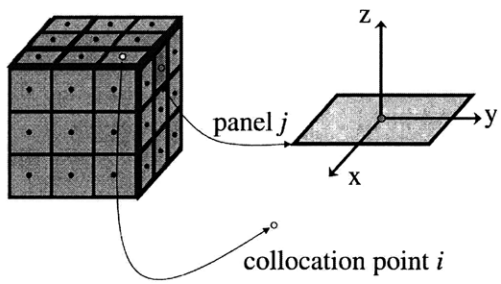
The system of equations from which to determine the basis function weights is given in the lower corner of the slide. The right hand side of the system is the vector of known potentials at the collocation points. The i, j^{th} element of the matrix A is the potential produced at collocation point i due to a unit charge density on panel j. The vector of α's are the unknown panel charge densities.

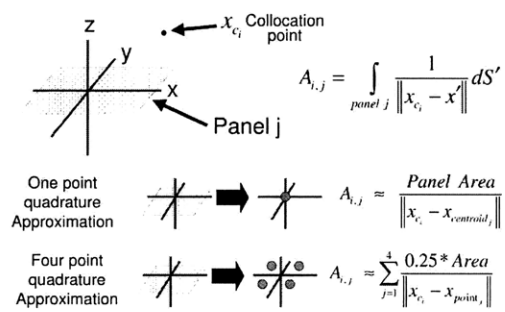
▷ ***Exercise 7*** Determine a scaling of the α's ($\hat{\alpha}_i = c_i \alpha_i$) such that the scaled matrix $\hat{A}$ has the property

$$A_{i,j} \approx \frac{1}{\|\vec{x}_{c_i} - \vec{x}_{c_j}\|}$$

when $\|\vec{x}_{c_i} - \vec{x}_{c_j}\|$ is much larger than a panel diameter. ■

6.2.1 Calculating Matrix Elements





In order to calculate the matrix entries for the system of equations described in the previous slide, recall that $A_{i,j}$ is the potential produced at collocation point i due to a unit charge density on panel j. The formula for $A_{i,j}$ is given on the top right of the above slide.

The figure on the left of the above slide is a diagram of how one typically computes the panel integral given on the top right. First, consider a shift and rotation of the coordinate system so that the panel lies in the x-y plane at $z = 0$, with the panel's center at $x = 0$, $y = 0$. The figure in the top left shows the panel in the shifted and rotated coordinate system. Note that the collocation point must also be placed in the new coordinate system.

If panel j is reasonably well separated from collocation point i, it is possible to approximate the integral given in the top right by a single point quadrature. More specifically, one could approximates the integral of $\frac{1}{\|\vec{x}_{c_i} - \vec{x}'\|}$ by a product of $\frac{1}{\|\vec{x}_{c_i} - \vec{x}_{centroid_j}\|}$ and the panel area. As show in the middle figure, a single point quadrature is like treating the panel as if it were a point charge at the panel's centroid, where the point charge's strength is equal to the panel area.

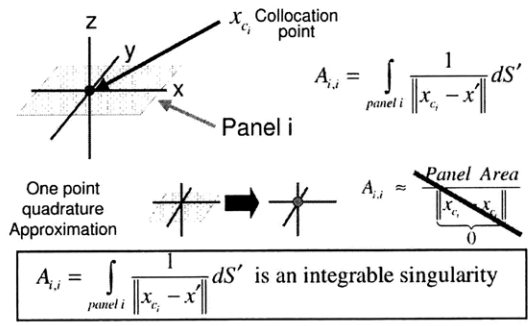
If the collocation point is close to the panel, then a single point quadrature will be insufficiently accurate. Instead, a more accurate four point quadrature scheme would be to break the panel into four subpanels, and then treat each of the subpanels as point charges at their respective centers. This simple idea is shown in the figure at the bottom of the above slide. This four point scheme is

equivalent to

$$\int_{panel_j} \frac{1}{\|\vec{x}_{c_i} - \vec{x}'\|} dS' \approx \sum_{j=1}^{4} \frac{0.25 * Area}{\|\vec{x}_{c_i} - \vec{x}_{point_j}\|}.$$

If the panel is a unit square in the x-y plane whose center is at the coordinate system origin, then the four $\vec{x}_{point_j}$'s are $(x, y, z) = (0.25, 0.25, 0)$, $(x, y, z) = (-0.25, 0.25, 0)$, $(x, y, z) = (-0.25, -0.25, 0)$, and $(x, y, z) = (0.25, -0.25, 0)$.

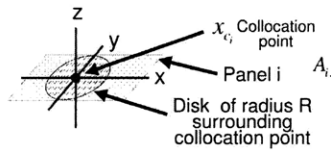
6.2.2 Calculating "Self Term"



The diagonal terms $A_{i,i}$ can not be computed using the quadrature approximation given on the previous slide. To see this, consider the figure at the top left of the above slide, where a panel has been shifted and rotated into the x-y plane, and the collocation point is the center of the panel. The integral that must be computed is given on the right side of the top of the above slide.
As shown in the middle of the slide, using a single point quadrature scheme will fail, because the distance between the point charge approximation to the panel and the collocation point will be zero. Therefore, the single point formula will require computing the reciprocal of zero, which is infinite. The problem is that the integrand in

$$\int_{panel_i} \frac{1}{\|x_{c_i} - x'\|} dS' \tag{19}$$

is singular. That is, the integrand approaches infinity at a point x' which is in the domain of integration. What is not so obvious is that (19) is an integrable singularity. Therefore, even though the integrand approaches infinity at some point, the "area under the curve" is finite.



$$A_{i,i} = \int_{panel\ i} \frac{1}{\|x_{c_i} - x'\|} dS'$$

Integrate in two pieces

$$A_{i,i} = \int_{disk} \frac{1}{\|x_{c_i} - x'\|} dS' + \int_{rest\ of\ panel} \frac{1}{\|x_{c_i} - x'\|} dS'$$

Disk Integral has singularity but has analytic formula

$$\int_{disk} \frac{1}{\|x_{c_i} - x'\|} dS' = \int_0^R \int_0^{2\pi} \frac{1}{r} r dr d\theta = \boxed{2\pi R}$$

In the above slide, we both show that

$$\int_{panel_i} \frac{1}{\|\vec{x}_{c_i} - \vec{x}'\|} dS'$$

is integrable, and also give an idea about how to compute the integral.
As shown in the slide, first rotate and shift the coordinate system so that the panel is in the x-y plane at $z = 0$, and so that the collocation point (or equivalently the panel centroid) is at the origin. In this new coordinate system, the integral can be written as

$$A_{i,i} = \int_{panel(rs)_i} \frac{1}{\|\vec{x}'\|} dS'$$

where the notation $panel(rs)$ is used to indicate that the integral is over the rotated and shifted panel.
On the top left of the above slide, a circular disk of radius R and center at the collocation point is inscribed in the rotated panel. In the equations that follow the figure, it is noted that the panel integral can be recast as the sum of an integral over the disk plus an integral over the rest of the panel. The integrand in the integral over the rest of the panel is no longer singular, but the integrand in the integral over the disk is still singular.
The integral over the disk can be computed analytically by using a change of variables. After rotating and shifting the panel, the disk is in the x-y plane and its center, equal to the collocation point, is at zero. Therefore,

$$\int_{disk} \frac{1}{\|\vec{x}_{c_i} - \vec{x}'\|} dS' = \int_{disk} \frac{1}{\|\vec{x}'\|} dS'$$

Apply a change of variables as transformations of two-dimensional regions. Recall this mapping from an earlier lecture in the finite difference method. Suppose that a region $\hat{\Omega}$ in the r-θ plane is transformed one-to-one into the region Ω by differentiable equations of the form

$$x = rcos\theta, \qquad y = rsin\theta.$$

Any function $f(x, y)$ defined on Ω can be thought of as a function $f(x(r, \theta), y(r, \theta))$ on $\hat{\Omega}$. The integrals of these functions are related by

$$\iint_{\Omega} f(x, y) dx dy = \iint_{\hat{\Omega}} f(x(r, \theta), y(r, \theta)) \, |J(r, \theta)| \, dr d\theta$$

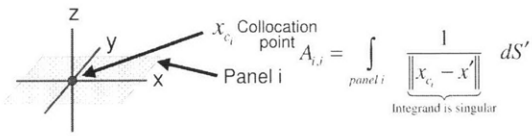
where $J(r,\theta)$ is the Jacobian determinant of the coordinate transformation where

$$J(r,\theta) = \begin{vmatrix} \frac{\partial x}{\partial r} & \frac{\partial x}{\partial \theta} \\ \frac{\partial y}{\partial r} & \frac{\partial y}{\partial \theta} \end{vmatrix} = \begin{vmatrix} cos\theta & -rsin\theta \\ sin\theta & rcos\theta \end{vmatrix} = r(cos^2\theta + sin^2\theta) = r$$

So, now the integral may be put into the transformed coordinates using this transformation

$$\int_{disk} \frac{1}{\|\vec{x}_{c_i} - \vec{x}'\|} dS' = \int_0^R \int_0^{2\pi} \frac{1}{r} r d\theta dr$$

The integral over the disk is easily seen to be $2\pi R$.



1. If panel is a flat polygon, analytical formulas exist.

2. Curved panels can be handled with projection.

7 Summary

Integral Equation Methods
Exterior versus interior problems
Start with using point sources
Standard Solution Methods
Collocation Method
Galerkin Method
Integrals for 3D Problems
Singular Integrals

1.2 Numerical Quadrature

Numerical Methods for PDEs

Methods, Lecture 2
Numerical Quadrature

Notes by L. Proctor, S. De and J. White

November 26, 2008

1 Outline

Gaussian Quadrature

Convergence properties
Essential role of orthogonal polynomials
Multidimensional Integrals

Techniques for singular kernels

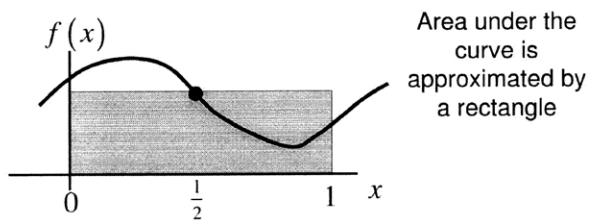
Adaptation and variable transformation
Singular quadrature.

2 Introduction

Numerical Quadrature is employed as an approximation used to evaluate integrals. We seek an appropriate numerical procedure applied to a definite integral, $I\{f\} \equiv \int_a^b f(x)dx$, where the approximation is essentially of the form $I_n\{f\} \equiv \sum_{i=1}^{n} \alpha_i f(x_i)$. The n distinct points, x_i are the quadrature nodes we have chosen and the quadrature coefficients, or weights, are the α_j terms. In general,we would like to have the smallest possible quadrature error, $E_n\{f\} \equiv I\{f\} - I_n\{f\}$.

2.1 Simple Quadrature Example

$$\int_0^1 f(x)dx \simeq f\left(\frac{1}{2}\right)$$

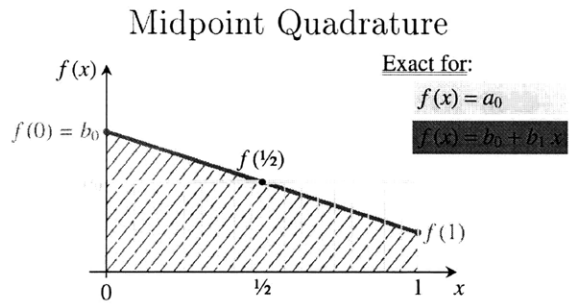


To simplify notation, consider the more generic problem of developing a good numerical technique for evaluating the integral of a function $f(x)$ on the domain $[0, 1]$. We assume that the integrand is a "smooth" function, though we will examine this assumption later. First we have developed a naive approach for obtaining a good approximation of the integral, one we call a **simple quadrature scheme**.

The simplest approach is to replace the integral with the the product of the interval (in this case one) and the integrand evaluated at a point inside the interval. If the selected evaluation point is the center of the interval, $x = 0.5$, we call the scheme **midpoint quadrature**.

A midpoint quadrature scheme replaces the area under the curve $f(x)$ by a

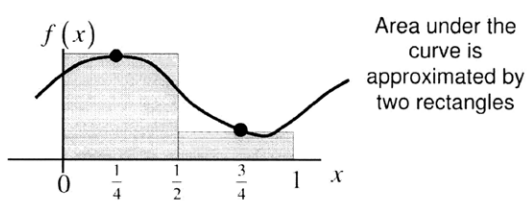
rectangle whose height is the function $f(x)$ evaluated at the midpoint $x = 0.5$. The scheme is exact when $f(x)$ is a constant. However, what is less obvious is that the scheme is exact when $f(x)$ is a line (an affine funcion of x) as well. The most obvious way of seeing this is by realizing that when $f(x)$ is a straight line, the area under it is a trapezoid. This trapezoid has exactly the same area as the rectangle which this scheme uses to approximate the integral (can you see why?).



▷ ***Exercise 1*** Suppose endpoint quadrature (in which the area under the curve is replaced by a rectangle whose height is $f(x)$ evaluated at $x = 0$ or $x = 1$) is used instead of midpoint quadrature. For what class of functions is endpoint quadrature exact? ■

2.2 Improving the Accuracy

$$\int_0^1 f(x)dx \simeq \frac{1}{2}f\left(\frac{1}{4}\right) + \frac{1}{2}f\left(\frac{3}{4}\right)$$



One way of improving the midpoint quadrature scheme is to divide the interval $[0, 1]$ into subintervals $[0, 0.5]$ and $[0.5, 1]$, then write the integral

$$\int_0^1 f(x)dx = \int_0^{0.5} f(x)dx + \int_{0.5}^1 f(x)dx,$$

and finally apply a midpoint rule to the integral in each subinterval. We obtain the scheme shown in the slide. The factor $\frac{1}{2}$ appearing in front of $f(\frac{1}{2})$ and $f(\frac{3}{4})$ are just the domain lengths.

▷ ***Exercise 1*** Can you come up with an expression for the error in this case? How much does the accuracy improve? ■

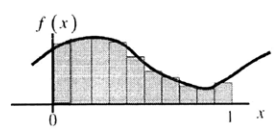
Dividing the interval into two reduces the error, now consider using n subintervals and repeating the midpoint quadrature rule on each subinterval. We obtain the scheme

$$\int_0^1 f(x)dx = \sum_{i=1}^{n} \underbrace{\frac{1}{n}}_{\substack{\text{subinterval}\\ \text{length}}} f(x_{c_i})$$

where the centroid of the ith subinterval is $x_{c_i} = \frac{1}{2}(\frac{i-1}{n} + \frac{i}{n}) = \frac{i-\frac{1}{2}}{n}$. There is no doubt that the accuracy improves, but the key question is by how much? How does this error decrease with the number of subintervals used? And finally, are there clever ways of obtaining better accuracy with less effort?

2.3 General n-point formula

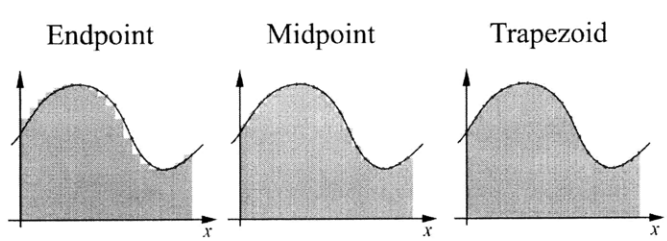
$$\int_0^1 f(x)dx \simeq \sum_{i=1}^{n} \underbrace{w_i}_{\substack{\text{weight}\\ \frac{1}{n}}} \underbrace{f(x_i)}_{\substack{\text{test point}\\ \frac{i-\frac{1}{2}}{n}}}$$



Key questions about the method:
How fast do the errors decay with n?
Are there better methods?

2.4 Different Geometric Approximations

Which geometry is the most accurate?



3 General Quadrature Formula

3.1 General 1-D Formula

$$\int_0^1 f(x)dx \simeq \sum_{i=1}^{n} \underbrace{w_i}_{\text{Weight}} \underbrace{f(x_i)}_{\text{Evaluation Point}}$$

Free to pick the **evaluation points**.
Free to pick the **weights** for each point.

An n-point formula has $2n$ degrees of freedom!

After all the hard work we did dividing the domain into subintervals, we realize that we cannot even integrate a quadratic function exactly on the domain. There must be something that we can do to improve this scheme. We go back and look at the general form of the quadrature approximation scheme. All we are doing is approximating an integral by a weighted sum of function evaluations. So far we have been choosing these weights as the subinterval lengths. We have also been choosing the evaluation points as the center of the interval, in the midpoint quadrature scheme. The weights are just some normalizing factors which we have taken to be the fraction of the domain over which we are evaluating. The equality of areas of trapezoids and rectangles that we previously discussed gives us the extra polynomial accuracy of being able to obtain the area under a straight line exactly. So, what would happen if we were to choose both the integration points and the weights intelligently? For an n-point formula we have n weights and n evaluation points to choose. That gives us $2n$ degrees of freedom. Hence we must be able to exactly integrate a polynomial of degree at most $(2n-1)$. This idea gives rise to the **Gaussian quadrature** scheme.

3.2 Evaluation Points & Weights Selection

Can make the result exact if $f(x)$ is a polynomial

$$f(x) = a_0 + a_1 x + a_2 x^2 + \cdots + a_l x^l = p_l(x)$$

Select x_i's and w_i's such that

$$\int_0^1 p_l(x)dx = \sum_{i=1}^{n} w_i p_l(x_i)$$

for ANY polynomial up to (and including) l^{th} order
With $2n$ degrees of freedom, $l = 2n - 1$

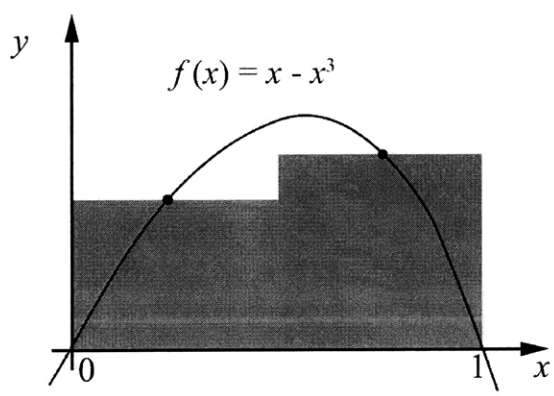
Let $p_l(x)$ denote a polynomial of degree l in the variable x ($a_l \neq 0$). We want to select the weights and integration points such that the formula

$$\int_0^1 f(x)dx = \int_0^1 p_l(x)dx = \sum_{i=1}^{n} w_i p_l(x_i)$$

is exact for all polynomials of degree up to (and including) l. Obviously, with $2n$ degrees of freedom, the best we can do is $l = 2n - 1$.

Note 1 **_Example: Third Order Polynomial_**

As an example, consider integratin the function $f(x) = x - x^3$. from $x = 0$ to $x = 1$. The exact solution is, $I\{f\} = \int_0^1 (x - x^3)dx = \frac{1}{4}$. It is stated above that since we have a polynomial of degree three ($l = 3$), then we will be able to find an exact solution using two point quadrature ($n = 2$). But, one must note, that finding the exact solution is not simply a matter of applying midpoint quadrature, as we have done previously. The solution using midpoint quadrature is 0.2813, not very close to 0.25. Using Gaussian Quadrature, the method of which will be studied in further detail later on, will provide the exact solution.



Assuming the weights, w_i, remain bounded, and the derivatives of $f(x)$ are bounded on $[0, 1]$,

$$\left| \int_0^1 f(x)dx - \sum_{i=1}^{n} w_i f(x_i) \right| \leq \frac{K}{(2n)!}$$

Gaussian quadrature converges **very** quickly!!

4 Error Analysis

4.1 Taylor Series Expansion

To derive the error of the midpoint quadrature scheme analytically, consider the interval $[0, h], h > 0$ and Taylor expand $f(x)$ about the center of this interval, $\bar{x} = \frac{h}{2}$,

$$f(x) = f(\bar{x}) + \Delta(x)\frac{df(\bar{x})}{dx} + \frac{\Delta(x)^2}{2!}\frac{d^2 f(\xi)}{dx^2} \quad for\ some\ \xi \in [0, h],$$

where $\Delta(x) = x - \bar{x}$. The last term in the expansion is the Taylor series remainder. Integrating this expansion over the interval $[0, h]$

$$\int_0^h f(x)dx = hf(\bar{x}) + \frac{h^3}{24}\frac{d^2 f(\xi)}{dx^2}.$$

Hence the error in the midpoint quadrature approximation is

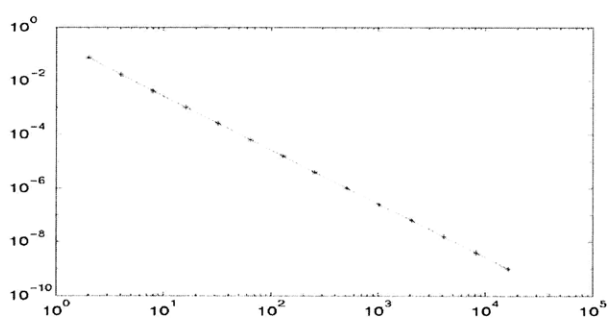
$$E = \int_0^h f(x)dx - hf(\bar{x}) = \frac{h^3}{24}\frac{d^2 f(\xi)}{dx^2}.$$

A function that is first-order polynomial in x would have zero as the second derivative, and therefore the above expression tells us that the error of midpoint quadrature for such functions is identically zero. In addition, the above expression tells us that the error scales as the cube of the domain length.

▷ ***Exercise 2*** If the midpoint quadrature scheme uses rectangular geometry to approximate the area under the curve, then why is the first derivative needed in the Taylor Series expansion? How does one represent the expansion of the trapezoidal approximation? ■

4.2 Example - Error vs. n

$$\int_0^1 sin(x)dx \simeq \sum_{i=1}^{n} \frac{1}{n} sin\left(\frac{i - \frac{1}{2}}{n}\right)$$



Note 2

Above is the example of integrating $f(x) = sin(x)$ on the domain $[0, 1]$. We obtain progressively better answers to the integration by increasing the number of subintervals n. The error in evaluating the integral is plotted as a function of the number of subintervals (n) above. The error appears to be going down as $\mathcal{O}\left(\frac{1}{n^2}\right)$.
From what we have just seen, the error inside the ith subinterval (of length $h = \frac{1}{n}$) is $\frac{h^3}{24}\frac{d^2 f(\xi_i)}{dx^2}$ for some $\xi_i \in [\frac{i-1}{n}, \frac{i}{n}]$. Hence, for the entire interval $[0, 1]$ we can sum these errors and obtain the error, E_n for an approximation using n subintervals as

$$E_n = \frac{nh^3}{24}\underbrace{\left(\frac{1}{n}\sum_{i=1}^{n}\frac{d^2 f(\xi_i)}{dx^2}\right)}_{M}$$

It is easy to see that if $f(x)$ is a continuous function, M (being the mean) must be bounded by the maximum and the minimum of $f(x)$ on the interval $[0, 1]$ and hence, there must exist some $\xi \in [0, 1]$ such that $M = d^2 f(\xi)/dx^2$. Hence we obtain the estimate

$$E_n = \frac{nh^3}{24}\frac{d^2 f(\xi)}{dx^2} = \frac{1}{24n^2}\frac{d^2 f(\xi)}{dx^2}$$

since $h = 1/n$. This error estimate tells us that the scheme is again exact for constants and linear functions on the domain (no higher order polynomials!) and, for a smooth function, the error decays **algebraically.**

4.3 The Exactness Criteria

Consider the Taylor series for $f(x)$ expanded about $x = 0$

$$f(x) = f(0) + \frac{\partial f(0)}{\partial x}x + \cdots + \frac{1}{l!}\frac{\partial^l f(0)}{\partial x^l}x^l + R_{l+1}$$

R_{l+1} is the **remainder**

$$R_{l+1} = \frac{1}{(l+1)!}\frac{\partial^{l+1} f(\tilde{x})}{\partial x^{l+1}}x^{l+1}$$

where $\tilde{x} \in [0, x]$

Note 3

Of all functions, why are we interested in integrating polynomials? The reason comes from the structure of Taylor's series expansion. The Taylor expansion of a function in a local neighborhood of a point (here this point is chosen as 0 without loss of generality) is nothing but a power series expansion. The higher

the order of polynomials that our scheme can integrate means a higher order of the remainder term in the expansion. The integral of the remainder over the domain is precisely the error in numerical integration.

The exactness condition requires

$$\int_0^1 p_l(x)dx = \int_0^1 (a_0 + a_1 x + a_2 x^2 + \cdots + a_l x^l)dx = \sum_{i=1}^{n} w_i p_l(x_i)$$

for any set of $l+1$ coefficients $a_0, a_1, \ldots, a_l$

Equivalently

$$\int_0^1 a_0 dx + \int_0^1 a_1 x dx + \int_0^1 a_2 x^2 dx + \cdots + \int_0^1 a_l x^l dx = \sum_{i=1}^{n} w_i p_l(x_i)$$

This slide needs little clarification. Our exactness criterion is

$$\int_0^1 p_l(x)dx = \int_0^1 (a_0 + a_1 x + a_2 x^2 + \cdots + a_l x^l)dx = \sum_{i=1}^{n} w_i p_l(x_i)$$

which is the same as

$$a_0 \int_0^1 dx + a_1 \int_0^1 x dx + \cdots + a_l \int_0^1 x^l dx = a_0 \sum_{i=1}^{n} w_i + a_1 \sum_{i=1}^{n} w_i x_i + \cdots + a_l \sum_{i=1}^{n} w_i {x_i}^l$$

For this to be an identity for the $(l+1)$ arbitrary coefficients a_i, we must have the $(l+1)$ conditions

$$\sum_{i=1}^{n} w_i {x_i}^j = \int_0^1 x^j dx \qquad for\ j = 0, 1, \ldots, l$$

Using the Taylor series results, the exactness criteria, and the innate linearity of the quadrature scheme

$$\int_0^1 f(x)dx - \sum_{i=1}^{n} w_i f(x_i) =$$

$$\underbrace{\frac{1}{(l+1)!} \frac{\partial^{l+1} f(\tilde{x})}{\partial x^{l+1}} \left(\int_0^1 x^{l+1} dx - \sum_{i=1}^{n} w_i x_i^{l+1} \right)}_{Remainder}$$

Exactness condition will be satisfied if and only if

$$\int_0^1 dx = \sum_{i=1}^{n} w_i \cdot 1$$

$$\int_0^1 x dx = \sum_{i=1}^{n} w_i \cdot x_i$$

$$\vdots$$

$$\int_0^1 x^l dx = \sum_{i=1}^{n} w_i \cdot {x_i}^l$$

Reorganizing exactness equations

$$\begin{bmatrix} 1 & 1 & \cdots & 1 \\ x_1 & x_2 & \cdots & x_n \\ \vdots & \vdots & \ddots & \vdots \\ x_1^l & x_2^l & \cdots & x_n^l \end{bmatrix} \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_n \end{bmatrix} - \begin{bmatrix} 1 \\ \vdots \\ \vdots \\ \int_0^1 x^l dx \end{bmatrix} = 0$$

Nonlinear, since x_i's and w_i's are unknowns

What is a practical way of computing the evaluation points and weights? The system of equations is not easy to solve since x_i's and w_i's are unknowns.

5 Computing the Points & Weights

5.1 Newton's Method

Could use **Newton's Method**

$$F(y) = 0 \Rightarrow J_F\left(y^k\right)\left(y^{k+1} - y^k\right) = -F\left(y^k\right)$$

The nonlinear function for Newton is then

$$F\begin{pmatrix} w \\ x \end{pmatrix} = \begin{bmatrix} 1 & 1 & \cdots & 1 \\ x_1 & x_2 & \cdots & x_n \\ \vdots & \vdots & \ddots & \vdots \\ x_1^l & x_2^l & \cdots & x_n^l \end{bmatrix} \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_n \end{bmatrix} - \begin{bmatrix} 1 \\ \vdots \\ \vdots \\ \int_0^1 x^l dx \end{bmatrix} = 0$$

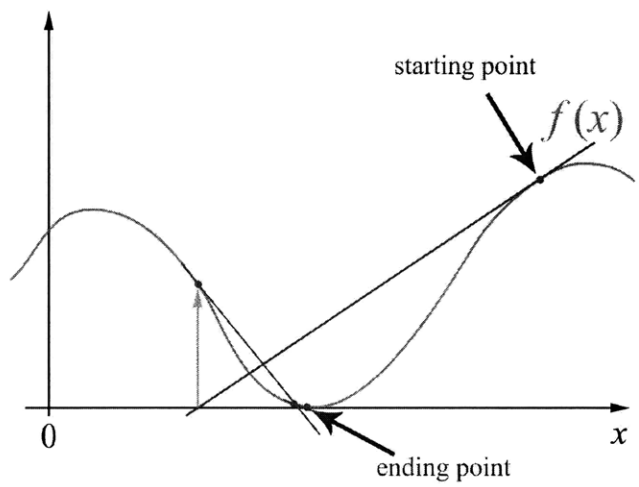
Note 4

Newton's method is an iterative technique for finding a value y such that $F(y) = 0$. The method is based on linearizing the problem about a guess at y, and then

updating the value of y by solving the linearized problem. In particular, the iterate y^{k+1} is determined from y^k by solving the linear system of equations

$$F(y^k) + J_F(y^k)\left(y^{k+1} - y^k\right) = 0$$

where $J_F(y^k)$ is the Jacobian (multidimensional derivative) of the nonlinear function $F(y)$. The iteration is continued until the updated y is sufficiently close to the exact solution, a criterion that can be difficult to verify. Newton's method does not always converge, a phenomenon that is more likely when $J_F(y)$ is nearly singular. For more about Newton's method, see the 6.336/16.910/2.096 course notes (available under open courseware).



Newton Method Jacobian reveals the problem

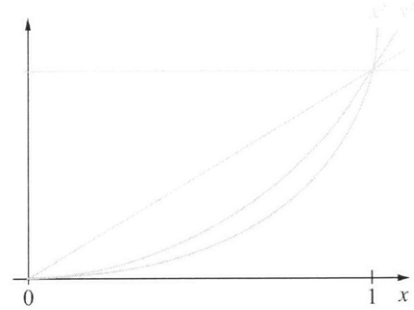
$$J_F\begin{pmatrix} w \\ x \end{pmatrix} = \overbrace{\begin{bmatrix} 1 & 1 & \cdots & 1 & 0 & 0 & \cdots & 0 \\ x_1 & x_2 & \cdots & x_n & w_1 & w_2 & \cdots & w_n \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ x_1^l & x_2^l & \cdots & x_n^l & l w_1 x_1^{l-1} & \cdots & \cdots & l w_1 x_n^{l-1} \end{bmatrix}}^{2n}$$

Columns become linearly dependent for high order

Looking at the Jacobian of the problem, we realize that the first n columns become increasingly linearly dependent for large l. This is bound to happen since we are looking into the space $span\{1, x, \ldots, x^l\}$. This basis always becomes ill-conditioned with increasing l. The solution is to obtain a polynomial basis that is "normalized" in some sense so that it is properly conditioned.

5.2 Orthogonal Polynomials

5.2.1 Introduction



Exactness criteria will be satisfied if and only if

$$\left.\begin{array}{l} \int_0^1 c_0(x)\,dx = \sum_{i=1}^{n} w_i c_0(x_i) \\ \int_0^1 c_1(x)\,dx = \sum_{i=1}^{n} w_i c_1(x_i) \\ \vdots \\ \int_0^1 c_i(x)\,dx = \sum_{i=1}^{n} w_i c_i(x_i) \end{array}\right\} \begin{array}{c} \text{BUT} \\ \text{Each } c_i \text{ polynomial must} \\ \text{Contain an } x^i \text{ term} \\ \text{Be linearly independent} \end{array}$$

The only difference from the previous set of criteria is that these polynomials have better properties than the ones we chose before.

5.2.2 Orthogonality

For the normalized integral, two polynomials are said to be **orthogonal** if

$$\int_0^1 c_i(x)c_j(x)dx = 0 \qquad for\ j \neq i$$

The above integral is often referred to as an inner product and ascribed the notation

$$(c_i, c_j) = \int_0^1 c_i(x)c_j(x)dx$$

The connection between polynomial inner products and vector inner products can be seen by sampling.

5.2.3 Exactness Criteria

Consider rewriting the exactness criteria

$$\underbrace{\begin{array}{c}\int_0^1 c_0(x)\,dx = \sum_{i=1}^{n} w_i c_0(x_i) \\ \vdots \\ \int_0^1 c_{n-1}(x)\,dx = \sum_{i=1}^{n} w_i c_{n-1}(x_i)\end{array}}_{\text{Low order terms}} \quad \underbrace{\begin{array}{c}\int_0^1 c_n(x)\,dx = \sum_{i=1}^{n} w_i c_n(x_i) \\ \vdots \\ \int_0^1 c_{2n-1}(x)\,dx = \sum_{i=1}^{n} w_i c_{2n-1}(x_i)\end{array}}_{\text{High Order Terms}}$$

Recall that $l = 2n - 1$

where l = degree of polynomial & n = number of coefficients

Call the first $(n-1)$ conditions the "lower order terms" and the last n conditions the "higher order terms."

5.2.4 Higher Order Terms Contain Lower Order Terms

Write the higher order terms differently

$$\int_0^1 c_n(x)dx = \sum_{i=1}^{n} w_i c_n(x_i) \quad \Rightarrow \quad \int_0^1 c_n(x)c_0(x)dx = \sum_{i=1}^{n} w_i c_n(x_i)c_0(x_i)$$

$$\vdots$$

$$\int_0^1 c_{2n-1}(x)dx = \sum_{i=1}^{n} w_i c_{2n-1}(x_i) \quad \Rightarrow \quad \int_0^1 c_n(x)c_{n-1}(x)dx = \sum_{i=1}^{n} w_i c_n(x_i)c_{n-1}(x_i)$$

The products $c_n(x)c_j(x)$ are linearly independent.

In this slide we express the "higher order terms" as conditions involving "lower order terms."

5.2.5 Using Orthogonality and Roots

Use orthogonal polynomials

$$\cancel{\int_0^1 c_n(x)c_0(x)\,dx} = \sum_{i=1}^{n} w_i \cancel{c_n(x_i)}\, c_0(x_i)$$

$$\vdots$$

$$\cancel{\int_0^1 c_n(x)c_{n-1}(x)\,dx} = \sum_{i=1}^{n} w_i c_n(x_i)\, \cancel{c_{n-1}(x_i)}$$

Pick the x_i's to be n roots of $c_n(x)$
The higher order constraints are exactly satisfied!

This elegant step relies on polynomial orthogonality. If we choose the polynomial $c_n(x)$ such that it is orthogonal to all polynomials of inferior degree (i.e. $c_0(x), c_1(x), \ldots, c_{n-1}(x)$) and the x_i's are roots of this polynomial, then the higher order n conditions are automatically satisfied. Note that for this derivation we used polynomials which are orthogonal on the interval $[0, 1]$. Such polynomials are shifted and scaled versions of the classical Legendre polynomials, which are orthogonal on the interval $[-1, 1]$.

5.2.6 Satisfying Lower Order Constraints

An abbreviated exactness equation

$$\begin{bmatrix} c_0(x_1) & \cdots & c_0(x_n) \\ \vdots & \ddots & \vdots \\ c_{n-1}(x_1) & \cdots & c_{n-1}(x_n) \end{bmatrix} \begin{bmatrix} w_1 \\ \vdots \\ w_n \end{bmatrix} = \begin{bmatrix} \int_0^1 c_0(x)dx \\ \vdots \\ \int_0^1 c_{n-1}(x)dx \end{bmatrix}$$

Now **linear**, x_i's are known
Rows are sampled orthogonal polynomials.

By using the roots of $c_n(x)$ for the x_i's, the higher order constraints are automatically satisfied. Since the x_i's are now known, only the weights are still unknown. The lower n constraints can be used to determine the weights, generating a linear system.

6 Gaussian Quadrature Algorithm

1. Construct $n+1$ orthogonal polynomials

$$\int_0^1 c_i(x)c_j(x)dx = 0 \quad for\ j \neq i$$

2. Compute n **roots**, $x_i,\ i = 1, \ldots, n$ of the nth order orthogonal polynomial such that $c_n(x_i) = 0$

3. Solve a linear system for the **weights** w_i

4. Approximate the integral as a sum

$$\int_0^1 f(x)dx = \sum_{i=1}^{n} w_i f(x_i)$$

6.1 Example

Note 5 ***Example: Third Order Polynomial***

Recall the example of the third-order polynomial, $f(x) = x - x^3$. We would like to determine w_1, w_2, x_1, and x_2 so that the integration formula,

$$I_2\{f\} = w_1 f(x_1) + c_2 f(x_2)$$

gives the exact result.
Generalize this problem to any third-order polynomial of the form

$$f(x) = a_0 + a_1 x + a_2 x^2 + a_3 x^3.$$

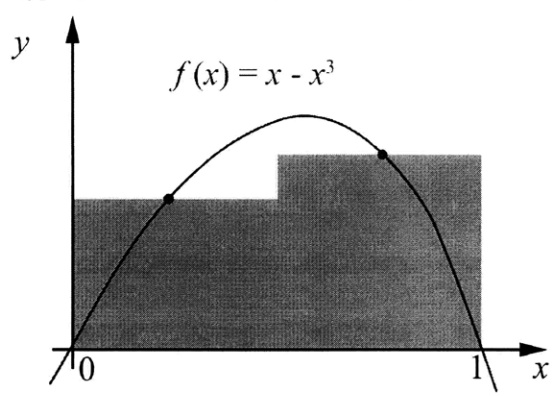
Given the exactness condition, the integral can be written:

$$\int_0^1 (a_0+a_1x+a_2x^2+a_3x^3)dx = w_1(a_0+a_1x_1+a_2x_1^2+a_3x_1^3)+w_2(a_0+a_1x_2+a_2x_2^2+a_3x_2^3)$$

Gather the like terms:

a_0 : $w_1 + w_2 = \int_0^1 dx = 1$
a_1 : $w_1x_1 + w_2x_2 = \int_0^1 xdx = \frac{1}{2}$
a_2 : $w_1x_1^2 + w_2x_2^2 = \int_0^1 x^2dx = \frac{1}{3}$
a_3 : $w_1x_1^3 + w_2x_2^3 = \int_0^1 x^3dx = \frac{1}{4}$

There are four equations above and four unknowns, so the system can be easily solved to give, $w_1 = \frac{1}{2}$ $w_2 = \frac{1}{2}$ $x_1 = \frac{3-\sqrt{3}}{6}$ $x_2 = \frac{3+\sqrt{3}}{6}$.
Plugging in these values, for the equation above, gives the exact solution of 0.25.



6.2 Summary

This slide summarizes the technique of finding weights and integration points for Gauss quadrature.

6.2.1 Accuracy Result

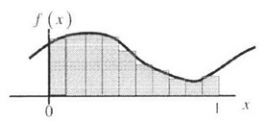
$$\int_0^1 f(x)dx \simeq \sum_{i=1}^{n} w_i f(x_i)$$

Key properties of the method

- An n-point Gauss quadrature rule is **exact** for polynomials of order $2n-1$
- Error is proportional to $\frac{1}{(2n)!}$ (like $\frac{1}{n^n}$)

6.2.2 Simple vs. Gauss Quadrature

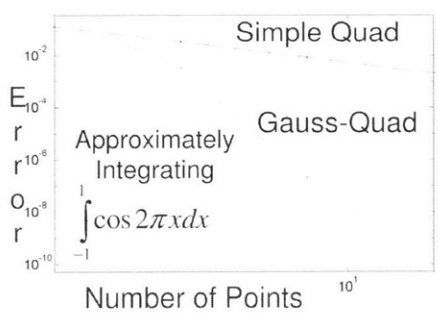
$$\int_0^1 f(x)dx \simeq \sum_{i=1}^{n} \frac{1}{n} f\left(\frac{i-\frac{1}{2}}{n}\right)$$



Key property of the method

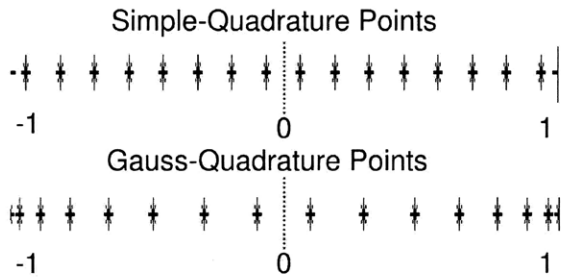
- Error is proportional to $\frac{1}{n^2}$

▷ ***Exercise 3*** Do you see that the simple quadrature scheme is a special case of Gauss quadrature? ■



Notice that for a smooth function $f(x) = cos(2\pi x)$, which is infinitely differentiable, Gauss quadrature far outperforms the simple quadrature scheme

6.2.3 Evaluation Point Placement



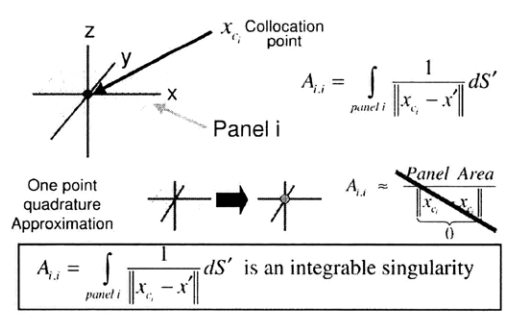
Notice the clustering at interval ends

In the Gauss quadrature scheme the evaluation points are roots of Legendre polynomials which are clustered at the ends of the interval.

7 The Singular Kernel Problem

7.1 Calculating the "Self-Term"

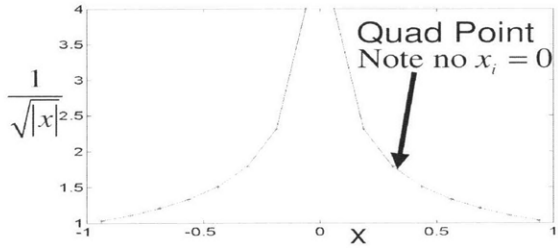
Now lets go back to our problem of solving Laplace's equation on a 3D domain using boundary integral representation. We realize that we now have some sophisticated tools to handle integrals of functions that are smooth. But what about the integral on the panel where the centroid x_{c_i} is located? The Green's function blows up at the centroid. However, the function is integrable because the integrand blows up at a rate that is slower than the rate at which the surface measure goes to zero in the vicinity of the singularity. So we know that the integral exists and is finite, but is Gauss quadrature capable of performing well in the presence of this singularity?



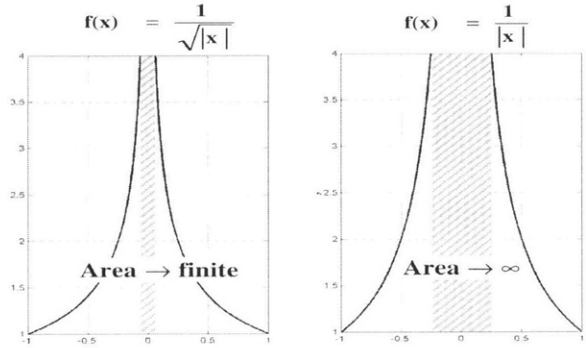
7.2 Symmetric 1-D Example

In 1D we look at a function $f(x) = \frac{1}{\sqrt{|x|}}$ which is integrable on $[-1, 1]$ but has a singularity at $x = 0$.

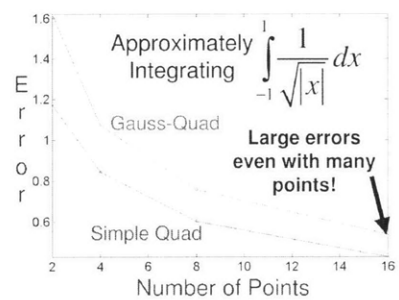
$$\int_{-1}^{1} \frac{1}{\sqrt{|x|}} dx \simeq \sum_{i=1}^{n} w_i \frac{1}{\sqrt{|x_i|}}$$



7.2.1 Integrable and Nonintegrable Singularities



7.2.2 Comparing Quadrature Schemes

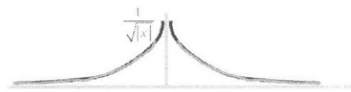


We observe that Gauss quadrature is not very good at integrating this function. The convergence is rather poor. As a matter of fact, it is more inaccurate than the simple quadrature scheme. In the next few slides we present several techniques of handling integrals with singularities (which are integrable, of course)

- Subinterval (adaptive) quadrature
- Change of variables of integration
- Singular (Gaussian) quadrature

7.3 Improved Techniques

7.3.1 Subinterval (Adaptive) Quadrature



Subdivide the integration interval

$$\int_{-1}^{1} \frac{1}{\sqrt{|x|}} dx = \int_{-1}^{-0.1} \frac{1}{\sqrt{|x|}} dx + \int_{-0.1}^{0} \frac{1}{\sqrt{|x|}} dx + \int_{0}^{0.1} \frac{1}{\sqrt{|x|}} dx + \int_{0.1}^{1} \frac{1}{\sqrt{|x|}} dx$$

Use Gauss quadrature in each subinterval
Polynomials fit subintervals better
Expensive if many subintervals used.

7.3.2 Change of Variables - for Simple Cases

Change variables to eliminate singularity

$$y^2 = x \qquad \Rightarrow \qquad 2ydy = dx$$

$$\int_{-1}^{1} \frac{1}{\sqrt{|x|}} dx = 2 \int_{0}^{1} \frac{1}{\sqrt{|y^2|}} 2ydy = 2 \int_{0}^{1} 2dy$$

Apply Gauss quadrature on desingularized integrand.

7.3.3 Singular Quadrature - Complicated Cases

Basic Concept

Integrand has known singularity $s(x)$

$\int_{-1}^{1} f(x)s(x)dx$ where $f(x)$ is smooth

Develop a quadrature formula exact for

$\int_{-1}^{1} p_l(x)s(x)dx$ where $p_l(x)$ is polynomial of order l

Calculate weights like Gauss quadrature

It is possible to generate Gaussian quadrature schemes of the form

$$\int_{-1}^{1} s(x)f(x)dx = \sum_{i=1}^{n} w_i f(x_i)$$

for functions which have a known singularity $s(x) > 0$. The quadrature formula needs to be exact when $f(x)$ is a polynomial of order at most l. Not surprisingly, it turns out that the integration points are the n roots of a polynomial $c_n(x)$ of degree $n = (l+1)/2$ which is orthogonal to all polynomials of inferior degree with respect to the weight $s(x)$, i.e.

$$\int_{-1}^{1} s(x)c_n(x)c_j(x) = 0 \qquad for\ j = 0, 1, \ldots, (n-1).$$

An example is the singular integral

$$I = \int_{-1}^{1} \frac{f(x)}{\sqrt{1-x^2}}$$

Here, $s(x) = 1/\sqrt{1-x^2}$ and the corresponding orthogonal polynomials turn out to be the Chebyshev polynomials. The integration points are given in closed form by

$$x_i = cos\left(\pi \frac{2i-1}{2n}\right)$$

and the corresponding weights are $w_i = \pi/n$.

Singular Quadrature Weights

$$\begin{bmatrix} c_0(x_1) & \cdots & c_0(x_n) \\ \vdots & \ddots & \vdots \\ c_{n-1}(x_1) & \cdots & c_{n-1}(x_n) \end{bmatrix} \begin{bmatrix} w_1 \\ \vdots \\ w_n \end{bmatrix} = \begin{bmatrix} \int_{-1}^{1} c_0(x)s(x)dx \\ \vdots \\ \int_{-1}^{1} c_{n-1}(x)s(x)dx \end{bmatrix}$$

Need (analytic) formulas for integrals of $c(x)s(x)$

The lower order constraints can be used to compute the integration weights.

8 Summary

Easy technique for computing integrals
Piecewise constant approach
Gaussian quadrature

Faster convergence
Essential role of orthogonal polynomials
Techniques for singular kernels
Adaptation and Variable Transformation
Singular quadrature
What about multiple dimensions?

1.3 First and Second Kind Equations

Numerical Methods for PDEs

Integral Equation Methods, Lecture 3
First and Second Kind Equations

Notes by L. Proctor, S. De and J. White

December 1, 2008

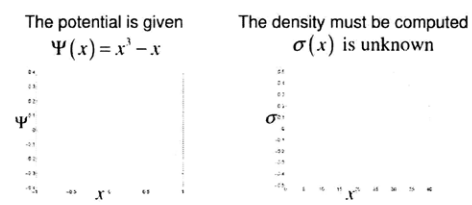
1 Outline

Convergence issues in 1D
First and second kind integral equations
Develop some intuition about the difficulties
Convergence for second kind equations
Consistency and stability issues
Nystrom Methods
High order convergence

2 Example Problems in 1D

2.1 First Kind Equation

$$\Psi(x) = \int_{-1}^{1} |x - x'|\sigma(x')dS' \qquad x \in [-1, 1]$$

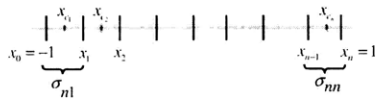


In the next several slides we will investigate the convergence properties of integral equation discretization methods. How these methods converge depends on what kind of integral equation is being solving. Examining this issue will introduce one of the subtle points about integral equations.
To begin, consider the example one-dimensional first-kind integral equation on the above slide. For this equation, we assume that the potential, $\Phi(x)$, is known and that the charge density $\sigma(x)$ is unknown. Here, x is in the interval $[-1, 1]$, and the integration is over that same interval. Note that for this example, the Green's function is given by $G(x, x') = |x - x'|$.
In the left plot below the equation, an example potential, $x^3 - x$, is given and plotted as a function of x. On the right is a plot of charge density as a function of x which is a possible solution to the integral equation. As we will see shortly, the solution for the charge density is not so easy to find.

2.1.1 Collocation Discretization

$$\Psi(x) = \int_{-1}^{1} |x - x'|\sigma(x')dS' \qquad x \in [-1, 1] \tag{1}$$

<u>**Piecewise-Constant Centroid-Collocation**</u>



$$\Psi(x_{c_i}) = \sum_{j=1}^{n} \sigma_{nj} \int_{x_{j-1}}^{x_j} |x_{c_i} - x'| dS' \tag{2}$$

To compute the numerical solution to this one-dimensional problem, consider solving integral equation (1) using a piecewise-constant collocation scheme. In such a scheme, we first select $n+1$ points on the interval, in this case $[-1, 1]$. We denote those points as $\{x_0, x_1, ..., x_n\}$, as shown in the above figure. For this example, $x_0 = -1$ and $x_n = 1$. Over the subintervals define a set of basis functions, $\{\varphi_1(x), \varphi_2(x), ..., \varphi_n(x)\}$, where

$$\varphi_i(x) = \left\{ \begin{array}{ll} 1 & x \in [x_{i-1}, x_i] \\ 0 & \text{otherwise} \end{array} \right\}. \tag{3}$$

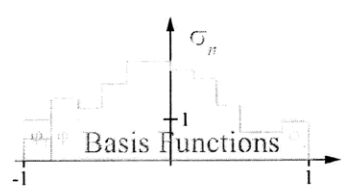
The charge density σ can then be represented approximately as

$$\sigma(x) \approx \sigma_n(x) \equiv \sum_{i=1}^{n} \sigma_{ni} \varphi_i(x), \tag{4}$$

where σ_{ni} is the weight associated with the ith basis function. It may seem odd that we used the same letter to represent the density and the basis function weights, but there is a reason. The above basis set is such that only one basis function is nonzero for a given x, and basis functions only take on the value zero or one. Therefore, σ_{ni} will be equal to the approximate charge density when $x \in [x_{i-1}, x_i]$.

Charge Density Representation

$$\sigma_n(x) \equiv \sum_{i=1}^{n} \sigma_{ni} \varphi_i(x)$$



Plugging the basis function representation of the charge density, equation (4), into equation (1) yields

$$\Psi(x) = \int_{-1}^{1} |x - x'| \sum_{i=1}^{n} \sigma_{ni} \varphi_i(x') dS',$$

which can be simplified by exploiting equation (3). Recall that the residual, $R(x)$, is defined as how well the weighted combination of basis functions satisfies the integral equation. In this centroid collocation case,

$$R(x) = \Psi(x) - \sum_{j=1}^{n} \sigma_{nj} \int_{x_{j-1}}^{x_j} |x - x'| dS'.$$

If collocation is used to solve this equation, then $R(x_{c_i}) = 0$ for all i, where x_{c_i} is the ith collocation point. The collocation points shown in the figure are the subinterval center points, $x_{c_i} = \frac{1}{2}(x_{i-1} + x_i)$. Note that there are other choices for collocation points, such as $x_{c_i} = x_i$.
Using the fact that $R(x_{c_i}) = 0$ leads to

$$R(x_{c_i}) = \Psi(x_{c_i}) - \sum_{j=1}^{n} \sigma_{nj} \int_{x_{j-1}}^{x_j} |x_{c_i} - x'| dS' = 0$$

which can be reorganized into equation (2).

The Matrix

We can now generate a system of equations that can be used to solve for the σ_{ni}'s, the piecewise constant charge densities for each of the subintervals.

One column for each density value

$$\begin{bmatrix} \int_{x_0}^{x_1} |x_{c_1} - x'| dS' & \cdots & \int_{x_{n-1}}^{x_n} |x_{c_1} - x'| dS' \\ \vdots & \ddots & \vdots \\ \int_{x_0}^{x_1} |x_{c_n} - x'| dS' & \cdots & \int_{x_{n-1}}^{x_n} |x_{c_n} - x'| dS' \end{bmatrix} \begin{bmatrix} \sigma_{n1} \\ \vdots \\ \sigma_{nn} \end{bmatrix} = \begin{bmatrix} \Psi(x_{c_1}) \\ \vdots \\ \Psi(x_{c_n}) \end{bmatrix}$$

One row for each collocation point

The right-hand side of this system of equations is a vector of known potentials at the interval centers (the collocation points). The ith row of the matrix corresponds to unfolding the sum in the collocation equation

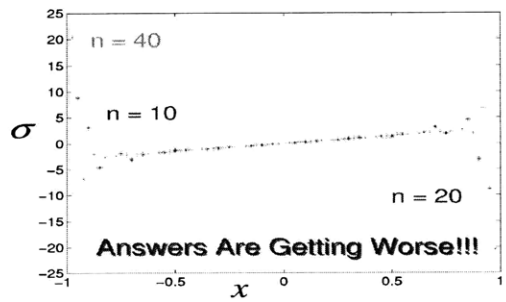
$$\Psi(x_{c_i}) = \sum_{j=1}^{n} \sigma_{nj} \int_{x_{j-1}}^{x_j} |x_{c_i} - x'| dS',$$

and the entries in the jth column correspond to how much the charge on the jth interval contributes to the ith potential.
Note that the matrix is square and dense.

▷ ***Exercise 1*** Is the above matrix symmetric? If we used $x_{c_i} = x_i$, would the matrix still be symmetric? ■

2.1.2 Numerical Results with Increasing n

One usually believes that a discretization scheme should produce progressively more accurate answers as the discretization is refined. In this case, as we divide the interval into progressively finer subintervals, one might expect that the piecewise constant representation of the charge density given by $\sigma_n(x) \approx \sum_{i=1}^{n} \sigma_{ni}\varphi_i(x)$ would become more accurate as n increases.

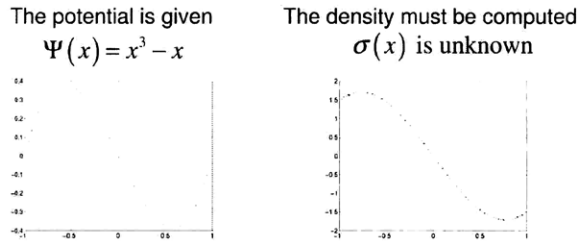


As is clear from the above plot, the piecewise-constant centroid-collocation discretization of (1) is not converging. In the plot, which is hard to decipher without looking at a color version, the σ_{ni}'s produced using $n = 10$, $n = 20$ and $n = 40$ subintervals are shown. For each discretization, a point is plotted at σ_{ni}, x_i for $i = 1, .., n$, so there are ten points plotted for the coarsest discretization and forty points plotted for the finest discretization, but all sets of points span the interval $x \in [-1, 1]$.
What is clear from comparing the blue points ($n = 10$) to the red points ($n = 20$) and to the green points ($n = 40$), is that the charge density seems to be approaching infinity as the discretization is refined. The results are certainly not converging.
Why is this happening? Is the numerical technique at fault, or is the integral equation a problem?

2.2 Second Kind Equation

We are going to postpone examining what went wrong with the discretization the first-kind integral equation, and instead examine a Second Kind integral equation example. As in the first-kind case, we are assuming the potential, $\Psi(x)$, is known and that the charge density $\sigma(x)$ is unknown. Also, x is in the interval $[-1, 1]$, and the integration is over that same interval. Once again, the Green's function is given by $G(x, x') = |x - x'|$.

$$\Psi(x) = \sigma(x) + \int_{-1}^{1} |x - x'|\sigma(x')dS' \qquad x \in [-1, 1]$$



The above equation is second-kind instead of first-kind because the unknown charge density appears both inside *and* outside of the integral. In the first-kind equation, the density appeared only inside the integral. This seemingly small difference has enormous numerical ramifications.
In the left plot above, an example given potential, $\Psi(x) = x^3 - x$ is plotted as a function of x. On the right is a plot of a charge density as a function of x which satisfies this second kind integral equation. As we will see below, this equation is easily solved numerically.

2.2.1 Collocation Discretization

$$\Psi(x) = \sigma(x) + \int_{-1}^{1} |x - x'|\sigma(x')dS' \quad x \in [-1, 1] \tag{5}$$

<u>Centroid Collocated Piecewise Constant Scheme</u>

$$\Psi(x_{c_i}) = \sigma_{ni} + \sum_{j=1}^{n} \sigma_{nj} \int_{x_{j-1}}^{x_j} |x_{c_i} - x'|dS' \tag{6}$$

To compute the numerical solution to the one-dimensional second-kind equation (5), once again consider using a piecewise-constant collocation scheme. Again, we select $n+1$ points on the interval and denote those points as $\{x_0, x_1, ..., x_n\}$, as shown in the figure above. For this example, $x_0 = -1$ and $x_n = 1$. The corresponding basis functions, $\{\varphi_1(x), \varphi_2(x), ..., \varphi_n(x)\}$, are the same as in equation (3):

$$\varphi_i(x) = \left\{ \begin{array}{ll} 1 & x \in [x_{i-1}, x_i] \\ 0 & \text{otherwise} \end{array} \right\}. \tag{7}$$

The charge density σ is approximately represented by

$$\sigma(x) \simeq \sigma_n(x) \equiv \sum_{i=1}^{n} \sigma_{ni}\varphi_i(x), \tag{8}$$

where σ_{ni} is the weight associated with the ith basis function.
Plugging the basis function representation of the charge density, equation (8) into the second kind integral equation (5) gives:

$$\Psi(x) = \sum_{j=1}^{n} \sigma_{nj}\varphi_j(x) + \int_{-1}^{1} |x - x'| \sum_{i=1}^{n} \sigma_{ni}\varphi_i(x')dS',$$

which can be simplified by exploiting the specific basis functions, equation (7) to

$$\Psi(x) = \sum_{j=1}^{n} \sigma_{nj}\varphi_j(x) + \sum_{j=1}^{n} \sigma_{nj} \int_{x_{j-1}}^{x_j} |x - x'|dS'. \tag{9}$$

As shown in the above figure, the collocation points are the subinterval center points, $x_{c_i} = \frac{1}{2}(x_{i-1} + x_i)$. When collocation is used, equation (9) must be satisfied exactly at the collocation points and therefore

$$\Psi(x_{c_i}) = \sum_{i=1}^{n} \sigma_{ni}\varphi_j(x_{c_i}) + \sum_{j=1}^{n} \sigma_{nj} \int_{x_{j-1}}^{x_j} |x_{c_i} - x'|dS'. \tag{10}$$

Note that $\varphi_j(x_{c_i}) = 0$ when $i \neq j$, and $\varphi_i(x_{c_i}) = 1$. Using this fact yields equation (6).

The Matrix

Just as in the discretized first-kind equation, we generate a system of equations that can be used to solve for the σ_{ni}'s, the piecewise constant charge densities for each of the subintervals.

$$\begin{bmatrix} 1+\int_{x_0}^{x_1}\left|x_{c_1} - x'\right|dS' & \cdots & \int_{x_{n-1}}^{x_n}\left|x_{c_1} - x'\right|dS' \\ \vdots & \ddots & \vdots \\ \int_{x_0}^{x_1}\left|x_{c_n} - x'\right|dS' & \cdots & 1+\int_{x_{n-1}}^{x_n}\left|x_{c_n} - x'\right|dS' \end{bmatrix} \begin{bmatrix} \sigma_{n1} \\ \vdots \\ \sigma_{nn} \end{bmatrix} = \begin{bmatrix} \Psi\left(x_{c_1}\right) \\ \vdots \\ \Psi\left(x_{c_n}\right) \end{bmatrix}$$

The right-hand side of this system of equations is a vector of known potentials at interval centers (the collocation points). The ith row of the matrix corresponds to unfolding the sum in the collocation equation

$$\Psi(x_{c_i}) = \sigma_{ni} + \sum_{j=1}^{n} \sigma_{nj} \int_{x_{j-1}}^{x_j} |x_{c_i} - x'|dS'$$

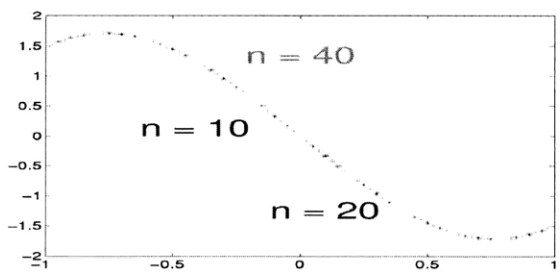
and the entries in the jth column corresponds to how much the charge on the jth interval contributes to the ith potential.

The major difference between the matrix in this discretized second-kind example and the first-kind example is circled on the above slide. There is an additional one on the diagonal of the discretized second-kind equation that did not appear in the first-kind equation. In other words,

$$A_{second\ kind} = I + A_{first\ kind}.$$

2.2.2 Numerical Results with Increasing n

Unlike the results from discretizing the first kind equation, progressively refining the discretization of the second kind equation produces more accurate answers.



Answers Are Improving!!!

Once again, the plot is a little hard to decipher without looking at the color version. It shows the σ_{ni}'s produced using $n = 10$, $n = 20$ and $n = 40$ subintervals. For each discretization, a point is plotted at σ_{ni}, x_i for $i = 1, .., n$, so there are ten points plotted for the coarsest discretization and forty points plotted for the finest discretization, but all sets of points span the interval $x \in [-1, 1]$.
What is clear from comparing the blue points ($n = 10$) to the red points ($n = 20$) and to the green points ($n = 40$), is that the charge density seems to be approaching a smooth solution.
What is the essential difference between first and second kind equations?
Is it some aspect of the numerical technique or are these two equations really that different? In the next slides, we will try to answer this question.

2.3 Difficulty with the First Kind Equation

We will make use of operator-function notation for much of the next sections, both for clarity and brevity. For example, the charge density,σ, and the potential, Ψ, are functions of the independent variable x. When we mean the function, we use just use the function name, such as σ or Ψ. When we give an explicit formula in terms of x for the function, or are denoting the function's value for a particular $\hat{x}$, we use follow the function name by the value of the

independent variable in parentheses. For example, $\sigma\hat{x})$ is the value of function σ when evaluated at $x = \hat{x}$.

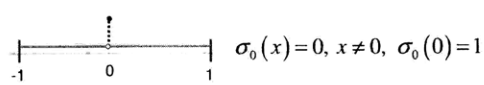
The operator-function notation is a little less obvious in the case of operators that map functions to functions, like the integral operator. The integral operator takes a function, in this case σ, and produces another function that we might refer to as the potential. If we denote the integral operator from (5) as K, then $K\sigma$ is a function. If we wish to evaluate the function generated by applying K to σ at some $\hat{x}$, then we write $(K\sigma)(x)$. Note that $K\sigma(x)$ would *NOT BE CORRECT*. The operator K takes functions and $\sigma(x)$ is a value.

2.3.1 Singular Integral Operator

Denote the integral operator as K

$$(K\sigma)(x) \equiv \int_{-1}^{1} |x - x'|\sigma(x')dS' = \Psi(x) \Rightarrow K\sigma = \Psi \tag{11}$$

The integral operator is **singular** : K **has a null space**



$$(K\sigma_0)(x) = \int_{-1}^{1} |x - x'|\sigma_0(x')dS' = 0 \ for\ all\ x \tag{12}$$

$$\Rightarrow K\sigma_0 = 0 \tag{13}$$

$$\text{If} \quad K\sigma^a = \Psi \quad \text{then} \quad K(\sigma^a + \sigma_0) = \Psi$$

In equation (11), we introduce the abstract notion that

$$\int_{-1}^{1} |x - x'|\sigma(x')dS'$$

is an operator on the function σ, which we denote with the symbol K. As shown on the top of the slide, this notation makes writing the integral equation look just like writing a matrix equation.
The key problem is that the operator K is singular. And if

$$K\sigma = \Psi$$

were a matrix equation with a singular K, one would not be surprised to discover the system of equations is hard, or impossible, to solve.
We will not try, in this lecture, to be formal about the concept of a singular operator. To do so, we would necessarily be examining details about certain types of function spaces. Instead, we will try to develop some intuition. In particular, we will draw an analogy to matrices and note that if an operator is

singular, it must have a null space.
To see that K does have a null space, consider the spike function $\sigma_0(x)$ depicted in equation (13). This spike function is one at $x = 0$ and zero otherwise. Note, this function is *not* an impulse function. Unlike the impulse function, the spike's value at $x = 0$ is finite and the area under its curve is obviously zero.
As noted in equation (13), $K\sigma_0 = 0$. To see this consider that since σ_0 is nonzero only at $x = 0$, and therefore

$$\int_{-1}^{1} |x - x'|\sigma_0(x')dS' = |x| \int_{-1}^{1} \sigma_0(x')dS'.$$

Since $\int_{-1}^{1} \sigma_0(x')dS' = 0$, as the area under σ_0's curve is zero, then $K\sigma_0 = 0$.
The statement "If $K\sigma^a = \Psi$ then $K(\sigma^a + \sigma_0) = \Psi$" says that if K has a null space, and there exists a solution, then there exist infinitely many solutions.
One last comment should be made. The spike function we generated is not unique. Simply shifting the nonzero point would generate and infinite number of spike functions which would all be in the null space of K. That is, K has an incredibly rich null space.

2.3.2 Eigenvalues

Difficulty from the Matrix

Collocation generates a discrete form of K

$$K\sigma = \Psi \quad \rightarrow \quad \underline{K}_n \vec{\sigma}_n = \vec{\Psi}_n$$

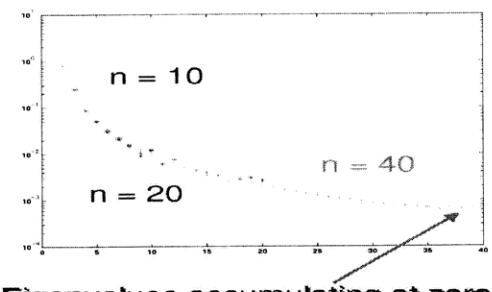
$$\underbrace{\begin{bmatrix} \int_{x_0}^{x_1} |x_{c_1} - x'|dS' & \cdots & \int_{x_{n-1}}^{x_n} |x_{c_1} - x'|dS' \\ \vdots & \ddots & \vdots \\ \int_{x_0}^{x_1} |x_{c_n} - x'|dS' & \cdots & \int_{x_{n-1}}^{x_n} |x_{c_n} - x'|dS' \end{bmatrix}}_{\underline{K}n} \begin{bmatrix} \sigma_{n1} \\ \vdots \\ \sigma_{nn} \end{bmatrix} = \begin{bmatrix} \Psi(x_{c_1}) \\ \vdots \\ \Psi(x_{c_n}) \end{bmatrix}$$

K: functions to functions, $\underline{K}_n$:: vectors to vectors!

As shown above, discretizing the integral equation by combining a piecewise constant charge density representation with collocation at subinterval centers results in a system of equations which relates the subinterval σ_i's to the collocation point potentials. From this perspective, the matrix above can be thought of as a discrete representation of the operator K. We denote the matrix with $\underline{K}_n$ with an underline to indicate that it is a matrix, and was generated using a discretization with n basis functions. We also denoted the vectors $\vec{\sigma}_n$ and $\vec{\Psi}_n$ with arrows to avoid confusing vectors with functions. Later, we will need an operator version of the discretized representation of the operator K, but for the moment, the matrix is sufficient.

Numerical Results with Increasing n

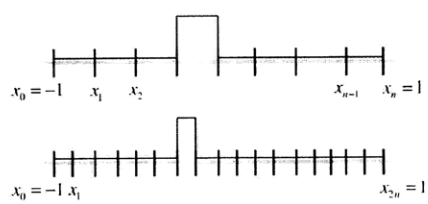
If the operator K is singular, one might expect to see that reflected in the eigenvalues of a matrix generated by discretizing K. In particular, one would expect the matrix to have eigenvalues that are near zero.



In the figure above, the eigenvalues of matrices generated by discretizing K for the 1-D problem are plotted. Discretizing using 10 subintervals generates a matrix with 10 eigenvalues plotted in blue. The blue eigenvalue closest to zero is ≈ 0.01. As the discretization is refined to 20 subintervals, the minimum eigenvalue (plotted in red) drops to ≈ 0.003, and with 40 subintervals the minimum eigenvalue (plotted in green) drops to ≈ 0.0009. Examining this data suggests that as the discretization is refined, the generated matrix more accurately reflects the operator K, and therefore the matrix is becoming closer to being singular.
As the discretization is refined, the matrix is larger and has more eigenvalues. Notice that as the discretization is refined from $n = 10$ to $n = 20$ to $n = 40$, all the additional eigenvalues are closer to zero.

Intuition About Eigenvalues

As the discretization is refined, $\sigma_0(x)$ becomes better approximated



As the discretization is refined, K's null space can be more accurately represented.

As an alternative view of why refining the discretization for the first kind equation produces a matrix with more and more smaller eigenvalues, consider the figures above. In the top plot, one of the basis functions is plotted for a coarse discretization. In the bottom plot, one of the basis functions is plotted for a finer discretization. Notice that as the discretization is refined, these basis functions look progressively more like the spike function mentioned previously. And since the spike function is in the null space of K, one would expect that

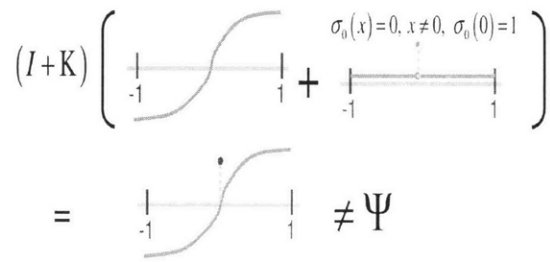
finer discretizations would generate "spikier" basis functions whose associated eigenvalues would be near zero.

2.4 Second Kind Equation Has Fewer Problems

Second Kind equation

$$((I+K)\sigma)(x) \equiv \sigma(x) + \int_{-1}^{1} |x-x'|\sigma(x')dS' = \Psi(x)$$
$$\Rightarrow (I+K)\sigma = \Psi \tag{14}$$

$$(I+K)(\sigma_0+\sigma) \neq (I+K)\sigma$$



As shown in equation (14), the abstract operator for the second-kind equation is denoted by $I+K$, where I here is just the identity operator and K is the integral operator.
To see why the spike function, σ_0, is not in the null space of the operator $I+K$, or equivalently that

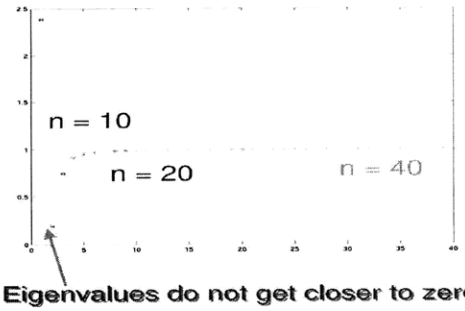
$$(I+K)(\sigma_0+\sigma) \neq (I+K)(\sigma)$$

consider the figures above. If a spike is added to a smooth σ, the $(I+K)$ operator will preserve the spike. Another way to see this is to consider that since σ_0 is in the null space of K,

$$(I+K)\sigma_0 = (I)\sigma_0 + K\sigma_0 = \sigma_0 \neq 0.$$

2.4.1 Eigenvalues

Numerical Results with Increasing n



As we noted before, the matrix associated with discretizing the operator $I + K$ is identical to the sum of the identity matrix and the matrix associated with discretizing K alone. In the plot above, we once again present the eigenvalues generated by discretizing the 1-D example problem. Discretizing using 10 subintervals generates a matrix with 10 eigenvalues plotted in blue. The blue eigenvalue closest to zero is ≈ 0.2. As the discretization is refined to 20 subintervals, the minimum eigenvalue (plotted in red) is still ≈ 0.2, and with 40 subintervals the minimum eigenvalue (plotted in green) is still ≈ 0.2. Examining this data suggests that as the discretization is refined, and the generated matrix more accurately reflects the operator $I + K$, the matrix is not becoming more singular. In fact, the eigenvalues are accumulating near one, an unsurprising result given that the eigenvalues of the discretized K operator were accumulating at zero.

▷ ***Exercise 2*** Estimate how many iterations will be needed for a Krylov-subspace based algorithm to converge for the 1-D discretized second-kind example. Will the number of iterations increase as the discretization is refined? ■

▷ ***Exercise 3*** Suppose the integral equation were changed to

$$\Psi(x) = \sigma(x) + \frac{1}{\lambda}\int_{-1}^{1} |x - x'|\sigma(x')dS'.$$

For what value of λ would the solution no longer be unique. (you can answer this just by looking at the eigenplot above). ■

As the above exercise makes clear, a second-kind integral equation does not always have a unique solution. However, a first-kind equation almost never has a unique solution, the exception being when the Green's function is singular, as we will investigate in a subsequent lecture.

3 Theory of 2nd Kind Equations

The convergence theory for discretization methods applied to second-kind integral equations has an elegant simplicity, but only when examined using a care-

fully chosen abstraction. The theory is also surprisingly practical; the insights gained can be used to construct very high-order discretization schemes.

3.1 Comparison problem

General Second kind integral equation

$$\Psi(x) = \sigma(x) + \int G(x, x')\sigma(x')dx' \Rightarrow \Psi = (I + K)\sigma \tag{15}$$

Discrete matrix equivalent

$$\vec{\Psi}_n = (I + \underline{K}_n)\,\vec{\sigma}_n \tag{16}$$

How to compare function σ to vector $\vec{\sigma}_n$?
How to compare operator K to matrix $\underline{K}_n$?

One approach to overcoming the comparison problems is to construct representations of the discretization that are functions and operators on functions. The most obvious approach to generating the functions associated with a discretization is by interpolation, but generating the operators associated with the discretization is a little more subtle.

3.1.1 Operator-Function Notation

General Second kind integral equation

$$\Psi(x) = \sigma(x) + \int G(x, x')\sigma(x')dx' \Rightarrow \Psi = (I + K)\sigma \tag{17}$$

Discretized operator-function equivalent

$$\Psi_n = (I + K_n)\,\sigma_n \tag{18}$$

σ_n, Ψ_n are functions of x (e.g. by interpolation)
K_n maps functions to functions like K (How constructed)?

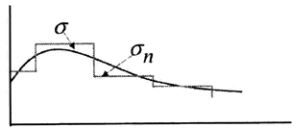
3.1.2 Orthogonal Galerkin

Representation $\quad \sigma_n(x) = \sum_{i=1}^{n} \sigma_{ni}\varphi_i(x) \quad \int \varphi_i(x)\varphi_j(x) = \delta(i-j)$

Projection $\quad \sigma_n = (P\sigma)(x)$

$$(P\sigma)(x) \equiv \sum_{i=1}^{n} \overbrace{\left(\int \sigma(x)\varphi_i(x)dx\right)}^{\sigma_{ni}} \varphi_i(x)$$

$$(P\sigma)(x) = \sum_{i=1}^{n} \sigma_{ni}\varphi_i(x)$$



If the density *sigma* is to be approximated by a weighted combination of n orthogonal basis functions, then the functional representation σ_n associated with the vector of weights $\vec{\sigma}_n$ is given. If the basis functions are orthonormal, then the σ_n associated with an arbitrary σ can be constructed by simple projection. It is worth noting that the projection operator, denoted P, has no effect on σ_n. That is $P\sigma_n = \sigma_n$.

▷ ***Exercise 4*** Why does the formula for the projection operator above require orthogonality of the basis functions? ■

If a Galerkin method is used to discretized the integral equation, then the associated operator is easy to construct, as shown below.

3.1.3 Ortho Galerkin Operator

$$(K\sigma_n)(x) = (KP\sigma)(x) = \sum_{i=1}^{n} \sigma_{ni} \int G(x,x')\varphi_i(x')dx'$$

$$(PKP\sigma)(x) = \sum_{j=1}^{n} \left(\int \varphi_j(x) KP\sigma(x) dx \right) \varphi_j(x)$$

$$= \sum_{j=1}^{n} \left(\sum_{i=1}^{n} \sigma_{ni} \int \int \varphi_j(x) G(x,x') \varphi_i(x') dx dx' \right) \varphi_j(x)$$

$$P(I+KP)\sigma_n = (I+PKP)\sigma_n = P\Psi$$

$$(I+K_n)\sigma_n = \Psi_n$$

The last equation on the above slide contains a subtle point. $P(I+KP)\sigma_n$ really equals $(P+PKP)\sigma_n$. However, P is equivalent to the identity operator when applied to σ_n, as projecting σ_n reproduces σ_n. So, we are free to conveniently chose to interpret $(P+PKP)$ as $(I+PKP)$ as a difference appears only when applying the operator to functions that are not weighted combinations of the n Galerkin basis functions.

For second-kind integral equations, one can prove a convergence theory for almost any reasonable discretization scheme, assuming that the equation has a

unique solution. As noted above, second-kind integral equations do not necessarily have unique solutions, but we will restrict ourselves to the unique solution case in analyzing convergence. In particular, we will assume that the second-kind integral equation operator has a bounded inverse.
Before beginning the derivation, let's readdress the notation definitions.
Let K denote the integral operator, and therefore the general form is

$$K\sigma = \int G(x, x')\sigma(x')dx'$$

Let σ_n denote a numerical approximation to σ on x based on using n basis functions. Note here that σ_n is a function of x and would typically be given by

$$\sigma_n(x) = \sum_{i=1}^{n} \sigma_{ni}\varphi_i(x).$$

Let K_n be the discrete representation of the integral operator. Note that K_n is *not* the matrix $\underline{K}_n$, but an operator that maps a function of x into another function of x. For example, if the discretization scheme uses a basis to approximate σ, and the basis weights were determined by a collocation scheme, a not necessarily unique associated K_n could be given by

$$(K_n\sigma)(x) = V\left(\int G(x, x')P\sigma(x')dx'\right)$$

where in the orthonormal basis set case

$$(P\sigma)(x) = \sum_{i=1}^{n}\left(\int \sigma(x')\varphi_i(x')dx'\right)\varphi_i(x), \tag{19}$$

and

$$(Vu)(x) = \sum_{i=1}^{n} u(x_{c_i})\varphi_i(x). \tag{20}$$

where u is a arbitrary function used to define the action of operator V.
Equations (19) and (20) deserve some explanation. The piecewise constant basis is orthonormal, so the formula in equation (19) is a simple projection of σ onto the basis. If centroid collocation is used, then the discrete potentials computed by evaluating the integral operator at the collocation points must be converted to a function of x by interpolation. In equation (20), the $\varphi_i(x)$'s act as interpolation functions.
With the examples of how Galerkin and centroid-collocation discretization schemes lead to function and operator representations, the second-kind integral equation convergence theory can be presented in a very transparent fashion, as will be show below. What the theorem demonstrates is that if a discretization scheme generates progressively more accurate representations of the integral operator as n increases, then the discretization method converges. That is,

$$\lim_{n\to\infty} \|\sigma - \sigma_n\| \to 0,$$

where the comparison between σ and σ_n is unambigious as both are functions of x.

▷ ***Exercise 5*** Suppose a nonorthogonal basis is used to represent σ. How would the projection operator in equation (19) change? ■

3.2 Main Theorem

$$\text{Given} \quad (I+K)\sigma = \Psi \quad \& \quad \|(I+K)^{-1}\| < C$$

Means Equation uniquely solvable

$$(I+K_n)\sigma_n = \Psi_n$$

Reminder of Discrete Equivalent

Consistency:

If $\lim_{n\to\infty} \max_{\|\sigma\|=1} \|(K-K_n)\sigma\| \to 0$ and $\lim_{n\to\infty} \|\Psi - \Psi_n\| \to 0$

Then $\lim_{n\to\infty} \|\sigma - \sigma_n\| \to 0$

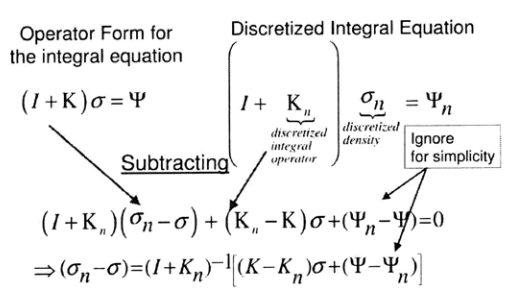
3.3 Rough Proof

To derive a relationship between the errors in the computed solution and the errors in the operator representation, we write the exact equation alongside the discrete equation.

Exact Equation	Discretized Equation
$\Psi = (I+K)\sigma$	$\Psi_n = (I+K_n)\sigma_n$

$$\begin{aligned}
&\Rightarrow \Psi - \Psi_n = (I+K)\sigma - (I+K_n)\sigma_n \\
&\Rightarrow (\Psi - \Psi_n) = (\sigma - \sigma_n) + K\sigma - K_n\sigma_n \\
&\Rightarrow (\Psi - \Psi_n) = (\sigma - \sigma_n) + K\sigma - K_n\sigma + K_n\sigma - K_n\sigma_n \\
&\Rightarrow (\Psi - \Psi_n) = (I+K_n)(\sigma - \sigma_n) + (K - K_n)\sigma \\
&\Rightarrow (\Psi - \Psi_n) - (K - K_n)\sigma = (I+K_n)(\sigma - \sigma_n) \\
&\Rightarrow (I+K_n)^{-1}\left[(\Psi - \Psi_n) - (K - K_n)\sigma\right] = (\sigma - \sigma_n).
\end{aligned}$$

The results on the slide below skip many of these intermediate steps but present the essential results.



The equation for the solution error (previous slide)

$$\underbrace{(\sigma_n - \sigma)}_{\text{solution error}} = (I + K_n)^{-1}(K - K_n)\sigma$$

Taking norms $\underbrace{||\sigma_n - \sigma||}_{\substack{\text{Error which}\\ \text{should go to}\\ \text{zero as } n\\ \text{increases}}} \leq \underbrace{||(I + K_n)^{-1}||}_{\substack{\text{Needs a}\\ \text{bound, that is}\\ \textbf{\textit{stability}}}} \underbrace{||(K - K_n)\sigma||}_{\substack{\text{Goes to}\\ \text{zero}\\ \text{by } \textbf{\textit{consistency}}}}$

We complete deriving a relationship between the errors in the computed solution and the errors in the operator representation. In order to establish that consistency implies convergence, the inverse of the discretized operator must be bounded.

3.3.1 Stability Bound

Norm of solution error
$||(\sigma_n - \sigma)|| \leq ||(I + K_n)^{-1}||\ ||(K - K_n)\sigma||$

Deriving the stability bound
$(I + K_n)^{-1} = [I + K - (K - K_n)]^{-1} = [(I - (I + K)^{-1}(K - K_n)]^{-1}(I + K)^{-1}$

Taking norms
$||(I + K_n)^{-1}|| \leq \underbrace{||(I + K)^{-1}||}_{\substack{\text{Bounded by C}\\ \text{by Assumption}}} \ ||\left(I - (I + K)^{-1}(K - K_n)\right)^{-1}||$

Repeating from last slide
$||(I + K_n)^{-1}|| \leq \underbrace{||(I + K)^{-1}||}_{\substack{\text{Bounded by C}\\ \text{by Assumption}}} \ ||\left(I - (I + K)^{-1}(K - K_n)\right)^{-1}||$

Bounding terms
$||(I + K_n)^{-1}|| \leq \dfrac{C}{1 - \underbrace{||(I + K)^{-1}(K - K_n)||}_{\substack{\text{Will be less than 0.5 for n larger}\\ \text{than some } n_0 \text{ by consistency and solvability}}}} < 2C$ for

$n \geq n_0$

Final result:

$$\lim_{n \to \infty} ||(K - K_n)\sigma|| = 0$$

IMPLIES

$$\lim_{n\to\infty} ||(\sigma_n - \sigma)|| = 0$$

What does this mean?

The discretization convergence of a second kind integral equation solver only depends on how well the integral is approximated.

The final result, noted on the above slide, is that the solution error is bounded by a constant multiplying the error in the integral operator representation. This suggests that any method which can accurately represent the integral operator can be used to discretize a second-kind integral equation.

4 Nystrom Method

4.1 1-D Second Kind Example

4.1.1 Collocation Discretization

Integral Equation

$$\Psi(x) = \sigma(x) + \int_{-1}^{1} G(x, x')\sigma(x')dS' \qquad x \in [-1, 1]$$

Apply **quadrature** to **Collocation equation**

$$\Psi(x_i) = \sigma(x_i) + \int_{-1}^{1} G(x_i, x')\sigma(x')dS'$$

$$\Rightarrow \Psi(x_i) = \sigma(x_i) + \sum_{j=1}^{n} w_j G(x_i, x_j)\sigma(x_j)$$

After applying quadrature to Collocation:

$$\Psi(x_i) = \sigma(x_i) + \sum_{j=1}^{n} w_j G(x_i, x_j)\sigma(x_j)$$

x_i is a collocation point
x_j's are quadrature points
Now set **quadrature points = collocation points**

In Gaussian quadrature, described in the previous lecture, an integral is approximated using weighted combinations of the integrand. As a reminder, the Gaussian quadrature formula for integrating a function on the unit interval is

$$\int_0^1 f(x)dx \simeq \sum_{i=1}^{n} w_i f(x_i)$$

where the x_i's are the evaluation points given by the zeros of an nth-order orthogonal polynomial on the unit interval, and the w_i's are the weights determined by solving exactness equations.

The key idea behind a Nystrom method for discretizing an integral equation is to use the Gaussian quadrature evaluation points as the test points in a collocation method for solving an integral equation. Then, the collocation method integrals can be approximated using the Gaussian quadrature scheme, resulting in a system of equations which only require evaluations of the integrand at the test=quadrature points. The second kind theory predicts that the error in such a scheme is proportional to the error in the quadrature scheme for computing the collocation integrals.

Set **quadrature points = collocation points**

$$\Psi(x_1) = \sigma_{n1} + \sum_{j=1}^{n} w_j G(x_1, x_j)\sigma_{nj}$$

$$\vdots$$

$$\Psi(x_n) = \sigma_{nn} + \sum_{j=1}^{n} w_j G(x_n, x_j)\sigma_{nj}$$

System of n equations in n unknowns

Collocation equation per quad/colloc point

Unknown density per quad/colloc point

4.1.2 Discretization-Matrix Comparison

Nystrom Matrix

$$\begin{bmatrix} 1+w_1G(x_1,x_1) & \cdots & w_nG(x_1,x_n) \\ \vdots & \ddots & \vdots \\ w_1G(x_n,x_1) & \cdots & 1+w_nG(x_n,x_n) \end{bmatrix} \begin{bmatrix} \sigma_{n1} \\ \vdots \\ \sigma_{nn} \end{bmatrix} = \begin{bmatrix} \Psi(x_1) \\ \vdots \\ \Psi(x_n) \end{bmatrix}$$

Piecewise Constant Collocation Matrix

$$\begin{bmatrix} 1+\int_{x_{n-1}}^{x_n} G(x_1,x')dS' & \cdots & \int_{x_{n-1}}^{x_n} G(x_1,x')dS' \\ \vdots & \ddots & \vdots \\ \int_{x_{n-1}}^{x_n} G(x_n,x')dS' & \cdots & 1+\int_{x_{n-1}}^{x_n} G(x_n,x')dS' \end{bmatrix} \begin{bmatrix} \sigma_{n1} \\ \vdots \\ \sigma_{nn} \end{bmatrix} = \begin{bmatrix} \Psi(x_1) \\ \vdots \\ \Psi(x_n) \end{bmatrix}$$

Nystrom Matrix

Just Green's function evaluations – No integrals
Entries each have a quadrature weight
Collocation points are quadrature points
High order quadrature=faster convergence?

Piecewise Constant Collocation Matrix

Integrals of Green's function over line sections
Distant terms equal Green's function
Collocation points are basis function centroids
Low order method always

4.2 K_n and Ψ_n for Nystrom Method

$$K_n\sigma = \sum_{i=1}^{n}\left(\sum_{j=1}^{n} w_j G(x_i, x_j)\sigma(x_j)\right)\varphi_i(x)$$

$$\Psi_n = \sum_{i=1}^{n}\Psi(x_i)\varphi_i(x)$$

4.3 Convergence

4.3.1 Theorem

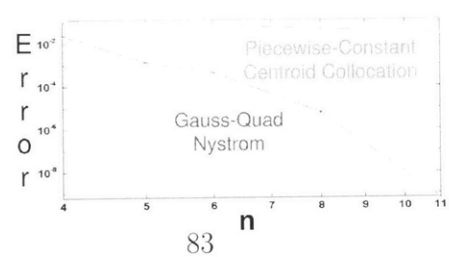
In the limit as $n \to \infty$ (number of quad points $\to \infty$)

The discretization error $= \max_{||\sigma||=1} ||(K - K_n)\sigma|| \to 0$
AT THE SAME RATE as the underlying quadrature!!

Gauss Quadrature $\Rightarrow$ Exponential Convergence!

4.3.2 Comparison

$$\cos 2\pi x = \sigma(x) + \int_{-1}^{1}(x - x')^2\sigma(x')dS'$$



4.3.3 Caveat

If Nystrom method can have exponential convergence, why use anything else?

Gaussian quadrature has exponential convergence for **nonsingular** kernels

Most physical problems of interest have **singular kernels** ($\frac{1}{r}$, $\frac{\exp ikr}{r}$, etc)

5 Summary

Convergence Issues in 1D

1st and 2nd kind integral equations, null spaces

Convergence for second kind equations

Show consistency and stability issues

Nystrom methods

High order convergence

Did not address singular integrands

1.4 Radiation Conditions and Formulations

Numerical Methods for PDEs

Integral Equation Methods, Lecture 4
Radiation Conditions and Formulations

Notes by L. Proctor, C. Coelho, and J. White

December 3, 2008

1 Outline

Laplace Problems
Exterior Radiation Condition
Potential Representations
Monopole and Dipole Densities
Principle Value
Ansantz and Green's Theorem
Dirichlet and Neumann problems

2 3D Problems

2.1 3D Laplace Equation

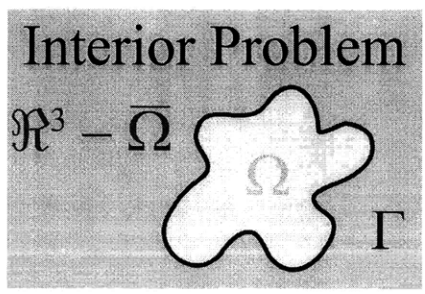
Laplace's equation in 3-D

$$\nabla^2 u(\vec{x}) = \frac{\partial^2 u(\vec{x})}{\partial x^2} + \frac{\partial^2 u(\vec{x})}{\partial y^2} + \frac{\partial^2 u(\vec{x})}{\partial z^2} = 0$$

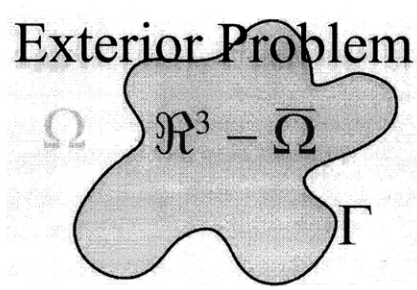
where

$$\vec{x} = (x, y, z) \in \Omega$$

and Ω is bounded by Γ.



The exterior problem is simply the region $\Re - \bar{\Omega}$ of the interior problem.



One feature of using integral equation methods is that exterior problems can be solved using the same surface discretization required to solve an interior problem (assuming a linear space-invariant problem like Laplace's equation). This is true even though the exterior domain is infinite and the interior domain is finite. Exterior problems do introduce an additional complication, one must consider the boundary condition "at infinity" (later).

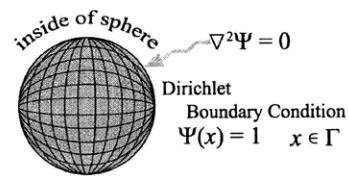
2.2 Boundary Conditions

2.2.1 Dirichlet

Dirichlet Condition

$$u(\vec{x}) = u_\Gamma(\vec{x}) \qquad \vec{x} \in \Gamma$$

Interior Problem

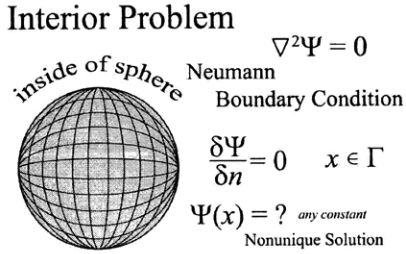


Can you determine the solution to Laplace's equation inside the sphere?

The solution of the interior Dirichlet problem is unique.

2.2.2 Neumann

Interior Problem



The solution of the <u>interior</u> Neumann problem is not unique.
For the solution of the <u>exterior</u> Neumann problem to be unique, it is sufficient to impose a *radiation condition*. In this case, a radiation condition would be a specification of how $u(\vec{x})$ approaches zero as $\vec{x} \to \infty$.

2.2.3 Exterior

Dirichlet Boundary Condition

$$u(\vec{x}) = u_\Gamma(\vec{x}) \qquad \vec{x} \in \Gamma$$

Neumann Boundary Condition

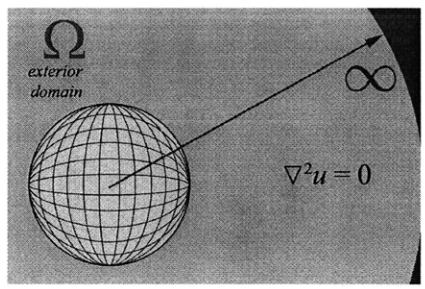
$$\frac{\partial u(\vec{x})}{\partial n_{\vec{x}}} = \frac{\partial u_\Gamma(\vec{x})}{\partial n_{\vec{x}}} \qquad \vec{x} \in \Gamma$$

PLUS

A Radiation Condition

3 Radiation Condition

3.1 Condition at "Infinity"



3.2 Types of Conditions

A radiation condition of the form

$$\lim_{\|\vec{x}\| \to \infty} u(\vec{x}) \to 0$$

is not specific enough!
Need

$$\lim_{\|\vec{x}\| \to \infty} u(\vec{x}) \to O(\|\vec{x}\|^{-1})$$

OR

$$\lim_{\|\vec{x}\| \to \infty} u(\vec{x}) \to O(\|\vec{x}\|^{-2})$$

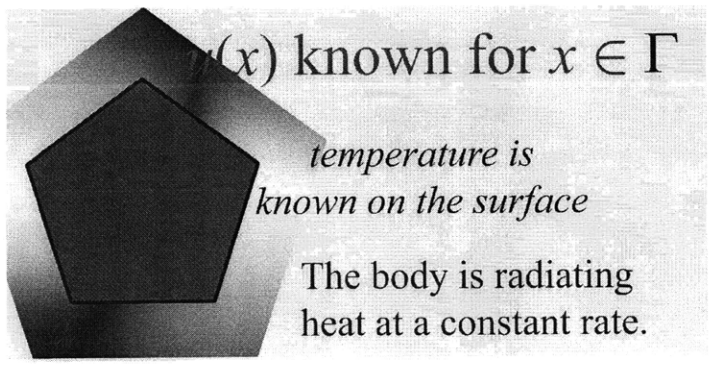
3.3 Examples

The criteria for chosing a radiation condition are best understood by considering several physical examples.

3.3.1 Radiated Heat

Problem Set-Up

$$\nabla^2 u(\vec{x}) = 0 \qquad \vec{x} \in \Omega$$
$$u(\vec{x}) = u_0(\vec{x}) \qquad \vec{x} \in \Gamma$$
$$\text{Radiated Heat} = \int_\Gamma \frac{\partial u}{\partial n} dS$$



Limit of Expanding Domain

- Infinite Problem
  Limit as $R \to \infty$

- Heat Leaving Sphere
  $\displaystyle\int_{Sphere_R} \frac{\partial u}{\partial n} dS$ is Constant!

3.3.2 Rad Heat Case 1

$$\lim_{\|\vec{x}\| \to \infty} u(\vec{x}) \to O(\|\vec{x}\|^{-1}) \to \lim_{\|\vec{x}\| \to \infty} \frac{\partial u(\vec{x})}{\partial n} \to O(\|\vec{x}\|^{-2})$$

Since the surface of a sphere increases as R^2:

$$\lim_{R \to \infty} \int_{Sphere_R} \frac{\partial u}{\partial n} dS \to \text{Constant}$$

Radiation condition models net heat loss.

3.3.3 Rad Heat Case 2

$$\lim_{\|\vec{x}\|\to\infty} u(\vec{x}) \to O(\|\vec{x}\|^{-2}) \to \lim_{\|\vec{x}\|\to\infty} \frac{\partial u(\vec{x})}{\partial n} \to O(\|\vec{x}\|^{-3})$$

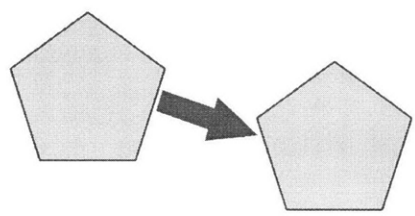
AND

$$\int_{Sphere_R} \frac{\partial u}{\partial n} dS \to 0$$

Can NOT model heat loss!

3.3.4 Heat Transfer

$$\nabla^2 u(\vec{x}) = 0 \quad \vec{x} \in \Omega \qquad u(\vec{x}) = u_0(\vec{x}) \quad \vec{x} \in \Gamma$$



Heat flows from higher temperature object to lower temperature object, but no heat radiates out.

If

$$\lim_{\|\vec{x}\|\to\infty} u(\vec{x}) \to O(\|\vec{x}\|^{-2})$$

Then

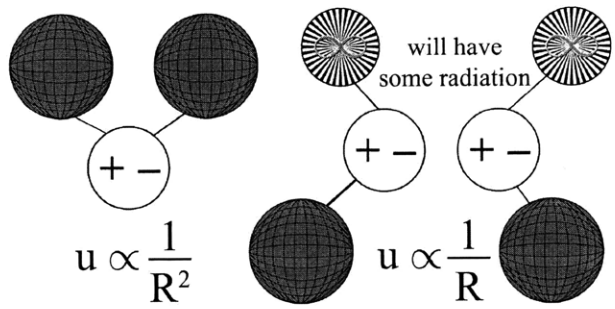
$$\lim_{\|\vec{x}\|\to\infty} \frac{\partial u(\vec{x})}{\partial n} \to O(\|\vec{x}\|^{-3})$$

And

$$\int_{Sphere_R} \frac{\partial u(\vec{x})}{\partial n} dS \to 0$$

This condition ensures all heat transferred.

3.3.5 Electrostatics



The above image is supposed to represent two scenarios, each senario has two conducting bodies.
In the first senario, on the left, there are two conductors that are treated as if there is a voltage source across them. There is a positive charge on one conductor, a negative charge on the other conductor, and the net charge should be zero. If the net charge on the two conductors is zero, then the integral of the normal electric field, $\frac{\partial u(\vec{x})}{\partial n}$) over a bounding sphere, one that contains both spherical conductors, should be zero. This is just a statement of the well-known Gauss's theorem in electrostatics. In particular

$$\int_{Sphere_R} \frac{\partial u(\vec{x})}{\partial n} dS = 0$$

for any $Sphere_R$ containing the two conductors. In order for this integral to be zero regardless as the radius of $Sphere_R$ approaches ∞,

$$\lim_{\|\vec{x}\| \to \infty} \frac{\partial u(\vec{x})}{\partial n}$$

must decay at least as fast as $O(\|\vec{x}\|^{-3})$. Therefore, $u(\vec{x})$ must decay like $O(\|\vec{x}\|^{-2})$.
For the senario on the right, the two spherical conductors are set to two different potentials with respect to the point at infinity. In such a case, it is unlikely that the sum of the charge on the two spheres will be zero,

$$\int_{Sphere_R} \frac{\partial u(\vec{x})}{\partial n} dS \neq 0$$

and therefore $u(\vec{x})$ should not be forced to decay any faster than $O(\|\vec{x}\|^{-2})$.

3.3.6 Potential Flow

Assumptions

Irrotational flow (velocity = potential gradient):

$$\mathbf{v}(\vec{x}) = \nabla u(\vec{x})$$

Air is incompressible (velocity divergence free):

$$\nabla \cdot \mathbf{v}(\vec{x}) = \nabla^2 u(\vec{x}) = 0.$$

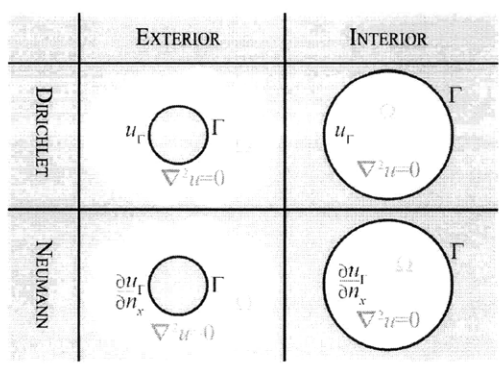
Nonpenetrating wing boundary condition:

$$\nabla u(\vec{x}) \cdot \mathbf{n}(\vec{x}) = \mathbf{v}_{wing}(\vec{x}) \cdot \mathbf{n}(\vec{x}).$$

What is the right radiation condition?

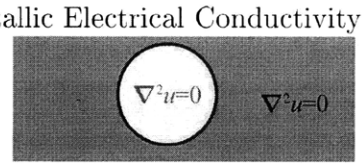
4 Formulations – Problem Types

4.1 Single Domain



4.2 Coupled Domain

Example: Bimetallic Electrical Conductivity



Potential and Electric Current Continuity:

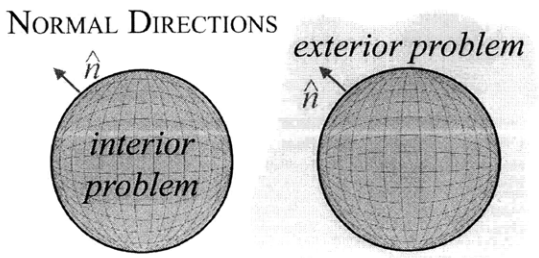
$$u(\vec{x}^+) = u(\vec{x}^-) \qquad \vec{x} \in \Gamma$$

$$\alpha^+ \frac{\partial u(\vec{x}^+)}{\partial n_{\vec{x}}} = \alpha^- \frac{\partial u(\vec{x}^-)}{\partial n_{\vec{x}}} \qquad \vec{x} \in \Gamma$$

An example of a coupled domain problem would be a conductivity problem involving multiple materials. To determine the electrical conductivity between two terminals of an object made of multiple materials, one would determine the ratio of the voltage across the object's terminals and the current flowing through the object. Electrical current density in an ideal linear conductor is a vector quantity given by the gradient of the potential, ∇u, scaled by a factor known as the conductivity of the material. In an ideal linear conductor there is no accumulation of charge at any interior point, implying that the current density has zero divergence. Therefore, the potential in an ideal linear conductor satifies Laplace's equation, $\nabla^2 u = 0$. If an object is made of multiple materials with different electrical conductivities, then the boundary between materials satisfies interface conditions. At the boundary between materials, both the potential and the current density in the surface-normal direction are continuous. Since the conductivities of the two materials are different, continuity of the current density implies a jump in the gradient of the potential across the material boundary.

4.3 Normals

Normals usually point from Interior to Exterior.



Typically, the surface normal is assumed to point in the direction from the interior domain to the exterior domain. There are many situations where this typical practice is confusing or ambigious, so it is often necessary to be explicit about the direction of the normal.

5 Surface Density Integrals

5.1 Monopole & Dipole

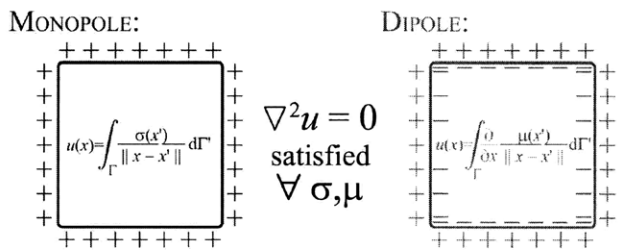
Potential due to a monopole density (σ):

$$u(\vec{x}) = \int_\Gamma \frac{1}{\|\vec{x} - \vec{x}'\|} \sigma(\vec{x}') d\Gamma'$$

Potential due to a dipole density (μ):

$$u(\vec{x}) = \int_\Gamma \frac{\partial}{\partial n_{\vec{x}'}} \frac{1}{\|\vec{x} - \vec{x}'\|} \mu(\vec{x}') d\Gamma'$$

where the normal points out of the domain Ω bounded by Γ.



Monopole or dipole densities can be used to generate potentials that satisfy $\nabla^2 u(\vec{x}) = 0$ for all $\vec{x} \in \Omega$. The monopole and dipole potentials differ in the radiation condition they satisfy. If the surface, γ, is finite in extent, then in the limit as $\|\vec{x}\| \to \infty$, the monopole potential decays like $\|\vec{x}\|^{-1}$, and the dipole potential decays like $\|\vec{x}\|^{-1}$.
Either representation can be used to derive surface integral equations, but care must be used when evaluating the associated potentials when $\vec{x} \in \Gamma$.

5.2 Surface Potentials

The monopole potential is continuous as x passes through Γ, so

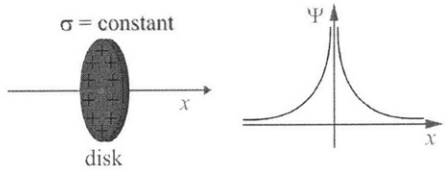
$$u_\Gamma(\vec{x}) = \int_\Gamma \frac{1}{\|\vec{x} - \vec{x}'\|} \sigma(\vec{x}') d\Gamma' \quad \vec{x} \in \Gamma$$

The dipole potential "jumps" as x passes through Γ, so the limit as $\vec{x} \to \Gamma$ of

$$u(\vec{x}) = \int_\Gamma \frac{\partial}{\partial n_{\vec{x}'}} \frac{1}{\|\vec{x} - \vec{x}'\|} \mu(\vec{x}') d\Gamma'$$

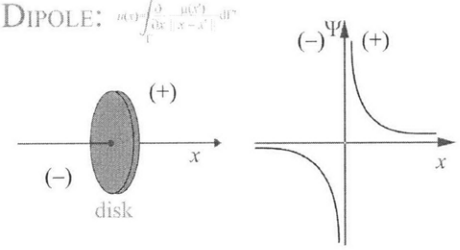
depends on how Γ is approached.

MONOPOLE: $u(x)=\int_\Gamma \frac{\sigma(x')}{\|x-x'\|}\,d\Gamma'$



Don't be put-off by the graph above. The monopole potential is continuous (it does not go off to infinity, as it may seem to in the above figure), but it is not continuously differentiable, there will be a discontinuity in the derivative at x_0.

DIPOLE: $u(x)=\int_\Gamma \frac{\partial}{\partial x}\frac{\mu(x')}{\|x-x'\|}\,d\Gamma'$



5.2.1 Principle Value Integral

If $f(y)$ is singular for some $y = x_0$, where $x_0 \in \Gamma$, then the principle value integral is

$$\int_\Gamma^{PV} f(\vec{y})d\Gamma \equiv \lim_{\epsilon\to 0}\int_{\Gamma - B(x_0,\epsilon)\cap\Gamma} f(\vec{y})d\Gamma$$

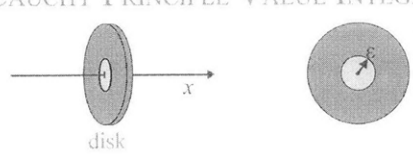
when $B(x_0,\epsilon)$ is the ϵ radius ball about x_0.

The P.V. is a special kind of limit

Limit of deleting and ever shrinking portion of the integration domain.

NOT EQUIVALENT TO limiting processes on f!

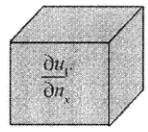
CAUCHY PRINCIPLE VALUE INTEGRAL



$\frac{\partial}{\partial n}$

5.2.2 Monopole Derivative (MD)

Consider a cube geometry:



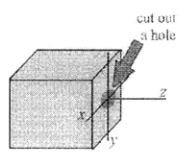
$$\frac{\partial u_\Gamma(\vec{x})}{\partial n_{\vec{x}}} = \lim_{\vec{x}\to\Gamma^+} \frac{\partial}{\partial n_{\vec{x}}} \int_\Gamma \frac{1}{\|\vec{x}-\vec{x}'\|}\sigma(\vec{x}')d\Gamma' \qquad \vec{x}\in\Gamma$$

The plus (+) in Γ^+ indicates exterior approach.

In the above slide, we consider computing the normal derivative of the monopole potential just outside the boundary γ. As will be shown in the next few slides, the derivative can be represented as the sum of a principle value integral and an extra term.

5.2.3 MD Disk Removal

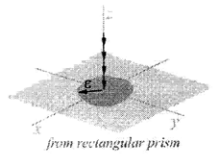
$$\frac{\partial u_\Gamma(\vec{x})}{\partial n_{\vec{x}}} = \lim_{\vec{x}\to\Gamma^+,\epsilon\to 0}\left[\frac{\partial}{\partial n_{\vec{x}}}\int_{\Gamma-B(x,\epsilon)} \frac{\sigma(\vec{x}')}{\|\vec{x}-\vec{x}'\|}d\Gamma' + \frac{\partial}{\partial n_{\vec{x}}}\int_{B(x,\epsilon)} \frac{\sigma(\vec{x}')}{\|\vec{x}-\vec{x}'\|}d\Gamma'\right]$$



Consider the entire side panel on the right of the cube in the above slide, and consider evaluating the normal derivative of the potential generated by a monopole distribution on the suface of the cube's right side. Specifically, assume that we wish to evaluate the derivative at a point $\vec{x}$ in the center of the green disk on the cube's right side. The matter is complicated by the fact that the integrand goes to infinity when $\vec{x}=\vec{x}'$. Thus, we need to break up the integral into two pieces. One piece is the entire panel minus the green disk, and the other piece is just the green disk.

5.2.4 MD Disk Picture

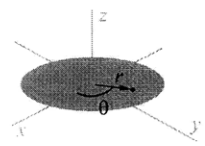
$$\frac{\partial u_\Gamma(\vec{x})}{\partial n_{\vec{x}}} = \lim_{\vec{x}\to\Gamma^+,\epsilon\to 0}\left[\frac{\partial}{\partial n_{\vec{x}}}\int_{\Gamma-B(x,\epsilon)} \frac{\sigma(\vec{x}')}{\|\vec{x}-\vec{x}'\|}d\Gamma' + \frac{\partial}{\partial n_{\vec{x}}}\int_{B(x,\epsilon)} \frac{\sigma(\vec{x}')}{\|\vec{x}-\vec{x}'\|}d\Gamma'\right]$$



The first integral is the Principle Value Integral and the second integral is the integral of just the disk.
Given that the disk was extracted from right the surface of a cube, the disk is flat, and the normal is in the z-axis direction.

5.2.5 MD Disk Eval

$$\begin{aligned}
\lim_{\vec{x}\to\Gamma^+} \frac{\partial}{\partial n_{\vec{x}}} \int_{B(x,\epsilon)} \frac{1}{\|\vec{x}-\vec{x}'\|}\sigma(\vec{x}')d\Gamma' \\
&\approx \lim_{z\to 0^+} \frac{\partial}{\partial z} \int_0^{2\pi}\int_0^{\epsilon} \frac{\sigma(\vec{x})}{\sqrt{r^2+z^2}} r dr d\theta \\
&= \lim_{z\to 0^+} \frac{\partial}{\partial z} 2\pi\sigma(\vec{x}) \left[\sqrt{\epsilon^2+z^2} - |z|\right] \\
&= -2\pi\sigma(\vec{x}).
\end{aligned}$$



Note 1 **Disk Evaluation Math**

For this problem, it is quite straightforward to see how one changes from cartesian to cylindrical coordinates. But, the algebra and calculus involved in solving this integral may not be as straightforward, herein is presented one method, broken-down into bite-size pieces:

$$\lim_{z\to 0^+} \frac{\partial}{\partial z} \int_0^{2\pi}\int_0^{\epsilon} \frac{\sigma(\vec{x})}{\sqrt{r^2+z^2}} r dr d\theta.$$

Use trigonometric substitution to solve the integral with respect to r. Substitute $r = z\tan\alpha$ and $dr = z\sec^2\alpha d\alpha$ and simplify to get the following integral:

$$= \lim_{z\to 0^+} \frac{\partial}{\partial z} \int_0^{2\pi}\int_0^{\alpha(\epsilon)} \frac{\sigma(\vec{x}) z\sin\alpha}{\cos^2\alpha} d\alpha d\theta.$$

This integral is easily solved using direct substitution of $u = \cos\alpha$:

$$= \lim_{z\to 0^+} \frac{\partial}{\partial z} \int_0^{2\pi}\int_0^{u[\alpha(\epsilon)]} -z\frac{\sigma(\vec{x})}{u^2} du d\theta = \lim_{z\to 0^+} \frac{\partial}{\partial z} \int_0^{2\pi} z\frac{\sigma(\vec{x})}{u}\Bigg|_0^{u[\alpha(\epsilon)]} d\theta.$$

Plug back in for $u = \cos\alpha = \frac{z}{\sqrt{r^2+z^2}}$

$$= \lim_{z\to 0^+} \frac{\partial}{\partial z} \int_0^{2\pi} \sigma(\vec{x})\sqrt{r^2+z^2}\Bigg|_0^{r=\epsilon} d\theta$$

$$= \lim_{z\to 0^+} \frac{\partial}{\partial z} \int_0^{2\pi} \sigma(\vec{x})\left[\sqrt{\epsilon^2+z^2} - \sqrt{z^2}\right] d\theta.$$

Integrating the last part is quite simple,

$$= \lim_{z\to 0^+} \frac{\partial}{\partial z}\sigma(\vec{x})\left(\sqrt{\epsilon^2+z^2}-\sqrt{z^2}\right)\theta\Big|_0^{2\pi} = 2\pi\sigma(\vec{x})\lim_{z\to 0^+}\frac{\partial}{\partial z}\left(\sqrt{\epsilon^2+z^2}-\sqrt{z^2}\right).$$

Finally, take the derivatives with respect to z, the normal,

$$= 2\pi\sigma(\vec{x})\lim_{z\to 0^+}\left(\frac{z}{\sqrt{\epsilon^2+z^2}}-\frac{z}{\sqrt{z^2}}\right).$$

It can now be seen that, since $\lim_{z\to 0^+}\frac{z}{\sqrt{\epsilon^2+z^2}} = 0$ and $\lim_{z\to 0^+}\frac{z}{\sqrt{z^2}} = \text{sign } z$ that

$$\lim_{z\to 0^+}\frac{\partial}{\partial z}\int_0^{2\pi}\int_0^{\epsilon}\frac{\sigma(\vec{x})}{\sqrt{r^2+z^2}}rdrd\theta = -2\pi\sigma(\vec{x}).$$

5.2.6 MD Final

$$\begin{aligned}
\frac{\partial u_\Gamma(\vec{x})}{\partial n_{\vec{x}}} &= \lim_{\vec{x}\to\Gamma^+}\frac{\partial}{\partial n_{\vec{x}}}\int_\Gamma \frac{1}{\|\vec{x}-\vec{x}'\|}\sigma(\vec{x}')d\Gamma' \\
&= \lim_{\vec{x}\to\Gamma^+}\lim_{\epsilon\to 0}\left[\frac{\partial}{\partial n_{\vec{x}}}\int_{\Gamma-B(x,\epsilon)}\frac{\sigma(\vec{x}')}{\|\vec{x}-\vec{x}'\|}d\Gamma' + \frac{\partial}{\partial n_{\vec{x}}}\int_{B(x,\epsilon)}\frac{\sigma(\vec{x}')}{\|\vec{x}-\vec{x}'\|}d\Gamma'\right] \\
&= \int_\Gamma^{PV}\frac{\partial}{\partial n_{\vec{x}}}\frac{1}{\|\vec{x}-\vec{x}'\|}\sigma(\vec{x}')d\Gamma' - 2\pi\sigma(\vec{x}')
\end{aligned}$$

5.2.7 Dipole Potentials (DP)

If Γ is a flat surface

$$\int_\Gamma^{PV}\frac{\partial}{\partial n_{\vec{x}'}}\frac{1}{\|\vec{x}-\vec{x}'\|}d\Gamma' = 0 \qquad \vec{x}\in\Gamma.$$

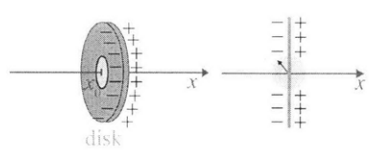
Why? Rewrite using explicit form of integrand

$$\int_\Gamma^{PV}\frac{(\vec{x}-\vec{x}')^T n_{\vec{x}'}}{(\|\vec{x}-\vec{x}'\|)^3}d\Gamma'$$

Integrand is zero when $\vec{x}-\vec{x}'$ orthogonal to surface normal

5.2.8 DP Flat Surface

Flat result applies locally on smooth surfaces.



5.2.9 DP General Surface

If Γ is a general surface

$$u_\Gamma(\vec{x}) = 2\pi\mu(\vec{x}) + \int_\Gamma^{PV} \frac{(\vec{x}-\vec{x}')^T n_{\vec{x}'}}{(\|\vec{x}-\vec{x}'\|)^3}\mu(\vec{x}')d\Gamma'$$

when $\vec{x}$ approaches Γ from outside Ω, and

$$u_\Gamma(\vec{x}) = -2\pi\mu(\vec{x}) + \int_\Gamma^{PV} \frac{(\vec{x}-\vec{x}')^T n_{\vec{x}'}}{(\|\vec{x}-\vec{x}'\|)^3}\mu(\vec{x}')d\Gamma'$$

when $\vec{x}$ approaches Γ from inside.

Note 1 ***Derivation of the Dipole Surface Potential***

The following derivation goes through the step-by-step process of deriving the dipole surface potential when $\vec{x}$ approaches Γ from inside.

$$\begin{aligned}
u(\vec{x}) &= \int_\Gamma \frac{\partial}{\partial n_{\vec{x}'}} \frac{1}{\|\vec{x}-\vec{x}'\|}\mu(\vec{x}')d\Gamma' \qquad \vec{x}\in\Gamma \\
&= \lim_{\epsilon\to 0}\int_{\Gamma - B(x_0,\epsilon)} \frac{\partial}{\partial n_{\vec{x}'}} \frac{1}{\|\vec{x}-\vec{x}'\|}\mu(\vec{x}')d\Gamma' + \lim_{\epsilon\to 0}\int_{B(x_0,\epsilon)} \frac{\partial}{\partial n_{\vec{x}'}} \frac{1}{\|\vec{x}-\vec{x}'\|}\mu(\vec{x}')d\Gamma' \\
&= \int_\Gamma^{PV} \frac{\partial}{\partial n_{\vec{x}'}} \frac{1}{\|\vec{x}-\vec{x}'\|}\mu(\vec{x}')d\Gamma' + \lim_{\epsilon\to 0}\int_{B(x_0,\epsilon)} \hat{n}_{x'}\cdot\nabla\frac{1}{\|\vec{x}-\vec{x}'\|}\mu(\vec{x}')d\Gamma' \\
&= \int_\Gamma^{PV} \frac{(\vec{x}-\vec{x}')^T n_{\vec{x}'}}{\|\vec{x}-\vec{x}'\|^3}\mu(\vec{x}')d\Gamma' + \lim_{r\to 0}\int_0^\pi\int_0^\pi \hat{n}_{x'}\cdot\nabla\left(\frac{1}{r}\right)\mu r^2\sin\phi\ d\phi\ d\theta \\
&= 0 + \lim_{r\to 0}\int_0^\pi\int_0^\pi \mu\frac{\partial}{\partial r}\left(\frac{1}{r}\right)r^2\sin\phi\ d\phi\ d\theta \\
&= \lim_{r\to 0}\int_0^\pi\int_0^\pi \mu\left(\frac{-1}{r^2}\right)r^2\sin\phi\ d\phi\ d\theta \\
&= -\int_0^\pi\int_0^\pi \mu\sin\phi\ d\phi\ d\theta \ = \ \left[\int_0^\pi \mu\cos\phi\ d\theta\right]_0^\pi \ = \ \int_0^\pi -2\mu\ d\theta \\
&= \Big[-2\mu\theta\Big]_0^\pi \ = \ -2\pi\mu
\end{aligned}$$

---

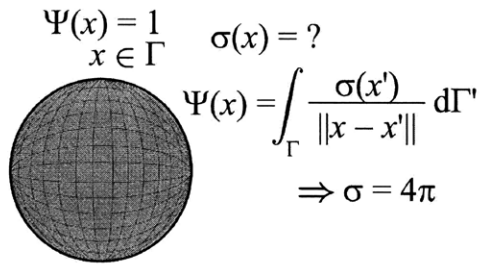
6 Ansatz Formulations

6.1 Dirichlet Problem

6.1.1 Monopole Potential

For an interior or exterior problem:

$$u_\Gamma(\vec{x}) = \int_\Gamma \frac{1}{\|\vec{x}-\vec{x}'\|}\sigma(\vec{x}')d\Gamma'$$



What radiation condition?

The point of the above slide is to show that when using monopole potentials to solve Dirichlet problems on a single domain bounded by γ, the equations are the same for either the interior or the exterior problem.

6.1.2 Dipole Potential

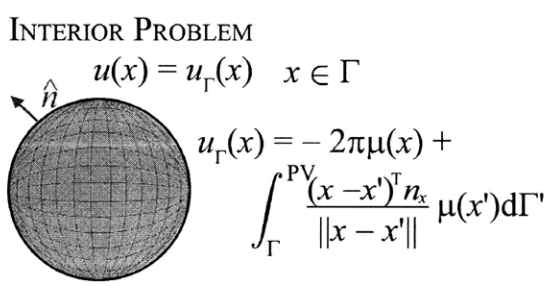
For an exterior problem:

$$u_\Gamma(\vec{x}) = 2\pi\mu(\vec{x}) + \int_\Gamma^{PV} \frac{(\vec{x} - \vec{x}')^T n_{\vec{x}'}}{(\|\vec{x} - \vec{x}'\|)^3} \mu(\vec{x}') d\Gamma'$$

and for an interior problem:

$$u_\Gamma(\vec{x}) = -2\pi\mu(\vec{x}) + \int_\Gamma^{PV} \frac{(\vec{x} - \vec{x}')^T n_{\vec{x}'}}{(\|\vec{x} - \vec{x}'\|)^3} \mu(\vec{x}') d\Gamma'$$

Normal points from interior to exterior.



Note that the radiation condition satisfied by the monopole potential is different than the radiation condition satisfied by the dipole potential, even when used to solve the same Dirichlet problem.

6.2 Neumann Problem

6.2.1 Monopole Potential (MP)

Derivative of the monopole potential "jumps" as x passes through Γ, so

$$\frac{\partial}{\partial n_{\vec{x}}} \int_\Gamma \frac{1}{\|\vec{x} - \vec{x}'\|} \sigma(\vec{x}') d\Gamma'$$

takes different values just inside and just outside Γ.

6.2.2 MP Int/Ext

For an exterior problem

$$\frac{\partial u_\Gamma(\vec{x})}{\partial n_{\vec{x}}} = -2\pi\sigma(\vec{x}') - \int_\Gamma^{PV} \frac{(\vec{x}-\vec{x}')^T n_{\vec{x}}}{(\|\vec{x}-\vec{x}'\|)^3}\sigma(\vec{x}')d\Gamma'$$

and for an interior problem

$$\frac{\partial u_\Gamma(\vec{x})}{\partial n_{\vec{x}}} = +2\pi\sigma(\vec{x}') - \int_\Gamma^{PV} \frac{(\vec{x}-\vec{x}')^T n_{\vec{x}}}{(\|\vec{x}-\vec{x}'\|)^3}\sigma(\vec{x}')d\Gamma'$$

Normal points from interior to exterior.

Note that the signs for the Neumann monopole potential integral equation are different than the signs for the integral equation in the Dirichlet Dipole case. The sign changes are due to the location of the derivative evaluation. In the Neumann Monopole potential case, the derivative is taken with respect to $n_{\vec{x}}$ whereas for the Dirichlet Dipole potential case, the derivative is taken with respect to $n_{\vec{x}'}$.

6.3 Kinds of Equations

First Kind Equations

- Dirichlet Monopole potential integral equation.

Second Kind Equations

- Dirichlet Dipole potential integral equation.
- Neumann Monopole potential integral equation.

Dipole potential for Neumann?

7 Green's Theorem

Laplace's Equation Green's Function

$$\nabla^2 G(\vec{x}) = 4\pi\delta(\vec{x})$$

$\delta(\vec{x}) \equiv$ impulse in 3-D
Defined by its behavior in an integral

$$\int \delta(\vec{x}')f(\vec{x}')d\Omega' = f(0)$$

Not too hard to show

$$G(\vec{x}) = \frac{1}{\|\vec{x}\|}$$

Note 1 4π

Just as an aside, Green's Function may be defined using a different scaling variable depending upon which source one is using. Sometimes Laplace's equation will be written:

$$\nabla^2 G(\vec{x}) = \delta(\vec{x}).$$

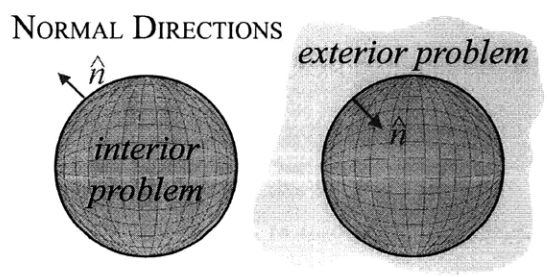
Where you can see that the value of 4π has been left off. This will simply mean that the Green's Function is now written:

$$G(\vec{x}) = \frac{4\pi}{\|\vec{x}\|}.$$

For our purposes, we will be using the notation in the above slide, and <u>not</u> the notation given in this note.

7.1 Normal Directions

A note here about normal directions is essential. In the above section, the "normal points from interior to exterior" whereas, in the image below, the normal points from inside the domain to outside the domain. How would this impact the solution?



When we go through Green's Theorem in the following section, remember that the normal always points "out" of Ω, as it does in the above figure.

7.2 Divergence Thm

The general Divergence Theorem:
For any sufficiently smooth $\vec{F}$

$$\int_\Omega \nabla \cdot \vec{F}(x) dV = \int_\Gamma \vec{F} \cdot \vec{n}_x dS$$

where Γ is the surface which encloses Ω.
Green's theorem follows from the **divergence theorem.**

7.3 Volume Theorem

If u satisfies Laplace's equation in Ω, then

$$4\pi u(\vec{x}) = \int_\Gamma \left[\frac{\frac{\partial u(\vec{x}')}{\partial n}}{\|\vec{x}-\vec{x}'\|} - \frac{\partial}{\partial n_{\vec{x}'}} \frac{1}{\|\vec{x}-\vec{x}'\|} u(\vec{x}') \right] d\Gamma' \quad \vec{x} \in \Omega$$

where the normal points out of Ω.

7.3.1 Surface

Using the Principle Value Integral:

$$2\pi u(\vec{x}) = \int_\Gamma \frac{\frac{\partial u(\vec{x}')}{\partial n}}{\|\vec{x}-\vec{x}'\|} d\Gamma' \; - \int_\Gamma^{PV} \frac{\partial}{\partial n_{\vec{x}'}} \frac{1}{\|\vec{x}-\vec{x}'\|} u(\vec{x}') \; d\Gamma' \qquad \vec{x} \in \Gamma$$

where the normal points out of Ω.

This is one of the cases where it is generally easier to define the normal as pointing out of Ω rather than having the normal point from the interior to the exterior.

7.3.2 Boundary Conditions

The boundary conditions, Dirichlet or Neumann, can be determined by using the surface form of Green's Theorem. For Dirichlet Problems, $u = u_\Gamma$ when $\vec{x} \in \Gamma$. So, put the known values for u into Green's Theorem for the surface, and put these known terms on the right hand side, leaving the unknown on the left hand side. Likewise, for Neumann problems, $\frac{\partial u}{\partial n} = \frac{\partial u_\Gamma}{\partial n}$ denotes that the derivative of u is known on the boundary when $\vec{x} \in \Gamma$. Again, put the known term on the right hand side, so that the unknown value, u is on the left hand side.

For Dirichlet Problems

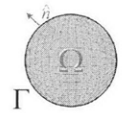
$$\int_\Gamma \frac{\frac{\partial u(\vec{x}')}{\partial n}}{\|\vec{x}-\vec{x}'\|} d\Gamma' = 2\pi u_\Gamma(\vec{x}) + \int_\Gamma^{PV} \frac{\partial}{\partial n_{\vec{x}'}} \frac{u_\Gamma(\vec{x}')}{\|\vec{x}-\vec{x}'\|} \; d\Gamma'$$

For Neuman Problems

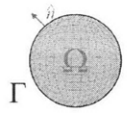
$$2\pi u(\vec{x}) + \int_\Gamma^{PV} \frac{\partial}{\partial n_{\vec{x}'}} \frac{1}{\|\vec{x}-\vec{x}'\|} u(\vec{x}') \; d\Gamma' = \int_\Gamma \frac{\frac{\partial u_\Gamma(\vec{x}')}{\partial n}}{\|\vec{x}-\vec{x}'\|} d\Gamma'$$

where normal points out of Ω (interior or exterior!)

7.4 Overview



$$\frac{\partial u(x')}{\partial n} \qquad \frac{\partial}{\partial n}\frac{1}{\|x-x'\|}$$



$$\frac{\partial u(x')}{\partial n} \qquad \frac{\partial}{\partial n}\frac{1}{\|x-x'\|}$$

8 Summary

Laplace Problems
- Exterior Radiation Condition

Potential Representations
- Monopole and Dipole potentials
- Principle Value

Ansatz and Green's Theorem
- Dirichlet and Neumann problems
- First and Second Kind Equations.

1.5 First and Second Kind Theory, part 2

Numerical Methods for PDEs

Boundary Element Methods, Lecture 5
First and Second Kind Theory, part 2

Notes by L. Proctor, C. Coelho and J. White

December 8, 2008

1 Outline

Interior Neumann
Use Nystrom to Solve
Look at 2-D Problems
Fredholm Alternative
Connection to Linear Algebra
First Kind Convergence Theory

2 Exterior Formulations

2.1 Dirichlet Problem

2.1.1 Monopole Potential

For an 3D exterior problem:

$$u_\Gamma(\vec{x}) = \int_\Gamma \frac{1}{\|\vec{x} - \vec{x}'\|} \sigma(\vec{x}') d\Gamma'$$

For an 2D exterior problem:

$$u_\Gamma(\vec{x}) = \int_\Gamma \log \|\vec{x} - \vec{x}'\| \sigma(\vec{x}') d\Gamma'$$

2.1.2 Dipole Potential

For a 3-D exterior problem:

$$u_\Gamma(\vec{x}) = 2\pi\mu(\vec{x}) + \int_\Gamma^{PV} \frac{(\vec{x} - \vec{x}')^T n_{\vec{x}'}}{(\|\vec{x} - \vec{x}'\|)^3} \mu(\vec{x}') d\Gamma'$$

For a 2-D exterior problem:

$$u_\Gamma(\vec{x}) = \pi\mu(\vec{x}) + \int_\Gamma^{PV} \frac{(\vec{x} - \vec{x}')^T n_{\vec{x}'}}{(\|\vec{x} - \vec{x}'\|)^2} \mu(\vec{x}') d\Gamma'$$

Normal points from interior to exterior.

2.2 Neumann Problem

2.2.1 Monopole

For an exterior 3D problem

$$\frac{\partial u_\Gamma(\vec{x})}{\partial n_{\vec{x}}} = -2\pi\sigma(\vec{x}') - \int_\Gamma^{PV} \frac{(\vec{x} - \vec{x}')^T n_{\vec{x}}}{(\|\vec{x} - \vec{x}'\|)^3} \sigma(\vec{x}') d\Gamma'$$

For an exterior 2D problem

$$\frac{\partial u_\Gamma(\vec{x})}{\partial n_{\vec{x}}} = -\pi\sigma(\vec{x}') - \int_\Gamma^{PV} \frac{(\vec{x}-\vec{x}')^T n_{\vec{x}}}{(\|\vec{x}-\vec{x}'\|)^2}\sigma(\vec{x}')d\Gamma'$$

Normal points from interior to exterior.

3 Interior Example

3.1 3D Case

3.1.1 Monopole Potential

Surface Potential

$$u_\Gamma(\vec{x}) = \int_\Gamma \frac{1}{\|\vec{x}-\vec{x}'\|}\sigma(\vec{x}')d\Gamma' \qquad \vec{x}\in\Gamma$$

Surface Normal Derivative

$$\frac{\partial u_\Gamma(\vec{x})}{\partial n_{\vec{x}}} = \frac{\partial}{\partial n_{\vec{x}}}\int_\Gamma \frac{1}{\|\vec{x}-\vec{x}'\|}\sigma(\vec{x}')d\Gamma' \qquad \vec{x}\in\Gamma$$

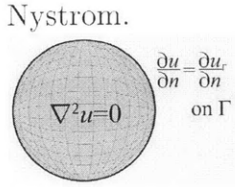
Normal points to exterior.

3.1.2 Interior Neumann

Monopole Potential Using the P.V. Integral

$$\frac{\partial u_\Gamma(\vec{x})}{\partial n_{\vec{x}}} = 2\pi\sigma(\vec{x}) + \int_\Gamma^{PV} \frac{\partial}{\partial n_{\vec{x}}}\frac{1}{\|\vec{x}-\vec{x}'\|}\sigma(\vec{x}')d\Gamma'$$

looks 2nd Kind Equation, Try Nystrom.



3.1.3 Nystrom Method

Set **quadrature points = collocation points**

$$\frac{\partial u_\Gamma(\vec{x}_1)}{\partial n_{\vec{x}}} = 2\pi\sigma_{n1} + \sum_{j=1}^{n} w_{1,j}\frac{\partial}{\partial n_{\vec{x}}}\frac{1}{\|\vec{x}_1-\vec{x}_j\|}\sigma_{nj}$$

$$\vdots$$

$$\frac{\partial u_\Gamma(\vec{x}_n)}{\partial n_{\vec{x}}} = 2\pi\sigma_{nn} + \sum_{j=1}^{n} w_{n,j} \frac{\partial}{\partial n_{\vec{x}}} \frac{1}{\|\vec{x}_n - \vec{x}_j\|} \sigma_{nj}$$

n equations in n unknowns

$j = i$ case (self-term)?

3.1.4 $i = j$

For the monopole 3-D Neumann Formulation,

$$G(\vec{x}, \vec{x}') = \frac{\partial}{\partial n_{\vec{x}}} \frac{1}{\|\vec{x} - \vec{x}'\|} = -\frac{(\vec{x} - \vec{x}')^T n_{\vec{x}}}{(\|\vec{x} - \vec{x}'\|)^3}$$

PROBLEM: $G(\vec{x}, \vec{x}')$ blows up as $\vec{x} \to \vec{x}'$.

3.2 2-D Case

Monopole Neumann Formulation

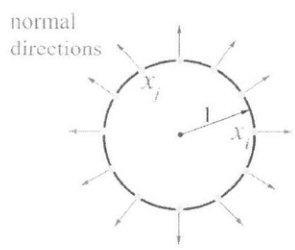
$$\frac{\partial u_\Gamma(\vec{x})}{\partial n_{\vec{x}}} = \pi\sigma(\vec{x}) + \int_\Gamma^{PV} \frac{\partial}{\partial n_{\vec{x}}} \log \|\vec{x} - \vec{x}'\| \sigma(\vec{x}') d\Gamma'$$

Simplifying the Green's function,

$$G(\vec{x}, \vec{x}') = \frac{\partial}{\partial n_{\vec{x}}} \log \|\vec{x} - \vec{x}'\| = -\frac{(\vec{x} - \vec{x}')^T n_{\vec{x}}}{(\|\vec{x} - \vec{x}'\|)^2}$$

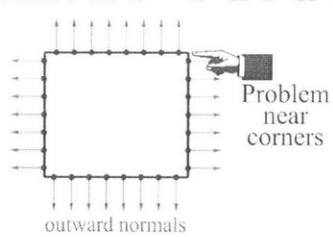
3.2.1 Smooth Γ

$G(\vec{x}, \vec{x}')$ finite as $\vec{x} \to \vec{x}'$ if Γ smooth.

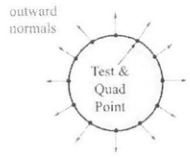


3.2.2 Nonsmooth Γ

$G(\vec{x}, \vec{x}')$ not finite as $\vec{x} \to \vec{x}'$ if x' is on a corner.



3.2.3 Disk Example



$$\frac{\partial u_\Gamma(x_i)}{\partial n_{x_i}} = \pi\sigma(x_i) - \frac{2\pi}{N}\sum \frac{\vec{n}_{x_i}\cdot(x_i - x_j)}{\|x_j - x_i\|^2}\sigma(x_j)$$

Note uniform quadrature weights on the circle.
Resulting matrix is singular! Why?

4 Second Kind Theorem

4.1 Theorem

Given

$$(I+K)\sigma = \Psi \quad \text{(Integral Eqn.)}$$
$$(I+K_n)\sigma_n = \Psi_n \quad \text{(Discretized Eqn.)}$$

AND

$$\|(I+K)^{-1}\| < C \quad \text{Unique solvability}$$

If

$$lim_{n\to\infty}\|(K-K_n)\| \to 0 \; and \; \|\Psi - \Psi_n\| \to 0$$

Then

$$\lim_{n\to\infty}\|\sigma - \sigma_n\| \to 0$$

4.1.1 Scaled Example

Define Scaled Variables

$$\Psi \equiv \frac{1}{\pi}\frac{\partial u_\Gamma(\vec{x})}{\partial n_{\vec{x}}}$$

$$K \equiv \frac{1}{\pi}\int_\Gamma^{PV} \frac{\partial}{\partial n_{\vec{x}}}\log\|\vec{x} - \vec{x}'\|\sigma(\vec{x}')d\Gamma'$$

The 2-D Neumann problem becomes

$$(I+K)\sigma = \Psi$$

4.1.2 Key Property

Main assumption of second kind theory:

$$(I+K)^{-1} \text{ is bounded.}$$

Is $(I+K)^{-1}$ bounded for Interior Neumann Problem?

4.2 Linear Algebra

Given $Ax = b$, $\quad A \in \Re^{n\times n}, \quad x, b \in \Re^n$

A^{-1} exists and is bounded iff

$Ay = 0$ implies $y = 0$ (no null space)

If $Ay = 0$ for $y \neq 0$ then either

$Ax = b$ has an infinite # of solutions

$Ax = b$ then $A(x + \alpha y) = b$

OR

$Ax = b$ does not have a solution

b is not in the column space of A

4.3 3-D Null Space

Consider $\tilde{\sigma}$ defined by

$$u_\Gamma(\vec{x}) = 1 = \int_\Gamma \frac{1}{\|\vec{x} - \vec{x}'\|}\tilde{\sigma}(\vec{x}')d\Gamma' \qquad \vec{x} \in \Gamma$$

Then

$$\frac{\partial u_\Gamma(\vec{x})}{\partial n_{\vec{x}}} = 0 = 2\pi\tilde{\sigma}(\vec{x}) + \int_\Gamma^{PV} \frac{\partial}{\partial n_{\vec{x}}} \frac{1}{\|\vec{x} - \vec{x}'\|}\tilde{\sigma}(\vec{x}')d\Gamma'$$

$\tilde{\sigma}$ is in the Null space of $I + K$

$$(I+K)^{-1} \text{ is not bounded!!}$$

4.4 Fredholm Alternative

General Theorem

For $I + K$ either

$(I+K)\sigma = \Psi$ has an infinite # of solutions

OR

$(I+K)\sigma = \Psi$ has no solution

4.4.1 2D Example

Scaled Equations:

$$\frac{1}{\pi}\frac{\partial u_\Gamma(\vec{x})}{\partial n_{\vec{x}}} = \sigma(\vec{x}) + \frac{1}{\pi}\int_\Gamma^{PV} \frac{\partial}{\partial n_{\vec{x}}} \log\|\vec{x}-\vec{x}'\|\sigma(\vec{x}')d\Gamma'$$

For a solution to exist

$$\int_\Gamma \frac{\partial u_\Gamma(\vec{x})}{\partial n_{\vec{x}}} d\Gamma = 0$$

2D Neumann Second Kind Integral equation
has a one-dimensional Null space.

4.4.2 Fixes

Add a point constraint
Fix u at some point
Force σ orthogonal to null space
Need the null space
May need to solve 1st kind equation
Use SVD to solve singular system
Can be computationally expensive

5 $1st$ Kind Convergence

Three-dimensional Laplace's equation

- Unknowns might be physically meaningful.

$$u(x) = \int \frac{1}{\|\vec{x}-\vec{x}'\|} \underbrace{\sigma(\vec{x}')}_{\substack{\text{charge}\\ \text{density}}} dS'$$

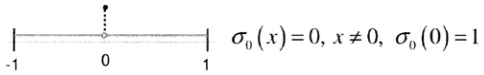
- Might match boundary conditions
  - Dirichlet and $\frac{1}{R}$ radiation condition

5.1 Nonsingular Green's Function

Denote the integral operator as K

$$K\sigma \equiv \int_{-1}^{1} |x-x'|\sigma(x')dS' \Rightarrow K\sigma = \Psi$$

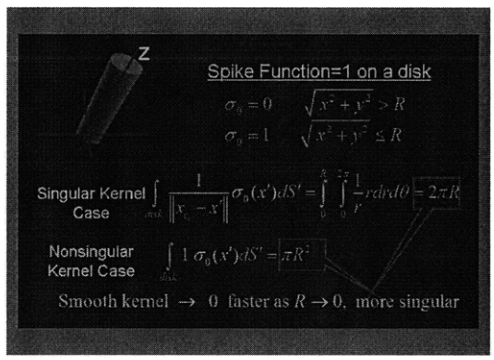
The integral operator is **singular** : K **has a null space**



$$K\sigma_0 = \int_{-1}^{1} |x - x'|\sigma_0(x')dS' = 0 \tag{2}$$

$$\text{If} \quad K\sigma^a = \Psi \quad \text{then} \quad K(\sigma^a + \sigma_0) = \Psi$$

5.2 The Singular Kernel



5.3 Convergence Analysis

Partial Differential Equation form:

$\nabla^2 u = f$ in Ω $\quad$ Ω is the volume domain

$u = 0$ on Γ $\quad$ Γ is the problem surface

"Nearly" Equivalent weak form

$$\underbrace{\int_\Omega \nabla u \nabla v dx}_{a(u,v)} = \underbrace{\int_\Omega f v dx}_{l(v)} \qquad \forall v \in H^1(\Omega)$$

Introduced an abstract notation for the equation, u must satisfy:

$$a(u, v) = l(v) \qquad \forall\ v \in H^1(\Omega)$$

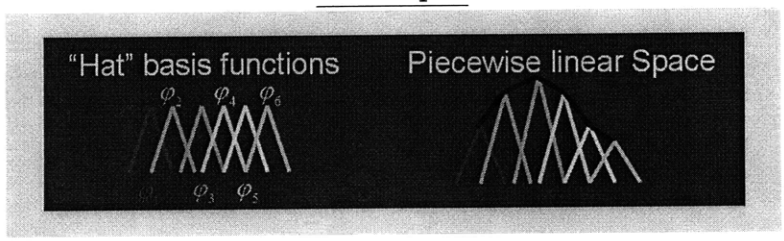
Introduce an approximate solution $u^n = \sum_{i=1}^{n} \alpha_i \varphi_i$

$\Rightarrow u^n$ is a weighted sum of basis functions

The basis functions define a space

$$X_n = \left\{ v \in X_n | v = \sum_{i=1}^{n} \beta_i \varphi_i \quad \text{for some } \beta_i' \text{s} \right\}$$

Example



Key Idea

$a(u,u)$ defines a norm on $H_0^1(\Omega)$ $\;a(u,u) \equiv |||u|||$

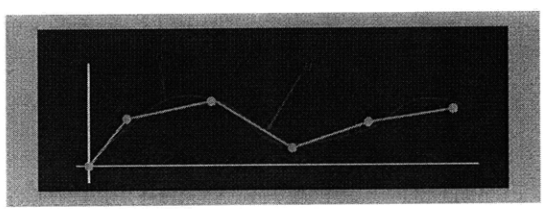
u is restricted to be 0 at 0 & 1!

Using the norm properties, it is possible to show

$$\text{If } a(u^n, \varphi_i) = l(\varphi_i) \qquad \forall \varphi_i \in \{\varphi_1, \varphi_2, \ldots, \varphi_n\}$$

$$\text{Then } \underbrace{|||u - u^n|||}_{\substack{\text{Solution}\\ \text{Error}}} = \min_{w_n \in X_n} \underbrace{|||u - w^n|||}_{\substack{\text{Projection}\\ \text{Error}}}$$

5.3.1 Optimality Result



How well can you fit the exact solution with a member of X_n?

You must measure the error in the $|||\;|||$ norm

5.3.2 Sobolov Space

"Weak" Form for the integral equation

$$\underbrace{\iint_\Gamma v(x) \frac{1}{\|x - x'\|} \sigma(x') dS' dS}_{a(\sigma, v)} = \underbrace{\int_\Gamma v(x) \Psi(x) dS}_{l(v)} \quad \forall v \in \bar{H}(\Gamma)$$

The difficulty is defining $\bar{H}(\Gamma)$ with right properties

Must exclude $\sigma(x)$'s where $\displaystyle\int \frac{1}{\|x - x'\|} \sigma(x') dS' = 0$

$\bar{H}(\Gamma)$ is a fractional Sobolev Space

We won't say more about this!

5.3.3 Use FEM Key Idea

$a(\sigma, \sigma)$ defines a norm on $\bar{H}(\Gamma)$ $\qquad a(\sigma, \sigma) = |||\sigma|||$

$$\sigma^n = \sum_{i=1}^{n} \alpha_i \quad \underbrace{\varphi_i(x)}_{\substack{\text{Basis} \\ \text{Functions}}} \qquad X_n = \left\{ v \in X_n | v = \sum_{i=1}^{n} \beta_i \varphi_i \quad \text{for some } \beta_i' \text{s} \right\}$$

5.3.4 FEM Idea Cont.

Using the norm properties, it is possible to show

If

$$a(u^n, \varphi_i) = l(\varphi_i) \qquad \forall \varphi_i \in \{\varphi_1, \varphi_2, \ldots, \varphi_n\}$$

$$\text{Then } \underbrace{\|u - u^n\|}_{\substack{\text{Solution} \\ \text{Error}}} = \min_{w_n \in X_n} \underbrace{\|u - w^n\|}_{\substack{\text{Projection} \\ \text{Error}}}$$

6 Summary

Interior Neumann

Use Nystrom to Solve

Look at 2-D Problems

Fredholm Alternative

Connection to Linear Algebra

First Kind Convergence Theory Mostly Waved hands.

1.6 Fast Algorithms for Integral Equation Methods

Numerical Methods for PDEs

Boundary Element Methods, Lecture 6
Fast Algorithms for Integral Equations

L. Proctor, S. De, K. Nabors, J. Phillips, B. Buchmann, & J. White

December 10, 2008

1 Outline

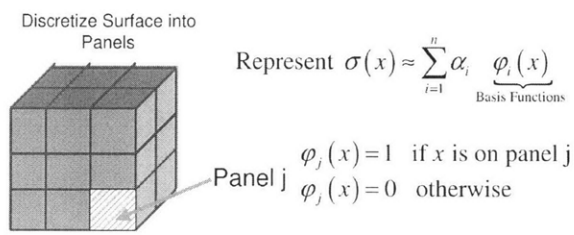
Reasons for Fast Solvers
Collocation System Reminder
Fast Solver General Approach
Using Iterative methods
Fast matrix-vector products
Fast Multipole Algorithms
Precorrected-FFT Algorithms

2 Background

2.1 Discretize Surface Into Panels

2.1.1 Piecewise Constant Basis

Integral Equation : $\Psi(x) = \int_{surface} \frac{\sigma(x')}{\|x-x'\|} dS'$



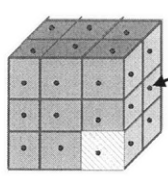
Represent $\sigma(x) \approx \sum_{i=1}^{n} \alpha_i \underbrace{\varphi_i(x)}_{\text{Basis Functions}}$

$\varphi_j(x) = 1$ if x is on panel j
$\varphi_j(x) = 0$ otherwise

2.1.2 Centroid Collocation

Put collocation points at panel centroids

$$\Psi(x_{c_i}) = \sum_{j=1}^{n} \alpha_j \underbrace{\int_{panel\ j} \frac{1}{\|x_{c_i} - x'\|} dS'}_{A_{i,j}}$$



$$\begin{bmatrix} A_{1,1} & \cdots & \cdots & A_{1,n} \\ \vdots & \ddots & & \vdots \\ \vdots & & \ddots & \vdots \\ A_{n,1} & \cdots & \cdots & A_{n,n} \end{bmatrix} \begin{bmatrix} \alpha_1 \\ \vdots \\ \vdots \\ \alpha_n \end{bmatrix} = \begin{bmatrix} \Psi(x_{c_1}) \\ \vdots \\ \vdots \\ \Psi(x_{c_n}) \end{bmatrix}$$

2.2 Dense Matrix

2.2.1 Resultant Dense Matrix

Matrix Entries Are Never Zero

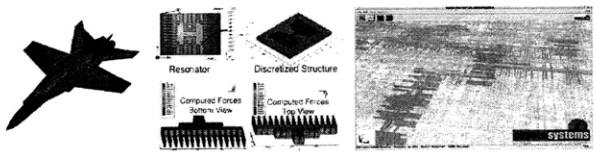
$$A_{i,j} = \int_{panel_j} \frac{1}{\|x - x'\|} dS'$$

Distant Elements Decay Slowly

$$\propto \frac{1}{\|x - x'\|}$$

Too Slow To Ignore.

2.2.2 Complicated Examples



Need More than 100,000 unknowns!!

Need 100 Gigabytes to Store Matrix.

2.2.3 Gaussian Elimination

```
For i = 1 to n-1 {          "For each Row"
  For j = i+1 to n  {       "For each Row below pivot"
      For k = i+1 to n  { "For each element beyond Pivot"
```

$$A_{jk} \Leftarrow A_{jk} - \frac{A_{ji}}{A_{ii}} A_{ik}$$

Multiplier

Pivot

```
      }
  }
}
```

Form n-1 reciprocals (pivots)

Form $\sum_{i=1}^{n-1}(n-i) = \frac{n^2}{2}$ multipliers

Perform $\sum_{i=1}^{n-1}(n-i)^2 = \frac{2}{3}n^3$ Multiply-adds

n^3 – **Too Expensive!**

3 Iterative Methods

3.1 Electrostatics Application

General Iterative "Algorithm"

0 : Guess at panel charges $\vec{\alpha}$

1 : Compute the centroid potentials from the charges $\mathbf{A}\vec{\alpha}$

2 : Compare the computed to known potentials
$\mathbf{R} = \mathbf{\Psi} - \mathbf{A}\vec{\alpha}$

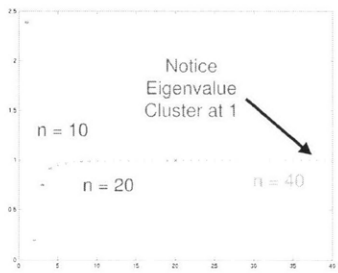
3 : Fix the panel charges, go to Step 1.
$\vec{\alpha}$

3.2 Conjugate Gradient (CG)

Conjugate Gradient (CG) Methods are iterative methods useful for solving systems of equations involving symmetric matrices $\mathbf{A} = \mathbf{A}^T$. The rate of iteration convergence for CG can be related to the ratio of the maximum to the minimum eigenvalue of A.

3.2.1 CG for 2nd Kind

Eigenvalues for 2nd Kind Integral Equation



3.2.2 CG for 2nd Kind Cont.

Conjugate-Gradient convergence rate

$$\|r^k\| \leq 2\left(\sqrt{\frac{\lambda_{\max}}{\lambda_{\min}}} - 1 \Big/ \sqrt{\frac{\lambda_{\max}}{\lambda_{\min}}} + 1\right)^k \|r^0\|$$

For discretized Second Kind equations

$$\frac{\lambda_{max}}{\lambda_{min}} \; is\ bounded\ independent\ of\ n$$

<u>Number of CG iterations independent of n!!</u>

3.2.3 Steps of CG

The kth step of the Conjugate Gradient Algorithm

compute Ap_k — For discretized Integral equations, A is dense

$$\alpha_k = \frac{(r^k)^T (Ap_k)}{(Ap_k)^T (Ap_k)}$$ — Determine optimal step size in kth search direction

$$x^{k+1} = x^k + \alpha_k p_k$$
$$r^{k+1} = r^k - \alpha_k Ap_k$$ — Update the solution and the residual

$$p_{k+1} = r^{k+1} - \frac{(Ar^{k+1})^T (Ap_k)}{(Ap_k)^T (Ap_k)} p_k$$ — Compute the new orthogonalized search direction

3.2.4 Cost of CG

Complexity of the Conjugate Gradient Method

compute Ap_k — Dense Matrix-vector product costs $O(n^2)$

$$\alpha_k = \frac{(r^k)^T (Ap_k)}{(Ap_k)^T (Ap_k)}$$ — Vector inner products, O(n)

$$x^{k+1} = x^k + \alpha_k p_k$$
$$r^{k+1} = r^k - \alpha_k Ap_k$$ — Vector Adds, O(n)

$$p_{k+1} = r^{k+1} - \frac{(Ar^{k+1})^T (Ap_k)}{(Ap_k)^T (Ap_k)} p_k$$ — Inner products, total cost O(n)

Algorithm is $O(n^2)$ for integral equations even though # of iterations, k, is small!

3.2.5 Accelerate CG?

Accelerate the Conjugate Gradient Method
Exactly compute $\mathbf{A}p_k$

Dense matrix-vector (M-V) product costs $O(n^2)$

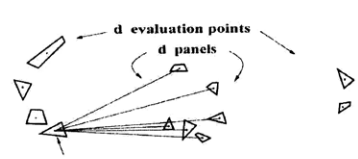
Approximately compute $\mathbf{A}p_k$

Reduces M-V product costs to $O(n)$ or $O(n \log n)$

Need a fast approximation for matrix-vector products

4 Fast Solvers

4.1 Direct Computation



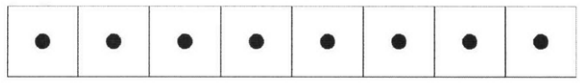
- Physical interpretation:
  $\mathbf{A}p = N$ "potentials" due to N charges.

- $O(N^2)$ if done naively

4.2 1D Strip of Charge in 3D Space

4.2.1 Simplification of the A Matrix

1-D Strip of Charge in 3-D Space



$$\begin{bmatrix} A_{11} & A_{12} & \cdots & A_{18} \\ A_{21} & A_{22} & \cdots & A_{28} \\ \vdots & \vdots & \ddots & \vdots \\ A_{81} & A_{82} & \cdots & A_{88} \end{bmatrix} \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \vdots \\ \alpha_8 \end{bmatrix} = \begin{bmatrix} \Psi_1 \\ \Psi_2 \\ \vdots \\ \Psi_8 \end{bmatrix}$$

What can one say about the **A** matrix?

4.2.2 Properties of A.

The A matrix is:

- Symmetric
  Panel i exerts the exact same charge on j that j exerts on i

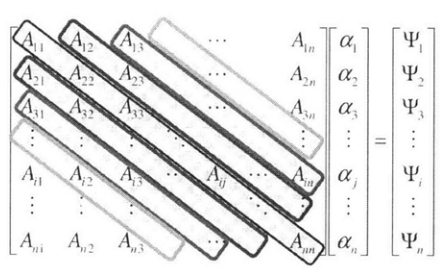
- All the Diagonal Values are the Same

$$A_{ii} = \int_{\text{panel}_i} \frac{1}{\|\vec{x}' - \vec{x}_{c_i}\|}$$

- Each Superdiagonal & Subdiagonal Element is Equal along Its Own Diagonal as Well

4.2.3 More Properties of A.

How many unique entry values are there in **A***?*

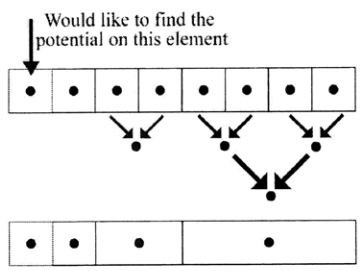


Note 1 ***Toeplitz Matrix***

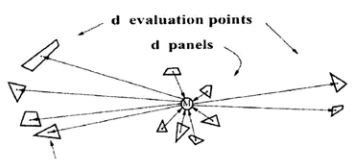
The above matrix is one of a class of matrices called Toeplitz Matrices. In a Toeplitz matrix, the matrix entries along any diagonal have the same value, but the different diagonals can have different values. For this reason, an $n \times n$ Toeplitz matrix has only $2n - 1$ distinct values. A particular important special case of Toeplitz matrices are Circulant Matrices. Circulant matrices are "periodized" Toeplitz matrices in that the first super-diagonal has the same values as the $n-1$ sub-diagonal, the second super-diagonal has the same values as the $n - 2$ sub-diagonal, etc. Circulatant matrices are diagonalized by the discrete Fourier transform, a property we will use in the section covering the Precorrected-FFT methods.

4.2.4 Geometric Simplification

Approximate (by grouping) the elements that are a "reasonable distance" away from the element which you are evaluating

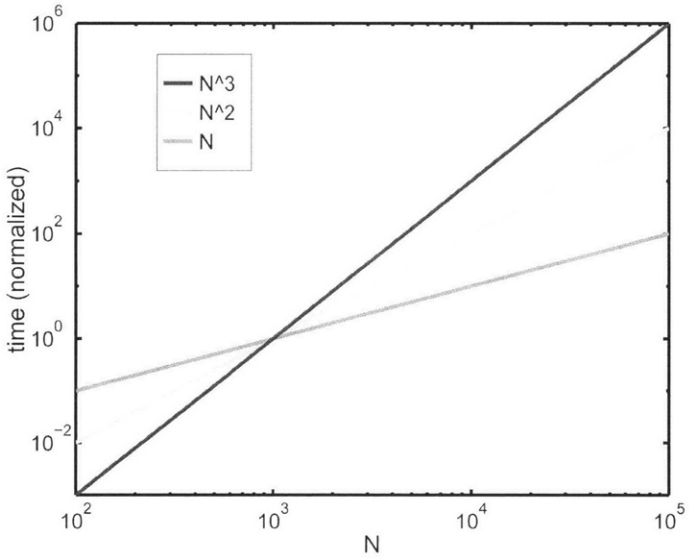


4.3 Fast Potential Concept



- Decompose potential into short- and long- range.
- Approximate long-range part of potential.
- Sum short-range part in normal manner
- Multilevel decomposition for "$O(N)$" algorithm

4.4 Computational Costs

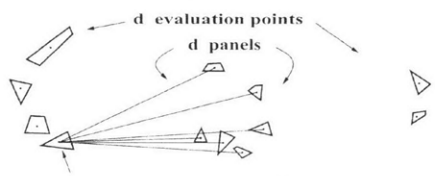


DEC 21164-333		
N	Gaussian Elim	"Fast" $O(N)$
	300 MFLOPS	30 MFLOPS
5e4	3 days,20GB	80sec, 130M
1e5	25 days,80GB	2.5min, 300M
5e5	8.8yrs,2TB	15min, 1.5GB

- Gaussian Elimination: $O(n^3)$ time, $O(n^2)$ memory
- Iterative with direct M-V: $O(n^2)$ time, $O(n^2)$ memory
- Fast Methods: $O(n)$ time, $O(n)$ memory

5 Multipole Algorithms

5.1 Direct Potential Evaluation

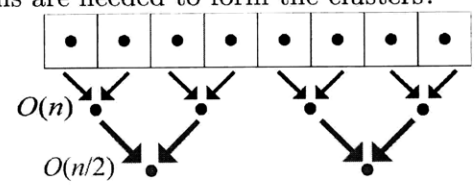


- Potential at point i: $\quad v_i(r_i, \phi_i, \theta_i) = \sum_{j=1}^{d} q_j \, P_{ij}.$
- Complete evaluation at d points costs d^2 operations.

5.2 Multipole Representation

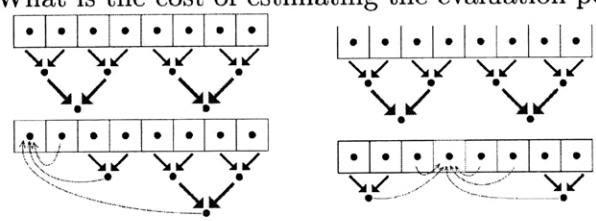
5.2.1 1D Strip in 3D Space

How many operations are needed to form the clusters?



The cost of forming clusters is, in general, $O(n + \frac{n}{2} + \frac{n}{4} + \frac{n}{8} + \ldots) \approx O(n)$

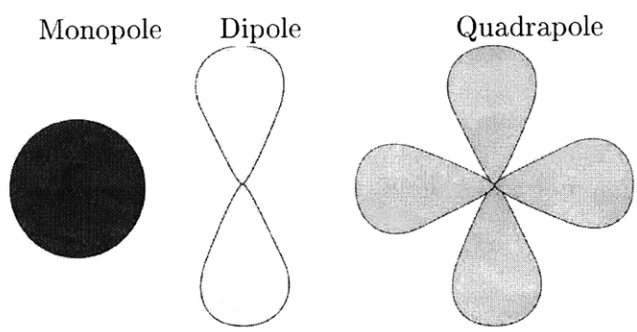
What is the cost of estimating the evaluation point potential?



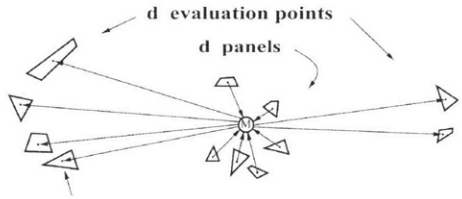
The cost of gathering clusters is $O(n \log n)$

5.2.2 Computational Example

A few multipoles (monopoles, dipoles, quadrapoles, etc) can accurately represent the potential due to a cluster of charges, with the accuracy improving with increasing distance from the cluster increases. For example, if one is very far from the cluster, the potential due to the cluster will be nearly identical to the potential of a point charge whose location is at the center of the cluster and whose value is the sum of the cluster charges. The accuracy of such a monopole representation can be improved by adding dipole, quadrapole and higher order multipoles. Note, however, that higher order multipoles generate potentials that depend on the multipole's orientation, and that must be considered.



5.2.3 General Case



- Approximate potential at point i:

$$v_i(r_i, \phi_i, \theta_i) \approx \sum_{j=0}^{order} \sum_{k=-j}^{j} \frac{M_j^k}{r_i^{j+1}} Y_j^k(\phi_i, \theta_i)$$

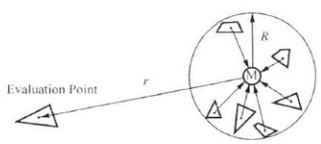
- Multipole coefficients function of panel charges:

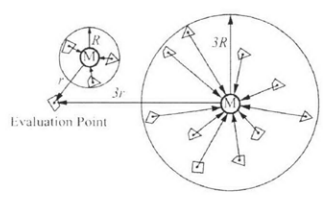
$$M_j^k \triangleq \sum_{i=1}^{d} \frac{q_i}{A_i} \int_{\text{panel } i} \rho^j Y_j^{-k}(\alpha, \beta) dA.$$

- Computing Multipole expansions costs order d operations.
- Each approximate potential evaluation costs order 1 operations.

d potential evaluation due to d panels in order d operations

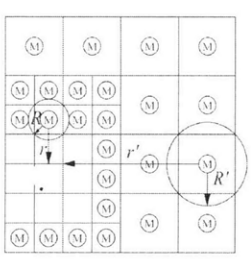
5.3 Error Scale Invariance





Error $\leq K \left(\frac{R}{r}\right)^{order+1}$ Error $\leq K \left(\frac{3R}{3r}\right)^{order+1}$

5.4 Multipole Algorithm Hierarchy



Hierarchy guarantees:

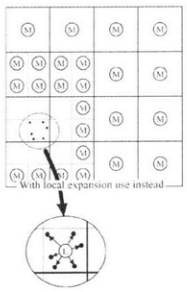
- Bounded error:

  Error $\leq K \left(\frac{R}{r}\right)^{order+1}$
  $\leq K \left(\frac{1}{2}\right)^{order+1}$

  $order = 2$ yields one percent accuracy.
- Order n ops for n potentials.

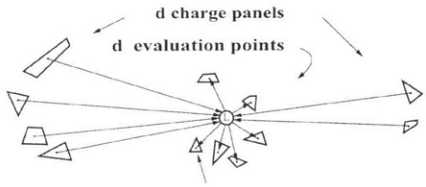
5.5 Local Representations

5.5.1 Cost Reduction



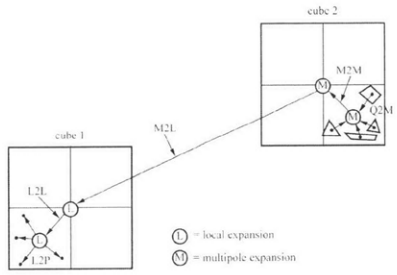
- Construct a local expansion to represent distant charge potentials.
- Evaluate a single local expansion, rather than many multipole expansions, at each evaluation point.

5.5.2 Clustered Evaluations



- Local expansion summarizes the influence of distant charge for clusters of evaluation points.
- Gives $O(n)$ potential evaluation when combined with coalescing of charge done by multipole expansions.
- Approximate potential at point i: $v_i(r_i, \phi_i, \theta_i) \approx \sum_{j=0}^{order} \sum_{k=-j}^{j} L_j^k Y_j^k(\phi_i, \theta_i) r_i^j$.

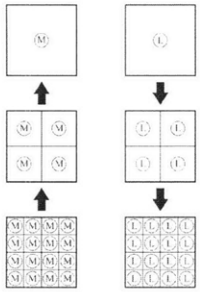
5.5.3 Summary of Operations



- Multipole and local expansions are built using complementary hierarchies.

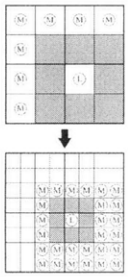
- Complete calculation consists of:
  1. Build multipoles (Upward Pass).
  2. Build locals (Downward Pass).
  3. Evaluate local expansions and nearby charge potential (Evaluation Pass).

5.5.4 Hierarchy Construction



- First build the multipole expansions moving upward from child to parent.
- Then build the local expansions by moving downward from parent to child.
- Computation has a tree structure.

5.5.5 Construction Details

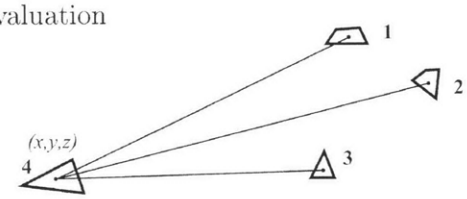


- Conversion of multipole expansions to local expansions.
- A child's local expansion is its parents local expansion plus conversions of multipole expansions in child's interaction range.

5.6 Adaptive Algorithm

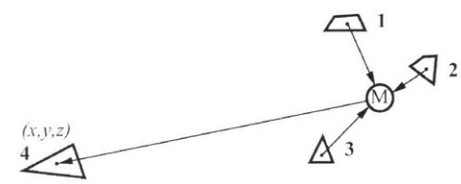
5.6.1 Multipole Inefficiency

Direct Evaluation



$$v_4(x, y, z) = q_1 P_{41} + q_2 P_{42} + q_3 P_{43}$$

Multipole Evaluation



$$v_4(x, y, z) \approx \bar{M}_0^0 \frac{1}{r} + \bar{M}_1^0 \frac{z}{r^3} - \bar{M}_1^1 \frac{x}{2r^3} - \tilde{M}_1^1 \frac{y}{2r^3}$$

Using Multipole MORE expensive than Direct.

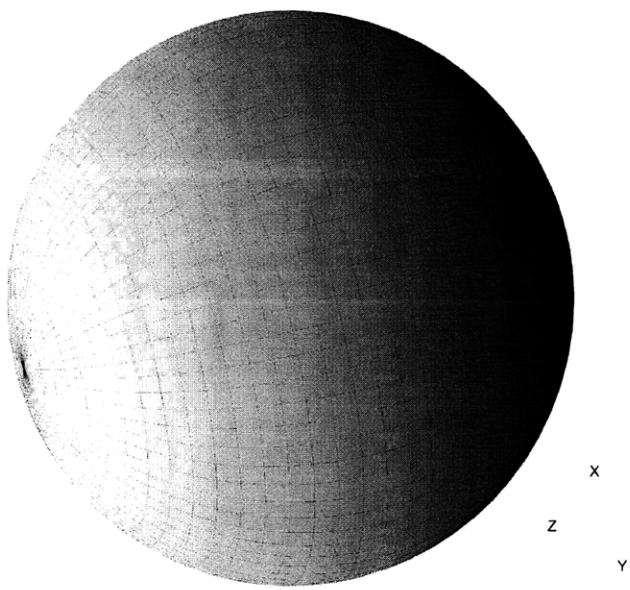
5.6.2 Simple Adaptive Scheme

If there are fewer panels than multipole coefficients, calculate the panels' influence directly.

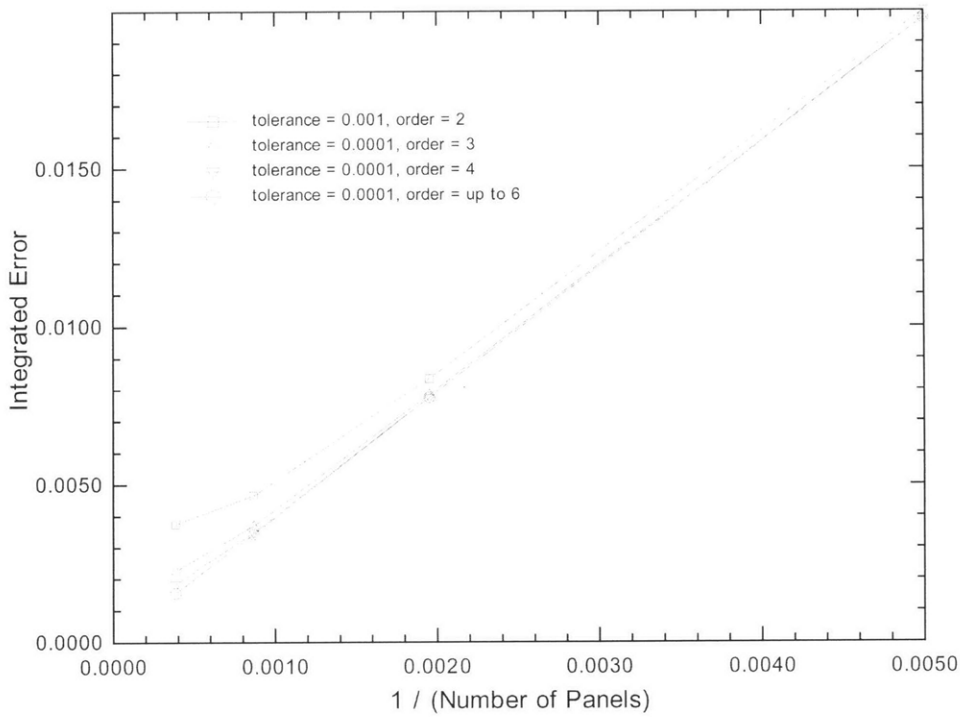
- Similarly, local expansions are not used if there are fewer evaluation points than local expansion coefficients.
- Retains $O(mn)$ complexity for nonuniform panel distributions.

5.7 Computational Examples

5.7.1 Sphere Potential Distribution

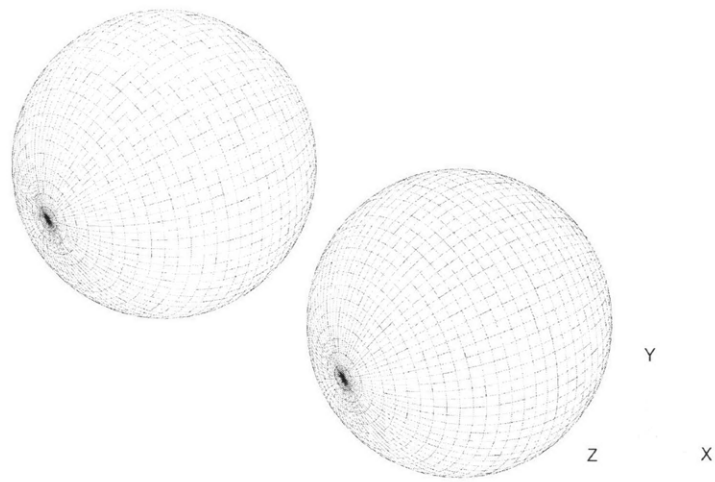


- Potential given by $\psi(x) = -\frac{x_3}{2\|x\|^3}$.
- Charge given by $\sigma(x) = \frac{-3}{8\pi} x_3$.

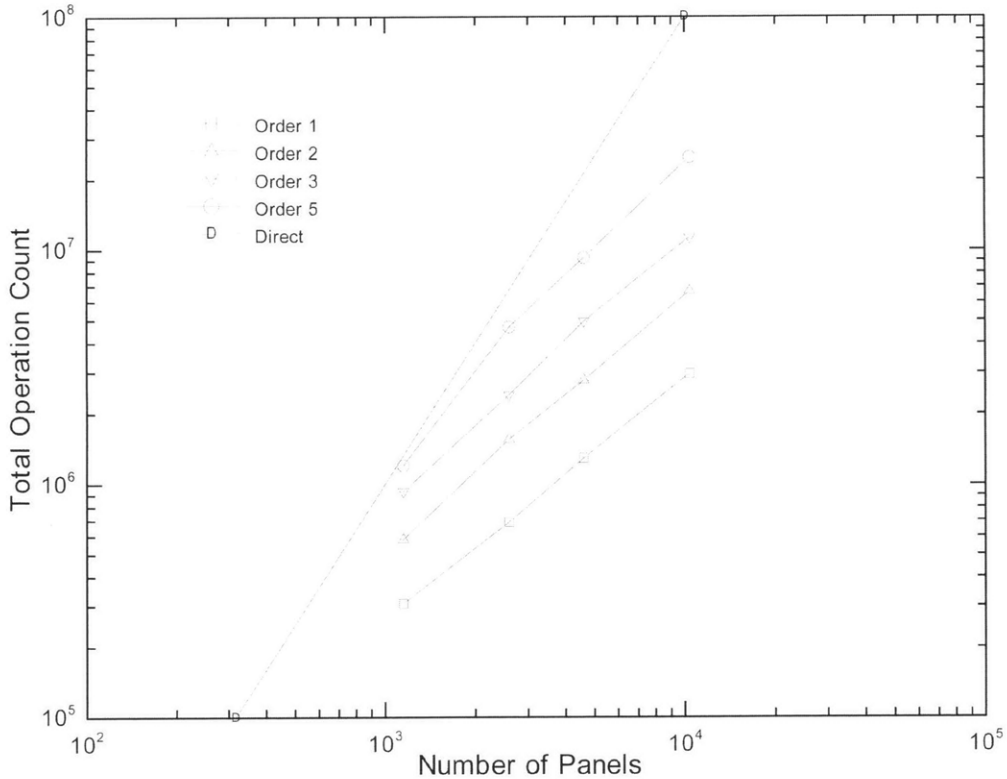


- Error should decay like $\frac{1}{n}$.
- Multipole approximations eventually interfere.
- Higher-order multipole expansions needed for higher accuracy.

5.7.2 Two Sphere Example



- Potential on each sphere: $\psi(x) = -\frac{x_3}{2\|x\|^3}$.
- Does not correspond to a simple physical problem.



- Direct matrix-vector product cost increases like n^2.
- Multipole matrix-vector product cost increases like n.
- The slope for the multipole algorithm depends on accuracy.
- For order 2 expansions, breakpoint is about $n = 400$.

5.8 Complexity Summary

For an integral equation discretized with n panels:

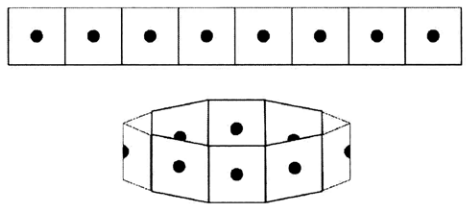
- Gaussian elimination: $O(n^3)$.
- Iterative Matrix Solution, direct M-V $O(n^2)$.
- Multipole accelerated Iterative method $O(n)$.

6 Precorrected-FFT

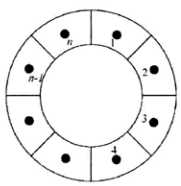
6.1 Introduction

Strip of Charge in Space

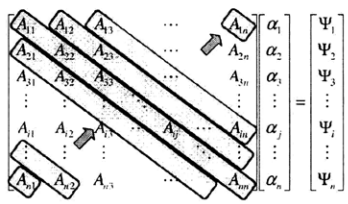
Bring the ends of the strip of charge together to form a ring.



Flattening the ring leads to the figure shown below on the left. Forming a ring from the strip of charges results in a system of equations with even more structure that the Toeplitz matrix system described above. The matrix in the ring case will be *circulent.*

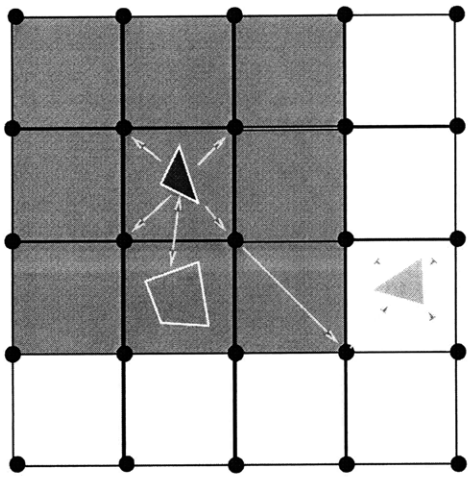


Produces a "**Circulant Matrix**"



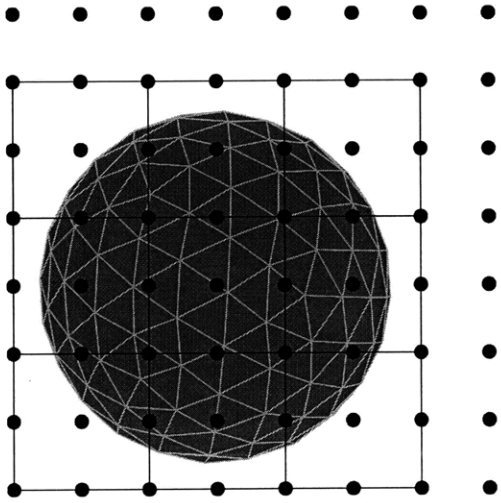
The above circulant matrix is the matrix representation of periodic convolution. This convolutional structure is partly due to the homogenity of the geometry, and partly because the Green's function is translation invariant. The Green's function for Laplace's equation, $G(x, x')$, is translationally invariant because $G(x, x')$ only depends on the difference, $x - x'$. As mentioned above, the discrete Fourier transform diagonalizes circulent matrices, and therefore circulent matrices can be inverted in $n \log n$ time using the fast Fourier transform.

6.2 Algorithm Outline



1. Project panel charges on grid
2. Calculate grid-charge potentials o
3. Interpolate grid potentials onto p
4. Local corrections
   [compute nearby interactions dire

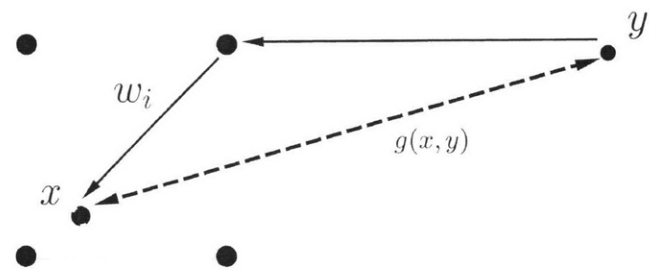
6.2.1 PFFT Grid Balances Costs



- Grid Selected So Direct Cost equals FFT Cost.
- Finer Discretizations Usually Yield Finer Grids.

6.3 Algorithm Analysis

6.3.1 Interpolation and Projection



Approximate potential Ψ at x due to charge at y by *interpolating* potential using points and weights x_i, w_i

Interpolate: potential at x due to unit charge at y

$$\Psi(x|y) \simeq \hat{\Psi}(x|y) = \sum w_i g(x_i, y)$$

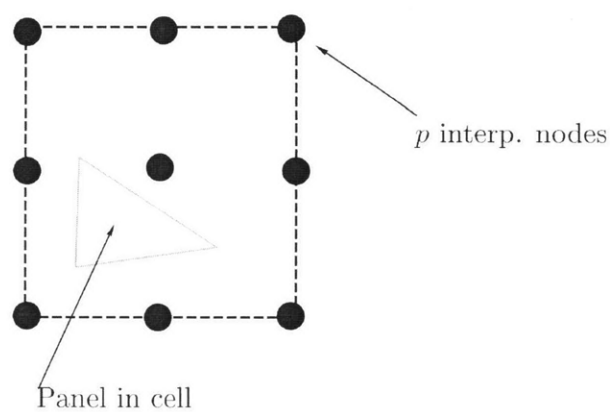
Anterpolate: potential at y due to unit charge at x

$$\Psi(y|x) \simeq \hat{\Psi}(y|x) = \sum w_i g(y, x_i)$$

So

$$\hat{\Psi}(y|x) = \hat{\Psi}(x|y)$$

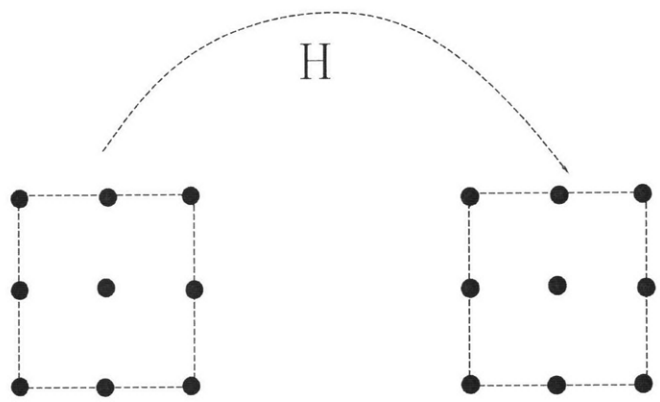
Same as representing charge at x with w_i and evaluating at y



Equivalent conditions:

- Approx Potential in cell due to charge at large R.
- Approx Potential at large R due to charge in cell.
- Cost is $O(N)$

6.3.2 Grid Potentials



- Let H be grid charge-potential mapping

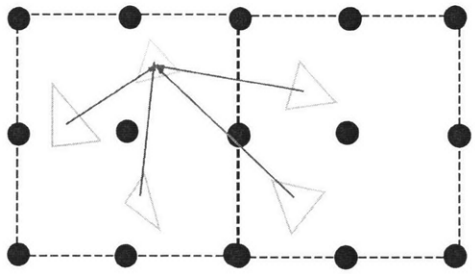
$$H : q_g \rightarrow \Psi_g$$

- H is Toeplitz
- Embed H in circulant matrix

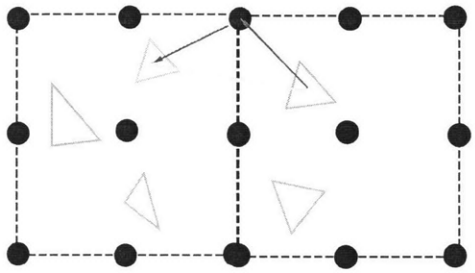
$$\begin{bmatrix} \psi_g \\ x \end{bmatrix} = \begin{bmatrix} H & X \\ X & X \end{bmatrix} \begin{bmatrix} q_g \\ 0 \end{bmatrix}$$

- Use FFT for matrix multiply
  Must Have Translation Invariance
- Cost $O(M \log_2 M)$, $M = \#$ cells

6.3.3 Nearby Interactions

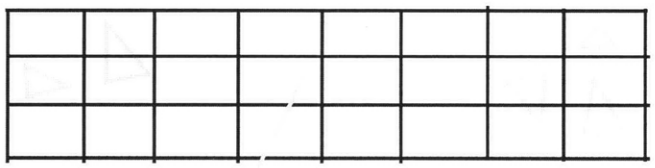


Direct interactions
Cost $O(N\lceil n_c \rceil)$
$\lceil n_c \rceil$ = max # panels /cell



Local corrections
Cost $O(1) - O(N) - O(Nn_I^2)$

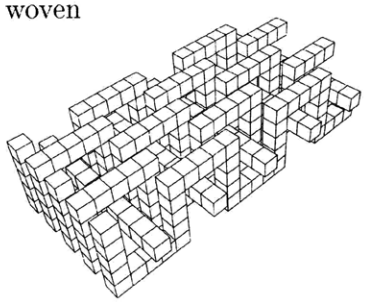
6.3.4 Inhomogeneity Problem



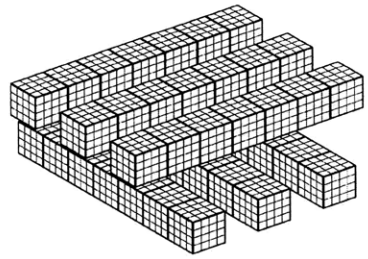
- Empty Grid due to FFT - Inefficient

6.4 Examples

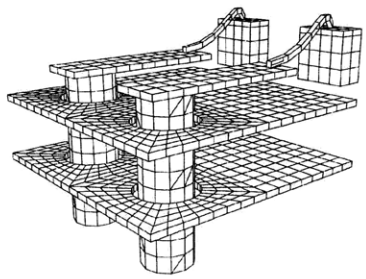
woven



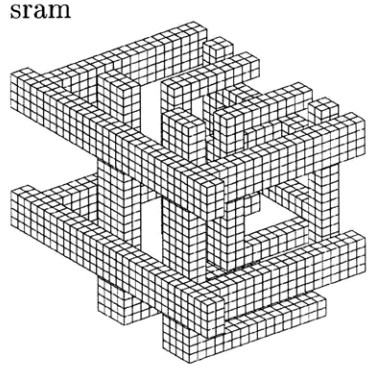
bus3x8



via



sram



6.5 PFFT vs. Multipole

- Comparisons: PFFT $p = 3$ to Multi $l = 2$

SLIDE 49

Example	CPU	Memory	Product	Error
via	0.61	0.37	0.23	0.18
woven5x5	0.45	0.48	0.22	0.09
cube	0.38	0.32	0.12	0.12
bus3x8	0.27	0.27	0.07	0.01
SRAM	0.39	0.43	0.17	0.07
mean	0.42	0.37	0.16	0.09

- Faster with 10× better accuracy !

6.6 PFFT vs. direct

6.6.1 Memory

Example		Memory Usage	
Name	Panels[conductors]	P/FFT	Direct
via	6120[4]	21 Mb	(286 Mb)
woven5x5	9360[10]	50 Mb	(668 Mb)
woven15	82080[30]	246 Mb	(50.2 Gb)
cube	126150[1]	225 Mb	(119 Gb)

6.6.2 Time

Example	CPU Usage		
Name	P/FFT	Dir. Iter.	Gauss. Elim.
via	1.1 min	(5.6 min)	(1.9 hrs)
woven5x5	5.2 min	(42 min)	(6.9 hrs)
woven15	1.7 hrs	(11.5 days)	(194 days)
cube	3.3 min	(8.4hrs)	(2.7 yrs)

7 Summary

Reasons for Fast Solvers
Collocation System Reminder
Fast Solver General Approach
Using Iterative methods
Fast matrix-vector products
Two Fast Methods
Fast Multipole - Multiresolution
Precorrected-FFT - Translation Invariance

Chapter 2

Foundations of Algorithms and Computational Techniques in Systems Biology

Engineering has always played a role in biology, specifically in the past couple of decades the field of computational biology has emerged and contributed greatly. Foundations of Algorithms and Computational Techniques in Systems Biology is a course that gives an overview of topics of interest to a computational biologist. The course covers protein modeling, modeling networks, and image processing. These are the top three areas in computational biology, and this course shows how one may use computational techniques to solve various problems with a biological application. This is very interesting to both the biologist and the computer scientist.

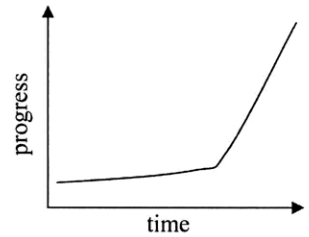
2.1 Motivation/Overview

MIT 6.581/BE.482
FOUNDATIONS OF ALGORITHMS AND COMPUTATIONAL TECHNIQUES IN SYSTEMS BIOLOGY
Spring 2006

7 February 2006
Tuesday

MOTIVATION/OVERVIEW

There is a disconnect between biology and computer science.
The biologist will pose the problem statement, but it may not be amenable for the computer scientist to solve it.
There is a need for scientists who posses the breadth of knowledge to marry the two realms.



PROBLEM ⟶ FORMULATION ⟶ SOLUTION

- assumptions
- set up

- algorithms
- computer techniques
- numerical methods

ecologies

populations

individuals
↕
organ systems
↕
organs
↕
tissues
↕
cellular ⇆ experiment
↕
molecular ⇆ physics

focus of this course

MOLECULAR LEVEL (atoms)	CELLULAR LEVEL (concentration of biomolecules)	IMAGING	
✓		✓	Fast Fourier Transform
✓	✓		Combinatorial Search
	✓		Model Reduction
✓	✓	✓	Singular Value Decomposition
✓			Multipole Algorithm
✓	✓	✓	Numerical Differentiation
✓	✓	✓	Optimization
✓	✓		Newton Methods

PHYSICAL, CHEMICAL, & BIOLOGICAL MODELING OF PROTEINS

Proteins:

- biological polymers of about 20 amino acids
  polymers are any kind of large molecules made of repeating identical or similar subunits called monomers
- "perfect" homogeneous, pure synthesis
- around 10k copies in a cell
- linear, unbranched chains of a unique sequence
- generally fold to characteristic structure with no additional information

sequence (1D)	*folding* →	structure (3D)	→	chemical functions	→	biological functions	→	network functions *
protein ↑ mRNA ↑ genome (DNA)		x-ray crystallography NMR		binding catalysis		synthesis/ degradation energy storage/ utilization gene expression development immune surveillance		control points – decision "robustness" time keepers oscillators important area of growth

Why Model?

- Understanding : model facilitates development of understanding reason for properties
  - mechanistic basis for function
  - disease
- Prediction
  - experiment planning
  - validate a model or select among models
- Design
  - perturbation : improve properties
  - intervention : repair

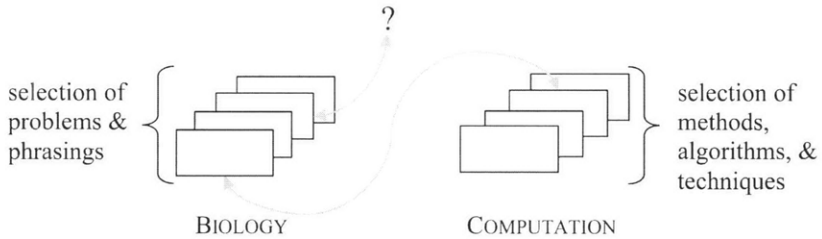
2

2.2 Models of Proteins

MIT 6.581/BE.482
FOUNDATIONS OF ALGORITHMS AND COMPUTATIONAL TECHNIQUES IN SYSTEMS BIOLOGY
Spring 2006

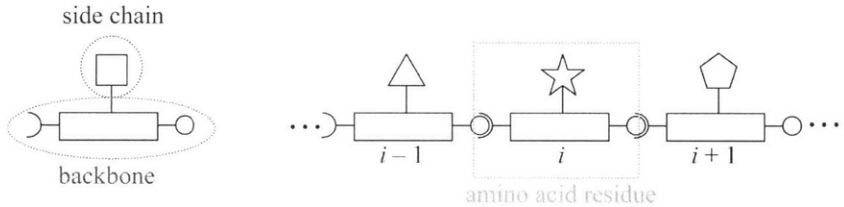
9 February 2006
Thursday

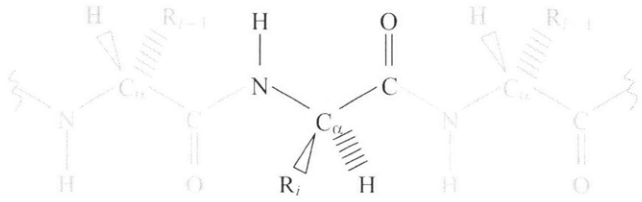
LECTURE 2 : MODELS OF PROTEINS



Fundamental role of models:
- Understanding
- Prediction
- Design

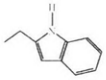
DNA – mRNA – Protein
(genome)



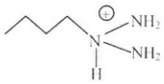


The "R_i" groups are chosen from the common 20 amino acid side chains – chemical diversity

(1) size: small – large
R_{Gly} : –H → R_{Trp}



(2) polarity: hydrophobic – polar – charged

R_{Leu} :  – R_{Asn} :  – R_{Arg} : 

(3) uniformity of character

(4) local backbone flexibility
Gly – Pro
(flexible) (rigidity)

Coordinate systems:

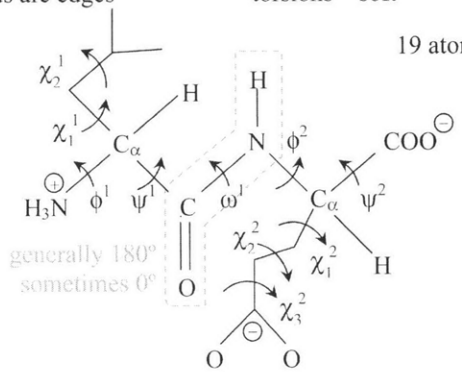
1) Absolute Cartesian Coordinates

$$\vec{\mathbf{X}} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ \vdots \\ x_{3N-1} \\ x_{3N} \end{bmatrix} \begin{matrix} \left.\begin{matrix} \\ \\ \\ \end{matrix}\right\} \text{cartesian coordinates of } 1^{\text{st}} \text{ atom} \\ \\ \\ \\ \\ \\ \left.\begin{matrix} \\ \\ \end{matrix}\right\} N^{\text{th}} \text{ atom} \end{matrix}$$

2) Relative Coordinates – Internal

Think of the molecules as graphs where

- atoms are vertices — bond lengths & bond angles – rigid
- bonds are edges — torsions – soft



19 atoms → 57 cartesian degrees of freedom

→ 10 torsional degrees of freedom

Desire : Mapping $\vec{\mathbf{X}}^{3N} \rightarrow \underset{\substack{\text{"energy"} \\ \text{scalar value}}}{E}\left(\vec{\mathbf{X}}^{3N}\right)$

$\Rightarrow$ Bias toward <u>mechanistic</u> basis for model

Chemistry – Physics (Quantum Mechanics)

Schrödinger Equation: $i\hbar \dfrac{\partial \Psi}{\partial t} = -\dfrac{\hbar^2}{2m}\nabla^2\Psi + \underbrace{V(x)}_{\text{nuclear \& electrons}}\Psi(x,t) \equiv \hat{H}\Psi(x,t)$

<u>Linus Pauling</u>

Observations:

- bond lengths, angles – fixed
- torsions – "soft" & sinusoidal
- atoms appear to have a fixed spherical size & approach to contact neighbors
- complementary electrostatics

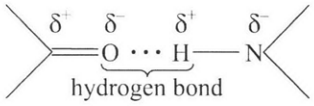
2

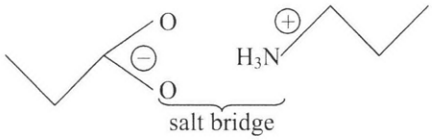
MIT 6.581/BE.482

FOUNDATIONS OF ALGORITHMS AND COMPUTATIONAL TECHNIQUES IN SYSTEMS BIOLOGY

Spring 2006

9 February 2006
Thursday



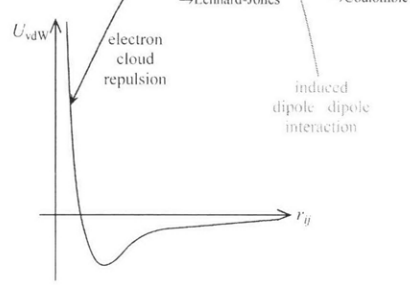


Molecular Mechanics Potential:

$$E\left(\vec{\mathbf{X}}^{3N}\right) = \underbrace{U_{\text{COVALENT}}}_{\text{bonded}} + \underbrace{U_{\text{NON-COVALENT}}}_{\text{through space}}$$

$$U_{\text{COVALENT}} = \sum_{i:\text{bonds}} \tfrac{1}{2} k_{b,i}\left(b_i - b_{o,i}\right)^2 + \sum_{i:\text{angles}} \tfrac{1}{2} k_{\theta,i}\left(\theta_i - \theta_{o,i}\right)^2 + \sum_{i:\text{impropers}} \tfrac{1}{2} k_{\Phi,i}\left(\Phi_i - \Phi_{o,i}\right)^2 + \sum_{i:\text{torsions}} \tfrac{1}{2} k_{\phi,i}\left[1 + \cos\left(n_i \phi_i - \delta_i\right)\right]$$

$$U_{\text{NON-COVALENT}} = \underbrace{\sum_{i>j}\left(\frac{B_{ij}}{r_{ij}^{12}} - \frac{C_{ij}}{r_{ij}^{6}}\right)}_{\substack{\text{van der Waals} \\ \rightarrow\text{Lennard-Jones}}} + \underbrace{\sum_{i>j} \frac{q_i q_j}{\varepsilon\, r_{ij}}}_{\substack{\text{Electrostatics} \\ \rightarrow\text{Coulombic}}}$$



2.3 Discrete Conformational Search

6.581 / BE 482

LECTURE 3: DISCRETE CONFORMATIONAL SEARCH

TUESDAY 14 FEB. 2006

REVIEW:

A.) COORDINATE REPRESENTATION – PROTEIN STRUCTURE

1.) CARTESIAN – ABSOLUTE

2.) INTERNAL – RELATIVE

$$\vec{X}^{3N} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_{3N-2} \\ x_{3N-1} \\ x_{3N} \end{bmatrix} \begin{matrix} \left.\vphantom{\begin{matrix}x\\x\\x\end{matrix}}\right\} 1^{st} \text{ atom} \\ \\ \left.\vphantom{\begin{matrix}x\\x\\x\end{matrix}}\right\} N^{th} \text{ atom} \end{matrix}$$

- bond lengths } "hard"
- bond angles
- torsional – "soft"

B.) ENERGY FUNCTION – PHYSICS BASED

- level of "atom"
- $\vec{X}^{3N} \rightarrow E(\vec{X}^{3N}) = \underbrace{U_{bonds} + U_{bond\ angles} + U_{improper\ dihedrals} + U_{torsions}}_{U_{covalent}\ O(N)} + \underbrace{U_{vdW} + U_{elec}}_{U_{non\text{-}covalent}\ O(N^2)}$

WHAT CAN WE DO WITH AN ENERGY FUNCTION?

* Fold a protein

- Find the final, folded shape
- Understand how a protein finds its final shape

- Average folding time for a protein: ~1ms – 1s typical
- Most detailed simulations: 1 cpu day ⇔ 1ns

  1ms ⇒ 10^6 CPU days ⇒ 2,740 years

Two Studies of Historical Note:

1) Levitt & Warshel Nature 253 694-698 (1975).

Simplified model of a small protein

- BPTI 58 residues
- Every center represents an amino acid
  - ran detailed atom-level simulations on pairs of amino acids to produce higher level model
- Accelerated simulation methodology

RESULT

Authors claim close to "native" structure

Subsequent analysis question this

2) Duan & Kollman Science 282: 740-744 (1998).

1 μs simulation of 36-residue protein (villin headpiece subdomain)

- model: all-atoms plus ~3,000 water molecules
- took ~4 months on a 256-processor Cray T3E

RESULT

Marginally stable state "close" to native state adopted at ~150ns, persisted briefly, then dissolved

FUNDAMENTAL PROBLEM

1. What result does the model produce?

   Interplay: Simulation Methodology ⇔ Model

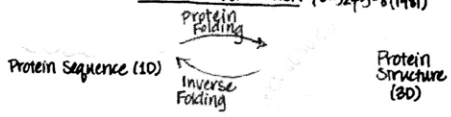
   Pathway ⇔ Final Structure

2. Can't relate problems to individual terms in the energy function

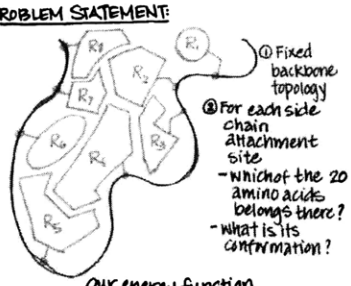
Two Visionary Statements:

C.O. Pabo Nature 301: 200 (1983)

E. Drexler Proc Natl Acad Sci USA 78: 5275-8 (1981)



PROBLEM STATEMENT:



① Fixed backbone topology

② For each side chain attachment site

- Which of the 20 amino acids belongs there?
- What is its conformation?

Our energy function

- adopts relaxed local (covalent) conformations
- vdW potential: efficient packing of space
- elec: complementary charge patterns

2 simplifications

1. fix all bond lengths & bond angles
2. discretize side chain torsions

The Dead-End Elimination Algorithm (DEE)

Desmet et. al. Nature 356: 539-42 (1992).

At each of N positions select from m_i choices for the amino acid and its conformation

Solve for $\vec{M} = \begin{bmatrix} M_1 \\ M_2 \\ M_3 \\ \vdots \\ M_N \end{bmatrix}$ that minimizes $E(\vec{X}^{3N})$ over all possible $\{\vec{M}\}$

Imagine $m_i = m \Rightarrow$ space is m^N

NOTE: OUR ENERGY FUNCTION HAS SIMPLE FORM

$$E_{\vec{M}} = \underbrace{\sum_{i=1}^{N} E^{self}_{M_i}}_{\text{terms between sc \& bb}} + \underbrace{\sum_{i=1}^{N-1} \sum_{j=i+1}^{N} E^{pair}_{M_i, M_j}}_{\text{sc-sc}} + \text{const.}$$

Looking at just one position

$E^{self}_{M_i^s} + \sum_{\substack{j=1 \\ j \neq i}}^{N} E^{pair}_{M_i^s, M_j}$ is the contribution if I choose m_i^s at position i

But I can bound the contribution

Lower Bound: $E^{self}_{M_i^s} + \sum_{\substack{j=1 \\ j \neq i}}^{N} \min_u E^{pair}_{M_i^s, m_j^u}$

Upper Bound $\max_u$

DEE Singles Criterion

If $E^{self}_{M_i^t} + \sum_{\substack{j=1 \\ j \neq i}}^{N} \min_u \left(E^{pair}_{M_i^t, m_j^u} \right) > E^{self}_{M_i^s} + \sum_{\substack{j=1 \\ j \neq i}}^{N} \max_u \left(E^{pair}_{M_i^s, m_j^u} \right)$

For some t ≠ s, then m_i^s can not be in the global optimum

This method is provably correct, but can't prove running time

Approach:

1. Eliminate iteratively singles until no more possible
2. Develop higher-order eliminations (pairs)
3. When all eliminations done, enumerate remaining space

Application:

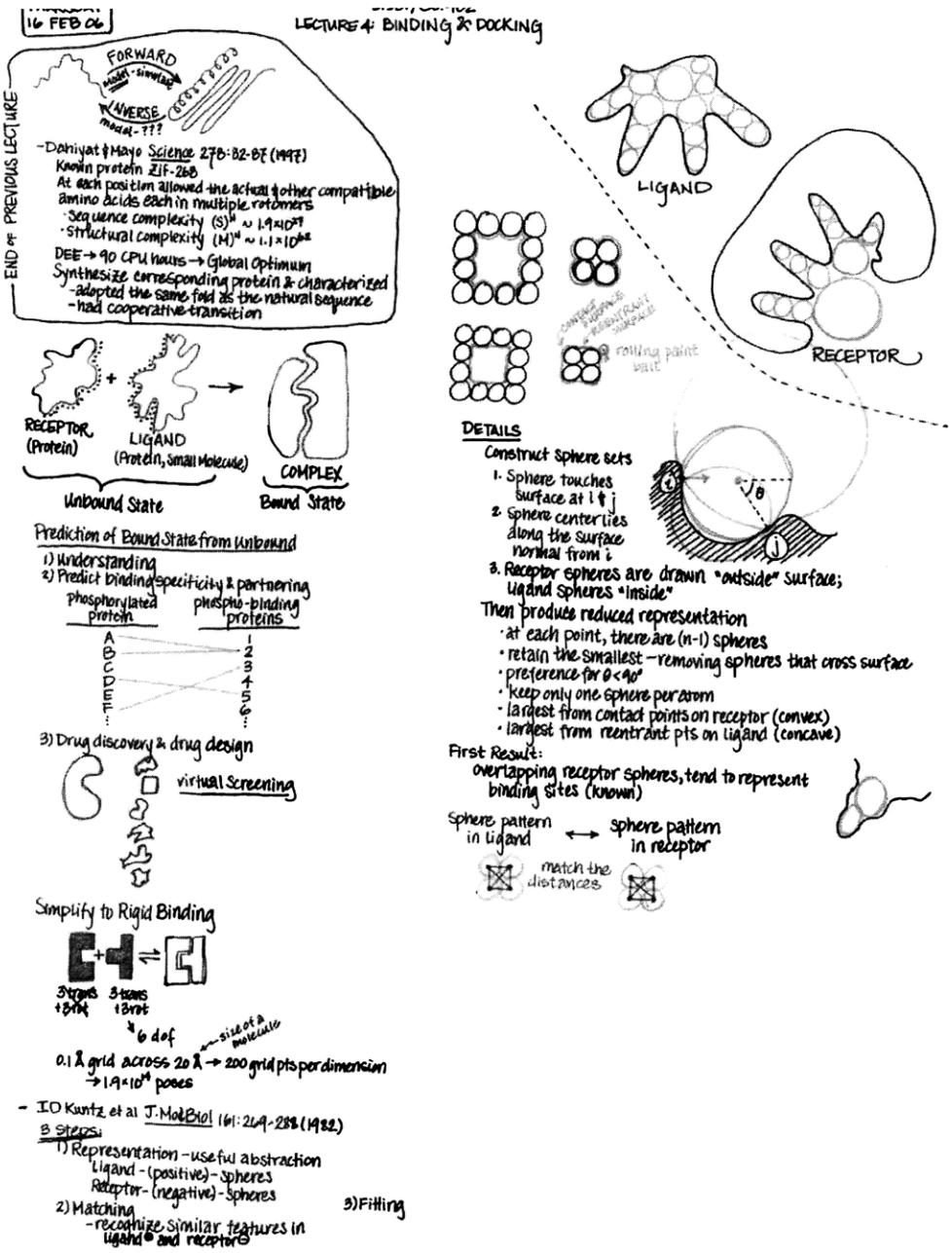
Insulin: 76 positions: 2.7×10^{76} conformations

Iterative approach (9 iterations)

$2.7 \times 10^{76} \longrightarrow 7200 \rightarrow$ search by enumeration

93% of buried positions "correct"

2.4 Binding and Docking



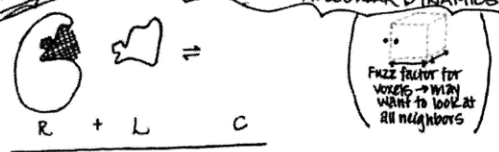
2.5 Binding and Docking - Molecular Dynamics Simulation

BINDING & DOCKING

6.581 / BE 482

LECTURE 5: BINDING & DOCKING
MOLECULAR DYNAMICS SIMULATION

THURSDAY
23 FEB 2006



GEOMETRIC HASHING Fischer et al J Mol Biol 248: 459-477 (1995)

BASIC STRATEGY

1.) Molecular Surface: Ligand & Receptor
2.) Identify "critical points" on each surface
- Centers of convex patches on ligand
- Centers of concave patches on receptor } & surface normals
3.) Create total orientations
- In each orientation, line up 2 critical points & their avg. surface normal from ligand with 2 more & avg. surface normal from receptor
- In each orientation, count number of other critical point coincidences
4.) Collect high scorers (based on geometry alone) & evaluate with energy function

TIMING: $2\binom{m}{2}\cdot\binom{n}{2} \sim m^2n^2$ trial orientations

$m \cdot n \rightarrow$ # of distances to measure in each orientation

$\sim m^3n^3$ Total $\rightarrow$ quite bad

EFFICIENT IMPLEMENTATION

1.) For each pair of critical points on ligand, transform into "standard orientation"
- mark positions of other critical points in space & store in hash table
  index by voxel of critical point $\rightarrow$ store the current ligand transformation
2.) For each pair of critical points on the receptor, carry out the complementary transformation
- look up each receptor critical point's voxel in hash table. Add a vote for each ligand transformation with a critical point in that voxel.
- tally votes for each ligand transformation
- pass high scorers on to energy function

TIMING: 1.) $\binom{m}{2}$ pairs of ligand points; $\mathcal{O}(m)$ operations on each $\Rightarrow m^3$

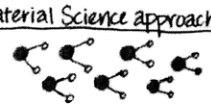
2.) $2\binom{n}{2}$ pairs of receptor points $\mathcal{O}(n)$ operations on each $\Rightarrow \sim n^3$

$\sim m^3 + n^3$ Total

MOLECULAR DYNAMICS

- Two users: Biologists & Mat Sci
- VMD, NAMD2
- Formulation of equations — Hamiltonian
- Discretization in time

Material Science approach



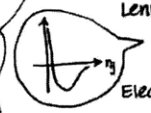
track thousands of molecules

Ask Questions like:
- Extract Diffusion Constants or flow rates, density
- Temperature, Pressure
- Ensemble Averages

} Extraction of Continuum Properties

Important Forces — Non-bonded

Lennard-Jones:



$$\sum_{i>j} \frac{B_{ij}}{r_{ij}^{12}} - \frac{C_{ij}}{r_{ij}^{6}}$$

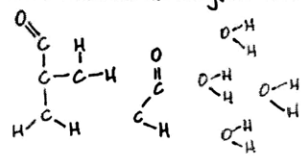
Electrostatics:

$$\sum_{i>j} \frac{D_{ij}}{r_{ij}}$$

← Potential functions

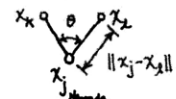
Look at VMD simulator

One or two molecules interacting (with water or salt)



QUESTIONS:
- Conformational Changes
- Different foldings
- How hard are they to pull apart

More Important Forces... (bonded/covalent)



$$E(\vec{x}) = \sum_{i=1}^{N_{bonds}} \frac{1}{2}k(\vec{x}_i - \vec{x}_j)^2 + \sum \frac{1}{2}k_{\theta i}(\theta_j - \theta_i)^2 + \ldots$$

Move an atom

Simple case

Force between A1 & A2: $\cancel{k\|\vec{x}_1 - \vec{x}_2\|}$
$k_b(\|\vec{x}_1 - \vec{x}_2\| - d_{eq})$ ← 0 don't worry

Spring potential energy
$\frac{1}{2}k_b(\|\vec{x}_1 - \vec{x}_2\|)^2$

Molecular Dynamics

$$\underbrace{\text{Force on atom } i = -\frac{\partial}{\partial x_i}E(\vec{x})}_{F(\vec{x}) = -\nabla_{\vec{x}}E(\vec{x})}$$

$\vec{x}$ ← All atom positions

HAMILTONIAN POINT OF VIEW

$$H(q,p) = \underbrace{\frac{1}{2}\sum \frac{1}{m_i}\vec{p}_i^{\,2}}_{\text{Kinetic energy}} + \underbrace{E(\vec{q})}_{\text{Potential energy}}$$

position; momentum ($m\vec{v}$); bonds, angle, VDW, es

$$\frac{d}{dt}\begin{bmatrix}\vec{q}\\ \vec{p}\end{bmatrix} = \begin{bmatrix}\frac{\partial}{\partial \vec{p}}H(p,q)\\ -\frac{\partial}{\partial \vec{q}}H(p,q)\end{bmatrix}$$

$M = [m_1 \ldots m_n]$

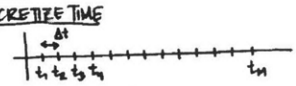
$$\frac{d}{dt}\vec{q} = M^{-1}\vec{p} = \frac{\partial}{\partial \vec{p}}H(q,p)$$

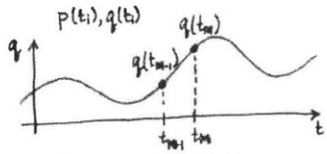
$$\frac{d}{dt}\vec{p} = -\frac{\partial}{\partial \vec{q}}H(p,q) = -\frac{\partial}{\partial \vec{q}}E(q)$$

Invariant: $H(p,q) = \text{const.}$

THURSDAY
25 FEB 06

DISCRETIZE TIME





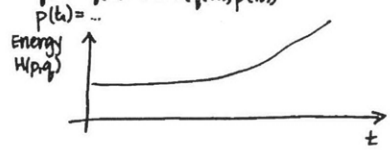
$$\frac{d}{dt} q \cong \frac{q(t_m) - q(t_{m-1})}{\Delta t}$$

FORWARD EULER

$$\begin{cases} q(t_{m+1}) = q(t_m) + \Delta t \, \frac{\partial}{\partial q} H(q(t_m), p(t_m)) \\ p(t_{m+1}) = p(t_m) + \Delta t \, \frac{\partial}{\partial p} H(q(t_m), p(t_m)) \end{cases}$$

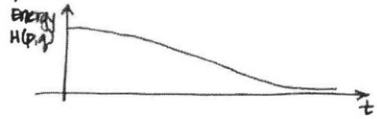
KNOWN

$$q(t_1) = q(t_0) + \Delta t \, H(q(t_0), p(t_0))$$
$$p(t_1) = \ldots$$



BACKWARD EULER

$$q(t_{m+1}) = q(t_m) + \Delta t \, \frac{\partial}{\partial q} H(q(t_{m+1}), p(t_{m+1}))$$
$$p(t_{m+1}) = \ldots$$



2.6 Molecular Dynamics and Electrostatics

6.581 / BE 482

LECTURE 6: MOLECULAR DYNAMICS & ELECTROSTATICS

TUESDAY 28 FEB 2006

OVERVIEW:

1) Remind about $E(x)$
2) Time Discretization
3) Electrostatics Evaluation
   - Costs low

$$E(x) = \sum_{\text{bonds}} + \sum_{\text{bond angles}} + \sum_{\text{torsions}} + \sum_{\text{all pairs}} \frac{B_{ij}}{r^{12}} - \frac{C_{ij}}{r^6} + \underbrace{\sum_{\text{all pairs}} \frac{q_i q_j}{r_{ij}}}_{\text{problematic term}}$$

$$m_i \frac{d}{dt} V_{i\{x,y,z\}} = -\frac{\partial}{\partial x_{i\{x,y,z\}}} E(\vec{x})$$

m_i: i^{th} atomic mass

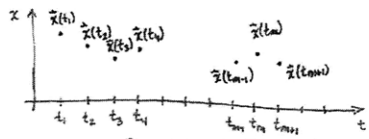
$$M \frac{d}{dt}\vec{V} = -\nabla_{\vec{x}} E(\vec{x})$$

$$\frac{d}{dt}\vec{x} = \vec{V}$$

$\vec{x}$: atomic positions; $\vec{V}$: atomic velocities

System of 2nd order Equations

$$M \frac{d^2}{dt^2}\vec{x} = -\nabla_{\vec{x}} E(\vec{x})$$



$$\frac{d}{dt}\vec{x}(t_m) \simeq \frac{\vec{x}(t_{m+1}) - \vec{x}(t_m)}{t_{m+1} - t_m} \simeq \frac{\vec{x}(t_m) - \vec{x}(t_{m-1})}{t_m - t_{m-1}}$$

$$\Delta t = t_{m+1} - t_m = t_m - t_{m-1}$$

$$\frac{d^2}{dt^2}\vec{x}(t_m) \simeq \frac{\dfrac{\vec{x}(t_{m+1}) - \vec{x}(t_m)}{\Delta t} - \dfrac{\vec{x}(t_m) - \vec{x}(t_{m-1})}{\Delta t}}{\Delta t}$$

$$\frac{d}{dt}\vec{V}(t_m) \simeq \underbrace{\frac{\vec{x}(t_{m+1}) - 2\vec{x}(t_m) + \vec{x}(t_{m-1})}{\Delta t^2}}_{\text{verlet}}$$

$$M \frac{\vec{x}(t_{m+1}) - 2\vec{x}(t_m) + \vec{x}(t_{m-1})}{\Delta t^2} = -\nabla_x E(\vec{x})$$

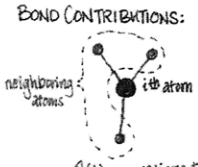
ALGORITHM:

$$\underbrace{x(t_{m+1}) = 2x(t_m) - x(t_{m-1}) - \Delta t^2 M^{-1} E(\vec{x})}_{\text{Explicit Integration Scheme}}$$

on each time step:

Evaluate $E_{(x)}(\vec{x}(t_m))$ + other stuff

BOND CONTRIBUTIONS:



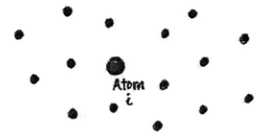
$\mathcal{O}(1)$ operations to compute

$$E(x) = \underbrace{\sum_{\text{bonds}} + \sum_{\text{bond angles}} + \sum_{\text{torsions}}}_{\text{inexpensive} \sim \mathcal{O}(N)} + \sum_{\text{all pairs}} \frac{B_{ij}}{r^{12}} - \frac{C_{ij}}{r^6} + \sum_{\text{all pairs}} \frac{q_i q_j}{r_{ij}}$$

van der Waals:

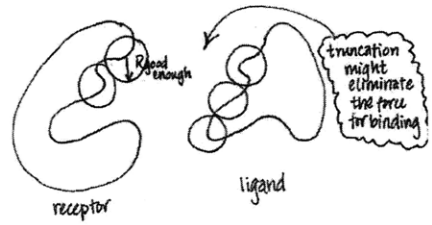
all pairwise interactions $\mathcal{O}(n)$ operations per atom $\Longrightarrow$ but truncate $\Rightarrow \mathcal{O}(1)$

Electrostatics:

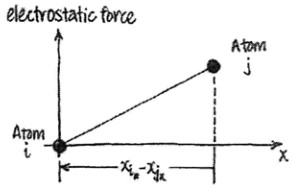


electrostatic forces die like $1/r^2$

can we truncate?

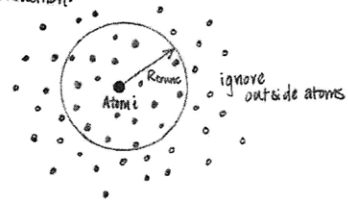


electrostatic force

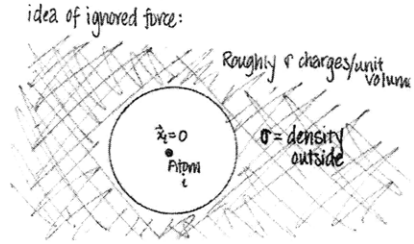


$$F = E_x = -\frac{\partial}{\partial x_{i_x}} \frac{q_i q_j}{|\vec{x}_i - \vec{x}_j|} = \frac{q_i q_j (x_{i_x} - x_{j_x})}{|\vec{x}_i - \vec{x}_j|^3}$$

try truncation:



idea of ignored force:



bound on ignored force:

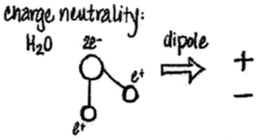
ignoring direction cancellation

$$\text{Force Bound} \simeq \int_{\substack{\text{Volume}\\ \text{outside}}} \frac{\sigma(\vec{x})}{|\vec{x}|^2}\, dV$$

$$\simeq \int_0^{2\pi}\int_0^{\pi}\int_{R_{trunc}}^{\infty} \sigma \frac{1}{R^2} R^2 \sin\phi \, dR\, d\phi\, d\theta$$

$$\simeq \int_0^{2\pi}\int_0^{\pi} \Big[\sigma R\Big]_{R_{trunc}}^{\infty} \sin\phi\, d\phi\, d\theta \rightarrow \infty$$

charge neutrality:



Potential due to dipole: $\dfrac{\vec{p}\cdot(\vec{x}_i-\vec{x}_j)}{|\vec{x}_i-\vec{x}_j|^3}$

dipole moment

force due to dipole $\propto \frac{1}{R^3}$

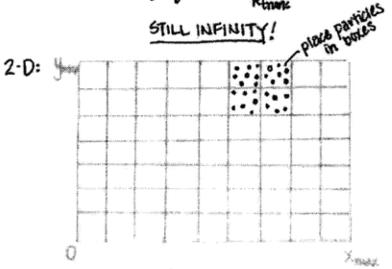
bound on ignored force (dipole assumption):

$$\text{Force bound} \simeq \int_{\Omega} \frac{\rho(\vec{x})}{R^3}\, dV \quad \text{dipole density}$$

$$\simeq \int_0^{2\pi}\int_0^{\pi}\int_{R_{trunc}}^{\infty} \frac{\rho}{R^3} R^2 \sin\phi \, dR\, d\phi\, d\theta$$

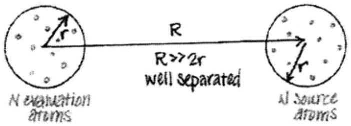
$$\simeq \int_0^{2\pi}\int_0^{\pi} \Big[\rho \ln R\Big]_{R_{trunc}}^{\infty} \sin\phi\, d\phi\, d\theta \rightarrow \infty$$

STILL INFINITY!



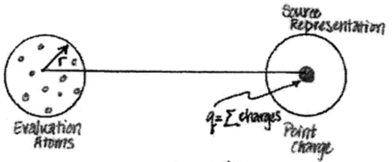
easy neighbors

MULTIPOLE IDEA:



direct force calculation is $\theta(N^2)$

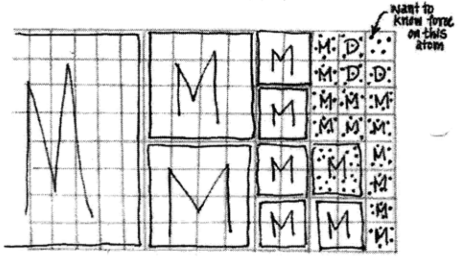
for each of N evaluation atoms: $F_i = \sum_{j \in \text{source atoms}} F_{ij}$



Source Charge Evaluation ~ $\theta(N)$

Single Charge Evaluation + 3 dipole Evaluations

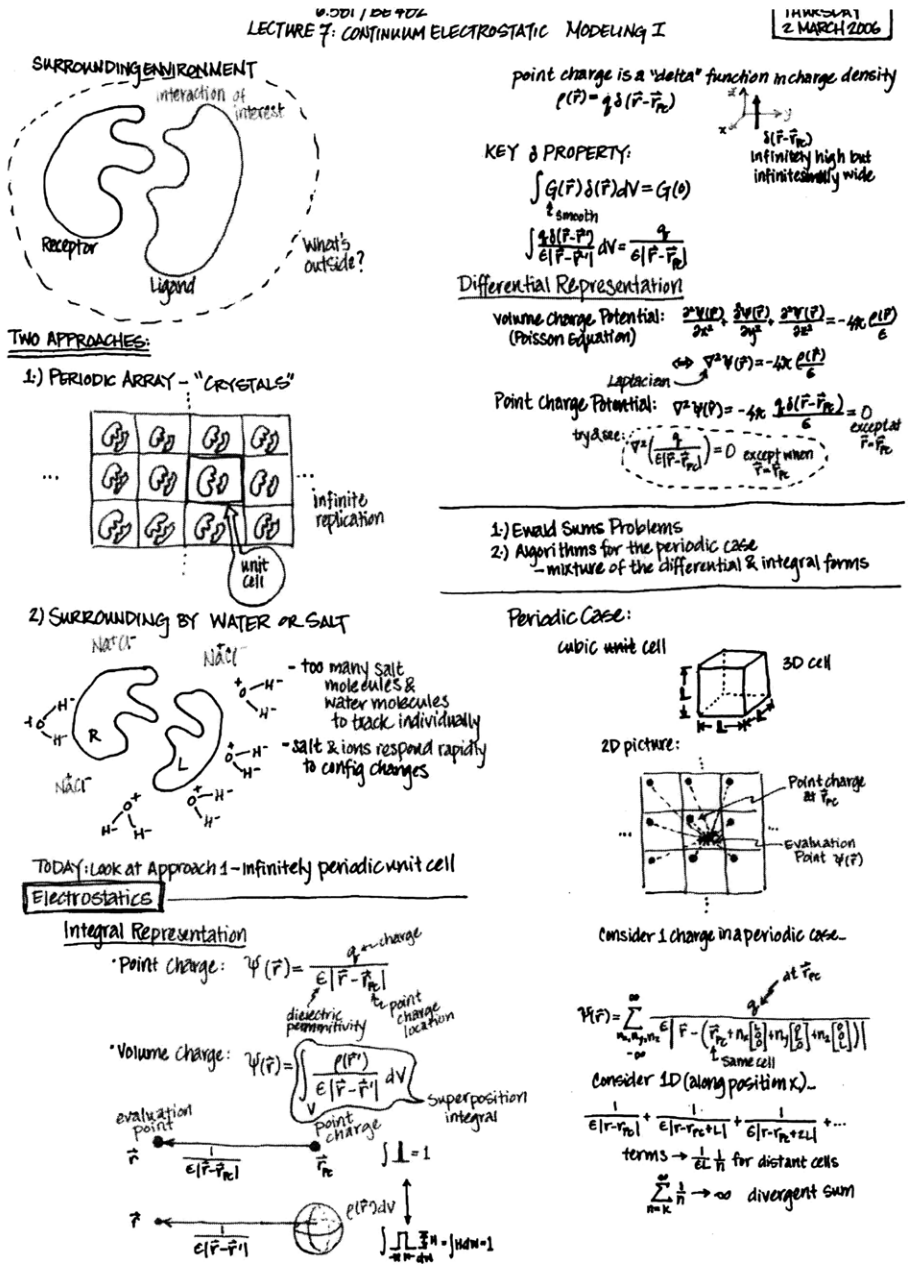
$\theta(N) + \theta(N)$



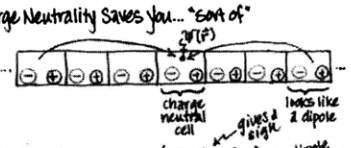
Multiresolution

$\theta(N\log N)$

2.7 Continuum Electrostatic Modeling I



Charge Neutrality Saves you... "sort of"



DIPOLE POTENTIAL: $\dfrac{(\vec{r}-\vec{r}_{dipole})\cdot\vec{p}}{\epsilon|\vec{r}-\vec{r}_{dipole}|^3}$ ← dipole moment; gives a sign

decays like $\frac{1}{R^2}$ ⇒ not fast enough in 3D

Symmetry ensures cancellation
left & right dipoles nearly cancel
difference dies like $\frac{1}{R^3}$ ⇒ fast enough

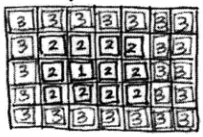
Computed sum depends on ORDER

symmetric

"shell" ordering

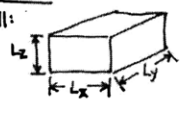
⇓

vs. asymmetric (diverges)



EWALD SUM ALGORITHMS

consider a 3-D cell:



has N charges $\vec{q}_1, \ldots, \vec{q}_N$ at $\vec{r}_1, \ldots, \vec{r}_N$

only finite potential if $\sum \vec{q}_i = 0$ charge neutral

finite potential ← must be periodic

[Periodic Series are described by Fourier Series:]

potential

$$U(\vec{r}) = \sum_{m_x,y,z=-\infty}^{+\infty} \underbrace{\vec{U}[m_x,m_y,m_z]}_{\text{determine}} e^{j2\pi m_x \frac{x}{L_x}} \cdot e^{j2\pi m_y \frac{y}{L_y}} \cdot e^{j2\pi m_z \frac{z}{L_z}}$$

periodic

charge density is also periodic:

$$\rho(\vec{r}) = \sum \underbrace{\vec{\rho}[m_x,m_y,m_z]}_{\text{charge density}} e^{j2\pi\left(\frac{m_x}{L_x}x + \frac{m_y}{L_y}y + \frac{m_z}{L_z}z\right)}$$

$$\vec{m}_L \cdot \vec{r} = \left[\frac{m_x}{L_x}\ \frac{m_y}{L_y}\ \frac{m_z}{L_z}\right] \cdot \begin{bmatrix} x \\ y \\ z \end{bmatrix}$$

Poisson Eqn: $\nabla^2 u = -4\pi\rho$

- plug in the Fourier Representation on both sides
- match Fourier terms...

$$\vec{U}[m_x,m_y,m_z] = \frac{-4\pi}{(2\pi)^2 \vec{m}_L^2} \vec{\rho}[m_x,m_y,m_z]$$

(scalar) (same Fourier components)

to solve periodic problem:

- take charge density compute Fourier Series ← hard due to point charges
- easily compute Fourier Series for periodic potential

2.8 Continuum Electrostatic Modeling II

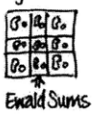
6.581/BE 482

LECTURE 8: CONTINUUM ELECTROSTATICS MODELING II

TUESDAY 7 MARCH 2006

TWO CASES:

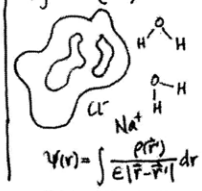
Periodic Case "Crystal"



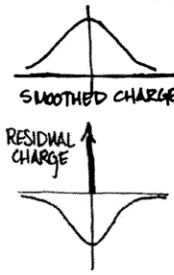
Ewald Sums

$\nabla^2 \psi = \frac{4\pi\rho}{\epsilon}$

Surrounded by Water (Salt)



$\psi(r) = \int \frac{\rho(\vec{r}')}{\epsilon|\vec{r}-\vec{r}'|} dr$



Fourier series of potential due to smoothed σ

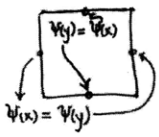
decays exponentially fast in Fourier space

few Fourier Coef's can be truncated

Potential dies exponentially fast in physical space

can be truncated

PERIODIC CASE:



Periodic Boundary conditions on potential

Fourier Series Coeffs for potential

$$\sum_{m_x,m_y,m_z=-\infty}^{\infty} \underbrace{\bar{U}(m_x,m_y,m_z)}_{\text{indices}} e^{j2\pi\left(\frac{m_x}{L_x}x+\frac{m_y}{L_y}y+\frac{m_z}{L_z}z\right)}$$

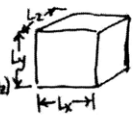
periodic function

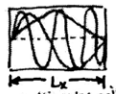
Fourier Series Coefficients for Potential

$$\bar{U}(m_x,m_y,m_z) = \frac{4\pi}{(2\pi)^2 \vec{m}^2} \cdot \bar{\sigma}(m_x,m_y,m_z)$$

F.S. for charge

(∇^2 doesn't mix frequency)





$[m_x\ m_y\ m_z]$

$$\sigma(\vec{r}) = \sum \bar{\sigma}(\vec{m}) \cdot e^{j2\pi\vec{m}\cdot\vec{r}}$$

→ multiply by $e^{-j2\pi\vec{m}\cdot\vec{r}}$ & integrate

$$\bar{\sigma}(\vec{m}) = \frac{1}{\underbrace{L_x L_y L_z}_{V}} \int_V \sigma(\vec{r}') e^{-j2\pi(\vec{m}\cdot\vec{r})} dV'$$

$$\sigma(\vec{r}) = \sum q_i \delta(\vec{r}-\vec{r}_i')$$

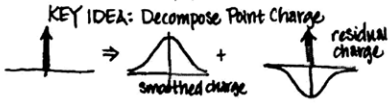
$$\bar{\sigma}(\vec{m}) = \frac{1}{V}\sum_{i=1}^{N} q_i e^{-j2\pi\vec{m}\cdot\vec{r}_i}$$

FROM MATCHING:

$$\bar{U}(\vec{m}) = \frac{1}{\pi V}\sum q_i \frac{e^{j2\pi\vec{m}\cdot\vec{r}_i}}{\underbrace{(\vec{m}\cdot\vec{m})}_{|m|^2}}$$

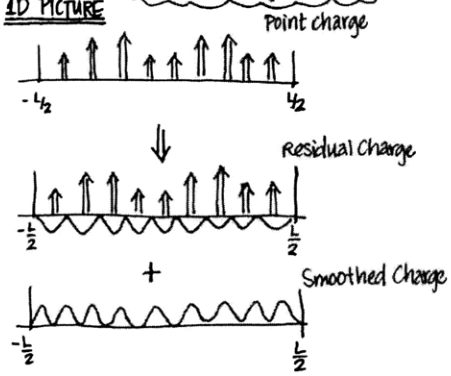
$$\bar{U}(\vec{m}) \propto \frac{1}{|m|^2} \Rightarrow \text{SLOW DECAY}$$

KEY IDEA: Decompose Point Charge



NOTE: Not charge neutral

1D PICTURE

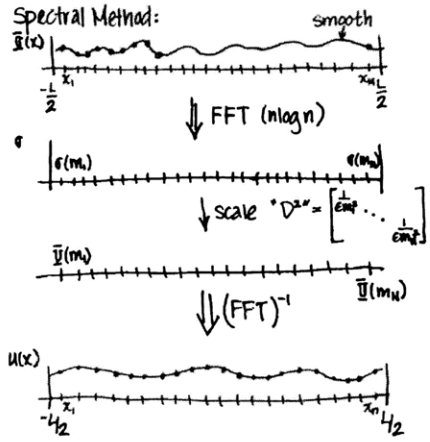


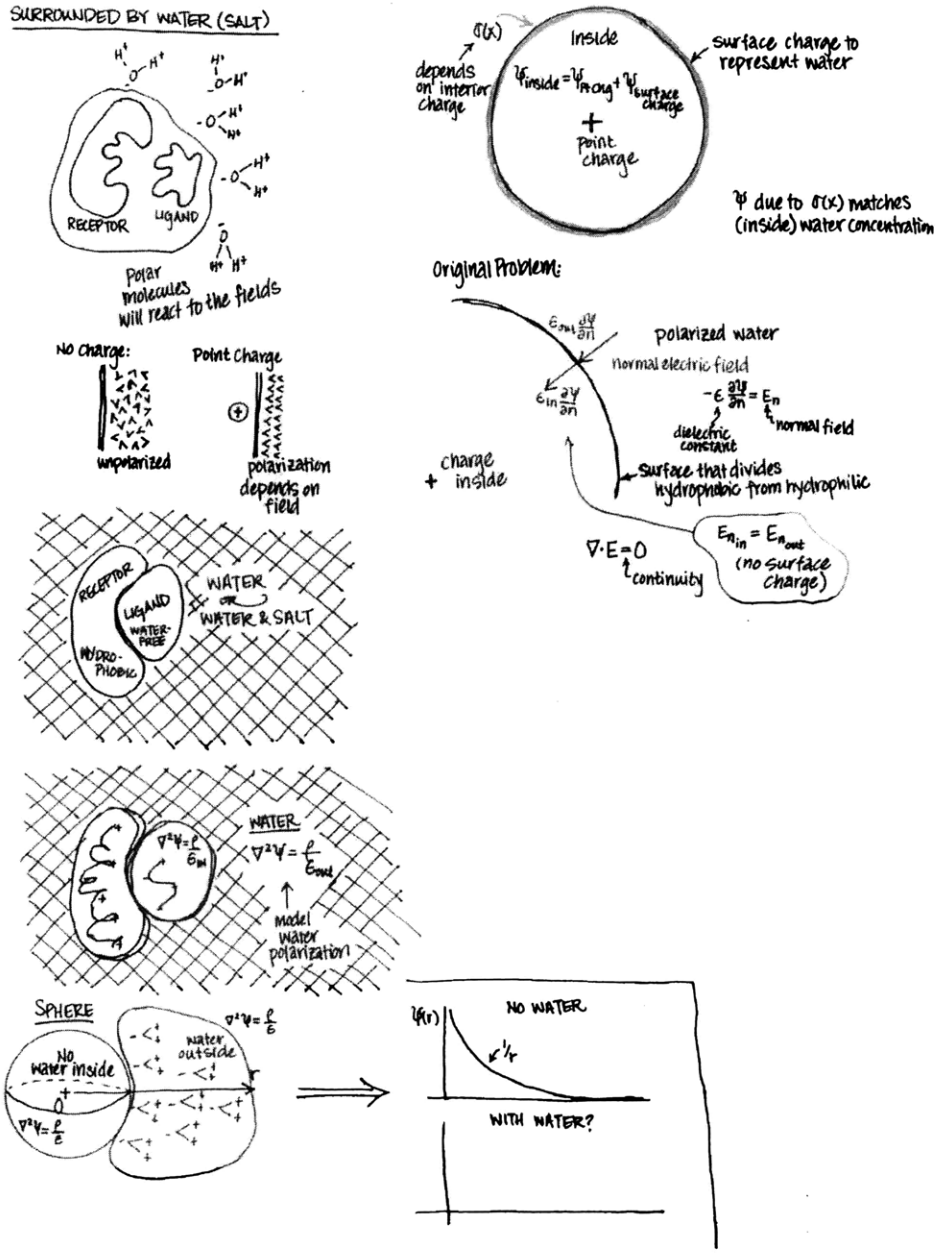
1.) Potential due to $\rho_{residual}$ affects nearby points only ($O(N)$ ops constant # pots per residual charge)

2) Use Spectral Method to compute: $\nabla^2 \psi_{smooth} = \frac{4\pi\rho_{smooth}}{\epsilon}$

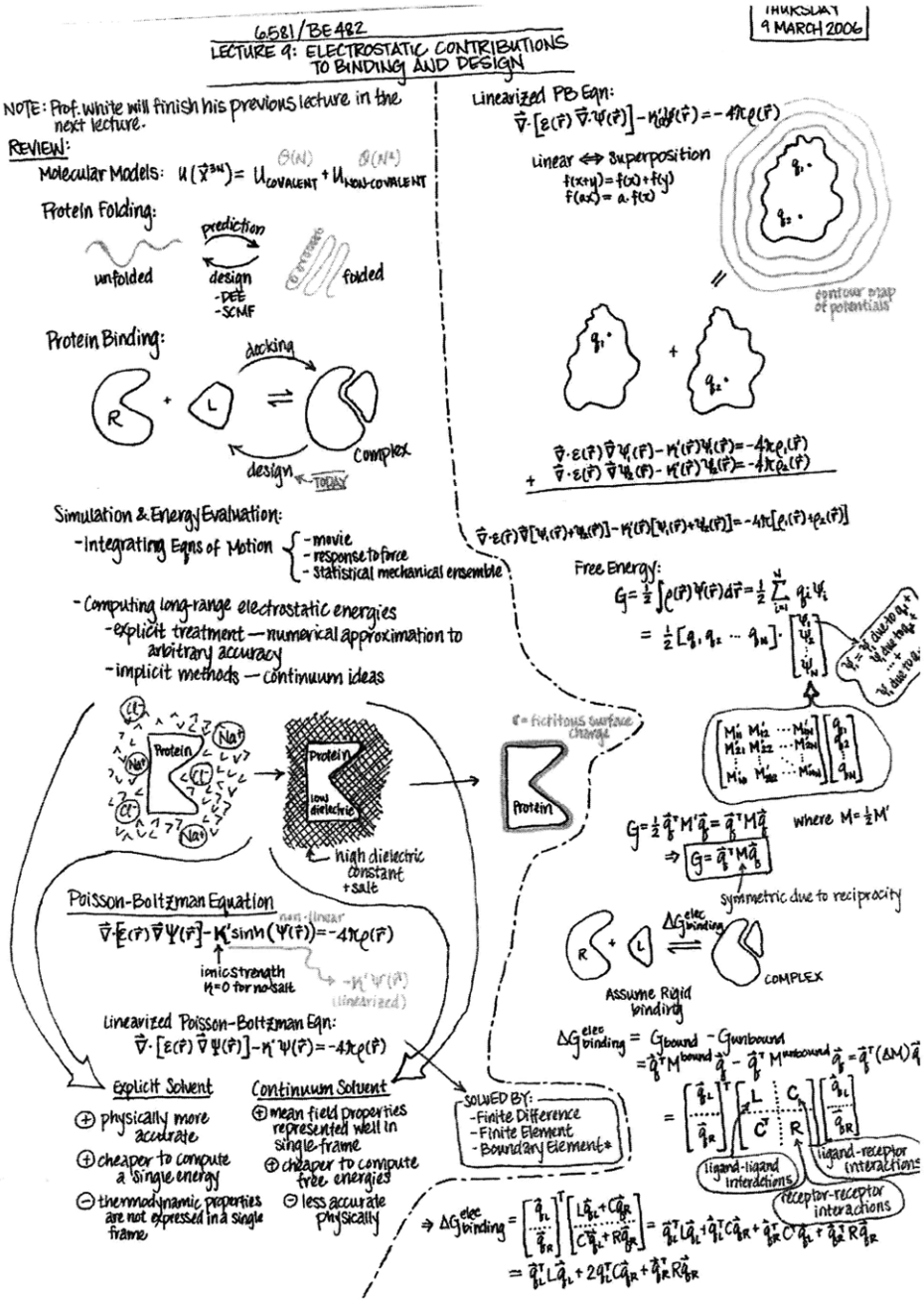
3) Add results

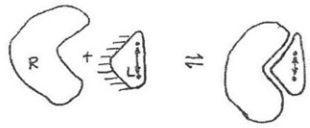
Spectral Method:




SURROUNDED BY WATER (SALT)
RECEPTOR
LIGAND
Polar molecules will react to the fields
No charge:
Point charge
unpolarized
polarization depends on field
RECEPTOR
LIGAND
WATER-FREE
HYDRO-PHOBIC
WATER
OR
WATER & SALT
WATER
model water polarization
SPHERE
No water inside
water outside
NO WATER
WITH WATER?
σ(x)
depends on interior charge
Inside
surface charge to represent water
+ point charge
ψ due to σ(x) matches (inside) water concentration
Original Problem:
polarized water
normal electric field
normal field
dielectric constant
charge + inside
surface that divides hydrophobic from hydrophilic
(no surface charge)
continuity

2.9 Electrostatic Contributions to Binding and Design





$$\Delta G^{elec}_{binding} = \underbrace{\vec{q}_L^{\,T} L \vec{q}_L}_{\text{ligand desolvation}} + \underbrace{2\vec{q}_L^{\,T} C \vec{q}_R}_{\text{interactions recovered}} + \underbrace{\vec{q}_R^{\,T} R \vec{q}_R}_{\text{receptor desolvation}}$$

$$\vec{\nabla}_{\vec{q}_L}\left(\Delta G^{elec}_{binding}\right) = 2L\vec{q}_L + 2C\vec{q}_R = \vec{0}$$

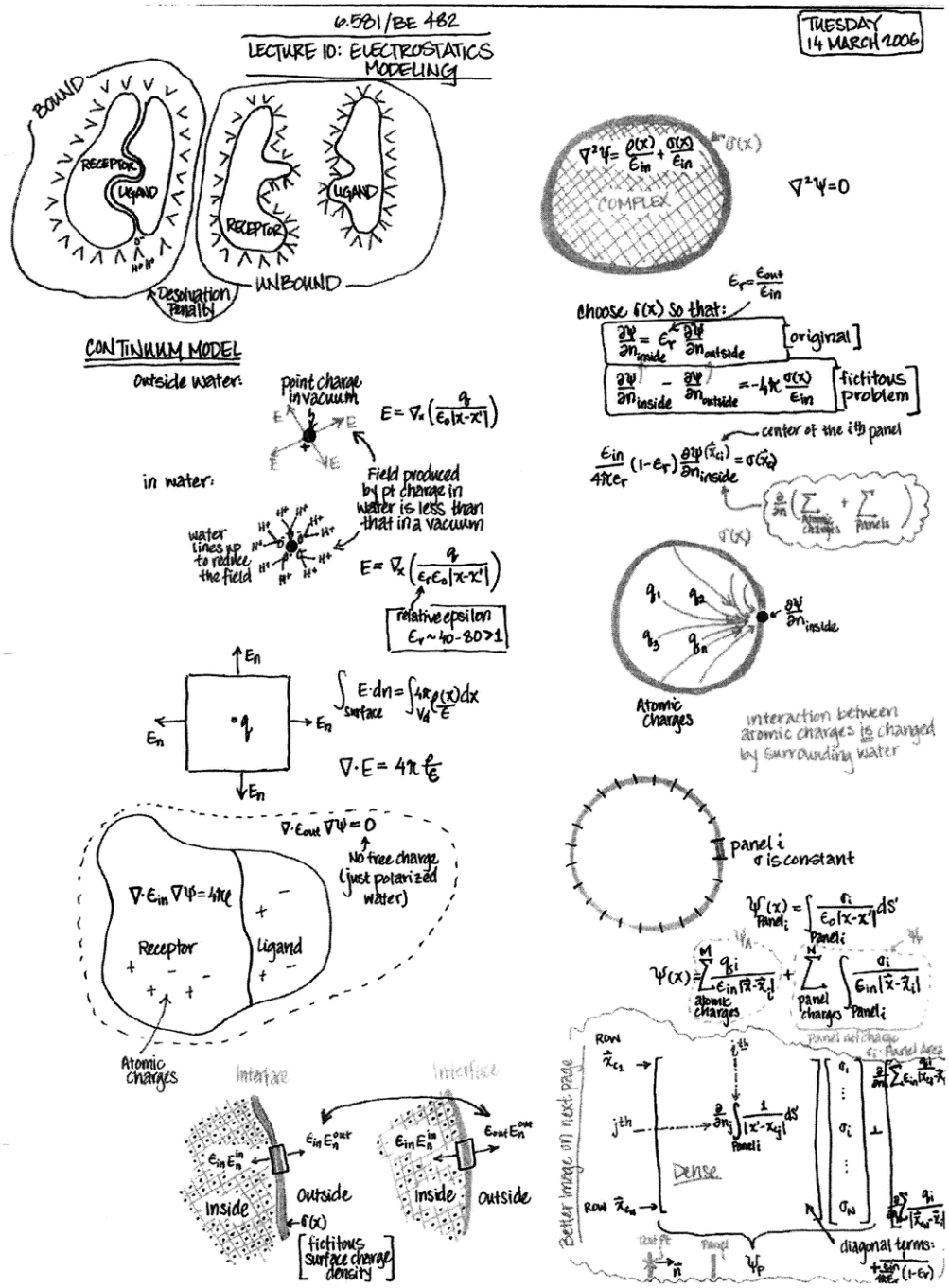
$$\Rightarrow L\vec{q}_L = -C\vec{q}_R$$

$$\Rightarrow \boxed{\vec{q}_L^{\,opt} = -L^{-1}C\vec{q}_R}$$

JPhys B 105: 880-888 (2001).
chorismate mutase

JACS 125: 5598-9 (2003).
10-fold improvement in binding

2.10 Electrostatics Modeling



$$\underbrace{\frac{\partial \psi_P(\vec{x}_{c_i})}{\partial n}}_{\substack{\text{Panel}\\ \text{Charge}\\ \text{Potential}}} - \frac{\sigma_i}{\frac{\epsilon_{in}}{4\pi\epsilon_r}(1-\epsilon_r)} = \underbrace{-\frac{\partial \psi_A(\vec{x}_{c_i})}{\partial n}}_{\substack{\text{Atomic}\\ \text{Charge}\\ \text{Potential}}}$$

known

Dense

$$P\vec{\sigma}_P = \vec{\psi}_A$$

Iterate: Guess at $\vec{\sigma}_P^0$

Calculate $\psi_A - P\vec{\sigma}_P^0 = \text{Residual}$

$\vec{\sigma}_P^1 = f(\text{Residual})$

~~Matrix Vector Product~~ ← N potential derivatives from N charges using Multipole $\mathcal{O}(N)$

$$\overbrace{\begin{array}{c} \text{Row } \vec{x}_{c_1} \rightarrow \\ \\ j^{th} \rightarrow \\ \\ \text{Row } \vec{x}_{c_N} \rightarrow \end{array}\begin{bmatrix} \\ \\ \frac{\partial}{\partial n_j}\int_{\text{Panel } i} \frac{1}{|x'-x_j|}\,dS' \\ \\ \\ \end{bmatrix}}^{\psi_P} \begin{bmatrix} \sigma_1 \\ \vdots \\ \sigma_i \\ \vdots \\ \sigma_N \end{bmatrix} = -\overbrace{\begin{bmatrix} \frac{\partial}{\partial n_1}\sum \frac{q_i}{\epsilon_{in}|\vec{x}_{c_1}-\vec{x}_i|} \\ \vdots \\ \frac{\partial}{\partial n_j}\sum \frac{q_i}{\epsilon_{in}|\vec{x}_{c_j}-\vec{x}_i|} \\ \vdots \\ \frac{\partial}{\partial n_N}\sum \frac{q_i}{\epsilon_{in}|\vec{x}_{c_N}-\vec{x}_i|} \end{bmatrix}}^{\psi_A}$$

i^{th}

DENSE MATRIX (Every element is nonzero)

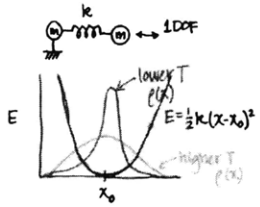
Diagonal terms: $+\frac{1}{\frac{\epsilon_{in}}{4\pi\epsilon_r}(1-\epsilon_r)}$

2.11 Statistical Mechanics

6.581 / BE.482

LECTURE 11: STATISTICAL MECHANICS

THURSDAY 16 MARCH 2006



Imagine a dense system of these packed together → see probability distributions above

$$\rho(x) = \text{probability of finding a copy of the system with configuration } x = \frac{e^{-\frac{E(x)}{k_BT}}}{\int_{-\infty}^{\infty} e^{-\frac{E(x')}{k_BT}}\,dx'}$$

$= Q$ Configurational Partition Function

k_B = Boltzmann Constant $= 1.987 \cdot 10^{-3}\ \frac{kcal}{mol \cdot K}$

T = Absolute Temperature $= 273.15 + t\ K$ (t (°C))

Note:

$$\int_{-\infty}^{\infty} \rho(x)\,dx = \int_{-\infty}^{\infty} \frac{e^{-\frac{E(x)}{k_BT}}}{Q}\,dx = \frac{1}{Q}Q = 1$$

What is average position?

$$\bar{x} = \sum_{\text{all possible configurational states}} (\text{probability of } x)\cdot(\text{value at that position, } x)$$

$$= \int_{-\infty}^{\infty} \rho(x)\,x\,dx = \int_{-\infty}^{\infty} \frac{x e^{-\frac{E(x)}{k_BT}}}{Q}\,dx = \frac{\int_{-\infty}^{\infty} x e^{-\frac{k(x-x_0)^2}{2k_BT}}\,dx}{Q}$$

GENERAL SPECIFIC

Change of variables: $y = x - x_0$, $dy = dx$

$$= \frac{1}{Q}\int_{-\infty}^{\infty} (y+x_0) e^{-\frac{ky^2}{2k_BT}}\,dy$$

$$= \frac{1}{Q}\int_{-\infty}^{\infty} y e^{-\frac{ky^2}{2k_BT}}\,dy + \frac{x_0}{Q}\int_{-\infty}^{\infty} e^{-\frac{ky^2}{2k_BT}}\,dy$$

$\to 0$, $\to Q$

$= x_0$

Computing Average Fluctuation:

$$\sqrt{\overline{(x-\bar{x})^2}} = \sqrt{\frac{1}{Q}\int_{-\infty}^{\infty} (x-x_0)^2 e^{-\frac{k(x-x_0)^2}{2k_BT}}\,dx}$$

$$= \sqrt{\frac{k_BT}{k}}$$

$k_BT \longrightarrow$ higher $T \Rightarrow$ greater fluctuation

$k \longrightarrow$ higher $k \Rightarrow$ lower fluctuation

Computing Average Potential Energy

$$\bar{E} = \langle E \rangle = \frac{1}{Q}\int_{-\infty}^{\infty} E(x)\, e^{-\frac{k(x-x_0)^2}{2k_BT}}\,dx = \frac{1}{Q}\int_{-\infty}^{\infty} \frac{k}{2}(x-x_0)^2 e^{-\frac{k(x-x_0)^2}{2k_BT}}\,dx$$

$$= \frac{k_BT}{2}$$

~3,000 atoms of protein
~10,000 atoms of solvent

N ~13,000 particles ⇒ ~39,000 degrees of freedom

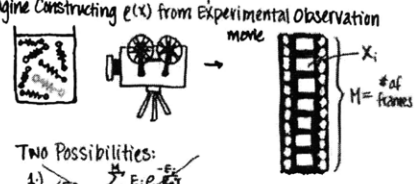
Compute Average Str. of Protein

$$\langle \vec{X}^{3N} \rangle = \frac{\int_{-\infty}^{\infty}\cdots\int_{-\infty}^{\infty} \vec{x}^{3N} e^{-\frac{E(\vec{x}^{3N})}{k_BT}}\,d\vec{x}^{3N}}{\int_{-\infty}^{\infty}\cdots\int_{-\infty}^{\infty} e^{-\frac{E(\vec{x}^{3N})}{k_BT}}\,d\vec{x}^{3N}}$$

first problem: $E(\vec{x}^{3N})$ is not analytically integrable → must do it numerically

second problem: 0.1 Å on a 100 Å grid
1000 points per dimension
$(1000)^{39,000}$ points total

Imagine Constructing $\rho(x)$ from Experimental Observation



Two Possibilities:

1.) $\cancel{\langle E \rangle = \frac{\sum_{i=1}^{M} E_i e^{-\frac{E_i}{k_BT}}}{\sum_{i=1}^{M} e^{-\frac{E_i}{k_BT}}}}$

or

2.) $$\langle E \rangle = \frac{\sum_{i=1}^{M} E_i}{M}$$

If we use a molecular dynamics simulation as the movie:

Geometric Quantity: $\langle |\vec{r}_{i\cdots j}| \rangle = \frac{1}{M}\sum_{k=1}^{M} |\vec{r}_{i\cdots j}(\vec{x}_k)|$

Interaction Energy: $\langle U_{i\cdots j} \rangle = \frac{1}{M}\sum_{k=1}^{M} U_{i\cdots j}(\vec{x}_k)$

Total Potential Energy: $\langle U \rangle = \frac{1}{M}\sum_{k=1}^{M} U(\vec{x}_k)$

Metropolis Monte Carlo, like MD, also produces a trajectory that converges to a statistical mechanical ensemble
Metropolis et al., J Chem Phys 21: 1087-1092 (1953).

Problem trying to get the free energy...

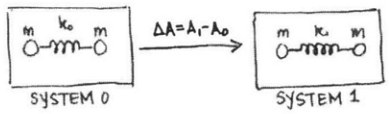
~~I attempted to write:~~ $\langle A \rangle = \frac{1}{M}\sum_{i=1}^{M} A_i$ ← this doesn't exist

↑ free energy

Statistical Mechanics gives us a definition:

$$A = -k_BT \ln Q = -k_BT \ln\left[\int_{-\infty}^{\infty} e^{-\frac{U(\vec{x}^{3N})}{k_BT}}\,d\vec{x}^{3N}\right]$$

Doesn't help, because Q is $1000^{39,000}$-dimensional integral



Thermodynamic Integration (Kirkwood, 1934)

1.) Construct a "hybrid" potential that smoothly connects $0 \to 1$.

$$U(\lambda) = (1-\lambda)U_0 + \lambda U_1 \qquad \begin{array}{l}\lambda : (0 \to 1) \\ U : (U_0 \to U_1)\end{array}$$

2.) Fundamental Theorem of Integral Calculus

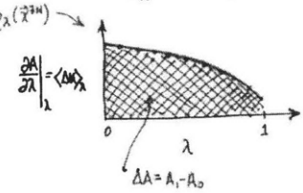
$$\Delta A = A_1 - A_0 = \int_0^1 \frac{\partial A}{\partial \lambda'} d\lambda'$$

$$A(\lambda) = -k_B T \ln\left[\int e^{-\frac{U(\vec{x}^{3N},\lambda)}{k_B T}} d\vec{x}^{3N}\right]$$

$$\frac{\partial A(\lambda)}{\partial \lambda} = \frac{\cancel{-k_B T} \int e^{-\frac{U(\vec{x}^{3N},\lambda)}{k_B T}} d\vec{x}^{3N} \left[\frac{\cancel{-1}}{k_B T} \frac{\partial U(\vec{x}^{3N},\lambda)}{\partial \lambda}\right]}{\int e^{-\frac{U(\vec{x}^{3N},\lambda)}{k_B T}} d\vec{x}^{3N}} = \left\langle \frac{\partial U}{\partial \lambda} \right\rangle_\lambda = \langle U_1 - U_0 \rangle_\lambda = \langle \Delta U \rangle_\lambda$$

$\rightarrow \rho_\lambda(\vec{x}^{3N})$

$$\Delta A = A_1 - A_0 = \int_0^1 \langle \Delta U \rangle_{\lambda'} d\lambda'$$



2.12 Statistical Mechanics

PSET 3 DUE ON TUES 4 APRIL

6.581 / BE.482
LECTURE 12: STATISTICAL MECHANICS

TUESDAY 21 MARCH 2006

SUMMARY OF LAST TIME

- Uniform Sampling

$$\overline{f(\vec{x})} = \langle f(\vec{x}) \rangle = \int d\vec{x}\, \rho(\vec{x}) f(\vec{x})$$

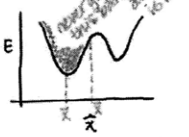
probability; desired function

$$\rho(\vec{x}) = \frac{e^{-\frac{E(\vec{x})}{k_B T}}}{\int d\vec{x}'\, e^{-\frac{E(\vec{x}')}{k_B T}}}$$

$= Q$ (configurational partition function)

- Boltzman Sampling
  Molecular Dynamics or
  Metropolis Monte Carlo Simulation

$$\langle f(\vec{x}) \rangle = \frac{1}{M} \sum_{i=1}^{M} f(\vec{x}_i)$$



Problem: Free Energy doesn't exist in single frame
~~$f(\vec{x}) = A(\vec{x})$~~ so can't compute this way

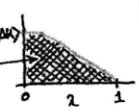
Solution: Can compute free energy changes from simulation (ensemble) averages

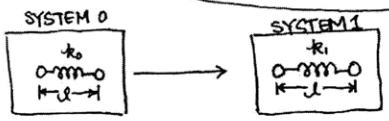
$$A = -k_B T \ln Q$$

If $U(\lambda) = (1-\lambda)U_0 + \lambda U_1$ and let $\Delta U = U_1 - U_0$

$$\Delta U(\vec{x}) = U_1(\vec{x}) - U_0(\vec{x})$$

Then $\Delta A = A_1 - A_0 = \int_0^1 d\lambda \langle \Delta U \rangle_\lambda$



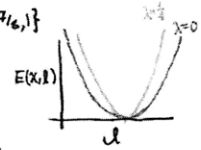


Run a series of simulations with

$$U(\lambda) = (1-\lambda)U_0 + \lambda U_1$$
$$= \tfrac{1}{2}(1-\lambda)k_0(x-\ell)^2 + \tfrac{1}{2}\lambda k_1 (x-\ell)^2$$
$$= \tfrac{1}{2}[k_0 + \lambda(k_1 - k_0)](x-\ell)^2$$

$\lambda = \{0, 1/8, 1/4, 3/8, 1/2, 5/8, 3/4, 7/16, 1\}$

$\Delta U = \frac{1}{2}(k_1 - k_0)(x-\ell)^2$

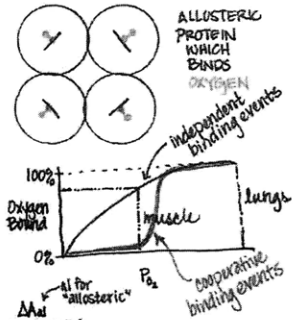


$$\Delta U = U_1 - U_0 = \sum_i \Delta U_i$$

$$\Delta A = \int_0^1 d\lambda \langle \Delta U \rangle = \int_0^1 d\lambda \langle \sum_i \Delta U_i \rangle = \sum_i \underbrace{\int_0^1 d\lambda \langle \Delta U_i \rangle}_{\Delta A_i} = \sum_i \Delta A_i$$

Science, 244:1069-1072 (1989)

Hemoglobin



wild type: DEOXY ⇌ OXY
(low O_2 affinity) (high O_2 affinity)

mutant: DEOXY ⇌ OXY ← Hb Radcliffe mutation [hemoglobin]
$\Delta A'_{al}$
Asp Gl(99)β → Ala

Experimentally:

$$\Delta\Delta A = \Delta A' - \Delta A = -3.4 \text{ kcal/mol/interface}$$

- Reduced cooperativity & increased oxygen affinity
- Shifts equilibrium toward OXY
- Less ability to release O_2 at muscles

DEOXY (wild type): Asp 99β, h-bond, Tyr C7(42)α, Asn G4(97)α

OXY: Asp 99β, solvent, Asp 94α, Tyr 42α, Asn 97α

thermodynamic cycle → $0 = \Delta A_{deoxy} + \Delta A'_{al} - \Delta A_{oxy} - \Delta A_{al}$

$$\Delta\Delta A = \Delta A'_{al} - \Delta A_{al} = \Delta A_{oxy} - \Delta A_{deoxy}$$

$$\Delta\Delta A_{calc} = -5.5 \text{ kcal/mol/interface}$$

	ΔΔA [kcal/mol/interface]	
Total	-5.5	
Solvent	+22.5	← ⓐ strong solvation in oxy state of Asp9
• Tyr42α	-12.7	← ⓑ consistent w/ loss of h-bond
Asp94α	-22.4	← ⓒ strong repulsion in oxy form
Val96α	5.5	
• Asn97α	3.3	← ⓓ inconsistent w/ std. explanatio

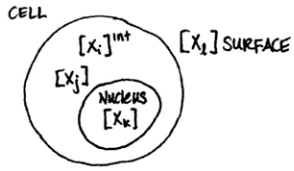
2.13 Formulating Models

6.581/BE.482

LECTURE 14: FORMULATING MODELS

TUESDAY 4 APRIL 2006

DIFFERENTIAL EQUATION MODEL



VERY GENERAL SETTING

$$\begin{bmatrix}[x_1]\\ \vdots \\ [x_N]\end{bmatrix} = \vec{X} \quad \vec{U} = \begin{bmatrix}[u_1]\\ \vdots \\ [u_K]\end{bmatrix}$$

unknown

time-variation of concentrations

$$\frac{d}{dt}[x_i] = \underbrace{V^+(\vec{X},\vec{U})}_{\text{rate of production}} - \underbrace{V^-(\vec{X},\vec{U})}_{\text{rate of consumption}}$$

Ex: $A + B \underset{K_2}{\overset{K_1}{\rightleftharpoons}} C$

$$\frac{d}{dt}[C] = K_1[A][B] - K_2[C]$$
$$\frac{d}{dt}[A] = K_2[C] - K_1[A][B]$$
$$\frac{d}{dt}[B] = K_2[C] - K_1[A][B]$$

+ Initial conditions $A(0), B(0), C(0)$

Ex: Dimerization

$$A \underset{K_2}{\overset{K_1}{\rightleftharpoons}} A_2$$

$$\frac{d}{dt}[A_2] = K_1[A][A] - K_2[A_2]$$
$$\frac{d}{dt}[A] = 2K_2[A_2] - 2K_1[A][A]$$

STEADY-STATE (Equilibrium)

Reactions are in balance

$$\frac{d}{dt}[x_i] = 0$$

continuum approximation b/c molecules are reacting, but overall the net rate is balanced

lots of copies of x

Dimerization case:

$$K_1[A][A] = K_2[A_2]$$
$$[A]^2 = \frac{K_2}{K_1}[A_2]$$
$$[A] = \sqrt{\frac{K_2}{K_1}}\,[A_2]^{1/2}$$

Also given:

$A(0) = A_0 \quad A_2(0) = 0$

→ intial conditions

Steady-state:

$$[A] + 2[A_2] = A_0$$

CHEMICAL KINETICS FORM

$$\frac{d}{dt}[x_i] = \sum_{j=1}^{N} \gamma_j^i [x_j]$$

destination

source

(Typical case $\gamma_i^i < 0$ degradation process)

$$\frac{d}{dt}[x_i] = \sum_{j=1}^{N} \gamma_j^i [x_j] + \sum_{j=1}^{N}\sum_{k=1}^{N} \gamma_{j,k}^i [x_j][x_k]$$

(Typical Consumption Process $\gamma_{j,k}^i < 0$

Typical Production Process $\gamma_{j,k}^i > 0$)

$$\frac{d}{dt}[x_i] = \sum_{j=1}^{N} \gamma_j^i [x_j] + \sum_{j=1}^{N}\sum_{k=1}^{N} \gamma_{j,k}^i [x_j][x_k] + \sum_{j=1}^{M} \alpha_j^i [u_j] + \sum_{j=1}^{M}\sum_{k=1}^{N} \alpha_{j,k}^i [u_j][x_k] + \underbrace{\sum_{j=1}^{M}\sum_{k=1}^{M} \beta_{j,k}^i [u_j][u_k]}_{\text{MULTIPLE INPUTS}}$$

$\gamma_i^i < 0 \Rightarrow$ models degradation

$\gamma_j^i \neq 0 \Rightarrow$ models conversion $\quad x_i \underset{\gamma_j^i}{\overset{\gamma_i^j}{\rightleftharpoons}} x_j$

$\gamma_{j,k}^i \qquad x_j + x_k \underset{\gamma_i^j \gamma_i^k}{\overset{\gamma_{j,k}^i}{\rightleftharpoons}} x_i$

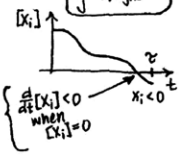
If $[x_i]$ is involved only in reactions that consume $[x_i] \Rightarrow \gamma_i^i \le 0, \gamma_{i,k}^i \le 0, \alpha_{j,i}^i \le 0$, and reactions that consume $[x_i]$ must include $[x_i] \Rightarrow$ $\boxed{j,k \neq i \quad \gamma_j^i \ge 0, \gamma_{j,k}^i \ge 0 \quad \alpha_j^i \ge 0, \alpha_{j,k}^i \ge 0}$

unnecessary for following proof

$[x_i] \ge 0$ at $t=0$

Then All $[x_i]$'s ≥ 0 for all t

PROOF: Suppose $[x_i] < 0$ for some i at some t



CONTRADICTION $\left\{ \frac{d}{dt}[x_i] < 0 \text{ when } [x_i] = 0 \right.$

$$\frac{d}{dt}\vec{X} = \underbrace{A^{(1)}\vec{X}}_{\text{terms}} + \underbrace{A^{(2)}}\begin{bmatrix}[X_1][X_1]\\ [X_1][X_2]\\ \vdots\\ [X_1][X_N]\\ [X_2][X_1]\\ [X_2][X_2]\\ \vdots\\ [X_2][X_N]\\ [X_3][X_1]\\ \vdots\\ [X_N][X_N]\end{bmatrix}$$

$$\begin{bmatrix}[X_1]\\ \vdots\\ [X_N]\end{bmatrix}$$

$$\sum \gamma^i_j [X_j]$$

$$\Rightarrow A^{(1)}_{ij} = \gamma^i_j$$

terms appear twice ... not a big deal

N^2 terms

$$\overset{\uparrow}{\underset{\downarrow}{N}}\ A^{(2)}$$

$$\leftarrow N^2 \rightarrow$$

$$\Rightarrow A^{(2)}_{ij} = \boxed{}\ \gamma^i_{j,k}$$

$$[(j-1)n+k+1]$$

KROENEKER PRODUCT:

$$\vec{X}\otimes\vec{X} = \begin{bmatrix}X_1\vec{X}\\ X_2\vec{X}\\ X_3\vec{X}\\ \vdots\\ X_N\vec{X}\end{bmatrix}$$

$$\frac{d}{dt}\vec{X} = A^{(1)}\vec{X} + A^{(2)}\vec{X}\otimes\vec{X} + B\vec{u} + C\underbrace{\vec{u}\otimes\vec{X}} + D\,\vec{u}\otimes\vec{u}$$

$$\Downarrow$$

$$\begin{bmatrix}u_1\vec{X}\\ u_2\vec{X}\\ \vdots\\ u_M\vec{X}\end{bmatrix} = \begin{bmatrix}[u_1][X_1]\\ [u_1][X_2]\\ \vdots\\ [u_1][X_N]\\ [u_2][X_1]\\ \vdots\\ [u_M][X_M]\end{bmatrix}$$

2.14 Nonlinear Dynamics and Stability

6.581/BE.482

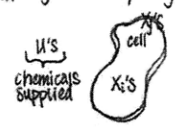
LECTURE 15: NONLINEAR DYNAMICS & STABILITY

THURSDAY
6 APRIL 2006

COMPUTING STEADY-STATES

$$\frac{d}{dt}\vec{X} = A^{(1)}\vec{X} + A^{(2)}\vec{X}\otimes\vec{X} + B^{(1)}\vec{u} + \underbrace{B^{(2)}\vec{u}\otimes\vec{X} + B^{(2)}\vec{u}\otimes\vec{u}}_{\text{ignore for simplicity}}$$

$\vec{u}$ known



KROENECKER PRODUCT:

$$n \overset{p}{\updownarrow} A \otimes m \overset{q}{\updownarrow} B = \overset{\longleftarrow\ pq\ \longrightarrow}{\begin{bmatrix} A_{11}B & A_{12}B & \cdots & A_{1n}B \\ A_{21}B & A_{22}B & \cdots & A_{2n}B \\ \vdots & \vdots & \ddots & \vdots \\ A_{n1}B & A_{n2}B & \cdots & A_{nn}B \end{bmatrix}} \updownarrow nm$$

$$\vec{X}\otimes\vec{X} = \begin{bmatrix} X_1\vec{X} \\ X_2\vec{X} \\ \vdots \\ X_n\vec{X} \end{bmatrix} = \begin{bmatrix} X_1X_1 \\ X_1X_2 \\ \vdots \\ X_1X_n \\ X_2X_1 \\ \vdots \\ X_nX_n \end{bmatrix}$$

REACTIONS ARE IN BALANCE: $X_1 + X_2 \underset{K_r}{\overset{K_f}{\rightleftharpoons}} X_3$

In Steady-State: $\frac{d}{dt}[X_i] = 0$ (concentration)

$$\frac{d}{dt}\vec{X} = 0 \quad (\text{vector case})$$

FIND $\vec{X}$'s S.T. $\rightarrow A^{(1)}\vec{X} + A^{(2)}\vec{X}\otimes\vec{X} + B^{1}\vec{u} = 0$

IS THE STEADY-STATE STABLE?

Suppose $\vec{X}^*$: $A^{(1)}\vec{X}^* + A^{(2)}\vec{X}^*\otimes\vec{X}^* + B^{(1)}\vec{u} = 0$

If $\vec{X}(0) = \vec{X}^* + \vec{\epsilon}$ for small $\vec{\epsilon}$,

does soln $\rightarrow \vec{X}^*$?

$$\frac{d}{dt}\vec{X} = A^{(1)}\vec{X} + A^{(2)}\vec{X}\otimes\vec{X} + B^{(1)}\vec{u}$$

only STABLE steady-states are observable

$\Downarrow$

stable means $\vec{X}^*$ is approached when starting from $\vec{X}^* + \vec{\epsilon}$

CASE: $A^{(2)} = 0 \leftarrow$ NOT Biological

$$\frac{d}{dt}\vec{X}(t) = A\vec{X}(t) + B\vec{u} \leftarrow \text{constant in time}$$

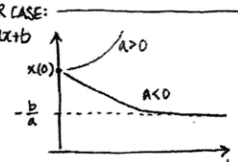
$$A\vec{X}^* + B\vec{u} = 0$$

$$\Rightarrow \vec{X}^* = -A^{-1}B\vec{u}$$

FOR SCALAR CASE:

$$\frac{d}{dt}x = ax + b$$

$$x^* = -\frac{b}{a}$$



observe that stability can only occur if $a<0$

FOR VECTOR CASE:

$$\frac{d}{dt}\vec{X} = A\vec{X} + B\vec{u}$$

$$\Rightarrow \vec{X} = -A^{-1}B\vec{u}$$

In order for this to be stable, what must be said about A?

$\mathbb{R}e\{\text{eigenvalues of } A\} < 0$

BUT, WHY EIGENVALUES?

$$A\vec{S}_i = \lambda\vec{S}_i$$

eigenvalue, eigenvector

$$A[\vec{S}_1\ \vec{S}_2 \cdots \vec{S}_n] = \begin{bmatrix} \lambda_1 & & 0 \\ & \lambda_2 & \\ 0 & & \ddots\ \lambda_n \end{bmatrix}[S]$$

EIGEN (SPECTRAL) DECOMPOSITION:

$$\vec{X}(t) = \alpha_1(t)\vec{S}_1 + \alpha_2(t)\vec{S}_2 + \ldots + \alpha_n(t)\vec{S}_n$$

$$\Rightarrow \vec{X}(t) = [S]\vec{\alpha}(t)$$

$$\frac{d}{dt}\vec{X} = A\vec{X} + B\vec{u}$$

$$\Rightarrow \frac{d}{dt}\vec{\alpha}(t) = \begin{bmatrix} \lambda_1 & & \\ & \lambda_2 & \\ & & \ddots\ \lambda_n \end{bmatrix}\vec{\alpha}(t) + [S]^{-1}B\vec{u}$$

$$\Rightarrow \left\{ \begin{aligned} \frac{d}{dt}\alpha_1 &= \lambda_1\alpha_1 + (S^{-1}B\vec{u})_1 \\ \frac{d}{dt}\alpha_2 &= \lambda_2\alpha_2 + (S^{-1}B\vec{u})_2 \\ &\vdots \\ \frac{d}{dt}\alpha_n &= \lambda_n\alpha_n + (S^{-1}B\vec{u})_n \end{aligned} \right\}$$

$\mathbb{R}e(\lambda_i) < 0\ \forall i \Rightarrow \alpha_i$'s don't blow up

CONSIDER A GENERAL, DYNAMIC, NONLINEAR SYSTEM OF EQUATIONS

$$\frac{d}{dt}\vec{X} = F(\vec{X}) = \begin{bmatrix} f_1(X_1, X_2, \ldots, X_n) \\ f_2(X_1, X_2, \ldots, X_n) \\ \vdots \\ f_n(X_1, X_2, \ldots, X_n) \end{bmatrix}$$

$$F(\vec{X}) = A^{(1)}\vec{X} + A^{(2)}\vec{X}\otimes\vec{X} + B\vec{u}$$

STEADY-STATE EQN:

Find $\vec{X}^*$ s.t. $F(\vec{X}^*) = 0$

$$F(\vec{X} + \vec{\epsilon}) = F(\vec{X}) + J_F(\vec{X})\vec{\epsilon}$$

Jacobian of $F(\vec{X})$

(for scalar case: $f(x+\epsilon) \cong f(x) + \frac{\partial f(x)}{\partial x}\epsilon + \text{H.O.T.}$)

SYSTEM ABOUT $\vec{X}^* \Rightarrow$

$$\frac{d}{dt}\vec{X} = F(\vec{X}) \Rightarrow \frac{d}{dt}(\vec{X}^* + \vec{\epsilon}) = F(\vec{X}^* + \vec{\epsilon})$$

$$\Rightarrow \frac{d}{dt}\vec{\epsilon} \cong J_F(\vec{X}^*)\vec{\epsilon}$$

$$J_F(\vec{X})\vec{\epsilon} = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \frac{\partial f_2}{\partial x_1} & \frac{\partial f_2}{\partial x_2} & \cdots & \frac{\partial f_2}{\partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial f_n}{\partial x_1} & \frac{\partial f_n}{\partial x_2} & \cdots & \frac{\partial f_n}{\partial x_n} \end{bmatrix} \begin{bmatrix} \epsilon_1 \\ \epsilon_2 \\ \vdots \\ \epsilon_n \end{bmatrix}$$

$\vec{X}^*$ is a stable, steady-state IF $\mathbb{R}e\{\text{eigenvalues of } J_F(\vec{X}^*)\} < 0$

CASE: $A^{(2)} \neq 0$ (Biological Case)

$$F(\vec{x}) = A^{(1)}\vec{x} + A^{(2)}\vec{x}\otimes\vec{x} + B\vec{u}$$

$$J_F(\vec{x}) = A^{(1)} + A^{(2)}(I\otimes\vec{x}) + A^{(2)}(\vec{x}\otimes I)$$

STEADY-STATE PROBLEM:

① FIND $\vec{x}^*$ s.t. $A^{(1)}\vec{x}^* + A^{(2)}\vec{x}^*\otimes\vec{x}^* + B\vec{u} = 0$

② VERIFY $\lambda(A^{(1)} + A^{(2)}(I\otimes\vec{x}^* + \vec{x}^*\otimes I))$ have negative real parts

use NEWTON'S METHOD to solve

Problem: Find $\vec{x}^*$ s.t. $F(\vec{x}^*) = 0$

FOR THE SCALAR CASE:

$f(x^*) = 0$

use Taylor's Expansion for nonlinear f:

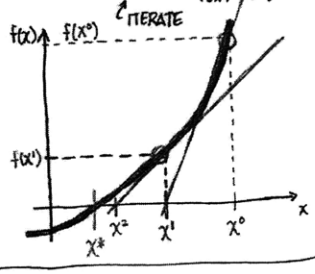
$$\underbrace{f(x^*)}_{0} = f(x) + \frac{df(x)}{dx}(x - x^*) + H.O.T.$$

$$\Rightarrow \frac{df}{dx}(x - x^*) = -f(x)$$

GUESS x^0

COMPUTE x^1: $x^1 - x^0 = -\left(\frac{\partial f}{\partial x}\right)^{-1} f(x^0)$

ITERATE



$\vec{x}^*$ s.t. $F(\vec{x}^*) = 0$

guess at $\vec{x}^0$

$$\left\{\begin{array}{l} J_F(\vec{x}^0)(\vec{x}^1 - \vec{x}^0) = -F(\vec{x}^0) \Rightarrow \vec{x}^1 \\ J_F(\vec{x}^1)(\vec{x}^2 - \vec{x}^1) = -F(\vec{x}^1) \Rightarrow \vec{x}^2 \\ \vdots \end{array}\right\}$$

where

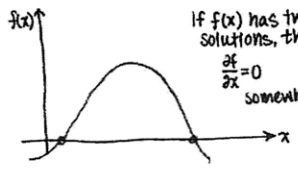
$$F(\vec{x}^0) = A^{(1)}\vec{x}^0 + A^{(2)}\vec{x}^0\otimes\vec{x}^0 + B\vec{u}$$

$$J_F(\vec{x}^0) = A^{(1)} + A^{(2)}(I\otimes\vec{x}^0 + \vec{x}^0\otimes I)$$

thus

$$\left\{\begin{array}{l} \left[A^{(1)} + A^{(2)}(I\otimes\vec{x}^0 + \vec{x}^0\otimes I)\right](\vec{x}^1 - \vec{x}^0) = -F(\vec{x}^0) \\ \left[A^{(1)} + A^{(2)}(I\otimes\vec{x}^1 + \vec{x}^1\otimes I)\right](\vec{x}^2 - \vec{x}^1) = -F(\vec{x}^1) \\ \vdots \end{array}\right\}$$

ONCE AGAIN, FOR THE SCALAR CASE:



If $f(x)$ has two solutions, then $\frac{\partial f}{\partial x} = 0$ somewhere

SO, FOR $F(\vec{x}) = 0$ TO HAVE MULTIPLE SOLNS

$\Rightarrow J_F(\vec{x})\vec{v} = 0$ For some $\vec{x}$, there is a flat direction

IF $J_F(\vec{x})$ IS NONSINGULAR FOR ALL $\vec{x}$

THEN $F(\vec{x}) = 0$ HAS ONE SOLUTION

If $A^{(1)} + A^{(2)}(I\otimes\vec{x} + \vec{x}\otimes I)$ is always NONSINGULAR

Then STEADY STATE SOLUTION is UNIQUE

2.15 Steady-State Problems

6.581/BE.482

LECTURE 16: STEADY-STATE PROBLEMS

TUESDAY 11 APRIL 2006

SIMPLIFIED SYSTEM

$$\frac{d}{dt}\vec{X} = A^{(1)}\vec{X} + A^{(2)}(\vec{X}\otimes\vec{X}) + B\vec{u}$$

STEADY STATE

Solve $\underbrace{A^{(1)}\vec{X} + A^{(2)}(\vec{X}\otimes\vec{X}) + B\vec{u}}_{F(\vec{X})=0} = 0$

NEWTON'S METHOD

Guess at $\vec{X}^0$

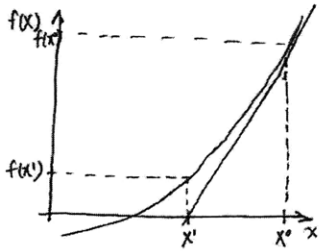
Evaluate $F(\vec{X}^0) \neq 0$

$$\simeq F(\vec{X}^*) + J_F(\underbrace{\vec{X}^0 - \vec{X}^*}_{\Delta X})$$

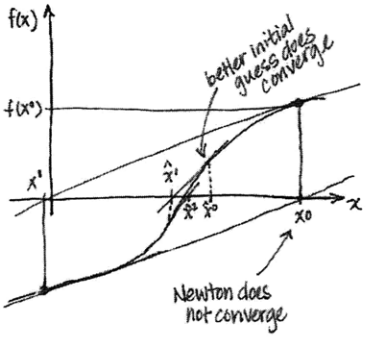
Use Approximation

$$J_F(\vec{X}^0)\Delta X = -F(\vec{X})$$

update $\vec{X}' = \vec{X}^0 + \Delta X$



$$J_F(X) = A^{(1)} + A^{(2)}(\vec{X}\otimes I + I\otimes\vec{X})$$



NEWTON PROPERTIES:

1. Converges if you are close enough
2. Converges very fast

} $J_F(\vec{X})$ is nonsingular

Original F:

$$F(\vec{X}) = A^{(1)}\vec{X} + A^{(2)}\vec{X}\otimes\vec{X} + B\vec{u}$$

Simpler F: $\vec{u}=0$

$$F(\vec{X},0) = A^{(1)}\vec{X} + A^{(2)}(\vec{X}\otimes\vec{X})$$

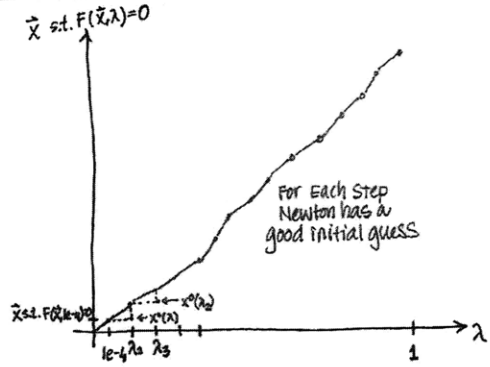
$\vec{X}$ s.t. $F(\vec{X},0)=0$ $\vec{u}=0$

Continuation scheme:

$$F(\vec{X},\lambda) = A^{(1)}\vec{X} + A^{(2)}\vec{X}\otimes\vec{X} + \lambda B\vec{u}$$

$$F(\vec{X},0) = A^{(1)}\vec{X} + A^{(2)}\vec{X}\otimes\vec{X} \Rightarrow \vec{X}=0 \;\&\; \lambda=0$$

(original problem: $F(\vec{X},1)=0$)



WANTED INFORMATION:

- How many stable steady-states
- What are the stable steady-states

NEWTON GIVES:

- Maybe one steady-state

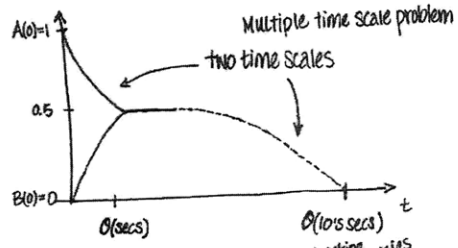
Example 1.

$$A \underset{1}{\overset{1}{\rightleftharpoons}} B$$

degrades 0.1 0.1 degrades

$$\frac{d[A]}{dt} = \underbrace{-1\cdot[A]}_{\text{degradation}} + 1[B]$$

$$\frac{d[B]}{dt} = 1[A] - 1\cdot[B]$$



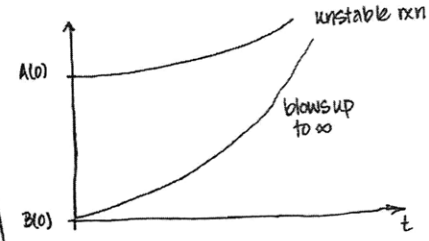
Example 2.

$$\overset{3}{\circlearrowleft} A \underset{1}{\overset{4}{\rightleftharpoons}} B$$

Not tracking every species

$$\frac{d}{dt}[A] = (-4+3)[A] + 1[B] = -1[A] + 1[B]$$

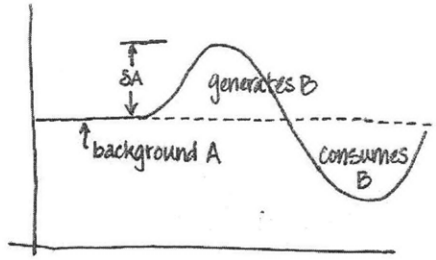
$$\frac{d}{dt}[B] = 4[A] - 1[B]$$

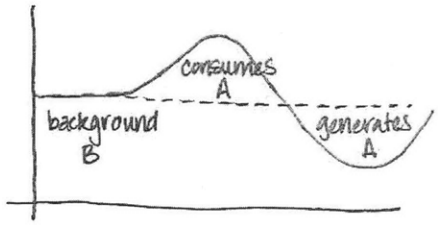


Example 3. Oscillatory System (nonphysical)

$$\frac{d}{dt}[A] = \qquad -1[B]$$

$$\frac{d}{dt}[B] = +1\,[A]$$





$$\frac{d}{dt}\vec{X} = \begin{bmatrix} -1.01 & 1 \\ 1 & -1.01 \end{bmatrix}\begin{bmatrix} X_1 \\ X_2 \end{bmatrix}$$

$$A \underset{1}{\overset{1}{\rightleftharpoons}} B \quad 0.01 \quad 0.01$$

$$\begin{bmatrix} 1 \\ -1 \end{bmatrix} \quad \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

$$\lambda_i = -2.01, -0.01$$

Fast Mode, Slow Mode

$$\frac{d}{dt}\vec{X} = \begin{bmatrix} -1 & 1 \\ 4 & -1 \end{bmatrix}\begin{bmatrix} X_1 \\ X_2 \end{bmatrix}$$

$$3 \circlearrowleft A \underset{1}{\overset{4}{\rightleftharpoons}} B$$

$$\lambda_1 = -3 \quad \underbrace{\lambda_2 = 1}_{\text{unstable}}$$

$$\frac{d}{dt}\vec{X} = \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix}\begin{bmatrix} X_1 \\ X_2 \end{bmatrix}$$

δA ? δB

$\lambda_1 = j \quad \lambda_2 = -j$

oscillatory soln w/o decay $\Rightarrow$ purely imaginary eigenvalues

$\longrightarrow$ go to slides

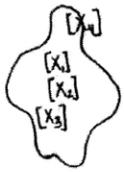
2.16 Parameter Fitting and Estimation

6.581/BE.482

LECTURE 17: PARAMETER FITTING & ESTIMATION

THURSDAY 13 APRIL 2006

Dynamical Net Cell



known:
$[u_1]$
$[u_2]$

$$\dot{\vec{X}} = A^{(1)}\vec{X} + A^{(2)}(\vec{X}\otimes\vec{X}) + B^{(1)}\vec{u} + \underbrace{B^{(2)}(\vec{u}\otimes\vec{X}) + B^{(3)}(\vec{u}\otimes\vec{u})}_{\text{ignored for simplicity}}$$

$$\frac{d}{dt}\begin{bmatrix}[X_1]\\ \vdots \\ [X_n]\end{bmatrix}$$

measurements

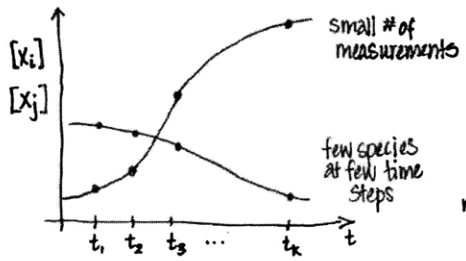
$$\begin{bmatrix}\vec{X}_M(t_1)\\ \vdots \\ \vec{X}_M(t_k)\end{bmatrix} \quad \left.\right\} Mk \quad \begin{bmatrix}Ce^{At_1}\\ \vdots \\ Ce^{At_k}\end{bmatrix}\begin{bmatrix}\vec{X}(0)\end{bmatrix} + \begin{bmatrix}\int_0^{t_1} e^{A(t_1-\tau)}Bu(\tau)d\tau \\ \vdots \\ \int_0^{t_k} e^{A(t_k-\tau)}Bu(\tau)d\tau\end{bmatrix}$$

M = # of measurements

inputs

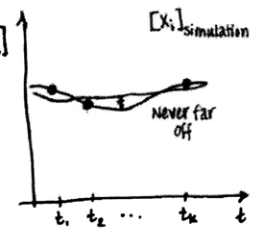
COMPARE MEASURED DATA TO THE SIMULATION VIEW

$$\begin{bmatrix}C\vec{X}(t_1)\\ \vdots \\ C\vec{X}(t_k)\end{bmatrix} \qquad \begin{bmatrix}\vec{b}(t_1)\\ \vdots \\ \vec{b}(t_k)\end{bmatrix}$$



$$\min_{\vec{X}_0} \text{measure}\left\{\underbrace{\vec{Y}}_{\begin{bmatrix}X_M(t_1)\\ \vdots \\ X_M(t_k)\end{bmatrix} - \begin{bmatrix}\vec{b}(t_1)\\ \vdots \\ \vec{b}(t_k)\end{bmatrix}} - \underbrace{L\vec{X}_0}_{\begin{bmatrix}Ce^{At_1}\\ \vdots \\ Ce^{At_k}\end{bmatrix}}\right\}$$

$$\min_{\vec{X}_0} \max_i |(\vec{Y}-L\vec{X}_0)_i|$$



$$\underbrace{\min_{\vec{X}_0} \sum_i (\vec{Y}-L\vec{X}_0)_i^2}_{\text{LEAST SQUARES}}$$

$$\min_{\vec{X}_0} \underbrace{\underbrace{(\vec{Y}-L\vec{X}_0)^T}_{\vec{e}^T}\underbrace{(\vec{Y}-L\vec{X}_0)}_{\vec{e}}}_{f(\vec{X}_0)} \quad \vec{e}^T\vec{e} = \sum \vec{e}_i^2$$

LINEAR LEAST SQUARES

- Dynamical system is linear
  $\dot{\vec{X}} = A^{(1)}\vec{X} + B\vec{u}$ ...
- Unknowns are initial conditions
- Measured quantities were linear functions of $\vec{X}$
  $C\vec{X}(t)$

What was $\vec{X}(0)$? (Initial Condition)

ESTIMATING INITIAL CONDITIONS

1.) START by looking at the linear case
$$\dot{\vec{X}} = A^{(1)}\vec{X} + B^{(1)}\vec{u}$$

2.) NONLINEAR CASE
$$\dot{\vec{X}} = A^{(1)}\vec{X} + A^{(2)}(\vec{X}\otimes\vec{X}) + B^{(1)}\vec{u}$$

LINEAR CASE

$$\dot{\vec{X}} = A\vec{X} + B\vec{u}$$

$$\vec{X}(t) = \underbrace{e^{At}}_{\text{MATRIX}}x(0) + \int_0^t e^{A(t-\tau)}Bu(\tau)d\tau$$

Scalar case:
$$\dot{x} = ax + \cancel{bu}$$
$$x(t) = e^{at}x(0)$$
$$\dot{x} = ax + bu$$
$$x(t) = e^{at}x(0) + \int_0^t e^{a(t-\tau)}bu(\tau)d\tau$$

$$[X_i(t_k)] = 1 \left[\underbrace{\overset{n}{0 \cdots 0\ 1\ 0 \cdots 0}}_{C}\right]\begin{bmatrix}X_1(t_k)\\ \vdots \\ X_n(t_k)\end{bmatrix}$$

i^{th} column

$$\begin{bmatrix}[X_i(t_k)]\\ [X_j(t_k)]\end{bmatrix} = 2\left[\underbrace{\overset{n}{\begin{matrix}0\cdots010\cdots000\cdots0\\ 0\cdots000\cdots010\cdots0\end{matrix}}}_{C}\right][\vec{X}(t_k)]$$

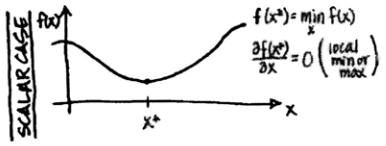
ith jth

ABSTRACT PROBLEM

$$\underbrace{f(\vec{X})}_{\text{cost function}} \quad f: \mathbb{R}^n \to \mathbb{R} \quad f: \text{vector} \to \text{scalar}$$

$$\min_{\vec{X}} f(\vec{X}) \leftarrow \text{optimization problem}$$

SCALAR CASE



$$f(x^*) = \min_x f(x)$$
$$\frac{\partial f(x^*)}{\partial x} = 0 \quad (\text{local min or max})$$

VECTOR CASE

vector function or a vector

$$\frac{\partial f}{\partial x} = \begin{bmatrix} \frac{\partial f}{\partial x_1} \\ \vdots \\ \frac{\partial f}{\partial x_n} \end{bmatrix} = 0 \quad \left(\text{local min or max}\right)$$

For some functions f

$\exists\, x^*$ s.t. $\frac{\partial f(x^*)}{\partial x} = 0$

means x^* is the global optimum

$$\min_{\vec{x}_0} (\vec{Y}-L\vec{X}_0)^T(\vec{Y}-L\vec{X}_0) = f(\vec{X}_0)$$

$$\frac{\partial f}{\partial \vec{X}_0} = \frac{\partial}{\partial \vec{X}_0}\left[\vec{Y}^T\vec{Y} - (L\vec{X}_0)^T\vec{Y} - \vec{Y}^T(L\vec{X}_0) + (L\vec{X}_0)^T(L\vec{X}_0)\right]$$

equal

$$= \frac{\partial}{\partial \vec{X}_0}\left[\cancel{\vec{Y}^T\vec{Y}}\right] - \frac{\partial}{\partial \vec{X}_0}\left[2\vec{Y}^T(L\vec{X}_0)\right] + \frac{\partial}{\partial \vec{X}_0}\left[(L\vec{X}_0)^T(L\vec{X}_0)\right]$$

0

$$= -2L^T\vec{Y} + 2L^TL\vec{X}_0 = 0$$

$$\Rightarrow \underbrace{L^TL\vec{X}_0 = L^T\vec{Y}}_{\text{NORMAL EQNS}}$$

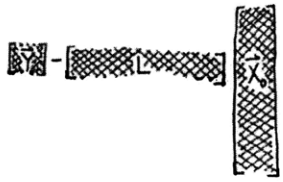
Lots of data

$$\begin{array}{c} \\ mk \end{array}\overset{\longleftarrow n \longrightarrow}{\begin{bmatrix} Ce^{At_1} \\ Ce^{At_2} \\ \vdots \\ Ce^{At_k} \end{bmatrix}} \begin{bmatrix} \vec{X}_0 \end{bmatrix}$$

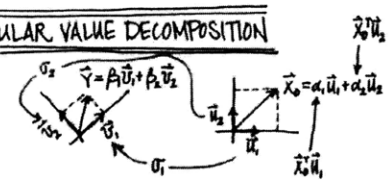
Limited data → typical

$$\begin{bmatrix} Ce^{At_1} \\ \vdots \\ Ce^{At_k} \end{bmatrix} \begin{bmatrix} \vec{X}_0 \end{bmatrix}$$

$$\underbrace{\begin{bmatrix} L^T \end{bmatrix}\overset{\longleftarrow n \longrightarrow}{\begin{bmatrix} \uparrow \\ mk \\ \downarrow \end{bmatrix} L}}_{n \times n} \begin{bmatrix} \vec{X}_0 \end{bmatrix} = \underbrace{\begin{bmatrix} L^T \end{bmatrix}\begin{bmatrix} \vec{Y} \end{bmatrix}}_{n \text{ length vector}}$$



SINGULAR VALUE DECOMPOSITION



$$\begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \vdots \\ \alpha_{mk} \end{bmatrix} = \begin{bmatrix} \longleftarrow & u_1^T & \longrightarrow \\ & \vdots & \\ \longleftarrow & u_{mk}^T & \longrightarrow \end{bmatrix} \begin{bmatrix} \vec{X}_0 \end{bmatrix}$$

$$\begin{bmatrix} \uparrow & & \uparrow \\ \vec{V}_1 & \cdots & \vec{V}_{mk} \\ \downarrow & & \downarrow \end{bmatrix} \begin{bmatrix} \sigma_1 & & \\ & \ddots & \\ & & \sigma_{mk} \end{bmatrix} \begin{bmatrix} \alpha_1 \\ \vdots \\ \alpha_{mk} \end{bmatrix}$$

$$\vec{Y} = \overbrace{\underbrace{\begin{bmatrix} \uparrow & & \uparrow \\ \vec{V}_1 & \cdots & \vec{V}_{mk} \\ \downarrow & & \downarrow \end{bmatrix}}_{\substack{\vec{V}_i^T\vec{V}_j = 0 \\ \text{for } i \neq j \\ \text{(orthogonal coordinate system)}}} \begin{bmatrix} \sigma_1 & & \\ & \ddots & \\ & & \sigma_{mk} \end{bmatrix} \underbrace{\begin{bmatrix} \longleftarrow & \vec{U}_1 & \longrightarrow \\ & \vdots & \\ \longleftarrow & \vec{U}_{mk} & \longrightarrow \end{bmatrix}}_{\substack{\vec{U}_i^T\vec{U}_j = 0 \\ \text{for } i \neq j}}}^{\text{SVD } L} \begin{bmatrix} \vec{X}_0 \end{bmatrix}$$

$$\begin{bmatrix} \uparrow & & \uparrow \\ \vec{U}_1 & \cdots & \vec{U}_{mk} \\ \downarrow & & \downarrow \end{bmatrix} \begin{bmatrix} \sigma_1 & & \\ & \ddots & \\ & & \sigma_{mk} \end{bmatrix} \begin{bmatrix} \longleftarrow & \vec{V}_1^T & \longrightarrow \\ & \vdots & \\ \longleftarrow & \vec{V}_{mk}^T & \longrightarrow \end{bmatrix} \vec{Y} = \vec{X}_0$$

2.17 Parameter Estimation; Robustness, Fragility, Control

6.581/BE 482

THURSDAY 20 APRIL 2006

LECTURE 18: PARAMETER ESTIMATION; ROBUST, FRAGILITY, CONTROL

BRIEF SYNOPSIS:

BIOCHEMICAL NETWORKS
- information processing
- decision & control
- effector function

λ PHAGE
- 2 states; long-term stability
- multiple interactions among proteins & binding sites

DIFFERENTIAL EQUATION FORMULATION:

$$\frac{d\vec{X}}{dt} = F(\vec{X},\vec{u}) = A^{(1)}\vec{X} + A^{(2)}(\vec{X}\otimes\vec{X}) + B\vec{u} + \underbrace{C(\vec{u}\otimes\vec{X}) + D(\vec{u}\otimes\vec{u})}_{\text{generally don't write}}$$

For steady states:

$$\frac{d\vec{X}}{dt} = 0 \Rightarrow F(\vec{X},\vec{u}) = 0$$

Solve using NEWTON'S METHOD:

$$\frac{dF(\vec{X},\vec{u})}{d\vec{X}} = \nabla_{\vec{X}} F \equiv J_F(\vec{X})$$

$$F(\vec{X}^1,\vec{u}) = F(\vec{X}^0,\vec{u}) + \underbrace{\frac{dF}{d\vec{X}}(\vec{X}^0,\vec{u})}_{J_F(\vec{X}^0)}(\vec{X}^1-\vec{X}^0) + \cancel{H.O.T.}$$

set $F(\vec{X}^1,\vec{u}) = 0 \Rightarrow J_F(\vec{X}^0)(\vec{X}^1-\vec{X}^0) = -F(\vec{X}^0,\vec{u})$

solve for $\vec{X}^1$

iterate: $J_F(\vec{X}^1)(\vec{X}^2-\vec{X}^1) = -F(\vec{X}^1,\vec{u})$

⋮

RUN SIMULATIONS:
- Forward Euler – explicit
- Backward Euler – implicit (iterative)

PARAMETER SENSITIVITIES: $\frac{d\vec{X}}{d(\Delta A)}$

$$\min_{\Delta A} f(\Delta A) = \min_{\Delta A} \sum_{j=1}^{L} \left[C^T\vec{X}^j(\Delta A) - \underbrace{C^T\vec{X}^{j,\text{measured}}}_{[00001000]}\right]^2$$

(L = # of experiments)

OPTIMIZE:

- GRADIENT DESCENT

$$\Delta A' = \Delta A^0 + \beta \nabla_{\Delta A} f$$

pick stepsize β s.t. $\min_{\beta} f(\Delta A^0 + \beta\nabla_{\Delta A} f)$

This method does not work well

- ADJOINT OPTIMIZATION

Set gradient to zero

$$\frac{\partial}{\partial(\Delta A)}\left[C^T\vec{X}^j(\Delta A) - C^T\vec{X}^{j,m}\right]^2$$

$$= 2\underbrace{\left[C^T\vec{X}^j(\Delta A) - C^T\vec{X}^{j,m}\right]}_{\text{straightforward}}\underbrace{\left[\frac{\partial}{\partial(\Delta A)} C^T\vec{X}^j(\Delta A)\right]}_{\text{difficult}}$$

→ sensitivities

$$\frac{\partial}{\partial(\Delta A)} C^T\vec{X}^j(\Delta A) = {}_1\overset{n}{\boxed{C^T}}\,{}_n\overset{q}{\boxed{\frac{\partial\vec{X}^j(\Delta A)}{\partial(\Delta A)}}}$$

n = # of concentrations
q = # of parameters

$$\frac{dF}{d(\Delta A)} = 0 \Rightarrow \underbrace{\frac{\partial F}{\partial(\Delta A)} + \frac{\partial F}{\partial\vec{X}}\cdot\frac{\partial\vec{X}}{\partial(\Delta A)}}_{(\star)} = 0 \Rightarrow {}_n\overset{n}{\boxed{\frac{\partial F}{\partial\vec{X}}}}\,{}_n\overset{q}{\boxed{\frac{\partial\vec{X}}{\partial(\Delta A)}}} = -{}_n\overset{q}{\boxed{\frac{\partial F}{\partial(\Delta A)}}}$$

& solve for this term & plug-in

HARDER TO SOLVE

Rather than solve the previous problem, SOLVE its DUAL:

$$\frac{\partial}{\partial(\Delta A)} C^T\vec{X}^j(\Delta A) = {}_1\overset{n}{\boxed{Z^T}}\,{}_n\overset{q}{\boxed{\left(\frac{\partial F}{\partial(\Delta A)}\right)}}$$

where Z is obtained from:

$${}_n\overset{n}{\boxed{\left(\frac{\partial F}{\partial\vec{X}}\right)^T}}\,{}_n\overset{1}{\boxed{Z}} = -{}_n\overset{1}{\boxed{C}}$$

EASIER TO SOLVE

SHOW THAT THE DUAL IS EQUIVALENT:

$$C^T\frac{\partial\vec{X}^j(\Delta A)}{\partial(\Delta A)} = \overbrace{C^T\left(-\frac{\partial F}{\partial\vec{X}}\right)^{-1}}^{Z^T}\left(\frac{\partial F}{\partial(\Delta A)}\right)$$

FROM ★

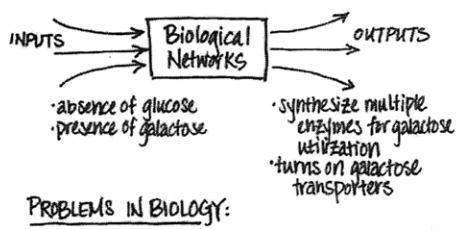
$$\Rightarrow \frac{\partial}{\partial(\Delta A)} C^T\vec{X}^j(\Delta A) = Z^T\frac{\partial F}{\partial(\Delta A)}$$

$$Z^T = C^T\left(-\frac{\partial F}{\partial\vec{X}}\right)^{-1}$$

$$\Rightarrow Z^T\left(\frac{\partial F}{\partial\vec{X}}\right) = -C^T$$

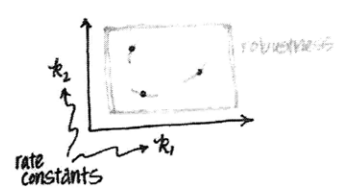
$$\Rightarrow \left(\frac{\partial F}{\partial\vec{X}}\right)^T Z = -C \quad \square$$

- ★Finding steady states
- ★Running simulations
- Model construction
  - parameter estimation can also be used in design mode
  - parameter sensitivities

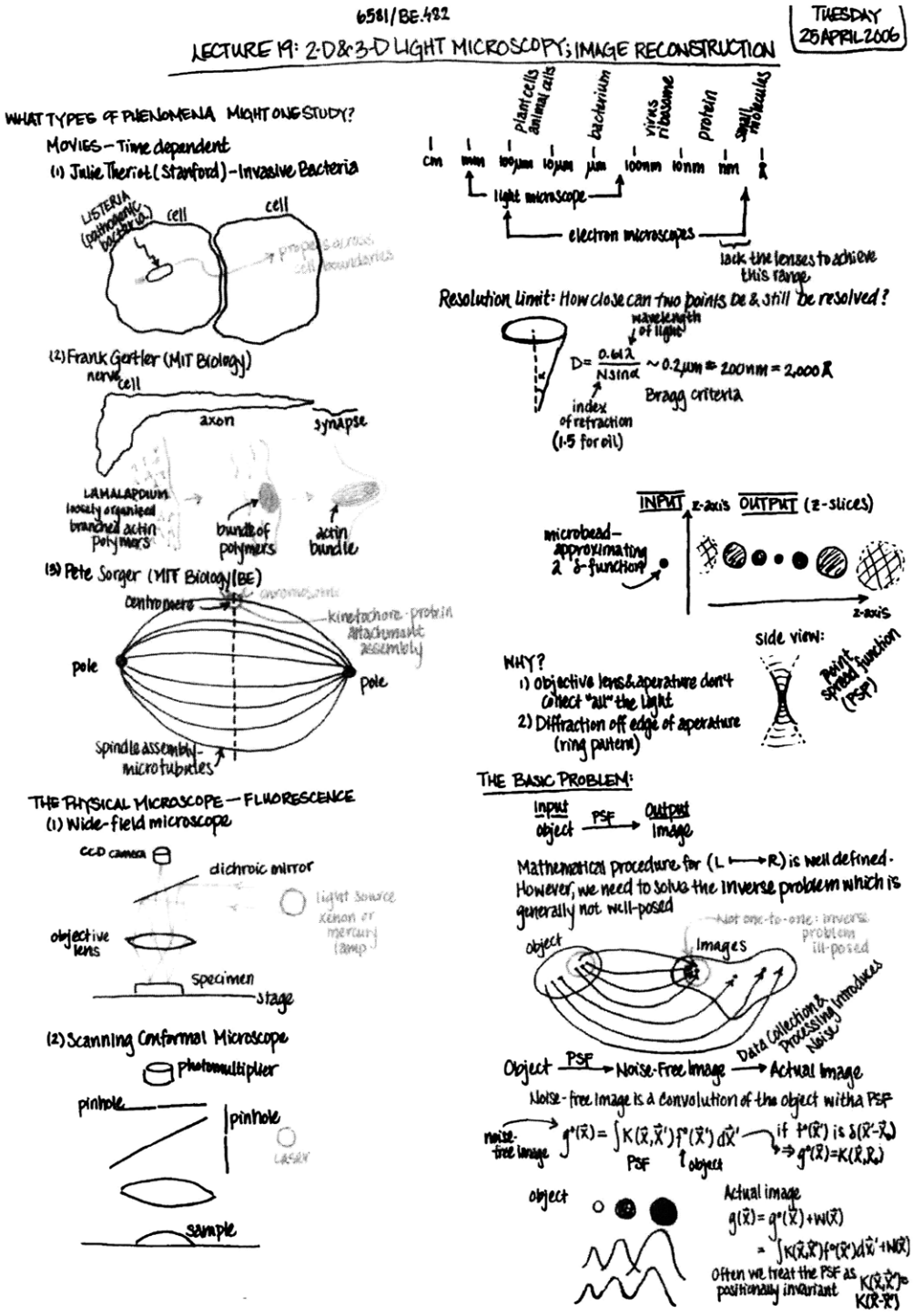


PROBLEMS IN BIOLOGY:

- Difficulty in maintaining constant concentration
  - cell volume changes
  - small counts make uniformity difficult
- Highly variable environment
  - Temperature
  - Humidity
- Certain functions persist across all stages of development, cell cycle, tissue types
- Evolve



2.18 2-D and 3-D Light Microscopy; Image Reconstruction



2.19 Deconvolution

6.581 / BE.492

LECTURE 20: DECONVOLUTION

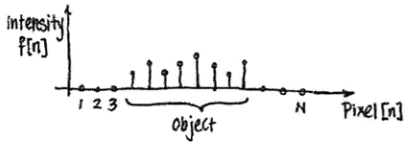
THURSDAY 27 APRIL 2006

DECONVOLUTION

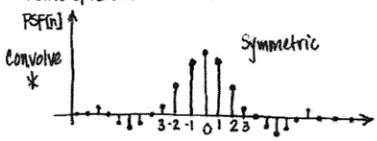
1.) Point spread function representation
2.) Straightforward deconvolution
3.) Handling noise (SVD)

1-D

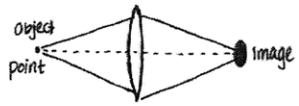
Object (black & white)



Point Spread Function (PSF)



convolve $*$



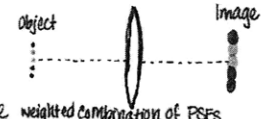
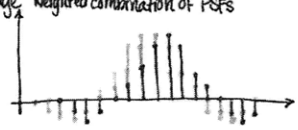


Image weighted combination of PSFs



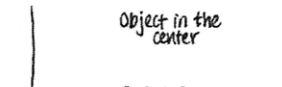
$$\begin{bmatrix} \vec{G}[1] \\ \vdots \\ \vec{G}[k] \\ \vdots \\ \vec{G}[N] \end{bmatrix} = \begin{bmatrix} PSF[0] & PSF[-1] & & & PSF[1] & PSF[0] \\ \cdots & PSF[k] & PSF[k-1] & PSF[k] & PSF[k+1] & PSF[k-2] \cdots \\ & & & & & \end{bmatrix} \begin{bmatrix} \vec{F}[1] \\ \vdots \\ \vec{F}[N] \end{bmatrix}$$

Image

assume periodicity

Object $\begin{bmatrix} \vdots \\ f[N-1] \\ f[N] \end{bmatrix}$ pretend object repeats periodically

K-1's contribution at K is the +1 contribution from K-1 ⇒ PSF[1]



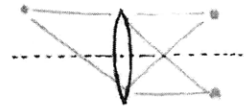
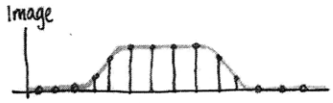


image object

$$\vec{G}^o = \underbrace{H}_{\text{Matrix}} \vec{F}^o$$



$$\vec{G}^o = H\vec{F}^o \qquad \vec{F} = H^{-1}\vec{G}^o$$

image pixels — object pixels — reconstructed image

When the smallest bit of noise is added to the image & then reconstructed, the resulting reconstructed image is garbage ⇒ very fragile

NOISE

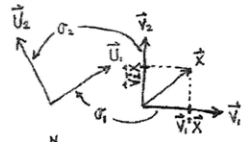
$$\vec{G} = \vec{G}^o + \underbrace{\vec{W}}_{\text{vector of noise}} = H\vec{F}^o + \vec{W}$$

$$\vec{F}^R = H^{-1}(H\vec{F}^o + \vec{W}) = \vec{F}^o + \underbrace{H^{-1}\vec{W}}_{\text{How large?}}$$

$$H = U\Sigma V^T$$

output coordinate system — input coordinate system

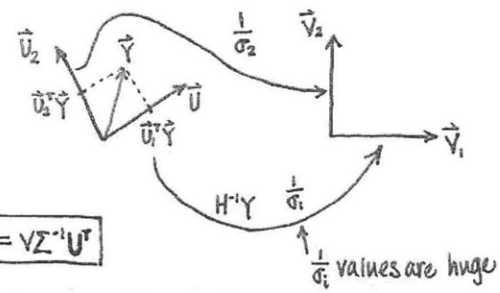
$$\begin{bmatrix} \sigma_1 & & & \\ & \sigma_2 & & \\ & & \ddots & \\ & & & \sigma_n \end{bmatrix}$$



$$HX = \sum_{i=1}^{N} \vec{U}_i \sigma_i \vec{V}_i^T X$$

σ_i values are small

$H = U\Sigma V^T$

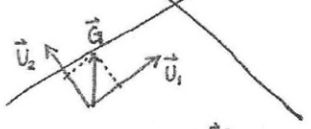


$$\boxed{H^{-1} = V\Sigma^{-1}U^T}$$

$$\vec{G}^o = H\vec{F}_o = \sum_i \vec{u}_i \sigma_i V_i^T \vec{F}^o = \sigma_1 \vec{V}_1^T \vec{F}^o \begin{bmatrix} \vdots \\ \vec{u}_1 \\ \vdots \end{bmatrix} + \sigma_2 \vec{V}_2^T \vec{F}^o \begin{bmatrix} \vdots \\ \vec{u}_2 \\ \vdots \end{bmatrix} + \ldots + \sigma_N \vec{V}_N^T \vec{F}^o \begin{bmatrix} \vdots \\ \vec{u}_N \\ \vdots \end{bmatrix}$$

<u>Perfect Reconstruction</u> (PR)

$$\cancel{\vec{F}^{PR} = V_1^T \vec{F}^o [\vec{u}_1] + V_2^T \vec{F}^o [\vec{u}_2] + \ldots}$$



$$\vec{G} = \vec{G}^o + \vec{w} = \overbrace{\sum_i \vec{u}_i \sigma_i \vec{V}_i^T \vec{F}^o}^{H\vec{F}^o} + \vec{w}$$

$$\vec{F}^R = \underbrace{\sum_i \vec{V}_i \frac{1}{\sigma_i} \vec{U}_i^T \vec{G}}_{H^{-1}\vec{G}}$$

$$\vec{F}^R = \sum_i \vec{V}_i \frac{1}{\sigma_i} \left(\sigma_i \vec{V}_i^T \vec{F}^o + u_i^T \vec{w} \right)$$

$$= \sum_i \left[\underbrace{\overbrace{\vec{V}_i \vec{V}_i^T \vec{F}^o}^{\text{"SIGNAL"}}}_{\text{perfect reconstruction}} + \overbrace{\vec{V}_i \frac{1}{\sigma_i} (u_i^T \vec{w})}^{\text{"NOISE"}} \right]$$

$$\vec{F}^R = \left(\vec{V}_1 \vec{V}_1^T \vec{F}^o + \vec{V}_1 \frac{1}{\sigma_1} \vec{U}_1^T \vec{w} \right) + \vec{V}_2 \left(\vec{V}_2^T \vec{F}^o + \frac{1}{\sigma_2} \vec{U}_2^T \vec{w} \right) + \ldots + \vec{V}_N \left(\vec{V}_N^T \vec{F}^o + \frac{1}{\sigma_N} \vec{U}_N^T \vec{w} \right)$$

$\rightarrow$ Remove terms with small σ

2.20 Deconvolution II

6.581/BE.482
LECTURE 21: DECONVOLUTION II

TUESDAY
2 MAY 2006

OUTLINE

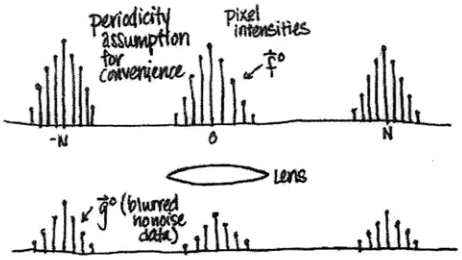
1) Reminder about SVD
2) Fourier (Freq. Domain) View
3) 2D Case

REMINDER

$$\vec{g} = H\vec{f}^{\,o} + \vec{w}$$

$\vec{g}$: data; H: effects of imaging system; $\vec{f}^{\,o}$: object; $\vec{w}$: noise

IN 1D



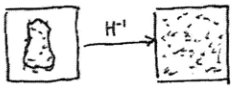
$$\begin{bmatrix} g^o[1] \\ \vdots \\ g^o[N] \end{bmatrix} = \begin{bmatrix} h[0] & h[N-1] & h[N-2] & \cdots & h[2] & h[1] \\ h[1] & h[0] & h[N-1] & \cdots & h[3] & h[2] \\ h[2] & h[1] & h[0] & \cdots & h[4] & h[3] \\ \vdots & \vdots & & \ddots & \vdots & \vdots \\ h[N-1] & h[N-2] & h[N-3] & \cdots & h[1] & h[0] \end{bmatrix} \begin{bmatrix} f^o[1] \\ \vdots \\ f^o[N] \end{bmatrix}$$

CIRCULENT MATRIX

1st DECONVOLUTION IDEA

$$\vec{f}^{\,BE} = H^{-1}\vec{g} = \vec{f}^{\,o} + H^{-1}\vec{w}$$

(bad estimate)



USING SVD

$$H = U \begin{bmatrix} \sigma_1 & & \\ & \ddots & \\ & & \sigma_n \end{bmatrix} V^T$$

$$\begin{bmatrix} \uparrow & & \uparrow \\ \vec{u}_1 & \cdots & \vec{u}_n \\ \downarrow & & \downarrow \end{bmatrix}$$

orthonormal (define coordinate system)

$$\vec{g} = \sum \vec{u}_i\, \sigma_i (\vec{v}_i^{\,T}\vec{f}^{\,o}) + \vec{w}$$

$$\vec{f}^{\,BE} = \underbrace{\sum \frac{1}{\sigma_i}\vec{v}_i(\vec{u}_i^{\,T}\vec{g})}_{H^{-1}\vec{g}}$$

$$\vec{f}^{\,BE} = \sum \vec{v}_i\left[(\vec{v}_i^{\,T}\vec{f}^{\,o}) + \frac{1}{\sigma_i}\vec{u}_i^{\,T}\vec{w}\right]$$

ESTIMATER

α_i = either zero or one

$$\vec{f}^{\,SE} = \alpha_1\left(\vec{v}_1^{\,T}\vec{f}^{\,o} + \frac{1}{\sigma_1}\vec{u}_1^{\,T}\vec{w}\right)\vec{v}_1 + \alpha_2\left(\overbrace{\vec{v}_2^{\,T}\vec{f}^{\,o}}^{\text{image}} + \overbrace{\frac{1}{\sigma_2}\vec{u}_2^{\,T}\vec{w}}^{\text{noise}}\right)\vec{v}_2 + \dots + \alpha_n\left(\underbrace{\vec{v}_n^{\,T}\vec{f}^{\,o}}_{\text{image}} + \underbrace{\frac{1}{\sigma_n}\vec{u}_n^{\,T}\vec{w}}_{\text{noise}}\right)$$

(simple estimate)

want to keep terms where image is much larger than the noise

$$\underline{\text{if}}\ \left|\vec{v}_i^{\,T}\vec{f}^{\,o}\right| > \text{thresh}\left|\frac{1}{\sigma_i}\vec{u}_i^{\,T}\vec{w}\right| \Rightarrow \alpha_i = 1$$

>1

$\underline{\underline{\text{Else}}}$ $\alpha_i = 0$

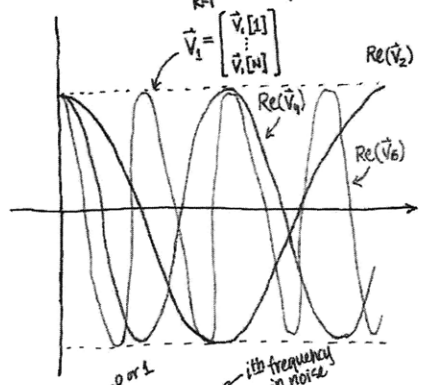
CONNECTION TO FREQUENCY RESPONSE

$$H_{\text{CIRCULENT}} = \begin{bmatrix} & & \\ & \text{DFT}^{-1} & \\ & & \end{bmatrix} \begin{bmatrix} \times & & & & & \\ & \times & & & & \\ & & \times & & & \\ & & & \times & & \\ & & & & \times & \\ & & & & & \times \end{bmatrix} \begin{bmatrix} & & \\ & \text{DFT} & \\ & & \end{bmatrix}$$

$\vec{u}$; diagonal; $\vec{v}^{\,T}$; DFT: discrete fourier transform; complex numbers

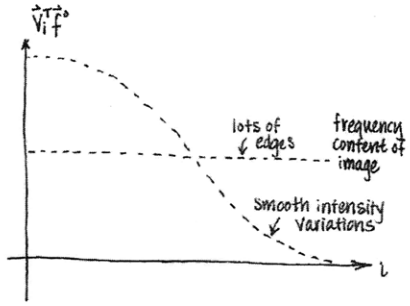
$$\frac{1}{N}\begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ \vdots & & & & & & & \\ \vdots & e^{j2\pi\frac{(k-1)(n-1)}{N}} & & & & & & \\ \vdots & & & & & & & \\ \vdots & & & & & & & \end{bmatrix} \begin{bmatrix} X[1] \\ \vdots \\ X[N] \end{bmatrix} = \begin{bmatrix} x[1] \\ \vdots \\ x[N] \end{bmatrix}$$

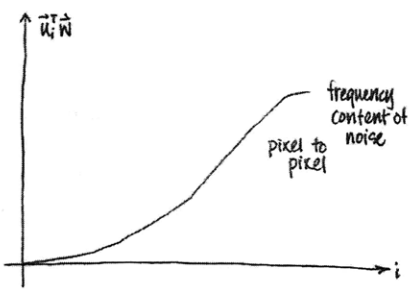
$$x[n] = \frac{1}{N}\sum_{k=1}^{N} e^{j2\pi\frac{(k-1)(n-1)}{N}} X[k]$$

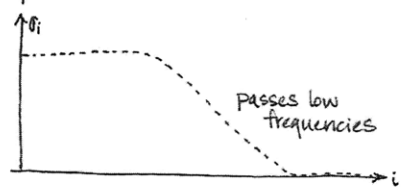


$$\vec{f}^{\,E} = \sum \alpha_i\left(\vec{v}_i\vec{f}^{\,o} + \frac{1}{\sigma_i}\vec{u}_i\vec{w}\right)\vec{v}_i$$

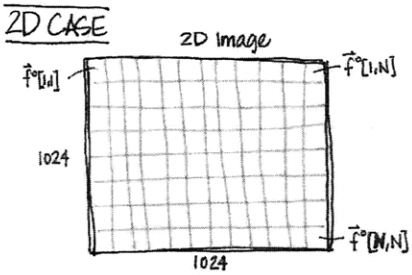
0 or 1; ith frequency in noise; ith frequency in object

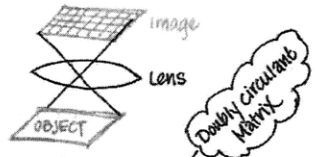






2D CASE





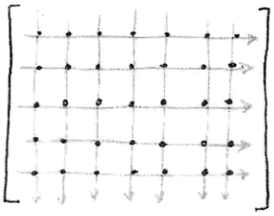
$$\begin{bmatrix} \vec{g}^o[1,1] \\ \vdots \\ \vec{g}^o[1,N] \\ \vec{g}^o[2,1] \\ \vec{g}^o[2,2] \\ \vdots \\ \vec{g}^o[2,N] \\ \vec{g}^o[3,1] \\ \vdots \\ \vec{g}^o[N,N] \end{bmatrix} = \begin{bmatrix} [H[0]] & [H[N-1]] & & \\ [H[1]] & [H[0]] & [H[N-1]] & \\ & & & \\ [H[N-1]] & & [H[\ ]] & [H[0]] \end{bmatrix} \begin{bmatrix} \vec{f}^o[1,1] \\ \vdots \\ \vec{f}^o[N,N] \end{bmatrix}$$

1 million

1 million × 1 million

2D DFT (done fast with FFT)

$$X[n_1,n_2] = \frac{1}{N^2}\sum e^{-j2\pi\frac{(n_1-1)(k_1-1)}{N}} e^{-j2\pi\frac{(n_2-1)(k_2-1)}{N}} X[k_1,k_2]$$



Which terms to keep

$$\vec{f}^E = \sum \alpha_i \vec{V}_i \left(\vec{V}_i^T\vec{f}^o + \frac{1}{\sigma_i}\vec{U}_i^T\vec{W}\right)$$

is α_i 1 or 0?

$\vec{V}_i^T\vec{f}^o$ versus $\frac{1}{\sigma_i}\vec{U}_i^T\vec{W}$

Suppose we know the "energy" in the noise: $\vec{W}^T\vec{W} = E_{Noise}$

& suppose we also know the "energy" in the image: $\vec{f}^{oT}\vec{f}^o = E_{Image}$

$\Rightarrow$ energy in the image is uniform in frequency

$$\vec{f}^o = \sum \vec{V}_i(\vec{V}_i^T\vec{f}^o)$$

$$\vec{f}^{oT}\vec{f}^o = \left(\sum \vec{V}_i(\vec{V}_i^T\vec{f}^o)\right)^T\left(\sum \vec{V}_i(\vec{V}_i^T\vec{f}^o)\right)$$

$$\left|\vec{V}_i^T\vec{f}^o\right| = \frac{\sqrt{E_{Image}}}{N}$$

Uniform Noise assumption: $\left|\vec{U}_i^T\vec{W}\right| = \frac{\sqrt{E_{Noise}}}{N}$

2.21 Blind Deconvolution

6.581/BE.482
LECTURE 22: BLIND DECONVOLUTION

THURSDAY
4 MAY 2006

$g(\vec{x}) = \int k(\vec{x}-\vec{x}')\cdot f^\circ(\vec{x}')d\vec{x}' + w(\vec{x})$

$(f^\circ, k) \rightarrow$ provides infinite number of possible solutions

What types of constraints apply?

- non-negativity
- symmetry on k
- frequency space expected for object
- support for PSF $\rightarrow$ zero beyond some distance
- maximum intensity for object

SAMPLE PROBLEM:

$f^\circ(\vec{x}) * k(\vec{x}) \longrightarrow g^\circ(\vec{x})$

input:

output:

Yang et al. J. Opt. Soc. Am A 11:2401 (1994)

Noiseless: $g^\circ(\vec{x}) = \int k(\vec{x}-\vec{x}')f^\circ(\vec{x}')d\vec{x}'$

$C_g \equiv \{(u,v) : \underset{\substack{\uparrow \\ \text{convolution}}}{u * v} = g^\circ\} \rightarrow (f^\circ, k) \in C_g$

$C_f \equiv \{(u,v) : u \text{ satisfies constraints on } f\}$

$C_k \equiv \{(u,v) : v \text{ satisfies constraints on } k\}$

Let $C_0 = C_g \cap C_f \cap C_k$

To find a solution in C_0:

Iterative projections:

Let P_g be an operator that projects into C_g

& P_f operator projects into C_f

& P_k operator projects into C_k

$(f,k)_{i+1} = P_k P_f P_g (f,k)_i$

initial guess $(f,k)_0$

$\rightarrow$ iterations converge to point in C_0

Projection operator: often a form of minimization

minimize $\|(u,v) - (f',k')\|$ subject to constraints

solution $\uparrow$ $\quad$ $\uparrow$ operand $\quad$ applied with Lagrange multipliers $\uparrow$ $\quad$ $(u,v) \in C_g$ or C_f or C_k

Let $J_{(u,v)} \equiv \|g^\circ - u * v\|^2 = \int [g^\circ(\vec{x}') - (u*v)(\vec{x}')]^2 d\vec{x}' \geq 0$

measure of error to measured image

$(u,v) \in C_g \Rightarrow J_{(u,v)} = 0$

C_f & C_k

1) initial guess: (f_0, k_0)

$f_0 = P_f g^\circ, [k_0 = 0]$

2) Solve for $k_i = \arg\left\{\min_{k \in C_k} J_{(f_{i-1},k)}\right\}$

3) Solve for $f_i = \arg\left\{\min_{f \in C_f} J_{(f,k_i)}\right\}$

4) increment $i \rightarrow i+1$

Poorer Alternative

$E = J_{(f,k)} + E_{restraint}$ $\quad k(\vec{x}) = 0$ for Ω_k

Ω_k  Ω_f 

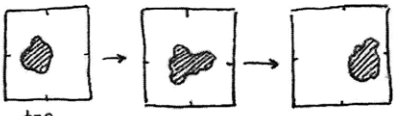
$E_{restraint} = \int_{\Omega_f} |f(\vec{x})|^2 d\vec{x} + \int_{\Omega_k} |k(\vec{x})|^2 d\vec{x}$

$g(\vec{x})$ is invariant to $(f,k) \rightarrow (\alpha f, \frac{k}{\alpha})$

2.22 Optical Flow

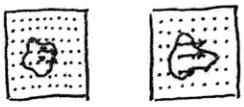
6581 / BE.482
LECTURE 23: OPTICAL FLOW

TUES
9 MAY 0[illegible]



DESCRIPTION:
- movement of C of M
- deformation described as movement expansion

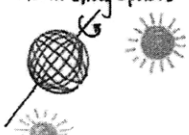
ALTERNATIVE DESCRIPTION:
- vector field flow



OPTICAL FLOW: measuring "motion" in visual field through changes in image across time
- usually, but not always, equivalent to motion of objects

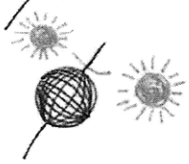
2 PATHOLOGICAL CASES (exceptions, rather than rule)

1. uniform Gray Sphere

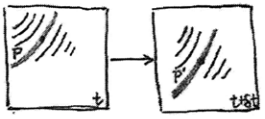


- optical flow zero
- but object is moving

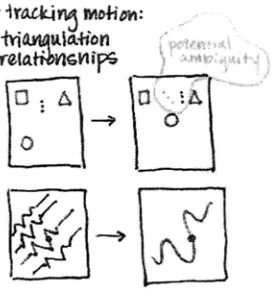
2.



- moving light ⇒ optical flow
- but object is still



2 ways of tracking motion:
1. Global triangulation
2. Local relationships



Imagine a series of 2D images with t as 3rd dimension

Intensity of each pixel: $E(x,y,t)$

Our flow field: $\begin{pmatrix} u(x,y) \\ v(x,y) \end{pmatrix}$ at time t

Most naive assumption

At $t+\delta t$, the location where some index pixel moves to has the same intensity as its index location at time t

$$E(x,y,t) = E(x+\delta x, y+\delta y, t+\delta t) \qquad \delta x = u\delta t, \quad \delta y = v\delta t$$
$$= E(x+u\delta t, y+v\delta t, t+\delta t)$$

Note:
1. At best, only true for small δt
2. Later will deal with issue that intensity may be different in new location
3. Need at least one more constraint to solve for (u,v)

Re-express 1st constraint:

$$E(x,y,t) = E(x,y,t) + \frac{\partial E}{\partial x}\delta x + \frac{\partial E}{\partial y}\delta y + \frac{\partial E}{\partial t}\delta t + \dots$$

$$\frac{\partial E}{\partial x}\frac{dx}{dt} + \frac{\partial E}{\partial y}\frac{dy}{dt} + \frac{\partial E}{\partial t} = 0$$

$$\Rightarrow E_x u + E_y v + E_t = 0$$

Express a "smoothness" constraint

Error function for smoothness

$$e_s = \iint \left[(u_x^2 + u_y^2) + (v_x^2 + v_y^2) \right] dx\,dy \qquad u_x = \frac{\partial u}{\partial x} \quad \vdots$$

$$e_c = \iint (E_x u + E_y v + E_t)^2 dx\,dy$$

we will minimize: $e_s + \lambda e_c$

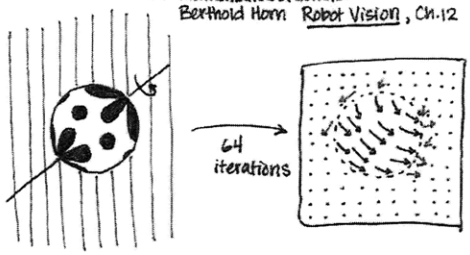
Minimize $\iint F\,dx\,dy$

where $F = \left[(u_x^2 + u_y^2) + (v_y^2 + v_x^2) \right] + \lambda (E_x u + E_y v + E_t)^2$

if $E_x = E_y = 0$

LAPLACE'S EQN FORM: $\nabla^2 u + \nabla^2 v = 0$

For mathematics & details:
Berthold Horn Robot Vision, Ch. 12



2.23 High-Throughput Data and Analysis

6.581 / BE.482

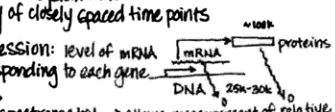
LECTURE 24: HIGH-THROUGHPUT DATA & ITS ANALYSIS

THURS 11 MAY 2006

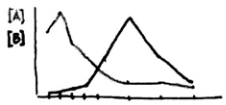
Data Mining

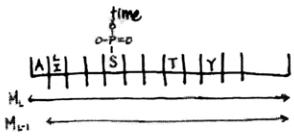
- multichannel – massively parallel
  - System wide
- possibility of closely spaced time points

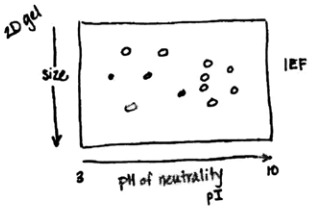
abundance ~concentration:
- gene expression: level of mRNA corresponding to each gene
- proteomics
  - mass spectrometry → allows measurement of relative amount of individual proteins
  - 2D gel electrophoresis



mass spec.

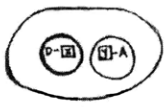


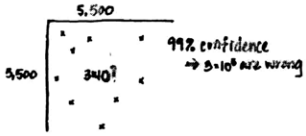




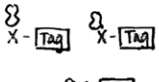
activity – enzyme activity assay

network structure:
- yeast 2-hybrid experiment



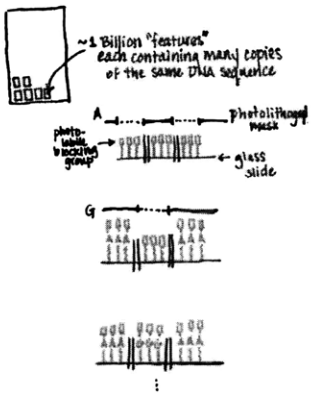


- IP mass spec.



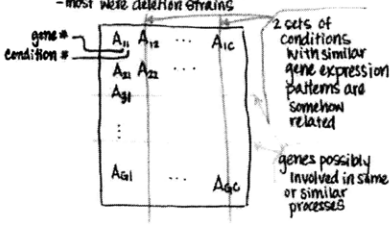
GFP (green fluorescent protein) tagged proteins
← localization

Affymetrix



R Hughes ... SH Friend Cell 102, 109-126 (2000)

300 sets of "conditions"
- some were chemically treated
- most were deletion strains

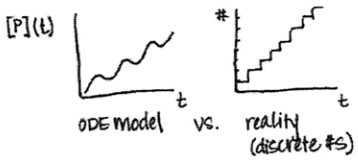


2.24 Inference and Statistics

6.581 / BE.482

LECTURE 25: INFERENCE & STATISTICS

TUESDAY
16 MAY 2006



Sachs, Gifford, Jaakkola, Sorger, & Lauffenburger
http://www.stke.org/cgi/content/full/sigtrans;2002/148/p...

Stimulates (cue) — fibronectin (fn)

(F) Focal Adhesion Kinase

(E) Extracellular signal-related kinase

Experiment was carried out twice

M0: [Cue] (E) (F)

M1: [Cue] → (F) → (E)

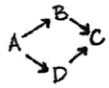
M2: [Cue] → (E) → (F)

M3: [Cue] → (E), (F)

M4: [Cue] → (E), (F)

	Score: Data Set I	Data Set II	(fold-change)
M0:	−50.5 (256)	−60.8	(1700)
M1:	−42.8* (1)	−58.0	(104)
M2:	−43.8 (3)	−57.6	(73)
M3:	−44.3 (4)	−55.6	(10)
M4:	−44.3 (4)	−53.4*	(1)

BAYESIAN NETWORK ANALYSIS



– can only be applied to a directed acyclic graph

Example.

P(C)	T	F
	0.5	0.5

P(S|C)	T	F
T	0.1	0.9
F	0.5	0.5

P(R|C)	T	F
T	0.8	0.2
F	0.2	0.8

P(W|S,R)		T	F
T	T	0.99	0.01
F	T	0.9	0.1
T	F	0.9	0.1
F	F	0.0	1.0

W=T

The chain for probability

$$P(C,S,R,W) = P(C)\cdot P(S|C)\cdot P(R|S,C)\cdot P(W|R,S,C)$$

current model topology

$$= P(C)\cdot P(S|C)\cdot P(R|C)\cdot P(W|S,R)$$

BAYES THEOREM: $P(A,B) \equiv P(A\cap B)$

$$P(A\cap B) = P(A)P(B|A) = P(B)P(A|B)$$

$$P(S=T|W=T) = \frac{P(S=T,W=T)}{P(W=T)} = \frac{\sum_{C,R} P(C,S=T,R,W=T)}{P(W=T)} = 0.4298$$

$$P(R=T|W=T) = \frac{P(R=T,W=T)}{P(W=T)} = \frac{\sum_{C,S} P(C,S,R=T,W=T)}{P(W=T)} = 0.7079$$

$$P(W=T) = 0.6471$$

Experiments give P(Data)

Models can produce P(Data|Model)

What we want to compare P(Model|Data)

$$P(\text{Model}|\text{Data}) = \frac{P(\text{Data}|\text{Model})\,P(\text{Model})}{P(\text{Data})}$$

$$\text{Score}_i = \log P(\text{Model}_i|\text{Data})$$

$$= \log P(\text{Data}|\text{Model}_i) + \log P(\text{Model}_i) - \log P(\text{Data})$$

prior knowledge about system

constant across all models

Simulation of Reaction Kinetics:

Continuous vs Stochastic
(ODE)

$$\frac{d\vec{x}}{dt} = f(\vec{x},\vec{u})$$

Alternative framework

$$\vec{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_M \end{bmatrix}$$

← # of molecules of species 1

Think about dynamics as:

1) System remains unchanged for time τ

2) System will change by a single chemical reaction, μ

Let $P(\tau,\mu)\,\delta\tau$ = probability that $\vec{y}$ at time t will evolve such that no reaction occurs in $(t, t+\tau)$, but that the next reaction occurs in $(t+\tau, t+\tau+\delta\tau)$ and, it is Reaction μ ≡ reaction probability density function

discrete list of reactions

$$= \underbrace{P_0(\tau)}_{\text{probability that waiting time is } \tau}\cdot \underbrace{a_\mu\, d\tau}_{\text{probability that } \mu \text{ is the reaction}}$$

It is shown that

$$P(\tau,\mu) = a_\mu e^{-a_0\tau} \quad \text{for } 0 \le \tau < \infty \ \& \ \mu\ \{1,\ldots,M\}$$

where

$$a_0 = \sum_{i=1}^{M} a_i$$

individual reaction propensities

$$a_\mu = h_\mu c_\mu$$

reaction rate

# of collision types

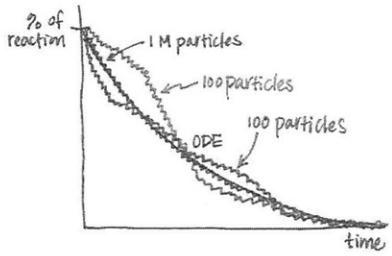
DT Gillespie J. Phys Chem 81. 2340-61 (1977)

J. Phys Chem 115. 1716-33 (2001)

Pick 2 random numbers R_1 & R_2 uniformly on [0,1]

$$\tau = \frac{1}{a_0}\ln\frac{1}{R_1}$$

Select μ such that $\sum_{i=1}^{\mu-1} a_i < R_2 a_0 < \sum_{i=1}^{\mu} a_i$


% of reaction
1 M particles
100 particles
100 particles
ODE
time

Chapter 3

Introduction to Numerical Simulation

Introduction to Numerical Simulation teaches an immense amount of material, it's a general overview for graduate level numerical analysis and covers everything from how to formulate equations based on a model to procedures one should use when solving linear equations. This chapter offers in depth analysis as to why these procedures work and allows the reader to look at problems from several different perspectives. The material is presented in an easy to follow manner with several examples to elucidate each point.

3.1 Example Problems and Basic Equations

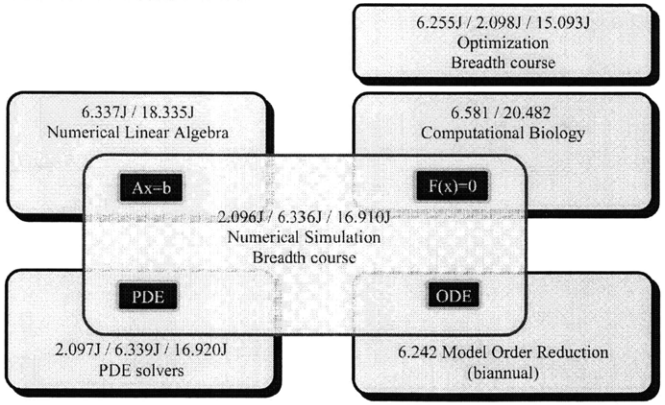
INTRODUCTION TO NUMERICAL SIMULATION

LECTURE 1. Example Problems and Basic Equations

COURSE OUTLINE:

- Assembling system of equations automatically [3 lec]
- Steady state solution
  - Linear Problems: Ax=b [7 lec]
  - Non-Linear Problems: F(x)=0 [3 lec]
- Dynamics (ODE solvers)
  - Time domain integration [3 lec]
  - Periodic steady state [2 lec]
- PDE solvers
  - Integral Equation Methods [2 lec]
  - Finite Element Methods (FEM) [1 lec]
  - Finite Difference Methods (FD) [2 lec]
  - Preconditioners for PDE solvers [1 lec]
- Model Order Reduction [2 lec]

RELATION TO OTHER COURSES

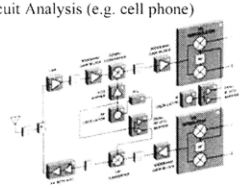


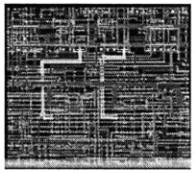
TODAY'S OUTLINE:

- Uses For Simulation
  - Engineering Design
  - Virtual Environments
  - Model Verification
- Course Philosophy
- Example Problems
  - Power distribution on an Integrated Circuit [Electrical]
  - Load bearing on a space frame [Structural]
  - Temperature distribution in an engine [Thermal]

USES FOR SIMULATION

- Circuit Analysis (e.g. cell phone)



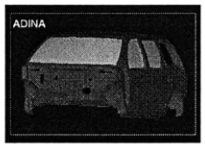


From www.maxim.com

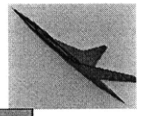
  - Equations
    - Current-voltage relations for circuit elements (resistors, capacitors, transistors, inductors), current balance equations
  - Recent Developments
    - Matrix-Implicit Krylov Subspace methods.
- Electromagnetic Analysis of Packages
  - Equations
    - Maxwell's Partial Differential Equations
  - Recent Developments
    - Fast Solvers for Integral Formulations

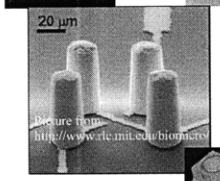
Thanks to Microcosm Inc. now Coventor

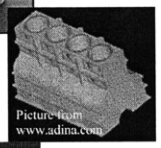
- Structural Analysis of Automobiles
  - Equations
    - Force-displacement relationships for mechanical elements (plates, beams, shells) and sum of forces = 0.
    - Partial Differential Equations of Continuum Mechanics
  - Recent Developments
    - Meshless Methods, Iterative Methods, Automatic Error Control

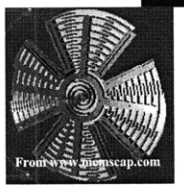


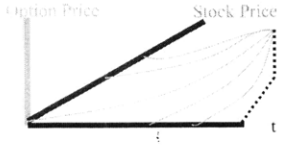
- Drag Force Analysis of Aircraft
  - Equations
    - Navier-Stokes Partial Differential Equations
  - Recent Developments
    - Multigrid Methods for Unstructured Grids
- Analysis of Cell Traps for Sorting Cytometry
  - Equations
    - Navier-Stokes Partial Differential Equations
  - Recent Developments
    - Multigrid Methods for Unstructured Grids
- Engine Thermal Analysis
  - Equations
    - Poisson Partial Differential Equations
  - Recent Developments
    - Fast Integral Equation Solvers, Monte-Carlo Methods
- Micromachined Device Performance Analysis
  - Equations
    - Elastomechanics, Electrostatics, Stokes Flow
  - Recent Developments
    - Fast Integral Equation Solvers, Matrix-Implicit Multi-level Newton Methods for coupled domain problems
- Stock Option Pricing for Hedge Funds
  - Equations
    - Black-Scholes Partial Differential Equation
  - Recent Developments
    - Financial Service Companies are hiring engineers, mathematicians, and physicists



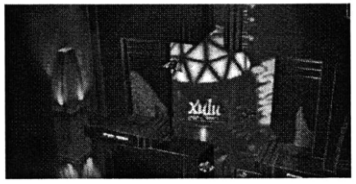


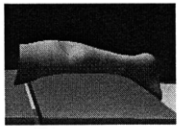


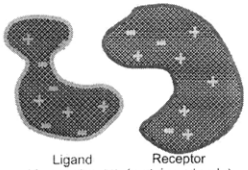


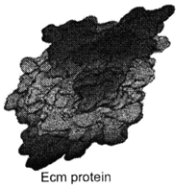


- Virtual Environments for Computer Games
  - Equations
    - Multibody Dynamics, Elastic Collision Equations
  - Recent Developments
    - Multirate integration methods, parallel simulation
- Virtual Surgery
  - Equations
    - Partial Differential Equations of Elastomechanics
  - Recent Developments
    - Parallel Computing, Fast Methods
- Biomolecule Electrostatic Optimization
  - Equations
    - The Poisson Partial Differential Equation
  - Recent Developments
    - Matrix-Implicit Iterative Methods, Fast Integral Equation Solvers

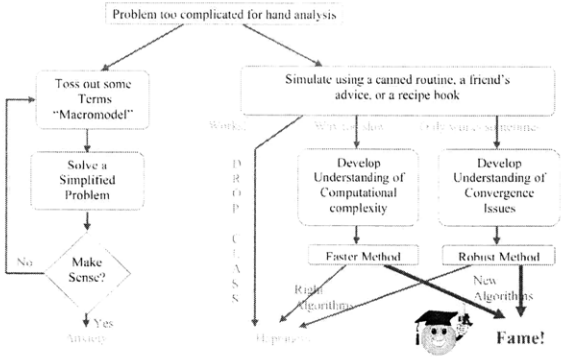








THE COMPUTER SIMULATION SCENARIO



COURSE PHILOSOPHY

Examine Several Modern Techniques

Understand, practically and theoretically, how the techniques perform on representative, but real, applications

Why Prove Theorems?

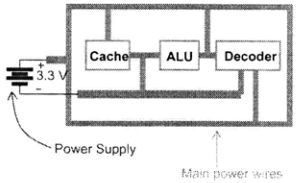
Guarantees, given assumptions, that the method will always work.

Can help debug programs.

The theorem proof can tell you what to do in practice.

EXAMPLE PROBLEMS

- **Power Distribution on an Integrated Circuit**



Is there at least 3V across the ALU?

- Design Objectives for the VLSI Problem

  Select topology and metal widths and lengths so that

  a) Voltage across every function block > 3V
  b) Minimize the area used for the metal wires

- First Step – Analysis Tools

  Given the topology and metal widths and lengths determine the voltage across the ALU, Cache and Decoder.

- Who uses VLSI Tools?
  - Several big companies

    IBM, Motorola, TI, Intel, Compaq, Sony, Hitachi

    Nonfunctional prototype costs:
    - Increases time to market
    - Design rework costs millions
  - 1000's of small companies
    - Small companies make application circuits disk drives, graphics accelerators, CD players, cell phones
    - What is the cost of nonfunctional prototypes?
      - Out of business

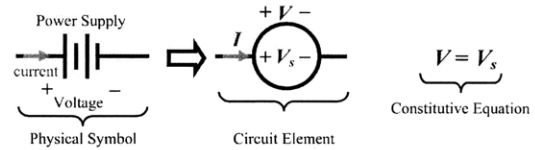
- Who makes VLSI Tools?

Company	Employees	Sales	Market capital
Cadence	4,000	1.3 billion	3.8 billion
Synopsis/Avanti	5,000	1.5 billion	6.9 billion
Mentor Graphics	2,600	0.6 billion	1.4 billion

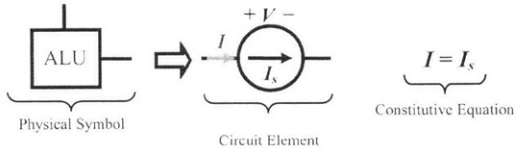
(Data from 2003)

  Companies compete by improving analysis efficiency.

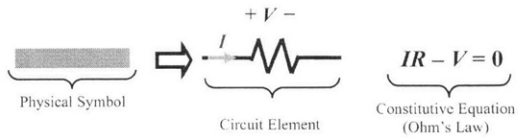
- Modeling VLSI Circuit Power Distribution
  - Power Supply provide current at a certain voltage
  - Functional blocks draw current
  - The wire resistance generates losses
  - Modeling the Circuit
    - Supply becomes a Voltage Source



- Functional Blocks become Current Sources

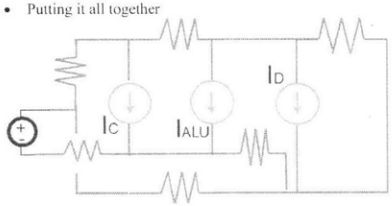


- Metal Lines become Resistors



$$R = \underbrace{\frac{\text{length}}{\text{area}}}_{\text{Design Parameter}} \cdot \underbrace{\text{resistivity}}_{\text{Material Property}}$$

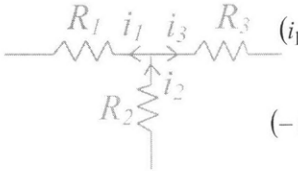
- Putting it all together



  - Power Supply → Voltage Source
  - Functional Blocks → Current Sources
  - Wires → Resistors

  Result is a schematic
- Formulating Equations from Schematics
  - Two Types of Unknowns
    Node voltages, element currents
  - Constitutive Equations

    Element current is related to voltage across the element
  - Two Types of Equations
    Sum of currents at each node = 0
  - Conservation/Balance Law Equations



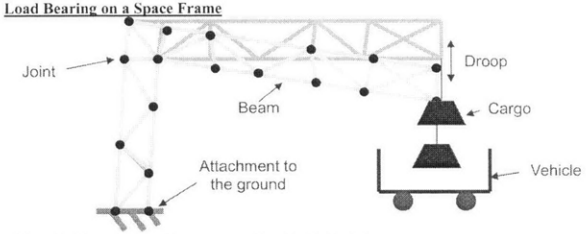
$$(i_1 - i_2 + i_3 = 0)$$

OR

$$(-i_1 + i_2 - i_3 = 0)$$

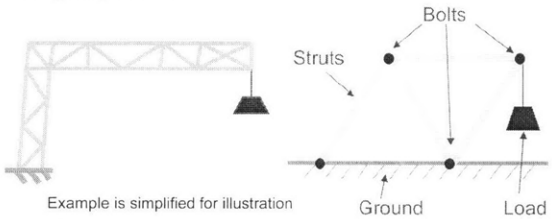
Note: here $i_1 = -3$ means it goes to the right i.e. opposite to the arrow

- **Load Bearing on a Space Frame**



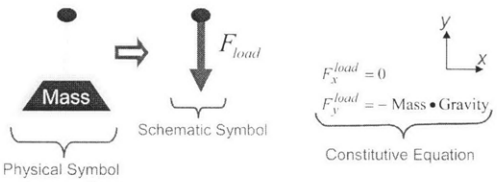
  Does the Space Frame droop too much under the load?
  - Design Objectives for the Space Frame
    Select topology and strut widths and lengths so that
    a) Droop is small enough
    b) Minimize the metal used
  - First Step – Analysis Tools
    Given the topology and metal widths and lengths determine the droop of the space frame under load

- Modeling the Space Frame

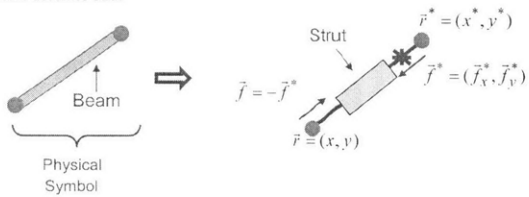


Example is simplified for illustration

- Load becomes Force Source



- Beam becomes Strut

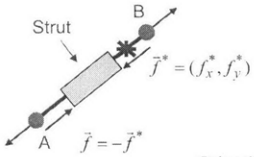


L_0 = Unstretched Length
A_c = Cross-Sectional Area
} *Design Parameters*

E = Young's Modulus } *Material Property*

$$f^* = EA_c \frac{\Delta L}{L_0}$$

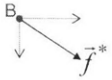
Abstraction – leads to simplifications, such as:
- No bending/buckling
- No twisting
- No breaking



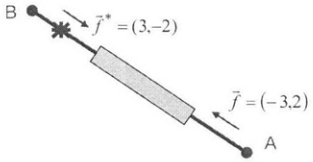
Only solve for the force, $\vec{f}^*$, on one side

The force on the other side is $\vec{f} = -\vec{f}^*$

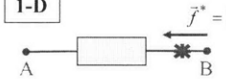
If $\vec{f}^* = (3,-2)$ it means



thus the strut is turned around



1-D

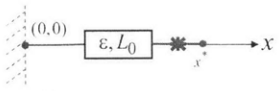


$\vec{f}^*$ is the restoring force of the strut (internal force)

Is this strut compressed or stretched?

The force of the strut is pulling the end in this means that the strut is being stretched.

Think of what happens when you pull on the ends of a spring. Is the force from the spring pushing your hands away from one another or pulling them towards one another?



$\hat{e}_1$ (*unit vector*) $\quad \vec{f}^* \propto \Delta L = L - L_0$

$$\vec{f}^* = -\,\hat{e}_1\,\varepsilon\,(L - L_0) = \underbrace{\hat{e}_1\,\varepsilon\left(L_0 - |x^* - 0|\right)}_{\text{Constitutive Equation (Hooke's Law)}}$$

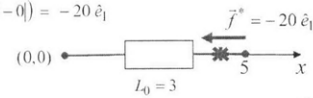
If $\Delta L > 0$, the strut is stretched, the internal force will pull to the left, and thus be negative

$$\varepsilon = \frac{EA_c}{L_0}$$

Example.

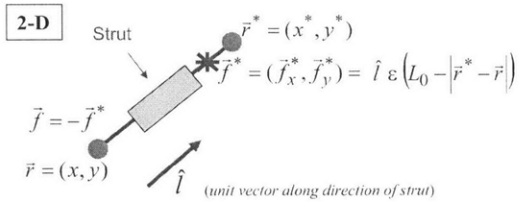
$\varepsilon = 10, \quad L_0 = 3, \quad x^* = 5$

$\vec{f}^* = \hat{e}_1\,10\,(3 - |5 - 0|) = -20\,\hat{e}_1$



In this case $\begin{cases} f_y = 0 \\ f_x = f^* \end{cases}$

2-D

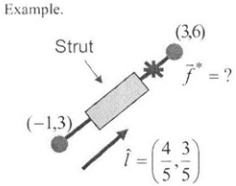


(*unit vector along direction of strut*)

$$\hat{l} = \frac{\vec{r}^* - \vec{r}}{|\vec{r}^* - \vec{r}|} = \left(\frac{x^* - x}{L}, \frac{y^* - y}{L}\right)$$

$$L = \sqrt{(x^* - x)^2 + (y^* - y)^2}$$

Example.



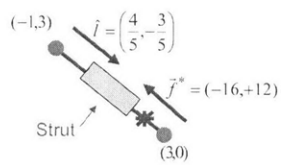
$\varepsilon = 10, L_0 = 3$

$$L = |\vec{r}^* - \vec{r}| = \sqrt{[3-(-1)]^2 + (6-3)^2} = 5 \qquad \text{stretched strut}$$

$$f_x^* = \frac{x^* - x}{L}\,\varepsilon\,(L_0 - L) = \frac{3-(-1)}{5}\,10\,(3-5) = -16$$

$$f_y^* = \frac{y^* - y}{L}\,\varepsilon\,(L_0 - L) = \frac{6-3}{5}\,10\,(3-5) = -12$$

Example. $\varepsilon = 10,\ L_0 = 3$



$$L = \vec{r}^* - \vec{r}\,\Big| = \sqrt{(3-(-1))^2 + (0-3)^2} = 5 \quad \text{stretched strut}$$

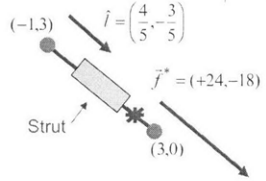
$$f_x^* = \frac{x^* - x}{L}\ \varepsilon\ (L_0 - L) = \frac{3-(-1)}{5}\ 10\ (3-5) = -16$$

$$f_y^* = \frac{y^* - y}{L}\ \varepsilon\ (L_0 - L) = \frac{0-3}{5}\ 10\ (3-5) = +12$$

Example. $\varepsilon = 10,\ L_0 = 8$

$$L = \left|\vec{r}^* - \vec{r}\right| = \sqrt{(3-(-1))^2 + (0-3)^2} = 5$$

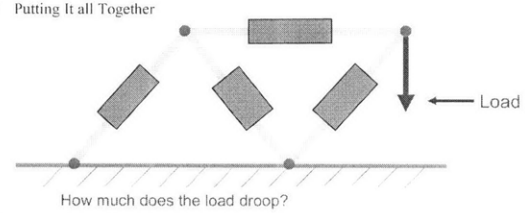
compressed strut



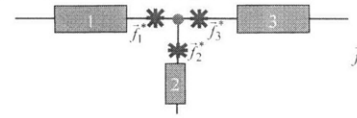
$$f_x^* = \frac{x^* - x}{L}\ \varepsilon\ (L_0 - L) = \frac{3-(-1)}{5}\ 10\ (8-5) = +24$$

$$f_y^* = \frac{y^* - y}{L}\ \varepsilon\ (L_0 - L) = \frac{0-3}{5}\ 10\ (8-5) = -18$$

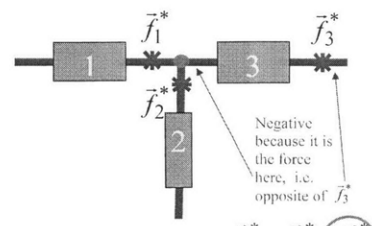
- Putting It all Together



- Formulating Equations from Schematics
  - Two Types of Unknowns
    Joint Positions, Strut Forces
  - Constitutive Equations
    Element Force is related to the change in Element Length
  - Two Types of Equations
    Sum of Forces at each joint = 0
  - Conservation/Balance Law Equations



$$\vec{f}_1^* + \vec{f}_2^* + \vec{f}_3^* = 0$$



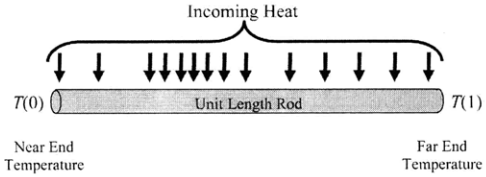
$$\vec{f}_1^* + \vec{f}_2^* - \vec{f}_3^* = 0$$

Note: $\vec{f}_1^* = (-3,0)$ means the force goes to the left.

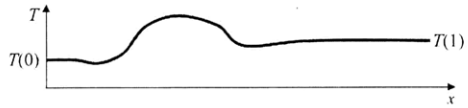
- **Temperature Distribution in an Engine**

  Does the engine get too hot?

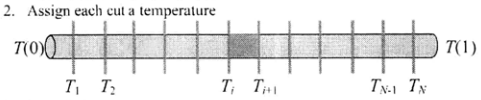
  - Design Objectives for the Engine
    Select the shape so that
    a) The temperature does not get too high
    b) Minimize the metal used
  - Heat Flow 1-D Example
    - Conservation Laws and Constitutive Equations



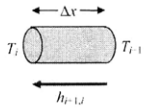
Question: What is the temperature distribution along the bar?



- Discrete Representation
  1. Cut the bar into short sections
  2. Assign each cut a temperature

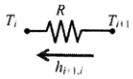


- Constitutive Relation
  Heat Flow through one section



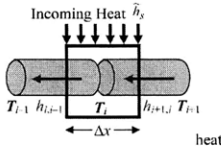
$$h_{i+1,i} = \text{heat flow} = \kappa \frac{T_{i+1} - T_i}{\Delta x}$$

thermal conductivity



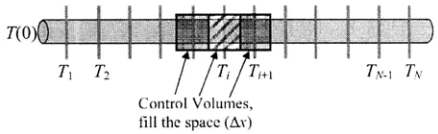
$$\frac{1}{R_{thermal}} = \frac{\kappa}{\Delta x}$$

- Conservation Law
  Net Heat Flow into Control Volume = 0



$$h_{i+1,i} - h_{i,i-1} = -\tilde{h}_s \Delta x$$

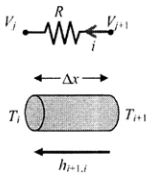
heat in from right — heat out from left — incoming heat per unit length



Limit as the sections become vanishingly small:

$$\lim_{\Delta x \to 0} \frac{h_{i+1,i} - h_{i,i-1}}{\Delta x} = -\tilde{h}_s$$

- Circuit Analogy

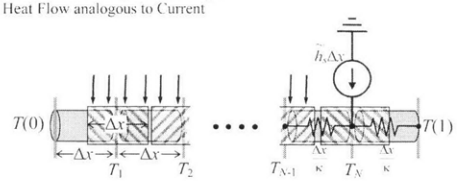


$$i = \frac{V_{j+1} - V_j}{R}$$

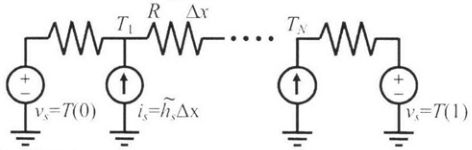
$$h_{i+1,i} = \frac{T_{i+1} - T_i}{\Delta x}$$

κ

Temperature analogous to Voltage
Heat Flow analogous to Current

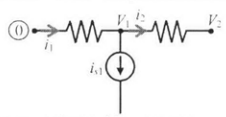


$$\frac{1}{R} = \frac{\kappa}{\Delta x}$$

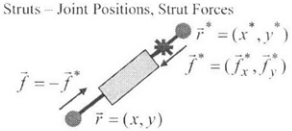


SUMMARY OF KEY POINTS:

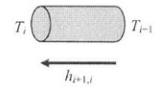
- Many Applications of simulation
  - Picked Three Representative Examples
    Circuits, Struts and Joints, Heat Flow in Bar
- Two Types of Unknowns
  - Circuit – Node Voltages, Element Currents



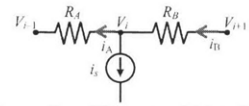
  - Struts – Joint Positions, Strut Forces



  - Conducting Bar – Temperature, Section Heat Flows

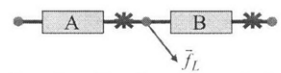


- Two Types of Equations
  - Conservation/Balance Laws
    - Circuit – Sum of Currents at each node = 0



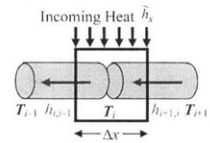
$$i_A - i_B + i_s = 0$$

    - Struts – Sum of Forces at each joint = 0



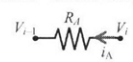
$$\vec{f}_A^* - \vec{f}_B^* + \vec{f}_L = 0$$

    - Bar – Sum of heat flows into control volume = 0



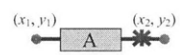
$$h_{i,i-1} - h_{i+1,i} - \tilde{h}_s\Delta x = 0$$

  - Constitutive Equation
    - Circuit – current-voltage relationship



$$R_A i_A = V_i - V_{i-1}$$

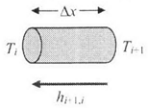
    - Struts – force-displacement relationship



$$f_{A,x}^* = \frac{x_2 - x_1}{L_A}\varepsilon\left(L_{A,0} - L_A\right)$$

$$f_{A,y}^* = \frac{y_2 - y_1}{L_A}\varepsilon\left(L_{A,0} - L_A\right)$$

- Bar – temperature drop-heat flow relationship



$$h_{i+1,i} = \kappa \frac{T_{i+1} - T_i}{\Delta x}$$

3.2 Equation Formulation - Node Branch Stamping

INTRODUCTION TO NUMERICAL SIMULATION

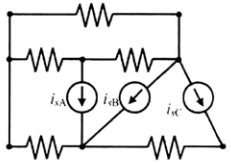
LECTURE 2. Equation Formulation & Node-Branch Stamping

TODAY'S OUTLINE:

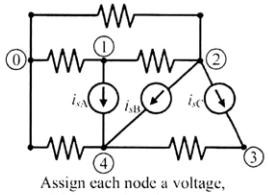
- Formulating Equations
  - Circuit Example
  - Struts and Joints Example
- Matrix Construction From Schematics
  - Node-Branch "Stamping Procedure"
    - Circuits
    - Struts and Joints

FORMULATING EQUATIONS FROM SCHEMATICS

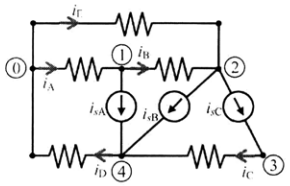
Circuit Example



- Step 1: Identifying Unknowns

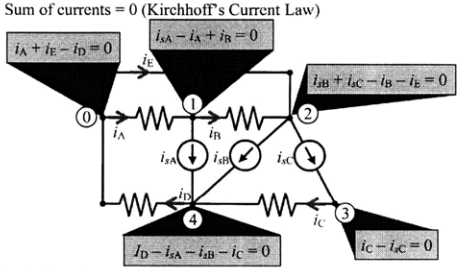


Assign each node a voltage, with one node as 0.

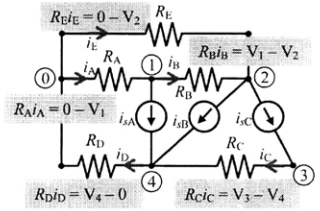


Assign each element a current.

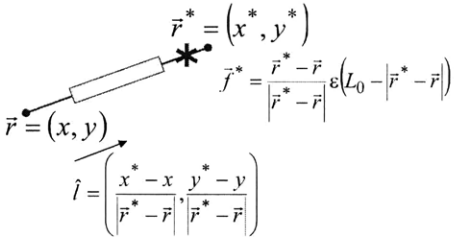
- Step 2: Conservation Laws
  Sum of currents = 0 (Kirchhoff's Current Law)

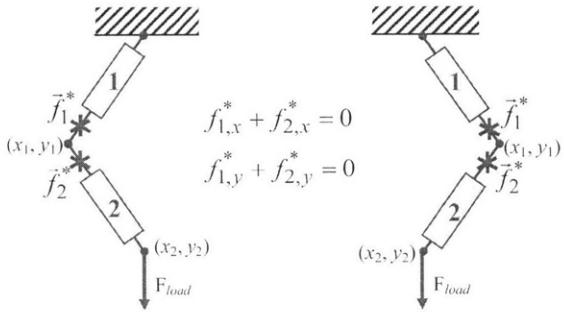


- Step 3: Constitutive Equations
  Use Constitutive Equations to relate branch currents to node voltages
  (Currents flow from plus node to minus node)



Struts Example



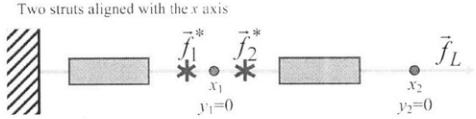


$$f_{1,x}^* + f_{2,x}^* = 0$$
$$f_{1,y}^* + f_{2,y}^* = 0$$

Will the solutions be the same?
Will the set of conservation law equations be different?

Conservation laws for the two examples will be exactly the same. The perceived force "direction" is inconsequential, it is the adjacent forces that matter.

Two struts aligned with the x axis



Conservation Law

At node 1: $f_{1,x}^* + f_{2,x}^* = 0$

At node 2: $-f_{2,x}^* + f_L = 0$

Constitutive Equations

$$f_{1,x}^* = \frac{x_1 - 0}{|x_1 - 0|}\varepsilon(L_0 - |x_1 - 0|)$$

$$f_{2,x}^* = \frac{x_1 - x_2}{|x_1 - x_2|}\varepsilon(L_0 - |x_1 - x_2|)$$

$$\vec{f}^* = \frac{\vec{r}^* - \vec{r}}{|\vec{r}^* - \vec{r}|}\varepsilon\left(L_0 - \underbrace{|\vec{r}^* - \vec{r}|}_{L}\right)$$

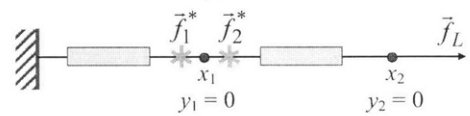
Reduced (Nodal) Equations

$$f_{1x}^* + f_{2x}^* = 0$$

$$\underbrace{\frac{x_1}{|x_1|}\varepsilon(L_0 - |x_1|)}_{f_{1,x}^*} + \underbrace{\frac{x_1 - x_2}{|x_1 - x_2|}\varepsilon(L_0 - |x_1 - x_2|)}_{f_{2,x}^*} = 0$$

$$-f_{2x}^* + f_L = 0$$

$$\underbrace{-\frac{x_1 - x_2}{|x_1 - x_2|}\varepsilon(L_0 - |x_1 - x_2|)}_{-f_{2,x}^*} + f_L = 0$$



Example.) $\vec{f}_L = 10\hat{e}_1$ (force in positive x direction)

Solution of Nodal Equations

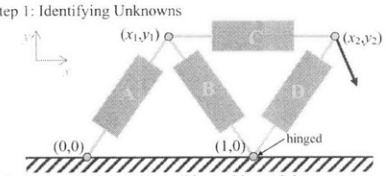
$$x_1 = L_0 + \frac{10}{\varepsilon} \qquad x_2 = x_1 + L_0 + \frac{10}{\varepsilon}$$

Notice the signs of the forces

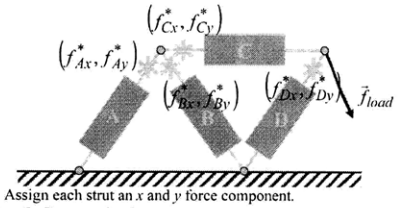
$f_{2,x}^* = 10$ force in positive x direction

$f_{1,x}^* = -10$ force in negative x direction

- Step 1: Identifying Unknowns



Assign each joint an x, y position, with one joint as zero.

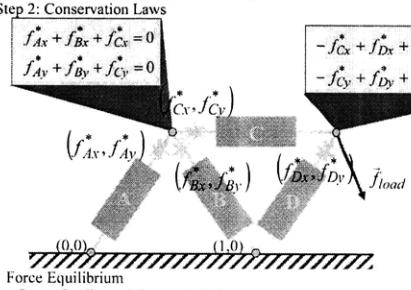


Assign each strut an x and y force component.

- Step 2: Conservation Laws

$$f^*_{Ax} + f^*_{Bx} + f^*_{Cx} = 0$$
$$f^*_{Ay} + f^*_{By} + f^*_{Cy} = 0$$

$$-f^*_{Cx} + f^*_{Dx} + f^*_{load,x} = 0$$
$$-f^*_{Cy} + f^*_{Dy} + f^*_{load,y} = 0$$

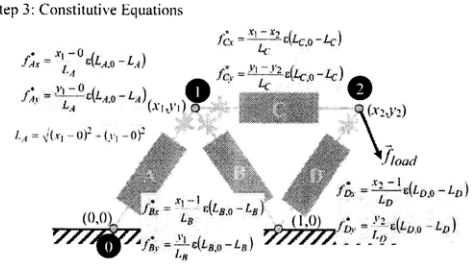


Force Equilibrium

Sum of x-directed forces at a joint = 0

Sum of y-directed forces at a joint = 0

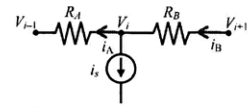
- Step 3: Constitutive Equations



Use Constitutive Equations to relate strut forces to joint positions.

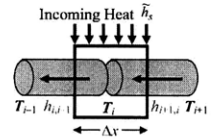
Comparing Conservation Laws

- Circuit



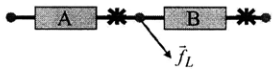
$$i_A - i_B + i_s = 0$$

- Heat Flow



$$h_{i,i-1} - h_{i+1,i} - \tilde{h}_s \Delta x = 0$$

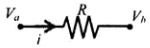
- Struts



$$\vec{f}^*_A - \vec{f}^*_B + \vec{f}_L = 0$$

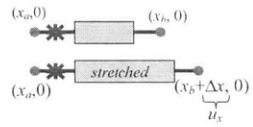
Summary of Key Points

- Two Types of Unknowns
  - Circuit: Node Voltages, Element Currents
  - Struts: Joint Positions, Strut Forces
  - Bar: Node Temperatures, Heat Flows
- Two Types of Equations
  - Conservation/Balance Laws
    - Circuit: Sum of Currents at each node = 0
    - Struts: Sum of Forces at each joint = 0
    - Bar: Sum of heat flows into control volume = 0
  - Constitutive Equation
    - Circuit: current-voltage relationship



$$i = \frac{1}{R}(V_a - V_b)$$

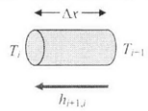
    - Struts: force-displacement relationship



$$f_x^* = \frac{x_a - (x_b + \Delta x)}{|x_a - (x_b + \Delta x)|}\varepsilon\left[|x_a - x_b| - |x_a - (x_b + \Delta x)|\right]$$

f_x^* is a positive number – force is to the right, this makes sense for a stretched strut.

- Bar: temperature drop-heat flow relationship



$$h_{i+1,i} = \kappa\frac{T_{i+1} - T_i}{\Delta x}$$

Generating Matrices from Schematics

$$\overbrace{\begin{bmatrix} & & \\ & A_{i,j} & \\ & & \end{bmatrix}}^{\text{Number of Columns} = \text{Number of Unknowns}} \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} b_1 \\ \vdots \\ b_n \end{bmatrix}$$

Number of Rows = Number of Equations

row column

Circuit Example

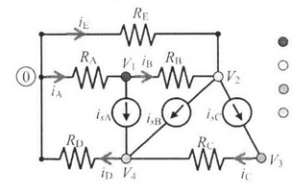
Assume linear constitutive equations…

$$\begin{array}{l} \\ \mathbf{N}\ \text{KCL Eqns} \\ \\ \mathbf{B}\ \text{Constitutive Eqns} \end{array} \begin{array}{c} \begin{array}{cc} \mathbf{B} & \mathbf{N} \\ \text{branch currents} & \text{node voltages} \end{array} \\ \begin{bmatrix} & \\ & \\ & \end{bmatrix} \end{array} \begin{bmatrix} i_b \\ V_N \end{bmatrix} = \begin{bmatrix} \\ \\ \\ \end{bmatrix}$$

- One matrix column for each unknown
  - N columns for the Node voltage
  - B columns for the Branch currents
- One matrix row for each equation
  - N rows for KCL
  - B rows for element constitutive equations

(linear and square system!)

- Conservation Equation



$$\begin{array}{ll} -i_A + i_B & = -i_{sA} \\ -i_B & -i_E = -i_{sB} - i_{sC} \\ i_C & = i_{sC} \\ -i_C + i_D & = i_{sA} + i_{sB} \end{array}$$

Matrix Form for the Equations

The matrix A is usually not square

$$\overbrace{\begin{array}{c} \\ \text{one row for each KCL equation} \\ \\ \end{array}}^{A} \begin{array}{c} \\ N \\ \\ \end{array} \begin{array}{c} \begin{array}{c} ❶ \\ ❷ \\ ❸ \\ ❹ \end{array} \begin{bmatrix} -1 & 1 & & & \\ & -1 & & & -1 \\ & & 1 & & \\ & & -1 & 1 & \end{bmatrix} \\ \underbrace{\qquad\qquad\qquad}_{B} \\ \text{one column for each branch current} \end{array} \begin{bmatrix} i_A \\ i_B \\ i_C \\ i_D \\ i_E \end{bmatrix} = \underbrace{\begin{bmatrix} -i_{sA} \\ -i_{sB} - i_{sC} \\ i_{sC} \\ i_{sA} + i_{sB} \end{bmatrix}}_{\text{right hand side for source currents}}$$

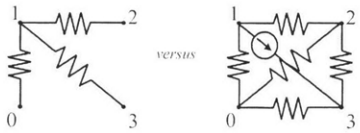
Four Nodes

Do we know how many rows the A matrix will have?

3 because there are 3 nonzero nodes

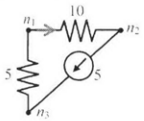
Do we know how many columns the A matrix will have?

Could have any number, depends on the number of elements



Input file of a circuit simulation has one line per resistor:

Rname	node1	node2	value
Spice → circuit simulator code...			
RnameA	n1	n2	10
RnameB	n1	n3	5
Iname	n2	n3	5



$$
\begin{array}{l}
 \\ \text{Node1} \bullet \text{KCL} \bullet \text{V}_1 \\ \text{Node2} \bullet \text{KCL} \bullet \text{V}_2 \\ \text{Node3} \bullet \text{KCL} \bullet \text{V}_3 \\ \vdots
\end{array}
\begin{array}{c}
\begin{array}{ccc} R_A & R_B & I \end{array} \\
\begin{bmatrix}
+1 & +1 & \\
-1 & & +1 \\
 & -1 & \\
 & & \\
 & & \\
\end{bmatrix}
\end{array}
$$

How does each resistor contribute to the matrix?

A has no more than two non-zeros per column

KCL at...

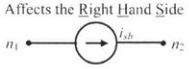
n_1: $\sum i_{other} + i_k = \sum i_s$

n_2: $\sum i_{other} - i_k = \sum i_s$

$$
\begin{array}{c|c}
 & k \\
\hline
n_1 & 1 \\
n_2 & -1
\end{array}
\quad \underbrace{}_{A}
$$

How does each current source contribute to the Conservation Law Equation?

Affects the Right Hand Side



KCL at...

n_1: $\sum i_b\text{'s} = \sum i_{s_{other}} - i_{sb}$

n_2: $\sum i_b\text{'s} = \sum i_{s_{other}} + i_{sb}$

$$
\begin{array}{c|c}
 & \\
\hline
n_1 & \sum i_{s_{other}} - i_{sb} \\
n_2 & \sum i_{s_{other}} + i_{sb}
\end{array}
\quad \underbrace{}_{\text{RHS}}
$$

Conservation Matrix Equation Generation Algorithm

For each resistor

if $(n_1 \neq 0)$ then $A(n_1,k) \leftarrow 1$

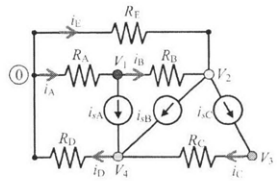
if $(n_2 \neq 0)$ then $A(n_2,k) \leftarrow -1$

Set I_s = zero vector

For each current source

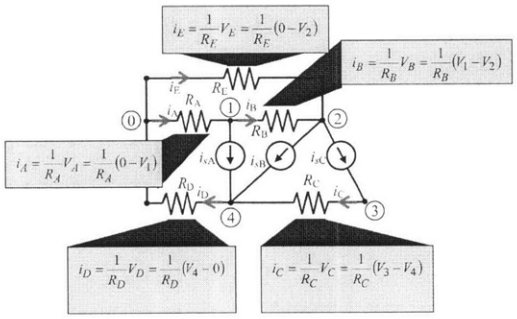
if $(n_1 \neq 0)$ then $I_s(n_1) \leftarrow I_s(n_1) - i_{sb}$

if $(n_2 \neq 0)$ then $I_s(n_2) \leftarrow I_s(n_2) + i_{sb}$

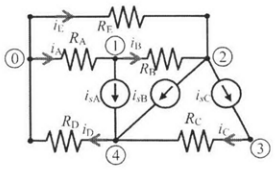


$$\begin{array}{c} ① \\ ② \\ ③ \\ ④ \end{array} \overbrace{\begin{bmatrix} -1 & 1 & & & \\ & -1 & & & -1 \\ & & 1 & & \\ & & -1 & 1 & \end{bmatrix}}^{A} \begin{bmatrix} i_A \\ i_B \\ i_C \\ i_D \\ i_E \end{bmatrix} = \begin{bmatrix} -i_{sA} \\ -i_{sB} - i_{sC} \\ i_{sC} \\ i_{sA} + i_{sB} \end{bmatrix}$$

- Constitutive Equation



First relate branch currents to branch voltages
Second determine voltages across resistors (Branch Voltages)



$$\begin{bmatrix} i_A \\ i_B \\ i_C \\ i_D \\ i_E \end{bmatrix} = \overbrace{\begin{bmatrix} \frac{1}{R_A} & & & & \\ & \frac{1}{R_B} & & & \\ & & \frac{1}{R_C} & & \\ & & & \frac{1}{R_D} & \\ & & & & \frac{1}{R_E} \end{bmatrix}}^{\alpha} \begin{bmatrix} V_A \\ V_B \\ V_C \\ V_D \\ V_E \end{bmatrix}$$

$$i_b = \frac{1}{R} V_b = \frac{1}{R}(V_1 - V_2)$$

$$i_b - \frac{1}{R}\underbrace{(V_1 - V_2)}_{\begin{bmatrix} V_b \end{bmatrix} = A^T \begin{bmatrix} V_N \end{bmatrix}} = 0$$

The kth resistor contributes $\frac{1}{R_k}$ to $\alpha\,(k,k)$

The matrix α relates branch voltages to branch currents.
- One row for each unknown current.
- One column for each associated branch voltage.

The matrix α is square and diagonal.
Relationship between branch voltages and node voltages:

$$\begin{aligned} V_A &= 0 - V_1 \\ V_B &= V_1 - V_2 \\ V_C &= V_3 - V_4 \\ V_D &= V_4 - 0 \\ V_E &= 0 - V_2 \end{aligned}$$

Examine Matrix Construction

$$\begin{bmatrix} V_A \\ V_B \\ V_C \\ V_D \\ V_E \end{bmatrix} = \begin{bmatrix} -1 & & & \\ 1 & -1 & & \\ & & 1 & -1 \\ & & & 1 \\ & -1 & & \end{bmatrix} \begin{bmatrix} V_1 \\ V_2 \\ V_3 \\ V_4 \end{bmatrix}$$

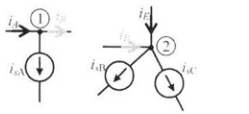
The node-to-Branch matrix is the transpose of the KCL Matrix.

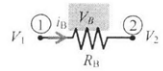
KCL Equations

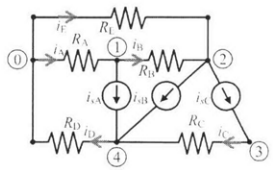
$$\underbrace{\begin{bmatrix} -1 & 1 & & & \\ & -1 & & & -1 \\ & & 1 & & \\ & & -1 & 1 & \end{bmatrix}}_{A} \begin{bmatrix} i_A \\ i_B \\ i_C \\ i_D \\ i_E \end{bmatrix} = \begin{bmatrix} \\ \\ I_s \\ \\ \end{bmatrix}$$

Node to Branch Relation

$$\underbrace{\begin{bmatrix} -1 & & & \\ 1 & -1 & & \\ & & 1 & -1 \\ & & & 1 \\ & -1 & & \end{bmatrix}}_{A^T} \begin{bmatrix} V_1 \\ V_2 \\ V_3 \\ V_4 \end{bmatrix} = \begin{bmatrix} V_A \\ V_B \\ V_C \\ V_D \\ V_E \end{bmatrix}$$







$$\begin{bmatrix} I_b \end{bmatrix} - \alpha A^T \begin{bmatrix} V \end{bmatrix} = 0$$

$$\begin{bmatrix} i_A \\ i_B \\ i_C \\ i_D \\ i_E \end{bmatrix} - \alpha A^T \begin{bmatrix} V_1 \\ V_2 \\ V_3 \\ V_4 \end{bmatrix} = \underbrace{\begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}}_{V_s}$$

The node voltages can be related to branch currents

- A^T relates node voltages to branch voltages.
- α relates branch voltages to branch currents.
  α is square and diagonal

- Node-Branch Form

  $I_b - \alpha A^T V_N = 0$ Constitutive Relation

  $A I_b = I_s$ Conservation Law

$$\begin{array}{c} B\updownarrow \\ N\updownarrow \\ \\ \end{array} \begin{bmatrix} I & -\alpha A^T \\ A & 0 \end{bmatrix} \begin{bmatrix} I_b \\ V_N \end{bmatrix} = \begin{bmatrix} 0 \\ I_s \end{bmatrix}$$

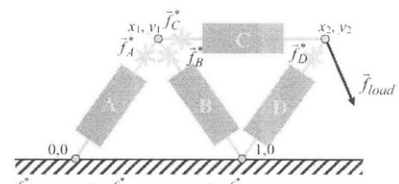
$$\begin{array}{cc} \leftrightarrow & \longleftrightarrow \\ B & N \end{array}$$

N = number of Nodes with unknown voltages

B = number of Branches with unknown currents

Struts Example

- In 2-D
  - One pair of columns for each unknown
    - J pairs of columns for the joint positions
    - S pairs of columns for the strut forces
  - One pair of matrix rows for each equation
    - J pairs of rows for the force equilibrium equations
    - S pairs of rows for the ***linearized*** constitutive relations
- Follow Approach Parallel to Circuits
  (1) Form an "Incidence Matrix," A, from Conservation Law.
  (2) Determine strut deformation using A^T.
  (3) Use linearized constitutive equations to relate strut deformation.
  (4) Combine (1), (2), and (3) to generate a node-branch form.
- Conservation Laws

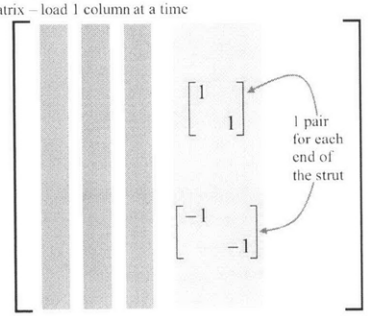


$$\begin{aligned} f^*_{A,x} + f^*_{B,x} + f^*_{C,x} &= 0 \\ f^*_{A,y} + f^*_{B,y} + f^*_{C,y} &= 0 \\ -f^*_{C,x} + f^*_{D,x} &= -f_{load,x} \\ -f^*_{C,y} + f^*_{D,y} &= -f_{load,y} \end{aligned}$$

Note that struts A & B only contribute one pair of entries into the A matrix and strut C contributes two pairs of entries into the A matrix. This is because struts A & B are connected to the wall (ground) and strut C has two free ends.

- Stamping Approach

  Matrix – load 1 column at a time



  Load pair of columns per strut

  Load right side for load

$$
\underbrace{\begin{array}{c} x_1 \\ y_1 \\ x_2 \\ y_2 \end{array}\begin{bmatrix} 1 & & 1 & & & \\ & 1 & & 1 & & \\ & & -1 & & 1 & \\ & & & -1 & & 1 \end{bmatrix}}_{A}
\begin{bmatrix} f^*_{A,x} \\ f^*_{A,y} \\ f^*_{B,x} \\ f^*_{B,y} \\ f^*_{C,x} \\ f^*_{C,y} \\ f^*_{D,x} \\ f^*_{D,y} \end{bmatrix}
=
\begin{bmatrix} 0 \\ 0 \\ -f_{load,x} \\ -f_{load,y} \end{bmatrix}
$$

- Conservation Matrix Generation Algorithm

  For each strut

  If (j_1 is not fixed) $\quad A(j_{1x},b_x) = 1 \quad A(j_{1y},b_y) = 1$

  If (j_2 is not fixed) $\quad A(j_{2x},b_x) = -1 \quad A(j_{2y},b_y) = -1$

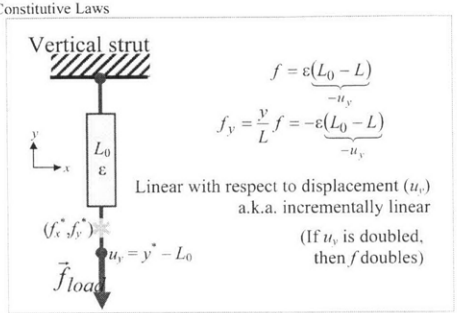
  For each load

  If (j_1 is not fixed) $\quad F_L(j_{1x}) = F_L(j_{1x}) - f_{load,x}$

  $F_L(j_{1y}) = F_L(j_{1y}) - f_{load,y}$

  A has at most 2 non-zeros per column

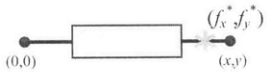
- Constitutive Laws



Linearize the Constitutive Equations.

$$f_x = \frac{x}{L} EA_c \frac{L_0 - L}{L_0} = \frac{x}{L}\varepsilon(L_0 - L)$$

$$f_y = \frac{y}{L} EA_c \frac{L_o - L}{L_0} = \frac{y}{L}\varepsilon(L_0 - L)$$



Determine the first derivatives of the non-linear constitutive equations as a first step to linearize the equations.

$$\frac{\partial f_x}{\partial x} = \frac{\partial}{\partial x}\left[\varepsilon x\left(\frac{L_0}{L} - 1\right)\right] = \frac{\partial}{\partial x}\left[\varepsilon x\left(\frac{L_0}{\sqrt{x^2 + y^2}} - 1\right)\right] = \varepsilon\left(\frac{L_0 y^2}{\left(x^2 + y^2\right)^{3/2}} - 1\right)$$

$$\frac{\partial f_x}{\partial y} = \frac{\partial}{\partial y}\left[\varepsilon x\left(\frac{L_0}{\sqrt{x^2 + y^2}} - 1\right)\right] = \varepsilon\frac{-L_0 xy}{\left(x^2 + y^2\right)^{3/2}}$$

$$\frac{\partial f_y}{\partial x} = \frac{\partial}{\partial x}\left[\varepsilon y\left(\frac{L_0}{\sqrt{x^2 + y^2}} - 1\right)\right] = \varepsilon\frac{-L_0 xy}{\left(x^2 + y^2\right)^{3/2}}$$

$$\frac{\partial f_y}{\partial y} = \frac{\partial}{\partial y}\left[\varepsilon y\left(\frac{L_0}{\sqrt{x^2 + y^2}} - 1\right)\right] = \varepsilon\left(\frac{L_0 x^2}{\left(x^2 + y^2\right)^{3/2}} - 1\right)$$

Evaluate these first derivatives about the point (x_0, y_0).

Linearization becomes

$$f_x \approx \left.\frac{\partial f_x}{\partial x}\right|_{(x_0,y_0)}(x - x_0) + \left.\frac{\partial f_x}{\partial y}\right|_{(x_0,y_0)}(y - y_0)$$

$$f_y \approx \left.\frac{\partial f_y}{\partial x}\right|_{(x_0,y_0)}(x - x_0) + \left.\frac{\partial f_y}{\partial y}\right|_{(x_0,y_0)}(y - y_0)$$

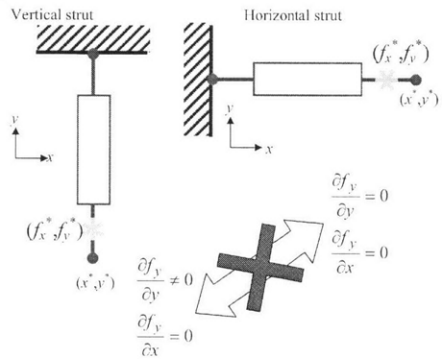
Linearization is only a valid estimate if the strut is very close to its original configuration – x and y are very close to x_0 and y_0.

What if the strut is *rotated*, is this linearization still valid?

It seems like it should be because the strut is not stretched (only rotated), but it isn't. Because the non-linearity is ***not*** in the relationship between how much the strut is stretched and the force going through the strut but the non-linearity is in the projection of that force onto the x and y axis.

Referred to as the "geometric" nonlinearity in finite element literature.

Example.

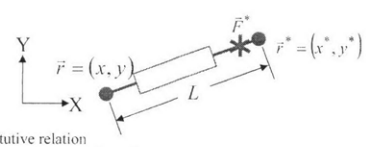


$$F_x^* = \frac{x^* - x}{L}\varepsilon(L_0 - L)$$

$$F_y^* = \frac{y^* - y}{L}\varepsilon(L_0 - L)$$

Note: NON-LINEAR

$$L = \sqrt{x^{*2} + y^{*2}}$$



LINEARIZE the constitutive relation

If x^*, y^* are close to some $x_0^*, y_0^* \rightarrow x_0^{*2} + y_0^{*2} = L_0^2$

$$\begin{bmatrix} f_x^* \\ f_y^* \end{bmatrix} = \underbrace{\begin{bmatrix} \frac{\partial F_x^*}{\partial x_1^*}(x_0^*, y_0^*) & \frac{\partial F_x^*}{\partial y_1^*}(x_0^*, y_0^*) \\ \frac{\partial F_y^*}{\partial x_1^*}(x_0^*, y_0^*) & \frac{\partial F_y^*}{\partial y_1^*}(x_0^*, y_0^*) \end{bmatrix}}_{\text{Jacobian Matrix}} \underbrace{\begin{bmatrix} u_x^* \\ u_y^* \end{bmatrix}}_{\Delta x}$$

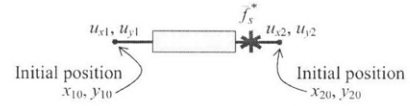
$$\vec{u}^* = (x^* - x_0^*, y^* - y_0^*)$$

$$\vec{u} = (x - x_0, y - y_0)$$

$$\vec{u}_b = \vec{u}^* - \vec{u}$$

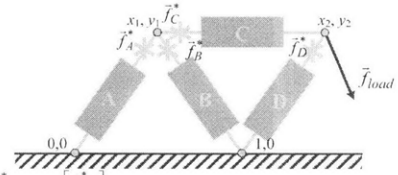
We will learn more about linearizing equations later on when we study Newton's Method.

The $\alpha(s,s)$ block



$$\alpha(s,s) = \begin{bmatrix} \frac{\partial F_x}{\partial x}(x_{2_0} - x_{1_0}, y_{2_0} - y_{1_0}) & \frac{\partial F_x}{\partial y}(x_{2_0} - x_{1_0}, y_{2_0} - y_{1_0}) \\ \frac{\partial F_y}{\partial x}(x_{2_0} - x_{1_0}, y_{2_0} - y_{1_0}) & \frac{\partial F_y}{\partial y}(x_{2_0} - x_{1_0}, y_{2_0} - y_{1_0}) \end{bmatrix}$$

Note that the arguments in the α matrix are the difference in the original position between node 1 and node 2. This is because the force of the strut is dependent upon the relative position of the two joints on either side.



$$f^* = \alpha u_b$$

$$u_b = A^T u$$

$$\begin{bmatrix} f_{A,x}^* \\ f_{A,y}^* \\ f_{B,x}^* \\ f_{B,y}^* \\ f_{C,x}^* \\ f_{C,y}^* \\ f_{D,x}^* \\ f_{D,y}^* \end{bmatrix} = \begin{bmatrix} & & \\ & \alpha & \\ & & \end{bmatrix} \begin{bmatrix} & & \\ & A^T & \\ & & \end{bmatrix} \begin{bmatrix} u_{x,1} \\ u_{y,1} \\ u_{x,2} \\ u_{y,2} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

- Node-Branch Form

  $f_S - \alpha A^T u = 0$ Constitutive Equation

  $A f_S = f_L$ Conservation Law

$$\begin{array}{c} 2\cdot S \updownarrow \\ 2\cdot J \updownarrow \end{array} \begin{bmatrix} I & -\alpha A^T \\ A & 0 \end{bmatrix} \begin{bmatrix} f_S \\ u \end{bmatrix} = \begin{bmatrix} 0 \\ f_L \end{bmatrix}$$

$$\begin{array}{cc} \overleftrightarrow{2\cdot S} & \overleftrightarrow{2\cdot J} \end{array}$$

S = Number of Struts
J = Number of Unfixed Joints

Comparison

struts

$$\begin{array}{c} 2\cdot S \updownarrow \\ 2\cdot J \updownarrow \end{array} \begin{bmatrix} I & -\alpha A^T \\ A & 0 \end{bmatrix} \begin{bmatrix} f_S \\ u \end{bmatrix} = \begin{bmatrix} 0 \\ f_L \end{bmatrix}$$

$$\begin{array}{cc} \overleftrightarrow{2\cdot S} & \overleftrightarrow{2\cdot J} \end{array}$$

circuit

$$\begin{array}{c} B \updownarrow \\ N \updownarrow \end{array} \begin{bmatrix} I & -\alpha A^T \\ A & 0 \end{bmatrix} \begin{bmatrix} I_b \\ V_N \end{bmatrix} = \begin{bmatrix} 0 \\ I_s \end{bmatrix}$$

$$\begin{array}{cc} \overleftrightarrow{B} & \overleftrightarrow{N} \end{array}$$

Summary of Key points…

- Developed algorithms for automatically constructing matrix equations from schematics using
  - Conservation law
  - Constitutive equations
- Looked at one formulation: node-branch
- Next time: nodal formulation

3.3 Equation Formulation - Nodal Analysis

INTRODUCTION TO NUMERICAL SIMULATION

LECTURE 3. Equation Formulation – Nodal Analysis

TODAY'S OUTLINE:

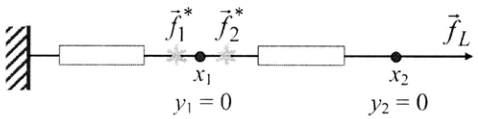
- Matrix Construction from Schematics
  - Nodal "Stamping Procedure"
    - Struts and Joints
    - Circuits
  - Comparing Node-Branch vs. Nodal
- Solution of Linear Systems
  - Existence and Uniqueness

MATRIX CONSTRUCTION FROM SCHEMATICS

Nodal "Stamping Procedure"

❖ Struts and Joints

Two struts aligned with the X axis



Conservation Law

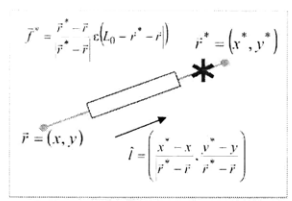
At node 1: $f_{1,x}^* + f_{2,x}^* = 0$

At node 2: $-f_{2,x}^* + f_L = 0$

Constitutive Equations

$$f_{1,x}^* = \frac{x_1 - 0}{|x_1 - 0|}\varepsilon(L_0 - |x_1 - 0|)$$

$$f_{2,x}^* = \frac{x_1 - x_2}{|x_1 - x_2|}\varepsilon(L_0 - |x_1 - x_2|)$$



Reduced (Nodal) Equations

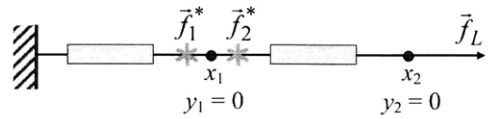
$$f_{1x}^* + f_{2x}^* = 0$$

$$\underbrace{\frac{x_1}{|x_1|}\varepsilon(L_0 - |x_1|)}_{f_{1,x}^*} + \underbrace{\frac{x_1 - x_2}{|x_1 - x_2|}\varepsilon(L_0 - |x_1 - x_2|)}_{f_{2,x}^*} = 0$$

$$-f_{2x}^* + f_L = 0$$

$$\underbrace{-\frac{x_1 - x_2}{|x_1 - x_2|}\varepsilon(L_0 - |x_1 - x_2|)}_{-f_{2,x}^*} + f_L = 0$$

Two struts aligned with the X axis



Example.) $\vec{f}_L = 10\hat{e}_1$ (force in positive x direction)

Solution of Nodal Equations

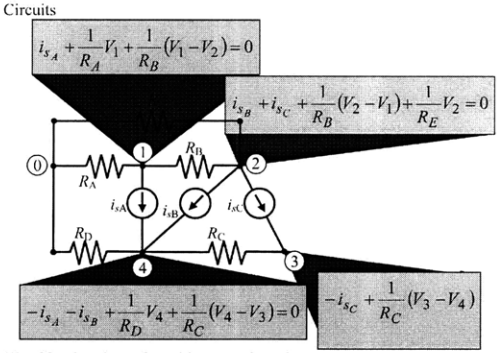
$$x_1 = L_0 + \frac{10}{\varepsilon} \qquad x_2 = x_1 + L_0 + \frac{10}{\varepsilon}$$

Notice the signs of the forces:

$f_{2,x}^* = 10$ force in positive x direction

$f_{1,x}^* = -10$ force in negative x direction

❖ Circuits



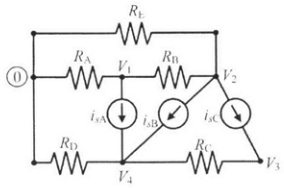
(1) Number the nodes with one node as 0.

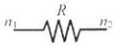
(2) Write a conservation law at each node except (0) in terms of the node voltages!

$$i_k = \frac{1}{R_k}\left(V_{n_2} - V_{n_1}\right)$$

at n_1: $\sum i_{other} - \frac{1}{R_k}\left(V_{n_2} - V_{n_1}\right) = \sum i_s$

at n_2: $\sum i_{other} + \frac{1}{R_k}\left(V_{n_2} - V_{n_1}\right) = \sum i_s$





One row per node,
One column per node
For each resistor

$$KCL\text{ equations}\left\{\underbrace{\begin{bmatrix} \frac{1}{R_A}+\frac{1}{R_B} & -\frac{1}{R_B} & & \\ -\frac{1}{R_B} & \frac{1}{R_B}+\frac{1}{R_E} & & \\ & & \frac{1}{R_C} & -\frac{1}{R_C} \\ & & -\frac{1}{R_C} & \frac{1}{R_C}+\frac{1}{R_D} \end{bmatrix}}_{G}\begin{bmatrix} V_1 \\ V_2 \\ V_3 \\ V_4 \end{bmatrix} = \underbrace{\begin{bmatrix} -i_{s_A} \\ -i_{s_B}-i_{s_C} \\ i_{s_C} \\ i_{s_A}+i_{s_B} \end{bmatrix}}_{I_s}\right.$$

Notice that the contributions are positive on the diagonal and negative on the off-diagonal.
G is square.

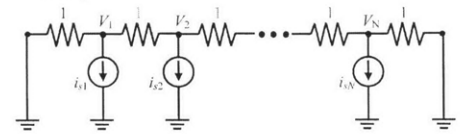
Node-Branch Matrix

$$\begin{array}{c} \\ N+B \end{array}\left[\begin{array}{cc} I & -\alpha A^T \\ A & 0 \end{array}\right] \quad \underbrace{\qquad\qquad}_{N+B}$$

Nodal Matrix

$$N\left\updownarrow\right.[G] \quad \underbrace{\qquad}_{N}$$

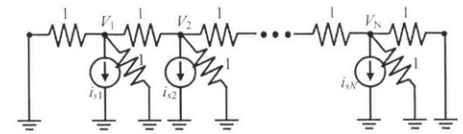
Circuit Example.)



What does the nodal form for the equations look like?

$$\begin{bmatrix} 1+1 & -1 & & & & \\ -1 & 1+1 & -1 & & & \\ & -1 & 1+1+1 & -1 & & \\ & & \ddots & \ddots & \ddots & \\ & & & -1 & 1+1 & -1 \\ & & & & -1 & 1+1 \end{bmatrix}\begin{bmatrix} V_1 \\ V_2 \\ \vdots \\ V_N \end{bmatrix} = \begin{bmatrix} i_{s1} \\ i_{s2} \\ \vdots \\ i_{sN} \end{bmatrix}$$

Diagonally dominant



What does the nodal form for the equations look like?

$$\begin{bmatrix} 1+1+1 & -1 & & & & \\ -1 & 1+1+1 & -1 & & & \\ & -1 & 1+1+1 & -1 & & \\ & & \ddots & \ddots & \ddots & \\ & & & -1 & 1+1+1 & -1 \\ & & & & -1 & 1+1+1 \end{bmatrix}\begin{bmatrix} V_1 \\ V_2 \\ \vdots \\ V_N \end{bmatrix} = \begin{bmatrix} i_{s1} \\ i_{s2} \\ \vdots \\ i_{sN} \end{bmatrix}$$

Strictly
Diagonally
Dominant

Nodal Matrix Generation Algorithm

if $(n_1 > 0)$ & $(n_2 > 0)$

$$G(n_1,n_1) = G(n_1,n_1) + \frac{1}{R} \qquad G(n_2,n_1) = G(n_2,n_1) - \frac{1}{R}$$

$$G(n_1,n_2) = G(n_1,n_2) - \frac{1}{R} \qquad G(n_2,n_2) = G(n_2,n_2) + \frac{1}{R}$$

else if $(n_1 > 0)$

$$G(n_1,n_1) = G(n_1,n_1) + \frac{1}{R}$$

else

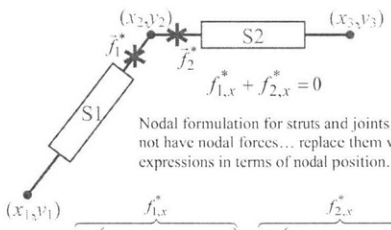
$$G(n_2,n_2) = G(n_2,n_2) + \frac{1}{R}$$

Resistor Networks

$$\underset{N}{\underset{\Updownarrow}{N \Phi}}\ \boldsymbol{G}\ \boldsymbol{V_n} = \boldsymbol{I_s}$$

Struts and Joints

$$\underset{2 \cdot J}{\underset{\Updownarrow}{2 \cdot J \Phi}}\ \boldsymbol{G}\ \boldsymbol{u_j} = \boldsymbol{F_l}$$



$$f_{1,x}^* + f_{2,x}^* = 0$$

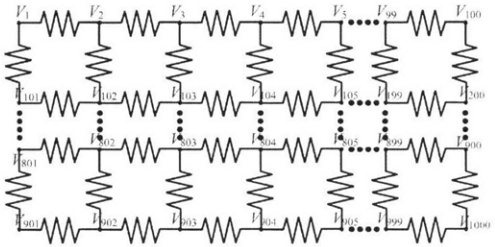
Nodal formulation for struts and joints will not have nodal forces… replace them with expressions in terms of nodal position.

$$\overbrace{F_x(x_2 - x_1, y_2 - y_1)}^{f_{1,x}^*} + \overbrace{F_x(x_2 - x_3, y_2 - y_3)}^{f_{2,x}^*} = 0$$

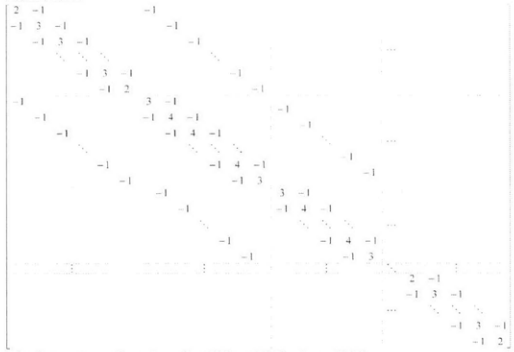
Comparing Node-Branch vs. Nodal

❖ Comparing Matrix Sparsity

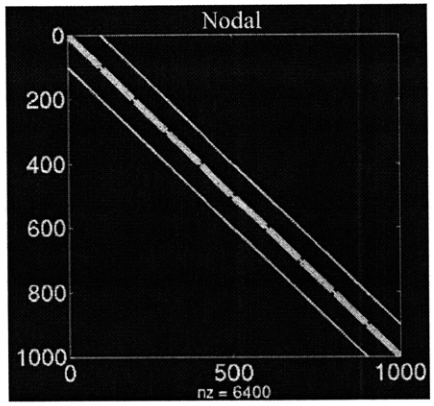
Example.) Resistor Grid

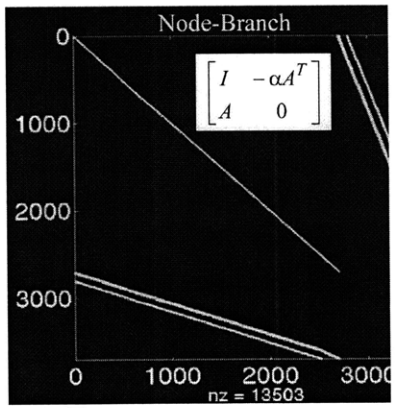


Nodal form



Matrix non-zero locations for 100 × 10 Resistor Grid.





Node-Branch Matrix

$$\begin{array}{r} \text{Constitutive} \\ \text{Conservation Law} \end{array} \begin{bmatrix} I & -\alpha A^T \\ A & 0 \end{bmatrix} \begin{bmatrix} I_b \\ V_N \end{bmatrix} = \begin{bmatrix} 0 \\ I_s \end{bmatrix}$$

Nodal Matrix

Conservation Law $[G][V_N] = [I_s]$

(constitutive pre-substituted)

- G matrix properties
  - Smaller $\quad N \times N \;<<\; (N+B)\times(N+B)$
    $\quad 2J \times 2J \;<<\; (2J+2S)\times(2J+2S)$
  - Symmetric $\quad G_{ij} = G_{ji}$
  - Diagonally Dominant $\quad |G_{ii}| \geq \sum_{j \neq i} |G_{ij}|$
- Node-Branch Form

$$\underbrace{\begin{bmatrix} I & -\alpha A^T \\ A & 0 \end{bmatrix}}_{M} \underbrace{\begin{bmatrix} I_b \\ V_n \end{bmatrix}}_{x} = \underbrace{\begin{bmatrix} 0 \\ I_s \end{bmatrix}}_{b}$$

  - Not Symmetric, nor Diagonally Dominant
  - Matrix is $(n+b)\times(n+b)$
- Deriving Formulation from Node-Branch constitutive equations

$$I_b - \alpha A^T V_N = 0 \;\Rightarrow\; I_b = \alpha A^T V_N$$

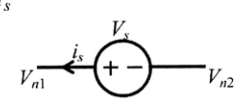
substitute into conservation equations:

$$A I_b = I_s \;\Rightarrow\; \underbrace{A \alpha A^T}_{G} V_N = I_s$$

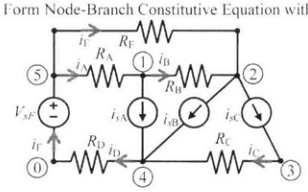
- Problem Element
  - Voltage Source

    Constitutive Equation

    $0 \cdot i_s + V_{n_1} - V_{n_2} = V_s$



Form Node-Branch Constitutive Equation with Voltage Source



$$\text{KCL 1:}\quad \ldots + \frac{1}{R_A}(V_1 - V_{sF}) + \ldots = RHS$$

$$\text{KCL 2:}\quad \ldots + \frac{1}{R_E}(V_2 - V_{sF}) + \ldots = RHS$$

$$\begin{bmatrix} 1 & & & & & \\ & 1 & & & & \\ & & 1 & & & \\ & & & 1 & & \\ & & & & 1 & \\ & & & & & 0 \end{bmatrix} \begin{bmatrix} i_A \\ i_B \\ i_C \\ i_D \\ i_E \\ i_F \end{bmatrix} - \alpha A^T \begin{bmatrix} V_1 \\ V_2 \\ V_3 \\ V_4 \\ V_5 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ -V_{sF} \end{bmatrix}$$

$$0 \cdot i_F - 1 \cdot (V_5 - 0) = -V_{sF}$$

$$\alpha = \begin{bmatrix} \frac{1}{R_A} & & & & & \\ & \frac{1}{R_B} & & & & \\ & & \frac{1}{R_C} & & & \\ & & & \frac{1}{R_D} & & \\ & & & & \frac{1}{R_E} & \\ & & & & & 1 \end{bmatrix}$$

Cannot Derive Nodal Formulation

resistor currents → $\begin{bmatrix} I_{br} \\ 0 \end{bmatrix} - \alpha A^T V_N = \begin{bmatrix} 0 \\ V_s \end{bmatrix}$ (Constitutive Equation)

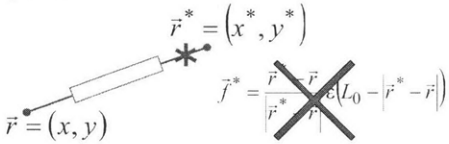
voltage source currents missing → 0

$$A \cdot I_b = I_s \qquad \text{(Conservation Law)}$$

Cannot Eliminate I_b since $I_b \neq \begin{bmatrix} I_{br} \\ 0 \end{bmatrix}$

Nodal Formulation requires Constitutive relations in the form:
Conserved Quantity = F (Node Voltages)!

Rigid Rod



$$\sqrt{(x^* - x)^2 + (y^* - y)^2} = L_{fixed} \qquad \text{constitutive equation}$$

The constitutive equation does not contain forces!!

- Summary of Equation Formulation
  - Developed algorithms for automatically constructing matrix equations from schematics
    - Used conservation laws + constitutive equations
    - Node-branch
      - General constitutive equations
      - Large sparser system
      - No diagonal dominance
    - Nodal
      - Conserved quantity must be a function of node variables
      - Smaller denser system
      - Diagonally dominant & symmetric

Solution of Linear Systems

$M\vec{x} = \vec{b}$

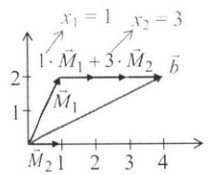
$$\begin{bmatrix} \uparrow & \uparrow & \cdots & \uparrow \\ \vec{M}_1 & \vec{M}_2 & \cdots & \vec{M}_N \\ \downarrow & \downarrow & \cdots & \downarrow \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_N \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_N \end{bmatrix} \Rightarrow x_1\vec{M}_1 + x_2\vec{M}_2 + \ldots + x_N\vec{M}_N = \vec{b}$$

Find a set of weights, $\boldsymbol{x}$, so that the weighted sum of the columns of the matrix **M** is equal to the right hand side **b**.

$$\overbrace{\begin{bmatrix} \underbrace{\begin{bmatrix} 1 \\ 2 \end{bmatrix}}_{\vec{M}_1} & \underbrace{\begin{bmatrix} 1 \\ 0 \end{bmatrix}}_{\vec{M}_2} \end{bmatrix}}^{M} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \underbrace{\begin{bmatrix} 4 \\ 2 \end{bmatrix}}_{\vec{b}}$$

$$x_1\vec{M}_1 + x_2\vec{M}_2 = \vec{b}$$

$$x_1\begin{bmatrix} 1 \\ 2 \end{bmatrix} + x_2\begin{bmatrix} 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 4 \\ 2 \end{bmatrix}$$



Key Questions

- Given $\mathbf{M}\boldsymbol{x} = \mathbf{b}$
  - Is there a solution?
  - Is the solution unique?

If $M = \begin{bmatrix} 1 & 1 & 0 \\ 0 & 1 & 1 \\ 0 & 0 & 0 \end{bmatrix}$ Can $b = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}$?

No – so change M so that b can be a solution $\Longrightarrow$ $M = \begin{bmatrix} 1 & 1 & 0 \\ 0 & 1 & 1 \\ 1 & 0 & 1 \end{bmatrix}$

What if $M = \begin{bmatrix} 1 & 1 & 2 \\ 0 & 1 & 1 \\ 1 & 0 & 1 \end{bmatrix}$ Note that the third column is the sum of the first two columns in this case. The columns of M are not linearly independent.

Can $b = \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix}$? $x = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}$

Can $b = \begin{bmatrix} 4 \\ 3 \\ 1 \end{bmatrix}$? $x = \begin{bmatrix} 1 \\ 3 \\ 0 \end{bmatrix}$ or $x = \begin{bmatrix} 0 \\ 2 \\ 1 \end{bmatrix}$

$$My = 0 \Longrightarrow y = \begin{bmatrix} 1 \\ 1 \\ -1 \end{bmatrix}$$

Existence and Uniqueness

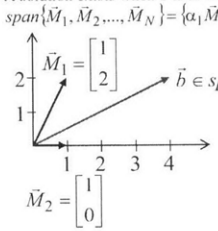
- Is there a solution?

  There exist weights $x_1, \ldots, x_N$, such that

  $x_1\vec{M}_1 + x_2\vec{M}_2 + \ldots + x_N\vec{M}_N = \vec{b}$

  A solution exists when **b** is in the span of the columns of **M**

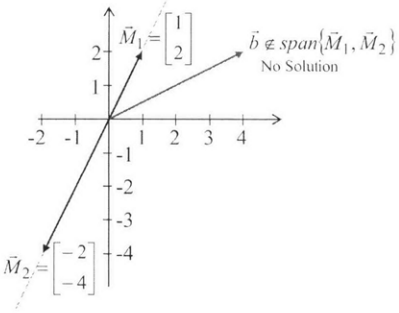
  $span\{\vec{M}_1, \vec{M}_2, \ldots, \vec{M}_N\} = \{\alpha_1\vec{M}_1 + \alpha_2\vec{M}_2 + \ldots + \alpha_N\vec{M}_N, \alpha_i \in \Re\}$



$$\vec{b} \in span\{\vec{M}_1, \vec{M}_2\} = \alpha_1\begin{bmatrix} 1 \\ 2 \end{bmatrix} + \alpha_2\begin{bmatrix} 1 \\ 0 \end{bmatrix} = \Re^2$$

$$\vec{b} = \begin{bmatrix} 4 \\ 2 \end{bmatrix}$$

$$\alpha_1\vec{M}_1 + \alpha_2\vec{M}_2 = span\{\vec{M}_1, \vec{M}_2\}$$

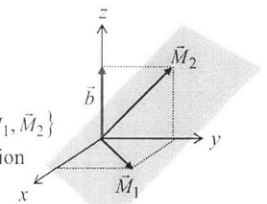


3-D

$$\begin{bmatrix}\begin{bmatrix}1\\1\\0\end{bmatrix} & \begin{bmatrix}0\\1\\1\end{bmatrix}\end{bmatrix}\begin{bmatrix}x_1\\x_2\end{bmatrix} = \begin{bmatrix}0\\0\\1\end{bmatrix}$$

$\vec{M}_1 \quad \vec{M}_2 \quad \vec{b} \notin span\{\vec{M}_1, \vec{M}_2\}$

➡ No Solution



$span\{\vec{M}_1, \vec{M}_2\}$
$= \alpha_1\vec{M}_1 + \alpha_2\vec{M}_2$

❖ Is the solution unique?

Suppose there exist weights, $y_1, \ldots, y_N$, not all zero

$y_1\vec{M}_1 + y_2\vec{M}_2 + \ldots + y_N\vec{M}_N = 0 \quad \rightarrow$[nullspace]

Then if $\mathbf{M}\vec{x} = \vec{b}$, then $\mathbf{M}(\vec{x} + \vec{y}) = \vec{b}$

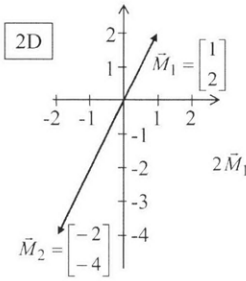
A solution is unique only if the columns of **M** are linearly independent.

Linearly independent columns means: $\sum_{i \neq k} \alpha_i \vec{M}_i \neq \vec{M}_k$

$\mathbf{M}\vec{x} = \vec{b}$	$\vec{x}$ is a solution
$\mathbf{M}\vec{y} = \mathbf{0}$	linearly dependent columns in **M**
$\mathbf{M}(\vec{x} + \vec{y}) = \vec{b}$	$\vec{x} + \vec{y}$ is a solution

If $\mathbf{M}\vec{y} = \mathbf{0}$ then $\alpha\mathbf{M}\vec{y} = \mathbf{0} \Rightarrow \mathbf{M}(\vec{x} + \alpha\vec{y}) = \vec{b} \Rightarrow$ infinite solutions

2D



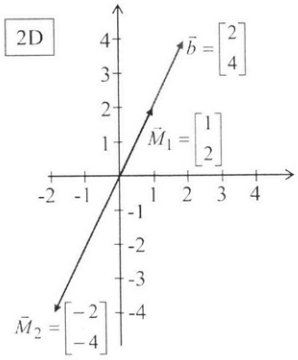
$$\begin{bmatrix}1 & -2\\2 & -4\end{bmatrix}\begin{bmatrix}y_1\\y_2\end{bmatrix} = [0]$$

$\vec{M}_1 \quad \vec{M}_2$

$$2\vec{M}_1 + \vec{M}_2 = 0$$

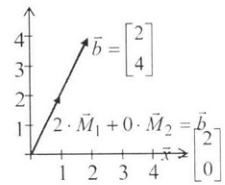
$$\vec{y} = \begin{bmatrix}2\\1\end{bmatrix}$$

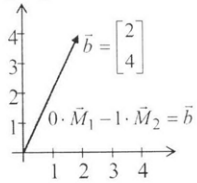
2D

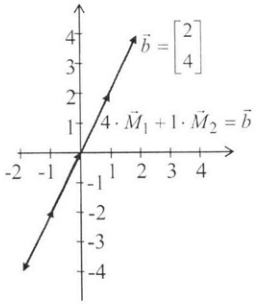


$$\begin{bmatrix} 1 & -2 \\ 2 & -4 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 2 \\ 4 \end{bmatrix}$$

$\vec{M}_1 \quad \vec{M}_2 \qquad \vec{b}$



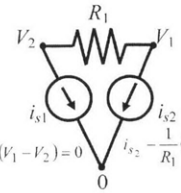




Infinite number of solutions, are of the form:

$$\vec{x} = \begin{bmatrix} 2 \\ 0 \end{bmatrix} + \alpha \begin{bmatrix} 2 \\ 1 \end{bmatrix}$$

Physical Circuit Example.)



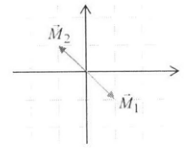
$$\begin{bmatrix} \frac{1}{R_1} & -\frac{1}{R_1} \\ -\frac{1}{R_1} & \frac{1}{R_1} \end{bmatrix} \begin{bmatrix} V_1 \\ V_2 \end{bmatrix} = \begin{bmatrix} -i_{s_1} \\ -i_{s_2} \end{bmatrix}$$

$$i_{s_1} + \frac{1}{R_1}(V_1 - V_2) = 0 \qquad i_{s_2} - \frac{1}{R_1}(V_1 - V_2) = 0$$

Let R_1=1, is there a right hand side where the system can be solved?

$$\begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix} \begin{bmatrix} V_1 \\ V_2 \end{bmatrix} = \begin{bmatrix} -i_{s_1} \\ -i_{s_2} \end{bmatrix} \boxed{= k \begin{bmatrix} 1 \\ -1 \end{bmatrix}}$$

$\vec{M}_1 \quad \vec{M}_2$



Physically this means that the current values must be equal and opposite. This makes sense as there is only one path for the current. Any k volt difference, where $V_1 = V_2 + k$ will give the solution.

L.D. → no solution / Infinite solns

e.g. $\vec{b} = \begin{bmatrix} -2 \\ +2 \end{bmatrix}$

$0\vec{M}_1 + 2\vec{M}_2 = \vec{b} \quad \rightarrow \quad V_1 = 0 \quad V_2 = 2$

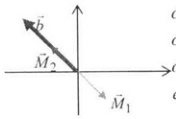
or $-\vec{M}_1 + \vec{M}_2 = \vec{b} \quad \rightarrow \quad V_1 = -1 \quad V_2 = 1$

or $8\vec{M}_1 + 10\vec{M}_2 = \vec{b} \quad \rightarrow \quad V_1 = 8 \quad V_2 = 10$

or $-2\vec{M}_1 + 0\vec{M}_2 = \vec{b} \quad \rightarrow \quad V_1 = -2 \quad V_2 = 0$

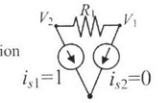
or $-103\vec{M}_1 + 105\vec{M}_2 = \vec{b} \quad \rightarrow \quad V_1 = -103 \quad V_2 = 105$

etc...



Physically we can shift V_1 and V_2 by the same amount!

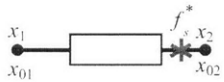
e.g. $\vec{b} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$ no solution



Where does this current (from i_{s1}) go?

Makes no sense.

Struts and Joints Example.)



Strut is on the x-axis

Forces must be equal and opposite

$u_{x_1} = x_1 - x_{01}$ $\qquad$ $u_{x_2} = x_2 - x_{02}$

$$f_2 = f_x^* = \varepsilon[(x_{02} - x_{01}) - (x_2 - x_1)] = \varepsilon(u_{x_1} - u_{x_2})$$

$$f_1 = -f_x^*$$

$$\begin{bmatrix} -\varepsilon & +\varepsilon \\ +\varepsilon & -\varepsilon \end{bmatrix} \begin{bmatrix} u_{x_1} \\ u_{x_2} \end{bmatrix} = \begin{bmatrix} f_1 \\ f_2 \end{bmatrix} \qquad \begin{aligned} -f_1 - \varepsilon(u_{x_1} - u_{x_2}) &= 0 \\ -f_2 + \varepsilon(u_{x_1} - u_{x_2}) &= 0 \end{aligned}$$

$$u_{x_1} \begin{bmatrix} \varepsilon \\ -\varepsilon \end{bmatrix} + u_{x_2} \begin{bmatrix} -\varepsilon \\ \varepsilon \end{bmatrix} = \begin{bmatrix} f_1 \\ f_2 \end{bmatrix}$$

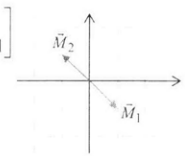
The solution is in the span of the matrix columns if $f_1 = -f_2$

Solution is not unique because for $y = \begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$

$$1 \cdot \begin{bmatrix} \varepsilon \\ -\varepsilon \end{bmatrix} + 1 \cdot \begin{bmatrix} -\varepsilon \\ \varepsilon \end{bmatrix} = 0!$$

Singular System.

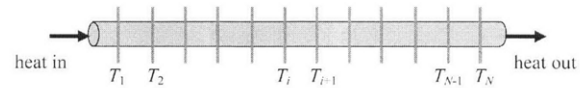
$$\begin{bmatrix} -1 & 1 \\ 1 & -1 \end{bmatrix}$$



If the solution physically exists, need $\vec{f}_1 = -\vec{f}_2$

there will be an infinite number of solutions

Heat Conducting Bar Example.)



Boundary conditions

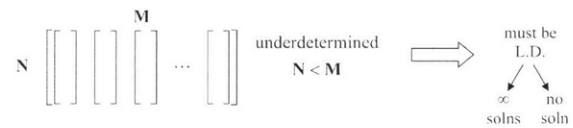
if heat in = heat out → no unique solution

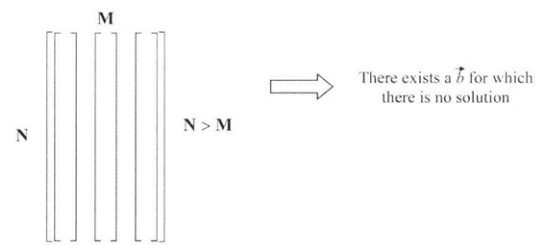
Could displace solution by a factor of 100, and it would still solve the system

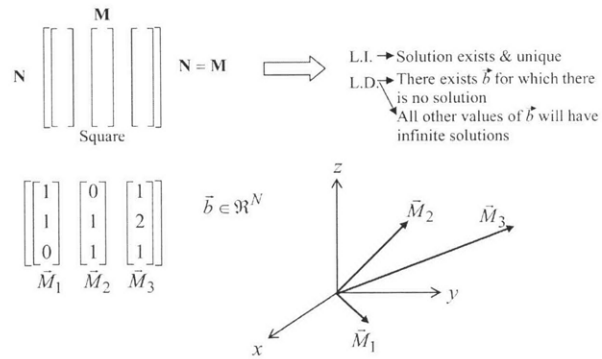
if heat in ≠ heat out → no solution

Summary Table

	$b \in \text{range}\{M\}$	$b \notin \text{range}\{M\}$
L.I. columns	Solution exists and is unique	No solution
L.D. columns	Infinite solutions	No solution







Square Matrices

Given $\mathbf{M}\boldsymbol{x} = \mathbf{b}$, where $\mathbf{M}$ is square

If a solution exists for all $\mathbf{b}$, then the solution for a specific $\mathbf{b}$ is unique.

For a solution to exist for any $\mathbf{b}$, the columns of $\mathbf{M}$ must span all $\mathbf{N}$-length vectors.

Since there are only $\mathbf{N}$ columns of the matrix $\mathbf{M}$ to span this space, these vectors must be linearly independent.

A square matrix with linearly independent columns is said to be **nonsingular**.

3.4 Linear Systems - LU

INTRODUCTION TO NUMERICAL SIMULATION

LECTURE 4. Linear Systems – LU Decomposition

TODAY'S OUTLINE:

- Solution of Linear Systems
  - Gaussian Elimination Basics
    - LU factorization
    - Computational Complexity
    - Pivoting for Growth Control

SOLUTION OF LINEAR SYSTEMS

Gaussian Elimination Basics

❖ LU Factorization

- Important Properties

  Gaussian Elimination Method for solving $\mathbf{M}\vec{x} = \vec{b}$
  - A "Direct" Method
    Finite Termination for exact result (ignoring roundoff error)
  - Produces accurate results for a broad range of matrices
  - Computationally expensive
- Reminder by Example

  3×3 Example

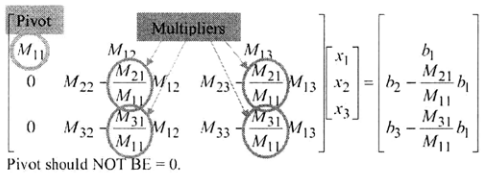
$$\begin{bmatrix} M_{11} & M_{12} & M_{13} \\ M_{21} & M_{22} & M_{23} \\ M_{31} & M_{32} & M_{33} \end{bmatrix}\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix} \Longrightarrow \begin{aligned} M_{11}x_1 + M_{12}x_2 + M_{13}x_3 &= b_1 \\ M_{21}x_1 + M_{22}x_2 + M_{23}x_3 &= b_2 \\ M_{31}x_1 + M_{32}x_2 + M_{33}x_3 &= b_3 \end{aligned}$$

  Use equation 1 to eliminate x_1 from equation 2 and 3

$$M_{11}x_1 + M_{12}x_2 + M_{13}x_3 = b_1$$

$$\left(M_{21} - \frac{M_{21}}{M_{11}}M_{11}\right)x_1 + \left(M_{22} - \frac{M_{21}}{M_{11}}M_{12}\right)x_2 + \left(M_{23} - \frac{M_{21}}{M_{11}}M_{13}\right)x_3 = b_2 - \frac{M_{21}}{M_{11}}b_1$$

$$\left(M_{31} - \frac{M_{31}}{M_{11}}M_{11}\right)x_1 + \left(M_{32} - \frac{M_{31}}{M_{11}}M_{12}\right)x_2 + \left(M_{33} - \frac{M_{31}}{M_{11}}M_{13}\right)x_3 = b_3 - \frac{M_{31}}{M_{11}}b_1$$



$$\begin{bmatrix} M_{11} & M_{12} & M_{13} \\ 0 & M_{22} - \frac{M_{21}}{M_{11}}M_{12} & M_{23} - \frac{M_{21}}{M_{11}}M_{13} \\ 0 & M_{32} - \frac{M_{31}}{M_{11}}M_{12} & M_{33} - \frac{M_{31}}{M_{11}}M_{13} \end{bmatrix}\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 - \frac{M_{21}}{M_{11}}b_1 \\ b_3 - \frac{M_{31}}{M_{11}}b_1 \end{bmatrix}$$

  Pivot should NOT BE = 0.

  Simplify the notation

$$\begin{bmatrix} M_{11} & M_{12} & M_{13} \\ 0 & \tilde{M}_{22} & \tilde{M}_{23} \\ 0 & \tilde{M}_{32} & \tilde{M}_{33} \end{bmatrix}\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} b_1 \\ \tilde{b}_2 \\ \tilde{b}_3 \end{bmatrix}$$

Use updated equation 2 to eliminate x_2 from the updated equation 3

$$\begin{bmatrix} M_{11} & M_{12} & M_{13} \\ 0 & \tilde{M}_{22} & \tilde{M}_{23} \\ 0 & 0 & \tilde{M}_{33} - \frac{\tilde{M}_{32}}{\tilde{M}_{22}}\tilde{M}_{23} \end{bmatrix}\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} b_1 \\ \tilde{b}_2 \\ \tilde{b}_3 - \frac{\tilde{M}_{32}}{\tilde{M}_{22}}\tilde{b}_2 \end{bmatrix}$$

Pivot

Multiplier

Right-hand side

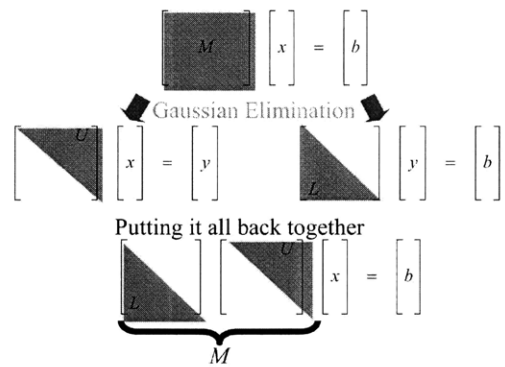
$$\begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 - \left(\frac{M_{21}}{M_{11}}\right)b_1 \\ b_3 - \left(\frac{M_{31}}{M_{11}}\right)b_1 - \frac{\tilde{M}_{32}}{\tilde{M}_{22}}\tilde{b}_2 \end{bmatrix} \Longrightarrow \begin{bmatrix} 1 & 0 & 0 \\ \frac{M_{21}}{M_{11}} & 1 & 0 \\ \frac{M_{31}}{M_{11}} & \frac{\tilde{M}_{32}}{\tilde{M}_{22}} & 1 \end{bmatrix}\begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix}$$

First loop multipliers

Second loop multiplier

$$\mathbf{M}\vec{x} = \vec{b} \quad \Rightarrow \overbrace{\mathbf{LU}}^{\mathbf{M}}\vec{x} = \vec{b} \qquad \mathbf{M} \leftrightarrow \mathbf{LU}$$

$$\Rightarrow \mathbf{L}\underbrace{\mathbf{U}\vec{x}}_{\vec{y}} = \vec{b} \quad \begin{cases} \mathbf{U}\vec{x} = \vec{y} \\ \mathbf{L}\vec{y} = \vec{b} \end{cases}$$



Fitting the pieces together

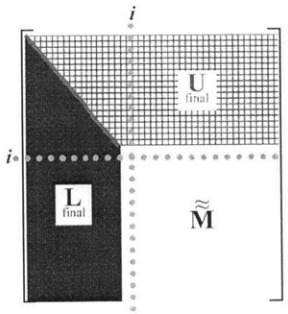
$$\begin{bmatrix} 1 & & \\ \frac{M_{21}}{M_{11}} & 1 & \\ \frac{M_{31}}{M_{11}} & \frac{\widetilde{M}_{32}}{\widetilde{M}_{22}} & 1 \end{bmatrix} \begin{bmatrix} M_{11} & M_{12} & M_{13} \\ & \widetilde{M}_{22} & \widetilde{M}_{23} \\ & & \widetilde{\widetilde{M}}_{33} \end{bmatrix}$$

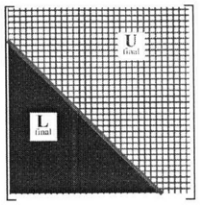
$$\boldsymbol{L} \qquad \boldsymbol{U}$$

$$\begin{bmatrix} M_{11} & M_{12} & M_{13} \\ \frac{M_{21}}{M_{11}} & \widetilde{M}_{22} & \widetilde{M}_{23} \\ \frac{M_{31}}{M_{11}} & \frac{\widetilde{M}_{32}}{\widetilde{M}_{22}} & \widetilde{\widetilde{M}}_{33} \end{bmatrix}$$

$$\boldsymbol{L}\ \boldsymbol{U}$$

Store all the data in one matrix – no extra memory needed.





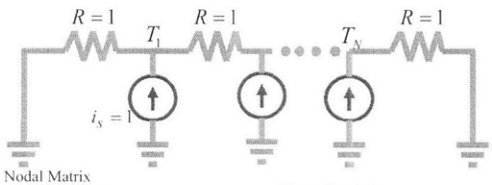
The **L** matrix will contain all subdiagonal elements with a diagonal of all 1's

- Factoring
  An "in place" implementation

$$\begin{bmatrix} M_{11} & M_{12} & M_{13} \\ M_{21} & M_{22} & M_{23} \\ M_{31} & M_{32} & M_{33} \end{bmatrix} \Rightarrow \begin{bmatrix} M_{11} & M_{12} & M_{13} \\ \frac{M_{21}}{M_{11}} & \widetilde{M}_{22} & \widetilde{M}_{23} \\ \frac{M_{31}}{M_{11}} & \widetilde{M}_{32} & \widetilde{M}_{33} \end{bmatrix}$$

$$\begin{bmatrix} M_{11} & M_{12} & M_{13} \\ \frac{M_{21}}{M_{11}} & \widetilde{M}_{22} & \widetilde{M}_{23} \\ \frac{M_{31}}{M_{11}} & \frac{\widetilde{M}_{32}}{\widetilde{M}_{22}} & \widetilde{\widetilde{M}}_{33} \end{bmatrix} \Leftarrow \begin{bmatrix} M_{11} & M_{12} & M_{13} \\ \frac{M_{21}}{M_{11}} & \widetilde{M}_{22} & \widetilde{M}_{23} \\ \frac{M_{31}}{M_{11}} & \widetilde{M}_{32} & \widetilde{M}_{33} \end{bmatrix}$$

- Example – Heat Flow
  Temperature analogous to Voltage
  Heat Flow analogous to Current



Nodal Matrix

$$\begin{bmatrix} 2 & -1 & & & & & \\ -1 & 2 & -1 & & & & \\ & -1 & 2 & -1 & & & \\ & & -1 & 2 & -1 & & \\ & & & \ddots & \ddots & \ddots & \\ & & & & -1 & 2 & -1 \\ & & & & & -1 & 2 \end{bmatrix} \begin{bmatrix} T_1 \\ T_2 \\ T_3 \\ T_4 \\ \vdots \\ T_{N-1} \\ T_N \end{bmatrix} = \begin{bmatrix} \vdots \\ 1 \\ \vdots \end{bmatrix}$$

$$G = \begin{bmatrix} 2 & -1 & & & & \\ -1 & 2 & -1 & & & \\ & -1 & 2 & -1 & & \\ & & \ddots & \ddots & \ddots & \\ & & & -1 & 2 & -1 \\ & & & & -1 & 2 \end{bmatrix} \Rightarrow \begin{bmatrix} 2 & -1 & & & & \\ -\frac{1}{2} & \frac{3}{2} & -1 & & & \\ & -1 & 2 & -1 & & \\ & & -1 & 2 & -1 & \\ & & & \ddots & \ddots & \ddots \end{bmatrix}$$

1st step multiplier

$$\Rightarrow \begin{bmatrix} 2 & -1 & & & & \\ -\frac{1}{2} & \frac{3}{2} & -1 & & & \\ & -\frac{2}{3} & \frac{4}{3} & -1 & & \\ & & -1 & 2 & -1 & \\ & & & \ddots & \ddots & \ddots \end{bmatrix}$$

2nd step multiplier

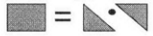
$$\Rightarrow \begin{bmatrix} 2 & -1 & & & & \\ -\frac{1}{2} & \frac{3}{2} & -1 & & & \\ & -\frac{2}{3} & \frac{4}{3} & -1 & & \\ & & -\frac{3}{4} & \frac{5}{4} & -1 & \\ & & & -\frac{4}{5} & \frac{6}{5} & \ddots \\ & & & & \ddots & \ddots \end{bmatrix}$$

In place LU decomposition

$$\mathbf{M} = \mathbf{L} \cdot \mathbf{U} = \begin{bmatrix} 1 & & & & & \\ -\frac{1}{2} & 1 & & & & \\ & -\frac{2}{3} & 1 & & & \\ & & -\frac{3}{4} & 1 & & \\ & & & \ddots & & \end{bmatrix} \begin{bmatrix} 2 & -1 & & & & \\ & \frac{3}{2} & -1 & & & \\ & & \frac{4}{3} & -1 & & \\ & & & \frac{5}{4} & & \\ & & & & \ddots & \end{bmatrix}$$

- Computational Complexity
  - Three Steps to Solving a System
    Solve $\mathbf{M}x = b$
    Step 1 Decomposition

$$\mathbf{M} = \mathbf{L} \cdot \mathbf{U}$$



    Step 2 Forward Elimination
    Solve $\mathbf{L}y = b$
    Step 3 Backward Substitution
    Solve $\mathbf{U}x = y$
  - Step 2: Solving Lower Triangular Systems

$$\begin{bmatrix} l_{11} & 0 & \cdots & 0 \\ l_{21} & l_{22} & & \vdots \\ l_{13} & & \ddots & \\ \vdots & & & 0 \\ l_{N1} & \cdots & & l_{NN} \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_N \end{bmatrix}$$

$$y_1 = \frac{1}{l_{11}} b_1$$

$$y_2 = \frac{1}{l_{22}} (b_2 - l_{21} y_1)$$

$$y_3 = \frac{1}{l_{33}} (b_3 - l_{31} y_1 - l_{32} y_2)$$

$$\vdots$$

$$y_N = \frac{1}{l_{NN}} \left(b_N - \sum_{i=1}^{N-1} l_{Ni} y_i \right)$$

Number of multiplications:

$$1 + 2 + 3 + 4 + \ldots + (n-1) = \sum_{i=1}^{n-1} i = \frac{(n-1)n}{2} = O(n^2)$$

- Step 3: Solving Upper Triangular Systems

$$\begin{bmatrix} U_{11} & U_{12} & U_{13} \\ 0 & U_{22} & U_{23} \\ 0 & 0 & U_{33} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix}$$

$$x_3 = \frac{y_3}{U_{33}}$$

$$x_2 = \frac{y_2 - U_{23}x_3}{U_{22}}$$

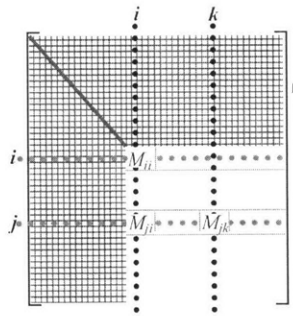
$$x_1 = \frac{y_1 - U_{12}x_2 - U_{13}x_3}{U_{11}}$$

Number of multiplications:

$$\sum_{i=1}^{n-1} i = \frac{(n-1)n}{2} = O(n^2)$$

- Step 1: Decomposition

$$\begin{bmatrix} M_{11} & M_{12} & M_{13} & M_{14} \\ \frac{M_{21}}{M_{11}} & M_{22} & M_{23} & M_{24} \\ \frac{M_{31}}{M_{11}} & \frac{M_{32}}{M_{22}} & M_{33} & M_{34} \\ \frac{M_{41}}{M_{11}} & \frac{M_{42}}{M_{22}} & \frac{M_{43}}{M_{33}} & M_{44} \end{bmatrix}$$



for $i = 1$ to $N - 1$
for $j = i+1$ to N
$\widetilde{M}_{ji} \leftarrow \dfrac{M_{ji}}{M_{ii}}$
for $k = i+1$ to N
$\widetilde{M}_{jk} \leftarrow M_{jk} - \widetilde{M}_{ji} \cdot M_{ik}$

What is the order? $O(n)$ $O(n!)$ $O(n^2)$ $O(n^3)$

- Algorithm

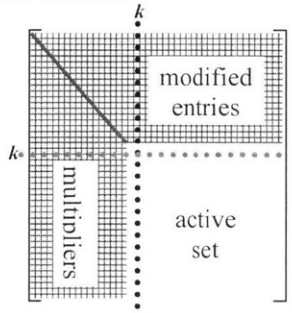
For $i = 1$ to $n - 1$ { "For each Row"
  For $j = i + 1$ to n { "For each target Row below the source"
    $M_{ji} = \dfrac{M_{ji}}{\boxed{M_{ii}}}$ Pivot $\qquad \sum_{i=1}^{n-1}(n-i) = \dfrac{n^2}{2}$ multipliers
    For $k = i + 1$ to n { "For each Row element beyond Pivot"
      $M_{jk} \leftarrow M_{jk} - \boxed{M_{ji}} M_{ik}$ $\qquad \sum_{i=1}^{n-1}(n-i)^2 \approx \dfrac{2}{3}n$
    } Multiplier Multiply-adds
  }
}

"Target Oriented"

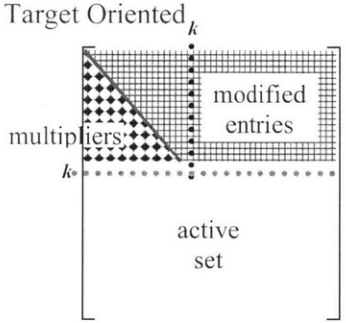
$$\begin{bmatrix} M_{11} & M_{12} & M_{13} & M_{14} \\ M_{21} & M_{22} & M_{23} & M_{24} \\ M_{31} & M_{32} & M_{33} & M_{34} \\ M_{41} & M_{42} & M_{43} & M_{44} \end{bmatrix}$$

for $i = 2$ to N "for each target"
  for $j = 1$ to $i - 1$ "for each source"
    $\widetilde{M}_{ij} \leftarrow \dfrac{M_{ij}}{M_{jj}}$
    target source
    for $k = j+1$ to N
      $\widetilde{M}_{ik} \leftarrow M_{ik} - \widetilde{M}_{ij} \cdot M_{jk}$

Source Oriented



Target Oriented

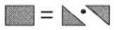


- Summary

  Solve $\mathbf{M}x = b$

  Step 1 Decomposition

  $\mathbf{M} = \mathbf{L} \cdot \mathbf{U}$

  $O(n^3)$

  Step 2 Forward Elimination

  Solve $\mathbf{L}y = b$

  $O(n^2)$

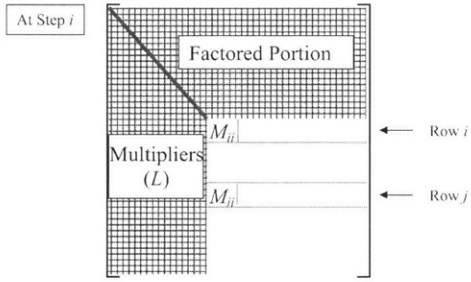
  Step 3 Backward Substitution

  Solve $\mathbf{U}x = y$

  $O(n^2)$

❖ Pivoting for Growth Control

- Zero Pivots



What if $M_{ii} = 0$?

Cannot form $\dfrac{M_{ji}}{M_{ii}}$

Simple Fix (Partial Pivoting)

If $M_{ii} = 0$

Find $M_{ji} \neq 0,\ j > i$

Swap Row j with Row i

Pick row j whose M_{ji} value is large

Swap rows 2 and 4

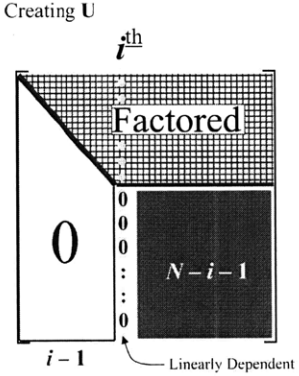
$$\begin{bmatrix} M_{11} & M_{12} & M_{13} & M_{14} \\ M_{21} & M_{22} & M_{23} & M_{24} \\ M_{31} & M_{32} & M_{33} & M_{34} \\ M_{41} & M_{42} & M_{43} & M_{44} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \end{bmatrix}$$

$$\Rightarrow \begin{bmatrix} M_{11} & M_{12} & M_{13} & M_{14} \\ M_{41} & M_{42} & M_{43} & M_{44} \\ M_{31} & M_{32} & M_{33} & M_{34} \\ M_{21} & M_{22} & M_{23} & M_{24} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} b_1 \\ b_4 \\ b_3 \\ b_2 \end{bmatrix}$$

Two Important Theorems

1. Partial pivoting (swapping rows) always succeeds if **M** is nonsingular.

Creating **U**



$(N-1)$ Linearly Independent Columns

2. **LU** factorization applied to a strictly diagonally dominant matrix will never produce a zero pivot.

- Small Pivots
  - Example [Singular Matrix]

$$\begin{bmatrix} 1 & -1 & 0 \\ -1 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \rightarrow \begin{bmatrix} 1 & -1 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Singular matrix produces a row of zeros

If **M** is singular, will not be able to solve for some unknowns.

  - Contrived Example

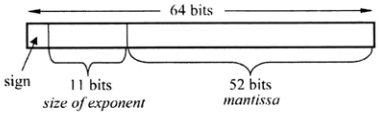
$$\begin{bmatrix} 10^{-17} & 1 \\ 1 & 2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 1 \\ 3 \end{bmatrix}$$

$$L = \begin{bmatrix} 1 & 0 \\ 10^{17} & 1 \end{bmatrix} \qquad U = \begin{bmatrix} 10^{-17} & 1 \\ 0 & \underbrace{2 - 1 \cdot 10^{17}}_{\text{Can we represent this?}} \end{bmatrix}$$

  - An Aside on Floating Point Arithmetic

Double Precision Number

$X.XXXX\ldots X \cdot 10^{\text{exponent}}$



How large a number can we represent with 52 bits? How many decimal digits is it?

3 bits	$\rightarrow$	$2^3 = 8$	≈ 10	$\rightarrow$	1 digit
7 bits	$\rightarrow$	$2^7 = 128$	≈ 100	$\rightarrow$	2 digits
10 bits	$\rightarrow$	$2^{10} = 1024$	≈ 1000	$\rightarrow$	3 digits

[Remember cubits $\leftrightarrow$ 3 decimal digits]

52 bits $\rightarrow$ $52 \text{ bits} \cdot \frac{3 \text{ digits}}{10 \text{ bits}} \approx 15 - 16$ decimal digits!

52 bits is the precision used in MATLAB, for example

Basic Problem

A.) $1.0000001 - 0.000000000000001 = 1.0$

B.) $\left(\pi \cdot 10^{-7}\right) + 1 - 1.0000001 = ?$

Look at order of procedure:

Ex #1.) $\quad -1.0000001 + 1 = -1 \times 10^{-7}$

$$-1\times10^{-7}+\pi\cdot10^{-7}=\underbrace{2.141592653589793}_{correct?}\cdot10^{-7}$$

All 15 decimal digits are correct in this case

Ex #2.)

$$1+\pi\cdot10^{-7}=\underbrace{1.000000314159265}_{\text{lost 7 digits of }\pi}$$

$$-1.0000001+1.000000314159265$$

$$=2.141592652\underbrace{105118}_{junk}\cdot10^{-7}$$

Only the first 8 decimal digits are correct in this case, the remainder are garbage. Lose digits of precision

Key Issue

Avoid additions and subtractions between large and small numbers!

EVEN BETTER: AVOID GENERATING LARGE NUMBERS AT ALL IF POSSIBLE!!!

- Back to the contrived Example

$$\begin{bmatrix}10^{-17} & 1\\ 1 & 2\end{bmatrix}\begin{bmatrix}x_1\\ x_2\end{bmatrix}=\begin{bmatrix}1\\ 3\end{bmatrix}$$

$$LU_{Exact}=\begin{bmatrix}1 & 0\\ 10^{17} & 1\end{bmatrix}\begin{bmatrix}10^{-17} & 1\\ 0 & 2-10^{17}\end{bmatrix}\begin{bmatrix}x_1\\ x_2\end{bmatrix}=\begin{bmatrix}1\\ 3\end{bmatrix}$$

$$LU_{Rounded}=\begin{bmatrix}1 & 0\\ 10^{17} & 1\end{bmatrix}\begin{bmatrix}10^{-17} & 1\\ 0 & -10^{17}\end{bmatrix}\begin{bmatrix}x_1\\ x_2\end{bmatrix}=\begin{bmatrix}1\\ 3\end{bmatrix}$$

$$\begin{bmatrix}x_1\\ x_2\end{bmatrix}_{Exact}=\begin{bmatrix}1\\ 1\end{bmatrix}\qquad\begin{bmatrix}x_1\\ x_2\end{bmatrix}_{Rounded}=\begin{bmatrix}0\\ 1\end{bmatrix}\times$$

Original Problem:

$$\begin{bmatrix}10^{-17} & 1\\ 1 & 2\end{bmatrix}\begin{bmatrix}x_1\\ x_2\end{bmatrix}=\begin{bmatrix}1\\ 3\end{bmatrix}$$

Swap the rows

$$\begin{bmatrix}1 & 2\\ 10^{-17} & 1\end{bmatrix}$$

Partial Pivoting for Roundoff Reduction

If $|M_{ii}|<\max_{j>i}|M_{ji}|$ swap row i with $\arg\left(\max_{j>i}|M_{ij}|\right)$

$$LU_{reordered}=\begin{bmatrix}1 & 0\\ 10^{-17} & 1\end{bmatrix}\begin{bmatrix}1 & 2\\ 0 & \underbrace{1-2\cdot10^{-17}}\end{bmatrix}$$

This multiplier is small

This term still gets rounded, but this time the multiplier does not 'overpower' the row.

$$L=\begin{bmatrix}1 & 0\\ 10^{-17} & 1\end{bmatrix}\quad U=\begin{bmatrix}1 & 2\\ 0 & 1\end{bmatrix}$$

Let's solve $LUx=b$.

First, solve $Ly=b$:

$$Ly=b\rightarrow\begin{bmatrix}1 & 0\\ 10^{-17} & 1\end{bmatrix}\begin{bmatrix}y_1\\ y_2\end{bmatrix}=\begin{bmatrix}1\\ 3\end{bmatrix}$$

$$\Rightarrow y=\begin{bmatrix}y_1\\ y_2\end{bmatrix}=\begin{bmatrix}1\\ 3-10^{-17}\end{bmatrix}\approx\begin{bmatrix}1\\ 3\end{bmatrix}$$

Next, solve $Ux=y$:

$$Ux=y\rightarrow\begin{bmatrix}1 & 2\\ 0 & 1\end{bmatrix}\begin{bmatrix}x_1\\ x_2\end{bmatrix}=\begin{bmatrix}1\\ 3\end{bmatrix}\Rightarrow x=\begin{bmatrix}x_1\\ x_2\end{bmatrix}=\begin{bmatrix}-5\\ 3\end{bmatrix}\times$$ STILL WRONG

PROBLEM: Need to swap the elements of b as well!

Solve $Ly=b$:

$$Ly=b\rightarrow\begin{bmatrix}1 & 0\\ 10^{-17} & 1\end{bmatrix}\begin{bmatrix}y_1\\ y_2\end{bmatrix}=\begin{bmatrix}3\\ 1\end{bmatrix}$$

$$\Rightarrow y=\begin{bmatrix}y_1\\ y_2\end{bmatrix}=\begin{bmatrix}3\\ 1-3\cdot10^{-17}\end{bmatrix}\approx\begin{bmatrix}3\\ 1\end{bmatrix}$$

Next, solve $Ux=y$:

$$Ux=y\rightarrow\begin{bmatrix}1 & 2\\ 0 & 1\end{bmatrix}\begin{bmatrix}x_1\\ x_2\end{bmatrix}=\begin{bmatrix}3\\ 1\end{bmatrix}\Rightarrow x=\begin{bmatrix}x_1\\ x_2\end{bmatrix}=\begin{bmatrix}1\\ 1\end{bmatrix}$$ Correct Answer.

- Example.

$$\begin{bmatrix} 1 & 0 & 0 & \cdots & 0 & 1 \\ 10 & 1 & 0 & \cdots & 0 & 0 \\ 0 & 10 & 1 & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \ddots & & \\ 0 & 0 & \cdots & 10 & 1 & 0 \\ 0 & 0 & \cdots & 0 & 10 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_N \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ \vdots \\ b_n \end{bmatrix}$$

$$\Rightarrow \begin{bmatrix} 1 & 0 & 0 & \cdots & 1 \\ 10 & 1 & 0 & & -10 \\ & 10 & 1 & & 0 \\ & & \ddots & \ddots & \vdots \\ & & & 10 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_N \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ \vdots \\ b_n \end{bmatrix}$$

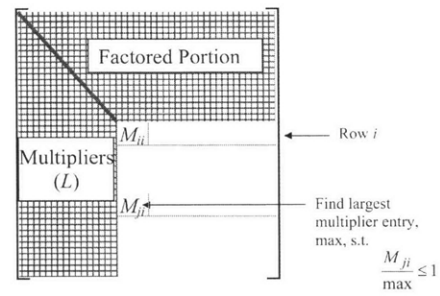
$$\Rightarrow \begin{bmatrix} 1 & 0 & 0 & \cdots & 1 \\ 10 & 1 & 0 & & -10 \\ 0 & 10 & 1 & & -100 \\ & & \ddots & \ddots & \vdots \\ & & & 10 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_N \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ \vdots \\ b_n \end{bmatrix}$$

$$\Rightarrow \cdots \Rightarrow \begin{bmatrix} 1 & 0 & 0 & \cdots & 1 \\ 10 & 1 & 0 & & -10 \\ 0 & 10 & 1 & & -100 \\ & & \ddots & \ddots & \vdots \\ & & & 10 & -10^{N-1} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_N \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ \vdots \\ b_n \end{bmatrix}$$

The last entry is very large! Use partial pivoting – swap first and second rows…

- If the matrix is strictly diagonally dominant
- or if use partial pivoting for round-off reduction:
  1. The multipliers will <u>always</u> be smaller than one in magnitude.
  2. The maximum magnitude entry in the LU factors will never be larger than $2^{(n-1)}$ times the maximum magnitude entry in the original matrix.

For 1000 nodes, what is 2^{1000}?
Know that $2^{10} \approx 10^3$.
So, $2^{1000} = (2^{10})^{100} = 10^{300}$ → Very large number!
Might not be a very useful theorem → generally not this large.



3.5 Linear Systems - Conditioning

INTRODUCTION TO NUMERICAL SIMULATION

LECTURE 5.
Linear Systems – Conditioning

TODAY'S OUTLINE:

- Solution of Dense Linear Systems
  - Hard to Solve Problems
  - Perturbation Analysis and Conditioning
- Solution of Sparse Linear Systems
  - LU Factorization Reminder.
  - Example of Problems with Sparse Matrices
    - Struts and joints, resistor grids, 3-D heat flow
  - Tridiagonal Matrix Factorization
  - General Sparse Factorization
    - Fill-in and Reordering
    - Graph Based Approach
  - Sparse Matrix Data Structures
    - Scattering

SOLUTION OF DENSE LINEAR SYSTEMS

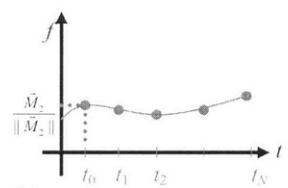
Hard to Solve Problems

❖ Fitting example

Polynomial Interpolation

Table of Data

t_0	$f(t_0)$
t_1	$f(t_1)$
$\vdots$	$\vdots$
t_N	$f(t_N)$



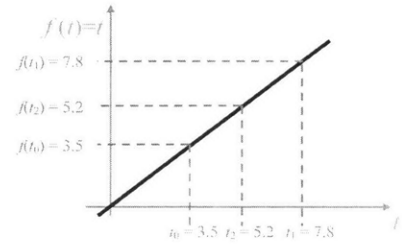
Problem: fit data with an Nth order polynomial

$$f(t) = \alpha_0 + \alpha_1 t + \alpha_2 t^2 + \alpha_3 t^3 + \ldots + \alpha_N t^N$$

Matrix Form

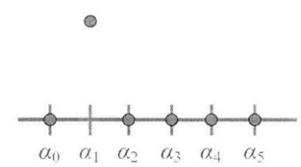
$$\underbrace{\begin{bmatrix} 1 & t_0 & t_0^2 & \cdots & t_0^N \\ 1 & t_1 & t_1^2 & \cdots & t_1^N \\ & & & & \\ 1 & t_N & t_N^2 & \cdots & t_N^N \end{bmatrix}}_{M_{\text{interp}}} \begin{bmatrix} \alpha_0 \\ \alpha_1 \\ \\ \alpha_N \end{bmatrix} = \begin{bmatrix} f(t_0) \\ f(t_1) \\ \\ f(t_N) \end{bmatrix}$$

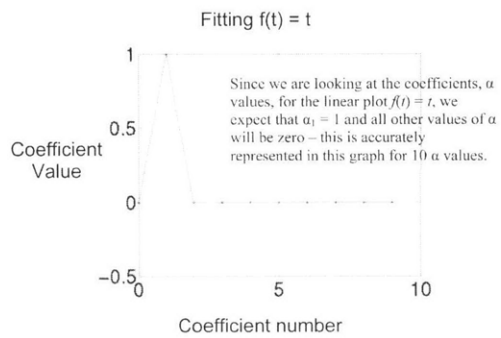
$$\alpha_0 + \alpha_1 t_0 + \alpha_2 t_0^2 + \cdots + \alpha_N t_0^N = f(t_0)$$
$$\alpha_0 + \alpha_1 t_1 + \alpha_2 t_1^2 + \cdots + \alpha_N t_1^N = f(t_1)$$
$$\vdots$$
$$\alpha_0 + \alpha_1 t_N + \alpha_2 t_N^2 + \cdots + \alpha_N t_N^N = f(t_N)$$

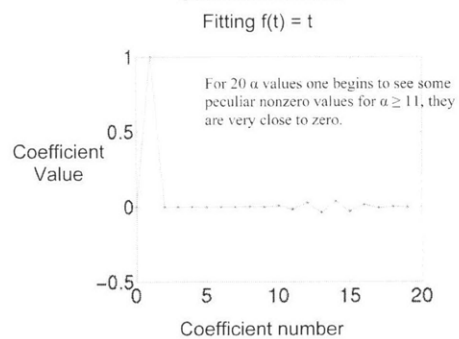


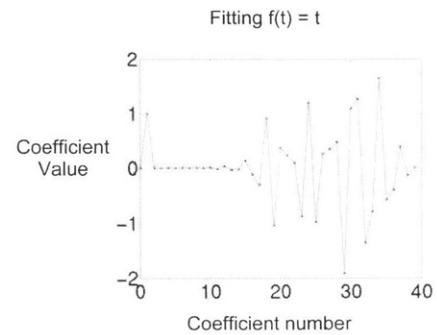
$$\alpha_0 + \alpha_1 t + \alpha_2 t^2 + \cdots = f(t) = t$$

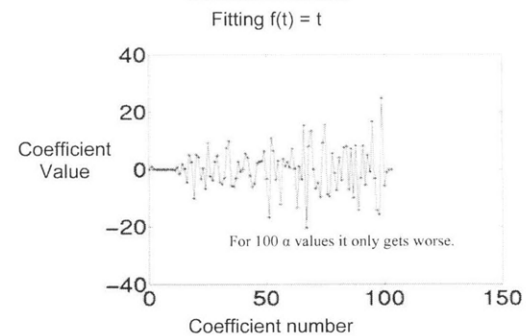
$\alpha_0 = 0$
$\alpha_1 = 1$
$\alpha_2 = 0$
$\vdots$
$\alpha_N = 0$

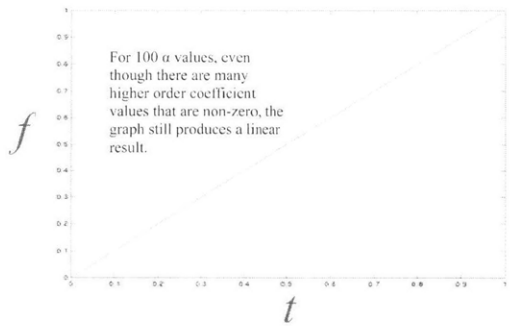



Fitting f(t) = t
Coefficient Value
Coefficient number
Since we are looking at the coefficients, α values, for the linear plot f(t) = t, we expect that α1 = 1 and all other values of α will be zero – this is accurately represented in this graph for 10 α values.


Fitting f(t) = t
Coefficient Value
Coefficient number
For 20 α values one begins to see some peculiar nonzero values for α ≥ 11, they are very close to zero.

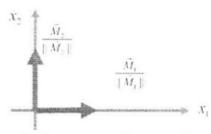

Fitting f(t) = t
Coefficient Value
Coefficient number


Fitting f(t) = t
Coefficient Value
Coefficient number
For 100 α values it only gets worse.

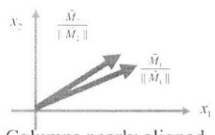


❖ Geometric Approach is clearer

$M = [\vec{M}_1, \vec{M}_2]$ Solving, $\mathbf{M}x = b$ is finding $x_1\vec{M}_1 + x_2\vec{M}_2 = \vec{b}$



Columns orthogonal



Columns nearly aligned

When vectors are nearly aligned, difficult to determine how much of $\vec{M}_1$ versus how much of $\vec{M}_2$

$$M = \begin{bmatrix} 1 & 0 \\ 0 & 10^{-6} \end{bmatrix}$$

orthogonal columns

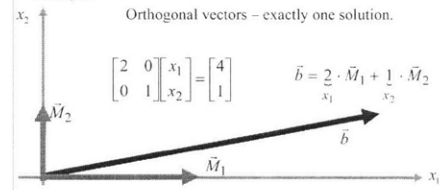
$$M = \begin{bmatrix} 1 & 1-10^{-6} \\ 1-10^{-6} & 1 \end{bmatrix}$$

columns nearly aligned

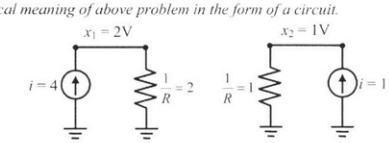
Example.

Orthogonal vectors – exactly one solution.

$$\begin{bmatrix} 2 & 0 \\ 0 & 1 \end{bmatrix}\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 4 \\ 1 \end{bmatrix} \qquad \vec{b} = \underset{x_1}{\underline{2}} \cdot \vec{M}_1 + \underset{x_2}{\underline{1}} \cdot \vec{M}_2$$

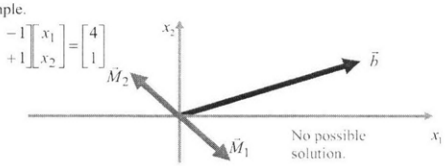


Physical meaning of above problem in the form of a circuit.

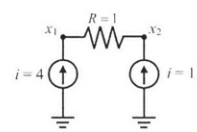


Example.

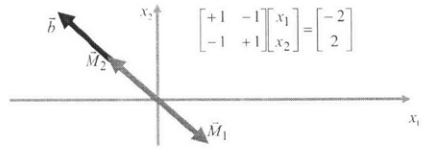
$$\begin{bmatrix} +1 & -1 \\ -1 & +1 \end{bmatrix}\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 4 \\ 1 \end{bmatrix}$$



Physical meaning of above problem in the form of a circuit.

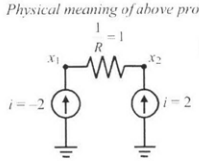


Example.



$$\begin{bmatrix} +1 & -1 \\ -1 & +1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} -2 \\ 2 \end{bmatrix}$$

Physical meaning of above problem in the form of a circuit.

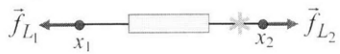


Solution(s):

$x_1 = -1$	$x_2 = 1$	$-1 \cdot \vec{M}_1 + 1 \cdot \vec{M}_2$
$x_1 = -2$	$x_2 = 0$	$-2 \cdot \vec{M}_1$
$x_1 = 0$	$x_2 = 2$	$2 \cdot \vec{M}_2$
$\vdots$	$\vdots$	$\vdots$

Infinite number of solutions

Example.



Node 1: $f_{L_1} - \varepsilon(u_{x_1} - u_{x_2}) = 0$

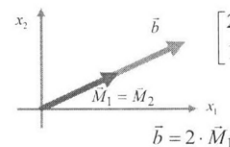
Node 2: $f_{L_2} + \varepsilon(u_{x_1} - u_{x_2}) = 0$

$$\begin{bmatrix} -\varepsilon & \varepsilon \\ \varepsilon & -\varepsilon \end{bmatrix} \begin{bmatrix} u_{x_1} \\ u_{x_2} \end{bmatrix} = \begin{bmatrix} -f_{L_1} \\ -f_{L_2} \end{bmatrix}$$

☞ either infinite number of solutions or none

Same problem as circuit with two current sources (above)

Linearly dependent vectors.

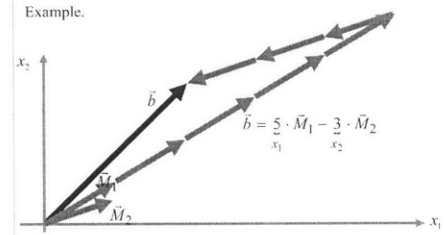


$$\begin{bmatrix} 2 & 2 \\ 1 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 4 \\ 1 \end{bmatrix}$$

Infinite number of solutions

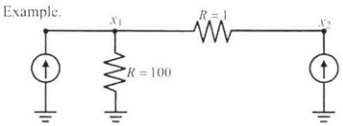
$$\vec{b} = 2 \cdot \vec{M}_1 = 2 \cdot \vec{M}_2 = \vec{M}_1 + \vec{M}_2 = etc.$$

Example.

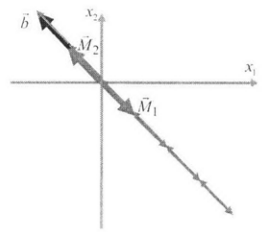


Columns are very close to being linearly dependent (closely aligned) – have a larger condition number / columns are not close to orthogonal

Example.



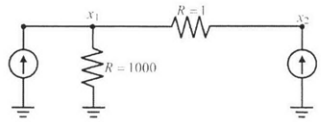
$$\begin{array}{cc} \begin{bmatrix} 1 & -1 \\ -1 & 1.01 \end{bmatrix} \\ \vec{M}_1 \quad \vec{M}_2 \end{array}$$



$$\vec{b} = \begin{bmatrix} -2 \\ 2.06 \end{bmatrix} = 4 \cdot \vec{M}_1 + 6 \cdot \vec{M}_2$$

Example.

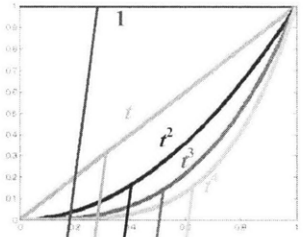
Same circuit as above, with the resistor value changed from 100 to 1000.



$$\begin{array}{cc} \begin{bmatrix} 1 & -1 \\ -1 & 1.001 \end{bmatrix} \\ \vec{M}_1 \quad \vec{M}_2 \end{array}$$

$$\vec{b} = \begin{bmatrix} -2 \\ 2.06 \end{bmatrix} = 58 \cdot \vec{M}_1 + 60 \cdot \vec{M}_2$$

- Polynomial Interpolation



The power series polynomials are nearly linearly dependent. (i.e. if viewed as vectors they are almost aligned)

$$\begin{bmatrix} 1 & t_1 & t_1^2 & t_1^3 & \cdots \\ 1 & t_2 & t_2^2 & t_2^3 & \cdots \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & t_N & t_N^2 & t_N^3 & \cdots \end{bmatrix}$$

Perturbation Analysis and Conditioning

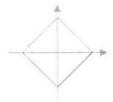
- Induced Norms
  - Matrix Magnification Question

    Suppose $y = \mathbf{M}x$

    How much larger is y than x? OR How much does $\mathbf{M}$ magnify x?

- Vector Norm Review
  - L_1 norm:

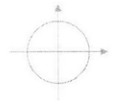
$$\|x\|_1 = \sum_{i=1}^{n} |x_i|$$



$$\|x\|_1 < 1$$

  - L_2 (Euclidean) norm:

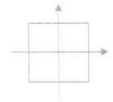
$$\|x\|_2 = \sqrt{\sum_{i=1}^{n} |x_i|^2}$$



$$\|x\|_2 < 1$$

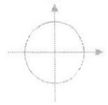
  - L_∞ norm:

$$\|x\|_\infty = \max_i |x_i|$$



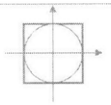
$$\|x\|_\infty < 1$$

If $\|x\|_2 < 1$

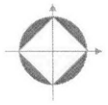


?

$\|x\|_\infty < 1$ yes



$\|x\|_1 < 1$ no



Try $x = \begin{bmatrix} \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{bmatrix}$

$$\|x\|_2 = \sqrt{\left(\tfrac{1}{\sqrt{2}}\right)^2 + \left(\tfrac{1}{\sqrt{2}}\right)^2} = 1$$

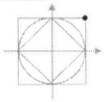
$$\|x\|_1 = \tfrac{1}{\sqrt{2}} + \tfrac{1}{\sqrt{2}} = \sqrt{2} > 1$$

Which one?

$$\|x\|_1 \le \|x\|_2 \le \|x\|_\infty$$

$$\boxed{\|x\|_\infty \le \|x\|_2 \le \|x\|_1} \ *$$

Example.) Try $x = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$



$$\|x\|_1 = 1 + 1 = 2$$

$$\|x\|_2 = \sqrt{1^2 + 1^2} = \sqrt{2}$$

$$\|x\|_\infty = \max(1,1) = 1$$

Is there an x such that $\|x\|_\infty = \|x\|_2 = \|x\|_1$?

$$x = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \Rightarrow \|x\|_\infty = \|x\|_2 = \|x\|_1 = 1$$

Example.) Two heat conducting bars.

$$x = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_N \end{bmatrix}$$

Vector of temperatures

Given:

Bar A: $\|x\|_\infty < 100^\circ$

Bar B: $\|x\|_1 < 100^\circ$

Which one do you touch?

Bar B will likely be cooler to the touch since all nodes on bar B will sum to 100°, whereas on bar A, the maximum node value is 100°.

Worst-case-scenario, bar A will be 100° everywhere and bar B will be 100° on one node and 0° elsewhere

- Matrix Norms [Standard Induced l-norms]
  - Definition:

$$\|M\|_l = \max_x \frac{\|Mx\|_l}{\|x\|_l} = \max_{\|x\|_l = 1} \|Mx\|_l$$

- $\|M\|_1 = \max_i \sum_{j=1}^{N} |M_{ij}| =$ max abs column sum
- $\|M\|_\infty = \max_j \sum_{i=1}^{N} |M_{ij}| =$ max abs row sum
- $\|M\|_2 = \max_i \sqrt{\lambda_i\left(M^H M\right)} = \max_i \lambda_i(M) =$ for symmetric, real matrices = max eigenvalue

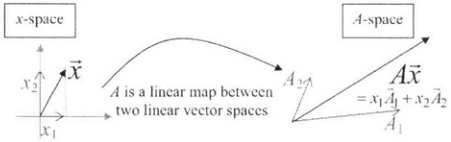
$\|\mathbf{M}\|_1 = \max_j \sum_{i=1}^{n} |M_{ij}| =$ max abs column sum

$$\begin{bmatrix} & & \\ & \mathbf{M} & \\ & & \end{bmatrix}\begin{bmatrix}1\\0\\\vdots\\0\end{bmatrix} = \begin{bmatrix}m_{11}\\m_{21}\\\vdots\\m_{n1}\end{bmatrix} \qquad \begin{bmatrix} & & \\ & \mathbf{M} & \\ & & \end{bmatrix}\begin{bmatrix}0\\1\\0\\\vdots\\0\end{bmatrix} = \begin{bmatrix}m_{12}\\m_{22}\\\vdots\\m_{n2}\end{bmatrix}$$

Which unit vector input will produce the largest output?

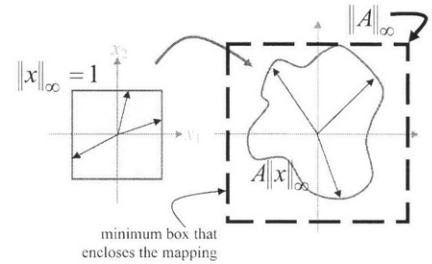
$\|\mathbf{M}\|_\infty = \max_i \sum_{j=1}^{n} |M_{ij}| =$ max abs row sum

$$\begin{bmatrix} & & \\ & \mathbf{M} & \\ & & \end{bmatrix}\begin{bmatrix}\pm 1\\\pm 1\\\vdots\\\pm 1\end{bmatrix} = \begin{bmatrix}\sum_{i=1}^{n} \pm m_{1i}\\\sum_{i=1}^{n} \pm m_{2i}\\\vdots\\\sum_{i=1}^{n} \pm m_{ni}\end{bmatrix}$$



$$\begin{bmatrix}\vec{A}_1 & \vec{A}_2\end{bmatrix}\begin{bmatrix}x_1\\x_2\end{bmatrix} = \begin{bmatrix}\vec{b}\end{bmatrix}$$

$\|A\|$ is a measure of the largest gain or magnification power



Example.) Given the following A, find $\|A\|_\infty$

$$A = \begin{bmatrix}2 & 7 & -4\\5 & -2 & 3\\1 & -2 & 4\end{bmatrix}$$

$$\|A\|_\infty = \max\begin{pmatrix}2+7+4\\5+2+3\\1+2+4\end{pmatrix} = \max\{13,10,7\} = 13$$

- Hard to Solve Systems
  - Perturbation Analysis

$$\frac{\|\delta x\|}{\|x+\delta x\|} \xrightarrow{?} \frac{\|\delta M\|}{\|M\|} \qquad \begin{array}{ccccc} [M+\delta M] & [x+\delta x] & = & b \\ M & x & = & b \end{array}$$

Perturbation Equation

$$\underbrace{(M+\delta M)}_{\substack{\text{models LU}\\ \text{roundoff}}} \underbrace{(x+\delta x)}_{\substack{\text{models solution}\\ \text{perturbation}}} = Mx + \delta Mx + M\delta x + \delta M\delta x = \underbrace{\bar{b}}_{\substack{\text{unperterbed}\\ \text{right hand side}}}$$

Since $\mathbf{M}x - b = 0$

$$M\delta x = -\delta M(x+\delta x) \Rightarrow \delta x = -M^{-1}\delta M(x+\delta x)$$

Taking Norms

$$\|\delta x\| \le \|M^{-1}\|\|M\|\frac{1}{\|M\|}\|\delta M\|\|x+\delta x\|$$

Relative Error Relation

$$\frac{\|\delta x\|}{\|x+\delta x\|} \le \underbrace{\|M^{-1}\|\|M\|}_{\text{"Condition Number"}}\frac{\|\delta M\|}{\|M\|}$$

$\mathbf{M}\vec{x} = \vec{b}$

Assume $\frac{\|\delta M\|}{\|M\|} = 10^{-16}$ What is $\frac{\|\delta x\|}{\|x\|} = \frac{\|\delta x\|}{\|x+\delta x\|} = ?$

e.g. $\|M^{-1}\|\|M\| = 100$ $\qquad \frac{\|\delta x\|}{\|x+\delta x\|} \le 100 \cdot 10^{-16} = 10^{-14}$

e.g. $\|M^{-1}\|\|M\| = 10^{13}$

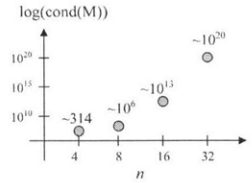
$$\frac{\|\delta x\|}{\|x+\delta x\|} \le 10^{13} \cdot 10^{-16} = 10^{-3} = 0.1\% \text{ error}$$

e.g. $\|M^{-1}\|\|M\| = 10^{20}$

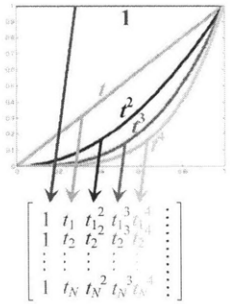
$$\frac{\|\delta x\|}{\|x+\delta x\|} \le 10^{20} \cdot 10^{-16} = 10^{4} = 10^{6}\% \text{ error!!}$$

Large condition number → Big error

- Geometric Analysis / Polynomial Interpolation



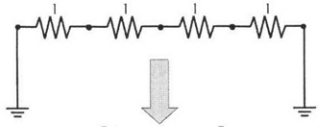
The power series polynomials are nearly linearly dependent.



$$\begin{bmatrix} 1 & t_1 & t_1^2 & t_1^3 & t_1^4 & \vdots \\ 1 & t_2 & t_2^2 & t_2^3 & t_2^4 & \vdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & t_N & t_N^2 & t_N^3 & t_N^4 & \vdots \end{bmatrix}$$

SOLUTION OF SPARSE LINEAR SYSTEMS

Examples of Problems with Sparse Matrices



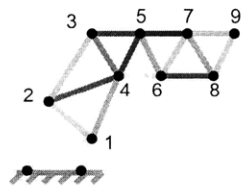
$$\begin{bmatrix} 2 & -1 & & \\ -1 & 2 & -1 & \\ & -1 & 2 & \ddots \\ & & \ddots & \ddots \end{bmatrix}$$

Nodal Equation Matrix **M**

number of nonzeros $\approx 3N - 2 = O(N)$

$\frac{3N}{N^2} = \frac{3000}{10^6} = 0.003 = 0.3\%$ of the entries are nonzero for a 1000×1000 matrix

❖ Space Frame

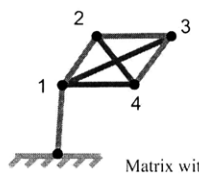


Nodal Matrix

$$\begin{bmatrix} X & & & X & & & & & \\ & X & X & X & & & & & \\ & X & X & & X & & & & \\ X & X & X & X & X & & & & \\ & & X & X & X & X & X & & \\ & & & & X & X & X & & \\ & & & & X & X & X & X & X \\ & & & & & & X & X & X \\ & & & & & & & X & X \end{bmatrix}$$

Unknowns : Joint positions
Equations : $\sum$ forces = 0

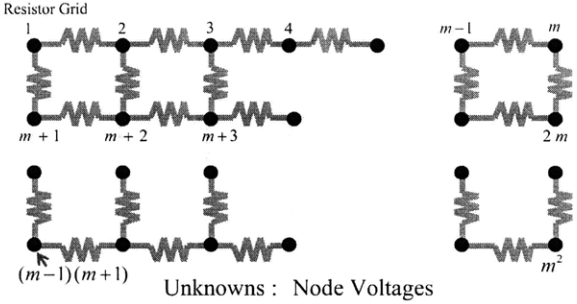
$$X = \underbrace{\begin{bmatrix} \bullet & \bullet \\ \bullet & \bullet \end{bmatrix}}_{2\times 2 \text{ block}}$$



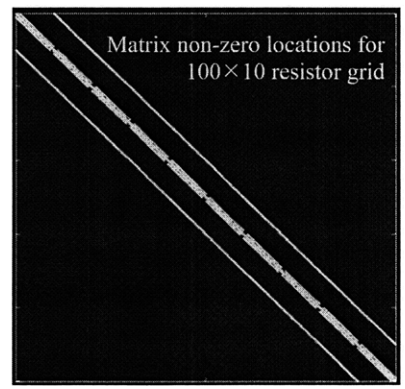
$$\begin{bmatrix} X & X & X & X \\ X & X & X & X \\ X & X & X & X \\ X & X & X & X \end{bmatrix}$$

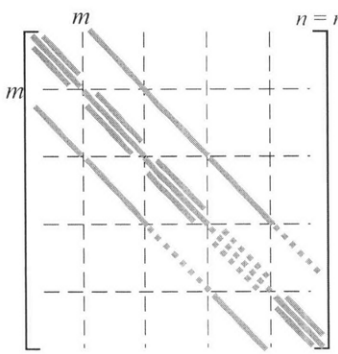
Matrix with all non-zero entries.
Struts connect all every joint in the frame.

❖ Resistor Grid



Unknowns : Node Voltages
Equations : $\sum$ currents = 0





$m^2 \times m^2$ matrix

Nonzeros: max 5 entries/row

$m = 100$
$n = 10^4$

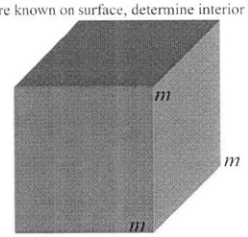
$$\frac{\#\text{ non-zeros}}{\#\text{ of entries}} = \frac{5n}{n^2}$$

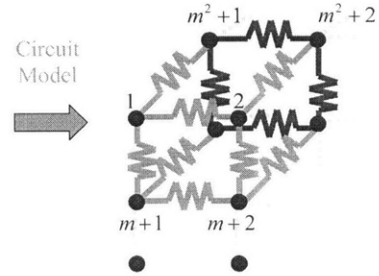
$$= \frac{5 \cdot 10^4}{10^8} = 5 \cdot 10^{-4}$$

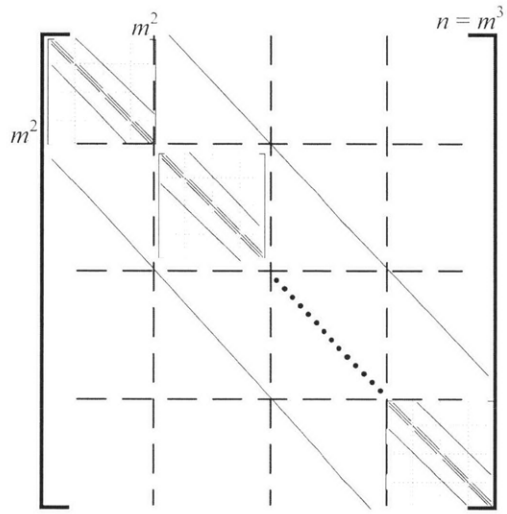
$$= 0.05\%$$

Largest entry = 4 (because there are at most 4 resistors at one node)
Smallest entry = 2 (there are at least 2 resistors at each node)

- Temperature in a cube
  Temperature known on surface, determine interior temperature







$m = 100$

max 7 entries/row

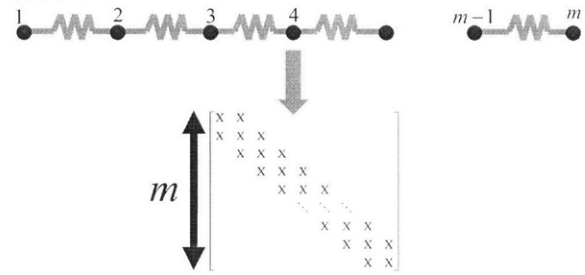
$n = m^3 = 10^6$

$$\frac{\#\text{ non-zeros}}{\#\text{ of entries}} = \frac{7n}{n^2} = \frac{7 \cdot 10^6}{10^{12}} = 7 \cdot 10^{-6} = 0.0007\%$$

Largest entry = 6
Smallest entry = 3

Tridiagonal Matrix Factorization

- Matrix Form



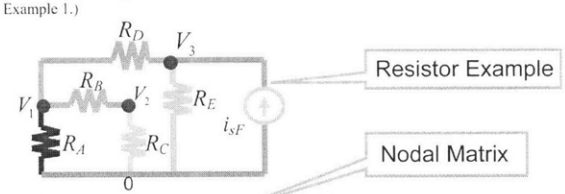
- GE Algorithm for tridiagonal matrix

```
For i = 1 to n-1 {                    "For each Row"
  For j = i+1 to n {                  "For each target Row below the source"
      M_ji = M_ji / M_ii              Pivot
      For k = i+1 to n { "For each Row element beyond Pivot"
          M_jk ← M_jk − M_ji M_ik
          Multiplier
      }
  }
}
```

Order N Operations!

General Sparse Factorization

- Fill-in and Reordering
  - Example 1.)



$$\begin{bmatrix} \frac{1}{R_A}+\frac{1}{R_B}+\frac{1}{R_D} & -\frac{1}{R_B} & -\frac{1}{R_D} \\ -\frac{1}{R_B} & \frac{1}{R_B}+\frac{1}{R_C} & \\ -\frac{1}{R_D} & & \frac{1}{R_D}+\frac{1}{R_E} \end{bmatrix} \begin{bmatrix} V_1 \\ V_2 \\ V_3 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ i_{sF} \end{bmatrix}$$

Symmetric Diagonally Dominant

Matrix Non Zero Structure

$$\begin{bmatrix} X & X & X \\ X & X & 0 \\ X & 0 & X \end{bmatrix}$$

X= Non zero

Matrix after one LU step

$$\begin{bmatrix} X & X & X \\ X & X & X \\ X & X & X \end{bmatrix}$$

- Example 2.)

Fill-ins Propagate

$$\begin{bmatrix} X & X & X & X \\ X & X & 0 & 0 \\ 0 & X & X & 0 \\ 0 & X & 0 & 0 \end{bmatrix} \qquad \begin{bmatrix} X & X & X & X \\ X & X & X & X \\ 0 & X & X & X \\ 0 & X & X & X \end{bmatrix}$$

Fill-ins from Step 1 result in Fill-ins in step 2

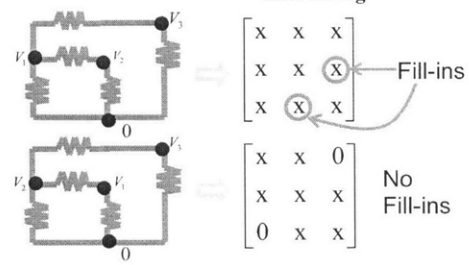
Use to 64 bits ~ 8 bytes represent each coefficient

Store all numbers (even zeros)…

1000×1000	→	8 MB	10 min
$10{,}000 \times 10{,}000$	→	800 MB	10,000 min
$100{,}000 \times 100{,}000$	→	80 GB	"forever"

- Reordering

Reordering



$$\begin{bmatrix} x & x & x \\ x & x & x \\ x & x & x \end{bmatrix} \qquad \begin{bmatrix} x & x & 0 \\ x & x & x \\ 0 & x & x \end{bmatrix}$$

Node Reordering Can Reduce Fill-In

- Preserves Properties (symmetry, diagonal dominance)
- Equivalent to swapping rows and columns

- Numeric Example

$$\begin{array}{c} \\ KCL_1 \\ KCL_2 \\ KCL_3 \end{array} \begin{array}{c} \begin{array}{ccc} V_1 & V_2 & V_3 \end{array} \\ \begin{bmatrix} 2 & -1 & -1 \\ -1 & 3 & 0 \\ -1 & 0 & 1 \end{bmatrix} \end{array}$$

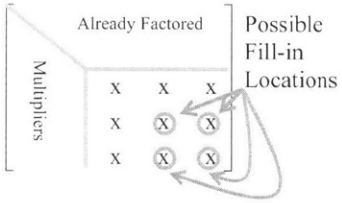
→ Swap KCL_1 and KCL_2

$$\begin{array}{c} \\ KCL_2 \\ KCL_1 \\ KCL_3 \end{array} \begin{array}{c} \begin{array}{ccc} V_1 & V_2 & V_3 \end{array} \\ \begin{bmatrix} -1 & 3 & 0 \\ 2 & -1 & -1 \\ -1 & 0 & 1 \end{bmatrix} \end{array}$$

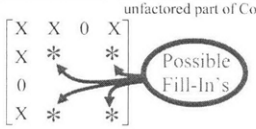
→ Swap variable order V_1 and V_2

$$\begin{array}{c} \\ KCL_2 \\ KCL_1 \\ KCL_3 \end{array} \begin{array}{c} \begin{array}{ccc} V_2 & V_1 & V_3 \end{array} \\ \begin{bmatrix} 3 & -1 & 0 \\ -1 & 2 & -1 \\ 0 & -1 & 1 \end{bmatrix} \end{array}$$

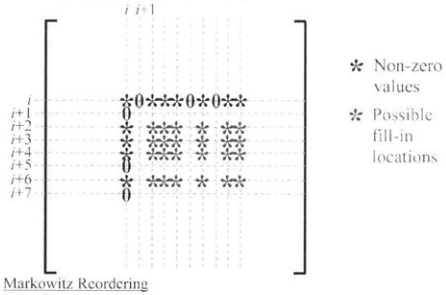
Where can fill-in occur?



Fill-in Estimate = (Non zeros in unfactored part of Row − 1) · (Non zeros in unfactored part of Col − 1) = Markowitz product



Fill-ins propagate down and to the right



Markowitz Reordering

For $i = 1$ to n

Find diagonal $j \geq i$ with min Markowitz Product

Swap rows $j \neq i$ and columns $j \neq i$

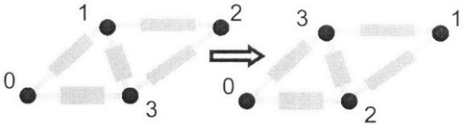
Factor the new row i and determine fill-ins

End

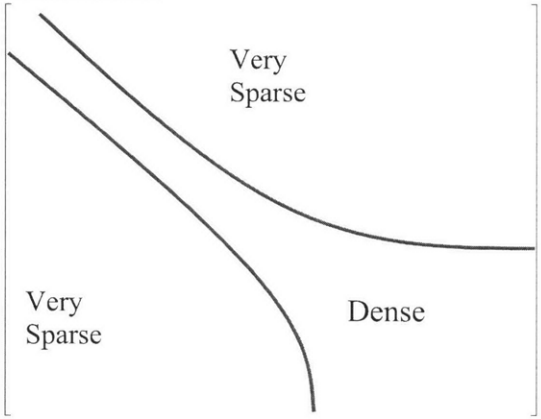
Greedy Algorithm!

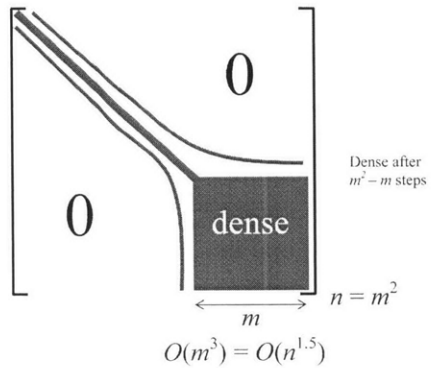
Why only try diagonals?

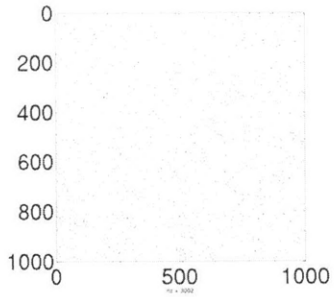
- Corresponds to node reordering in Nodal formulation



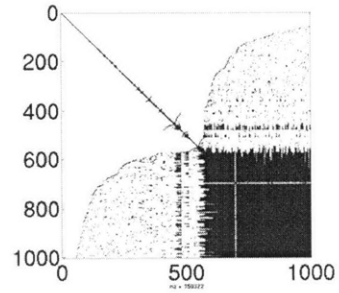
- Reduces search cost
- Preserves Matrix Properties
  - Diagonal Dominance
  - Symmetry
- Pattern of a Filled-in Matrix







Unfactored Random Matrix



Factored Random Matrix

3.6 Linear Systems - LU for Sparse Systems

INTRODUCTION TO NUMERICAL SIMULATION

LECTURE 6.A.
Solution of Sparse Linear Systems

TODAY'S OUTLINE:

- ❖ Solution of Sparse Linear Systems
  - – General Sparse Factorization
    - • Graph Based Approach
    - • Sparse Matrix Data Structures
      - ▪ Scattering

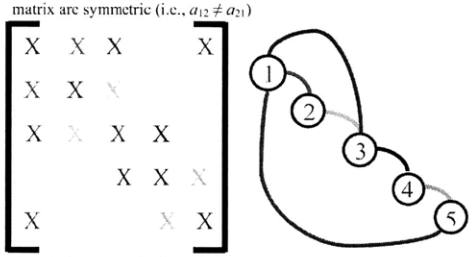
SOLUTION OF SPARSE LINEAR SYSTEMS

General Sparse Factorization

- ❖ Graph Based Approach
  - ○ Construction

Structurally Symmetric Matrices and Graphs

Note that *structurally* symmetric does not imply that the values of the matrix are symmetric (i.e., $a_{12} \neq a_{21}$)



- ▪ One Node Per Matrix Row
- ▪ One Edge Per Off-Diagonal Pair

Can one apply these graph-based techniques to the following matrix?

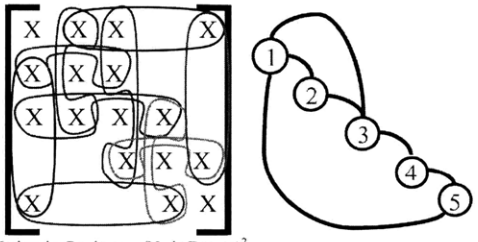
$$\begin{bmatrix} 1 & 3 & 0 & 0 \\ 0 & 1 & 2 & 0 \\ 0 & 2 & 1 & 5 \\ 0 & 0 & 5 & 1 \end{bmatrix}$$

This matrix is not "structurally symmetric" in that there is a zero in a_{21} and a non-zero value in a_{12}.

Can still use this approach, just treat it as if there is a nonzero in the a_{21} place and use the graphs to do the analysis; there will be some efficiency loss, but the methods will still work.

Thus, this technique may be applied to "mildly" structurally symmetric matrices by assuming there is structural symmetry with some loss of efficiency in treating some of the zeros as if they are non-zeros.

- ○ Markowitz Products

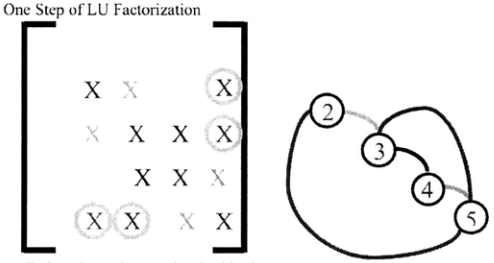


Markowitz Products = (Node Degree)^2

M_{11} ⇨ $3 \times 3 = 9$ (degree 1)$^2 = 3^2 = 9$
M_{22} ⇨ $2 \times 2 = 4$ (degree 2)$^2 = 2^2 = 4$
M_{33} ⇨ $3 \times 3 = 9$ (degree 3)$^2 = 3^2 = 9$
M_{44} ⇨ $2 \times 2 = 4$ (degree 4)$^2 = 2^2 = 4$
M_{55} ⇨ $2 \times 2 = 4$ (degree 5)$^2 = 2^2 = 4$

- ○ Factorization

One Step of LU Factorization

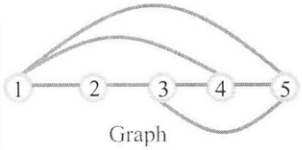


- ▪ Delete the node associated with pivot row
- ▪ "Tie together" the graph edges

When node 1 is removed, in the matrix, the non-zero entries in the first row create fill-ins that connect up the other nodes that can be seen in the graph.

- Example

$$\begin{bmatrix} X & X & & X & X \\ X & X & X & & \\ & X & X & X & X \\ X & & X & X & X \\ X & & X & X & X \end{bmatrix}$$



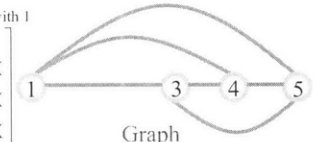
Graph

Markowitz Products
(= node degree)

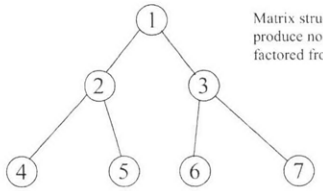
	Col	Row		
(1)	3	3	=	9
(2)	2	2	=	4
(3)	3	3	=	9
(4)	3	3	=	9
(5)	3	3	=	9

Swap (row & column) 2 with 1

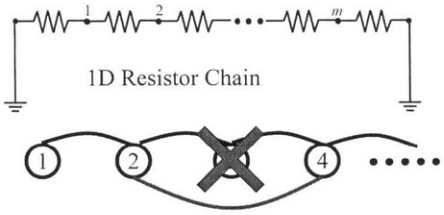
$$\begin{bmatrix} X & X & X & & \\ X & X & X & X & X \\ X & X & X & X & X \\ & X & X & X & X \\ & X & X & X & X \end{bmatrix}$$



Graph



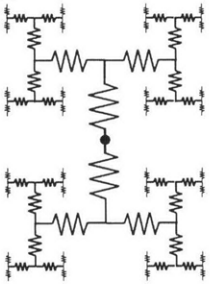
Matrix structure that will produce no fill-ins if factored from the base up



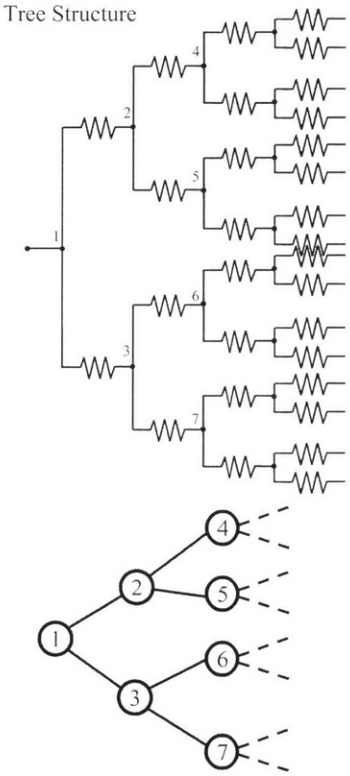
$$\begin{bmatrix} * & * & & & \\ * & * & * & & \\ & * & * & * & \\ & & * & * & \ddots \\ & & & \ddots & \ddots \end{bmatrix}$$

Nodal Equation Matrix **M**

Clock Tree

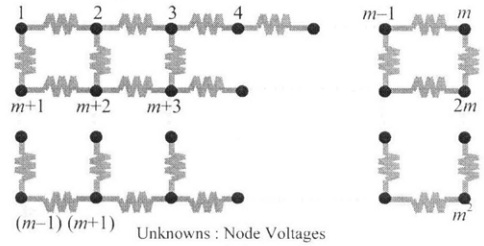


Tree Structure



5

- Resistor Grid Example

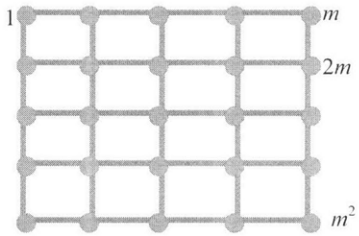


Unknowns : Node Voltages

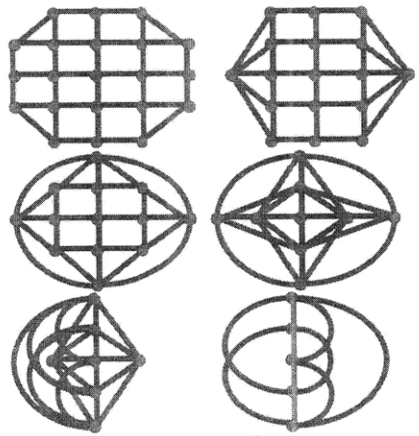
Equations : Σ currents = 0

- Grid Example

How long does it take to factor an $m \times m$ grid?



6



Suppose the center column is eliminated last

Factoring an $m \times m$ grid.

Dense LU: cost is $O(m^2)^3 = O(m^6)$

Sparse LU: each row has approximately 5 nonzeros
matrix has approximately $5m^2$ nonzeros
cost is at least $O(m^3)$

Separator is of size $2m$

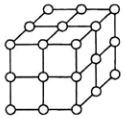
Factoring: cost is $O(2m)^3 = O(8m^3)$ {dense matrix}

Subgraph: size is $m/2$

Create 4 separators: size is $2 \times m/2 = m$
Cost is $O(m^3)$ per separator → $O(4m^3)$

As one continues to subdivide, the cost keeps halving.
Adding together these costs, the final cost will be roughly $O(m^3)$

3-D Example.



$3 \times 3 \times 3$ grid → 27 nodes

in general: $m \times m \times m$ grid → m^3 nodes

What does the separator look like for this grid?

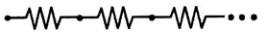
$m \times m$ nodes

m^2 nodes

Factoring cost = $O(m^6)$

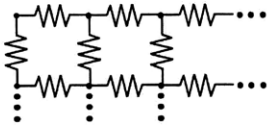
Structure & Dimension

1D Row $n = m$ tridiagonal



$O(n) = O(m)$

2D Grid $n = m^2$ 5–diagonals



$O(n^{1.5}) = O(m^3)$

3D Grid $n = m^3$ $O(n^2) = O(m^6)$

- What should you pivot for?
  - Growth control?
  - or to Avoid Fill-ins?
  - A.) LU factorization applied to a strictly diagonally dominant matrix will <u>never</u> produce a zero pivot.
  - B.) The matrix entries produced by LU factorization applied to a strictly diagonally dominant matrix will never increase by more than a factor of $2^{(n-1)}$. [which is the best you can do by pivoting for growth control]

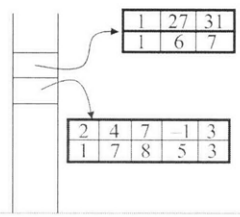
  Bottom Line:
  If your matrix is strictly diagonally dominant no need for numerical pivot for growth control so just pivot for sparsity control!
- Sparse Factorization Approach
  1. Assume matrix requires NO numerical pivoting.
     Diagonally dominant or symmetric positive definite.
  2. Use graphs to determine matrix ordering.
     Many graph manipulation tricks used.
  3. Form data structures for storing filled-in matrix.
     Lots of additional nonzeros added.
  4. Put numerical values in data structure and factor
     Computation must be organized carefully!

❖ Sparse Matrix Data Structures

Example 1.

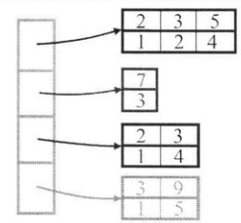
$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 27 & 31 & 0 & \cdots \\ 2 & 0 & 3 & 0 & -1 & 0 & 4 & 7 & \cdots \end{bmatrix}$$

$\longleftarrow n^2 \longrightarrow$



Example 2.

$$\begin{bmatrix} 2 & 3 & 0 & 5 & 0 \\ 0 & 0 & 7 & 0 & 0 \\ 2 & 0 & 0 & 3 & 0 \\ 3 & 0 & 0 & 0 & 9 \end{bmatrix}$$



Example 3. 1000×1000 matrix

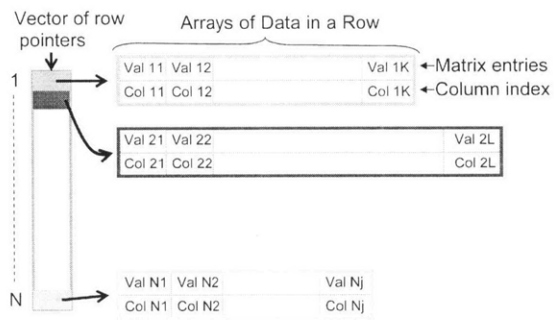
$$\begin{bmatrix} * & & * & * & & & & \\ & * & & & * & & * & \\ * & & & & & & * & * \\ & & * & & * & & & * \\ & & & * & & * & * & \\ & * & * & & & & & * \\ * & & & * & & & & * \\ & & & & * & * & * & \end{bmatrix}$$

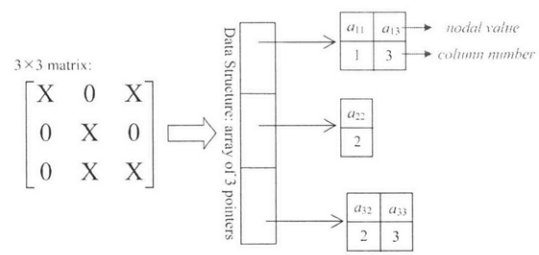
store it all: $1000 \times 1000 \rightarrow$ 8MB

3000 nonzero entries to store

$3000 \times 2 \times 8\text{B} + 8\text{B} \times 3000$

$\rightarrow$ 50 KB

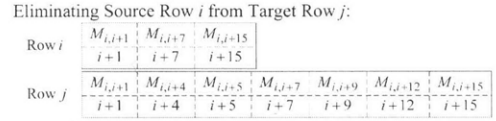




Why store sparse matrix information in a data structure array?

- Too much storage space storing all the zero entries
- Avoid floating point computation on all the zeros (minimal in comparison to the memory cost)
- Memory reference cost

Eliminating Source Row i from Target Row j:



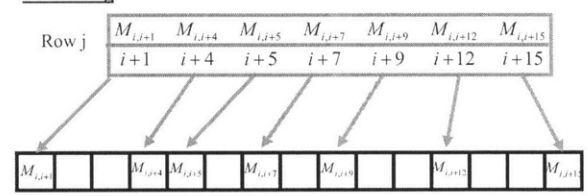
Must read in **all** the row j entries to find the three that match row i.

	Rows	Ops	Misses
Res	300	904,387	248,967
RAM	2,806	1,017,289	3,817,587
Grid	4,356	3,180,726	3,597,746

More misses than operations!

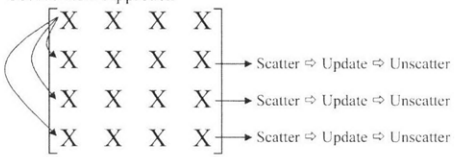
Every miss is an unneeded memory reference!

Scattering

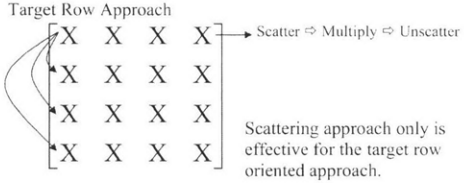


1) Read all the elements in Row j, and scatter them in an n-length vector
2) Access only the needed elements using array indexing!

Source Row Approach



Target Row Approach



Scattering approach only is effective for the target row oriented approach.

- Summary of Sparse Systems
  - Sparse Matrices
    - Struts, resistor grids, 3-D heat flow → $O(N)$ nonzeros
  - Tridiagonal Matrix Factorization
    - Factor in $O(N)$ operations
  - General Sparse Factorization
    - Markowitz Reordering to minimize fill
  - Graph Based Approach
    - Factorization and Fill-in
    - Useful for estimating Sparse GE complexity
  - Sparse Data Structures
    - Scattering

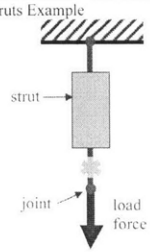
3.7 Linear Systems - QR Factorization

INTRODUCTION TO NUMERICAL SIMULATION

LECTURE 6.B.
QR Factorization

TODAY'S OUTLINE:
- Singular Problems
- Projection Formulas
- Modified Gram-Schmidt Algorithm

QR FACTORIZATION
Singular Problems – LU Factorization Fails
- Struts Example

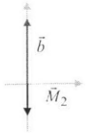


$$f_x^* + f_{L_x} = 0 \qquad u_x = x - x_0$$
$$f_y^* + f_{L_y} = 0 \qquad u_y = y - y_0$$

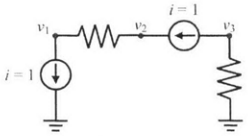
$$\underbrace{\begin{bmatrix} 0 \\ 0 \end{bmatrix}}_{\vec{M}_1} \underbrace{\begin{bmatrix} 0 \\ -\varepsilon \end{bmatrix}}_{\vec{M}_2} \begin{bmatrix} u_x \\ u_y \end{bmatrix} = \underbrace{\begin{bmatrix} 0 \\ +3 \end{bmatrix}}_{\vec{b}}$$

$\vec{f}_L = (0,-3)$

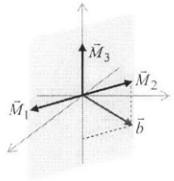
$\vec{b} \in span\{\vec{M}_1, \vec{M}_2\} \Rightarrow$ infinite solutions



The resulting nodal matrix is SINGULAR:
- LU decomposition fails
- But a solution exists! Actually, many....



$$\begin{bmatrix} +1 & -1 & 0 \\ -1 & +1 & 0 \\ 0 & 0 & +1 \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \\ v_3 \end{bmatrix} = \begin{bmatrix} -1 \\ +1 \\ -1 \end{bmatrix}$$
$$\vec{M}_1 \quad \vec{M}_2 \quad \vec{M}_3 \qquad \vec{b}$$



$\vec{b} \in span\{\vec{M}_1, \vec{M}_2, \vec{M}_3\} \Rightarrow$ solution exists

plane

$\vec{M}_1$ and $\vec{M}_2$ are L.D. $\Rightarrow$ infinite solutions

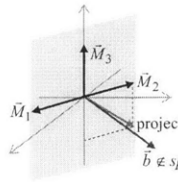
e.g. $\vec{b} = \underbrace{0}_{x_1} \cdot \vec{M}_1 + \underbrace{1}_{x_2} \cdot \vec{M}_2 - \underbrace{1}_{x_3} \cdot \vec{M}_3$

$\vec{b} = \underbrace{-1}_{x_1} \cdot \vec{M}_1 + \underbrace{0}_{x_2} \cdot \vec{M}_2 - \underbrace{1}_{x_3} \cdot \vec{M}_3$

LU will fail

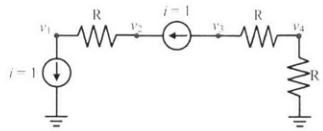
$$\begin{bmatrix} +1 & -1 & 0 \\ -1 & +1 & 0 \\ 0 & 0 & +1 \end{bmatrix} \rightarrow \begin{bmatrix} 1 & -1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

No pivots available



projection of $\vec{b}$ onto $span\{\vec{M}_1, \vec{M}_2, \vec{M}_3\}$

$\vec{b} \notin span\{\vec{M}_1, \vec{M}_2, \vec{M}_3\}$



$$G = \frac{1}{R}$$

$$\begin{bmatrix} G & -G & & \\ -G & G & & \\ & & G & -G \\ & & -G & 2G \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \\ v_3 \\ v_4 \end{bmatrix} = \begin{bmatrix} -1 \\ 1 \\ -1 \\ 0 \end{bmatrix}$$

Let $R = 1$

$$\begin{bmatrix} 1 & -1 & & \\ -1 & 1 & & \\ & & 1 & -1 \\ & & -1 & 2 \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \\ v_3 \\ v_4 \end{bmatrix} = \begin{bmatrix} -1 \\ 1 \\ -1 \\ 0 \end{bmatrix}$$

$$\xrightarrow[\text{after 1 LUstep}]{} \begin{bmatrix} 1 & -1 & & \\ 0 & 0 & & \\ & & 1 & -1 \\ & & -1 & 2 \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \\ v_3 \\ v_4 \end{bmatrix} = \begin{bmatrix} -1 \\ 1 \\ -1 \\ 0 \end{bmatrix}$$

Recall weighted sum of columns view of systems of equations

$$\begin{bmatrix} \uparrow & \uparrow & & \uparrow \\ M_1 & M_2 & \cdots & M_N \\ \downarrow & \downarrow & & \downarrow \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_N \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_N \end{bmatrix} \Rightarrow x_1\vec{M}_1 + x_2\vec{M}_2 + \cdots + x_N\vec{M}_N = \vec{b}$$

M is singular but b is in the span of the columns of **M** so there is a solution, actually lots of them.
How do we find them?

❖ Systems of Linear Equations – Summary Table

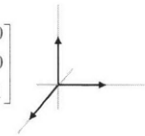
	$b \in range\{M\}$	$b \notin range\{M\}$
L.I. columns	Solution exists and is unique Use **LU**	No solutions Find the "closest"
L.D. columns	Infinite solutions exist Find one ... or all	No solutions Find the "closest"

Projection Formulas – Orthogonal Projection

3-D Orthogonal Vectors

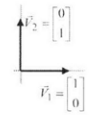
Example of orthogonal vectors

$$\begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$$

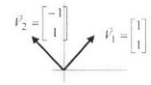


❖ Orthonormal **Q** Picture

$$\vec{Q}_i \cdot \vec{Q}_j = \vec{Q}_i^T \vec{Q}_j = \begin{bmatrix} & \vec{Q}_i^T & \end{bmatrix} \begin{bmatrix} \\ \vec{Q}_j \\ \\ \end{bmatrix}$$



$$\vec{V}_2 \cdot \vec{V}_1 = \vec{V}_2^T \vec{V}_1 = \begin{bmatrix} 0 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix} = 0$$

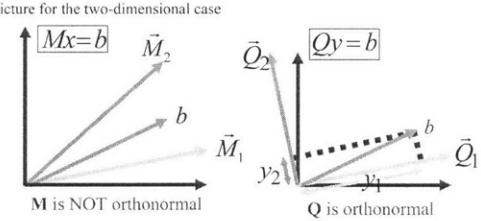


$$\vec{V}_2 \cdot \vec{V}_1 = \vec{V}_2^T \vec{V}_1 = \begin{bmatrix} -1 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \end{bmatrix} = -1 + 1 = 0$$

Definition of orthonormal matrix **Q**:

$\vec{Q}_i \cdot \vec{Q}_j = 0$ if $i \neq j$ and $\vec{Q}_i \cdot \vec{Q}_i = 1$

Picture for the two-dimensional case



M is NOT orthonormal

Q is orthonormal

$$\mathbf{Q} = \begin{bmatrix} \uparrow & \uparrow \\ \vec{Q}_1 & \vec{Q}_2 \\ \downarrow & \downarrow \end{bmatrix} \qquad \vec{Q}_1 \cdot \vec{Q}_2 = \vec{Q}_1^T \vec{Q}_2 \qquad y_1 = \vec{Q}_1^T \vec{b}$$

$$\vec{y} = \mathbf{Q}^T \vec{b} \qquad y_2 = \vec{Q}_2^T \vec{b}$$

❖ Projection Formula

$\boxed{\mathbf{Q}\vec{y} = \vec{b}}$ solving an orthonormal system is easy…

$y_1\vec{Q}_1 + y_2\vec{Q}_2 + \ldots + y_N\vec{Q}_N = \vec{b}$

Example.

$$Qy = b \Rightarrow y_1\begin{bmatrix} \vec{Q}_1 \end{bmatrix} + y_2\begin{bmatrix} \vec{Q}_2 \end{bmatrix} = \begin{bmatrix} \vec{b} \end{bmatrix}$$

Multiplying the weighted columns equation by i^{th} column:

$\vec{Q}_i \cdot \left(y_1\vec{Q}_1 + y_2\vec{Q}_2 + \ldots + y_N\vec{Q}_N\right) = \vec{Q}_i \cdot \vec{b}$

Example.

$$\vec{Q}_1 \cdot \left(y_1\begin{bmatrix} \vec{Q}_1 \end{bmatrix} + y_2\begin{bmatrix} \vec{Q}_2 \end{bmatrix} = \begin{bmatrix} \vec{b} \end{bmatrix} \right) \Rightarrow y_1\underbrace{\left(\vec{Q}_1 \cdot \vec{Q}_1\right)}_{1} + y_2\underbrace{\left(\vec{Q}_1 \cdot \vec{Q}_2\right)}_{0} = \vec{Q}_1 \cdot \vec{b}$$

$$\vec{Q}_2 \cdot \left(y_1\begin{bmatrix} \vec{Q}_1 \end{bmatrix} + y_2\begin{bmatrix} \vec{Q}_2 \end{bmatrix} = \begin{bmatrix} \vec{b} \end{bmatrix} \right) \Rightarrow y_1\underbrace{\left(\vec{Q}_2 \cdot \vec{Q}_1\right)}_{0} + y_2\underbrace{\left(\vec{Q}_2 \cdot \vec{Q}_2\right)}_{1} = \vec{Q}_2 \cdot \vec{b}$$

Simplifying using orthogonormality we get the projection formula:

$y_i = \vec{Q}_i \cdot \vec{b}$

Example.

$$\begin{aligned} y_1 &= \vec{Q}_1 \cdot \vec{b} = \vec{Q}_1^T \vec{b} \\ y_2 &= \vec{Q}_2 \cdot \vec{b} = \vec{Q}_2^T \vec{b} \\ y_3 &= \vec{Q}_3 \cdot \vec{b} = \vec{Q}_3^T \vec{b} \\ &\vdots \\ y_N &= \vec{Q}_N \cdot \vec{b} = \vec{Q}_N^T \vec{b} \end{aligned} \qquad \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ \vdots \\ y_N \end{bmatrix} = \underbrace{\begin{bmatrix} \leftarrow & \vec{Q}_1^T & \rightarrow \\ \leftarrow & \vec{Q}_2^T & \rightarrow \\ \leftarrow & \vec{Q}_3^T & \rightarrow \\ & \vdots & \\ \leftarrow & \vec{Q}_N^T & \rightarrow \end{bmatrix}}_{\mathbf{Q}^T} \begin{bmatrix} \vec{b} \end{bmatrix}$$

Know $\mathbf{Q}\vec{y} = \vec{b}$; have shown that $\vec{y} = \mathbf{Q}^T\vec{b}$

$$\mathbf{Q}^T\mathbf{Q} = \begin{bmatrix} \overbrace{\vec{Q}_1^T\vec{Q}_1}^{1} & \overbrace{\vec{Q}_1^T\vec{Q}_2}^{0} & \overbrace{\vec{Q}_1^T\vec{Q}_3}^{0} & \cdots \\ \underbrace{\vec{Q}_2^T\vec{Q}_1}_{0} & \underbrace{\vec{Q}_2^T\vec{Q}_2}_{1} & \underbrace{\vec{Q}_2^T\vec{Q}_3}_{0} & \\ & & \ddots & \\ & & & \underbrace{\vec{Q}_N^T\vec{Q}_N}_{1} \end{bmatrix} = I$$

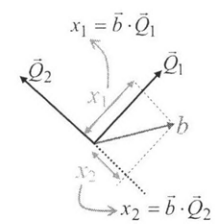
$$\mathbf{Q}^{-1}\mathbf{Q} = \mathbf{Q}^T\mathbf{Q} = I \quad \Rightarrow \quad \mathbf{Q}^T = \mathbf{Q}^{-1}$$

❖ QR Algorithm Key Idea

$$\underbrace{\begin{bmatrix} \uparrow & \uparrow & & \uparrow \\ \vec{M}_1 & \vec{M}_2 & \cdots & \vec{M}_N \\ \downarrow & \downarrow & & \downarrow \end{bmatrix}}_{\text{original matrix}} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_N \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_N \end{bmatrix} \Rightarrow \underbrace{\begin{bmatrix} \uparrow & \uparrow & & \uparrow \\ \vec{Q}_1 & \vec{Q}_2 & \cdots & \vec{Q}_N \\ \downarrow & \downarrow & & \downarrow \end{bmatrix}}_{\text{matrix with orthonormal columns}} \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_N \end{bmatrix}$$

If we have a set of orthonormal columns that span the same subspace as the columns of **M**, then solving the system is very easy:

$$\boxed{\mathbf{Q}\vec{y} = \vec{b} \Rightarrow \vec{y} = \mathbf{Q}^T\vec{b}}$$



$$\begin{bmatrix} \vec{Q}_1 & \vec{Q}_2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = x_1\vec{Q}_1 + x_2\vec{Q}_2 = \vec{b}$$

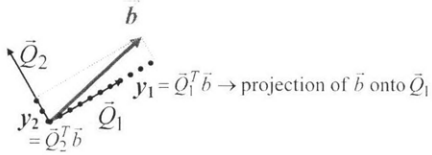
$$\vec{Q}_1 \cdot \vec{Q}_1 = 1$$
$$\vec{Q}_2 \cdot \vec{Q}_2 = 1$$
$$\vec{Q}_1 \cdot \vec{Q}_2 = \vec{Q}_2 \cdot \vec{Q}_1 = 0$$

$$\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} \vec{Q}_1 \\ \vec{Q}_2 \end{bmatrix} \begin{bmatrix} \vec{b} \end{bmatrix} = \begin{bmatrix} \vec{Q}_1^T\vec{b} \\ \vec{Q}_2^T\vec{b} \end{bmatrix}$$

$$\begin{bmatrix} \uparrow \\ \vec{y} \\ \downarrow \end{bmatrix} = \begin{bmatrix} \leftarrow & \vec{Q}_1^T & \rightarrow \\ \leftarrow & \vec{Q}_2^T & \rightarrow \\ & \vdots & \end{bmatrix} \begin{bmatrix} \uparrow \\ \vec{b} \\ \downarrow \end{bmatrix} = \begin{bmatrix} \vec{Q}_1^T \vec{b} \\ \vec{Q}_2^T \vec{b} \\ \vdots \end{bmatrix} \rightarrow \text{projection of } \vec{b} \text{ onto } \vec{Q}_1$$

$y_i = \vec{Q}_i^T \vec{b}$

Result is the projection of $\vec{b}$ onto the columns of $\vec{Q}$



But how do we perform the conversion from **M** to **Q**?

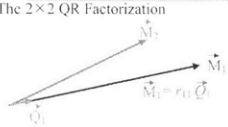
Example.

$$\mathbf{M} = \begin{bmatrix} \uparrow & \uparrow \\ M_1 & M_2 \\ \downarrow & \downarrow \end{bmatrix} = \begin{bmatrix} +1 & 0 \\ -1 & 0 \\ 0 & 0 \end{bmatrix}$$

$$r_{11} = \sqrt{1^2 + (-1)^2} = \sqrt{2}$$

$$\vec{Q}_1 = \frac{\vec{M}_1}{r_{11}} = \begin{bmatrix} \frac{1}{\sqrt{2}} \\ -\frac{1}{\sqrt{2}} \\ 0 \end{bmatrix}$$

❖ The 2×2 QR Factorization



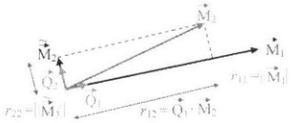
$$r_{11} = \|\vec{M}_1\| = \sqrt{\vec{M}_1 \cdot \vec{M}_1} = \sqrt{\vec{M}_1^T \vec{M}_1}$$

Getting the first orthonormal vector is very easy:

$$\vec{Q}_1 = \frac{\vec{M}_1}{r_{11}} = \frac{\vec{M}_1}{|\vec{M}_1|}$$

Given $\vec{M}_1, \vec{M}_2$, find $\vec{Q}_2 = \vec{M}_2 - r_{12}\vec{M}_1$ so that

$$\vec{M}_1 \cdot \vec{Q}_2 = \vec{M}_1 \cdot (\vec{M}_2 - r_{12}\vec{M}_1) = 0 \Rightarrow r_{12} = \frac{\vec{M}_1 \cdot \vec{M}_2}{\vec{M}_1 \cdot \vec{M}_1}$$



Using the Projection formula

$$\vec{M}_1 = r_{11}\vec{Q}_1$$

$$\vec{M}_2 = r_{12}\vec{Q}_1 + r_{22}\vec{Q}_2 \Rightarrow \begin{cases} \tilde{\vec{M}}_2 = \vec{M}_2 - r_{12}\vec{Q}_1 \\ \vec{Q}_2 = \dfrac{\tilde{\vec{M}}_2}{|\tilde{\vec{M}}_2|} \end{cases}$$

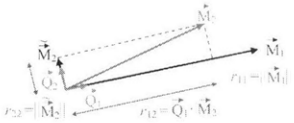
$$r_{12} = \vec{M}_1 \cdot \vec{M}_2 = \vec{Q}_1 \cdot \vec{M}_2 = \vec{Q}_1 \cdot (r_{12} \cdot \vec{Q}_1 + r_{22} \cdot \vec{Q}_2) = r_{12}\underbrace{\vec{Q}_1 \cdot \vec{Q}_1}_{1} + r_{22}\underbrace{\vec{Q}_1 \cdot \vec{Q}_2}_{0}$$

Formulas simplify if we normalize

$$\vec{Q}_1 = \frac{\vec{M}_1}{\sqrt{\vec{M}_1 \cdot \vec{M}_1}} = \frac{1}{r_{11}}\vec{M}_1 \Rightarrow \vec{Q}_1 \cdot \vec{Q}_1 = 1$$

Now find $\tilde{\vec{Q}}_2 = \vec{M}_2 - r_{12}\vec{Q}_1$ so that $\tilde{\vec{Q}}_2 \cdot \vec{Q}_1 = 0 \Rightarrow r_{12} = \vec{Q}_1 \cdot \vec{M}_2$

Finally $\vec{Q}_2 = \dfrac{1}{\sqrt{\tilde{\vec{Q}}_2 \cdot \tilde{\vec{Q}}_2}}\tilde{\vec{Q}}_2 = \dfrac{1}{r_{22}}\tilde{\vec{Q}}_2$



$$\begin{aligned} \vec{M}_1 &= r_{11}\vec{Q}_1 & \vec{M}_1 &= r_{11}\vec{Q}_1 + 0\vec{Q}_2 \\ \vec{M}_2 &= r_{12}\vec{Q}_1 + r_{22}\vec{Q}_2 & \vec{M}_2 &= r_{12}\vec{Q}_1 + r_{22}\vec{Q}_2 \end{aligned}$$

Another way to write this is: **M** is factored into **Q** and **R**

$$\underbrace{\begin{bmatrix} \vec{M}_1 & \vec{M}_2 \end{bmatrix}}_{\mathbf{M}} = \underbrace{\begin{bmatrix} \vec{Q}_1 & \vec{Q}_2 \end{bmatrix}}_{\mathbf{Q}} \underbrace{\begin{bmatrix} r_{11} & r_{12} \\ 0 & r_{22} \end{bmatrix}}_{\mathbf{R}}$$

Since $\mathbf{M}x$ should equal $\mathbf{Q}y$, we can relate x to y

$$\begin{bmatrix} \uparrow & \uparrow \\ \vec{M}_1 & \vec{M}_2 \\ \downarrow & \downarrow \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = x_1\vec{M}_1 + x_2\vec{M}_2 = \begin{bmatrix} \uparrow & \uparrow \\ \vec{Q}_1 & \vec{Q}_2 \\ \downarrow & \downarrow \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = y_1\vec{Q}_1 + y_2\vec{Q}_2$$

$$\begin{aligned} \vec{M}_1 &= r_{11}\vec{Q}_1 \\ \vec{M}_2 &= r_{22}\vec{Q}_2 + r_{12}\vec{Q}_1 \end{aligned} \Rightarrow \begin{bmatrix} r_{11} & r_{12} \\ 0 & r_{22} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \end{bmatrix}$$

Three Step Solve Procedure:

1. Factor $\mathbf{M} = \mathbf{QR}$, $\mathbf{Q}$ orthonormal, $\mathbf{R}$ upper triangular

$$\mathbf{M}\vec{x} = \vec{b} \qquad x_1\vec{M}_1 + x_2\vec{M}_2 = \vec{b}$$

Decompose $\mathbf{Q}\cdot\underbrace{\mathbf{R}\vec{x}}_{\vec{y}} = \vec{b}$

2. Solve $\mathbf{Q}y = b$. (Very easy: $y = \mathbf{Q}^T b$)
3. Backsolve the triangular system: $\mathbf{R}x = y$.

❖ The 3×3 QR Factorization

Use previously orthonormalized vectors

$$\begin{bmatrix} \uparrow & \uparrow & \uparrow \\ \vec{M}_1 & \vec{M}_2 & \vec{M}_3 \\ \downarrow & \downarrow & \downarrow \end{bmatrix} \Rightarrow \begin{bmatrix} \uparrow & \uparrow & \uparrow \\ \vec{M}_1 & \vec{M}_2 - r_{12}\vec{Q}_1 & \vec{M}_3 - r_{13}\vec{Q}_1 - r_{23}\vec{Q}_2 \\ \downarrow & \downarrow & \downarrow \end{bmatrix}$$

To ensure the third column is orthogonal

$$\vec{Q}_1 \cdot \left(\vec{M}_3 - \vec{Q}_1 r_{13} - \vec{Q}_2 r_{23}\right) = 0 \quad \Rightarrow \quad r_{13} = \vec{Q}_1 \cdot \vec{M}_3$$

$$\vec{Q}_2 \cdot \left(\vec{M}_3 - \vec{Q}_1 r_{13} - \vec{Q}_2 r_{23}\right) = 0 \quad \Rightarrow \quad r_{23} = \vec{Q}_2 \cdot \vec{M}_3$$

$$\begin{aligned} \vec{M}_1 \cdot (\vec{M}_3 - r_{13}\vec{M}_1 - r_{23}\vec{M}_2) &= 0 \\ \vec{M}_2 \cdot (\vec{M}_3 - r_{13}\vec{M}_1 - r_{23}\vec{M}_2) &= 0 \end{aligned} \Rightarrow \begin{bmatrix} \vec{M}_1 \cdot \vec{M}_1 & \vec{M}_1 \cdot \vec{M}_2 \\ \vec{M}_2 \cdot \vec{M}_1 & \vec{M}_2 \cdot \vec{M}_2 \end{bmatrix} \begin{bmatrix} r_{13} \\ r_{23} \end{bmatrix} = \begin{bmatrix} \vec{M}_1 \cdot \vec{M}_3 \\ \vec{M}_2 \cdot \vec{M}_3 \end{bmatrix}$$

To Orthogonalize the Nth Vector:

$$\begin{bmatrix} \vec{M}_1 \cdot \vec{M}_1 & \cdots & \vec{M}_1 \cdot \vec{M}_{N-1} \\ \vdots & \ddots & \vdots \\ \vec{M}_{N-1} \cdot \vec{M}_1 & \cdots & \vec{M}_{N-1} \cdot \vec{M}_{N-1} \end{bmatrix} \begin{bmatrix} r_{1,N} \\ \vdots \\ r_{N-1,N} \end{bmatrix} = \begin{bmatrix} \vec{M}_1 \cdot \vec{M}_N \\ \vdots \\ \vec{M}_N \cdot \vec{M}_N \end{bmatrix}$$

N^2 inner products or N^3 work.

$$\begin{aligned} \vec{M}_3 &= r_{13}\vec{Q}_1 + r_{23}\vec{Q}_2 + r_{33}\vec{Q}_3 \\ \tilde{\vec{M}}_3 &= \vec{M}_3 - (\vec{Q}_1 \cdot \vec{M}_3)\vec{Q}_1 - (\vec{Q}_2 \cdot \vec{M}_3)\vec{Q}_2 \\ \Rightarrow \tilde{\vec{M}}_3 &= \vec{M}_3 - r_{13}\vec{Q}_1 - r_{23}\vec{Q}_2 \\ \vec{Q}_3 &= \frac{\tilde{\vec{M}}_3}{\left\|\tilde{\vec{M}}_3\right\|} \end{aligned} \qquad \begin{aligned} r_{13} &= \vec{Q}_1 \cdot \vec{M}_3 \\ r_{23} &= \vec{Q}_2 \cdot \vec{M}_3 \\ r_{33} &= \vec{Q}_3 \cdot \vec{M}_3 = \left\|\tilde{\vec{M}}_3\right\| \end{aligned}$$

$$\vec{M}_1 \quad \vec{Q}_1 \leftarrow \frac{\vec{M}_1}{\|\vec{M}_1\|} \leftarrow r_{11}$$

$$\vec{M}_2 \quad \vec{M}_2 \leftarrow \vec{M}_2 - \underbrace{(\vec{Q}_1 \cdot \vec{M}_2)}_{r_{12}}\vec{Q}_1 \qquad \vec{Q}_2 \leftarrow \frac{\vec{M}_2}{\|\vec{M}_2\|} \leftarrow r_{22}$$

$$\vec{M}_3 \quad \vec{M}_3 \leftarrow \vec{M}_3 - \underbrace{(\vec{Q}_1 \cdot \vec{M}_3)}_{r_{13}}\vec{Q}_1 - \underbrace{(\vec{Q}_2 \cdot \vec{M}_3)}_{r_{23}}\vec{Q}_2 \qquad \vec{Q}_3 \leftarrow \frac{\vec{M}_3}{\|\vec{M}_3\|} \leftarrow r_{33}$$

$$\vec{M}_4 \quad \vec{M}_4 \leftarrow \vec{M}_4 - \underbrace{(\vec{Q}_1 \cdot \vec{M}_4)}_{r_{14}}\vec{Q}_1 - \underbrace{(\vec{Q}_2 \cdot \vec{M}_4)}_{r_{24}}\vec{Q}_2 - \underbrace{(\vec{Q}_3 \cdot \vec{M}_4)}_{r_{34}}\vec{Q}_3 \qquad \vec{Q}_4 \leftarrow \frac{\vec{M}_4}{\|\vec{M}_4\|} \leftarrow r_{44}$$

$$\vdots$$

$$\vec{M}_N \quad \vec{M}_N \leftarrow \vec{M}_N - \underbrace{(\vec{Q}_1 \cdot \vec{M}_N)}_{r_{1N}}\vec{Q}_1 - \underbrace{(\vec{Q}_2 \cdot \vec{M}_N)}_{r_{2N}}\vec{Q} - \cdots - \underbrace{(\vec{Q}_{N-1} \cdot \vec{M}_N)}_{r_{N-1,N}}\vec{Q}_{N-1}$$

$$\begin{bmatrix} \vec{M}_1 & \vec{M}_2 & \vec{M}_3 \end{bmatrix} = \begin{bmatrix} \cancel{\vec{M}_1} & \cancel{\vec{M}_2} & \cancel{\vec{M}_3} \\ \vec{Q}_1 & \tilde{\vec{M}}_2 & \tilde{\vec{M}}_3 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ & & \\ & & \end{bmatrix}$$

$$\begin{bmatrix} \vec{M}_1 & \vec{M}_2 & \vec{M}_3 \end{bmatrix} = \begin{bmatrix} \vec{Q}_1 & \cancel{\tilde{\vec{M}}_2} & \cancel{\tilde{\vec{M}}_3} \\ & \vec{Q}_2 & \tilde{\tilde{\vec{M}}}_3 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ & r_{22} & r_{23} \\ & & \end{bmatrix}$$

$$\begin{bmatrix} \vec{M}_1 & \vec{M}_2 & \vec{M}_3 \end{bmatrix} = \begin{bmatrix} \vec{Q}_1 & \vec{Q}_2 & \cancel{\vec{M}_3} \\ & & \vec{Q}_3 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ & r_{22} & r_{23} \\ & & r_{33} \end{bmatrix}$$

Modified Gram-Schmidt Algorithm

- "By Picture"

$$\vec{M}_j \leftarrow \vec{M}_j - r_{ij}\vec{Q}_i$$

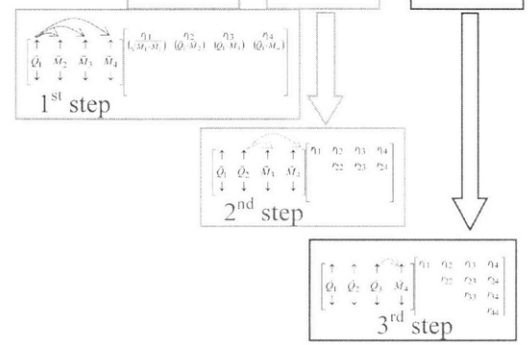
$$\begin{bmatrix} \uparrow & \uparrow & \uparrow & \uparrow \\ \vec{Q}_1 & \vec{Q}_2 & \vec{Q}_3 & \vec{Q}_4 \\ \downarrow & \downarrow & \downarrow & \downarrow \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & r_{14} \\ & r_{22} & r_{23} & r_{24} \\ & & r_{33} & r_{34} \\ & & & r_{44} \end{bmatrix}$$

$$\vec{Q}_i = \frac{1}{r_{ii}}\vec{M}_i \qquad r_{ii} = \sqrt{\vec{M}_i \cdot \vec{M}_i}$$

$$r_{ij} \leftarrow \vec{M}_j \cdot \vec{Q}_i$$

$$\begin{aligned} \vec{M}_2 &\leftarrow \vec{M}_2 - (\vec{Q}_1 \cdot \vec{M}_2)\vec{Q}_1 \\ \vec{M}_3 &\leftarrow \vec{M}_3 - (\vec{Q}_1 \cdot \vec{M}_3)\vec{Q}_1 - (\vec{Q}_2 \cdot \vec{M}_3)\vec{Q}_2 \\ \vec{M}_4 &\leftarrow \vec{M}_4 - (\vec{Q}_1 \cdot \vec{M}_4)\vec{Q}_1 - (\vec{Q}_2 \cdot \vec{M}_4)\vec{Q}_2 - (\vec{Q}_3 \cdot \vec{M}_4)\vec{Q}_3 \end{aligned}$$

$$\vec{Q}_1 \leftarrow \frac{\vec{M}_1}{\|\vec{M}_1\|} \qquad \vec{Q}_2 \leftarrow \frac{\vec{M}_2}{\|\vec{M}_2\|} \qquad \vec{Q}_3 \leftarrow \frac{\vec{M}_3}{\|\vec{M}_3\|}$$



- Basic Algorithm
  - Source Row Approach

For $i = 1$ to N { "For each Source Column"

$$\left.\begin{aligned} r_{ii} &= \sqrt{\vec{M}_i \cdot \vec{M}_i} \\ \vec{Q}_i &= \frac{1}{r_{ii}}\vec{M}_i \end{aligned}\right\} \text{Normalize} \quad \sum_{i=1}^{N} 2N \approx 2N^2 \text{ operations}$$

For $j = i + 1$ to N { "For each target column right of source"

$$\left.\begin{aligned} r_{ij} &\leftarrow \vec{M}_j \cdot \vec{Q}_i \\ \vec{M}_j &\leftarrow \vec{M}_j - r_{ij}\vec{Q}_i \end{aligned}\right\} \sum_{i=1}^{N} (N-i)2N \approx N^3 \text{ operations}$$

○ Target Row Approach

For $i = 1$ to N { "For each Target Column"

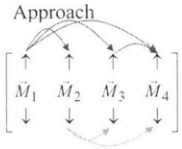
$\vec{M}_i \leftarrow \mathbf{M}\vec{e}_i$ "matrix-vector product"

For $j = 1$ to $i - 1$ { "For each source column left of target"

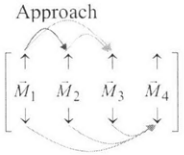
$$\left.\begin{aligned} r_{ji} &\leftarrow \vec{Q}_j \cdot \vec{M}_i \\ \vec{M}_i &\leftarrow \vec{M}_i - r_{ji}\vec{Q}_j \end{aligned}\right\} \sum_{i=1}^{N}(N-i)2N \approx N^3 \text{ operations}$$

$$\left.\begin{aligned} r_{ii} &= \sqrt{\vec{M}_i \cdot \vec{M}_i} \\ \vec{Q}_i &= \frac{1}{r_{ii}}\vec{M}_i \end{aligned}\right\} \text{Normalize} \quad \sum_{i=1}^{N} 2N \approx 2N^2 \text{ operations}$$

Source Row Approach



Target Row Approach



❖ Three Step Solve Procedure

		Computational Complexity
Step 1)	Factor $\mathbf{M} = \mathbf{QR}$ ($\mathbf{Q}$ orthonormal, $\mathbf{R}$ upper triangular)	$O(N^3)$
Step 2)	Solve $\mathbf{Q}\vec{y} = \vec{b}$. $\left(\text{Very easy}: \vec{y} = \mathbf{Q}^T\vec{b}\right)$	$O(N^2)$
Step 3)	Backsolve the triangular system $R\vec{x} = \vec{y}$	$O(N^2)$

$\mathbf{M}\vec{x} = \vec{b}$

Step 1) Factor $\mathbf{M} = \mathbf{QR}$

$$\mathbf{M}\vec{x} = \vec{b} \Leftrightarrow \mathbf{Q}\underbrace{\mathbf{R}\vec{x}}_{\vec{y}} = \vec{b}$$

Step 2) Solve for $\vec{y}$: $\mathbf{Q}\vec{y} = \vec{b}$

$$y_i = \vec{Q}_i \cdot \vec{b} = \vec{Q}_i^T\vec{b} = \begin{bmatrix}\leftarrow & \vec{Q}_i^T & \rightarrow\end{bmatrix}\begin{bmatrix}\uparrow \\ \vec{b} \\ \downarrow\end{bmatrix}$$

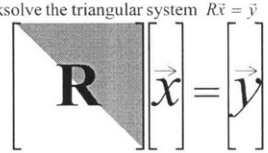
$\left(\text{Very easy}: \vec{y} = \mathbf{Q}^T\vec{b}\right)$

$$\vec{y} = \mathbf{Q}^T\vec{b} = \begin{bmatrix}\leftarrow & \vec{Q}_1 & \rightarrow \\ \leftarrow & \vec{Q}_2 & \rightarrow \\ \leftarrow & \vec{Q}_3 & \rightarrow \\ & \vdots & \\ \leftarrow & \vec{Q}_n & \rightarrow\end{bmatrix}\begin{bmatrix}\uparrow \\ \vec{b} \\ \downarrow\end{bmatrix}$$

$$\vec{b} = y_1\vec{Q}_1 + y_2\vec{Q}_2 + y_3\vec{Q}_3 + \cdots y_n\vec{Q}_n$$

$$\vec{b} = x_1\vec{M}_1 + x_2\vec{M}_2 + x_3\vec{M}_3 + \cdots x_n\vec{M}_n$$

Step 3) Backsolve the triangular system $R\vec{x} = \vec{y}$



3.8 Linear Systems - GCR Iterative Method

INTRODUCTION TO NUMERICAL SIMULATION

LECTURES 7 & 8.

QR and Krylov-Subspace Matrix Solution Methods

TODAY'S OUTLINE:

- Minimization View of QR
  - Singular Matrix
  - Basic Minimization Approach
  - Orthogonalized Search Directions
  - QR and Length Minimization Produce Identical Results
- Arbitrary Subspace Algorithm
  - Orthogonalization of Search Directions
- Generalized Conjugate Residual Algorithm
  - Krylov-subspace
  - Simplification in the symmetric case
  - Leaky and insulating examples

QR FACTORIZATION

Example.

$$\mathbf{A} = \begin{bmatrix} 2 & -1 & 0 \\ -1 & 2 & -1 \\ 0 & -1 & 2 \end{bmatrix}$$

$$\underbrace{\begin{bmatrix} 1 & 0 & 0 \\ -\frac{1}{2} & 1 & 0 \\ - & -\frac{1}{3} & 1 \end{bmatrix}}_{\mathbf{L}} \underbrace{\begin{bmatrix} 2 & -1 & 0 \\ 0 & \frac{3}{2} & -1 \\ 0 & 0 & \frac{4}{3} \end{bmatrix}}_{\mathbf{U}}$$

$$\underbrace{\begin{bmatrix} -0.89 & -0.35 & 0.27 \\ 0.44 & -0.72 & 0.53 \\ 0 & 0.60 & 0.80 \end{bmatrix}}_{\mathbf{Q}} \underbrace{\begin{bmatrix} -2.2 & 1.8 & -0.45 \\ 0 & -1.7 & 1.9 \\ 0 & 0 & 1.1 \end{bmatrix}}_{\mathbf{R}}$$

Matrix is Singular, column of Q is zero

- Zero Column

  If a column is zero

  $$\vec{M}_i = \sum_{j=1}^{i+1} w_j \vec{M}_j$$

  $\{\vec{M}_1, \ldots, \vec{M}_i\}$ not linearly independent

  What if a column becomes zero?

  $$\begin{bmatrix} \uparrow & \uparrow & \uparrow & & \uparrow \\ \vec{Q}_1 & 0 & \tilde{\vec{M}}_3 & \cdots & \tilde{\vec{M}}_N \\ \downarrow & \downarrow & \downarrow & & \downarrow \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & \cdots & r_{1N} \\ 0 & 0 & 0 & \cdots & 0 \\ 0 & 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 0 \end{bmatrix}$$

  $$\begin{bmatrix} \uparrow & \uparrow & \uparrow & & \uparrow \\ \vec{Q}_1 & 0 & \vec{Q}_3 & \cdots & \vec{Q}_N \\ \downarrow & \downarrow & \downarrow & & \downarrow \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & \cdots & r_{1N} \\ 0 & 0 & 0 & \cdots & 0 \\ 0 & 0 & r_{33} & \cdots & r_{3N} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & r_{NN} \end{bmatrix}$$

  Matrix MUST be Singular!!

  1. Do not try to normalize the column.
  2. Do not use the column as a source for orthogonalization.
  3. Perform backward substitution as well as possible.

$$\mathbf{QR}\vec{x} = \vec{b} \Rightarrow \mathbf{R}\vec{x} = \mathbf{Q}^T\vec{b}$$

$$\Rightarrow \begin{bmatrix} r_{11} & r_{12} & r_{13} & \cdots & r_{1N} \\ 0 & 0 & 0 & \cdots & 0 \\ & & r_{33} & \cdots & r_{3N} \\ & & & \ddots & \vdots \\ & & & & r_{NN} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_N \end{bmatrix} = \begin{bmatrix} \leftarrow & \vec{Q}_1 & \rightarrow \\ 0 & \cdots & 0 \\ \leftarrow & \vec{Q}_3 & \rightarrow \\ & \vdots & \\ \leftarrow & \vec{Q}_N & \rightarrow \end{bmatrix} \begin{bmatrix} \uparrow \\ \vec{b} \\ \downarrow \end{bmatrix}$$

QR Factorization

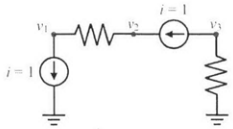
Problem $\mathbf{M}\vec{x} = \vec{b}$:

(1) $\mathbf{Q} \cdot \underbrace{\mathbf{R}\vec{x}}_{\vec{y}} = \vec{b}$

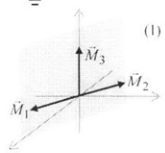
(2) Solve $\mathbf{Q}\vec{y} = \vec{b} \quad \Rightarrow \vec{y} = \mathbf{Q}^T\vec{b}$

(3) Solve $\mathbf{R}\vec{x} = \vec{y} \quad \rightarrow \vec{x}$

❖ Singular Example



$$\begin{array}{c} \begin{bmatrix} 1 & -1 & 0 \\ -1 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \\ \vec{M}_1 \quad \vec{M}_2 \quad \vec{M}_3 \end{array} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} -1 \\ 1 \\ -1 \end{bmatrix}$$



(1) Column $\vec{M}_1$

$r_{11} = \sqrt{\vec{M}_1 \cdot \vec{M}_1} = \sqrt{2}$ $\quad \vec{Q}_1 = \frac{1}{r_{11}}\vec{M}_1 = \begin{bmatrix} \frac{1}{\sqrt{2}} \\ -\frac{1}{\sqrt{2}} \\ 0 \end{bmatrix}$

$r_{12} = \vec{Q}_1 \cdot \vec{M}_2 = -\frac{1}{\sqrt{2}} - \frac{1}{\sqrt{2}} = -\sqrt{2}$

$r_{13} = \vec{Q}_1 \cdot \vec{M}_3 = 0$

$$\vec{M}_2 \leftarrow \vec{M}_2 - r_{12}\vec{Q}_1 = \begin{bmatrix} -1 \\ 1 \\ 0 \end{bmatrix} - \begin{bmatrix} -1 \\ 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

$$\vec{M}_3 \leftarrow \vec{M}_3 - r_{13}\vec{Q}_1 = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$$

$$\begin{array}{c} \begin{bmatrix} \frac{1}{\sqrt{2}} & 0 & 0 \\ -\frac{1}{\sqrt{2}} & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \\ \vec{Q}_1 \quad \vec{M}_2 \quad \vec{M}_3 \end{array} \begin{bmatrix} \sqrt{2} & -\sqrt{2} & 0 \\ 0 & & \\ 0 & & \end{bmatrix}$$

Column $\vec{M}_2$

Since $\vec{M}_2$ is zero, don't need to do anything.

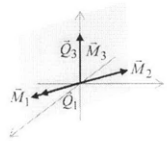
Move on to the next step

$$\begin{array}{c} \begin{bmatrix} \frac{1}{\sqrt{2}} & 0 & 0 \\ -\frac{1}{\sqrt{2}} & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \\ \vec{Q}_1 \quad \vec{Q}_2 \quad \vec{M}_3 \end{array} \begin{bmatrix} \sqrt{2} & -\sqrt{2} & 0 \\ 0 & 0 & 0 \\ 0 & 0 & \end{bmatrix}$$

Column $\vec{M}_3$

$r_{33} = \sqrt{\vec{M}_3 \cdot \vec{M}_3} = 1 \quad \vec{Q}_3 = \frac{1}{r_{33}}\vec{M}_3$

$$\begin{array}{c} \vec{Q}_1 \quad \vec{Q}_2 \quad \vec{Q}_3 \\ \underbrace{\begin{bmatrix} \frac{1}{\sqrt{2}} & 0 & 0 \\ -\frac{1}{\sqrt{2}} & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}}_{\mathbf{Q}} \end{array} \underbrace{\begin{bmatrix} \sqrt{2} & -\sqrt{2} & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}}_{\mathbf{R}}$$

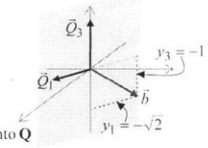


(2) $\vec{y} = \mathbf{Q}^T\vec{b}$

$y_1 = \vec{Q}_1^T\vec{b} = \begin{bmatrix} \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} & 0 \end{bmatrix} \cdot \begin{bmatrix} -1 \\ 1 \\ -1 \end{bmatrix} = -\frac{1}{\sqrt{2}} - \frac{1}{\sqrt{2}} = -\sqrt{2}$

$y_2 = 0$

$y_3 = \vec{Q}_3^T\vec{b} = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} -1 \\ 1 \\ -1 \end{bmatrix} = -1$

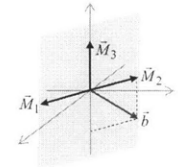


y is the projection of $\vec{b}$ onto $\mathbf{Q}$

(3) $\mathbf{R}\vec{x} = \vec{y}$

$$\begin{bmatrix} \sqrt{2} & -\sqrt{2} & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} -\sqrt{2} \\ 0 \\ -1 \end{bmatrix}$$

$$\Rightarrow \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} -1+a \\ a \\ -1 \end{bmatrix}$$



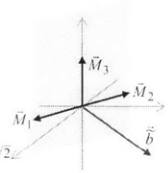
$\vec{b} \in span\{\mathbf{M}\} \rightarrow$

The resulting nodal matrix is SINGULAR, but a solution exists!

→ infinite number of solutions.

If $\tilde{\bar{b}} \notin span\{\mathbf{M}\} \rightarrow$ there is no solution

$$\begin{bmatrix} 1 & -1 & 0 \\ -1 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \overbrace{\begin{bmatrix} 0 \\ 2 \\ -1 \end{bmatrix}}^{\tilde{\bar{b}}}$$



(1) Step (1) stays the same

(2) $\tilde{y} = \mathbf{Q}^T \tilde{\bar{b}}$

$y_1 = Q_1^T \tilde{\bar{b}} = \begin{bmatrix} \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} & 0 \end{bmatrix} \cdot [0 \quad 2 \quad -1] = -\frac{2}{\sqrt{2}} = -\sqrt{2}$

$y_2 = 0$

$y_3 = Q_3^T \tilde{\bar{b}} = [0 \quad 0 \quad 1] \cdot [0 \quad 2 \quad -1] = -1$

$\tilde{y}$ is the same as the above example

(3) $\mathbf{R}\tilde{x} = \tilde{y}$

Since y is the same as the example above, the solution is the same.

This solution will give us the solution $\tilde{b}$, but not $\tilde{\bar{b}}$ (we know this because we choose this $\tilde{\bar{b}}$ outside of span$\{\mathbf{M}\}$ knowing that we cannot get a solution).

What is going on?

We find that $\tilde{b}$ is the projection of $\tilde{\bar{b}}$ onto the subspace of the columns of $\mathbf{M}$.

Minimization View of QR

❖ Alternative Formulations

Definition of the residual, R:

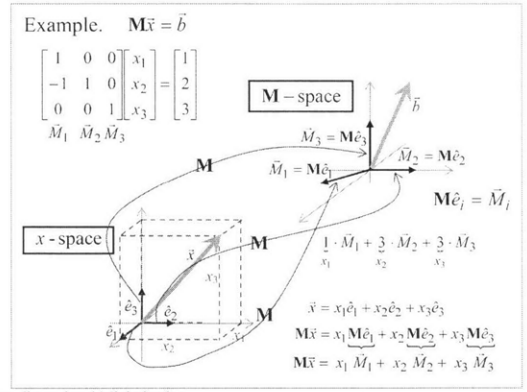
$R(\tilde{x}) \equiv \tilde{b} - \mathbf{M}\tilde{x}$

Find $\tilde{x}$ which satisfies	Minimize over all $\tilde{x}$
$\mathbf{M}\tilde{x} = \tilde{b}$	$\lvert R(\tilde{x}) \rvert^2$

If $\tilde{b} \in range\{\mathbf{M}\}$

Then $\mathbf{M}\tilde{x} = \tilde{b} \Leftrightarrow \min_{\tilde{x}} \lvert R(\tilde{x}) \rvert^2 = 0$

Minimization extends to singular or non-square case!



❖ One-dimensional Minimization

Suppose $\tilde{x} = x_1\tilde{e}_1$ and therefore $\mathbf{M}\tilde{x} = x_1\mathbf{M}\tilde{e}_1 = x_1\tilde{M}_1$

One-dimensional minimization:

$$R(\tilde{x})^2 = R(\tilde{x})^T R(\tilde{x}) = (\tilde{b} - x_1\mathbf{M}\tilde{e}_1)^T (\tilde{b} - x_1\mathbf{M}\tilde{e}_1)$$

$$= \tilde{b}^T\tilde{b} - 2x_1\tilde{b}^T\mathbf{M}\tilde{e}_1 + x_1^2(\mathbf{M}\tilde{e}_1)^T(\mathbf{M}\tilde{e}_1)$$

Note:
$\underbrace{\tilde{b}^T x_1\mathbf{M}\tilde{e}_1}_{\text{scalar}} = \underbrace{(x_1\mathbf{M}\tilde{e}_1)^T\tilde{b}}_{\text{scalar}}$

$$\frac{d}{dx} R(\tilde{x})^T R(\tilde{x}) = -2\tilde{b}^T\mathbf{M}\tilde{e}_1 + 2x_1(\mathbf{M}\tilde{e}_1)^T(\mathbf{M}\tilde{e}_1) = 0$$

$(\mathbf{M}\tilde{e}_1)^T(\mathbf{M}\tilde{e}_1) = \tilde{M}_1^T\tilde{M}_1$

$$x_1 = \frac{b^T\mathbf{M}\tilde{e}_1}{\tilde{e}_1{}^T\mathbf{M}^T\mathbf{M}\tilde{e}_1}$$

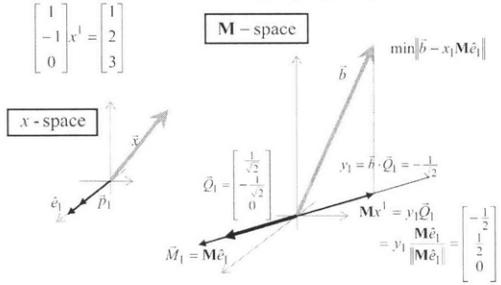
Normalization

Example. $\mathbf{M}\vec{x} = \vec{b}$

$$\begin{bmatrix} 1 & 0 & 0 \\ -1 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix}$$

Let's look for the best solution using only $\vec{M}_1 = \mathbf{M}\vec{e}_1$

$$\begin{bmatrix} 1 \\ -1 \\ 0 \end{bmatrix} x^1 = \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix}$$



(1) Normalize $\mathbf{M}\vec{e}_1 \to \vec{Q}_1$

$$\vec{Q}_1 = \frac{\mathbf{M}\vec{e}_1}{\|\mathbf{M}\vec{e}_1\|} = \begin{bmatrix} \frac{1}{\sqrt{2}} \\ -\frac{1}{\sqrt{2}} \\ 0 \end{bmatrix}$$

M – Normalize $\to \vec{p}_1$

$$\vec{p}_1 = \frac{\vec{e}_1}{\|\mathbf{M}\vec{e}_1\|} = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} \frac{1}{\sqrt{2}} \\ 0 \\ 0 \end{bmatrix}$$

(2) Projection

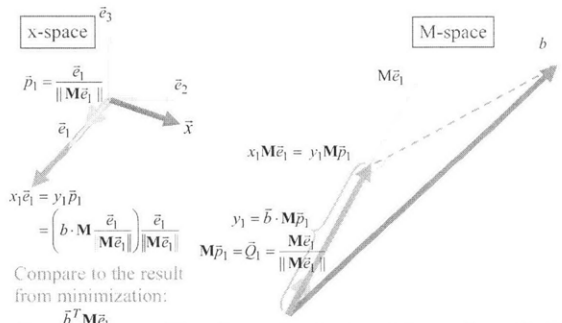
The Closest is

$$\vec{y}_1 = \vec{b} \cdot (\mathbf{M}\vec{p}_1) = \vec{b} \cdot \vec{Q}_1 = -\frac{1}{\sqrt{2}}$$

$$\mathbf{M}\vec{x}^1 = \underbrace{(\vec{b} \cdot \vec{Q}_1)}_{y_1 = -\frac{1}{\sqrt{2}}}\vec{Q}_1 = y_1\mathbf{M}\vec{p}_1 = \begin{bmatrix} -\frac{1}{2} \\ \frac{1}{2} \\ 0 \end{bmatrix}$$

(3) Best Approximation

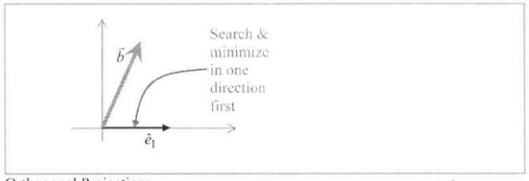
$$\vec{x}^1 = y_1\vec{p}_1 = -\frac{1}{\sqrt{2}} \cdot \vec{p}_1 = -\frac{1}{\sqrt{2}} \begin{bmatrix} \frac{1}{\sqrt{2}} \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} -\frac{1}{2} \\ 0 \\ 0 \end{bmatrix}$$



Compare to the result from minimization:

$$x_1 = \frac{\vec{b}^T\mathbf{M}\vec{e}_1}{\vec{e}_1^T\mathbf{M}^T\mathbf{M}\vec{e}_1}$$

The orthogonal projection minimizes the residual!

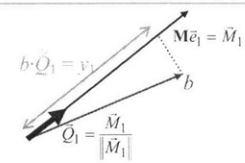


Orthogonal Projection:

$$x_1\vec{M}_1 = y_1\vec{Q}_1 = \left(\vec{b} \cdot \frac{\vec{M}_1}{\|\vec{M}_1\|}\right)\frac{\vec{M}_1}{\|\vec{M}_1\|}$$

$$x_1\vec{M}_1 = \frac{\vec{b}^T\mathbf{M}\vec{e}_1}{\vec{e}_1^T\mathbf{M}^T\mathbf{M}\vec{e}_1}\vec{M}_1$$

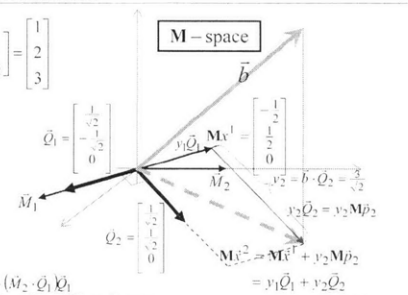
$$x_1 = \frac{\vec{b}^T\mathbf{M}\vec{e}_1}{\vec{e}_1^T\mathbf{M}^T\mathbf{M}\vec{e}_1}$$



One-dimensional minimization yields same result as orthogonal projection on the column!

❖ Two-dimensional Minimization

$$\begin{bmatrix} 1 & 0 \\ -1 & 1 \\ 0 & 0 \end{bmatrix}\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix}$$



$$\vec{M}_2 = \overbrace{\vec{M}_2}^{\mathbf{M}\vec{e}_2} - (\vec{M}_2 \cdot \vec{Q}_1)\vec{Q}_1$$

$$= \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} - \left(\begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} \cdot \begin{bmatrix} \frac{1}{\sqrt{2}} \\ -\frac{1}{\sqrt{2}} \\ 0 \end{bmatrix} \right) \begin{bmatrix} \frac{1}{\sqrt{2}} \\ -\frac{1}{\sqrt{2}} \\ 0 \end{bmatrix} = \begin{bmatrix} \frac{1}{2} \\ \frac{1}{2} \\ 0 \end{bmatrix}$$

$$= \begin{bmatrix} -\frac{1}{2} \\ \frac{1}{2} \\ 0 \end{bmatrix} + \frac{3}{\sqrt{2}} \begin{bmatrix} \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \\ 0 \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \\ 0 \end{bmatrix}$$

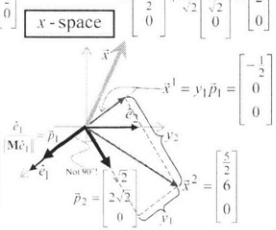
$$\vec{Q}_2 = \frac{\vec{M}_2}{|\vec{M}_2|} = \begin{bmatrix} \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \\ 0 \end{bmatrix}$$

$$\vec{p}_2 = \vec{e}_2 - (\mathbf{M}\vec{e}_2 \cdot \vec{Q}_1)\vec{p}_1$$

$$= \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} - \left(\begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} \cdot \begin{bmatrix} \frac{1}{\sqrt{2}} \\ -\frac{1}{\sqrt{2}} \\ 0 \end{bmatrix} \right) \begin{bmatrix} \frac{1}{\sqrt{2}} \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} \frac{1}{2} \\ 1 \\ 0 \end{bmatrix}$$

$$\vec{p}_2 = \frac{\vec{p}_2}{|\mathbf{M}\vec{p}_2|} = 2\sqrt{2} \begin{bmatrix} \frac{1}{2} \\ 1 \\ 0 \end{bmatrix} = \begin{bmatrix} \sqrt{2} \\ 2\sqrt{2} \\ 0 \end{bmatrix}$$

$$y_2 = \vec{b} \cdot \underbrace{\vec{Q}_2}_{\mathbf{M}\vec{p}_2} = \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix} \cdot \begin{bmatrix} \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \\ 0 \end{bmatrix} = \frac{3}{\sqrt{2}}$$



$$\vec{x}^2 = \vec{x}^1 + y_2\vec{p}_2 = \begin{bmatrix} -\frac{1}{2} \\ 0 \\ 0 \end{bmatrix} + \frac{3}{\sqrt{2}} \begin{bmatrix} \sqrt{2} \\ 2\sqrt{2} \\ 0 \end{bmatrix} = \begin{bmatrix} \frac{5}{2} \\ 6 \\ 0 \end{bmatrix}$$

Now $\vec{x} = x_1\vec{e}_1 + x_2\vec{e}_2$ and $\mathbf{M}\vec{x} = x_1\mathbf{M}\vec{e}_1 + x_2\mathbf{M}\vec{e}_2$

Residual Minimization

$$\|R(\vec{x})\|^2 = R(\vec{x})^T R(\vec{x}) = \left(\vec{b} - x_1\mathbf{M}\vec{e}_1 - x_2\mathbf{M}\vec{e}_2\right)^T \left(\vec{b} - x_1\mathbf{M}\vec{e}_1 - x_2\mathbf{M}\vec{e}_2\right)$$

$$= \vec{b}^T\vec{b} - 2x_1\vec{b}^T\mathbf{M}\vec{e}_1 + x_1^2(\mathbf{M}\vec{e}_1)^T(\mathbf{M}\vec{e}_1) - 2x_2\vec{b}^T\mathbf{M}\vec{e}_2 + x_2^2(\mathbf{M}\vec{e}_2)^T(\mathbf{M}\vec{e}_2)$$

$$+ \boxed{2x_1x_2(\mathbf{M}\vec{e}_1)^T(\mathbf{M}\vec{e}_2)} \Rightarrow \textit{coupling term}$$

Minimization is difficult because of the coupling term.

More general search directions

$x = y_1\vec{p}_1 + y_2\vec{p}_2$ such that:

$\mathbf{M}x = y_1\mathbf{M}\vec{p}_1 + y_2\mathbf{M}\vec{p}_2 \qquad span\{\vec{p}_1, \vec{p}_2\} = span\{\vec{e}_1, \vec{e}_2\}$

$$\mathbf{R}(\vec{x})^T\mathbf{R}(\vec{x}) = \vec{b}^T\vec{b} - 2y_1\vec{b}^T\mathbf{M}\vec{p}_1 + y_1^2(\mathbf{M}\vec{p}_1)^T(\mathbf{M}\vec{p}_1)$$

$$- 2y_2\vec{b}^T\mathbf{M}\vec{p}_2 + y_2^2(\mathbf{M}\vec{p}_2)^T(\mathbf{M}\vec{p}_2) + \boxed{2y_1y_2(\mathbf{M}\vec{p}_1)^T(\mathbf{M}\vec{p}_2)}$$

If $\vec{p}_1^T\mathbf{M}^T\mathbf{M}\vec{p}_2 = 0$ Minimizations Decouple!!! *coupling term*

If search directions are M-orthonormal

$$\underbrace{\mathbf{M}\vec{p}_i}_{\vec{Q}_i} \cdot \underbrace{\mathbf{M}\vec{p}_j}_{\vec{Q}_j} = \begin{cases} 0 & i \neq j \\ 1 & i = j \end{cases}$$

Decoupled minimizations can be done individually

$$\text{Minimize:} \quad -2y_i\left(\vec{b}^T\mathbf{M}\vec{p}_i\right) + y_i^2(\mathbf{M}\vec{p}_i)^T(\mathbf{M}\vec{p}_i)$$

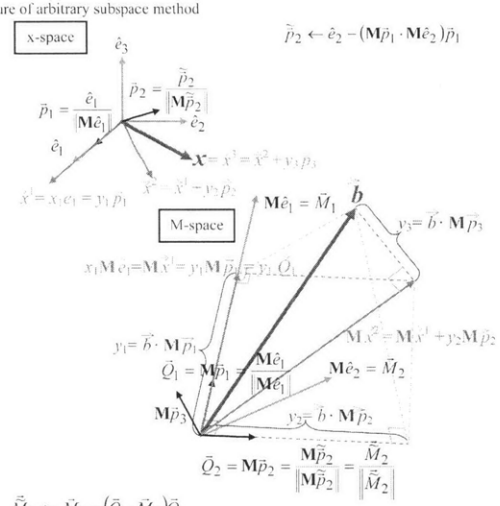
$$-2y_i\left(\vec{b} \cdot \vec{Q}_i\right) + y_i^2\underbrace{\left(\vec{Q}_i \cdot \vec{Q}_i\right)}_{1}$$

$$\text{Differentiating:} \quad -2\left(\vec{b} \cdot \vec{Q}_i\right) + 2y_i = 0 \Rightarrow y_i = \vec{b} \cdot \vec{Q}_i$$

Minimization yields same result as orthogonal projection!

❖ Picture of arbitrary subspace method

$$\tilde{p}_2 \leftarrow \hat{e}_2 - (\mathbf{M}\vec{p}_1 \cdot \mathbf{M}\hat{e}_2)\vec{p}_1$$



$$\tilde{M}_2 \leftarrow \vec{M}_2 - (\vec{Q}_1 \cdot \vec{M}_2)\vec{Q}_1$$

$$\mathbf{M}\tilde{p}_2 \leftarrow \mathbf{M}\hat{e}_2 - (\mathbf{M}\vec{p}_1 \cdot \mathbf{M}\hat{e}_2)\mathbf{M}\vec{p}_1$$

$$\{\hat{e}_1, \hat{e}_2, \ldots, \hat{e}_i\}$$

$$\{\vec{p}_1, \vec{p}_2, \ldots, \vec{p}_i\}$$

$$\left\{ \overbrace{\mathbf{M}\vec{p}_1}^{\vec{Q}_1}, \overbrace{\mathbf{M}\vec{p}_2}^{\vec{Q}_2}, \ldots, \mathbf{M}\vec{p}_i \right\}$$

Orthonormal

$$\vec{x}^i = \underbrace{(\vec{b} \cdot \mathbf{M}\vec{p}_1)}_{y_1}\vec{p}_1 + \ldots + \underbrace{(\vec{b} \cdot \mathbf{M}\vec{p}_i)}_{y_i}\vec{p}_i$$

$$\vec{x}^{i+1} = \underbrace{(\vec{b} \cdot \mathbf{M}\vec{p}_1)\vec{p}_1 + \ldots + (\vec{b} \cdot \mathbf{M}\vec{p}_i)\vec{p}_i}_{x^i} + (\vec{b} \cdot \mathbf{M}\vec{p}_{i+1})\vec{p}_{i+1}$$

Forming M-orthonormal minimization directions

The ith search direction equals **M**-orthonormalized unit vector

***x*-space**	***M*-space**
$\tilde{\vec{p}}_i \leftarrow \vec{e}_i - \sum_{j=1}^{i-1} r_{ji}\vec{p}_j$	$\mathbf{M}\tilde{\vec{p}}_i \leftarrow \mathbf{M}\vec{e}_i - \sum_{j=1}^{i-1} r_{ji}\underbrace{\mathbf{M}\vec{p}_j}_{\vec{Q}_j}$
	$r_{ji} = \underbrace{(\mathbf{M}\vec{p}_j)}_{\vec{Q}_j} \cdot (\mathbf{M}\vec{e}_i)$
$\vec{p}_i \leftarrow \frac{\tilde{\vec{p}}_i}{\|\mathbf{M}\tilde{\vec{p}}_i\|}$	$\underbrace{\mathbf{M}\vec{p}_i}_{\vec{Q}_i} \leftarrow \frac{\mathbf{M}\tilde{\vec{p}}_i}{\|\mathbf{M}\tilde{\vec{p}}_i\|}$ r_{ii}

❖ Minimization Algorithm

For $i = 1$ to N — "For each Target Column"

$\vec{p}_i \leftarrow \vec{e}_i$

For $j = 1$ to $i - 1$ — "For each Source Column left of target"

$$\left.\begin{aligned} r_{ji} &\leftarrow \underbrace{\vec{Q}_j}_{\mathbf{M}\vec{p}_j} \cdot \underbrace{\vec{M}_i}_{\mathbf{M}\vec{p}_i} \\ \vec{p}_i &\leftarrow \vec{p}_i - r_{ji}\vec{p}_j \end{aligned}\right\} \text{M - Orthogonalize Search Direction}$$

$$\left.\begin{aligned} r_{ii} &\leftarrow \left\| \underbrace{\vec{Q}_i}_{\mathbf{M}\vec{p}_i} \right\| \\ \vec{p}_i &\leftarrow \frac{\vec{p}_i}{r_{ii}} \end{aligned}\right\} \text{M - Normalize Search Direction}$$

$y_i \leftarrow b \cdot \underbrace{(\vec{Q}_i)}_{\mathbf{M}\vec{p}_i}$ — Calculate projection on new direction

$x \leftarrow x + y_i\vec{p}_i$ — Update the solution

i directions

$$\{\hat{e}_1, \hat{e}_2, \ldots, \hat{e}_i\}$$

$$\{\vec{p}_1, \vec{p}_2, \ldots, \vec{p}_i\} \qquad \underset{\text{orthonormal}}{\{\mathbf{M}\vec{p}_1, \mathbf{M}\vec{p}_2, \ldots, \mathbf{M}\vec{p}_i\}}$$

$$\vec{x}^i = \underbrace{(\vec{b} \cdot \mathbf{M}\vec{p}_1)}_{y_1}\vec{p}_1 + \ldots + \underbrace{(\vec{b} \cdot \mathbf{M}\vec{p}_i)}_{y_i}\vec{p}_i$$

i+1 directions

$$\{\hat{e}_1, \hat{e}_2, \ldots, \hat{e}_i, \hat{e}_{i+1}\}$$

$$\{\vec{p}_1, \vec{p}_2, \ldots, \vec{p}_i, \vec{p}_{i+1}\} \qquad \underset{\text{orthonormal}}{\{\mathbf{M}\vec{p}_1, \mathbf{M}\vec{p}_2, \ldots, \mathbf{M}\vec{p}_i, \mathbf{M}\vec{p}_{i+1}\}}$$

$$\vec{x}^{i+1} = \underbrace{\underbrace{(\vec{b} \cdot \mathbf{M}\vec{p}_1)}_{y_1}\vec{p}_1 + \ldots + \underbrace{(\vec{b} \cdot \mathbf{M}\vec{p}_i)}_{y_i}\vec{p}_i}_{\vec{x}^i} + \underbrace{(\vec{b} \cdot \mathbf{M}\vec{p}_{i+1})}_{y_{i+1}}\vec{p}_{i+1}$$

❖ Comparison – Minimization and QR

$$\left.\vec{Q}_1 \quad \vec{Q}_2 \cdots\cdots\cdots \vec{Q}_N\right\} \text{Orthonormal}$$

$$\uparrow \mathbf{M} \qquad \uparrow \mathbf{M} \qquad \uparrow \mathbf{M}$$

$$\left.\underbrace{\frac{1}{r_{11}}\vec{e}_1}_{\vec{p}_1} \quad \underbrace{\frac{1}{r_{22}}(\vec{e}_2 - r_{12}\vec{e}_1)}_{\vec{p}_2} \cdots\cdots \underbrace{\frac{1}{r_{NN}}\left(\vec{e}_N - \sum r_{iN}\vec{e}_i\right)}_{\vec{p}_N}\right\} \mathbf{M}^T\mathbf{M} \text{ Orthonormal}$$

$$\begin{bmatrix} & & \\ & I & \\ & & \end{bmatrix}\begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix} = \vec{b} \qquad \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix}$$

exact solution

Solve this – choose the first search direction to be:

- $\hat{e}_1$ $\quad \vec{x}^1 = x_1\hat{e}_1 = b_1\hat{e}_1 \rightarrow \begin{bmatrix} b_1 \\ 0 \\ \vdots \\ 0 \end{bmatrix}$ first step
- $\hat{e}_{12}$ $\quad \vec{x}^1 = x_1\hat{e}_{12} = b_{12}\hat{e}_{12} \rightarrow \begin{bmatrix} 0 \\ \vdots \\ 0 \\ b_{12} \\ 0 \\ \vdots \\ 0 \end{bmatrix}$ first step
- $\vec{b}$ $\quad \vec{x}^1 = \underbrace{v_1}_{1}\vec{b} \rightarrow \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix}$ first step

search direction

❖ Search Direction

Orthogonalized unit vectors → search directions

$$\underbrace{\{\vec{e}_1, \vec{e}_2, \ldots, \vec{e}_N\}}_{\text{unit vectors}} \xrightarrow[\text{Orthogonalization}]{\mathbf{M}^T\mathbf{M}} \underbrace{\{\vec{p}_1, \vec{p}_2, \ldots, \vec{p}_N\}}_{\text{search directions}}$$

Example. Simple Problem.

$$\begin{bmatrix} 1 & & & \\ & 1 & & \\ & & \ddots & \\ & & & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_N \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_N \end{bmatrix}$$

Step 1. $\vec{x}^1 = \begin{bmatrix} b_1 \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}$ Step 2. $\vec{x}^2 = \begin{bmatrix} b_1 \\ b_2 \\ 0 \\ \vdots \\ 0 \end{bmatrix}$ • • • • Step N. $\vec{x}^N = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ \vdots \\ b_N \end{bmatrix}$

If I am searching along $\mathbf{M}\vec{p}_1 = \vec{e}_1$ the best I can do is $\vec{b} - \mathbf{M}\vec{p}_1 = \vec{x}^1$

$$\vec{e}_1, \vec{e}_2, \ldots \vec{e}_N \xrightarrow{\mathbf{M}} \vec{e}_1, \vec{e}_2, \ldots \vec{e}_N$$

$$\downarrow \qquad \downarrow$$

$$\mathbf{M}\vec{p}_1 \qquad \mathbf{M}\vec{p}_N$$

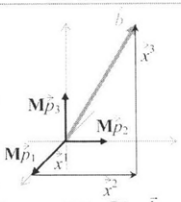
Already M-orthonormal

OR needs N steps!

What about using other sets of starting vectors?

$$\underbrace{\{\vec{b}, \mathbf{M}\vec{b}, \mathbf{M}^2\vec{b}, \ldots\}}_{\text{Krylov-Subspace}} \xrightarrow[\text{Orthogonalization}]{\text{M}^t\text{M}} \underbrace{\{\vec{p}_1, \vec{p}_2, \ldots, \vec{p}_N\}}_{\text{search directions}}$$

Example. $\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix}$



$\vec{e}_1 \leftarrow \mathbf{M}\hat{e}_1$ $\quad \vec{e}_2 \leftarrow \mathbf{M}\hat{e}_2$

$\vec{p}_1 = \hat{e}_1$ $\quad \vec{x}^1 = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}$ $\quad \vec{x}^2 = \begin{bmatrix} 1 \\ 2 \\ 0 \end{bmatrix}$

What if this is the problem:

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 300 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix}$$

In general, for $\mathbf{I}\vec{x} = \vec{b}$ it will take N iterations to get to the solution using this method

If $\hat{e}_3$ is chosen as the first search direction, what is the result?

Example. $\begin{bmatrix} 1 & & & & \\ & 1 & & & \\ & & 1 & & \\ & & & 1 & \\ & & & & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{bmatrix} = \underbrace{\begin{bmatrix} 7 \\ 23 \\ 3 \\ -2 \\ 1 \end{bmatrix}}_{\vec{b}}$

$\vec{x}^1 = \begin{bmatrix} 7 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$ $\quad \vec{x}^2 = \begin{bmatrix} 7 \\ 23 \\ 0 \\ 0 \\ 0 \end{bmatrix}$ $\quad \vec{x}^3 = \begin{bmatrix} 7 \\ 23 \\ 3 \\ 0 \\ 0 \end{bmatrix}$ $\quad \vec{x}^4 = \begin{bmatrix} 7 \\ 23 \\ 3 \\ -2 \\ 0 \end{bmatrix}$ $\quad \vec{x}^5 = \begin{bmatrix} 7 \\ 23 \\ 3 \\ -2 \\ 1 \end{bmatrix}$

$\vec{M}_i = \vec{Q}_i = \vec{p}_i$ $\quad$ QR takes n steps

What if $\vec{b} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 23 \end{bmatrix}$ $\quad \underbrace{\vec{x}^1 = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \cdots\cdots \vec{x}^4 = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}}_{\text{huge error}}$ $\quad \vec{x}^5 = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 23 \end{bmatrix}$

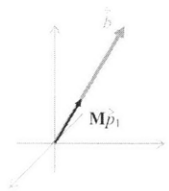
Solution: take $\vec{x}^1 = \vec{b}$

Example.

$$\begin{bmatrix}1&0&0\\0&1&0\\0&0&1\end{bmatrix}\begin{bmatrix}x_1\\x_2\\x_3\end{bmatrix}=\begin{bmatrix}1\\2\\3\end{bmatrix}$$

What is the best initial search direction?

$$\boxed{\vec{b}=\begin{bmatrix}1\\2\\3\end{bmatrix}}$$



$$\vec{p}_1=\begin{bmatrix}1\\2\\3\end{bmatrix}\frac{1}{\left\|\mathbf{M}\begin{bmatrix}1\\2\\3\end{bmatrix}\right\|}$$

x-space

$\mathbf{M}$ is the identity, so

$$\mathbf{M}\vec{b}=\vec{b}$$

$$\Rightarrow \vec{p}_1=\frac{\vec{b}}{\|\vec{b}\|}$$

$$y_1=\vec{b}\cdot\mathbf{M}\vec{p}_1=\|\vec{b}\|$$

$$y_1\cdot\mathbf{M}\vec{p}_1=\|\vec{b}\|\cdot\frac{\vec{b}}{\|\vec{b}\|}=\vec{b}$$

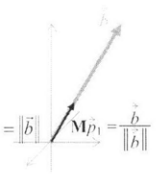
Try $\{\vec{b},\ldots\}$

$$\vec{b}\xrightarrow{\mathbf{M}}\mathbf{M}\vec{b}=\vec{b}$$

Normalize

$$\vec{p}_1=\frac{\vec{b}}{\|\mathbf{M}\vec{b}\|}=\frac{\vec{b}}{\|\vec{b}\|}\xrightarrow{\mathbf{M}}\frac{\mathbf{M}\vec{b}}{\|\mathbf{M}\vec{b}\|}=\frac{\vec{b}}{\|\vec{b}\|}$$

$$y_1=\vec{b}\cdot\mathbf{M}\vec{p}_1=\|\vec{b}\|$$



$$\vec{p}_1=y_1\mathbf{M}\vec{p}_1$$

$$=\|\vec{b}\|\frac{I\cdot\vec{b}}{\|\vec{b}\|}=\vec{b}$$

One Step

Use QR to solve
$\mathbf{M}x=b$ $\mathbf{M}$ is singular

x minimizes $(b-\mathbf{M}x)^T(b-\mathbf{M}x)$

$\mathbf{M}z=0$

z nonzero exists if $\mathbf{M}$ is singular

ARBITRARY SUBSPACE ALGORITHM

$$\underbrace{\begin{bmatrix}1&0&0&\cdots\\0&1&0&\cdots\\0&0&1&\cdots\\0&0&0&\cdots\\\vdots&\vdots&\vdots&\\0&0&0&\cdots\end{bmatrix}}_{\text{QR Algorithm Search Directions}}$$

Approach to Approximately Solving $\mathbf{M}x=b$

Pick a k-dimensional subspace $\Rightarrow \left\{\begin{bmatrix}w_{0_1}\\\vdots\\w_{0_N}\end{bmatrix},\cdots,\begin{bmatrix}w_{k-1_1}\\\vdots\\w_{k-1_N}\end{bmatrix}\right\}\equiv\{\vec{w}_0,\ldots,\vec{w}_{k-1}\}$

Approximate $\vec{x}^k$ as a weighted sum of $\{\vec{w}_0,\ldots,\vec{w}_{k-1}\}\Rightarrow\vec{x}^k=\sum_{i=0}^{k-1}\alpha_i\vec{w}_i$

The residual is defined as $\vec{r}^k\equiv\vec{b}-\mathbf{M}\vec{x}^k$

If $\vec{x}^k=\sum_{i=0}^{k-1}\alpha_i\vec{w}_i\Rightarrow\vec{r}^k=\vec{b}-\mathbf{M}\vec{x}^k=\vec{b}-\sum_{i=0}^{k-1}\alpha_i\mathbf{M}\vec{w}_i$

Residual Minimizing Idea:

Pick α_i's to minimize

$$\|\vec{r}^k\|_2^2\equiv(\vec{r}^k)^T(\vec{r}^k)=\left(\vec{b}-\sum_{i=0}^{k-1}\alpha_i\mathbf{M}\vec{w}_i\right)^T\left(\vec{b}-\sum_{i=0}^{k-1}\alpha_i\mathbf{M}\vec{w}_i\right)$$

Computational Approach

Minimizing $\|\vec{r}^k\|_2^2=\left\|\vec{b}-\sum_{i=0}^{k-1}\alpha_i\mathbf{M}\vec{w}_i\right\|_2^2$ is easy if $\mathbf{M}\vec{w}_i$ were orthonormal!

Create a set of vectors $\{\vec{p}_0,\vec{p}_1,\ldots,\vec{p}_{k-1}\}$ such that

$span\{\vec{p}_0,\ldots,\vec{p}_{k-1}\}=span\{\vec{w}_0,\ldots,\vec{w}_{k-1}\}$

$$\mathbf{M}\vec{p}_i\cdot\mathbf{M}\vec{p}_j=\begin{cases}0 & i\neq j\\1 & i=j\end{cases}$$

Gram-Schmidt on $\mathbf{M}\vec{w}_i$'s

$\vec{w}_0 \to \vec{p}_0$

$\vec{w}_1 \to \vec{p}_1$ want $(\mathbf{M}\vec{p}_1)\cdot(\mathbf{M}\vec{p}_0)=0$

$$\vec{p}_1 = \vec{w}_1 - \alpha\vec{w}_0 = \vec{w}_1 - \alpha\vec{p}_0$$

$$\Rightarrow (\mathbf{M}\vec{w}_1 - \mathbf{M}\alpha\vec{p}_0)^T(\mathbf{M}\vec{p}_0)=0$$

$$\Rightarrow \underbrace{(\mathbf{M}\vec{w}_1)^T(\mathbf{M}\vec{p}_0)}_{\text{if } \vec{w}_1 \perp \vec{p}_0 \Rightarrow 0} = \alpha\underbrace{(\mathbf{M}\vec{p}_0)^T(\mathbf{M}\vec{p}_0)}_{1}$$

Tie this back to the QR algorithm.

$$\begin{bmatrix}\vec{M}_1 & \cdots & \vec{M}_N\end{bmatrix} \Rightarrow \underbrace{\begin{bmatrix}\mathbf{M}\hat{e}_1 & \cdots & \mathbf{M}\hat{e}_N\end{bmatrix}}_{\text{columns orthogonal}}$$

$$= \begin{bmatrix}\vec{Q}_1 & \cdots & \vec{Q}_N\end{bmatrix}\begin{bmatrix}\mathbf{R}\end{bmatrix}$$

M-space

$$\mathbf{M}\tilde{\vec{p}}_i \leftarrow \mathbf{M}\vec{w}_i - \sum_{j=0}^{i-1}\underbrace{\left(\overbrace{\mathbf{M}\vec{p}_j}^{\vec{Q}_j}\cdot\mathbf{M}\vec{w}_i\right)}_{r_{ij}}\underbrace{\mathbf{M}\vec{p}_j}_{\vec{Q}_j}$$

x-space

$$\tilde{\vec{p}}_i \leftarrow \vec{w}_i - \sum_{j=0}^{i-1}\left(\mathbf{M}\vec{p}_j\cdot\mathbf{M}\vec{w}_i\right)\vec{p}_j$$

Arbitrary Subspace Solution Algorithm

Given $\mathbf{M}, \vec{b}$ and a set of search directions $\{\vec{w}_0, \vec{w}_1, \ldots, \vec{w}_k\}$

1) orthogonalize the $\mathbf{M}\vec{w}_i$'s

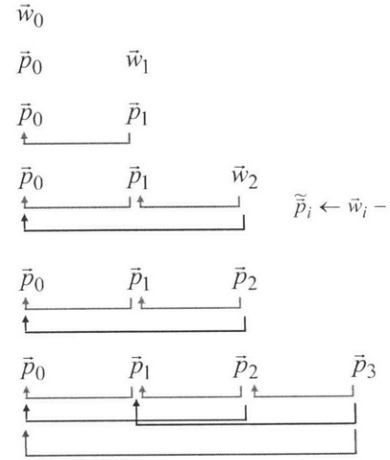
for $i = 1$ to k $\quad \vec{p}_k = \vec{w}_k - \sum_{i=0}^{k-1}\frac{(\mathbf{M}\vec{w}_k)^T(\mathbf{M}\vec{p}_i)}{(\mathbf{M}\vec{p}_i)^T(\mathbf{M}\vec{p}_i)}\vec{p}_i$

2) compute the $\vec{r}$ minimizing solution $\vec{x}^{k+1}$

$$\vec{x}^{k+1} = \sum_{i=0}^{k}\frac{(\vec{r}^{\,0})^T(\mathbf{M}\vec{p}_i)}{(\mathbf{M}\vec{p}_i)^T(\mathbf{M}\vec{p}_i)}\vec{p}_i$$

Orthogonalization of Search Directions

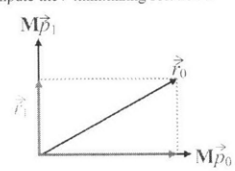
M - orthonormalizing $\vec{w}_i$



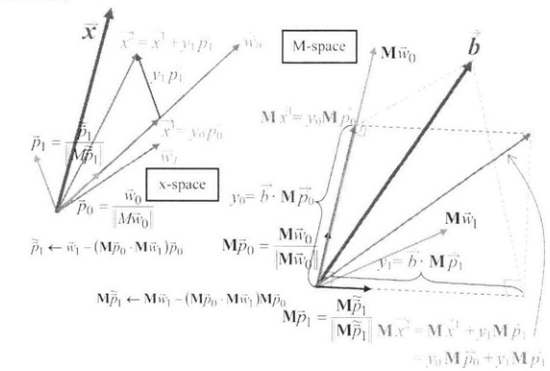
$$\tilde{\vec{p}}_i \leftarrow \vec{w}_i - \sum_{j=0}^{i-1}\left(M\vec{p}_j\cdot M\vec{w}_i\right)\vec{p}_j$$

$$\tilde{\vec{p}}_i \leftarrow \vec{w}_i - \sum_{j=0}^{i-1}\left(M\vec{p}_j\cdot M\vec{w}_i\right)\vec{p}_j \qquad \vec{p}_i \leftarrow \frac{\tilde{\vec{p}}_i}{\|M\tilde{\vec{p}}_i\|}$$

Compute the $\vec{r}$ minimizing solution $\vec{x}^{k+1}$



The Picture



Arbitrary Subspace Generation

If $span\{\vec{w}_0, \vec{w}_1, \ldots, \vec{w}_k\} = span\{\vec{r}^0, \mathbf{M}\vec{r}^0, \ldots, \mathbf{M}^k\vec{r}^0\}$

$$\vec{x}^{k+1} = \sum_{i=0}^{k} \alpha_i \mathbf{M}^i \vec{r}^0 = \underbrace{\xi_k(\mathbf{M})}_{k^{th}\text{ order polynomial}} \vec{r}^0$$

$$\vec{r}^{k+1} = \vec{r}^0 - \sum_{i=0}^{k} \alpha_i \mathbf{M}^{i+1} \vec{r}^0 = (\mathbf{I} - \mathbf{M}\xi_k(\mathbf{M}))\vec{r}^0$$

Note: for any $\alpha_0 \neq 0$

$$span\{\vec{r}^0, \vec{r}^1 = \vec{r}^0 - \alpha_0 \mathbf{M}\vec{r}^0\} = span\{\vec{r}^0, \mathbf{M}\vec{r}^0\}$$

$$\vec{x}^3 = \alpha_1 \vec{b} + \alpha_2 \mathbf{M}\vec{b} + \alpha_3 \mathbf{M}^2 \vec{b}$$

$$= \underbrace{(\alpha_1 \mathbf{I} + \alpha_2 \mathbf{M} + \alpha_3 \mathbf{M}^2)}_{\text{polynomial}} \vec{b}$$

Krylov Subspace Steps:

(1) Pick a Search Direction
(2) Make search direction $\mathbf{M}^T\mathbf{M}$ orthogonal
(3) Update x (minimize residual)
(4) Update residual

$$\underbrace{\begin{aligned} \vec{r}^0 &= \vec{b} \\ \vec{p}^0 &= \vec{r}^0 \\ \vec{x}^0 &= \alpha_0 \vec{p}^0 \end{aligned}}_{\text{First Pass}}$$

Algorithm

For $k = 0$ to $N - 1$

$\vec{p}_i \leftarrow \vec{w}_i$

The algorithm is the same even if I use random search directions.

For $j = 1$ to $k - 1$

$\vec{p}_k \leftarrow \vec{p}_k - (\mathbf{M}\vec{p}_j \cdot \mathbf{M}\vec{p}_k)\vec{p}_j$ — M - Orthogonalize Search Direction

$\vec{p}_k \leftarrow \frac{\vec{p}_k}{\|M\vec{p}_k\|}$ — M - Normalize search direction

$y_k \leftarrow \vec{b} \cdot (\mathbf{M}\vec{p}_k)$ — Calculate projection on new direction

$\vec{x}^{k+1} \leftarrow \vec{x}^k + y_k \vec{p}_k$ — Update the solution

After i^{th} iteration:

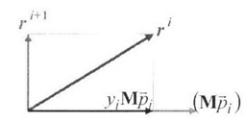
$$\vec{b} = y_0 \mathbf{M}\vec{p}_0 + \ldots + y_{i-1}\mathbf{M}\vec{p}_{i-1} + \vec{r}^i$$

residual

Since they are all orthogonal to $(\mathbf{M}\vec{p}_i) \Rightarrow y_i = \vec{b} \cdot (\mathbf{M}\vec{p}_i) = \vec{r}^i \cdot (\mathbf{M}\vec{p}_i)$

After $i + 1$ iterations:

$$\vec{b} = y_0 \mathbf{M}\vec{p}_0 + \ldots + y_{i-1}\mathbf{M}\vec{p}_{i-1} + \underbrace{y_i \mathbf{M}\vec{p}_i + \vec{r}^{i+1}}_{\vec{r}_i}$$

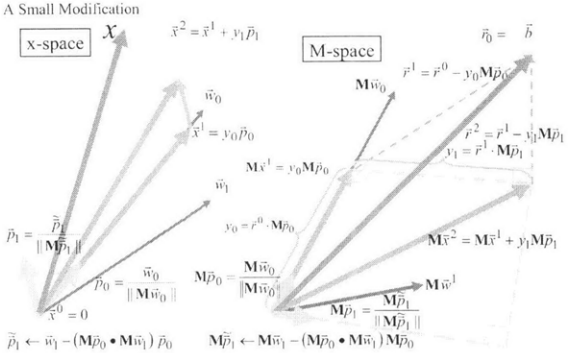


Residual Formula.

$$\vec{x}^{j+1} = \vec{x}^j + \alpha_j \vec{p}_j$$

$$\vec{r}^{j+1} = \vec{b} - \mathbf{M}x^{j+1} = \vec{b} - \mathbf{M}\left(x^j + \alpha_j \vec{p}_j\right) = \underbrace{\vec{b} - \mathbf{M}x^j}_{r^j} - \mathbf{M}\alpha_j \vec{p}_j$$

A Small Modification



$\tilde{\vec{p}}_1 \leftarrow \vec{w}_1 - (\mathbf{M}\vec{p}_0 \bullet \mathbf{M}\vec{w}_1)\,\vec{p}_0$ $\qquad$ $\mathbf{M}\tilde{\vec{p}}_1 \leftarrow \mathbf{M}\vec{w}_1 - (\mathbf{M}\vec{p}_0 \bullet \mathbf{M}\vec{w}_1)\,\mathbf{M}\vec{p}_0$

Algorithm

For $i = 1$ to N

$\vec{p}_i \leftarrow \vec{w}_i$

For $j = 1$ to $i - 1$

$\vec{p}_i \leftarrow \vec{p}_i - (\mathbf{M}\vec{p}_j \cdot \mathbf{M}\vec{p}_i)\vec{p}_j\}$ $\qquad$ M - Orthogonalize Search Direction

$\vec{p}_i \leftarrow \left.\frac{\vec{p}_i}{\|\mathbf{M}\vec{p}_i\|}\right\}$ $\qquad$ M - Normalize search direction

$y_i \leftarrow \vec{r}^i\,(\mathbf{M}\vec{p}_i)$ $\qquad$ Calculate projection on new direction

$\vec{x}^{i+1} \leftarrow \vec{x}^i + y_i\vec{p}_i$ $\qquad$ Update the solution

$\vec{r}^{i+1} \leftarrow \vec{r}^i - y_i\mathbf{M}\vec{p}_i$ $\qquad$ Update the residual

Vector Span.

$$\vec{x} = \alpha_1\begin{bmatrix}1\\1\end{bmatrix} + \alpha_2\begin{bmatrix}0\\1\end{bmatrix} \qquad \vec{x} = \tilde{\alpha}_1\begin{bmatrix}1\\0\end{bmatrix} + \tilde{\alpha}_2\begin{bmatrix}0\\1\end{bmatrix}$$

Span the same space!

Any weighted combination you can get with one set, you can get with the other set and a different set of weights.

Generalized Conjugate Residual Algorithm

Solve $\mathbf{M}\vec{x} = \vec{b}$

Too expensive!

Best selection:

$\vec{x} = \alpha_0\vec{w}_0 \longrightarrow \vec{w}_0 = \mathbf{M}^{-1}\vec{b}$

Selection of the Search Directions

Criteria for selecting $\vec{w}_0, \ldots, \vec{w}_{k-1}$

All that matters is the $span\{\vec{w}_0, \ldots \vec{w}_{k-1}\}$

$\exists\, \alpha_i$'s such that $\vec{b} - \mathbf{M}\vec{x}^k = \vec{b} - \sum_{i=0}^{k-1}\alpha_i\mathbf{M}\vec{w}_i$ is small

$\mathbf{M}^{-1}\vec{b} \approx$ in the $span\{\vec{w}_0, \ldots, \vec{w}_{k-1}\}$ for $k < N$

One choice, unit vectors, $\vec{x}^k \in span\{\vec{e}_1, \ldots \vec{e}_k\}$

Generates the QR algorithm if $k = N$

Can be terrible if $k < N$

Example.

$$\begin{bmatrix} 1 & & 0 \\ & \ddots & \\ 0 & & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ \vdots \\ x_N \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \qquad \vec{w}'s = \left\{ \begin{bmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix}, \cdots \right\}$$

Only need (1) $\vec{w}$ to solve this problem → the first one

$$\begin{bmatrix} 1 & & 0 \\ & \ddots & \\ 0 & & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ \vdots \\ x_N \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 1 \end{bmatrix}$$

If you start from the first search direction, this will take all N search directions to find the correct answer.

Only need (1) $\vec{w}$ to solve this problem → the last one

$$\begin{bmatrix} 1 & & \\ & \ddots & \\ & & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ \vdots \\ x_N \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix}$$

Need all N $\vec{w}$'s to solve this problem

Could also solve this problem with ONE search direction if that search direction is all ones.

Historical Development

Assume $\mathbf{M} = \mathbf{M}^T$ (symmetric) and $\vec{x}^T \mathbf{M} \vec{x} > 0$ (positive definite)

Then the solution of $\mathbf{M}\vec{x} = \vec{b}$ corresponds to the location of the minimum of $f(\vec{x}) = \frac{1}{2}\vec{x}^T \mathbf{M}\vec{x} - \vec{x}^T \vec{b}$

$\nabla_x f(\vec{x}) = \mathbf{M}\vec{x} - \vec{b} \Rightarrow \vec{x} = \mathbf{M}^{-1}\vec{b}$ minimizes f

Gradient of $f(x)$ = Residual at x

Idea: search along the gradient i.e. the steepest descent directions for $f(x)$ i.e. along the current residual direction.

Pick $span\{\vec{w}_0, \ldots, \vec{w}_{k-1}\} = span\{\nabla_x f(\vec{x}^0), \ldots, \nabla_x f(\vec{x}^{k-1})\} = span\{\vec{r}^0, \vec{r}^1, \ldots, \vec{r}^{k-1}\}$

Does not extend to non-symmetric, non positive definite case.

Krylov Subspace

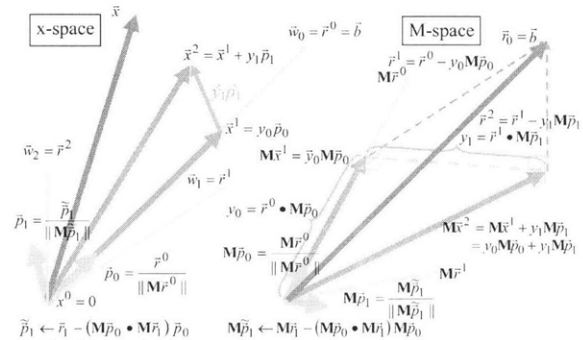
Note : $span\{\nabla_x f(\vec{x}^0), \ldots \nabla_x f(\vec{x}^{k-1})\} = span\{\vec{r}^0, \ldots, \vec{r}^{k-1}\}$

If $span\{\vec{w}_0, \ldots, \vec{w}_{k-1}\} = span\{\vec{r}^0, \ldots, \vec{r}^{k-1}\}$

Then $\vec{r}^k = \vec{r}^0 - \sum_{i=0}^{k-1} \alpha_i \mathbf{M}\vec{r}^i$

and $span\{\vec{r}^0, \ldots, \vec{r}^{k-1}\} = \underbrace{span\{\vec{r}^0, \mathbf{M}\vec{r}^0, \ldots, \mathbf{M}^{k-1}\vec{r}^0\}}_{\text{Krylov Subspace}}$

The Picture



Computational Approach

$\vec{r}^0 = \vec{b} - \mathbf{M}\vec{x}^0 = \vec{b}$

REPEAT

$\vec{p}_k \leftarrow \vec{r}^k$ — Pick next Search Direction

For $j = 1$ to $k-1$

$\vec{p}_k \leftarrow \vec{p}_k - (\mathbf{M}\vec{p}_j \cdot \mathbf{M}\vec{p}_k)\vec{p}_j$ — M-Orthogonalize Search Direction

$\vec{p}_k \leftarrow \frac{\vec{p}_k}{\|\mathbf{M}\vec{p}_k\|}$ — M - Normalize search direction

$y_k \leftarrow \vec{r}^k \cdot (\mathbf{M}\vec{p}_k)$ — Optimal step size in new direction

$\vec{x}^{k+1} \leftarrow \vec{x}^k + y_k\vec{p}_k$ — Update the solution

$\vec{r}^{k+1} \leftarrow \vec{r}^k - y_k\mathbf{M}\vec{p}_k$ — Update the residual

Computational Cost

Algorithm Cost for the kth iteration of GCR:

$\vec{r}^0 = \vec{b} - \mathbf{M}\vec{x}^0 = \vec{b}$

REPEAT

$\vec{p}_k \leftarrow \vec{r}^k$

For $j = 1$ to $k-1$ — $O(k)$ inner products: $O(nk)$ mult.

$\vec{p}_k \leftarrow \vec{p}_k - (\mathbf{M}\vec{p}_j)\cdot(\mathbf{M}\vec{p}_k)\vec{p}_j$

Matrix vector product: $O(n)$ if sparse

$\vec{p}_k \leftarrow \dfrac{\vec{p}_k}{\|\mathbf{M}\vec{p}_k\|}$

$y_k \leftarrow \vec{r}^k \cdot (\mathbf{M}\vec{p}_k)$ — Vector inner products: $O(n)$ mult.

$\vec{x}^{k+1} \leftarrow \vec{x}^k + y_k \vec{p}_k$ — $O(n)$ multiplications

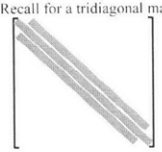
$\vec{r}^{k+1} \leftarrow \vec{r}^k + y_k \mathbf{M}\vec{p}_k$ — $O(n)$ multiplications

$$\text{total cost} = O(n) + O(2n) + \ldots + O(kn) = O(k^2 n)$$

If M is sparse, as k (number of iterations) approaches n: $O(n^3)$

Better Converge Fast!!

Recall for a tridiagonal matrix



Gaussian Elimination cost = $O(n)$

Algorithm Cost for iteration k

For $j = 1$ to $k-1$ — $O(k)$ inner products: $O(kn)$ mult.

$\vec{p}_k \leftarrow \vec{p}_k - (\mathbf{M}\vec{p}_j)\cdot(\mathbf{M}\vec{p}_k)\vec{p}_j$

If M is sparse the inner products are the dominant cost:

$$\text{total cost} = O(n) + O(2n) + \ldots + O(kn) = O(k^2 n)$$

As k (# of iters) approaches N, total cost = $O(n^3)$

Better Converge Fast!!

Symmetric Case – An amazing fact that will not be derived

If $\mathbf{M} = \mathbf{M}^T$ then $\mathbf{M}\vec{r}^k \perp \mathbf{M}\vec{p}_j$ for $j < k-2$

For $j = 1$ to $k-1$ — Only ONE inner products: $O(n)$ mult.

$\vec{p}_k \leftarrow \vec{r}_k - (\mathbf{M}\vec{p}_{k-1})\cdot(\mathbf{M}\vec{r}_k)\vec{p}_{k-1}$

If k (# of iters) $\rightarrow n$, then symmetric, sparse, GCR is worst case $O(n^2)$

Better Converge Fast!!

		SPARSE	DENSE
$\mathbf{M}\vec{r}^k$	Matrix-Vector Product	$O(n)$	$O(n^2)$
$\sum(\mathbf{M}\vec{r}^k)^T(\mathbf{M}\vec{p}_j)$	M-Orthogonalize	$\cancel{O(kn)}$ $O(n)$	$\cancel{O(kn)}$ $O(n)$
$(\vec{r}^k)^T(\mathbf{M}\vec{p}_k)$	Optimal Step Size Update	$O(n)$	$O(n)$
	Total	Symmetric: $\underbrace{O(Kn)}_{\text{matrix-vector product cost}} + \cancel{O(K^2 n)}$ $O(Kn)$ (inner products cost)	$\underbrace{O(Kn^2)}_{\text{matrix-vector product cost}} + \cancel{O(K^2 n)}$ $O(Kn)$ (inner products cost)
	Worst Case As $K \rightarrow n$	$\underbrace{O(n^2) + \cancel{O(n^3)}}_{O(n^2)}$ $O(n^2)$ Need to reduce k!!	$\underbrace{O(n^3) + \cancel{O(n^5)}}_{O(n^3)}$ $O(n^3)$ Need to reduce k AND matrix-vector cost
	Best Case For small K (5-10 iterations)	$\underbrace{O(n) + O(n)}_{O(n)}$	$\underbrace{O(n^2) + O(n)}_{O(n^2)}$

Cost Per Iteration

Note: If "fast" $\{=O(n)\}$ matrix-vector product then $O(n)$ total also for dense case.

Comparison – GCR & LU

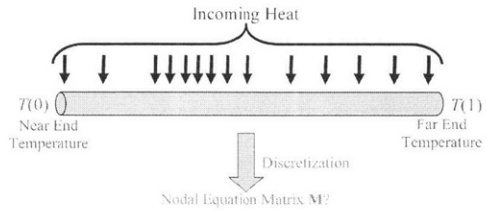
	Sparse Symmetric	Sparse Asymmetric	Dense
GCR	$O(Kn)$	$O(K^2 n)$	$O(Kn^2)$
LU	$O(n^{1.2-1.8})$	$O(n^{1.2-1.8})$	$O(n^3)$

- Making GCR converge fast can be very problem specific
- If the number of iterations k is about constant and small GCR is much faster than LU for dense problems.

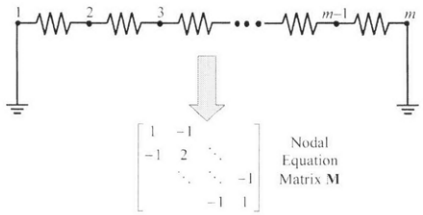
	Sparse	Dense
GCR	$O(n)$	$O(n^2)$
LU	$O(n^{1.2-1.8})$	$O(n^3)$

Convergence Examples

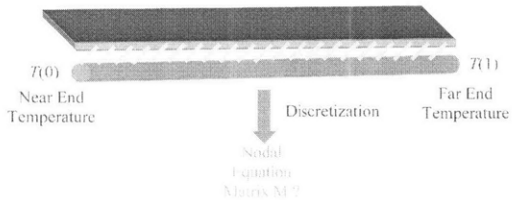
- "No-leak" Examples
  - Insulated Bar and Matrix



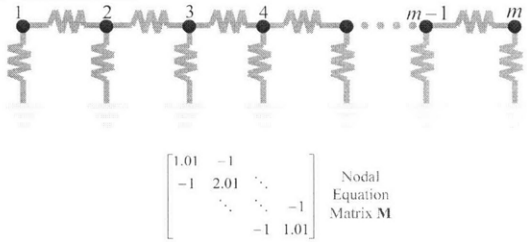
  - Circuit and Matrix



- "Leaky" Examples
  - Conducting Bar and Matrix

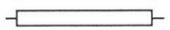


  - Circuit and Matrix

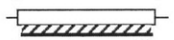


Example. Heat conducting bar.

$$\begin{bmatrix} 2 & -1 & & & \\ -1 & 2 & -1 & & \\ & -1 & 2 & \ddots & \\ & & \ddots & \ddots & -1 \\ & & & -1 & 2 \end{bmatrix} \qquad \begin{bmatrix} 3 & -1 & & & \\ -1 & 3 & -1 & & \\ & -1 & 3 & \ddots & \\ & & \ddots & \ddots & -1 \\ & & & -1 & 3 \end{bmatrix}$$



Insulating bar



Conduction to outside

$$u_{ii} = \underbrace{d}_{2 \text{ or } 3} - \frac{1}{u_{i-1,i-1}}$$

Suppose $d = 2$

$$u_{22} = 2 - \frac{1}{2} = \frac{3}{2}$$

$$u_{33} = 2 - \frac{1}{\frac{3}{2}} = \frac{4}{3}$$

$$\vdots$$

$$u_{ii} = \frac{i+1}{i}$$

Suppose $d = 3$

$$u_{22} = 3 - \frac{1}{3} = \frac{8}{3}$$

$$u_{33} = 3 - \frac{1}{\frac{8}{3}} = \frac{21}{8}$$

$$\vdots$$

pattern is not so obvious

Recurrence formula: $u_{ii} = d - \dfrac{1}{u_{i-1,i-1}}$

$$= d - \cfrac{1}{d - \cfrac{1}{d - \cfrac{1}{d - \cfrac{1}{d - \ddots}}}}$$

Continued Fraction

Let the asymptotic value be x : $u_{ii} = x$ & $u_{i-1,i-1} = x$ for $\lim_{i \to \infty}$

$$\Rightarrow x = d - \frac{1}{x} \Rightarrow x^2 - dx + 1 = 0$$

For $d = 2$: $x \to 1$

For $d = 3$: $x \to \dfrac{3 - \sqrt{5}}{2}$

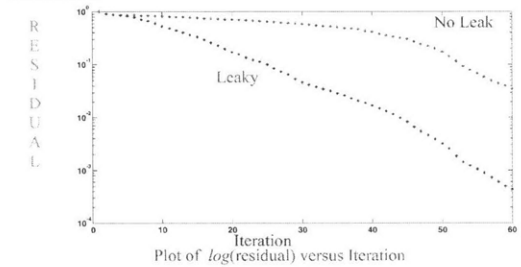
Continued Fraction…

$$u_{ii} = \left.\begin{matrix} x[i] \leftarrow d \cdot x[i-1] - x[i-2] \\ y[i] \leftarrow d \cdot y[i-1] - y[i-2] \end{matrix}\right\} \text{simple linear difference equations}$$

❖ GCR Performance

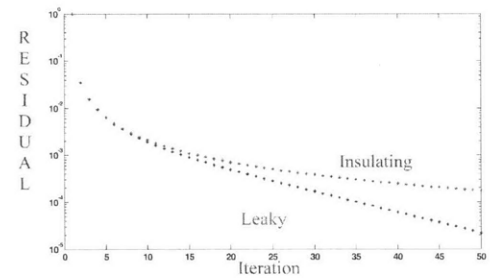
Example with 100 nodes (102 resistors)

- Random RHS



Plot of *log*(residual) versus Iteration

- RHS = (−1,+1, −1,+1, −1,…)



Plot of *log*(residual) versus Iteration

3.9 Linear Systems - GCR Convergence

INTRODUCTION TO NUMERICAL SIMULATION

LECTURE 9.
GCR Convergence

TODAY'S OUTLINE:

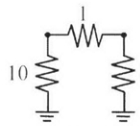
- Review – Eigenvalues
  - Norms and Spectral Radius
  - Spectral Mapping Theorem
- GCR Convergence Analysis
  - Polynomial Error Bound
  - Chebychev Bounds
  - Gershgorin Circle Theorem

EIGENVALUES

$$\mathbf{M}\vec{u}_i - \lambda_i \vec{u}_i = 0$$

$$\Rightarrow (\mathbf{M} - \lambda_i \mathbf{I})\vec{u}_i = 0$$

Example.



$$\Rightarrow \begin{bmatrix} 1.1 & -1 \\ -1 & 1.1 \end{bmatrix} \underbrace{\begin{bmatrix} \\ \end{bmatrix}}_{?}$$

Circuit is symmetric – expect the node voltages might be the same

$$\begin{bmatrix} 1.1 & -1 \\ -1 & 1.1 \end{bmatrix}\begin{bmatrix} 1 \\ 1 \end{bmatrix} \xrightarrow{\text{expected eigenvalue}} (0.1)\begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

corresponds to 10Ω resistor

Suppose the voltages are different at the nodes…

$$\begin{bmatrix} 1.1 & -1 \\ -1 & 1.1 \end{bmatrix}\begin{bmatrix} -1 \\ 1 \end{bmatrix} \xrightarrow{\text{expected eigenvalue}} (2.1)\begin{bmatrix} -1 \\ 1 \end{bmatrix}$$

Example. Block Diagonal Matrix

$$\begin{bmatrix} 1 & -1 & 0 & 0 \\ -1 & 1 & 0 & 0 \\ 0 & 0 & 1 & -1 \\ 0 & 0 & -1 & 2 \end{bmatrix}$$

Example. Lower Triangular Matrix

$$\begin{bmatrix} M_{11} & 0 & 0 & \cdots & 0 \\ M_{21} & M_{22} & 0 & \cdots & 0 \\ M_{31} & \ddots & \ddots & \ddots & \vdots \\ \vdots & \ddots & M_{N-1,N-2} & M_{N-1,N-1} & 0 \\ M_{N,1} & \cdots & M_{N,N-2} & M_{N,N-1} & M_{N,N} \end{bmatrix}$$

Eigenvalues are the diagonal elements.

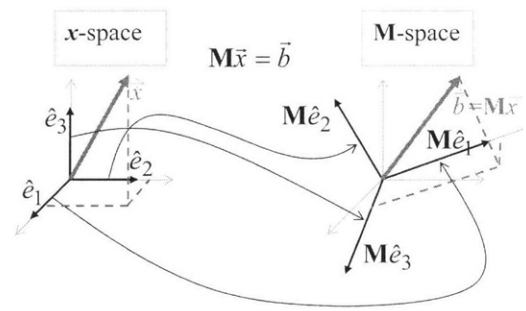
Eigenvalues and eigenvectors of a matrix $\mathbf{M}$ satisfy

$$\mathbf{M}\vec{u}_i = \lambda_i \vec{u}_i$$

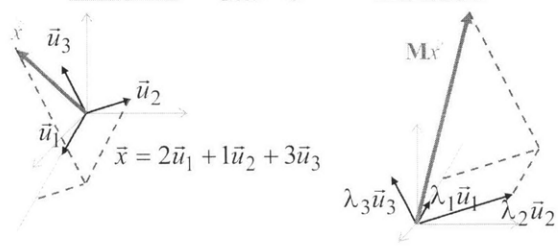
eigenvalue

eigenvector

Or, λ_i is an eigenvalue of $\mathbf{M}$ if
$\mathbf{M} - \lambda_i \mathbf{I}$ is singular
$\vec{u}_i$ is an eigenvector of $\mathbf{M}$ if
$(\mathbf{M} - \lambda_i \mathbf{I})\vec{u}_i = 0$



x-space $\qquad \mathbf{M}\vec{x} = \vec{b} \qquad$ $\mathbf{M}$-space



$$\mathbf{M}\vec{x} = 2\underbrace{\lambda_1}_{-\frac{1}{2}}\vec{u}_1 + 1\underbrace{\lambda_2}_{2}\vec{u}_2 + 3\underbrace{\lambda_3}_{1}\vec{u}_3$$

$$\mathbf{M}\vec{x} = 2\left(-\tfrac{1}{2}\right)\vec{u}_1 + 1(2)\vec{u}_2 + 3(1)\vec{u}_3$$

Easy to do – the $\mathbf{M}$ space uses the same vectors as the x-space, they are just stretched by multiplier λ_i.

<u>A Simplifying Assumption</u>

$$\mathbf{M}\begin{bmatrix} \uparrow & \uparrow & \uparrow & & \uparrow \\ \vec{u}_1 & \vec{u}_2 & \vec{u}_3 & \cdots & \vec{u}_N \\ \downarrow & \downarrow & \downarrow & & \downarrow \end{bmatrix} = \begin{bmatrix} \uparrow & \uparrow & \uparrow & & \uparrow \\ \lambda_1\vec{u}_1 & \lambda_2\vec{u}_2 & \lambda_3\vec{u}_3 & \cdots & \lambda_N\vec{u}_N \\ \downarrow & \downarrow & \downarrow & & \downarrow \end{bmatrix}$$

$$\left.\begin{aligned} &\mathbf{M}\vec{u}_1 = \lambda_1\vec{u}_1 \\ \Rightarrow\ &\mathbf{M}\vec{u}_2 = \lambda_2\vec{u}_2 \\ &\vdots \end{aligned}\right\} \Rightarrow \mathbf{MU} = \Lambda\mathbf{U}$$

The set of all eigenvalues of $\mathbf{M}$ is known as the Spectrum of $\mathbf{M}$.

$$\mathbf{MU} = \mathbf{U}\begin{bmatrix} \lambda_1 & & \\ & \ddots & \\ & & \lambda_N \end{bmatrix} \Rightarrow \mathbf{U}^{-1}\mathbf{MU} = \Lambda \ \text{ or } \ \mathbf{M} = \mathbf{U}\Lambda\mathbf{U}^{-1}$$

Almost all $N \times N$ matrices have N linearly independent eigenvectors (i.e. $\mathbf{U}$ non-singular)

Does NOT imply distinct eigenvalues, i.e. λ_i can equal λ_j

Does NOT imply $\mathbf{M}$ is nonsingular

Example.) $\mathbf{U}$ nonsingular $\not\longrightarrow$ Distinct Eigenvalues

$$\mathbf{U} = \mathbf{I} = \begin{bmatrix} 1 & & & \\ & 1 & & \\ & & 1 & \\ & & & 1 \end{bmatrix}$$

- All eigenvalues are equal, $\lambda_i = 1 \ \forall i$
- Eigenvectors are distinct, $\vec{u}_i = \vec{e}_i$

Example.) $\mathbf{U}$ nonsingular $\not\longrightarrow$ $\mathbf{M}$ nonsingular

$$\mathbf{M} = \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix}$$

zero eigenvalue

$\mathbf{M}$ singular

has L.I. eigenvectors : $\vec{v}_1 = \vec{\rho}_1 \quad \vec{v}_2 = \vec{\rho}_2$

$$\left.\begin{aligned} &\begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix}\underbrace{\begin{bmatrix} 1 \\ 0 \end{bmatrix}}_{\vec{u}_1} = \underbrace{1}_{\lambda_1}\begin{bmatrix} 1 \\ 0 \end{bmatrix} \\ &\begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix}\underbrace{\begin{bmatrix} 0 \\ 1 \end{bmatrix}}_{\vec{u}_2} = \underbrace{0}_{\lambda_2}\begin{bmatrix} 0 \\ 1 \end{bmatrix} \end{aligned}\right\} \mathbf{U} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

$\mathbf{U}$ nonsingular

Example.) Real Matrices $\longrightarrow$ Real or Complex Conjugate Eigenvalues

$\mathbf{M} = \mathbf{U}\Lambda\mathbf{U}^{-1}$ Real Matrix

λ eigenvalue $\Leftrightarrow$ $(\mathbf{M} - \lambda\mathbf{I})$ is singular

$\mathbf{U}\Lambda\mathbf{U}^{-1} - \lambda\mathbf{U}\mathbf{I}\mathbf{U}^{-1} \rightarrow$ singular

$$\mathbf{U}\begin{bmatrix} \lambda_1 - \lambda & & \\ & \ddots & \\ & & \lambda_N - \lambda \end{bmatrix}\mathbf{U}^{-1} \rightarrow \text{singular}$$

$\varphi(\lambda) = \pi(\lambda_i - \lambda) = 0 \ \rightarrow$ characteristic polynomial

eigenvalues $\rightarrow$ real or complex conjugate pairs

from the *Fundamental Theorem of Algebra*

Symmetric Matrices $\longrightarrow \mathbf{U}^T\mathbf{U} = \mathbf{I}$ Orthonormal Eigenvectors

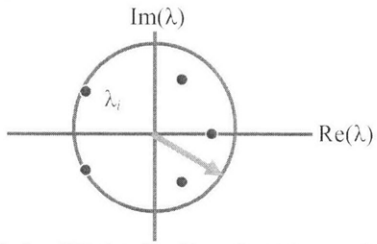
$\mathbf{M} = \mathbf{U}\Lambda\mathbf{U}^{-1} \qquad \mathbf{M}^T = \left(\mathbf{U}^{-1}\right)^T\Lambda\mathbf{U}^T$

Symmetric $\longrightarrow \mathbf{M} = \mathbf{M}^T$

$\mathbf{U}\Lambda\mathbf{U}^{-1} = \left(\mathbf{U}^{-1}\right)^T\Lambda\mathbf{U}^T \Leftrightarrow \mathbf{U}^{-1} = \mathbf{U}^T$

$\longrightarrow$ $\mathbf{U}$ orthonormal

Spectral Radius

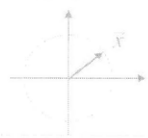


The spectral Radius of **M** is the radius of the smallest circle, centered at the origin, which encloses all of **M**'s eigenvalues.

Suppose $\vec{y} = \mathbf{M}\vec{x}$

How much larger is $\vec{y}$ than $\vec{x}$? OR How much does **M** magnify $\vec{x}$?

$$\begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix}\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \Rightarrow$$



$$\begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix}\begin{bmatrix} 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

$$\begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix}\begin{bmatrix} 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \bigstar$$

This vector will get the most magnification from matrix **M**

Look at the direction corresponding to the largest eigenvalue

Any induced norm is a bound on the spectral radius

If **M** is symmetric then $\|\mathbf{M}\|_2$ = spectral radius

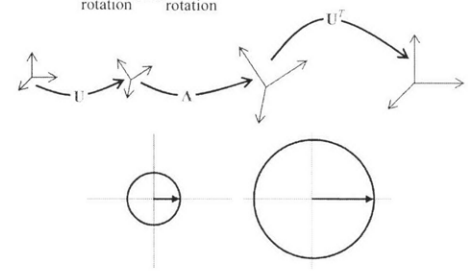
Example. $\begin{bmatrix} 1 & 0 \\ 100 & 1 \end{bmatrix}$ $\quad eig\left(\begin{bmatrix} 1 & 0 \\ 100 & 1 \end{bmatrix}\right) = \{1,1\}$

$$\left\|\begin{bmatrix} 1 & 0 \\ 100 & 1 \end{bmatrix}\right\|_{\infty,1} = 101$$

$\mathbf{M} = \mathbf{U}\Lambda\mathbf{U}^{-1} = \mathbf{U}\Lambda\mathbf{U}^T \qquad \mathbf{M} = \mathbf{M}^T$

$$\|\mathbf{M}\|_2 = \left\|\mathbf{U}\Lambda\mathbf{U}^T\right\|_2 = \|\mathbf{U}\|_2\|\Lambda\|_2\left\|\mathbf{U}^T\right\|_2 = \|\Lambda\|_2 = \max_i|\lambda_i|$$

rotation stretch rotation



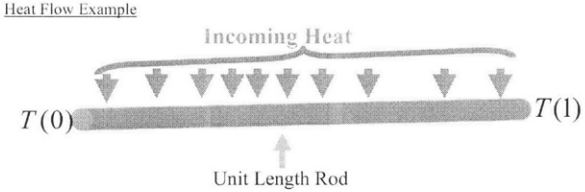
Induced Matrix Norms – standard induced l-norms

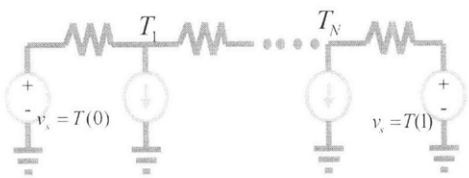
Definition:

$$\|\mathbf{M}\|_l \equiv \max_x \frac{\|\mathbf{M}x\|_l}{\|x\|_l} = \max_{\|x\|_l=1}\|\mathbf{M}x\|_l$$

$$\|\mathbf{M}\|_2 = \max_i \sqrt{\lambda_i\left(\mathbf{M}^H\mathbf{M}\right)} = \max_i|\lambda_i(\mathbf{M})| = \begin{cases} \text{for symmetric real} \\ \text{matrices = max eigenvalue} \end{cases}$$

Heat Flow Example





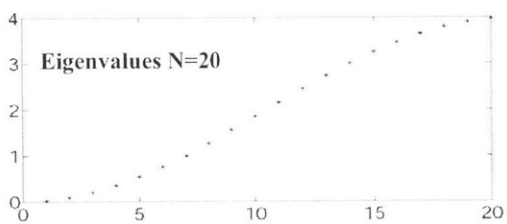
$\vec{x}$ = Distribution of temperatures

$\mathbf{M}\vec{x}$ = Distribution of incoming/outgoing heat

$\mathbf{M}\vec{x} = \lambda\vec{x}$ The "shape" of the temperature distribution is the same as the incoming/outgoing heat

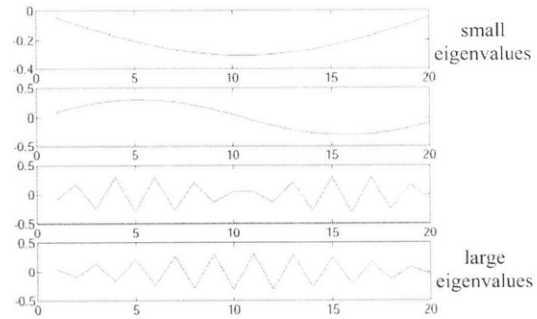
λ small means that for that eigenvector distribution of temperatures the resulting heat flow is small.

$$\begin{bmatrix} 1 & -1 & & \\ -1 & 2 & \ddots & \\ & \ddots & \ddots & -1 \\ & & -1 & 1 \end{bmatrix}$$



heat flow $(h_i) \propto T_i - T_{i+1}$

Four Eigenvectors – Which ones?



GCR Convergence Analysis

The Iterative Algorithm

$x^0 = 0$

$r^0 = b$

REPEAT

Compute $\mathbf{M}r^k$

$\mathbf{M}^T\mathbf{M}$ Orthonormalize search directions

Update solution x^{k+1}

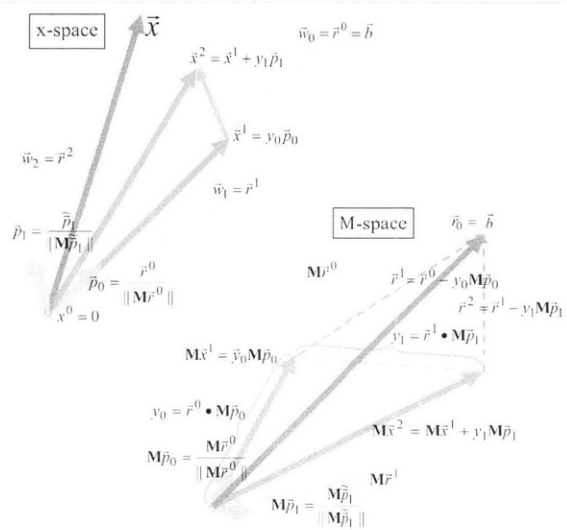
Update residue r^{k+1}

UNTIL residue small "enough"

Problem Statement

Let's estimate an upper bound for how much we have reduced the residual after k iterations

$$\frac{\|\vec{r}^k\|}{\|\vec{r}^0\|} \le ?$$



Krylov Subspace

Note : $span\{\nabla_x f(x^0),\ldots,\nabla_x f(x^{k-1})\} = span\{r^0,\ldots,r^{k-1}\}$

If $span\{\vec{w}_0,\ldots,\vec{w}_{k-1}\} = span\{r^0,\ldots,r^{k-1}\}$

then $r^k = r^0 - \sum_{i=0}^{k-1}\alpha_i M r^i$

and $span\{r^0,\ldots,r^{k-1}\} = \underbrace{span\{r^0,\mathbf{M}r^0,\ldots,\mathbf{M}^{k-1}r^0\}}_{\text{Krylov Subspace}}$

$k = 1$

Step 1.

$$\vec{p}_0 \leftarrow \frac{\vec{r}^0}{\|\mathbf{M}\vec{r}^0\|} \quad \in span\{\vec{r}^0\}$$

$$y_0 \leftarrow \vec{r}^{0^T}\mathbf{M}\vec{p}_0 \quad \text{scalar}$$

$$\vec{x}^1 \leftarrow 0 + y_0\vec{p}_0 \quad \in span\{\vec{p}_0\} = span\{\vec{r}^0\}$$

$$\vec{r}^1 \leftarrow \vec{r}^0 - y_0\mathbf{M}\vec{p}_0 \quad \in span\{\vec{r}^0,\mathbf{M}\vec{p}_0\} = span\{\vec{r}^0,\mathbf{M}\vec{r}^0\}$$

$k = 2$

Step 2.

$$\tilde{\vec{p}}_1 \leftarrow \vec{r}^1 - \overbrace{(\mathbf{M}\vec{p}_0)^T(\mathbf{M}\vec{r}^1)}^{\text{scalar}}\vec{p}_0 \quad \in span\{\vec{r}^1,\vec{p}_0\} = span\{\vec{r}^0,\mathbf{M}\vec{r}^0\}$$

$$\vec{p}_1 \leftarrow \frac{\tilde{\vec{p}}_1}{\|\mathbf{M}\tilde{\vec{p}}_1\|} \quad \in span\{\vec{r}^0,\mathbf{M}\vec{r}^0\}$$

$$y_1 \leftarrow \vec{r}^{1^T}\mathbf{M}\vec{p}_0 \quad \text{scalar}$$

$$\vec{x}^2 \leftarrow \vec{x}^1 + y_1\vec{p}_1 \quad \in span\{\vec{x}^1,\mathbf{M}\vec{p}_1\} = span\{\vec{r}^0,\mathbf{M}\vec{r}^0\}$$

$$\vec{r}^2 \leftarrow \vec{r}^1 - y_1\mathbf{M}\vec{p}_1 \quad \in span\{\vec{r}^1,\mathbf{M}\vec{p}_1\} = span\{\vec{r}^0,\mathbf{M}\vec{r}^0,\mathbf{M}^2\vec{r}^0\}$$

$$\vec{x}^k \leftarrow \alpha_0\mathbf{I}\vec{r}^0 + \alpha_1\mathbf{M}\vec{r}^0 + \ldots + \alpha_{k-1}\mathbf{M}^{k-1}\vec{r}^0 = \xi_{k-1}(\mathbf{M})\vec{r}^0$$

$$\vec{r}^k \leftarrow \beta_0\vec{r}^0 + \beta_1\mathbf{M}\vec{r}^0 + \ldots + \beta_k\mathbf{M}^k\vec{r}^0 = \varphi_k(\mathbf{M})\vec{r}^0$$

Polynomial View

$$\tilde{\vec{p}}_{k-1} \leftarrow r^{k-1} - (\mathbf{M}\vec{p}_{k-2})^T(\vec{r}^{k-1})\vec{p}_{k-2}$$

$$\vec{p}_{k-1} \in span\{\vec{r}^0,\mathbf{M}\vec{r}^0,\ldots,\mathbf{M}^{k-1}\vec{r}^0\}$$

$$\vec{x}^k \leftarrow \vec{x}^{k-1} + y_{k-1}\vec{p}_{k-1}$$

$$\vec{x}^k \in span\{\vec{r}^0,\mathbf{M}\vec{r}^0,\ldots,\mathbf{M}^{k-1}\vec{r}^0\}$$

$$\vec{x}^k = \xi_{k-1}(\mathbf{M})\vec{r}^0 \quad \rightarrow (k-1)^{\text{th}} \text{ order polynomial in } \mathbf{M}$$

$$\vec{r}^k \leftarrow \vec{r}^{k-1} - y_{k-1}\mathbf{M}\vec{p}_{k-1}$$

$$\vec{r}^k \in span\{\vec{r}^0,\mathbf{M}\vec{r}^0,\ldots,\mathbf{M}^{k-1}\vec{r}^0,\mathbf{M}^k\vec{r}^0\}$$

$$\vec{r}^k = \varphi_k(\mathbf{M})\vec{r}^0 \quad \rightarrow k^{\text{th}} \text{ order polynomial in } \mathbf{M}$$

$\varphi_k(0) \quad \mathbf{M}=0 \qquad \mathbf{M}\vec{x}=\vec{b} \qquad 0\vec{x}=\vec{b} \qquad \vec{r}^{\,k}=\vec{b}$

$\varphi_k(0)=\beta_0\mathbf{I}+\cancel{\beta_1\mathbf{M}}+\cancel{\beta_2\mathbf{M}^2}+\cdots+\cancel{\beta_k\mathbf{M}^k}=\beta_0\mathbf{I}$

$\vec{b}=\vec{r}^{\,k}=\varphi_k(0)\vec{r}^{\,0} \quad \Rightarrow \varphi_k(0)=\mathbf{I}$

Residual Minimization

If $\vec{x}^{\,k}\in span\{\vec{r}^{\,0},\mathbf{M}\vec{r}^{\,0},\ldots,\mathbf{M}^{k-1}\vec{r}^{\,0}\}$ minimizing $\left\|\vec{r}^{\,k}\right\|_2^2$

1.) $\vec{x}^{\,k}=\xi_{k-1}(\mathbf{M})\vec{r}^{\,0}$

$\xi_{k-1}(\mathbf{M})$ is the kth order poly minimizing $\left\|\vec{r}^{\,k}\right\|_2^2$

2.) $\vec{r}^{\,k}=\varphi_k(\mathbf{M})\vec{r}^{\,0}$

$=\vec{b}-\mathbf{M}\vec{x}^{\,k}=(\mathbf{I}-\mathbf{M}\xi_k(\mathbf{M}))\vec{r}^{\,0}$

where $\varphi_k(\mathbf{M})r^0$ is the kth order poly minimizing $\left\|\vec{r}^{\,k}\right\|_2^2$ subject to $\varphi_k(0)=\mathbf{I}$

Polynomial Property only a function of solution space and residual minimization

Explicitly construct Krylov subspace:

$\vec{r}^{\,0}=\vec{b}-\mathbf{M}\underbrace{\vec{x}^{\,0}}_{0}$

$\mathbf{M}\vec{r}^{\,0}=\mathbf{M}\cdot\vec{r}^{\,0}$

$\mathbf{M}^2\vec{r}^{\,0}=\mathbf{M}\cdot\mathbf{M}\vec{r}^{\,0}$

$\vdots$

$\mathbf{M}\vec{r}^{\,k}=\mathbf{M}\cdot\mathbf{M}^{k-1}\vec{r}^{\,0}$

Orthogonalize & Project

Residual Minimizing Optimality Property:

$\left|\vec{r}^{\,k}\right|=\left|\varphi_k(\mathbf{M})\vec{r}^{\,0}\right|\le\left|\tilde{\varphi}_k(\mathbf{M})\vec{r}^{\,0}\right|\le\left\|\tilde{\varphi}_k(\mathbf{M})\right\|\left|\vec{r}^{\,0}\right|$

$\tilde{\varphi}_k$ is any k^{th} order poly such that $\tilde{\varphi}_k(0)=\mathbf{I}$

Therefore

Any polynomial which satisfies the constraint can be used to get

an upper bound on $\dfrac{\left|\vec{r}^{\,k}\right|}{\left|\vec{r}^{\,0}\right|}\le\left\|\tilde{\varphi}_k(\mathbf{M})\right\|$

Bound on quantity of residual reduction

$\left\|\vec{r}^{\,k+1}\right\|<\varepsilon \longrightarrow \mathbf{M}\vec{x}^{\,k+1}\cong\vec{b}$

$\left\|\vec{x}^{\,k+1}-\vec{x}^{\,*}\right\|<\varepsilon$

$\vec{x}^{\,*}=\mathbf{M}^{-1}\vec{b}$

$$\left.\begin{aligned}\vec{b}-\mathbf{M}\vec{x}^{\,k+1}&=\vec{r}^{\,k+1}\\ \vec{b}-\mathbf{M}\vec{x}^{\,k+1}&=0\end{aligned}\right\}\mathbf{M}\left(\vec{x}^{\,k+1}-\vec{x}^{\,*}\right)=-\vec{r}^{\,k+1}$$

$$\Rightarrow\left\|\vec{x}^{\,k+1}-\vec{x}^{\,*}\right\|=\left\|\mathbf{M}^{-1}\vec{r}^{\,k+1}\right\|\le\left\|\mathbf{M}^{-1}\right\|\left\|\vec{r}^{\,k+1}\right\|=\left\|\mathbf{M}^{-1}\right\|\varepsilon$$

$\tilde{\varphi}_2(x)=3+2x-4x^2$

$$\tilde{\varphi}_2(\mathbf{M})=3\mathbf{I}+2\mathbf{M}-4\mathbf{M}^2=3\mathbf{U}\mathbf{U}^{-1}+2\mathbf{U}\Lambda\mathbf{U}^{-1}-4\mathbf{U}\Lambda^2\mathbf{U}^{-1}$$
$$=\mathbf{U}\left(3+2\Lambda-4\Lambda^2\right)\mathbf{U}^{-1}$$

$$\mathbf{M}=\mathbf{U}\begin{bmatrix}\lambda_1 & & \\ & \lambda_2 & \\ & & \lambda_3\end{bmatrix}\mathbf{U}^{-1}$$

$\lambda_i\rightarrow\varphi_2(\lambda_i)$

$$\tilde{\varphi}_2(\mathbf{M})=\mathbf{U}\begin{bmatrix}3+2\lambda_1-4\lambda_1^2 & & \\ & 3+2\lambda_2-4\lambda_2^2 & \\ & & 3+2\lambda_3-4\lambda_3^2\end{bmatrix}\mathbf{U}^T$$

Spectral Mapping Theorem.

Given a polynomial $\tilde{\varphi}_p(x)=a_0+a_1x+\ldots+a_px^p$

Apply the polynomial to a matrix $\tilde{\varphi}_p(\mathbf{M})=a_0I+a_1\mathbf{M}+\ldots+a_p\mathbf{M}^p$

Then $spectrum(\tilde{\varphi}_p(\mathbf{M}))=\tilde{\varphi}_p(spectrum(\mathbf{M}))$

Proof.

Note a property of matrix powers

$\mathbf{M}\mathbf{M}=\mathbf{U}\lambda\mathbf{U}^{-1}\mathbf{U}\lambda\mathbf{U}^{-1}=\mathbf{U}\lambda^2\mathbf{U}^{-1} \quad\Rightarrow\quad \mathbf{M}^p=\mathbf{U}\lambda^p\mathbf{U}^{-1}$

Apply to the polynomial of the matrix

$\tilde{\varphi}_p(\mathbf{M})=a_0\mathbf{U}\mathbf{U}^{-1}+a_1\mathbf{U}\lambda\mathbf{U}^{-1}+\ldots+a_p\mathbf{U}\lambda^p\mathbf{U}^{-1}$

Factoring $\tilde{\varphi}_p(\mathbf{M})=\mathbf{U}\underbrace{\left(a_0I+a_1\lambda+\ldots+a_p\lambda^p\right)}_{diagonal}\mathbf{U}^{-1}$

$$spectrum\left(\widetilde{\wp}_p(\mathbf{M})\right) = spectrum\left(a_0 I + a_1\lambda + \ldots + a_p\lambda^p\right) = \widetilde{\wp}_p\left(spectrum(\mathbf{M})\right)$$

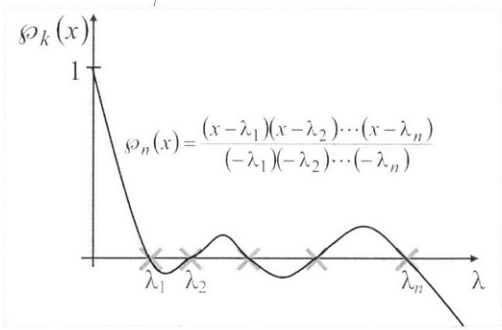
$$\begin{bmatrix} a_0 + a_1\lambda_1 + \ldots + a_p\lambda_1^p & & \\ & \ddots & \\ & & a_0 + a_1\lambda_N + \ldots + a_p\lambda_N^p \end{bmatrix}$$

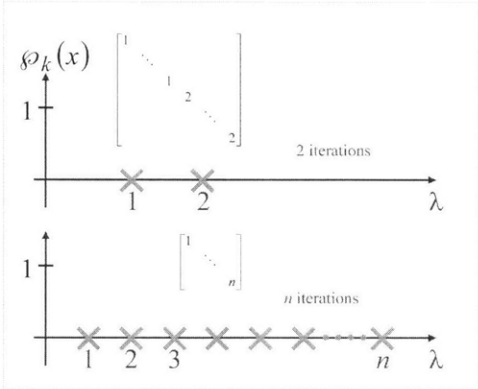
Norm of matrix polynomials.

$$\left\|\widetilde{\wp}_k(\mathbf{M})\right\| = \left\| \underbrace{\begin{bmatrix} \uparrow & & \uparrow \\ \vec{u}_1 & \cdots & \vec{u}_N \\ \downarrow & & \downarrow \end{bmatrix}}_{\text{eigenvectors of } \mathbf{M}} \begin{bmatrix} \widetilde{\wp}_k(\lambda_1) & & \\ & \ddots & \\ & & \widetilde{\wp}_k(\lambda_N) \end{bmatrix} \begin{bmatrix} \uparrow & & \uparrow \\ \vec{u}_1 & \cdots & \vec{u}_N \\ \downarrow & & \downarrow \end{bmatrix}^{-1} \right\|$$

$$\leq \underbrace{\left\| \begin{bmatrix} \uparrow & & \uparrow \\ \vec{u}_1 & \cdots & \vec{u}_N \\ \downarrow & & \downarrow \end{bmatrix} \begin{bmatrix} \uparrow & & \uparrow \\ \vec{u}_1 & \cdots & \vec{u}_N \\ \downarrow & & \downarrow \end{bmatrix}^{-1} \right\|}_{\substack{\text{condition number} \\ \text{of } \mathbf{M}\text{'s eigenspace}}} \left\| \begin{bmatrix} \widetilde{\wp}_k(\lambda_1) & & \\ & \ddots & \\ & & \widetilde{\wp}_k(\lambda_N) \end{bmatrix} \right\|$$

$$\leq cond(\mathbf{U}) \max_i \left|\widetilde{\wp}_k(\lambda_i)\right|$$





Symmetric positive semidefinite case

If $\mathbf{M} = \mathbf{M}^T$ then

1. $\mathbf{M}$ has orthonormal eigenvectors

$$cond(\mathbf{U}) = \left\| \begin{bmatrix} \uparrow & & \uparrow \\ \vec{u}_1 & \cdots & \vec{u}_N \\ \downarrow & & \downarrow \end{bmatrix} \begin{bmatrix} \uparrow & & \uparrow \\ \vec{u}_1 & \cdots & \vec{u}_N \\ \downarrow & & \downarrow \end{bmatrix}^{-1} \right\| = 1$$

$$\Rightarrow \left\|\widetilde{\wp}_k(\mathbf{M})\right\| = \max_i \left|\widetilde{\wp}_k(\lambda_i)\right|$$

2. $\mathbf{M}$ has real eigenvalues

   If $\mathbf{M}$ is positive definite, then $\lambda(\mathbf{M}) > 0$

Bound on quantity of residual reduction

$$\left|\vec{r}^{\,k+1}\right| < \varepsilon \longrightarrow \mathbf{M}\vec{x}^{\,k+1} \cong \vec{b}$$

$$\left|\vec{x}^{\,k+1} - \vec{x}^{\,*}\right| < \varepsilon$$

$$\vec{x}^{\,*} = \mathbf{M}^{-1}\vec{b}$$

$$\left.\begin{aligned} \vec{b} - \mathbf{M}\vec{x}^{\,k+1} &= \vec{r}^{\,k+1} \\ \vec{b} - \mathbf{M}\vec{x}^{\,k+1} &= 0 \end{aligned}\right\} \mathbf{M}\left(\vec{x}^{\,k+1} - \vec{x}^{\,*}\right) = -\vec{r}^{\,k+1}$$

$$\Rightarrow \left|\vec{x}^{\,k+1} - \vec{x}^{\,*}\right| = \left\|\mathbf{M}^{-1}\vec{r}^{\,k+1}\right\| \le \left\|\mathbf{M}^{-1}\right\|\left\|\vec{r}^{\,k+1}\right\| = \left\|\mathbf{M}^{-1}\right\|\varepsilon$$

Polynomial Error Bound

Upper bound on residual after k iterations

$$\frac{\left\|r^k\right\|}{\left\|r^0\right\|} = \left\|\varphi_k(\mathbf{M})\right\|$$

$\varphi_k(\mathbf{M})$ is the GCR polynomial of order k

$$\le \left\|\widetilde{\varphi}_k(\mathbf{M})\right\|$$

$\widetilde{\varphi}_k(\mathbf{M})$ is any k^{th} order polynomial such that $\widetilde{\varphi}_k(0) = \mathbf{I}$

$$\le cond(\mathbf{U}) \cdot \max_i \left|\widetilde{\varphi}_k(\lambda_i)\right|$$

$\mathbf{U}$ are the eigenvectors of $\mathbf{M}$

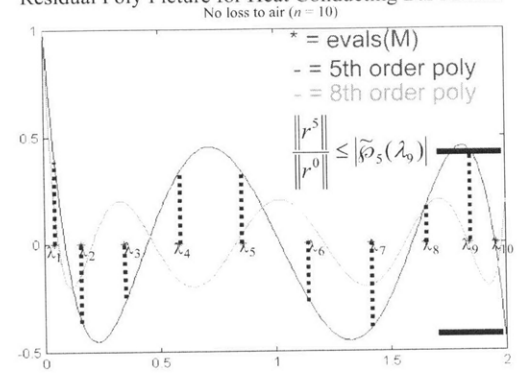
λ_i are the eigenvalues of $\mathbf{M}$

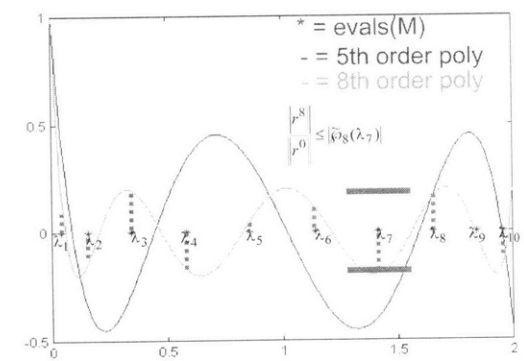
$$\le \max_i \left|\widetilde{\varphi}_k(\lambda_i)\right|$$

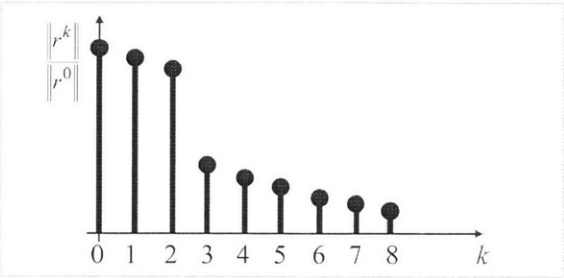
if $\mathbf{M}$ is symmetric then $\text{cond}(\mathbf{U}) = 1$

Residual Poly Picture for Heat Conducting Bar Matrix

No loss to air ($n = 10$)







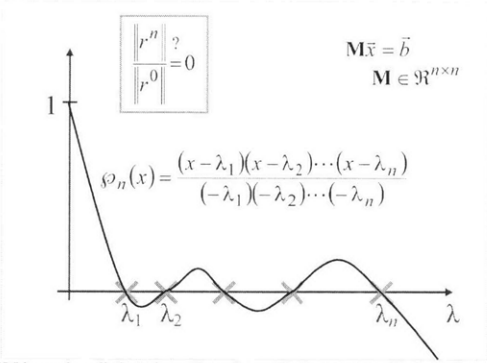
Important Observations

1. A residual minimizing Krylov subspace algorithm converges to the exact solution in at most n steps

   Proof:

   Let $\tilde{\wp}_n(x) = \dfrac{(x-\lambda_1)(x-\lambda_2)\cdots(x-\lambda_n)}{(-\lambda_1)(-\lambda_2)\cdots(-\lambda_n)}$ where $\lambda_i \in \lambda(M)$.

   Then, $\max_i \tilde{\wp}_n(\lambda_i) = 0 \Rightarrow \|\wp_n(\mathbf{M})\| = 0 \therefore \|r^n\| = 0$



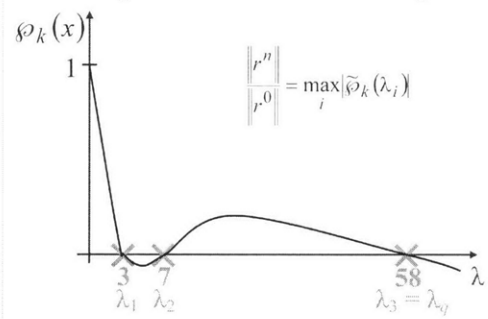
2. If $\mathbf{M}$ has only q distinct eigenvalues, the residual minimizing Krylov subspace algorithm converges in at most q steps

   Proof:

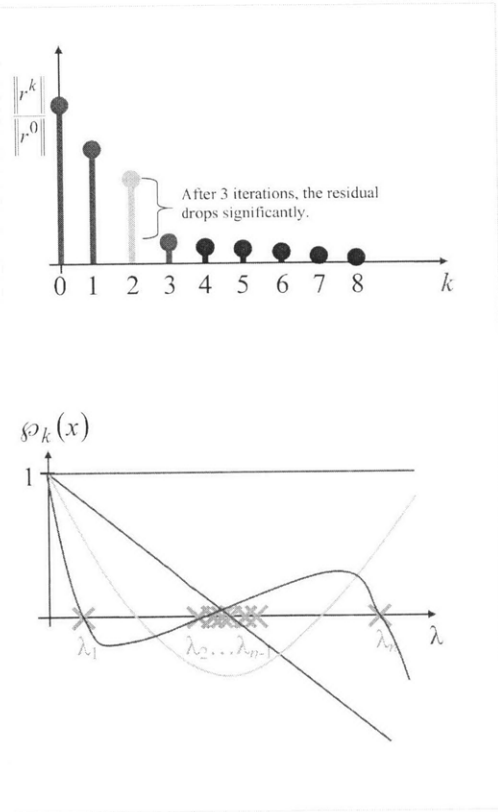
Let $\tilde{\wp}_q(x) = \dfrac{(x-\lambda_1)(x-\lambda_2)\cdots(x-\lambda_q)}{(-\lambda_1)(-\lambda_2)\cdots(-\lambda_q)}$

$$\mathbf{M} \in \Re^{n\times n}$$

$$\mathbf{M} = \mathbf{U}\begin{bmatrix} 3 & & & & & & & \\ & \ddots & & & & & & \\ & & 3 & & & & & \\ & & & 7 & & & & \\ & & & & \ddots & & & \\ & & & & & 7 & & \\ & & & & & & 58 & \\ & & & & & & & \ddots \\ & & & & & & & & 58 \end{bmatrix}\mathbf{U}^{-1}$$

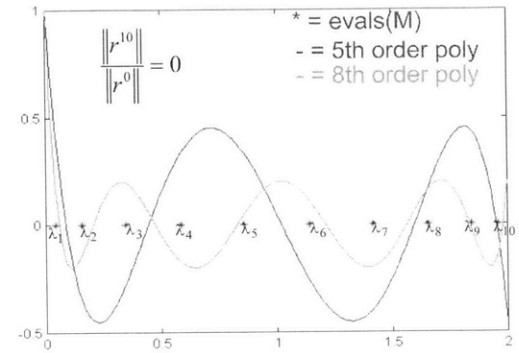


Repeated roots – converges much faster!! Only need to use a third order polynomial

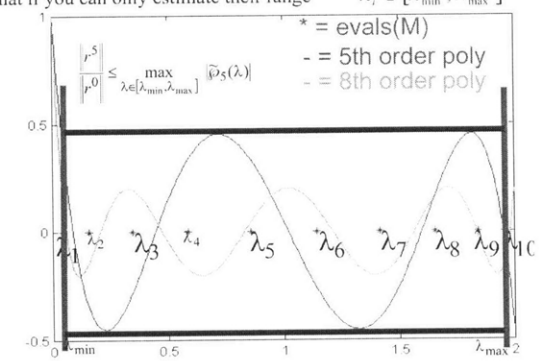


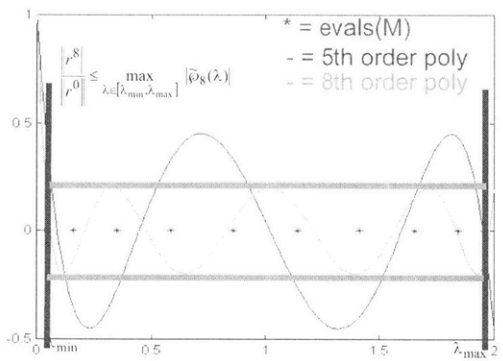
Chebyshev Bounds

How many iterations to converge?



What if you don't know how many and where the eigenvalues are?
What if you can only estimate their range $\lambda_i \in [\lambda_{min}, \lambda_{max}]$





Polynomial Min-Max Problem
[Convergence for $\mathbf{M} = \mathbf{M}^T$]

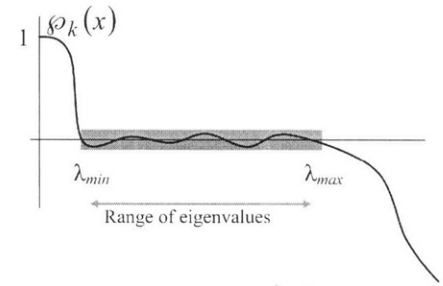
Consider $\lambda(M) \in [\lambda_{min}, \lambda_{max}]$, $\lambda_{min} > 0$

Then a good polynomial (i.e. $\|\tilde{\varphi}_k(\mathbf{M})\|$ is small) can be found by solving the min - max problem

$$\min_{\substack{k^{th}\text{ order} \\ \text{polys s.t.} \\ \tilde{\varphi}_k(0)=1}} \max_{x \in [\lambda_{min}, \lambda_{max}]} |\tilde{\varphi}_k(x)|$$

The min-max problem is exactly solved by Chebyshev Polynomials

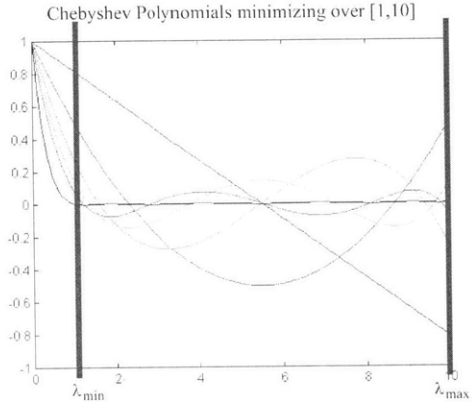
Find the polynomial that minimizes the value within the range $[\lambda_{min}, \lambda_{max}]$ and is equal to 1 at $\lambda=0$



$$\min_{\substack{k^{th}\text{ order} \\ \text{polys s.t.} \\ \tilde{\varphi}_k(0)=1}} \max_{x \in [\lambda_{min}, \lambda_{max}]} |\tilde{\varphi}_k(x)| = \max_{x \in [\lambda_{min}, \lambda_{max}]} \left| \frac{C_k\left(1 + 2\frac{\lambda_{min} - x}{\lambda_{max} - \lambda_{min}}\right)}{C_k\left(1 + 2\frac{\lambda_{min}}{\lambda_{max} - \lambda_{min}}\right)} \right|$$

The Chebyshev Polynomial $C_k(x) \equiv \cos(k \cos^{-1}(x)) \quad x \in [-1,1]$

Chebyshev Polynomials minimizing over [1,10]



Chebyshev Bounds

$$\min_{\substack{k^{th}\text{ order polys.t.}\\ \tilde{\varphi}_k(0)=1}} \max_{x\in[\lambda_{min},\lambda_{max}]} \tilde{\varphi}_k(x) = \max_{x\in[\lambda_{min},\lambda_{max}]} \frac{C_k\left(1+2\frac{\lambda_{min}-x}{\lambda_{max}-\lambda_{min}}\right)}{C_k\left(1+2\frac{\lambda_{min}}{\lambda_{max}-\lambda_{min}}\right)}$$

$$= \frac{1}{\left|C_k\left(1-2\frac{\lambda_{max}}{\lambda_{max}-\lambda_{min}}\right)\right|}$$

$$\le 2\left(\frac{\sqrt{\frac{\lambda_{max}}{\lambda_{min}}}-1}{\sqrt{\frac{\lambda_{max}}{\lambda_{min}}}+1}\right)^k$$

Example.

Given the ratio $\frac{\lambda_{max}}{\lambda_{min}} = 4$, what is the residual after k iterations?

$$\|r^k\| \le 2\left(\frac{\sqrt{4}-1}{\sqrt{4}+1}\right)^k \|r^0\| \Rightarrow \|r^k\| \le \frac{2}{3^k}\|r^0\|$$

Given the ratio $\frac{\lambda_{max}}{\lambda_{min}} = 100$, what is the residual after k iterations?

$$\|r^k\| \le 2\left(\frac{\sqrt{100}-1}{\sqrt{100}+1}\right)^k \|r^0\| \Rightarrow \|r^k\| \le 2\left(\frac{9}{11}\right)^k \|r^0\|$$

Given the ratio $\frac{\lambda_{max}}{\lambda_{min}} = 10{,}000$, what is the residual after k iterations?

$$\|r^k\| \le 2\left(\frac{\sqrt{10{,}000}-1}{\sqrt{10{,}000}+1}\right)^k \|r^0\| \Rightarrow \|r^k\| \le 2\left(\frac{99}{101}\right)^k \|r^0\|$$

Which case converges faster?

If the ratio of eigenvalues is large, than the bound on the convergence will be slower.

Chebyshev Result

If $\lambda(\mathbf{M}) \in [\lambda_{min}, \lambda_{max}]$, $\lambda_{min} > 0$

$$\|r^k\| \le 2\left(\frac{\sqrt{\frac{\lambda_{max}}{\lambda_{min}}}-1}{\sqrt{\frac{\lambda_{max}}{\lambda_{min}}}+1}\right)^k \|r^0\|$$

$$cond(\mathbf{M}) = \|\mathbf{M}\|\|\mathbf{M}^{-1}\| = \lambda_{max}\cdot\frac{1}{\lambda_{min}}$$

$$\mathbf{M} = \mathbf{U}\Lambda\mathbf{U}^{-1} = \mathbf{U}\Lambda\mathbf{U}^T \qquad \lambda_{max} = \|\mathbf{M}\|_2$$

$$\mathbf{M}^{-1} = \mathbf{U}\Lambda^{-1}\mathbf{U}^{-1} = \mathbf{U}\Lambda^{-1}\mathbf{U}^T \qquad \lambda_{min} = ?$$

$$\|\mathbf{M}^{-1}\|_2 = \max_i \frac{1}{\lambda_i} = \frac{1}{\lambda_{min}} \quad \Rightarrow \lambda_{min} = \left[\|\mathbf{M}^{-1}\|_2\right]^{-1}$$

$$\frac{\lambda_{max}}{\lambda_{min}} = \|\mathbf{M}\|_2\|\mathbf{M}^{-1}\|_2 = cond(\mathbf{M})$$

Examples.

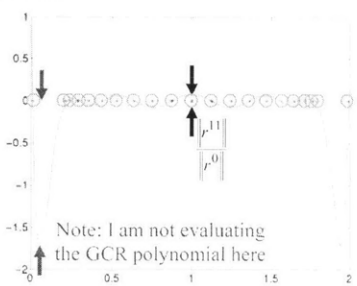
$cond(\mathbf{M}) = 100$ $\quad \dfrac{\|r^k\|}{\|r^0\|} \le 2\left(\dfrac{10-1}{10+1}\right)^k = 2\left(\dfrac{9}{11}\right)^k$ Worst Case (bound) $\quad$ At each iteration, improves by ~10%

$cond(\mathbf{M}) = 10{,}000$ $\quad \dfrac{\|r^k\|}{\|r^0\|} \le 2\left(\dfrac{100-1}{100+1}\right)^k$

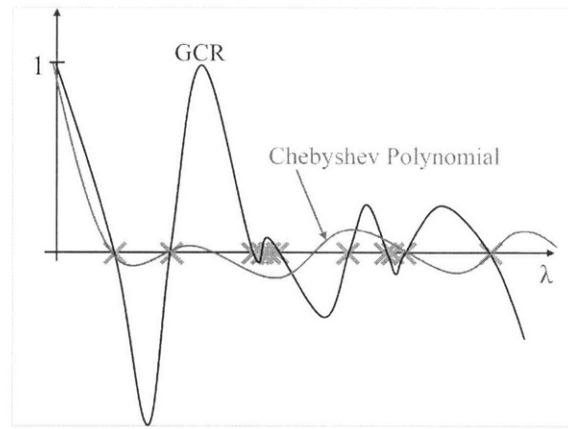
$cond(\mathbf{M}) = 10^{12}$ $\quad \dfrac{\|r^k\|}{\|r^0\|} \le 2\left(\dfrac{10^6-1}{10^6+1}\right)^k$ $\quad$ At each iteration, improves by a very, very small amount.

Heat Conducting Bar example

Sometimes GCR can do much better than Chebyshev bound



Residual Minimizing Krylov-subspace Algorithm can eliminate outlying eigenvalues by placing polynomial zeros directly on them.



Gershgorin Circle Theorem

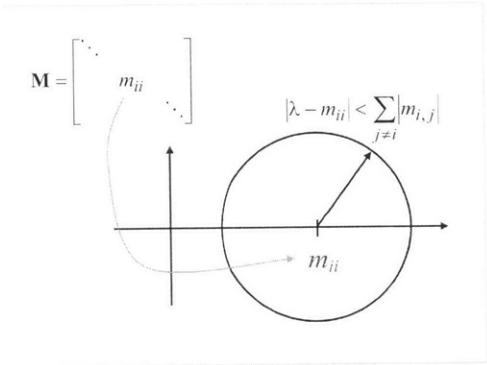
- Theorem Statement

Given a matrix $\mathbf{M} = \begin{bmatrix} m_{1,1} & \cdots & m_{1,N} \\ \vdots & \ddots & \vdots \\ m_{N,1} & \cdots & m_{N,N} \end{bmatrix}$

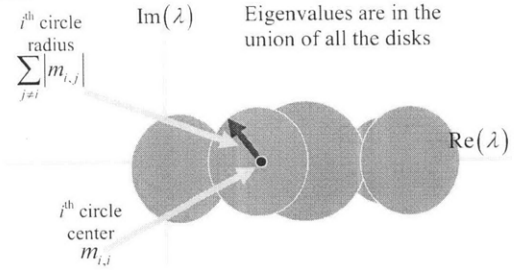
For each eigenvalue of $\mathbf{M}$ there exists an i, $1 < i < N$ such that

$$|\lambda - m_{i,i}| \le \sum_{j \ne i} |m_{i,j}|$$

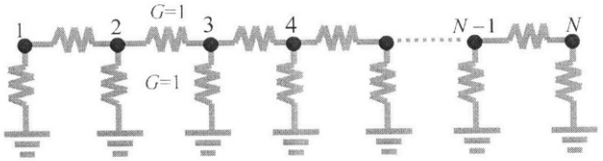
We say that the eigenvalues are contained in the union of the Gershgorin circles

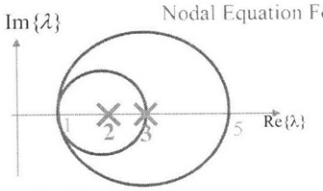


- Picture



- Grounded Resistor Line – Nodal Matrix





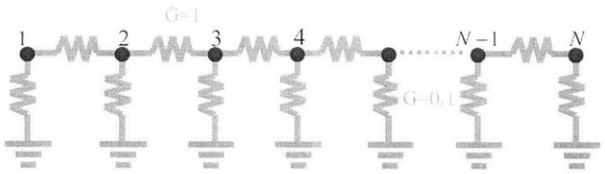
Nodal Equation Form

$$\underbrace{\begin{bmatrix} 2 & -1 & & \\ -1 & 3 & & \\ & & \ddots & -1 \\ & & -1 & 2 \end{bmatrix}}_{M} \Big\updownarrow N$$

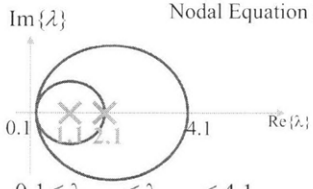
$1 \le \lambda_{\min} \le \lambda_{\max} \le 5$

$$\frac{\|r^k\|}{\|r^0\|} \le 2\left[\frac{\sqrt{\frac{5}{1}}-1}{\sqrt{\frac{5}{1}}+1}\right]^k = 2\left[\frac{1}{3.8}\right]^k$$

about one digit every two iterations
about 0.1% error after 6 iterations



Nodal Equation Form



$$N \begin{bmatrix} 1.1 & -1 & & \\ -1 & 2.1 & & \\ & & \ddots & -1 \\ & & -1 & 2.1 \end{bmatrix}$$

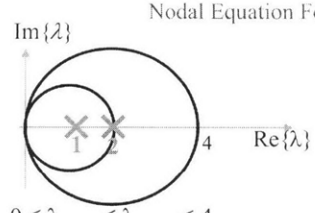
M

$0.1 \le \lambda_{min} \le \lambda_{max} \le 4.1$

about one digit every 6 iterations
about 0.1% error after 18 iterations

$$\frac{\|r^k\|}{\|r^0\|} \le 2\left[\frac{\sqrt{\frac{4.1}{0.1}}-1}{\sqrt{\frac{4.1}{0.1}}+1}\right]^k = 2\left[\frac{1}{1.4}\right]^k$$

Nodal Equation Form



$$N \begin{bmatrix} 1 & -1 & & \\ -1 & 2 & & \\ & & \ddots & -1 \\ & & -1 & 2.1 \end{bmatrix}$$

M

$0 \le \lambda_{min} \le \lambda_{max} \le 4$

This does not mean that
GCR cannot converge in this case
but it is a good hint that we
might expect slow convergence

$$\frac{\|r^k\|}{\|r^0\|} \le 2\left[\frac{\sqrt{\frac{4}{0}}-1}{\sqrt{\frac{4}{0}}+1}\right]^k \le 2$$

3.10 Linear Systems - GCR Preconditioners

INTRODUCTION TO NUMERICAL SIMULATION

LECTURE 10.
Preconditioners

TODAY'S OUTLINE:

- Preconditioners
  - Diagonal Preconditioner
  - Blockdiagonal Preconditioner
  - Incomplete Factorization Preconditioner
- GCR for different Right Hand Sides
  - Recycling the Krylov subspace

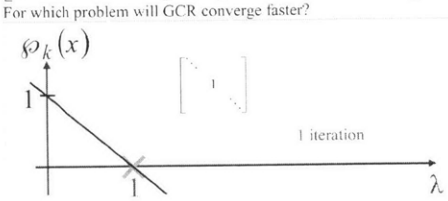
PRECONDITIONERS

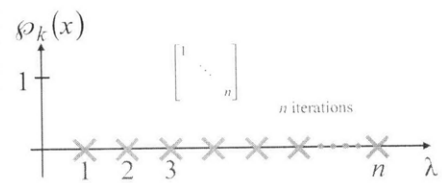
Diagonal Preconditioner

- Diagonal Example

$$\begin{bmatrix} 1 & 0 & 0 & \cdots & 0 \\ 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & \ddots & \ddots & \vdots \\ \vdots & \vdots & \ddots & 1 & 0 \\ 0 & \cdots & 0 & 0 & 1 \end{bmatrix} \qquad \begin{bmatrix} 1 & 0 & 0 & \cdots & 0 \\ 0 & 2 & 0 & \cdots & 0 \\ 0 & 0 & \ddots & \ddots & \vdots \\ \vdots & \vdots & \ddots & N-1 & 0 \\ 0 & \cdots & 0 & 0 & N \end{bmatrix}$$

For which problem will GCR converge faster?





Suppose $\mathbf{M}\vec{x} = \vec{b}$ converges slowly

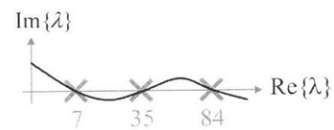
Try $\mathbf{PM}\vec{x} = \mathbf{P}\vec{b}$ for some $\mathbf{P}$ (left pre - conditioner matrix)

e.g. How many iterations for this diagonal $\mathbf{M}$ to converge?

$$\mathbf{M} = \mathbf{D} = \begin{bmatrix} 1 & 0 & 0 & \cdots & 0 \\ 0 & 2 & 0 & \cdots & 0 \\ 0 & 0 & 3 & \ddots & \vdots \\ \vdots & \vdots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & 0 & N \end{bmatrix} \qquad \mathbf{P} = \begin{bmatrix} 1 & 0 & 0 & \cdots & 0 \\ 0 & \frac{1}{2} & 0 & \cdots & 0 \\ 0 & 0 & \frac{1}{3} & \ddots & \vdots \\ \vdots & \vdots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & 0 & \frac{1}{N} \end{bmatrix}$$

Can we find a preconditioner $\mathbf{P}$ that does better?

$$M = \begin{bmatrix} 7 & & & & & & & \\ & \ddots & & & & & & \\ & & 7 & & & & & \\ & & & 35 & & & & \\ & & & & \ddots & & & \\ & & & & & 35 & & \\ & & & & & & 84 & \\ & & & & & & & \ddots \\ & & & & & & & & 84 \end{bmatrix} \qquad \frac{\|r^3\|}{\|r^0\|} = 0$$



❖ Diagonally Dominant Example

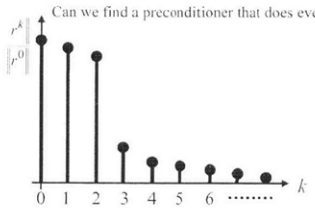
How many iterations for this diagonally dominant **M** to "get very close" to the exact answer?

$$M = D + \delta = \begin{bmatrix} 7 & * & * & * & * & * & * & * & * \\ * & \ddots & * & * & * & * & * & * & * \\ * & * & 7 & * & * & * & * & * & * \\ * & * & * & 35 & * & * & * & * & * \\ * & * & * & * & \ddots & * & * & * & * \\ * & * & * & * & * & 35 & * & * & * \\ * & * & * & * & * & * & 84 & * & * \\ * & * & * & * & * & * & * & \ddots & * \\ * & * & * & * & * & * & * & * & 84 \end{bmatrix}$$

* is a very small entry compared to diagonals

$\left\|\frac{r^3}{r^0}\right\| \approx$ small

Can we find a preconditioner that does even better?

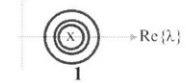


❖ Diagonal Preconditioner

Try as preconditioner the inverse of the diagonal

$$D^{-1}(D+\delta) = \begin{bmatrix} 1 & * & * & * & * & * & * & * & * \\ * & \ddots & * & * & * & * & * & * & * \\ * & * & 1 & * & * & * & * & * & * \\ * & * & * & 1 & * & * & * & * & * \\ * & * & * & * & \ddots & * & * & * & * \\ * & * & * & * & * & 1 & * & * & * \\ * & * & * & * & * & * & 1 & * & * \\ * & * & * & * & * & * & * & \ddots & * \\ * & * & * & * & * & * & * & * & 1 \end{bmatrix}$$

$\left\|\frac{r^1}{r^0}\right\| \approx$ small



❖ General Idea

Suppose $\mathbf{M}\bar{x} = \bar{b}$ converges slowly

Try $\mathbf{PM}\bar{x} = \mathbf{P}\bar{b}$ for some **P** (left pre - conditioner matrix)

If $\mathbf{PM} = I$ then convergence happens in one step

however $\mathbf{P} = \mathbf{M}^{-1}$ is VERY hard to compute

If $\mathbf{PM} \approx I$ then we hope convergence happens in "few" steps

Any general idea for picking **P**?

Pick $\tilde{\mathbf{M}}$ such that :

a) $\tilde{\mathbf{M}} \approx \mathbf{M}$

b) $\tilde{\mathbf{M}}$ is easy to invert or factor

Preconditioner : $\mathbf{P} = \tilde{\mathbf{M}}^{-1}$

e.g. if **M** is diaonally dominant

its diagonal $\tilde{\mathbf{M}} = \mathrm{diag}(\mathbf{M})$ is :

- a good approximation of **M**

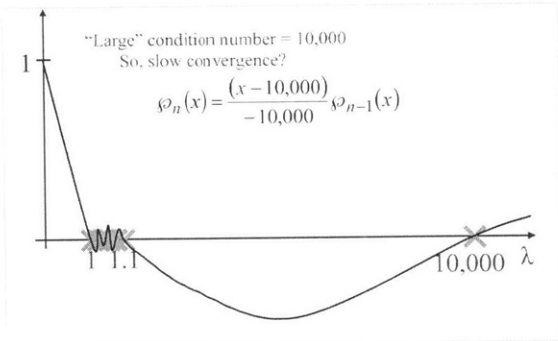
- and easy to invert

Let $\mathbf{A} = \mathbf{D} + \mathbf{A}_{nd}$

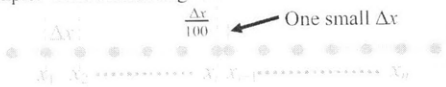
Apply GCR to

$$\left(\mathbf{D}^{-1}\mathbf{A}\right)\bar{x} = \left(I + \mathbf{D}^{-1}\mathbf{A}_{nd}\right)\bar{x} = \mathbf{D}^{-1}\bar{b}$$

- The inverse of a diagonal is cheap to compute
- Usually improves convergence



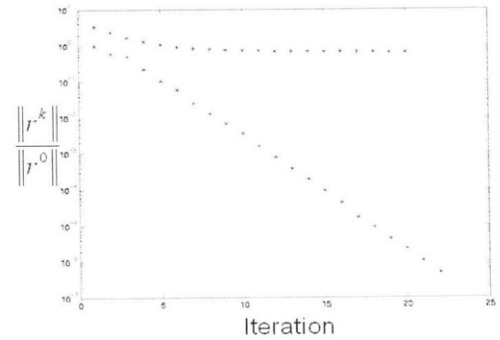
Example. Heat conducting bar



$$\begin{bmatrix} 2+\gamma & -1 & & & & & \\ -1 & 2+\gamma & & & & & \\ & & \ddots & & & & \\ & & -1 & 1+\gamma+100 & -100 & & \\ & & & -100 & 1+\gamma+100 & -1 & \\ & & & & -1 & \ddots & -1 \\ & & & & & -1 & 2+\gamma \end{bmatrix} \begin{bmatrix} \hat{u}_1 \\ \\ \\ \vdots \\ \\ \\ \hat{u}_n \end{bmatrix} = \begin{bmatrix} f(x_1) \\ \\ \\ \vdots \\ \\ \\ f(x_n) \end{bmatrix}$$

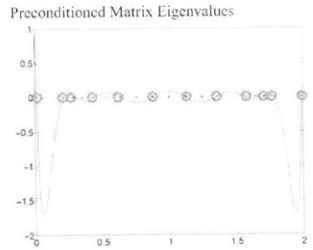
$$\frac{\lambda_{\max}}{\lambda_{\min}} > 100$$

For the heat conducting bar, which convergence curve is GCR?



Heat Conducting Bar example continued…

Preconditioned Matrix Eigenvalues



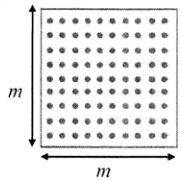
Residual Minimizing Krylov-subspace Algorithm can eliminate outlying eigenvalues by placing polynomial zeros directly on them.

Heat Flow Comparison

1-D

m unknowns $O(m)$ nonzeros

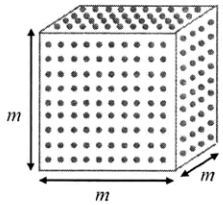
2-D Plate



m^2 unknowns

$O(m^2)$ nonzeros

3-D Cube



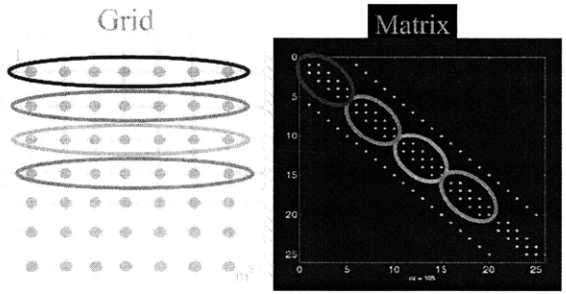
m^3 unknowns

$O(m^3)$ nonzeros

Dimension	Dense GE	Sparse GE	GCR
1	$O(m^3)$	$O(m)$	$O(m^2)$
2	$O(m^6)$	$O(m^3)$	$O(m^3)$
3	$O(m^9)$	$O(m^6)$	$O(m^4)$

Blockdiagonal Preconditioner

- Line Schemes

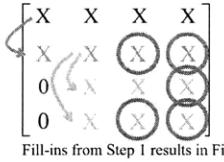


Tridiagonal Matrices factor quickly

Incomplete Factorization Preconditioner

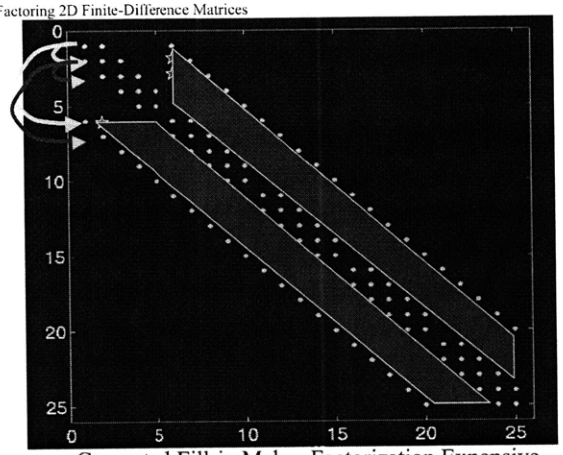
- Fill-Ins Example

Fill-ins Propagate



Fill-ins from Step 1 results in Fill-ins in step 2.

❖ Factoring 2D Finite-Difference Matrices



Generated Fill-in Makes Factorization Expensive

❖ Key Idea

- Throw away fill-ins
  - Throw away all fill-ins
  - Throw away only fill-ins with small values
  - Throw away fill-ins produced by other fill-ins
  - Throw away fill-ins produced by fill-ins of other fill-ins, etc.

- Key Idea

  Pick $\widetilde{\mathbf{M}}$ such that :

  a.) Pick $\widetilde{\mathbf{M}} \approx \mathbf{M}$

  b.) $\widetilde{\mathbf{M}}$ is easy to factor : $\widetilde{\mathbf{M}} = \widetilde{\mathbf{L}}\widetilde{\mathbf{U}}$

  Use as preconditioner : $\mathbf{P} = \widetilde{\mathbf{M}}^{-1}$

  $\mathbf{M}\vec{x} = \vec{b}$

  $\widetilde{\mathbf{M}}^{-1}\mathbf{M}\vec{x} = \widetilde{\mathbf{M}}^{-1}\vec{b}$

  But do NOT calculate $\mathbf{P} = \widetilde{\mathbf{M}}^{-1}$!!!!!!!!!!!!!!!!

OR do NOT calculate $\widetilde{\mathbf{M}}^{-1}\mathbf{M}$!!!!!!!!!!!!!!!!

$$\widetilde{\mathbf{M}}^{-1}\mathbf{M}\vec{x} = \underbrace{\widetilde{\mathbf{M}}^{-1}\vec{b}}_{\vec{b} = \widetilde{\mathbf{M}}^{-1}\vec{b}} \qquad \widetilde{\mathbf{M}} = \widetilde{\mathbf{L}}\widetilde{\mathbf{U}}$$

solve for $\vec{b}$: $\widetilde{\mathbf{L}}\widetilde{\mathbf{U}}\vec{b} = \vec{b}$

At each iteration of GCR we need : $\widetilde{\mathbf{M}}^{-1}\mathbf{M}r^k = \hat{r}^k$

Then solve for $\hat{r}^k$: $\widetilde{\mathbf{L}}\widetilde{\mathbf{U}}\hat{r}^k = \mathbf{M}r^k$

$\widetilde{\mathbf{M}}^{-1}\vec{b} = \vec{\tilde{b}}$

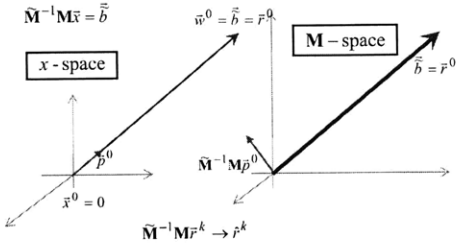
$\widetilde{\mathbf{M}} = \widetilde{\mathbf{L}}\widetilde{\mathbf{U}}$

$\widetilde{\mathbf{M}}\vec{\tilde{b}} = \vec{b}$ — Solve $\widetilde{\mathbf{L}}\vec{y} = \vec{b} \rightarrow \vec{y}$

$\widetilde{\mathbf{L}}\underbrace{\widetilde{\mathbf{U}}\vec{\tilde{b}}}_{\vec{y}} = \vec{b}$ — Solve $\widetilde{\mathbf{U}}\vec{\tilde{b}} = \vec{y} \rightarrow \vec{\tilde{b}}$

$\widetilde{\mathbf{M}}^{-1}\mathbf{M} \rightarrow ?$ *using GCR*

$\widetilde{\mathbf{M}}^{-1}\mathbf{M}\vec{x} = \vec{\tilde{b}}$



$\widetilde{\mathbf{M}}^{-1}\mathbf{M}\vec{r}^k \rightarrow \hat{r}^k$

$\widetilde{\mathbf{M}} = \widetilde{\mathbf{L}}\widetilde{\mathbf{U}}$

$\widetilde{\mathbf{M}}^{-1}\mathbf{M}\vec{r}^k = \hat{r}^k \quad \Rightarrow \underbrace{\widetilde{\mathbf{M}}}_{\widetilde{\mathbf{L}}\widetilde{\mathbf{U}}}\hat{r}^k = \mathbf{M}\vec{r}^k$

- Complete Algorithm

  Pick $\widetilde{\mathbf{M}}$ such that :

  a.) Pick $\widetilde{\mathbf{M}} \approx \mathbf{M}$

  b.) $\widetilde{\mathbf{M}}$ is easy to factor : $\widetilde{\mathbf{M}} = \widetilde{\mathbf{L}}\widetilde{\mathbf{U}}$

  Factor $\widetilde{\mathbf{M}} = \widetilde{\mathbf{L}}\widetilde{\mathbf{U}}$

  Calculate new RHS solving for $\widetilde{b}$: $\widetilde{\mathbf{L}}\widetilde{\mathbf{U}}\widetilde{b} = \vec{b}$

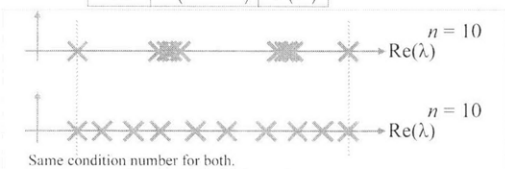
  At the kth step of GCR:

  Calculate $\mathbf{M}r^k$

  solve for $\tilde{r}^k$: $\widetilde{\mathbf{L}}\widetilde{\mathbf{U}}\tilde{r}^k = \mathbf{M}r^k$

- GCR Summary
  - Comparison CGR & GE
    - Making GCR converge fast can be very problem specific: pre-conditioning
    - Assume we have a good pre-conditioner (e.g. # iterations $k < 10,20$)

	Sparse	Dense
GCR	$O(n)$	$O(n^2)$
GE	$O(n^{1.2-1.8})$	$O(n^3)$



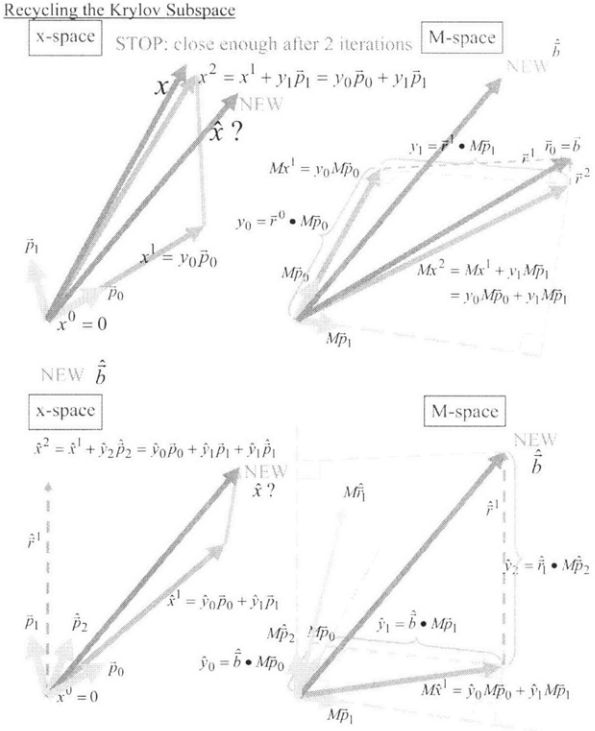
Same condition number for both.
Top converges faster – clustered eigenvalues.

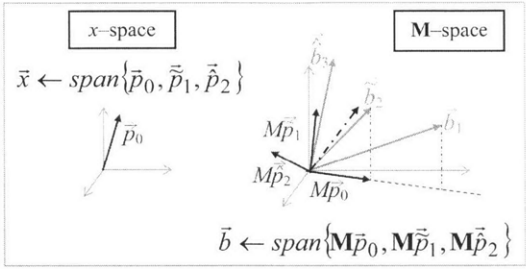
  - When would one use GCR?
    - Need to have fast convergence rate:
      1. *cond* (**M**) is not large
      2. or eigenvalues of **M** are clustered in few groups
      3. or we have a good "pre-conditioner"
    - And one of the following:
      1. need fewer than 16 digits of precision
      2. or **M** is dense
      3. or have a fast matrix-vector product algorithm

    And what about if you need to re-solve a system with different RHS?

GCR FOR DIFFERENT RIGHT HAND SIDES

Recycling the Krylov Subspace




x–space
M–space
$\vec{x} \leftarrow span\{\vec{p}_0, \vec{\tilde{p}}_1, \vec{\tilde{p}}_2\}$
$\vec{p}_0$
$\vec{\tilde{b}}_3$
$\vec{\tilde{b}}_2$
$\vec{\tilde{b}}_1$
$M\vec{\tilde{p}}_1$
$M\vec{\tilde{p}}_2$
$M\vec{p}_0$
$\vec{b} \leftarrow span\{\mathbf{M}\vec{p}_0, \mathbf{M}\vec{\tilde{p}}_1, \mathbf{M}\vec{\tilde{p}}_2\}$

3.11 NonLinear Systems - Newton Method

INTRODUCTION TO NUMERICAL SIMULATION

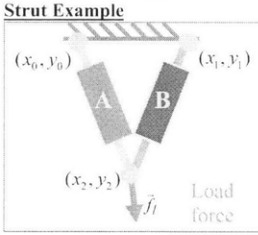
LECTURE 11.
1-D Nonlinear Solution Methods

TODAY'S OUTLINE:

- Nonlinear Problems
  - Struts and Circuit Example
- Richardson and Linear Convergence
- Newton's Method
  - Derivation of the basic algorithm
  - Convergence Analysis
  - Multidimensional Newton's Method

NONLINEAR PROBLEMS

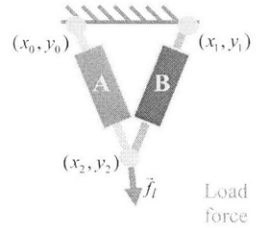
Strut Example



Given: x_0, y_0, x_1, y_1, f_l

Find: x_2, y_2

Need to Solve: $\sum f_x = 0$, $\sum f_y = 0$



$$f^*_{A,x} + f^*_{B,x} + f_{l,x} = 0$$

$$f^*_{A,y} + f^*_{B,y} + f_{l,y} = 0$$

$$f^*_{A,x} = \frac{x_2 - x_0}{L_A}\,\varepsilon_A\left(L_{A,0} - L_A\right)$$

$$f^*_{A,y} = \frac{y_2 - y_0}{L_A}\,\varepsilon_A\left(L_{A,0} - L_A\right)$$

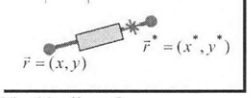
$$L_A = \sqrt{(x_2 - x_0)^2 + (y_2 - y_0)^2}$$

$$f^*_{B,x} = \frac{x_2 - x_1}{L_B}\,\varepsilon_B\left(L_{B,0} - L_B\right)$$

$$f^*_{B,y} = \frac{y_2 - y_1}{L_B}\,\varepsilon_B\left(L_{B,0} - L_B\right)$$

$$L_B = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

$$\vec{f}^* = \frac{\vec{r}^* - \vec{r}}{\left|\vec{r}^* - \vec{r}\right|}\,\varepsilon\left(L_0 - \left|\vec{r}^* - \vec{r}\right|\right)$$



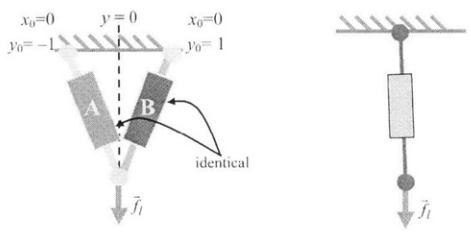
Why Nonlinear?

Pull Hard on the Struts

The strut forces change in both magnitude and direction

$$f^*_{A,y} + f^*_{B,y} + f_{l,y} = 0$$

$$\frac{y_2 - y_0}{L_A}\,\varepsilon_A\left(L_0 - L_A\right) + \frac{y_2 - y_1}{L_B}\,\varepsilon_B\left(L_0 - L_B\right) + f_{l,y} = 0$$



2 strut problem

Struts stretch & rotate

1 strut problem

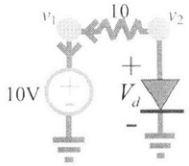
$f_L = \varepsilon\underbrace{(L_0 - L)}_{\Delta L}$

$\Delta L = \frac{f_L}{\varepsilon}$

linear

Circuit Example

Need to Solve



$I_d + I_r = 0$

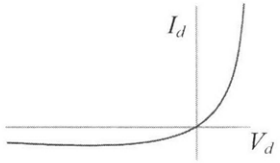
$I_{v_{oc}} - I_r = 0$

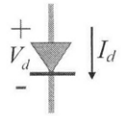
Branch Equations

$I_r - \frac{1}{10}V_r = 0$

$I_d - I_s(e^{V_d/V_t} - 1) = 0$

Non-Linear System of Equations $f(x) = 0$





Solve Iteratively

Hard to find analytical solution for $f(x) = 0$

Solve iteratively

guess a solution $x^0 = x_0$

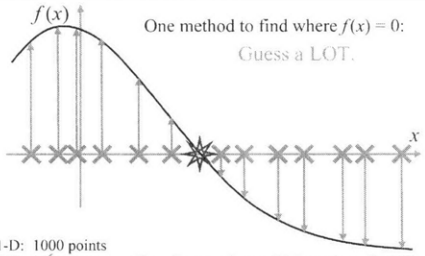
repeat for $k = 0, 1, 2, \ldots$

$x^{k+1} = W(x^k)$

until $f(x^{k+1}) \approx 0$

Ask

- Does the iteration converge to correct solution?
- How fast does the iteration converge?



One method to find where $f(x) = 0$:

Guess a LOT.

1-D: 1000 points
2-D: 10^6 points
3-D: 10^9 points

Guessing may be an alright strategy for 1-D case, but not so when there are more dimensions.

RICHARDSON AND LINEAR CONVERGENCE

Richardson Iteration

- Definition

Richardson Iteration Definition

$$x^{k+1} = x^k + f(x^k)$$

An iteration stationary point is a solution

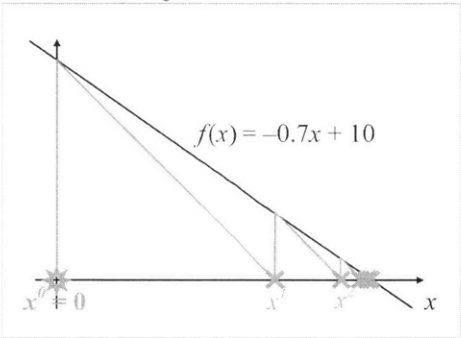
$$x^{k+1} = x^k$$
$$\Rightarrow f\left(x^k\right) = 0$$
$$\Rightarrow x^k = x^* \quad \text{(Solution)}$$

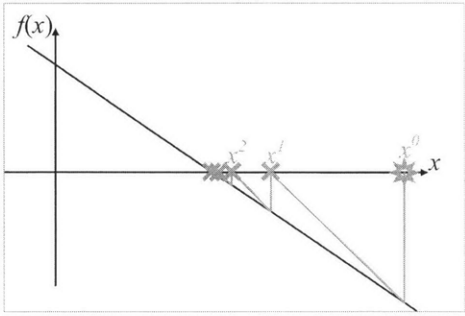
- Example 1

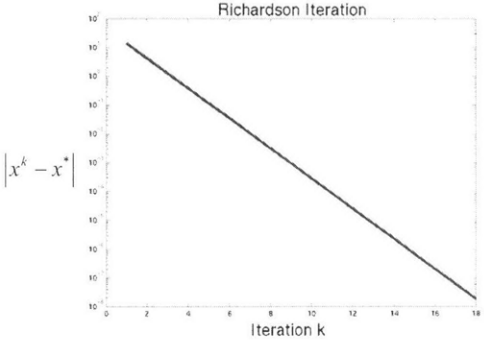
$$f(x) = -0.7x + 10$$

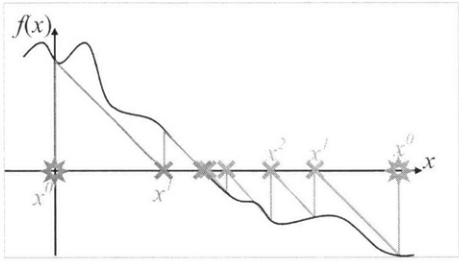
Start with $x^0 = 0$

$$x^1 = x^0 + f\left(x^0\right) = 0 + 10 = 10$$
$$x^2 = x^1 + f\left(x^1\right) = 10 + (-7 + 10) = 13$$
$$x^3 = x^2 + f\left(x^2\right) = 13 + (-0.7 \cdot 13 + 10) = 13.9$$
$$x^4 = x^3 + f\left(x^3\right) = 13.9 + (-0.7 \cdot 13.9 + 10) = 14.17$$
$$x^5 = 14.25$$
$$x^6 = 14.27$$
$$x^7 = 14.28$$
$$x^8 = 14.28 \longleftarrow \text{Converged!}$$









- Example 2

  $f(x) = 2x + 10$

  Start with $x^0 = 0$

  $x^1 = x^0 + f(x^0) = 0 + 10 = 10$

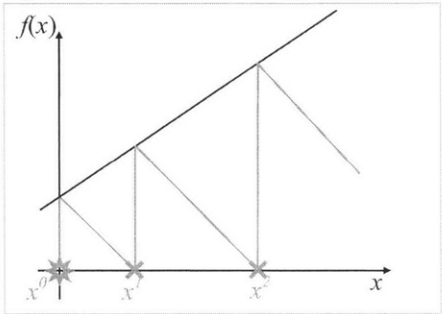
  $x^2 = x^1 + f(x^1) = 10 + (2 \cdot 10 + 10) = 40$

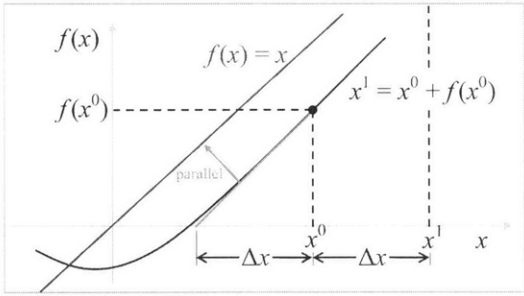
  $x^3 = x^2 + f(x^2) = 40 + (2 \cdot 40 + 10) = 130$

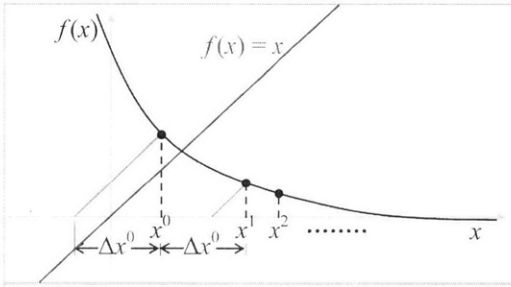
  $x^4 = x^3 + f(x^3) = 130 + (2 \cdot 130 + 10) = 400$

  $\vdots$

  No Convergence!!







$$f(x) = \underset{\text{given}}{b} - \mathbf{A}x$$

$$x^{k+1} = x^k + f(x) = x^k + b - \mathbf{A}x^k$$

$$x^* = x^* + \underbrace{f(x^*)}_{0} = x^* + \underbrace{b - \mathbf{A}x^*}_{0}$$

$$x^{k+1} - x^* = (\mathbf{I} - \mathbf{A})(x^k - x^*)$$

$$x^k - x^* = (\mathbf{I} - \mathbf{A})^k (x^0 - x^*)$$

$$x^1 - x^* = (\mathbf{I} - \mathbf{A})(x^0 - x^*)$$

$$x^2 - x^* = (\mathbf{I} - \mathbf{A})(x^1 - x^*) = (\mathbf{I} - \mathbf{A})^2 (x^0 - x^*)$$

$$\lim_{k \to \infty} \underbrace{(\mathbf{I} - \mathbf{A})^k}_{\substack{\text{spectral} \\ \text{radius}}} \to 0 \quad < 1$$

$$|(1 - \lambda(\mathbf{A}))| < 1$$

- Convergence

$$x^{k+1} = x^k + f(x^k)$$

$$x^* = x^* + \underbrace{f(x^*)}_{=0}$$

$$f(x) = f(x^*) + \frac{df}{dx}(x^*)(x - x^*) + \frac{1}{2}\frac{d^2 f}{dx^2}(x^*)(x - x^*)^2 + \cdots$$

$$x^{k+1} - x^* = x^k - x^* + f(x^*) + \frac{df}{dx}(x^*)(x^k - x^*) + \frac{1}{2}\frac{d^2 f}{dx^2}(x^*)(x^k - x^*)^2 + \ldots$$

$$|x^{k+1} - x^*| = \underbrace{\left|1 + \frac{df}{dx}(x^*) + \frac{1}{2}\frac{d^2 f}{dx^2}(x^*)(x^k - x^*) + \ldots\right|}_{K^k} |x^k - x^*|$$

Find a bound: $K^k \le K^*$ $\quad x^k \to x^*$

$$K^k \le \left|1 + \frac{\partial f}{\partial x}(x^*) + \varepsilon\right| \ ?$$

$$|x^{k+1} - x^*| = K^* |x^k - x^*| \quad \text{for large } k$$

If $K^* < 1$ then Richardson converges linearly.

$$|x^{k+1} - x^*| \le \tfrac{1}{2}|x^k - x^*|$$

$$|x^k - x^*| = 10$$

$$|x^{k+1} - x^*| \le \tfrac{1}{2} \cdot 10 = 5$$

$$|x^{k+10} - x^*| \le \left(\tfrac{1}{2}\right)^{10} \cdot 10$$

- Setup

Iteration Equation $\quad x^{k+1} = x^k + f(x^k)$

Exact Solution $\quad x^* = x^* + \underbrace{f(x^*)}_{=0}$

Computing Differences

$$x^{k+1} - x^* = x^k - x^* + \underbrace{f(x^k) - f(x^*)}_{\text{Need to Estimate}}$$

$$\left|x^{k+1}-x^*\right| = g\left(x^k - x^*\right)$$

$$\left|x^{k+1}-x^*\right| = \gamma\left|x^k - x^*\right|$$

$$\gamma \le 1$$

$$K^* = \left|1+\frac{\partial f}{\partial x}\left(x^*\right)+\frac{1}{2}\frac{\partial^2 f}{\partial x^2}\left(x-x^*\right)+\varepsilon\right|$$

Example 1. $f(x) = -0.7x + 10$

$$K^* = |1+(-0.7)+0+\varepsilon| = |0.3+\varepsilon|$$

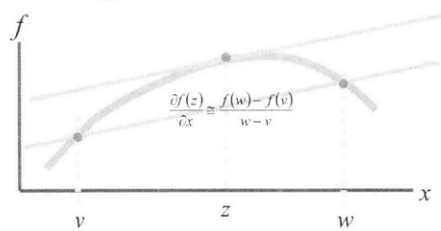
Example 2. $f(x) = 2x + 10$

$$K^* = 1+2+0+\varepsilon| = |3+\varepsilon|$$

No convergence

- Mean Value Theorem

$$f(w) - f(v) = \frac{\delta f(z)}{\delta x}(w-v) \qquad z \in [v, w]$$



- Use Mean Value Theorem

Iteration Equation $x^{k+1} = x^k + f\left(x^k\right)$

Exact Solution $x^* = x^* + \underbrace{f\left(x^*\right)}_{=0}$

Computing Differences

$$x^{k+1} - x^* = x^k - x^* + f\left(x^k\right) - f\left(x^*\right) = \left(1+\frac{\partial f(\tilde{x})}{\partial x}\right)\left(x^k - x^*\right)$$

$$\tilde{x} \in \left[x^k, x^*\right]$$

$$\begin{aligned}
\left|x^{k+1}-x^*\right| &= \left|x^k - x^* + \frac{\partial f}{\partial x}(\tilde{x})\left(x^k - x^*\right)\right| \\
&= \underbrace{\left|1+\frac{\partial f}{\partial x}(\tilde{x})\right|}_{\le\gamma}\left(x^k - x^*\right) \\
&\le \gamma\left(x^k - x^*\right) \\
&\le \gamma\gamma\left(x^{k-1} - x^*\right) \\
&\le \gamma^k\left(x^0 - x^*\right)
\end{aligned}$$

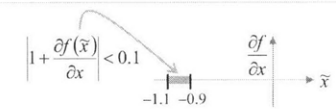
- Richardson Theorem

IF $\left|1+\frac{\partial f(\tilde{x})}{\partial x}\right| \le \gamma < 1$ for all $\tilde{x}$ s.t. $\left|\tilde{x} - x^*\right| < \delta$

AND $\left|x^0 - x^*\right| < \delta$

THEN $\left|x^{k+1} - x^*\right| \le \gamma\left|x^k - x^*\right|$

OR $\lim_{k\to\infty}\left|x^{k+1} - x^*\right| = \lim_{k\to\infty}\left|x^0 - x^*\right| = 0$



$$x^0 - x^* = 1$$

$$\left|x^1 - x^*\right| < 0.1$$

$$\left|x^1 - x^*\right| < 0.01$$

$$\vdots$$

1 digit/iteration → linear convergence

Example 1. $f(x) = -0.7x + 10$

$$\gamma = \left|1 + \frac{\partial f}{\partial x}\right| = |1 + (-0.7)| = 0.3 < 1$$

Convergence

Example 2. $f(x) = 2x + 10$

$$\gamma = \left|1 + \frac{\partial f}{\partial x}\right| = |1 + 2| = 3 > 1$$

No convergence

- Problems
  - Convergence is only linear
  - x, $f(x)$ not in the same units:
    - x is a voltage, $f(x)$ a current in circuits
    - x is a displacement, $f(x)$ a force in struts
    - adding two different physical quantities

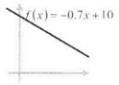
$$x^{k+1} \leftarrow x^k + f(x^k)$$

V voltage	i current
T temperature	h heat flow
x position	F force

- Advantage
  - Simple – only need to calculate $f(x)$ and update

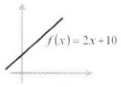
$$x^{k+1} \leftarrow x^k + \alpha f(x^k)$$

Example 1.



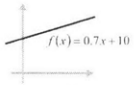
$\alpha = 1$
$\gamma = |1 - 0.7| = 0.3$

Example 2.

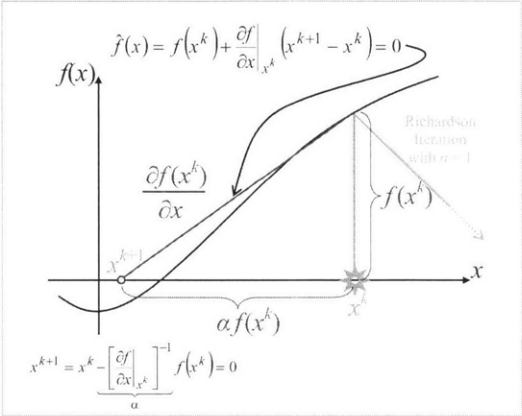


$\alpha = 1$
$\longrightarrow$ No convergence!!
$\alpha = -1$
$\gamma = |1 - 2| = 1$
$\longrightarrow$ Still no convergence!!
$\alpha = ?$

Example 3.



$\alpha = -1$
$\gamma = |1 - 0.7| = 0.3$



NEWTON'S METHOD

Derivation of the Basic Algorithm

- 1-D Reminder
  - Newton Idea

Problem : Find x^* such that $f(x^*) = 0$

Approximate $f(x)$ with its Taylor series about x^k :

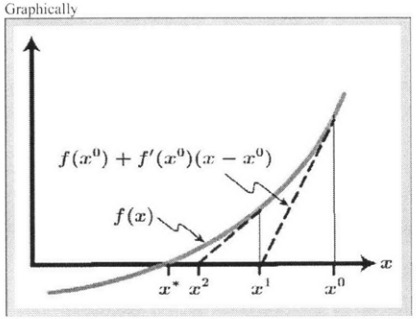
$$\hat{f}(x) \cong f(x^k) + \frac{\delta f}{\delta x}(x^k)(x - x^k) \quad \text{a straight line}$$

Find the solution of the approximation x^{k+1} :

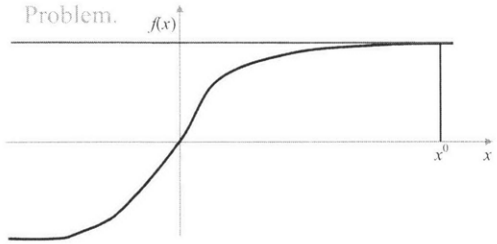
$$f(x^k) + \frac{\delta f}{\delta x}(x^k)(x^{k+1} - x^k) = 0$$

$$\Rightarrow x^{k+1} = x^k - \left[\frac{\partial f}{\partial x}(x^k)\right]^{-1} f(x^k)$$

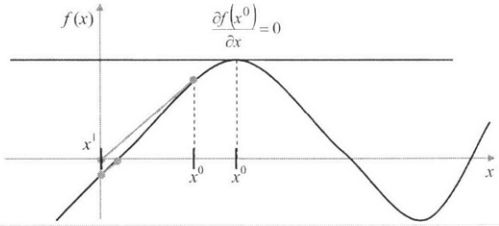
- Graphically



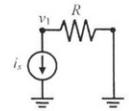
Problem.



Problem.



Example.



$$f(v_1) = \frac{1}{R} v_1 + i_s$$

$$\frac{\partial f}{\partial v} = \frac{1}{R}$$

Resistance Current

$$\left(\frac{\partial f}{\partial v}\right)^{-1} (f(v)) = \underbrace{R(f(v))}_{\text{Voltage}}$$

- Newton Algorithm

x^0 = Initial Guess, $k = 0$

Repeat {

$$x^{k+1} = x^k - \left[\frac{\partial f}{\partial x}\left(x^k\right)\right]^{-1} f\left(x^k\right)$$

$k = k + 1$

} Until ?

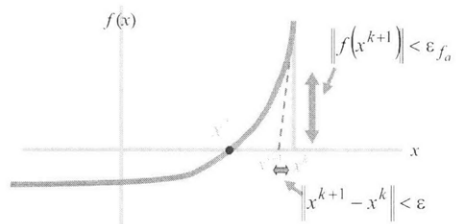
$\left\| x^{k+1} - x^k \right\| <$ threshold

$\left\| f\left(x^{k+1}\right) \right\| <$ threshold

Convergence Analysis

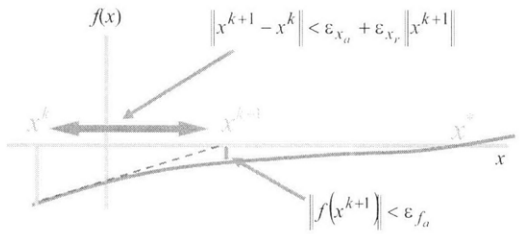
- Convergence Checks
  - $f(x)$

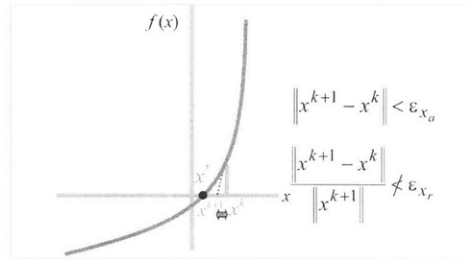
    Need an "$f(x)$" check to avoid false convergence.



  - Δx

    Need a "Δx" check to avoid false convergence.





- Convergence Example

  $f(x) = x^3 - 2 \qquad x^* = \sqrt[3]{2} \approx 1.259921$

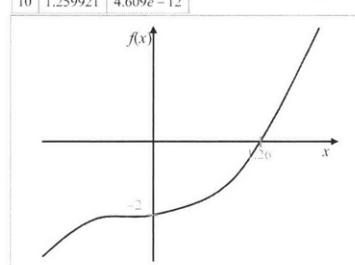
k	x^k	$\left\| x^k - x^* \right\|$
0	10.0	8.740
1	6.673333	5.413
•	•	•
8	1.261665	$1.744e-03$
9	1.259924	$2.410e-06$
10	1.259921	$4.609e-12$

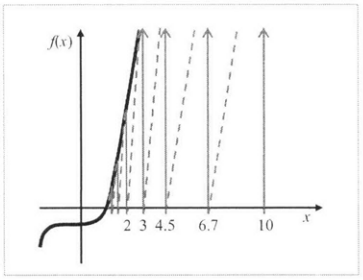
Asymptotically,

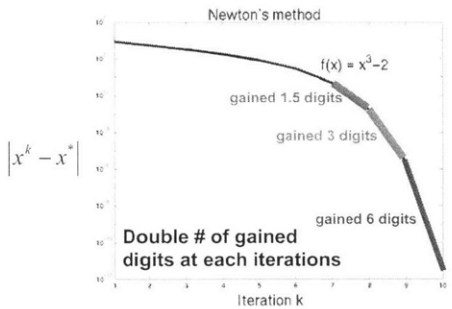
$$\left| x^{k+1} - x^* \right| \approx C \left| x^k - x^* \right|^\alpha$$

$C = 0.7951$

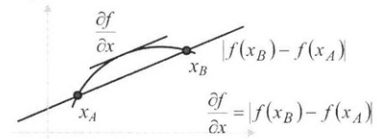
$\alpha = 2.000$ $\boxed{\text{Quadratic}}$







❖ Taylor Expansion



Taylor Series Expansion

$$f(x) = f(x^k) + \frac{\partial}{\partial x} f(x^k)(x - x^k) + \frac{1}{2!}\frac{\partial^2}{\partial x^2} f(x^k)(x - x^k)^2 + \cdots$$
$$\cdots + \frac{1}{n!}\frac{\partial^n}{\partial x^n} f(x^k)(x - x^k)^n + O\left((x - x^k)^{n+1}\right)$$
$$f(x) = f(x^k) + \frac{\partial}{\partial x} f(x^k)(x - x^k) + \frac{1}{2!}\frac{\partial^2}{\partial x^2} f(x^k)(x - x^k)^2 + \cdots$$
$$\cdots + \frac{1}{n!}\frac{\partial^n}{\partial x^n} f(x^k)(x - x^k)^n + \frac{1}{(n+1)!}\frac{\partial^{n+1}}{\partial x^{n+1}} f(\tilde{x})(x - x^k)^{n+1}$$
$$\tilde{x} \in \lfloor x, x^k \rfloor$$

Exact Taylor expansion about the iteration x^k evaluated at the solution x^*

$$0 = f(x^*) = f(x^k) + \frac{df}{dx}(x^k)(x^* - x^k) + \frac{1}{2}\frac{d^2 f}{dx^2}(\tilde{x})(x^* - x^k)^2 \qquad (1)$$

$$\text{some} \quad \underbrace{\tilde{x} \in \lfloor x^k, x^* \rfloor}_{\substack{\text{Mean Value Theorem} \\ \text{truncates Taylor Series}}}$$

Approximate Taylor expansion about the solution x^k evaluated at the next iteration x^{k+1}

$$0 = f(x^k) + \frac{df}{dx}(x^k)(x^{k+1} - x^k) \qquad (2)$$

Subtracting (1) from (2)

$$\frac{df}{dx}(x^k)(x^{k+1} - x^*) = \frac{d^2 f}{dx^2}(\tilde{x})(x^* - x^k)^2$$

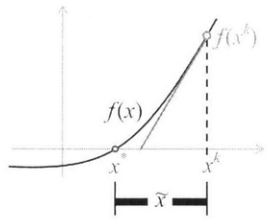
Dividing through

$$(x^{k+1} - x^*) = \left[\frac{df}{dx}(x^k)\right]^{-1} \frac{d^2 f}{dx^2}(\tilde{x})(x^* - x^k)^2$$

Suppose $\left|\left[\frac{df}{dx}(x)\right]^{-1}\frac{d^2 f}{dx^2}(x)\right| \le L$ for all x

then $x^{k+1} - x^* \le L\left|x^k - x^*\right|^2$

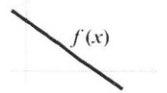
Convergence is quadratic if L is bounded.



$$f(x) = f\left(x^k\right) + \frac{\partial}{\partial x} f\left(x^k\right)\left(x^* - x^k\right) + \frac{1}{2}\frac{\partial^2}{\partial x^2} f(\tilde{x})\left(x^* - x^k\right)^2$$

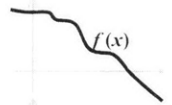
$$\tilde{x} \in \left|x^k, x^*\right|$$

$\frac{\partial^2 f}{\partial x^2}(\tilde{x}) = 0 \Rightarrow L = 0$



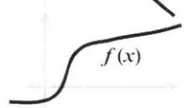
Very Good

$\frac{\partial^2 f}{\partial x^2}(\tilde{x})$ small



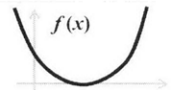
Good

$\frac{\partial^2 f}{\partial x^2}(\tilde{x})$ large



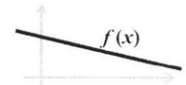
Bad

$\frac{\partial f}{\partial x}(\tilde{x}) = 0$



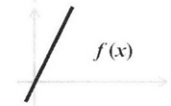
Very Bad

$\frac{\partial f}{\partial x}(\tilde{x})$ small



Bad

$\frac{\partial f}{\partial x}(\tilde{x})$ large



good

❖ Example 1

$f(x) = x^2 - 1 = 0$, find x $\left(x^* = 1\right)$

$\frac{df}{dx}\left(x^k\right) = 2x^k$

$2x^k\left(x^{k+1} - x^k\right) = -\left[\left(x^k\right)^2 - 1\right]$

$2x^k\left(x^{k+1} - x^*\right) + \frac{1}{2}2\left(x^* - x^k\right) = -\left[\left(x^k\right)^2 - 1\right]$ or $\left(x^{k+1} - x^*\right) = \frac{1}{2x^k}\left(x^k - x^*\right)^2$

convergence is quadratic

$x^0 = 1.1$ $\qquad$ $x^* = 1$

$\left(x^1 - x^*\right) = \frac{1}{2.2}\underbrace{\left(x^0 - x^*\right)}_{0.1}{}^2 = \frac{1}{220} = 0.0045$

❖ Example 2

$f(x) = x^2 = 0,\ x^* = 0$

$\frac{df}{dx}\left(x^k\right) = 2x^k$

$\Rightarrow 2x^k\left(x^{k+1} - 0\right) = \left(x^k - 0\right)^2$

$x^{k+1} - 0 = \frac{1}{2}\left(x^k - 0\right)$ for $x^k \ne x^* = 0$

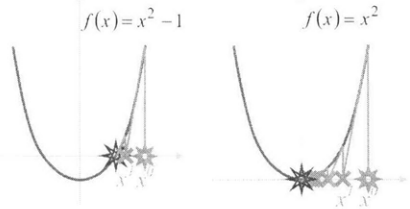
or $\left(x_{k+1} - x^*\right) = \frac{1}{2}\left(x_k - x^*\right)$

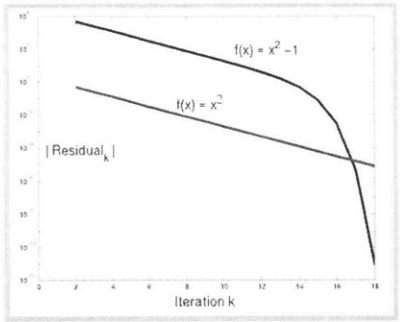
$$\boxed{\frac{df}{dx}\left(x^k\right)\left(x^{k+1} - x^*\right) = \frac{d^2 f}{dx^2}(\tilde{x})\left(x^k - x^*\right)^2}$$

Note: $\left(\frac{df}{dx}\right)^{-1}$ not bounded away from zero

convergence is linear

- Graph – convergence of examples 1 & 2





- Theorem

Suppose $\left|\left[\frac{df}{dx}(x)\right]^{-1}\frac{d^2 f}{dx^2}(x)\right| \le L$ for all x

if $L\left|x_0 - x^*\right| \le \gamma < 1$ then x_k converges to x^*

Proof. $\left|x_1 - x^*\right| \le L\left(x_0 - x^*\right)\left|x_0 - x^*\right|$

$$\Rightarrow \left|x_1 - x^*\right| \le \gamma\left|x_0 - x^*\right|$$

$$\Rightarrow \left|x_2 - x^*\right| \le L\gamma\left|x_0 - x^*\right|\left|x_1 - x^*\right|$$

$$or\ \left|x_2 - x^*\right| \le \gamma^2\left|x_1 - x^*\right| \le \gamma^3\left|x_0 - x^*\right|$$

$$\Rightarrow \left|x_3 - x^*\right| \le \gamma^4\left|x_2 - x^*\right| \le \gamma^7\left|x_0 - x^*\right|$$

$$\left|x^{k+1} - x^*\right| = \underbrace{\left[\frac{\partial f}{\partial x}\left(x^k\right)\right]^{-1}\frac{\partial^2 f}{\partial x^2}(\bar{x})}_{\le L}\left|x^k - x^*\right|^2$$

$$k = 0: \quad \left|x^1 - x^*\right| \le \underbrace{L\left|x^0 - x^*\right|}_{\le \gamma}\left|x^0 - x^*\right|$$

$$k = 1: \quad \left|x^2 - x^*\right| \le L\left|x^1 - x^*\right|\left|x^1 - x^*\right|$$

$$\le L\gamma\left|x^0 - x^*\right|\gamma\left|x^0 - x^*\right|$$

$$\le \underbrace{\gamma L\left|x^0 - x^*\right|}_{\gamma}\gamma\left|x^0 - x^*\right|$$

$$\le \gamma^3\left|x^0 - x^*\right|$$

$$\vdots$$

$$k: \quad \left|x^{k+1} - x^*\right| \le \gamma^{2(k+1)-1}\left|x^0 - x^*\right|$$

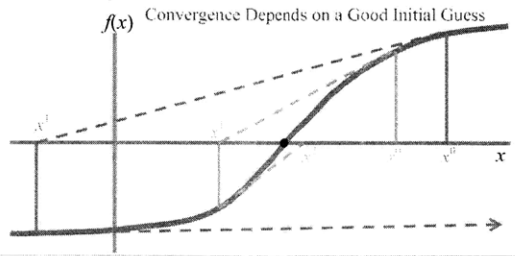
$$\gamma = \left[\frac{\partial f}{\partial x}(x)\right]^{-1}\left[\frac{\partial^2 f}{\partial x^2}(x)\right]\left|x^0 - x^*\right| < 1$$

- Theorem

If L is bounded $\left(\frac{df}{dx}\text{ bounded away from zero}; \frac{d^2 f}{dx^2}\text{ bounded}\right)$ then Newton's method is guaranteed to converge given a "close enough"guess

Always converges?

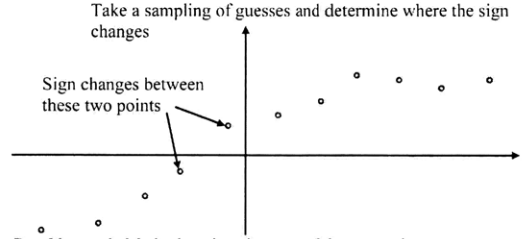
❖ 1D Pictoral Example



Aside on Global Convergence

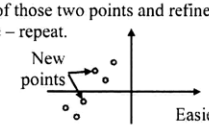
What if your initial guess does not converge?

Take a sampling of guesses and determine where the sign changes



Run Newton's Method again using one of the two points surrounding the sign change as your initial guess.

If it still does not converge? Sample points within the range of those two points and refine the area of the sign change – repeat.



1D	10 points
2D	100 points
3D	1000 points

Easier to do in 1D – becomes more taxing in larger dimensions.

Example: Heat Conducting Bar – Insulating

h_0 h_n

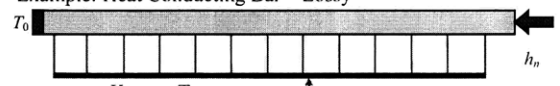
Pure Neumann Problem:
Boundary conditions – heat flow only
leads to a singular matrix

If $h_0 = h_n$ there are multiple solutions,
if $h_0 \neq h_n$ then there is NO solution

T_0 h_n

Partially Dirichlet Problem:
Boundary conditions – fixed temperature on one end & heat flow on the other

Example: Heat Conducting Bar – Lossy



T_0 h_n

Known Temperature

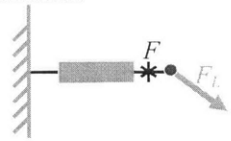
Krylov Subspace Methods converge more rapidly (small ratio of eigenvalues)

$$\begin{bmatrix} & & \\ & \mathbf{M} & \\ & & \end{bmatrix}\begin{bmatrix} T_0 \\ \vdots \\ T_n \end{bmatrix} = \begin{bmatrix} h_0 \\ \vdots \\ h_n \end{bmatrix} \qquad \begin{bmatrix} T_0 \\ \vdots \\ T_n \end{bmatrix} = \mathbf{M}^{-1}\begin{bmatrix} h_0 \\ \vdots \\ h_n \end{bmatrix}$$

λ = eigenvalue of $\mathbf{M}$ $\Longrightarrow$ $\frac{1}{\lambda}$ is an eigenvalue of $\mathbf{M}^{-1}$

Multidimensional Newton's Method

- Examples
  - Strut and Joint



$$l = \sqrt{x^2 + y^2}$$

$$F = \varepsilon A_c \frac{(l - l_o)}{l_o} = E(l - l_o)$$

$$f_x = \frac{x}{l} F = \frac{x}{l} E(l - l_o)$$

$$f_y = \frac{y}{l} F = \frac{y}{l} E(l - l_o)$$

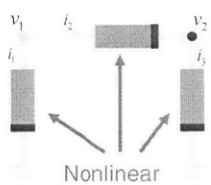
$$F(\vec{x}) = \begin{matrix} f_x + F_{L_x} = 0 \\ f_y + F_{L_y} = 0 \end{matrix}$$

OR

$$\frac{x}{l} E(l_0 - l) + F_{L_x} = 0$$

$$\frac{y}{l} E(l_0 - l) + F_{L_y} = 0$$

  - Nonlinear Resistors



Nonlinear Resistors

$i = g(v)$

Nodal Analysis

At Node 1: $i_1 + i_2 = 0$

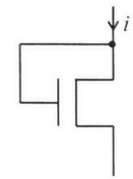
$$\Rightarrow g(v_1) + g(v_1 - v_2) = 0$$

At Node 2: $i_3 - i_2 = 0$

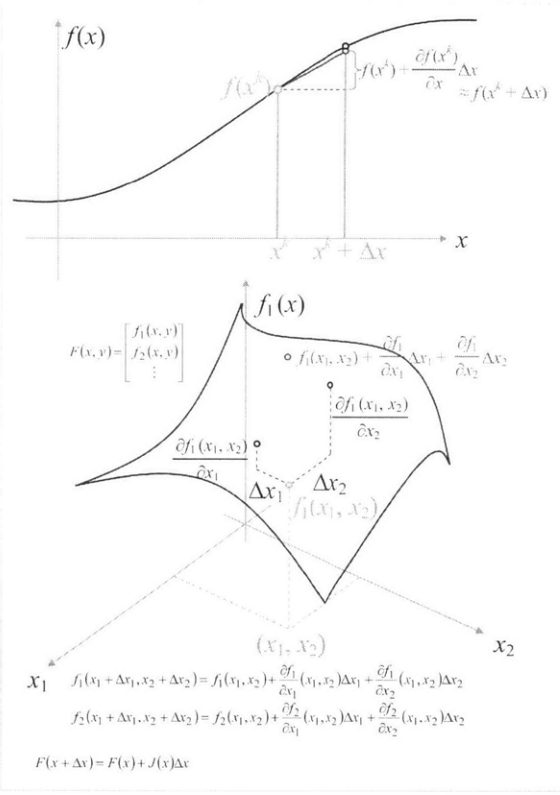
$$\Rightarrow g(v_3) - g(v_1 - v_2) = 0$$

Two coupled nonlinear equations in two unknowns

Example. Nonlinear circuit element: MOSFET



$i = kv^2$



- Jacobian Matrix

$$J_F(x)\Delta x \approx F(x+\Delta x)-F(x)$$

$$J_F(x)\Delta x = \begin{bmatrix} \frac{\partial F_1(x)}{\partial x_1} & \cdots & \frac{\partial F_1(x)}{\partial x_N} \\ \vdots & \ddots & \vdots \\ \frac{\partial F_N(x)}{\partial x_1} & \cdots & \frac{\partial F_N(x)}{\partial x_N} \end{bmatrix} \begin{bmatrix} \Delta x_1 \\ \vdots \\ \Delta x_N \end{bmatrix}$$

- General Setting

Problem : Find x^* such that $F(x^*)=0$, $x^* \in \Re^N$ and $F:\Re^N \to \Re^N$

Approximate $F(x)$ with its Taylor Series about x^k:

$$\hat{F}(x) \cong F(x^k) + \underbrace{J_F(x^k)}_{\text{Jacobian Matrix}}(x^{k+1}-x^k)$$

Find the solution of the approximation: x^{k+1}

$$F(x^k)+J_F(x^k)(x-x^k)=0 \Rightarrow x^{k+1}=x^k-\left[J_F(x^k)\right]^{-1}F(x^k)$$

- Newton Algorithm

x^0 = initial guess, $k=0$

Repeat {

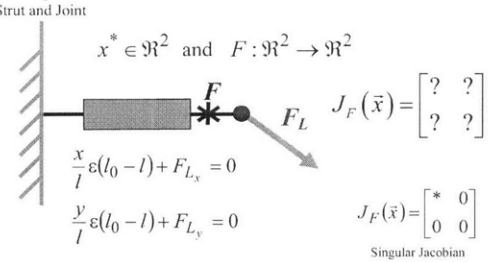
Compute $F(x^k)$, $J_F(x^k)$

Solve $J_F(x^k)(x^{k+1}-x^k)=-F(x^k)$ for x^{k+1}

$k=k+1$

} Until $\|x^{k+1}-x^k\|$, $\|f(x^{k+1})\|$ small enough

- Nodal Analysis
  - Strut and Joint



$$J_F(\vec{x}) = \begin{bmatrix} * & 0 \\ 0 & 0 \end{bmatrix}$$

$$\frac{\partial}{\partial y}\left[\frac{y}{l}\varepsilon(l_0 - l) + F_{L_y}\right]_{\substack{l_0=\sqrt{x_0^2+y_0^2} \\ y=0}}$$

$$= \left[\underbrace{\cancel{\frac{1}{l}\varepsilon(l_0 - l)}}_{0} + \underbrace{\cancel{\frac{y}{l}\varepsilon\left(\frac{\partial y}{\partial y}\right)}}_{0} + \underbrace{\cancel{y\varepsilon(l_0 - l)}}_{0}\left\{\frac{\partial}{\partial y}\left(\frac{1}{l}\right)\right\}\right]_{\substack{l_0=\sqrt{x_0^2+y_0^2} \\ y=0}}$$

scalar functions

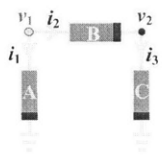
$$F_1\begin{pmatrix} x+\Delta x \\ y+\Delta y \end{pmatrix} = F_1\begin{pmatrix} x \\ y \end{pmatrix} + \frac{\partial F_1}{\partial x}\Delta x + \frac{\partial F_1}{\partial y}\Delta y$$

$$F_2\begin{pmatrix} x+\Delta x \\ y+\Delta y \end{pmatrix} = F_2\begin{pmatrix} x \\ y \end{pmatrix} + \frac{\partial F_2}{\partial x}\Delta x + \frac{\partial F_2}{\partial y}\Delta y$$

$$\Rightarrow \vec{F}(\vec{x}+\Delta\vec{x}) \approx \vec{F}(\vec{x}) + \begin{bmatrix} \frac{\partial F_1}{\partial x} & \frac{\partial F_1}{\partial y} \\ \frac{\partial F_2}{\partial x} & \frac{\partial F_2}{\partial y} \end{bmatrix}\begin{bmatrix} \Delta x \\ \Delta y \end{bmatrix}$$

- Nonlinear Resistor

$$x^* \in \Re^2 \quad \text{and} \quad F : \Re^2 \to \Re^2$$



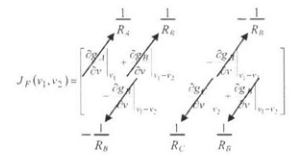
At Node 1: $i_1 + i_2 = 0$

$$\Rightarrow F_1(\vec{v}) = g_A(v_1) + g_B(v_1 - v_2) = 0$$

At Node 2: $i_3 - i_2 = 0$

$$\Rightarrow F_2(\vec{v}) = g_C(v_3) - g_B(v_1 - v_2) = 0$$

$$J_F(\vec{x}) = \begin{bmatrix} ? & ? \\ ? & ? \end{bmatrix}$$



❖ Computing the Jacobian and the Function

Consider the contribution of one nonlinear resistor Connected between nodes n_1 and n_2

$$i^b = g(v^b)$$

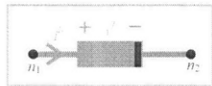
Summing currents at Node n_1 : $F_{n_1}(v) = g(v_{n_1} - v_{n_2}) + \ldots$

Summing currents at Node n_2 : $F_{n_2}(v) = -g(v_{n_1} - v_{n_2}) + \ldots$

Differentiating at Node n_1 :

$$\frac{\partial F_{n_1}(v)}{\partial v_{n_1}} = \underbrace{\frac{\partial g(v_{n_1} - v_{n_2})}{\partial v_{n_1}}}_{\frac{\partial g}{\partial v}} + \ldots \qquad \frac{\partial F_{n_1}(v)}{\partial v_{n_2}} = \underbrace{\frac{\partial g(v_{n_1} - v_{n_2})}{\partial v_{n_2}}}_{-\frac{\partial g}{\partial v}} + \ldots$$

Stamping a Resistor



$$
\underbrace{\begin{array}{c|ccccc} & & n_1 & & n_2 & \\ \hline n_1 & \cdots & \frac{\partial g(v_{n_1}-v_{n_2})}{\partial v} & \cdots & -\frac{\partial g(v_{n_1}-v_{n_2})}{\partial v} & \cdots \\ n_2 & \cdots & -\frac{\partial g(v_{n_1}-v_{n_2})}{\partial v} & \cdots & \frac{\partial g(v_{n_1}-v_{n_2})}{\partial v} & \cdots \end{array}}_{J_F(v)}
\quad
\underbrace{\begin{array}{c} \\ n_1 \\ n_2 \end{array}\begin{bmatrix} \vdots \\ g(v_{n_1}-v_{n_2}) \\ \vdots \\ -g(v_{n_1}-v_{n_2}) \\ \vdots \end{bmatrix}}_{F(v)}
$$

- More Complete Newton Algorithm

  x^0 = initial guess, $k = 0$

  Repeat {

  Zero J_F and F

  for each element

  Compute element currents and derivatives

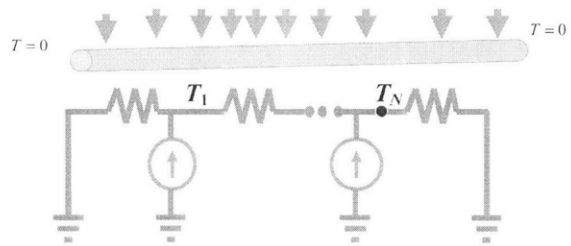
  Sum currents to F, sum derivatives to J_F

  Solve $J_F(x^k)(x^{k+1} - x^k) = -F(x^k)$ for x^{k+1}

  $k = k + 1$

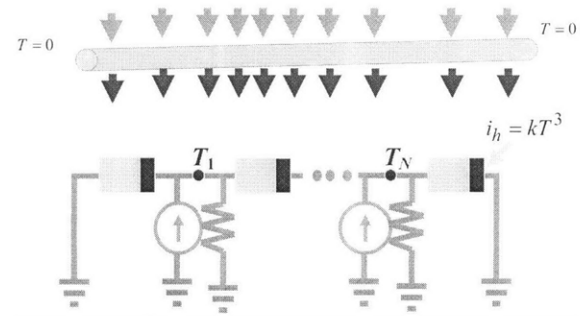
  } Until $\|x^{k+1} - x^k\|$, $|f(x^{k+1})|$ small enough

- Example: Heat Flow in Immersed Bar



What is the Jacobian?

$$
\begin{bmatrix} \frac{2}{R} & -\frac{1}{R} & & \\ -\frac{1}{R} & \frac{2}{R} & -\frac{1}{R} & \\ & -\frac{1}{R} & \frac{2}{R} & \ddots \\ & & \ddots & \ddots \end{bmatrix}
\begin{bmatrix} T_1 \\ T_2 \\ T_3 \\ \vdots \end{bmatrix}
$$



$$i_h = k(T_i - T_{i+1})^3$$

What is the Jacobian?

$$
\begin{bmatrix} \frac{1}{R} + 3k(0-T_1)^2 + 3k(T_1-T_2)^2 & -3k(T_1-T_2)^2 & \cdots \\ -3k(T_1-T_2)^2 & \frac{1}{R} + 3k(T_1-T_2)^2 + 3k(T_2-T_3)^2 & \cdots \\ \vdots & \vdots & \ddots \end{bmatrix}
$$

3.12 NonLinear Systems - Multidimensional Newton

INTRODUCTION TO NUMERICAL SIMULATION

LECTURE 12.
Multi-Dimensional Newton Method

TODAY'S OUTLINE:

- Multi-Dimensional Newton Method
  - Convergence Analysis
- Damped Newton Schemes
  - Global Convergence
  - Difficulty with Singular Jacobians
- Continuation Schemes (Homotopy Methods)
  - Source/Load Stepping
  - Improving Continuation Efficiency

MULTI-DIMENSIONAL NEWTON METHOD

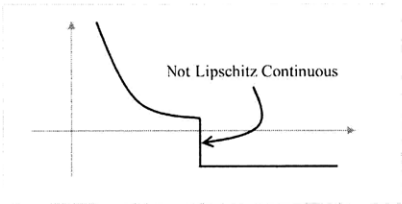
Convergence Analysis

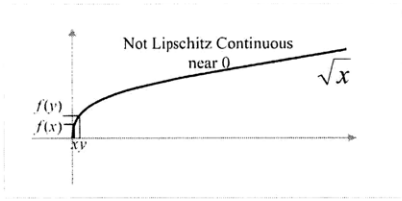
- Theorem Statement

  If *a.)* $\left\|J_F^{-1}\left(x^k\right)\right\| \le \beta$ (Inverse is bounded)

  b.) $\|J_F(x) - J_F(y)\| \le \ell\|x - y\|$ (Derivative is Lipschitz Cont)

  Then Newton's method converges given a sufficiently close initial guess.





- Key Lemma

  If $\|J_F(x) - J_F(y)\| \le \ell\|x - y\|$ (Derivative is Lipschitz Continuous)

  Then $\|F(x) - F(y) - J_F(y)(x - y)\| \le \frac{\ell}{2}\|x - y\|^2$

  There is no multidimensional mean value theorem
- Theorem Proof

  By definition of the Newton Iteration and the assumed bound on the inverse of the Jacobian

  $$\left\|x^{k+1} - x^k\right\| = \left\|J_F^{-1}\left(x^k\right)F\left(x^k\right)\right\| \le \beta\left\|F\left(x^k\right)\right\|$$

  Again applying the Newton iteration definition

  $$\left\|x^{k+1} - x^k\right\| \le \beta\left\|F\left(x^k\right) - \underbrace{F\left(x^{k-1}\right) - J_F\left(x^{k-1}\right)\left(x^k - x^{k-1}\right)}_{0}\right\|$$

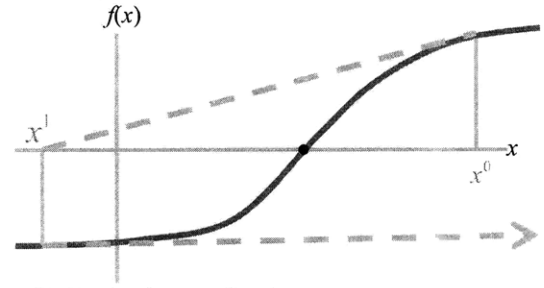
  Finally using the Lemma

  $$\left\|x^{k+1} - x^k\right\| \le \frac{\beta\ell}{2}\left\|x^k - x^{k-1}\right\|^2$$
- Implications

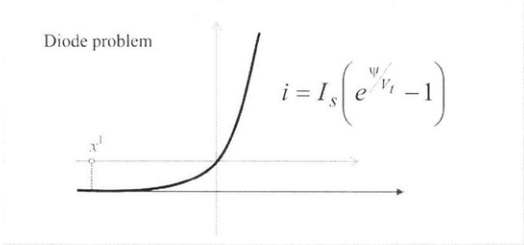
  If the function is not very steep in some direction, or not very smooth…

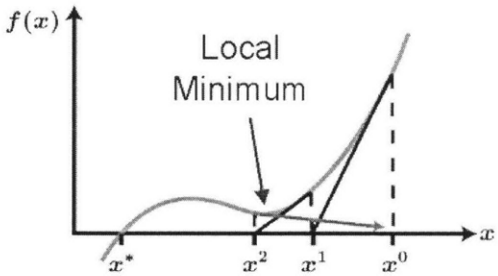
  Then Newton's method can be used to find the zero of the function but only if you all ready know the answer…

  Need a way to develop Newton methods which converge regardless of initial guess!
- Non-converging Case: 1-D Picture



Limiting the changes in X might improve convergence





At a local minimum, $\frac{\partial f}{\partial x} = 0$

Multidimensional Case: $J_F(x)$ is singular

DAMPED NEWTON SCHEMES

Global Convergence

- Newton Algorithm for Solving $F(x) = 0$

  $\vec{x}^0$ = initial guess, $k = 0$

  Repeat {

  Compute $F(\vec{x}^k)$, $J_F(\vec{x}^k)$

  Solve $J_F(\vec{x}^k)\Delta\vec{x}^{k+1} = -F(\vec{x}^k)$ for $\Delta\vec{x}^{k+1}$

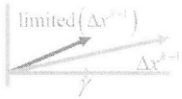
  $\vec{x}^{k=k+1} = \vec{x}^k + \text{limited}(\Delta\vec{x}^{k+1})$

  $k = k + 1$

  } Until $\|\Delta\vec{x}^{k+1}\|$, $\|F(\vec{x}^{k+1})\|$ small enough

- Limiting Methods
  - Direction Corrupting

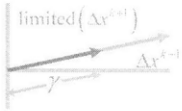
    $$\text{limited}(\Delta\vec{x}^{k+1})_i = \begin{cases} \Delta x_i^{k+1} & \text{if } |\Delta x_i^{k+1}| < \gamma \\ \gamma\,\text{sign}(\Delta x_i^{k+1}) & \text{otherwise} \end{cases}$$



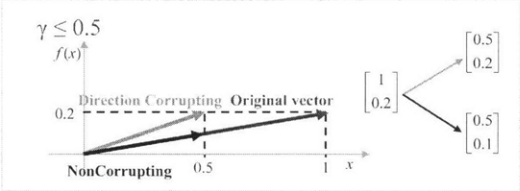
  - NonCorrupting

    $$\text{limited}(\Delta\vec{x}^{k+1}) = \alpha\Delta x^{k+1}$$

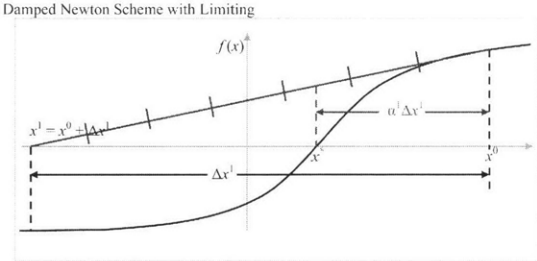
    $$\alpha = \min\left\{1, \frac{\gamma}{\|\Delta x^{k+1}\|}\right\}$$



Heuristics, No Guarantee of Global Convergence



- Damped Newton Scheme with Limiting



General Damping Scheme

Solve $J_F(x^k)\Delta x^{k+1} = -F(x^k)$ for Δx^{k+1}

$x^{k+1} = x^k + \alpha^k\Delta x^{k+1}$

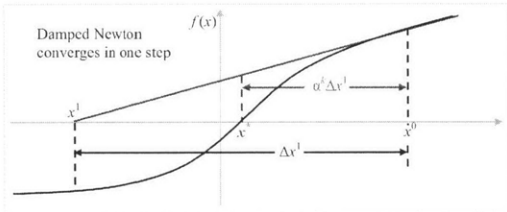
Key Idea: Line Search

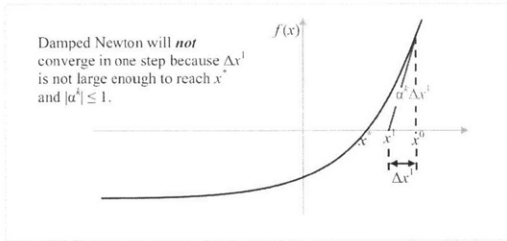
Pick α^k to minimize $\left\|F\left(x^k + \alpha^k \Delta x^{k+1}\right)\right\|_2^2 \equiv \left[F\left(x^k + \alpha^k \Delta x^{k+1}\right)\right]^T \left[F\left(x^k + \alpha^k \Delta x^{k+1}\right)\right]$

Example.

$\alpha = 1 \quad \left\|F\left(x^k + \alpha^k \Delta x^{k+1}\right)\right\|_2^2 >> \left\|F\left(x^k\right)\right\|_2^2$

$\alpha = \frac{1}{2} \quad \left\|F\left(x^k + \alpha^k \Delta x^{k+1}\right)\right\|_2^2 << \left\|F\left(x^k\right)\right\|_2^2$





- Nested Iteration

  $\bar{x}^0$ = initial guess, $k = 0$

  Repeat {

  Compute $F\left(\bar{x}^k\right), J_F\left(\bar{x}^k\right)$

  Solve $J_F\left(\bar{x}^k\right)\Delta\bar{x}^{k+1} = -F\left(\bar{x}^k\right)$ for $\Delta\bar{x}^{k+1}$

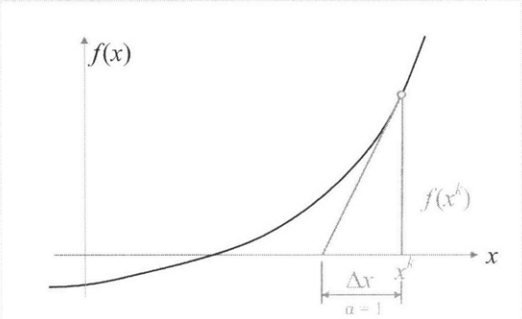
  Find $\alpha^k \in (0,1]$ such that $\left\|F\left(x^k + \alpha^k \Delta x^{k+1}\right)\right\|$ is minimized

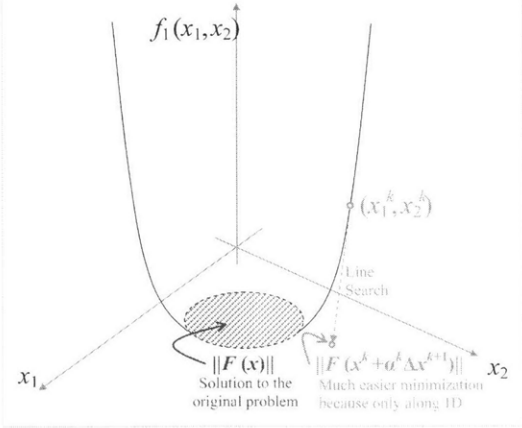
  $\bar{x}^{k+1} = \bar{x}^k + \alpha^k \Delta\bar{x}^{k+1}$

  $k = k + 1$

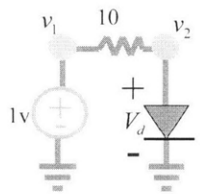
} Until $\left\|\Delta\bar{x}^{k+1}\right\|, \left\|F\left(\bar{x}^{k+1}\right)\right\|$ small enough

Method Performs a one-dimensional search in Newton Direction.





- Example

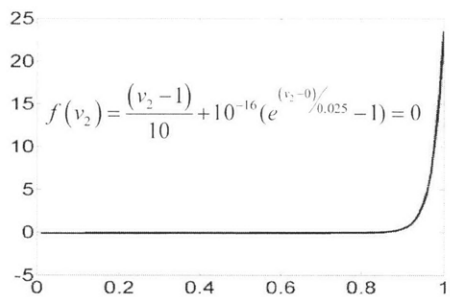


$$I_r - \frac{1}{10}V_r = 0$$

$$I_d - I_s(e^{V_d/V_t} - 1) = 0$$

Nodal Equations with Numerical Values

$$f(v_2) = \frac{(v_2 - 1)}{10} + 10^{-16}(e^{(v_2 - 0)/0.025} - 1) = 0$$

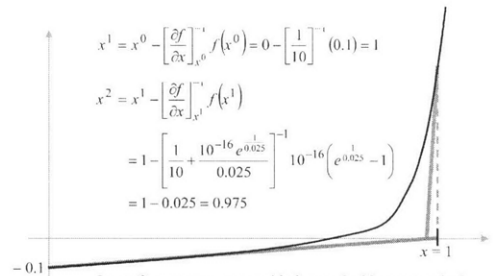


$$x^1 = x^0 - \left[\frac{\partial f}{\partial x}\right]_{x^0}^{-1} f(x^0) = 0 - \left[\frac{1}{10}\right]^{-1}(0.1) = 1$$

$$x^2 = x^1 - \left[\frac{\partial f}{\partial x}\right]_{x^1}^{-1} f(x^1)$$

$$= 1 - \left[\frac{1}{10} + \frac{10^{-16} e^{\frac{1}{0.025}}}{0.025}\right]^{-1} 10^{-16}\left(e^{\frac{1}{0.025}} - 1\right)$$

$$= 1 - 0.025 = 0.975$$



Lots of steps to converge with the regular Newton method.

- Nested Iteration

$\bar{x}^0$ = initial guess, $k = 0$

Repeat {

Compute $F(\bar{x}^k)$, $J_F(\bar{x}^k)$

Solve $J_F(\bar{x}^k)\Delta\bar{x}^{k+1} = -F(\bar{x}^k)$ for $\Delta\bar{x}^{k+1}$

Find $\alpha^k \in (0,1]$ such that $\left\|F\left(x^k + \alpha^k \Delta x^{k+1}\right)\right\|$ is minimized

$\bar{x}^{k+1} = \bar{x}^k + \alpha^k \Delta\bar{x}^{k+1}$

$k = k + 1$

} Until $\|\Delta\bar{x}^{k+1}\|$, $\left\|F\left(\bar{x}^{k+1}\right)\right\|$ small enough

How can one find the damping coefficients?

- Convergence Theorem

If a.) $\left\|J_F^{-1}(x^k)\right\| \le \beta$ (Inverse is bounded)

b.) $\|J_F(x) - J_F(y)\| \le \ell\|x - y\|$ (Derivative is Lipschitz Cont)

Then There exists a set of α^k's $\in (0,1]$ such that

$$\left\|F\left(x^{k+1}\right)\right\| = \left\|F\left(x^k + \alpha^k \Delta x^{k+1}\right)\right\| < \gamma\left\|F\left(x^k\right)\right\| \text{ with } \gamma < 1$$

Every Step reduces F—Global Convergence!

Example.

$$\left\| F\left(\overbrace{x^k + \alpha^k \Delta x^{k+1}}^{x^{k+1}}\right) \right\|_2^2 \le 0.9 \left\| F\left(x^k\right) \right\|_2^2$$

$$\left\| F\left(x^{k+1} + \alpha^{k+1} \Delta x^{k+2}\right) \right\|_2^2 \le 0.9 \left\| F\left(x^{k+1}\right) \right\|_2^2 \le 0.9^2 \left\| F\left(x^k\right) \right\|_2^2$$

$$\left\| F\left(x^{k+100} + \alpha^{k+100} \Delta x^{k+101}\right) \right\|_2^2 \le 0.9 \left\| F\left(x^{k+100}\right) \right\|_2^2 \le \underbrace{0.9^{101}}_{\to 0} \left\| F\left(x^k\right) \right\|_2^2 \to 0$$

- Proof.

  By definition of the Newton Iteration

$$x^{k+1} = x^k - \alpha^k \underbrace{J_F\left(x^k\right)^{-1} F\left(x^k\right)}_{\text{Newton Direction}}$$

  Multidimensional Mean Value Lemma

$$\left\| F(x) - F(y) - J_F(y)(x-y) \right\| \le \frac{\ell}{2} \left\| x - y \right\|^2$$

  Combining

$$\left\| F\left(x^{k+1}\right) - F\left(x^k\right) - J_F\left(x^k\right)\left[\alpha^k J_F\left(x^k\right)^{-1} F\left(x^k\right)\right] \right\| \le \frac{\ell}{2} \left\| \alpha^k J_F\left(x^k\right)^{-1} F\left(x^k\right) \right\|^2$$

  Combining terms and moving scalars out of norms

$$\left\| F\left(x^{k+1}\right) - \left(1-\alpha^k\right) F\left(x^k\right) \right\| \le \left(\alpha^k\right)^2 \frac{\ell}{2} \left\| J_F\left(x^k\right)^{-1} F\left(x^k\right) \right\|^2$$

  Using the Jacobian Bound and splitting the norm

$$\left\| F\left(x^{k+1}\right) \right\| \le \left[\left(1-\alpha^k\right) \left\| F\left(x^k\right) \right\| + \left(\alpha^k\right)^2 \frac{\beta^2 \ell}{2} \left\| F\left(x^k\right) \right\|^2 \right]$$

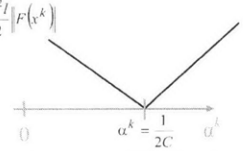
  Yields a quadratic in the damping coefficient.
  Simplifying quadratic

$$\left\| F\left(x^{k+1}\right) \right\| \le \left[1 - \alpha^k + \left(\alpha^k\right)^2 \frac{\beta^2 \ell}{2} \left\| F\left(x^k\right) \right\| \right] \left\| F\left(x^k\right) \right\|$$

$$\min_{\alpha^k} \left[1 - \alpha^k + \left(\alpha^k\right)^2 C \right] \qquad C = \frac{\beta^2 l}{2} \left\| F\left(x^k\right) \right\|$$

$$-1 + 2\alpha^k C = 0$$

$$\alpha^k = \frac{1}{2C}$$



Two Cases:

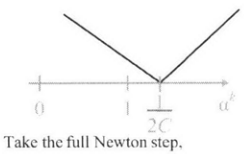
1.) $\left\| F\left(x^k\right) \right\| \le \frac{1}{\beta^2 l}$ pick $\alpha^k = 1$ (Standard Newton)

$$\left\| F\left(x^{k+1}\right) \right\| \le \left[1 - 1 + 1^2 \frac{\beta^2 \ell}{2} \left\| F\left(x^k\right) \right\| \right] \left\| F\left(x^k\right) \right\| \le \frac{1}{2} \left\| F\left(x^k\right) \right\|$$

Case 1. $\frac{1}{2C} \ge 1$

$$\frac{1}{2\left[\frac{\beta^2 l}{2} \left\| F\left(x^k\right) \right\|\right]} \ge 1$$

$$\Rightarrow \left\| F\left(x^k\right) \right\| \le \frac{1}{\beta^2 l}$$

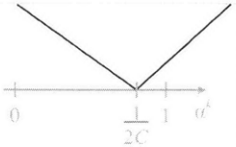


Take the full Newton step,
pick $\alpha^k = 1$ (Standard Newton)

2.) $\left\| F\left(x^{k+1}\right) \right\| > \frac{1}{\beta^2 l}$ pick $\alpha^k = \frac{1}{\beta^2 l \left\| F\left(x^k\right) \right\|}$ (Damped Newton)

$$\left\| F\left(x^{k+1}\right) \right\| \le \left[1 - \frac{1}{\beta^2 l \left\| F\left(x^k\right) \right\|} \right] \left\| F\left(x^k\right) \right\| < \left\| F\left(x^k\right) \right\|$$

Case 2. $\frac{1}{2C} < 1$



$$\frac{\beta^2 l}{2}\left\|F\left(x^k\right)\right\| > \frac{1}{2} \Rightarrow \left\|F\left(x^k\right)\right\| > \frac{1}{\beta^2 l}$$

$$\alpha^k = \frac{1}{2C} = \frac{1}{\beta^2 l\left\|F\left(x^k\right)\right\|}$$

$$\left\|F\left(x^{k+1}\right)\right\| \le \left\|1 - \frac{1}{2C} + \frac{1}{\beta^2 l} C\right\|\left\|F\left(x^k\right)\right\| = \underbrace{\left\|1 - \frac{1}{2\beta^2 l\left\|F\left(x^k\right)\right\|}\right\|}_{\gamma < 1}\left\|F\left(x^k\right)\right\|$$

$$0 < \frac{1}{2\beta^2 l\left\|F\left(x^k\right)\right\|} < \frac{1}{2}$$

$$1 > 1 - \frac{1}{2\beta^2 l\left\|F\left(x^k\right)\right\|} > \frac{1}{2}$$

$$\frac{1}{2} < \underbrace{1 - \frac{1}{2\beta^2 l\left\|F\left(x^k\right)\right\|}}_{\gamma} < 1$$

$$\gamma^k = \left(1 - \frac{1}{k+1}\right) \qquad \left\|F\left(x^0\right)\right\| = 1$$

$$\left\|F\left(x^k\right)\right\| \le \left(1 - \frac{1}{k+1}\right)\left\|F\left(x^{k-1}\right)\right\| \le \underbrace{\prod_{i=1}^{k}\left(1 - \frac{1}{i+1}\right)\cdot 1}_{\left(1-\frac{1}{2}\right)\left(1-\frac{1}{3}\right)\left(1-\frac{1}{4}\right)\left(1-\frac{1}{5}\right)\left(1-\frac{1}{6}\right)\cdots \not\to 0}$$

If β small $\xrightarrow{1D}$ $\frac{\partial f}{\partial x}$ large

ℓ small $\to \frac{1}{\beta^2 \ell}$ large $\xrightarrow{1D}$ $\frac{\partial^2 f}{\partial x^2}$ small

Always take the full Newton

If β large $\xrightarrow{1D}$ $\frac{\partial f}{\partial x}$ small

ℓ large $\to \frac{1}{\beta^2 \ell}$ small $\xrightarrow{1D}$ $\frac{\partial^2 f}{\partial x^2}$ large

$\left\|F\left(x^{k+1}\right)\right\| \le \gamma^k \left\|F\left(x^k\right)\right\|$ not good enough, need γ independent from k

The above result does imply

$\left\|F\left(x^{k+1}\right)\right\| \le \left\|F\left(x^0\right)\right\|$ not yet a convergence theorem

$$\Rightarrow 1 - \frac{1}{2\beta^2 \ell\left\|F\left(x^k\right)\right\|} \le 1 - \frac{1}{2\beta^2 \ell\left\|F\left(x^0\right)\right\|} \le \gamma^0$$

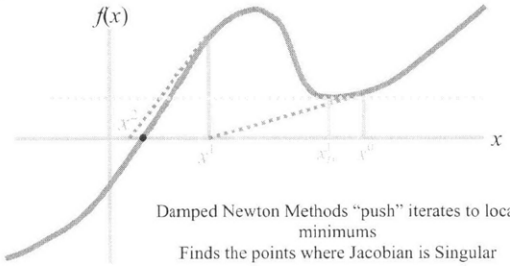
$$\left\|F\left(x^{k+1}\right)\right\| \le 0.9\left\|F\left(x^k\right)\right\|$$

$$\left\|F\left(x^{k+2}\right)\right\| \le 0.99\left\|F\left(x^{k+1}\right)\right\|$$

$$\left\|F\left(x^{k+3}\right)\right\| \le 0.999\left\|F\left(x^{k+2}\right)\right\|$$

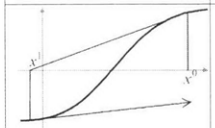
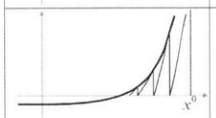
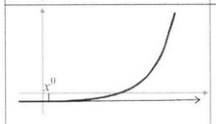
$$\lim_{k\to\infty}\left\|F\left(x^k\right)\right\| \to 0?$$

Difficulty with Singular Jacobians

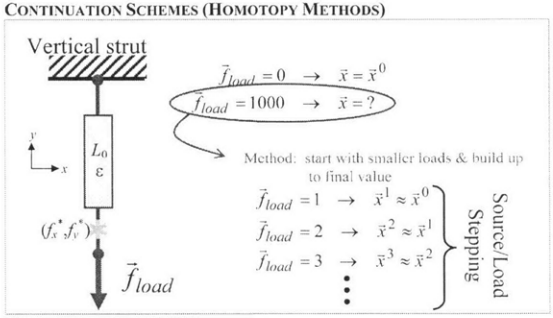


Damped Newton Methods "push" iterates to local minimums

Finds the points where Jacobian is Singular

	Regular Newton	Damped Newton	Damped Newton with half-step ($\alpha^k = ½$)
	✗	C 1 STEP expensive more convenient multidim.	C fast
	C	C SAME # STEPS more expensive	C more steps than Newton
	C SEVERAL STEPS	C 1 STEP minimization costs	C Fast Cheap

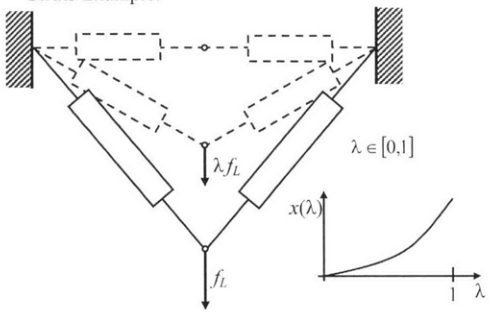
Continuation Schemes (Homotopy Methods)



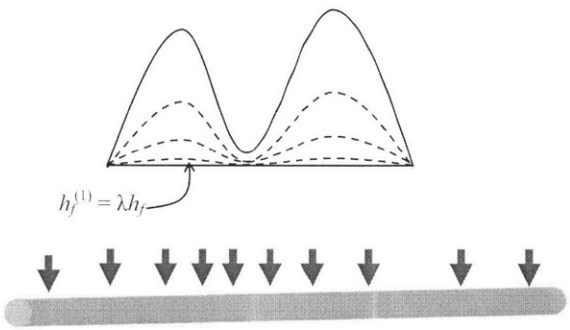
Source/Load Stepping

- Basic Concepts
  - Newton converges given a close initial guess
    - Generate a sequence of problems
    - Make sure previous problem generates guess for next problem

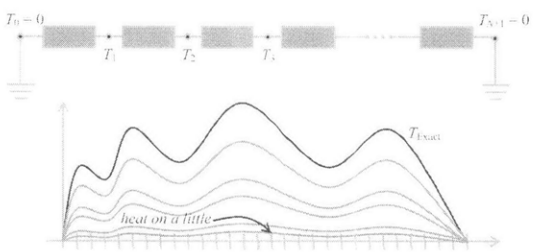
Struts Example.



- Heat-conducting bar example



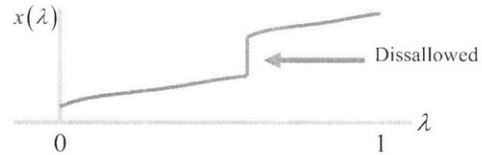
1. Start with heat off, $T = 0$ is a very close initial guess
2. Increase the heat slightly, $T = 0$ is a good initial guess.
3. Increase heat again…

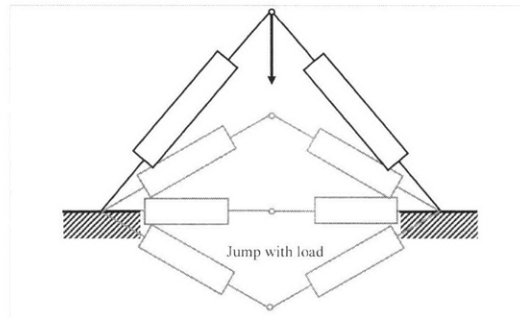


❖ General Setting

Solve $\tilde{F}(x(\lambda),\lambda) = 0$ where :

a)	$\tilde{F}(x(0),0) = 0$ is easy to solve	Starts the continuation
b)	$\tilde{F}(x(1),1) = F(x)$	Ends the continuation
c)	$x(\lambda)$ is sufficiently smooth	Hard to ensure!





❖ Template Algorithm

Solve $\tilde{F}(x(\lambda),\lambda) = 0,\ x(\lambda_{prev}) = x(0)$

$\delta\lambda = 0.01,\ \lambda = \delta\lambda$

While $\lambda < 1$ {

  $x^0(\lambda) = x(\lambda_{prev})$

  Try to solve $\tilde{F}(x(\lambda),\lambda) = 0$ with Newton

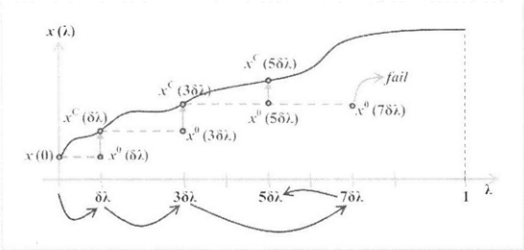
  If Newton Converged

    $x(\lambda_{prev}) = x(\lambda),\ \lambda = \lambda + \delta\lambda,\ \delta\lambda = 2\delta\lambda$

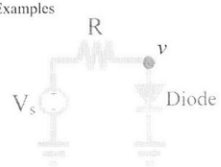
  Else

    $\delta\lambda = \frac{1}{2}\delta\lambda,\ \ \lambda = \lambda_{prev} + \delta\lambda$

}

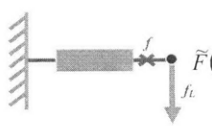


❖ Examples



$$\tilde{f}(v(\lambda),\lambda) = i_{diode}(v) + \frac{1}{R}(v - \lambda V_s) = 0$$

$$\frac{\partial \tilde{f}(v,\lambda)}{\partial v} = \frac{\partial i_{diode}(v)}{\partial v} + \frac{1}{R} \leftarrow \text{Not } \lambda \text{ dependent!}$$

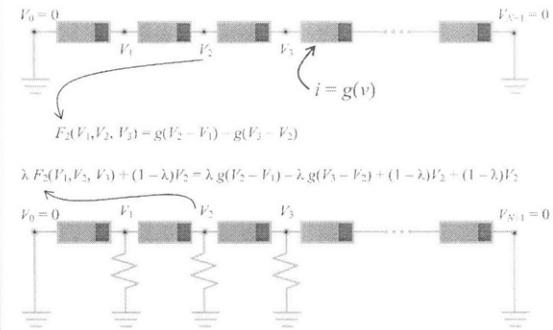


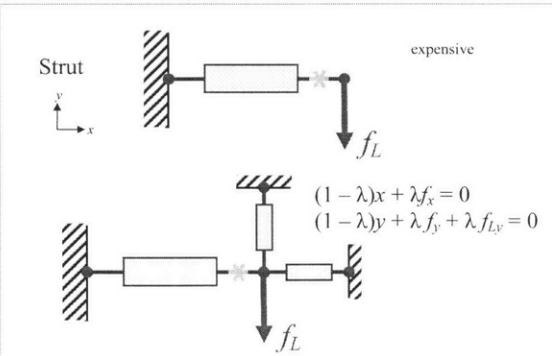
$$\tilde{F}(\vec{x},\lambda) = \begin{cases} f_x(x,y) = 0 \\ f_y(x,y) + \lambda f_l = 0 \end{cases}$$

Source/Load Stepping Does Not Alter Jacobian

Improving Continuation Efficiency

Nonlinear Circuit





- Jacobian Altering Scheme – Description

$$\tilde{F}(x(\lambda),\lambda) = \lambda F(x(\lambda)) + (1-\lambda)x(\lambda)$$

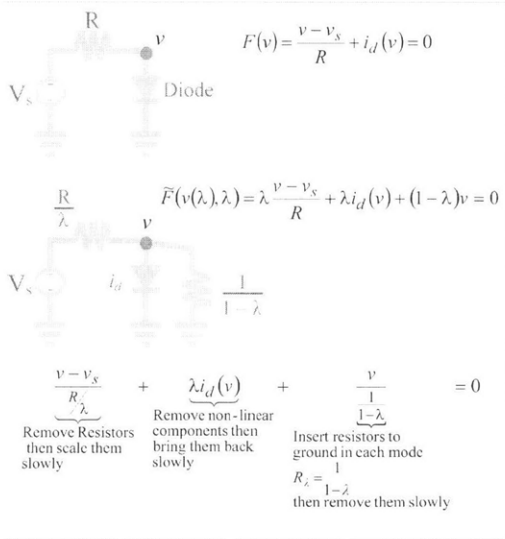
Observations

$\lambda = 0$ $\quad \tilde{F}(x(0),0) = x(0) = 0$

$\dfrac{\partial \tilde{F}(x(0),0)}{\partial x} = \mathbf{I}$ $\quad$ Problem is easy to solve and Jacobian is definitely nonsingular.

$\lambda = 1$ $\quad \tilde{F}(x(1),1) = F(x(1))$

$\dfrac{\partial \tilde{F}(x(1),1)}{\partial x} = \dfrac{\partial F(x(1))}{\partial x}$ $\quad$ Back to the original problem and original Jacobian



$$F(v) = \frac{v - v_s}{R} + i_d(v) = 0$$

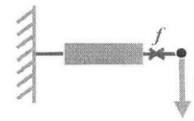
$$\tilde{F}(v(\lambda),\lambda) = \lambda \frac{v - v_s}{R} + \lambda i_d(v) + (1-\lambda)v = 0$$

$$\underbrace{\frac{v - v_s}{R/\lambda}} + \underbrace{\lambda i_d(v)} + \underbrace{\frac{v}{\frac{1}{1-\lambda}}} = 0$$

Remove Resistors then scale them slowly

Remove non-linear components then bring them back slowly

Insert resistors to ground in each mode $R_\lambda = \dfrac{1}{1-\lambda}$ then remove them slowly



$$f = \lambda\varepsilon(l - l_0)$$

$$(1-\lambda)x + \lambda f_x(x,y) = 0$$

$$(1-\lambda)y + \lambda f_y(x,y) + \lambda f_L = 0$$

- Basic Algorithm

Solve $\tilde{F}(x(\lambda),\lambda) = 0,\ x(\lambda_{prev}) = x(0)$

$\delta\lambda = 0.01,\ \lambda = \delta\lambda$

While $\lambda < 1$ {

$x^0(\lambda) = x(\lambda_{prev}) + ?$

Try to solve $\tilde{F}(x(\lambda),\lambda) = 0$ with Newton

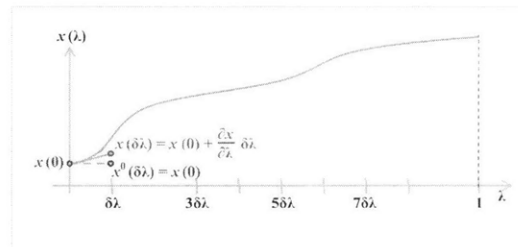
If Newton Converged

$x(\lambda_{prev}) = x(\lambda),\ \lambda = \lambda + \delta\lambda,\ \delta\lambda = 2\delta\lambda$

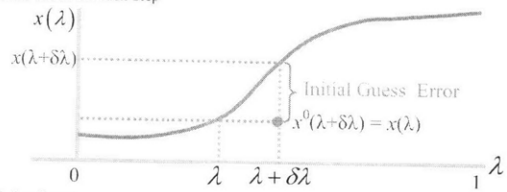
Else

$\delta\lambda = \frac{1}{2}\delta\lambda,\ \lambda = \lambda_{prev} + \delta\lambda$

}

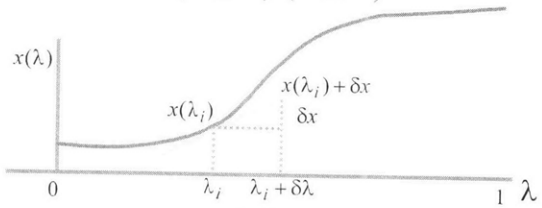


❖ Initial Guess for each Step



❖ Update Improvement

$$x^0(\lambda+\delta\lambda) = x(\lambda) - \left(\frac{\partial\widetilde{F}(x(\lambda),\lambda)}{\partial x}\right)^{-1}\left(\frac{\partial\widetilde{F}(x(\lambda),\lambda)}{\partial\lambda}\right)\delta\lambda$$



$$\widetilde{F}(x(\lambda+\delta\lambda),\lambda+\delta\lambda) \approx \underbrace{\widetilde{F}(x(\lambda),\lambda)}_{0} + \frac{\partial\widetilde{F}(x(\lambda),\lambda)}{\partial x}(x(\lambda+\delta\lambda)-x(\lambda)) + \frac{\partial\widetilde{F}(x(\lambda),\lambda)}{\partial\lambda}\delta\lambda$$

$$\Rightarrow \underbrace{\frac{\partial\widetilde{F}(x(\lambda),\lambda)}{\partial x}}_{\text{have from last step's Newton}}\left(\underbrace{x^0(\lambda+\delta\lambda)}_{\text{Better guess for next step's Newton}} - x(\lambda)\right) = -\frac{\partial\widetilde{F}(x(\lambda),\lambda)}{\partial\lambda}\delta\lambda$$

$$\widetilde{F}(x,\lambda) = \lambda F(x) + (1-\lambda)x$$

Set $\delta\widetilde{F} = 0$

$$\delta\widetilde{F} = \underbrace{\frac{\delta\widetilde{F}}{\delta x}}_{\text{Have From last step's Newton what is it ?}}\delta x + \underbrace{\frac{\delta\widetilde{F}}{\delta\lambda}}_{\text{Have From last step's Newton}}\delta\lambda$$

Solve for δx : $\left[\frac{\delta\widetilde{F}}{\delta x}\right]\delta x = -\left[F(x(\lambda_{prev})) - x(\lambda_{prev})\right]\delta\lambda$

$\Rightarrow$ Initial guess for next Newton : $x^0(\lambda) = x(\lambda_{prev}) + \delta x$

3.13 NonLinear Systems - A Case Study

INTRODUCTION TO NUMERICAL SIMULATION

LECTURE 13.
Newton-Method Case Study – Simulating an Image Smother

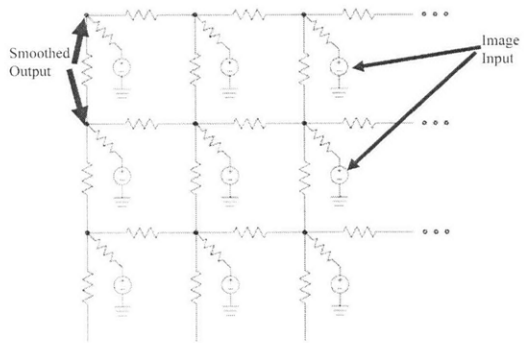
TODAY'S OUTLINE:

- Image segmentation example description
- Formulation: node-branch or nodal?
- What Solver for Newton step: sparse LU or GCR?
  - Convergence of Newton GCR
  - Matrix-free idea
- Continuation?
  - Jacobian altering scheme (for singular Jacobians)
  - Jacobian altering with update improvement
  - Arc-Length Continuation (for multiple solutions)

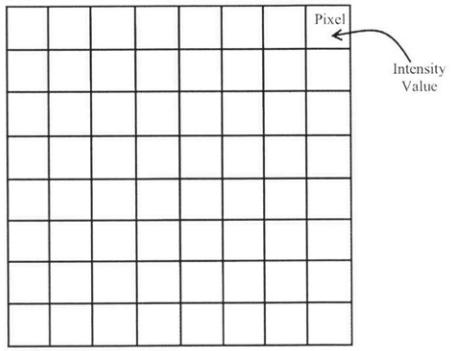
IMAGE EXAMPLE

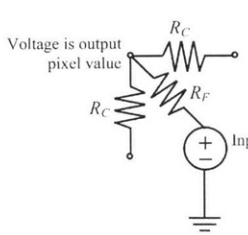
Simple Smoother

- Circuit Diagram



Array of photo diodes or photo transistors that provide some voltage proportional to the light in that particular spot of the chip



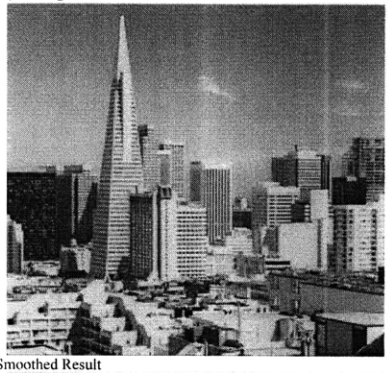


R_F : Fidelity Resistor
If R_F is small $\rightarrow V_{out_i} \approx V_{in_i}$

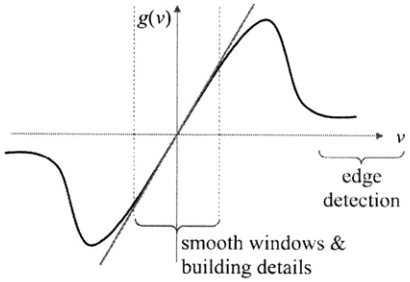
R_C : Coupling (Smoothing) Resistor
If R_C is small $\rightarrow V_{out_i} \approx V_{out_{i-1}}$

We can make images sharp or smooth by changing R_C

- Input Image

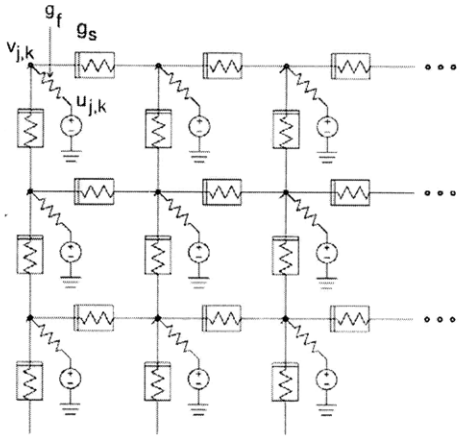


- Smoothed Result



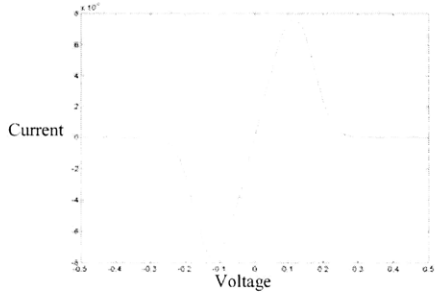
Nonlinear Smoother

- Circuit Diagram

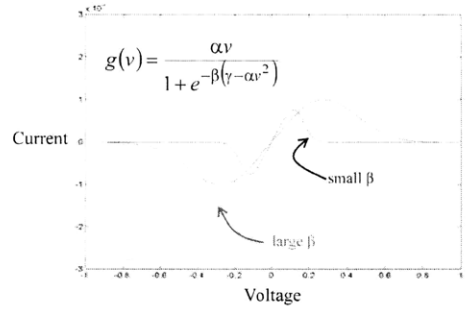


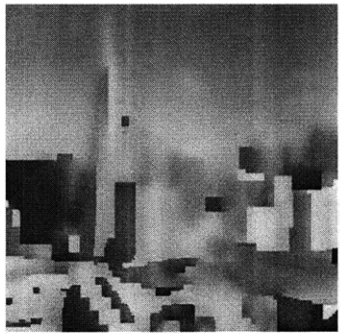
❖ Constitutive Equation

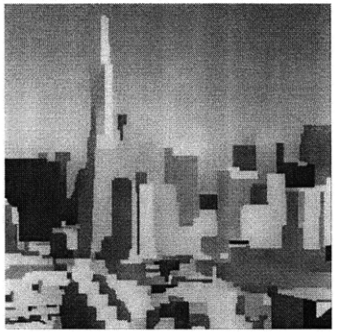
$$i(v) = \frac{\alpha v}{1 + e^{-\beta(\gamma - \alpha v^2)}}$$



○ Varying Beta

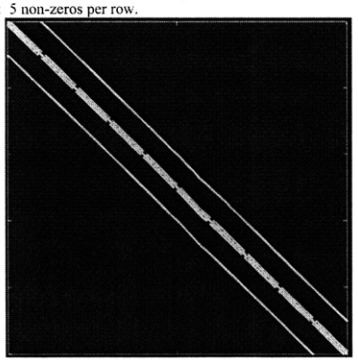


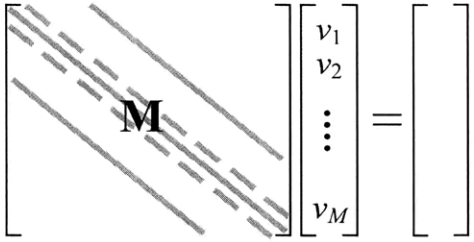




FORMULATION – NODE-BRANCH OR NODAL?

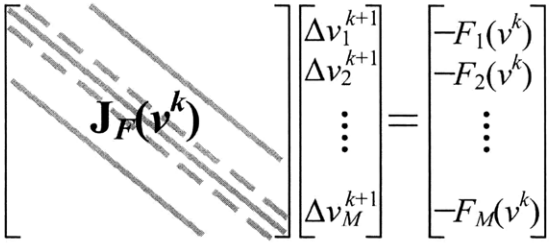
The nodal matrix is smaller and diagonally dominant in the linear case. It is also very sparse: 5 non-zeros per row.





For nonlinear resistors:
- 5 non-zero entries per row
- Diagonally dominant
- Symmetric positive definite

$$\mathbf{M}x = b$$

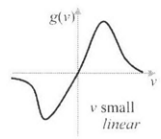


Same structure as linear – 5 nonzero entries per row
Diagonally dominant?

$$\begin{bmatrix} \mathbf{J}_F \end{bmatrix} \begin{bmatrix} \Delta x^{k+1} \end{bmatrix} = -\begin{bmatrix} \mathrm{F}(x^k) \end{bmatrix}$$

$$g(v) = \frac{\alpha v}{1 + e^{-\beta(\gamma - \alpha v^2)}}$$

$$\frac{\partial g}{\partial v} = \frac{\alpha}{1 + e^{-\beta(\gamma - \alpha v^2)}} - \frac{2\alpha^2 \beta v^2 e^{-\beta(\gamma - \alpha v^2)}}{\left[1 + e^{-\beta(\gamma - \alpha v^2)}\right]^2}$$

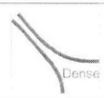


$$\begin{bmatrix} & & \\ & \boldsymbol{J_F} & \\ & & \end{bmatrix} \begin{bmatrix} v_{g_1} \\ \vdots \\ v_{g_{m^2}} \end{bmatrix}$$

Linear Case
- sparse
- strictly diagonally dominant

Nonlinear Case
- sparsity same as linear
- strict diag. dom only if Δv's are small

NEWTON SOLVER – SPARSE LU OR GCR?

	LU	GCR
Computation Time	$O(n^{1.5})$	$O(nq)$ q = # GCR steps
Memory	 $O(n)$	$O(n)$ Just store vectors $\mathbf{M}p$
How accurately should we solve the system?	16 digits – no choice	Can stop after 1 digit!
Jacobian needed?	YES	NO

Basic Algorithm

- Nested Iteration

  x^0 = Initial Guess, $k = 0$

  Repeat {

  Compute $F(x^k), J_F(x^k)$

  Solve (Using GCR)

  $J_F(x^k)\Delta x^{k+1} = -F(x^k)$ for Δx^{k+1}

  $x^{k+1} = x^k + \Delta x^{k+1}$

  $k = k + 1$

  } Until $\|\Delta x^{k+1}\|, \|F(x^{k+1})$ small enough

  – How Accurately Should We Solve with GCR?

  – And do we REALLY need to assemble the Jacobian?

Matrix-free Idea

Consider Applying GCR to the Newton Iterate Equation

$J_F(x^k)\Delta x^{k+1} = -F(x^k)$

At each iteration GCR forms a matrix-vector product

$J_F(x^k)r^l \approx \frac{1}{\varepsilon}\left[F(x^k + \varepsilon r^l) - F(x^k)\right]$

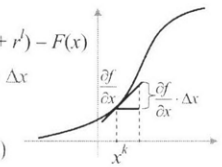
It is possible to use Newton-GCR without Jacobians!!

Need to select a good ε

l^{th} step of GCR

$r^0 \rightarrow [\mathbf{J}_F]\, r^l \neq F(x + r^l) - F(x)$

even if $r^l \leftrightarrow \Delta x$



$\frac{\partial f}{\partial x} \cdot \Delta x \cong f(x + \Delta x) - f(x)$

$J_F(x^k) \cdot \Delta x \cong F(x + \Delta x) - F(x)$

Δx small

choose $\Delta x = \varepsilon r^l$ $\quad$ ε small

$J_F(x^k)\varepsilon r^l \cong F(x^k + \varepsilon r^l) - F(x^k)$

How to choose ε?

If too large, not a good approximation

If too small,

$F(x^k + \varepsilon r^l) \approx F(x^k)$

numerical problems/ machine precision

- Basic Algorithm – Nested Iteration

  x^0 = Initial Guess, $k = 0$

  Repeat {

  Compute $F(x^k), J_F(x^k)$

  $r^0 = -F(x^k)$

  Repeat {

  Calculate $J_F(x^k)r^l \approx \frac{1}{\varepsilon}\left[F(x^k + \varepsilon r^l) - F(x^k)\right]$

  M - Orthonormalize that vector

  Update solution and residual r^l

  } Until $\|r^l\|$ small

  $x^{k+1} = x^k + \Delta x^{k+1}$

  $k = k + 1$

  } Until $\|\Delta x^{k+1}\|, \|F(x^{k+1})$ small enough

Convergence of Newton-GCR

- Basic Algorithm – Solve Accuracy Required

  After l steps of GCR

$$J_F(x^k)\underbrace{\Delta x^{k+1,l}}_{\substack{\text{Newton}\\ \text{delta from}\\ l\text{ GCRsteps}}} = -F(x^k) + \underbrace{r^{k,l}}_{\substack{\text{GCR}\\ \text{Residual}}}$$

If

a.) $\|J_F^{-1}(x^k)\| \le \beta$ (Inverse is bounded)

b.) $\|J_F(x) - J_F(y)\| \le \ell\|x - y\|$ (Derivatve is Lipschitz Cont.)

c.) $\|r^{k,l}\| \le C\|F(x^k)\|^2$ (More accurate near convergence)

Then

The Newton-Iterative Method Converges Quadratically

Count the number of iterations

$\Rightarrow F(x^k)$ $\qquad F(x^k + \varepsilon p^l)$

Number of digits – doubles with every Newton step (quadratic convergence)

$$1 \xrightarrow{1} 2 \xrightarrow{2} 4 \xrightarrow{3} 8 \xrightarrow{4} 16$$

Number of digits – linear convergence

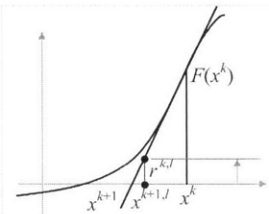
$$1 \xrightarrow{1} 2 \xrightarrow{2} 3 \xrightarrow{3} 4 \xrightarrow{4} 5$$

3 steps GCR – 1 digit $\dfrac{\|r^3\|}{\|r^0\|} = 0.1$

Accuracy: 8 digits

Quadratic $\quad \underset{\text{Newton}}{4} \times \underset{\text{GCR}}{(3\times 8)} \times \underset{\text{eval}}{2} = 192$

Linear $\quad \underset{\text{Newton}}{8} \times \underset{\text{GCR}}{3} \times \underset{\text{eval}}{2} = 48$



Just make sure $\|r^{k,l}\| \le C\|F(x^k)\|^2$

1. Need more accuracy near solution
2. But can use as initial guess of GCR the previous Newton iterate which is very close to the solution!

How accurately should we solve GCR?

First idea: so accurate that it does not interfere with the quadratic convergence of the Newton.

For sure we don't want to solve it more accurately than $\|r^{k,l}\|$ because this is an approximation, but maybe we can do even less…

THEOREM	PRACTICALLY
Use as many GCR as needed for quadratic convergence i.e. $\|r^j\| < C\|F(x^k)\|^2$ so that only need a few Newton steps	Stop GCR when residual is down by one order of magnitude (might need more Newton steps) What we really want to count is not the number of Newton steps but rather the total number of system evaluations (2 per each GCR)
Newton Quadratic 1 2 4 8 16 32 digits GCR 3 steps per digit 2 evaluations per GCR step e.g. want 8 digits: $4_{\text{Newton steps}} \times (3\times 8)_{\text{GCR steps}} \times 2_{\text{eval}}$ $= 192$ evaluations e.g. want 16 digits: $5_{\text{Newton steps}} \times (3\times 16)_{\text{GCR steps}} \times 2_{\text{eval}}$ $= 480$ evaluations	Newton Linear 1 2 3 4 5 6 digits GCR 3 steps per digit 2 evaluations per GCR step e.g. want 8 digits: $8_{\text{Newton steps}} \times 3_{\text{GCR steps}} \times 2_{\text{eval}}$ $= 48$ evaluations e.g. want 16 digits: $16_{\text{Newton steps}} \times 3_{\text{GCR steps}} \times 2_{\text{eval}}$ $= 96$ evaluations

Proof:

By definition of the Newton-Iterative Method

$$x^{k+1} = x^k - \underbrace{J_F\left(x^k\right)^{-1}\left(F\left(x^k\right) + r^{k,l}\right)}_{\text{Approximate Newton Direction}}$$

Multidimensional Mean Value Lemma

$$\left\|F(x) - F(y) - J_F(y)(x-y)\right\| \le \frac{\ell}{2}\|x-y\|^2$$

Combining

$$\left\|F\left(x^{k+1}\right) - F\left(x^k\right) - J_F\left(x^k\right)\left[J_F\left(x^k\right)^{-1}\left(F\left(x^k\right) + r^{k,l}\right)\right]\right\| \le \frac{\ell}{2}\left\|\left[J_F\left(x^k\right)^{-1}\left(F\left(x^k\right) + r^{k,l}\right)\right]\right\|^2$$

$$\left\|F\left(x^{k+1}\right) - F\left(x^k\right) - F\left(x^k\right) + r^{k,l}\right\| \le \frac{\ell}{2}\left\|\left[J_F\left(x^k\right)^{-1}\left(F\left(x^k\right) + r^{k,l}\right)\right]\right\|^2$$

Using the triangle inequality

$$\left\|F\left(x^{k+1}\right)\right\| \le \frac{\ell}{2}\left\|\left[J_F\left(x^k\right)^{-1}\left(F\left(x^k\right) + r^{k,l}\right)\right]\right\|^2 + \left\|r^{k,j}\right\|$$

Using the Jacobian Bound

$$\left\|F\left(x^{k+1}\right)\right\| \le \frac{\beta^2\ell}{2}\left\|F\left(x^k\right)\right\|^2 + \left(1 + \frac{\beta^2\ell\left\|r^{k,j}\right\|}{2}\right)\left\|r^{k,j}\right\|$$

Using the bound on the iterative solver error

$$\left\|F\left(x^{k+1}\right)\right\| \le \frac{\beta^2\ell}{2}\left\|F\left(x^k\right)\right\|^2 + \left(1 + \frac{\beta^2\ell\left\|F\left(x^k\right)\right\|^2}{2}\right)C\left\|F\left(x^k\right)\right\|^2$$

Combining Terms

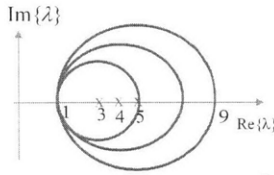
$$\left\|F\left(x^{k+1}\right)\right\| \le \underbrace{\left[\frac{\beta^2\ell}{2} + \left(1 + \frac{\beta^2\ell\left\|F\left(x^k\right)\right\|^2}{2}\right)C\right]}_{\text{Easily Bounded}}\left\|F\left(x^k\right)\right\|^2$$

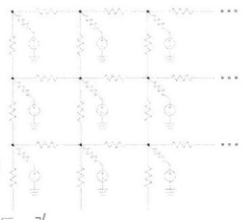
$$\Gamma^{k,l} = \underbrace{-F\left(x^k\right)}_{b} - \underbrace{J_F\left(x^k\right)}_{\mathbf{M}}\underbrace{\Delta x^{k,l}}_{x^k}$$

❖ Gershgorin & Chebyshev

$$\begin{bmatrix}
3 & -1 & & & & & -1 & & & & & \\
-1 & 4 & -1 & & & & & -1 & & & & \\
 & -1 & 4 & -1 & & & & & -1 & & & \\
 & & -1 & 4 & \ddots & & & & & -1 & & \\
 & & & \ddots & \ddots & & & & & & \ddots & \\
-1 & & & & & \ddots & -1 & & & & & -1 \\
 & -1 & & & & -1 & 5 & -1 & & & -1 & \\
 & & -1 & & & & -1 & 5 & -1 & & & -1 \\
 & & & -1 & & & & -1 & 5 & \ddots & & \\
 & & & & \ddots & & & & \ddots & \ddots & & \ddots \\
 & & & & & -1 & & & & & & \\
 & & & & & & -1 & & & & & \\
 & & & & & & & -1 & & & & \\
 & & & & & & & & -1 & & & \\
 & & & & & & & & & \ddots & &
\end{bmatrix}$$

For linear case





$1 \le \lambda_{\min} \le \lambda_{\max} \le 9$

$$\left\|\frac{r^l}{r^0}\right\| \le 2\left[\frac{\sqrt{\frac{9}{1}}-1}{\sqrt{\frac{9}{1}}+1}\right]^l = 2\left(\frac{2}{4}\right)^l = \left(\frac{1}{2}\right)^{l-1}$$

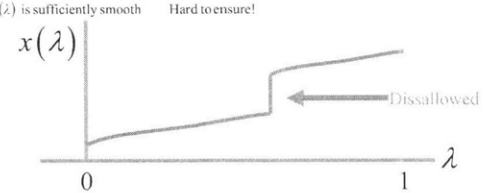
about one digit every 3 iterations:
about 0.1% error after 3 iterations

For nonlinear case:
Could be bad.
Could be singular.
Need continuation scheme that fixes it.

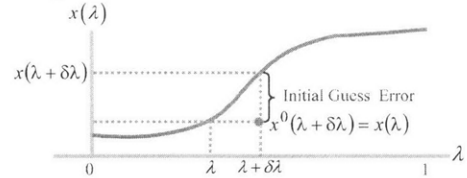
DAMPING?
CONTINUATION?
Basic Concepts

Solve $\tilde{F}(x(\lambda),\lambda) = 0$ where :

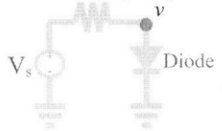
a.)	$\tilde{F}(x(0),0) = 0$ is easy to solve	Starts the continuation
b.)	$\tilde{F}(x(1),1) = F(x)$	Ends the continuation
c.)	$x(\lambda)$ is sufficiently smooth	Hard to ensure!



- Initial Guess for each Step
  Convergence of Newton-GCR
  Basic Algorithm – Nested Iteration

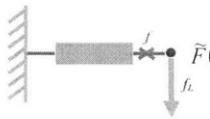


- Template Algorithm
  Solve $\tilde{F}(x(0),0)$, $x(\lambda_{prev}) = x(0)$
  $\delta\lambda = 0.01$, $\lambda = \delta\lambda$
  While $\lambda < 1$ {
  $x^0(\lambda) = x(\lambda_{prev})$
  Try to Solve $\tilde{F}(x(\lambda),\lambda) = 0$ with Newton
  If Newton Converged
  $x(\lambda_{prev}) = x(\lambda)$, $\lambda = \lambda + \delta\lambda$, $\delta\lambda = 2\delta\lambda$
  Else
  $\delta\lambda = \frac{1}{2}\delta\lambda$, $\lambda = \lambda_{prev} + \delta\lambda$
  }
- Practical Answer
  Look for another continuation scheme
  $\tilde{F}(x(\lambda),\lambda) = \lambda F(x(\lambda)) + (1-\lambda)x(\lambda)$
  Observations:
  $\lambda = 0$ $\tilde{F}(x(0),0) = x(0) = 0$
  $\frac{\partial \tilde{F}(x(0),0)}{\partial x} = \mathbf{I}$ — Problem is easy to solve and Jacobian definitely nonsingular
  $\lambda = 1$ $\tilde{F}(x(1),1) = F(x(1))$
  $\frac{\partial \tilde{F}(x(1),1)}{\partial x} = \frac{\partial F(x(1))}{\partial x}$ — Back to the original problem and original Jacobian
- Source/Load Stepping Examples



$$\tilde{f}(v(\lambda),\lambda) = i_{diode}(v) + \frac{1}{R}(v - \lambda V_s) = 0$$

$$\frac{\partial \tilde{f}(v,\lambda)}{\partial v} = \frac{\partial i_{diode}(v)}{\partial v} + \frac{1}{R} \quad \leftarrow \text{Not } \lambda \text{ dependent!}$$

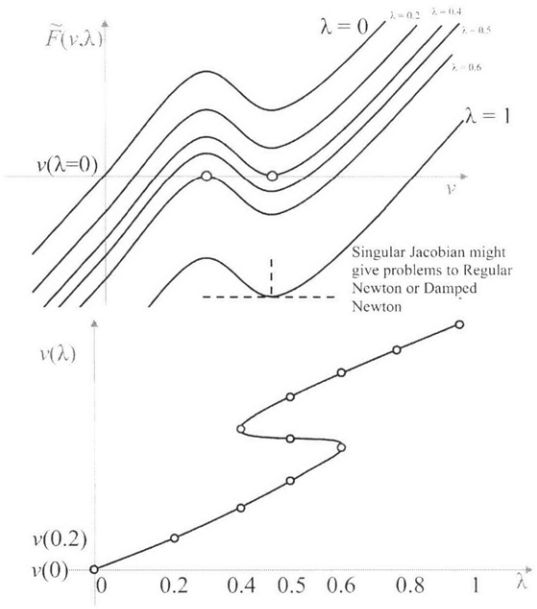


$$\widetilde{F}(\vec{x},\lambda)=\begin{cases} f_x(x,y)=0 \\ f_y(x,y)+\lambda f_l=0 \end{cases}$$

Source/Load Stepping Does Not Alter Jacobian

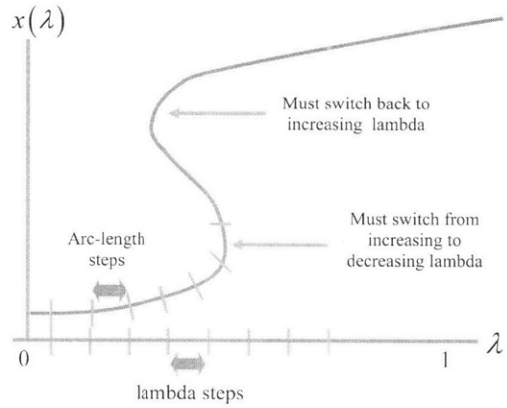
$$F(v)=i_d(v)+\frac{v-v_s}{R}$$

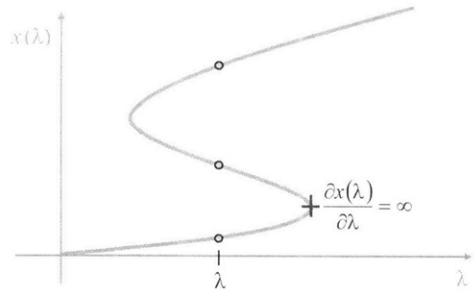
$$\widetilde{F}(v(\lambda),\lambda)=i_d(v)+\frac{v}{R}-\lambda\frac{v_s}{R}$$



Arc-Length Continuation (for multiple solutions)

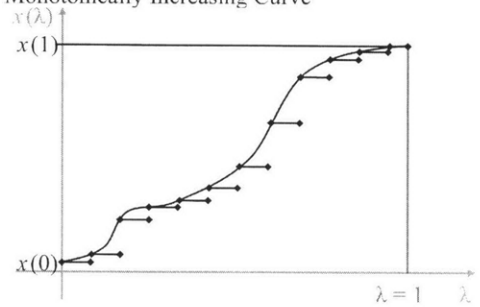
- Still can have problems



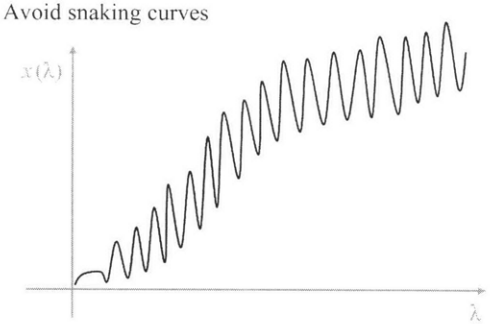


- Arc-length steps?

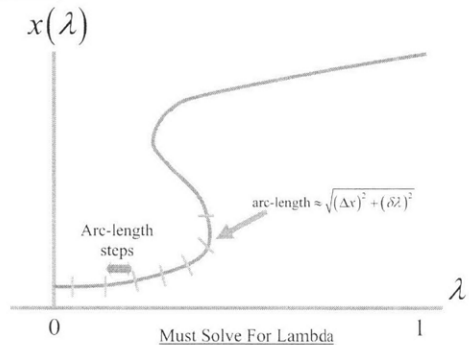
Monotonically Increasing Curve



Avoid snaking curves



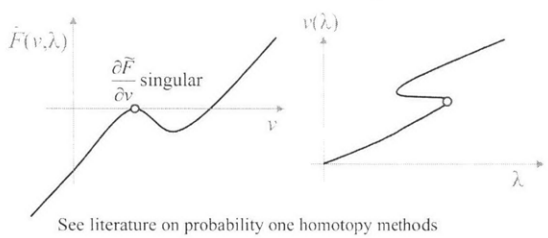
- Pictorially



Must Solve For Lambda

$$\tilde{F}(x,\lambda) = 0$$

$$(\lambda - \lambda_{prev})^2 + \left\|x - x(\lambda_{prev})\right\|_2^2 - arc^2 = 0$$



See literature on probability one homotopy methods

→ (Detect if close to singular)
in this case change arc length and hope to jump across the singularity

→ (or start decreasing λ but keep moving Δx in the same direction as previous step) etc…

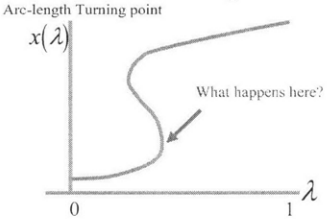
→ Better idea: Look for another continuation method.

- By Newton

$$\begin{bmatrix} \frac{\partial \tilde{F}(x^k,\lambda^k)}{\partial x} & \frac{\partial \tilde{F}(x^k,\lambda^k)}{\partial \lambda} \\ 2\left(x^k - x(\lambda_{prev})\right)^T & 2\left(\lambda^k - \lambda_{prev}\right) \end{bmatrix} \begin{bmatrix} x^{k+1} - x^k \\ \lambda^{k+1} - \lambda^k \end{bmatrix} =$$

$$= -\begin{bmatrix} \tilde{F}(x^k,\lambda^k) \\ \left(\lambda^k - \lambda_{prev}\right)^2 + \left\|x^k - x(\lambda_{prev})\right\|_2^2 - arc^2 \end{bmatrix}$$

❖ Arc-length Turning point



Upper left-hand Block is singular

$$\begin{bmatrix} \frac{\partial \tilde{F}(x^k,\lambda^k)}{\partial x} & \frac{\partial \tilde{F}(x^k,\lambda^k)}{\partial \lambda} \\ 2\left(x^k - x(\lambda_{prev})\right)^T & 2\left(\lambda^k - \lambda_{prev}\right) \end{bmatrix}$$

Jacobian Altering Scheme (for singular Jacobians)

❖ Practical Answer

Look for another continuation scheme

$$\tilde{F}(x(\lambda),\lambda) = \lambda F(x(\lambda)) + (1-\lambda)x(\lambda)$$

Observations:

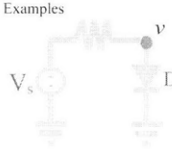
$\lambda = 0$ $\quad \tilde{F}(x(0),0) = x(0) = 0$

$\frac{\partial \tilde{F}(x(0),0)}{\partial x} = \mathbf{I}$ — Problem is easy to solve and Jacobian definitely nonsingular

$\lambda = 1$ $\quad \tilde{F}(x(1),1) = F(x(1))$

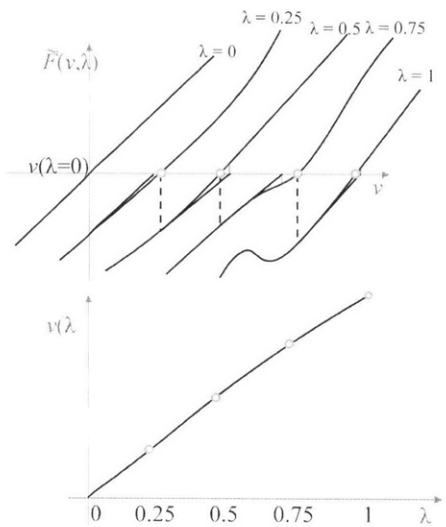
$\frac{\partial \tilde{F}(x(1),1)}{\partial x} = \frac{\partial F(x(1))}{\partial x}$ — Back to the original problem and original Jacobian

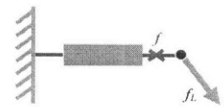
❖ Examples



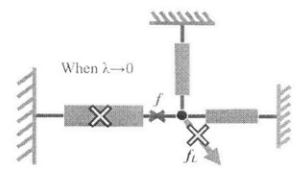
$$\lambda \frac{v - V_s}{R} + \lambda g_d(v) + (1-\lambda)v = 0$$

Try $\widetilde{F}(v,\lambda) = \lambda F(v,\lambda) + (1-\lambda)v = \lambda i(v) + \lambda \frac{v}{R} - \lambda \frac{v_s}{R} + (1-\lambda)v$

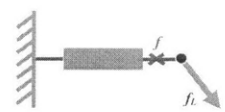




$$\widetilde{F}(x(\lambda),\lambda) = \begin{cases} \underbrace{\lambda f_x^*(x,y) + \lambda f_{x,L}}_{\lambda F(x)} + \underbrace{(1-\lambda)x}_{(1-\lambda)x(\lambda)} = 0 \\ \lambda f_y^*(x,y) + \lambda f_{y,L} + (1-\lambda)y = 0 \end{cases}$$



$$\widetilde{F}(x(0),0) = \begin{cases} x = 0 \\ y = 0 \end{cases}$$



$$\widetilde{F}(x(1),1) = \begin{cases} f_x^*(x,y) + f_{x,L} = 0 \\ f_y^*(x,y) + f_{y,L} = 0 \end{cases}$$

Jacobian Altering with Update Improvement

❖ Basic Algorithm

Solve $\widetilde{F}(x(0),0)$, $x(\lambda_{prev}) = x(0)$

$\delta\lambda = 0.01$, $\lambda = \delta\lambda$

While $\lambda < 1$ {

$\boxed{x^0(\lambda) = x(\lambda_{prev}) + ?}$

Try to Solve $\widetilde{F}(x(\lambda),\lambda) = 0$ with Newton

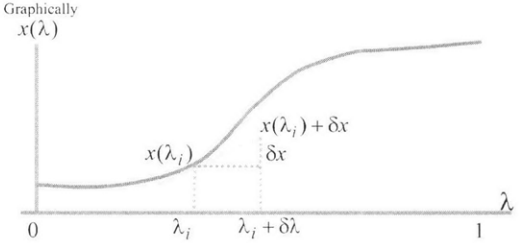
If Newton Converged

$x(\lambda_{prev}) = x(\lambda)$, $\lambda = \lambda + \delta\lambda$, $\delta\lambda = 2\delta\lambda$

Else

$\delta\lambda = \frac{1}{2}\delta\lambda$, $\lambda = \lambda_{prev} + \delta\lambda$

❖ Graphically



❖ Update Improvement

$$\widetilde{F}(x,\lambda)=\lambda F(x)+(1-\lambda)x$$

$$\delta\widetilde{F}=\frac{\delta\widetilde{F}}{\delta x}\delta x+\frac{\delta\widetilde{F}}{\delta\lambda}\delta\lambda$$

Have From last step's Newton what is it ?

Have From last step's Newton

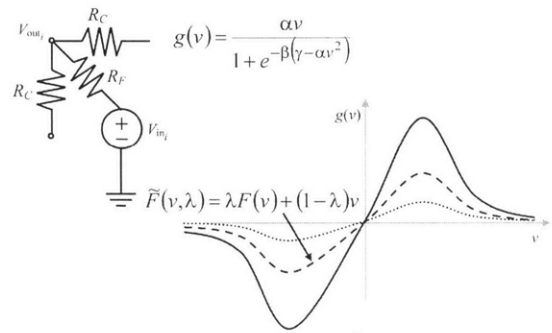
The Jacobian I used to solve the last Newton probem.

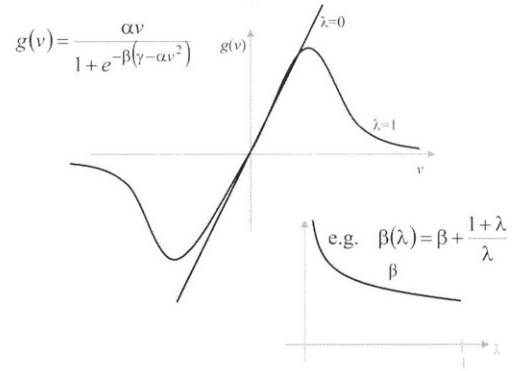
Set $\delta\widetilde{F}=0$

Solve for δx : $\left[\frac{\delta\widetilde{F}}{\delta x}\right]\delta x=-\left[F\left(x\left(\lambda_{prev}\right)\right)-x\left(\lambda_{prev}\right)\right]\delta\lambda$

$\Rightarrow$ Initial guess for next Newton :

$$x^0(\lambda)=x\left(\lambda_{prev}\right)+\delta x$$





3.14 ODE - Backward Euler, Forward Euler, Trapezoidal Rule

INTRODUCTION TO NUMERICAL SIMULATION

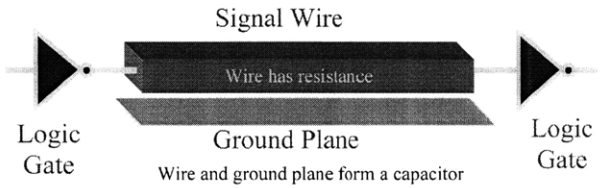
LECTURE 14.
Methods for Ordinary Differential Equations

TODAY'S OUTLINE:

- Initial Value Problem Examples
  - Signal Propagation (circuits with capacitors)
  - Space Frame Dynamics (struts and masses)
  - Chemical Reaction Dynamics
- Eigenvalue Analysis for Dynamical Systems
- Investigate the Simple Finite-Difference Methods
  - Forward-Euler, Backward-Euler, Trapezoidal Rule
  - Examine Properties Experimentally

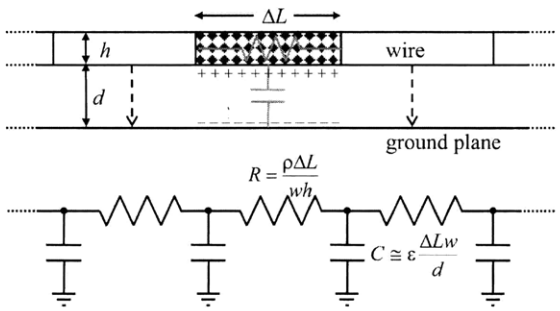
APPLICATION PROBLEMS

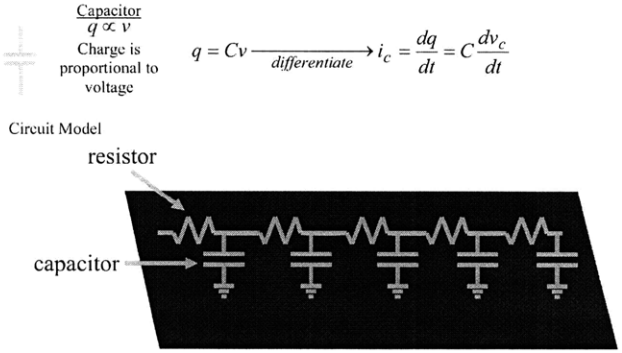
Signal Transmission in an Integrated Circuit



Wire and ground plane form a capacitor

- Metal Wires carry signals from gate to gate.
- How long is the signal delayed?





Capacitor
$q \propto v$
Charge is proportional to voltage

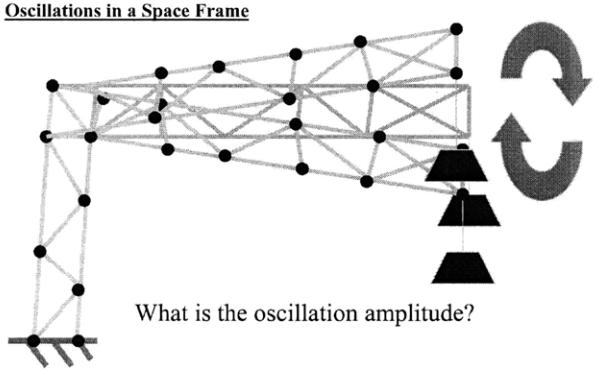
$$q = Cv \xrightarrow{differentiate} i_c = \frac{dq}{dt} = C\frac{dv_c}{dt}$$

- Circuit Model

Constructing the Model

- Cut the wire into sections.
- Model wire resistance with resistors.
- Model wire-plane capacitance with capacitors.

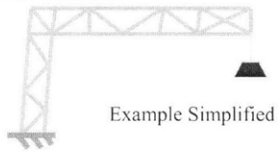
Oscillations in a Space Frame



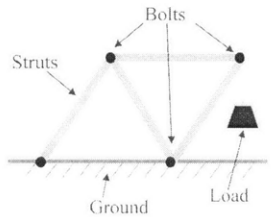
What is the oscillation amplitude?

2

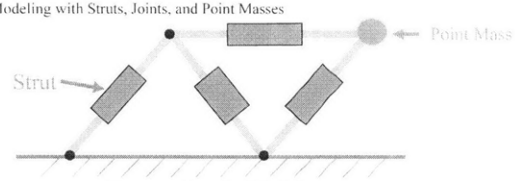
- Simplified Structure



Example Simplified for Illustration



- Modeling with Struts, Joints, and Point Masses



Constructing the Model

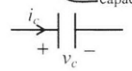
- Replace Metal Beams with Struts.
- Replace cargo with point mass.

Constitutive Equations

Capacitor

$$i_c = C\frac{dv_C}{dt}$$

capacitance



Point Mass

$$f = M\frac{d^2x}{dt^2}$$

mass

Involve t

Resistor

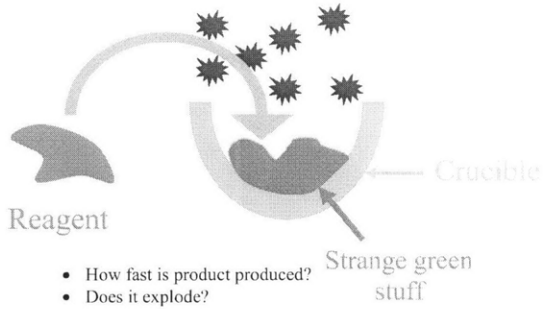
$$i_c = \frac{1}{R}v_R$$

Strut

$$f = \varepsilon(l_0 - l)$$

Independent of t

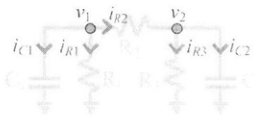
Chemical Reaction Dynamics



- How fast is product produced?
- Does it explode?

Signal Transmission in an Integrated Circuit

❖ A 2×2 Example

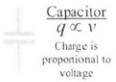


Constitutive Equations	Conservation Laws
$i_C = C\dfrac{dv_C}{dt}$	$i_{C_1} + i_{R_1} + i_{R_2} = 0$
$i_R = \dfrac{1}{R}v_R$	$i_{C_2} + i_{R_3} - i_{R_2} = 0$

Nodal Equations Yields 2×2 System

$$\begin{bmatrix} C_1 & 0 \\ 0 & C_2 \end{bmatrix}\begin{bmatrix} \dfrac{dv_1}{dt} \\ \dfrac{dv_2}{dt} \end{bmatrix} = -\begin{bmatrix} \dfrac{1}{R_1}+\dfrac{1}{R_2} & -\dfrac{1}{R_2} \\ -\dfrac{1}{R_2} & \dfrac{1}{R_3}+\dfrac{1}{R_2} \end{bmatrix}\begin{bmatrix} v_1 \\ v_2 \end{bmatrix}$$

Stamping Procedure for Nodal Formulation in Circuits



Capacitor

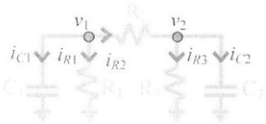
$q \propto v$

Charge is proportional to voltage

$$q = Cv \xrightarrow{\textit{differentiate}} i_c = \frac{dq}{dt} = C\frac{dv_c}{dt}$$

Conservation Laws:

$$i_{C_1} + i_{R_1} + i_{R_2} = 0$$
$$i_{C_2} + i_{R_3} - i_{R_2} = 0$$

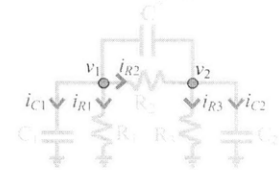


Use Nodal Analysis (substitute branch equations)

$$\begin{cases} C_1\dfrac{dv_1}{dt} + \dfrac{v_1}{R_1} + \dfrac{1}{R_2}(v_1 - v_2) = 0 \\ C_2\dfrac{dv_2}{dt} + \dfrac{v_2}{R_3} + \dfrac{1}{R_2}(v_2 - v_2) = 0 \end{cases}$$

$$\underbrace{\begin{bmatrix} C_1 & 0 \\ 0 & C_2 \end{bmatrix}}_{\substack{\text{Capacitance} \\ \text{Matrix}}}\begin{bmatrix} \dfrac{dv_1}{dt} \\ \dfrac{dv_2}{dt} \end{bmatrix} = -\underbrace{\begin{bmatrix} \dfrac{1}{R_1}+\dfrac{1}{R_2} & -\dfrac{1}{R_2} \\ -\dfrac{1}{R_2} & \dfrac{1}{R_2}+\dfrac{1}{R_3} \end{bmatrix}}_{\substack{\mathbf{G} \\ \text{Conductance} \\ \text{Matrix}}}\begin{bmatrix} v_1(t) \\ v_2(t) \end{bmatrix}$$

Stamping for Capacitors is like Stamping for Resistors



Try adding $\mathbf{C}^*$

$$\begin{cases} C_1\dfrac{dv_1}{dt} + C^*\left(\dfrac{dv_1}{dt} - \dfrac{dv_2}{dt}\right) = -\left[\dfrac{v_1(t)}{R_1} + \dfrac{1}{R_2}(v_1(t) - v_2(t))\right] \\ C^*\left(\dfrac{dv_2}{dt} - \dfrac{dv_1}{dt}\right) + C_2\dfrac{dv_2}{dt} = -\left[\dfrac{v_2(t)}{R_3} + \dfrac{1}{R_2}(v_2(t) - v_1(t))\right] \end{cases}$$

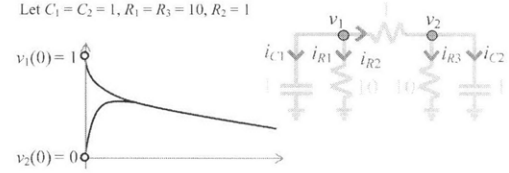
$$\begin{bmatrix} C_1 + C^* & -C^* \\ -C^* & C_2 + C^* \end{bmatrix}\begin{bmatrix} \dfrac{dv_1}{dt} \\ \dfrac{dv_2}{dt} \end{bmatrix} = -\begin{bmatrix} \dfrac{1}{R_1}+\dfrac{1}{R_2} & -\dfrac{1}{R_2} \\ -\dfrac{1}{R_2} & \dfrac{1}{R_2}+\dfrac{1}{R_3} \end{bmatrix}\begin{bmatrix} v_1(t) \\ v_2(t) \end{bmatrix}$$

EIGENVALUE ANALYSIS FOR DYNAMICAL SYSTEMS

Signal Transmission in an Integrated Circuit

❖ A 2×2 Example

Let $C_1 = C_2 = 1$, $R_1 = R_3 = 10$, $R_2 = 1$



$$\begin{bmatrix} C_1 & 0 \\ 0 & C_2 \end{bmatrix}\begin{bmatrix} \dfrac{dv_1}{dt} \\ \dfrac{dv_2}{dt} \end{bmatrix} = -\begin{bmatrix} \frac{1}{R_1}+\frac{1}{R_2} & -\frac{1}{R_2} \\ -\frac{1}{R_2} & \frac{1}{R_3}+\frac{1}{R_2} \end{bmatrix}\begin{bmatrix} v_1 \\ v_2 \end{bmatrix} \Rightarrow \frac{dx}{dt} = \underbrace{\begin{bmatrix} -1.1 & 1.0 \\ 1.0 & -1.1 \end{bmatrix}}_{A} x$$

Eigenvalues and Eigenvectors

$$A = \begin{bmatrix} 1 & -1 \\ 1 & 1 \end{bmatrix} \begin{bmatrix} -0.1 & 0 \\ 0 & -2.1 \end{bmatrix} \begin{bmatrix} 1 & -1 \\ 1 & 1 \end{bmatrix}^{-1}$$

eigenvectors eigenvalues

Aside: Eigenanalysis

Change of Variables $\Leftarrow \mathbf{E}\vec{y} = \vec{x}$

$$y_1 \begin{bmatrix} \\ \mathbf{E}_1 \\ \\ \end{bmatrix} + y_2 \begin{bmatrix} \\ \mathbf{E}_2 \\ \\ \end{bmatrix} + \cdots + y_N \begin{bmatrix} \\ \mathbf{E}_N \\ \\ \end{bmatrix} = \begin{bmatrix} \\ \vec{x} \\ \\ \end{bmatrix}$$

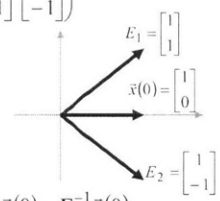
Columns of **E** (eigenvectors) are new basis

e.g.

$$\vec{x}(0) = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \qquad \mathbf{E} = \left(\begin{bmatrix} 1 \\ 1 \end{bmatrix} \begin{bmatrix} 1 \\ -1 \end{bmatrix} \right)$$

$$\vec{y}(0) = ?$$

$$y_1(0) \begin{bmatrix} 1 \\ 1 \end{bmatrix} + y_2(0) \begin{bmatrix} 1 \\ -1 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$



$$\vec{y}(0) = \begin{bmatrix} 1/2 \\ 1/2 \end{bmatrix} \qquad \vec{y}(0) = \mathbf{E}^{-1}\vec{x}(0)$$

Consider an ODE : $\dfrac{dx(t)}{dt} = \mathbf{A}x(t), \quad x(0) = x_0$

Eigendecomposition : $\mathbf{A} = \underbrace{\begin{bmatrix} \uparrow & \uparrow & & \uparrow \\ E_1 & E_2 & \cdots & E_n \\ \downarrow & \downarrow & & \downarrow \end{bmatrix}}_{\mathbf{E}} \begin{bmatrix} \lambda_1 & & \\ & \ddots & \\ & & \lambda_n \end{bmatrix} \begin{bmatrix} \uparrow & \uparrow & & \uparrow \\ E_1 & E_2 & \cdots & E_n \\ \downarrow & \downarrow & & \downarrow \end{bmatrix}^{-1}$

Change of variables : $\mathbf{E}y(t) = x(t) \Leftrightarrow y(t) = \mathbf{E}^{-1}x(t)$

Substituting : $\dfrac{d\mathbf{E}y(t)}{dt} = \mathbf{A}\mathbf{E}y(t), \quad \mathbf{E}y(0) = x_0$

Multiply by $\mathbf{E}^{-1}$: $\dfrac{dy(t)}{dt} = \mathbf{E}^{-1}\mathbf{A}\mathbf{E}y(t) = \begin{bmatrix} \lambda_1 & & \\ & \ddots & \\ & & \lambda_n \end{bmatrix} y(t)$ Decoupled Equations!

Decoupling : $\dfrac{dy_i(t)}{dt} = \lambda_i y_i(t) \Rightarrow y_i(t) = e^{\lambda_i t} y(0)$

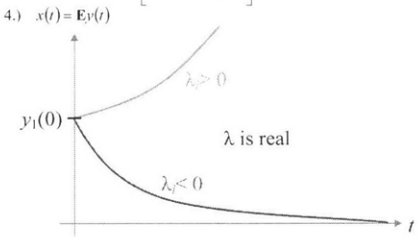
Steps for Solving $\dfrac{dx(t)}{dt} = \mathbf{A}x(t), \quad x(0) = x_0$

1.) Determine **E**, λ

2.) Compute $y(0) = \mathbf{E}^{-1}x_0$

3.) Compute $y(t) = \begin{bmatrix} e^{\lambda_1 t} & & \\ & \ddots & \\ & & e^{\lambda_n t} \end{bmatrix} y(0)$

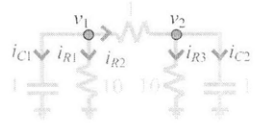
4.) $x(t) = \mathbf{E}y(t)$



$$x(t) = \underbrace{\mathbf{E}}_{\text{convergence}} \begin{bmatrix} e^{\lambda_1 t} & & & \\ & e^{\lambda_2 t} & & \\ & & \ddots & \\ & & & e^{\lambda_n t} \end{bmatrix} \underbrace{\mathbf{E}^{-1} x(0)}_{\text{convergence}}$$

Signal Transmission in an Integrated Circuit

- A 2×2 Example

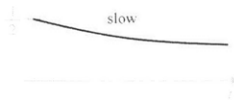


$$A = \begin{bmatrix} -1.1 & 1 \\ 1 & -1.1 \end{bmatrix} \quad E = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \quad \Lambda = \begin{bmatrix} -0.1 & 0 \\ 0 & -2.1 \end{bmatrix}$$

$$\bar{x}(0) = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \quad \bar{y}(0) = \begin{bmatrix} 1/2 \\ 1/2 \end{bmatrix}$$

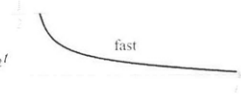
$$\frac{dy_1(t)}{dt} = -0.1 y_1(t)$$

$$y_1(t) = \tfrac{1}{2} e^{-0.1t} = y_1(0) e^{\lambda_1 t}$$



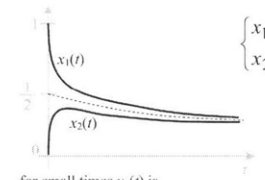
$$\frac{dy_2(t)}{dt} = -2.1 y_1(t)$$

$$y_2(t) = \tfrac{1}{2} e^{-2.1t} = y_2(0) e^{\lambda_2 t}$$



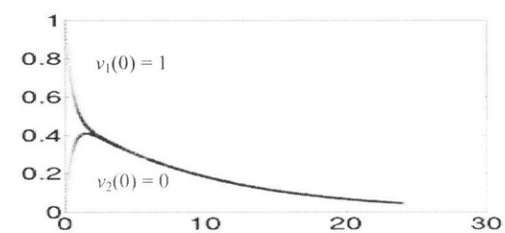
$$\bar{x}(t) = \mathbf{E}\bar{y}(t) = y_1(t)\begin{bmatrix} 1 \\ 1 \end{bmatrix} + y_2(t)\begin{bmatrix} 1 \\ -1 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \end{bmatrix} e^{-0.1t} y_1(0) + \begin{bmatrix} 1 \\ -1 \end{bmatrix} e^{-2.1t} y_2(0)$$

$$\begin{cases} x_1(t) = y_1(t) + y_2(t) \\ x_2(t) = y_1(t) - y_2(t) \end{cases}$$



for small times $x_2(t)$ is very active (fast mode) and decreases rapidly (the difference gets smaller)

for large times the difference ≈ 0 (fast mode disappears) they both follow the slow mode $x_1(t) \approx y_1(t) \approx x_2(t)$

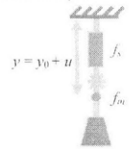


Notice two time scale behavior

- v_1 and v_2 come together quickly (fast eigenmode).
- v_1 and v_2 decay to zero slowly (slow eigenmode).

Struts, Joints and Point Mass Example

❖ A 2×2 Example



Constitutive Equations	Conservation Laws
$f_s = EA_c \dfrac{y - y_0}{y_0} = \dfrac{EA_c}{y_0} u$	$f_s + f_m = 0$
$f_m = M \dfrac{d^2 u}{dt^2}$	

Define v as velocity (du/dt) to yield a 2×2 System

$$\begin{cases} M \dfrac{d^2 u}{dt^2} + \dfrac{EA_c}{y_0} u(t) = 0 \\ v(t) = \dfrac{du}{dt} \end{cases} \rightarrow \begin{cases} M \dfrac{dv}{dt} = -\dfrac{EA_c}{y_0} u(t) \\ \dfrac{du}{dt} = v(t) \end{cases}$$

$$\begin{bmatrix} M & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \dfrac{dv}{dt} \\ \dfrac{du}{dt} \end{bmatrix} = \begin{bmatrix} 0 & -\dfrac{EA_c}{y_0} \\ 1 & 0 \end{bmatrix} \begin{bmatrix} v \\ u \end{bmatrix}$$

Let $M = 1$, $\dfrac{EA_c}{y_0} = 1$

$$\begin{bmatrix} M & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \dfrac{dv}{dt} \\ \dfrac{du}{dt} \end{bmatrix} = \begin{bmatrix} 0 & -\dfrac{EA_c}{y_0} \\ 1 & 0 \end{bmatrix} \begin{bmatrix} v \\ u \end{bmatrix} \Rightarrow \frac{dx}{dt} = \underbrace{\begin{bmatrix} 0 & -1.0 \\ 1.0 & 0 \end{bmatrix}}_{A} x$$

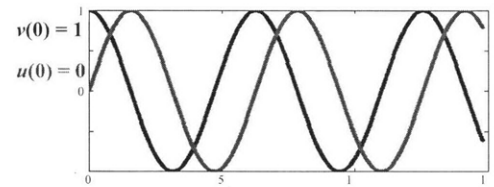
Eigenvalues and Eigenvectors

$$A = \begin{bmatrix} -1 & -1 \\ i & -i \end{bmatrix} \begin{bmatrix} i & 0 \\ 0 & -i \end{bmatrix} \begin{bmatrix} -1 & -1 \\ i & -i \end{bmatrix}^{-1}$$

$$y_1(t) = e^{it} y_1(0) = y_1(0)(\cos t + i \sin t)$$
$$y_2(t) = e^{-it} y_2(0) = y_2(0)(\cos t - i \sin t)$$

eigenvectors eigenvalues

$$\underset{\text{real}}{\mathbf{A}} \; \underset{\text{complex}}{\vec{E}} = \underset{\text{complex}}{\lambda} \; \vec{E}$$



Note the system has imaginary eigenvalues

- Persistent Oscillation
- Velocity, v, peaks when displacement, u, is zero.

Chemical Reaction Example

❖ A 2×2 Example

Amount of reactant = R, the temperature = T

$$\frac{dT}{dt} = -T + R$$

More reactant causes the temperature to rise, higher temperatures increase heat dissipation causing temperature to fall.

$$\frac{dR}{dt} = -R + 4T$$

Higher temperature raises reaction rates, increased reactant interferes with reaction and slows rate.

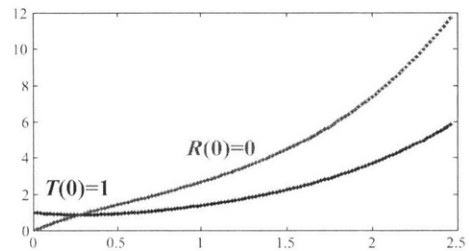
$$\begin{bmatrix} \dfrac{dT}{dt} \\ \dfrac{dR}{dt} \end{bmatrix} = \begin{bmatrix} -1 & 1 \\ 4 & -1 \end{bmatrix} \begin{bmatrix} T \\ R \end{bmatrix} \Rightarrow \frac{dx}{dt} = \underbrace{\begin{bmatrix} -1 & 1 \\ 4 & -1 \end{bmatrix}}_{A} x$$

Eigenvalues and Eigenvectors

$$A = \begin{bmatrix} 1 & -1 \\ 2 & 2 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & -3 \end{bmatrix} \begin{bmatrix} 1 & -1 \\ 2 & 2 \end{bmatrix}^{-1}$$

$$y_1(t) = e^{t} y_1(0)$$
$$y_2(t) = e^{-3t} y_2(0)$$

eigenvectors eigenvalues



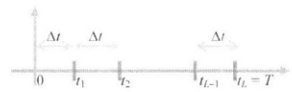
Note the system has a positive eigenvalue

Solutions grow exponentially with time.

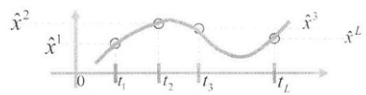
FINITE DIFFERENCE METHODS

Basic Concepts

FIRST: discretize time



SECOND: Represent $x(t)$ using values at t_l

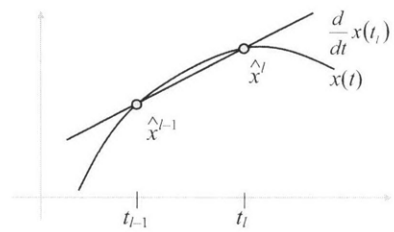


$$\hat{x}^l \cong x(t_l)$$

Approximate solution — Exact solution

THIRD: Approximate $\frac{dx}{dt}$ using the discrete $\hat{x}^l$

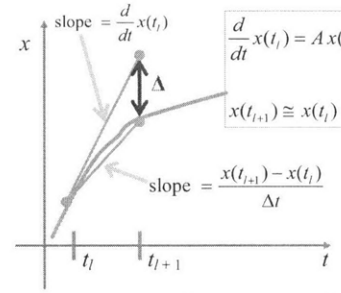
Example: $\frac{d}{dt}x(t_l) \overset{?}{\approx} \frac{\hat{x}^{l+1} - \hat{x}^l}{\Delta t_{l+1}} \overset{?}{\approx} \frac{d}{dt}x(t_{l+1})$



$$\frac{d}{dt}x(t_l) = \frac{\hat{x}^l - \hat{x}^{l-1}}{\underbrace{t_l - t_{l-1}}_{\Delta t}}$$

$$\frac{x(t_{l-1}) - x(t_l)}{\Delta t} \cong \begin{cases} \frac{dx}{dt}(t_l) & \text{F.E.} \quad \frac{x(t_{l-1}) - x(t_l)}{\Delta t} = \mathbf{A}x(t_l) \\ \frac{dx}{dt}(t_{l+1}) & \text{B.E.} \quad \frac{x(t_{l-1}) - x(t_l)}{\Delta t} = \mathbf{A}x(t_{l+1}) \\ \frac{1}{2}\left[\frac{dx}{dt}(t_l) + \frac{dx}{dt}(t_{l+1})\right] & \text{TRAP} \quad \frac{x(t_{l-1}) - x(t_l)}{\Delta t} = \ldots \\ & \quad = \frac{1}{2}\left[\mathbf{A}x(t_l) + \mathbf{A}x(t_{l+1})\right] \end{cases}$$

- Forward Euler
  - Approximation



$$\frac{d}{dt}x(t_l) = A\,x(t_l) \cong \frac{x(t_{l+1}) - x(t_l)}{\Delta t}$$

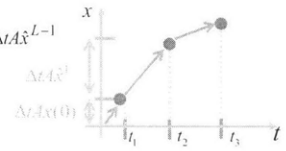
or

$$x(t_{l+1}) \cong x(t_l) + \Delta t\, A\, x(t_l)$$

$$\Delta = x(t_{l+1}) - \left(x(t_l) + \Delta t\, A\, x(t_l)\right)$$

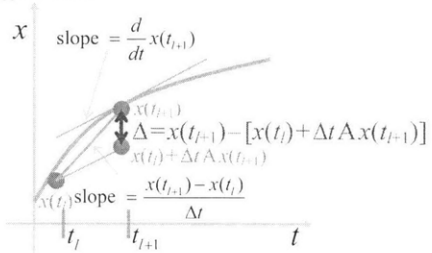
- Algorithm

$$x(t_1) \cong \hat{x}^1 = x(0) + \Delta t A x(0)$$
$$x(t_2) \cong \hat{x}^2 = \hat{x}^1 + \Delta t A \hat{x}^1$$
$$\vdots$$
$$x(t_L) \cong \hat{x}^L = \hat{x}^{L-1} + \Delta t A \hat{x}^{L-1}$$



❖ Backward Euler

- Approximation



$$\frac{d}{dt} x(t_{l+1}) = A\, x(t_{l+1}) \cong \frac{x(t_{l+1}) - x(t_l)}{\Delta t}$$

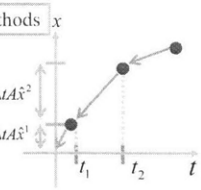
or

$$x(t_{l+1}) \cong x(t_l) + \Delta t\, A\, x(t_{l+1})$$

- Algorithm

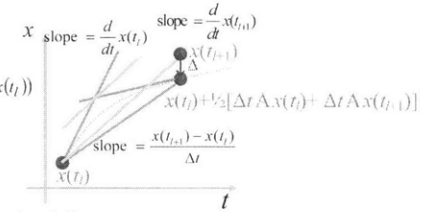
Solve with LU or iterative methods

$$x(t_1) \cong \hat{x}^1 = x(0) + \Delta t \mathbf{A} \hat{x}^1$$
$$[\mathbf{I} - \Delta t \mathbf{A}] \hat{x}^1 = x(0)$$
$$x(t_2) \cong \hat{x}^2 = [\mathbf{I} + \Delta t \mathbf{A}]^{-1} \hat{x}^1$$
$$\vdots$$
$$x(t_L) \cong \hat{x}^L = [\mathbf{I} + \Delta t \mathbf{A}]^{-1} \hat{x}^{L-1}$$



❖ Trapezoidal Rule

$$\frac{1}{2}\left(\frac{d}{dt} x(t_{l+1}) + \frac{d}{dt} x(t_l)\right)$$
$$= \frac{1}{2}(\mathbf{A}x(t_{l+1}) + \mathbf{A}x(t_l))$$
$$\cong \frac{x(t_{l+1}) - x(t_l)}{\Delta t}$$

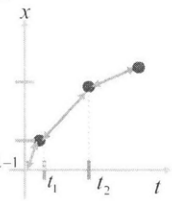


$$x(t_{l+1}) \cong x(t_l) + \tfrac{1}{2}\Delta t \mathbf{A}(x(t_{l+1}) + x(t_l))$$
$$\Delta = (x(t_{l+1}) - \frac{1}{2}\Delta t A x(t_l)) - (x(t_l) + \frac{1}{2}\Delta t A x(t_{l+1}))$$

- Algorithm

Solve with LU or iterative methods

$$x(t_1) \cong \hat{x}^1 = x(0) + \tfrac{\Delta t}{2}\left(\mathbf{A}x(0) + \mathbf{A}\hat{x}^1\right)$$
$$\Rightarrow \left[\mathbf{I} - \tfrac{\Delta t}{2}\mathbf{A}\right]\hat{x}^1 = \left[\mathbf{I} + \tfrac{\Delta t}{2}\mathbf{A}\right]x(0)$$
$$x(t_2) \cong \hat{x}^2 = \left[\mathbf{I} - \tfrac{\Delta t}{2}\mathbf{A}\right]^{-1}\left[\mathbf{I} + \tfrac{\Delta t}{2}\mathbf{A}\right]\hat{x}^1$$
$$\vdots$$
$$x(t_L) \cong \hat{x}^L = \left[\mathbf{I} - \tfrac{\Delta t}{2}\mathbf{A}\right]^{-1}\left[\mathbf{I} + \tfrac{\Delta t}{2}\mathbf{A}\right]\hat{x}^{L-1}$$



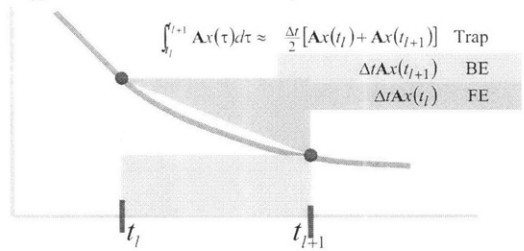
❖ Numerical Integration View

$$\frac{d}{dt}x(t) = \mathbf{A}x(t) \Rightarrow x(t_{l+1}) = x(t_l) + \int_{t_l}^{t_{l+1}} \mathbf{A}x(\tau)d\tau$$

$$\int_{t_l}^{t_{l+1}} \mathbf{A}x(\tau)d\tau \approx \frac{\Delta t}{2}\left[\mathbf{A}x(t_l) + \mathbf{A}x(t_{l+1})\right] \quad \text{Trap}$$

$$\Delta t\mathbf{A}x(t_{l+1}) \quad \text{BE}$$

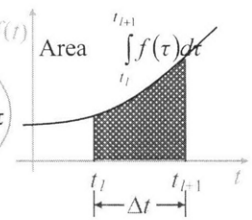
$$\Delta t\mathbf{A}x(t_l) \quad \text{FE}$$



$$\frac{dx}{dt} = f(t)$$

$$x(t_{l+1}) = x(t_l) + \int_{t_l}^{t_{l+1}} f(\tau)d\tau$$

Fundamental Theorem of Calculus



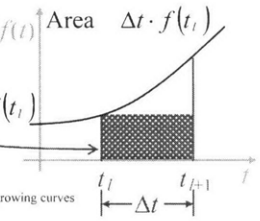
Forward Euler (F.E.)

$$\left.\frac{dx}{dt}\right|_{t_l} \cong \frac{x(t_{l+1}) - x(t_l)}{\Delta t}$$

$$x(t_{l+1}) \cong x(t_l) + \Delta t \cdot f(t_l)$$

Approximate the area with the box

Underestimates exact result for growing curves



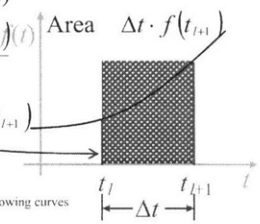
Backward Euler (B.E.)

$$\left.\frac{dx}{dt}\right|_{t_{l+1}} \cong \frac{x(t_{l+1}) - x(t_l)}{\Delta t}$$

$$x(t_{l+1}) \cong x(t_l) + \Delta t \cdot f(t_{l+1})$$

Approximate the area with the box

Overestimates exact result for growing curves

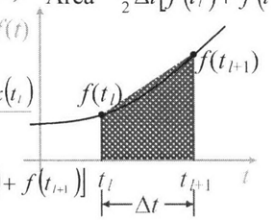


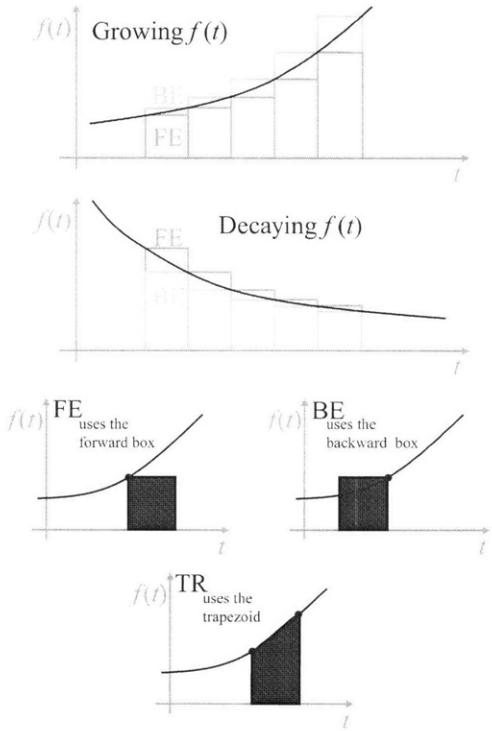
Trapezoidal Rule (T.R.)

Area $\frac{1}{2}\Delta t\left[f(t_l) + f(t_{l+1})\right]$

$$\left[\left.\frac{dx}{dt}\right|_{t_l} + \left.\frac{dx}{dt}\right|_{t_{l+1}}\right] \cong \frac{x(t_{l+1}) - x(t_l)}{\Delta t}$$

$$x(t_{l+1}) \cong x(t_l) + \tfrac{1}{2}\Delta t\left[f(t_l) + f(t_{l+1})\right]$$

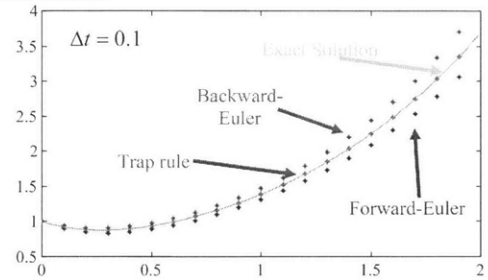




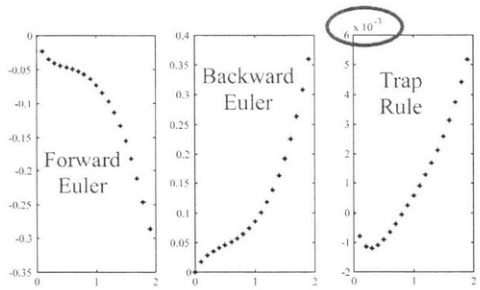
- Summary
  - Trap Rule, Forward-Euler, Backward-Euler
    - All are one-step methods
    - $\hat{x}^l$ is computed using only $\hat{x}^{l-1}$, not $\hat{x}^{l-2}$, $\hat{x}^{l-3}$, etc.
  - Forward-Euler is simplest
    - No equation solution ⇨ explicit method
    - Boxcar approximation to integral
  - Backward-Euler is more expensive
    - Equation solution each step ⇨ implicit method
  - Trapezoidal Rule might be more accurate
    - Equation solution each step ⇨ implicit method
    - Trapezoidal approximation to integral

Properties – Numerical Experiments

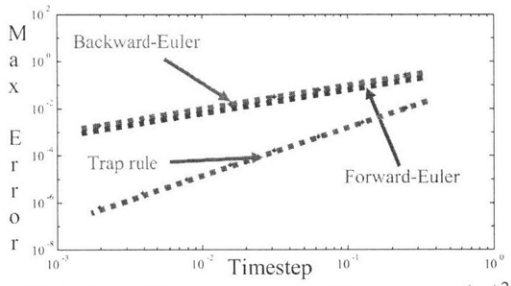
- Unstable Reaction



FE and BE results have larger errors than Trap Rule, and the errors grow with time.

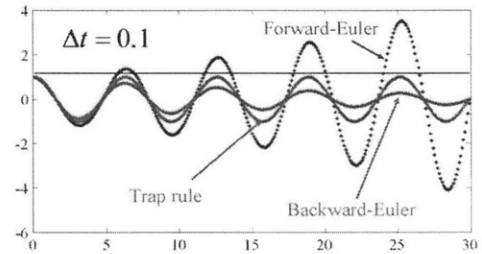


All methods have errors which grow exponentially



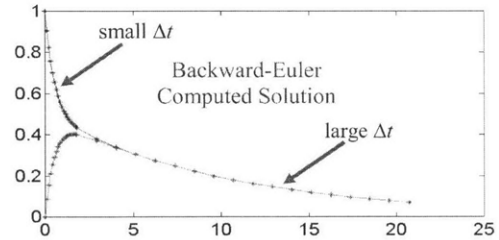
For FE and BE, $error \propto \Delta t$ For Trap, $error \propto (\Delta t)^2$

- Oscillating Strut and Mass

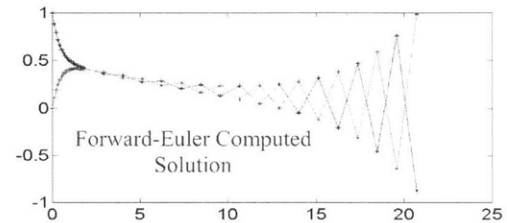


Why does FE result grow, BE result decay and the Trap rule preserves oscillations?

- Two timescale RC Circuit



With Backward-Euler it is easy to use small timesteps for the fast dynamics and then switch to large timesteps for the slow decay



The Forward-Euler is accurate for small timesteps, but goes unstable when the timestep is enlarged

	F.E. explicit	B.E. implicit	Trap implicit
COST	cheap	expensive	expensive
ACCURACY	ok	ok	great
CONVERGENCE	linear	linear	quadratic
OSCILLATION	generates	dissipates	conserves
ADJUST Δt	unstable	ok	ok

- Summary
  - Convergence
    - Did the computed solution approach the exact solution?
    - Why did the trap rule approach faster than BE or FE?
  - Energy Preservation
    - Why did BE produce a decaying oscillation?
    - Why did FE produce a growing oscillation?
    - Why did trap rule maintain oscillation amplitude?
  - Two timeconstant (stiff) problems
    - Why did FE go unstable when the timestep increased?

3.15 ODE - Multistep Methods I

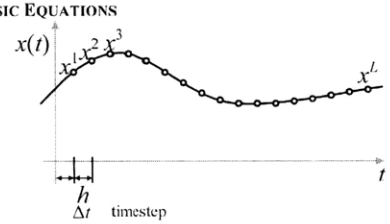
INTRODUCTION TO NUMERICAL SIMULATION

LECTURE 15.
Multistep Methods I

TODAY'S OUTLINE:

- Multistep Methods
  - e.g. FE, BE, TR
  - Convergence Definitions
  - Local Truncation Error (LTE) and Global Error (GE)
  - Example: LTE and GE for F.E.
  - Minimizing LTE (consistency)
  - Minimizing the accumulation of errors (stability)

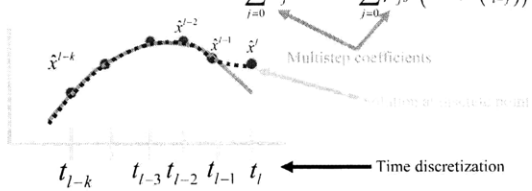
BASIC EQUATIONS

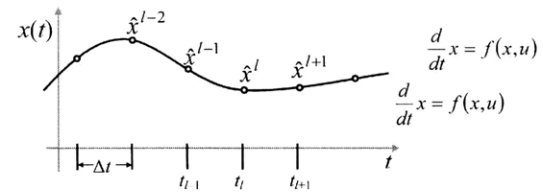


General Notation

Nonlinear Differential Equation: $\frac{d}{dt}x(t) = f(x(t),u(t))$

k-Step Multistep Approach: $\sum_{j=0}^{k}\alpha_j \hat{x}^{l-j} = \Delta t \sum_{j=0}^{k}\beta_j f\left(\hat{x}^{l-j}, u\left(t_{l-j}\right)\right)$





Forward Euler

$$\hat{x}^{l+1} = \hat{x}^{l} + \Delta t \cdot \dot{x}(l\Delta t)$$
$$= \hat{x}^{l} + \Delta t \cdot f\left(\hat{x}^{l}\right)$$

Backward Euler

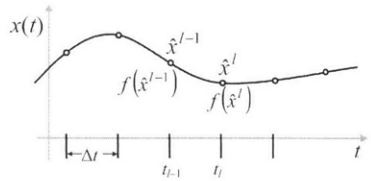
$$\hat{x}^{l+1} = \hat{x}^{l} + \Delta t \cdot \dot{x}(l\Delta t)$$
$$= \hat{x}^{l} + \Delta t \cdot f\left(\hat{x}^{l+1}\right)$$

Trapezoidal Rule

$$\hat{x}^{l+1} = \hat{x}^{l} + \Delta t \cdot \left[\tfrac{1}{2} f\left(\hat{x}^{l}\right) + \tfrac{1}{2} f\left(\hat{x}^{l+1}\right)\right]$$

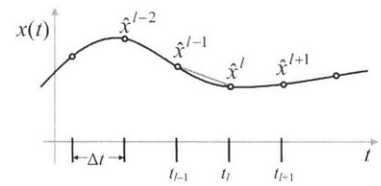
Common Algorithms

Multistep Equation:	$\sum_{j=0}^{k}\alpha_j \hat{x}^{l-j} = \Delta t \sum_{j=0}^{k}\beta_j f\left(\hat{x}^{l-j}, u\left(t_{l-j}\right)\right)$
Forward-Euler Approximation:	$x(t_l) \approx x(t_{l-1}) + \Delta t f(x(t_{l-1}), u(t_{l-1}))$
FE Discrete Equation:	$\hat{x}^{l} - \hat{x}^{l-1} = \Delta t\, f(x(t_{l-1}), u(t_{l-1}))$
Multistep Coefficients:	$k=1,\ \alpha_0=1,\ \alpha_1=-1,\ \beta_0=0,\ \beta_1=1$
BE Discrete Equation:	$\hat{x}^{l} - \hat{x}^{l-1} = \Delta t f\left(\hat{x}^{l}, u(t_l)\right)$
Multistep Coefficients:	$k=1,\ \alpha_0=1,\ \alpha_1=-1,\ \beta_0=1,\ \beta_1=0$
Trap Discrete Equation:	$\hat{x}^{l} - \hat{x}^{l-1} = \frac{\Delta t}{2}\left[f\left(\hat{x}^{l}, u(t_l)\right) + f\left(\hat{x}^{l-1}, u(t_{l-1})\right)\right]$
Multistep Coefficients:	$k=1,\ \alpha_0=1,\ \alpha_1=-1,\ \beta_0=\frac{1}{2},\ \beta_1=\frac{1}{2}$



$$\alpha_0 \hat{x}^l + \alpha_1 \hat{x}^{l-1} = \Delta t\left[\beta_0 f\left(\hat{x}^l\right) + \beta_1 f\left(\hat{x}^{l-1}\right)\right]$$

Forward Euler	1	−1	0	1
Backward Euler	1	−1	1	0
Trapezoidal Rule	1	−1	½	½



$$\frac{\hat{x}^l - \hat{x}^{l-1}}{\Delta t} = ?$$

Forward Euler

$$\frac{\hat{x}^l - \hat{x}^{l-1}}{\Delta t} = \frac{dx((l-1)\Delta t)}{dt} = f\left(\hat{x}\left(t_{l-1}\right)\right) = f\left(\hat{x}^{l-1}\right)$$

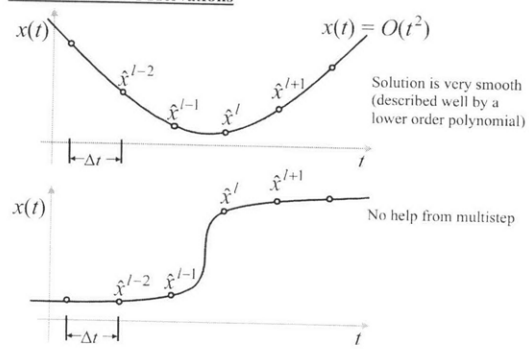
Backward Euler

$$\frac{\hat{x}^l - \hat{x}^{l-1}}{\Delta t} = \frac{dx(l\Delta t)}{dt} = f\left(\hat{x}^l\right)$$

Trapezoidal Rule

$$\frac{\hat{x}^l - \hat{x}^{l-1}}{\Delta t} = \tfrac{1}{2} f\left(\hat{x}^{l-1}\right) + \tfrac{1}{2} f\left(\hat{x}^l\right)$$

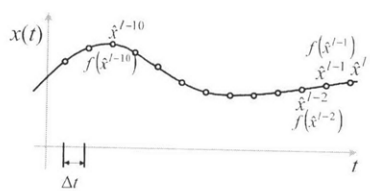
Definitions and Observations



Multistep Equation $\displaystyle\sum_{j=0}^{k} \alpha_j \hat{x}^{l-j} = \Delta t \sum_{j=0}^{k} \beta_j f\left(\hat{x}^{l-j}, u(t_{l-j})\right)$

1. If $\beta_0 \neq 0$ the multistep method is implicit
2. A k-step multistep method uses k previous x's and f's
3. A normalization is needed, $\alpha_0 = 1$ is common
4. A k-step method has $2k+1$ free coefficients

How does one pick good coefficients?



$\dot{x} = \lambda x \quad x(0) = x_0$

$k = 1$ Multistep Method

$$\alpha_0 \hat{x}^{l} + \alpha_1 \hat{x}^{l-1} = \Delta t\left(\beta_0 \lambda \hat{x}^{l} + \beta_1 \lambda \hat{x}^{l-1}\right)$$

Forward Euler $\quad \hat{x}^{l} = \hat{x}^{l-1} + \Delta t \cdot \lambda \hat{x}^{l-1} \quad \alpha_0 = 1 \quad \beta_0 = 0 \quad \alpha_1 = -1 \quad \beta_1 = 1$

When does $x(t)$ grow?

$$\lambda > 0 \quad x(t) = x_0 e^{\lambda t}$$

What about F.E.?

$$\hat{x}^{l} = \underbrace{(1 + \Delta t \cdot \lambda)}_{\hat{x}^{l} \text{ grows if } (1+\Delta t \cdot \lambda_i) > 1} \hat{x}^{l-1}$$

Simplified Problems for Analysis

Scalar ODE : $\dfrac{d}{dt} v(t) = \lambda v(t), \; v(0) = v_0 \quad \lambda \in C$

Why such a simple Test Problem?

- Nonlinear Analysis has many *unrevealing* subtleties
- Scalar is equivalent to vector for multistep methods

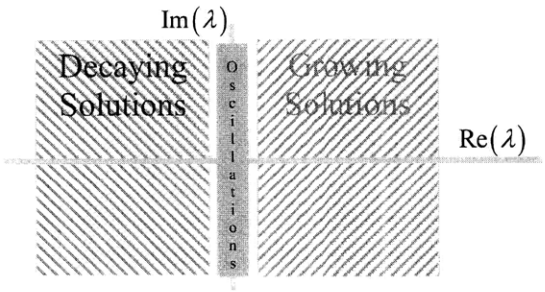
$$\frac{d}{dt} x(t) = Ax(t) \xrightarrow{\text{multistep discretization}} \sum_{j=0}^{k} \alpha_j \hat{x}^{l-j} = \Delta t \sum_{j=0}^{k} \beta_j A \hat{x}^{l-j}$$

Let $Ey(t) = x(t)$ $\longrightarrow$ $\displaystyle\sum_{j=0}^{k} \alpha_j \hat{y}^{l-j} = \Delta t \sum_{j=0}^{k} \beta_j E^{-1} A E \hat{y}^{l-j}$

Decoupled Equations $\longrightarrow$ $\displaystyle\sum_{j=0}^{k} \alpha_j \hat{y}^{l-j} = \Delta t \sum_{j=0}^{k} \beta_j \begin{bmatrix} \lambda_1 & & \\ & \ddots & \\ & & \lambda_n \end{bmatrix} \hat{y}^{l-j}$

Scalar Multistep formula : $\displaystyle\sum_{j=0}^{k} \alpha_j \hat{v}^{l-j} = \Delta t \sum_{j=0}^{k} \beta_j \lambda \hat{v}^{l-j}$

Must consider ALL $\lambda \in C$

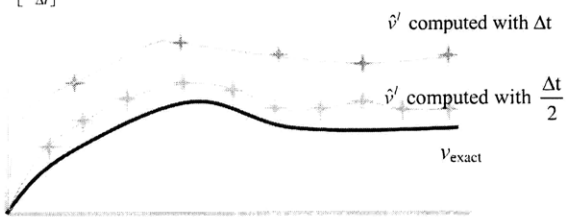


CONVERGENCE ANALYSIS

Convergence Definition

Definition: A multistep method for solving initial value problems on $[0,T]$ is said to be convergent if given any initial condition

$$\max_{l\in\left[0,\frac{T}{\Delta t}\right]}\left\|\hat{v}^l - v(l\Delta t)\right\| \to 0 \text{ as } \Delta t \to 0$$



Order-*p* Convergence

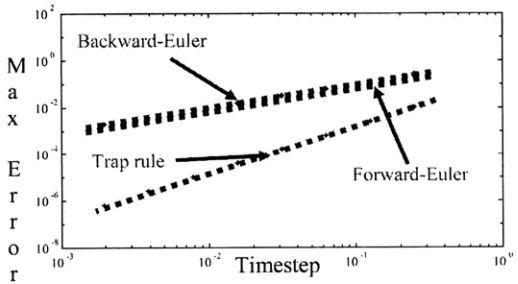
Definition: A multistep method for solving initial value problems on $[0,T]$ is said to be convergent if given any initial condition

$$\max_{l\in\left[0,\frac{T}{\Delta t}\right]}\left\|\hat{v}^l - v(l\Delta t)\right\| \le C(\Delta t)^p$$

for all Δt less than a given Δt_0

Forward- and Backward-Euler are order 1 convergent;
Trapezoidal Rule is order 2 convergent

Reaction Equation Example



For FE and BE, $error \propto \Delta t$ For Trap, $error \propto (\Delta t)^2$

LOCAL TRUNCATION ERROR (LTE) AND GLOBAL ERROR (GE)

Convergence Analysis

- Two Conditions for Convergence
  1. Local Condition: "One step" errors are small (consistency)
     Typically verified using Taylor Series
  2. Global Condition: The single step errors do not grow too quickly (stability)
     All one-step ($k = 1$) methods are stable in this sense.
     Multi-step ($k > 1$) methods require careful analysis

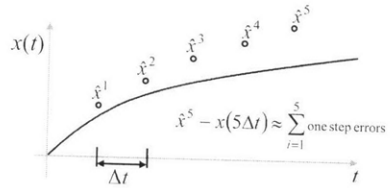
$$\hat{x}^l = \hat{x}^{l-1} + \Delta t\cdot\lambda\hat{x}^{l-1} \qquad \Rightarrow \hat{x}^l = (1+\Delta t\cdot\lambda)\hat{x}^{l-1}$$

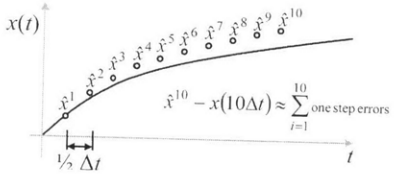
$$x(l\Delta t) = x((l-1)\Delta t) + \Delta t\cdot\underbrace{\lambda x((l-1)\Delta t)}_{\dot{x}((l-1)\Delta t)} + O(\Delta t^2)$$

Forward Euler Global Error

$$E^l = E^{l-1} + \Delta t\cdot\lambda E^{l-1} + e^l$$

$$E^l - (1+\Delta t\cdot\lambda)E^{l-1} = e^l$$





❖ Global Error Equation

Multistep formula : $\sum_{j=0}^{k} \alpha_j \hat{v}^{l-j} - \Delta t \sum_{j=0}^{k} \beta_j \lambda \hat{v}^{l-j} = 0$

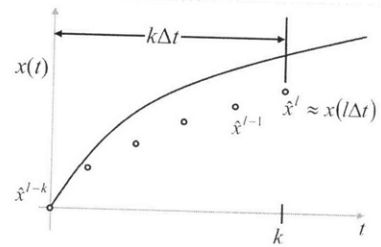
Exact solution Amost satisfies Multistep Formula : $\sum_{j=0}^{k} \alpha_j v(t_{l-j}) - \Delta t \sum_{j=0}^{k} \beta_j \frac{d}{dt} v(t_{l-j}) = \boxed{e^l}$

Local Truncation Error (LTE)

Global Error : $E^l \equiv v(t_l) - \hat{v}^l$

Difference equation relates LTE to Global error

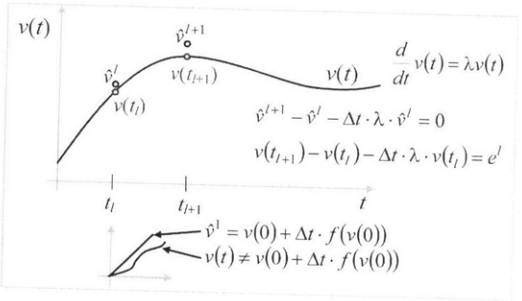
$$(\alpha_0 - \lambda \Delta t \beta_0) E^l + (\alpha_1 - \lambda \Delta t \beta_1) E^{l-1} + \cdots + (\alpha_k - \lambda \Delta t \beta_k) E^{l-k} = e^l$$

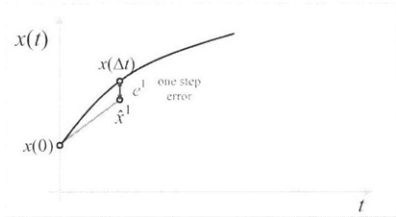


$x(t)$ for $t \in [0, k\Delta t]$

$x(t) \cong a_0 + a_1 t + \cdots + a_{10} t^{10} + \cdots$

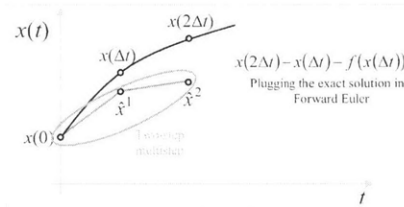
$x(t) \cong x(0) + \dot{x}(0) t + \frac{1}{2} \ddot{x}(0) t^2 + \cdots$





$$\left.\begin{aligned}&\alpha_0\underbrace{\left[v(t_l)-\hat{v}^l\right]}_{E^l}-\Delta t\beta_0\lambda\underbrace{\left(v(t_l)-\hat{v}^l\right)}_{E^l}+\cdots\\&\cdots+\alpha_1\left[v(t_{l-1})-\hat{v}^{l-1}\right]-\Delta t\beta_1\lambda\left(v(t_{l-1})-\hat{v}^{l-1}\right)+\cdots\end{aligned}\right\}k=e^l$$

$$(\alpha_0-\Delta t\beta_0\lambda)E^l+(\alpha_1-\Delta t\beta_1\lambda)E^{l-1}+\ldots=e^l$$



$x(2\Delta t)-x(\Delta t)-f(x(\Delta t))=e^2$

Plugging the exact solution into Forward Euler

EXAMPLE: LTE AND GE FOR FORWARD-EULER

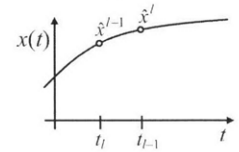
Convergence Analysis

- One Step Methods

  Definition: A one-step method for solving initial value problems on an interval $[0,T]$ is said to be consistent if for any A and any initial condition

$$\frac{\left|\hat{x}^1-x(\Delta t)\right|}{\Delta t}\to 0 \quad \text{as} \quad \Delta t\to 0$$

$$\hat{x}^l=\hat{x}^{l-1}+\Delta t\left[\beta_0 f\left(\hat{x}^l\right)+\beta_1 f\left(\hat{x}^{l-1}\right)\right]$$



Taylor Series Expansion

$$v((l+1)\Delta t)=v(l\Delta t)+\Delta t\frac{dv}{dt}(l\Delta t)+\frac{\Delta t^2}{2}\frac{d^2v}{dt^2}(l\Delta t)+\cdots$$

$$v((l+1)\Delta t)=v(l\Delta t)+\Delta t\frac{dv}{dt}(l\Delta t)+\frac{\Delta t^2}{2}\frac{d^2v}{dt^2}(\tau)$$

$$\tau\in\left[l\Delta t,(l+1)\Delta t\right]$$

- Consistency for Forward Euler

  Forward-Euler definition

$$\hat{v}^{l+1}-\hat{v}^l-\Delta t\lambda\hat{v}^l=0$$

$\tau\in[l\Delta t,(l+1)\Delta t]$

Substituting the exact $v(t)$ and expanding

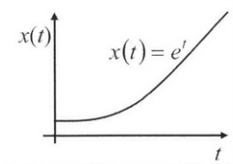
$$v((l+1)\Delta t)-v(l\Delta t)-\Delta t\frac{dv(l\Delta t)}{dt}=\underbrace{\frac{(\Delta t)^2}{2}\frac{d^2v(\tau)}{dt^2}}_{e^l}$$

$$\frac{d}{dt}v=\lambda v$$

where e^l is the LTE and is bounded by

$$\left|e^l\right|\le C(\Delta t)^2,\ \text{where}\ C=0.5\max_{\tau\in[0,T]}\left|\frac{d^2v(\tau)}{dt^2}\right|$$

$$\dot{x}=x \qquad x(t)=e^t$$
$$x(0)=1 \qquad \dot{x}(t)=e^t$$



$$E^0 = 0$$

$$E^1 = e^1$$

$$E^2 = (1+\Delta t\lambda)E^1 + e^2 = (1+\Delta t\lambda)e^1 + e^2$$

$$E^3 = (1+\Delta t\lambda)^2 e^1 + (1+\Delta t\lambda)e^2 + e^3$$

$$E^l \underset{\text{roughly}}{\approx} \sum_{i=1}^{l} e^i$$

If $|E^l|$ roughly $\sum |e^i| \le \sum C(\Delta t)^2$

$$\left|E^{l_{\max}/\Delta t}\right| \approx \underbrace{\frac{T}{\Delta t}}_{\text{number of errors}} \cdot \underbrace{C(\Delta t)^2}_{\text{size of errors shrinking faster}}$$

$$\frac{1}{2}\max_{\tau\in[0,T]}\frac{d^2v(\tau)}{d\tau^2}$$

❖ Global Error Equation

Forward-Euler definition

$$\hat{v}^{l+1} = \hat{v}^l + \Delta t\lambda\hat{v}^l$$

Using the LTE definition

$$v((l+1)\Delta t) = v(l\Delta t) + \Delta t\lambda v(l\Delta t) + e^l$$

Subtracting yields global error equation

$$E^{l+1} = (l+\Delta t\lambda)E^l + e^l$$

Using magnitudes and the bound on e^l

$$|E^{l+1}| = |l+\Delta t\lambda||E^l| + |e^l| \le (1+\Delta t|\lambda|)|E^l| + C(\Delta t)^2$$

❖ A helpful bound on difference equations

A lemma bounding difference equation solutions

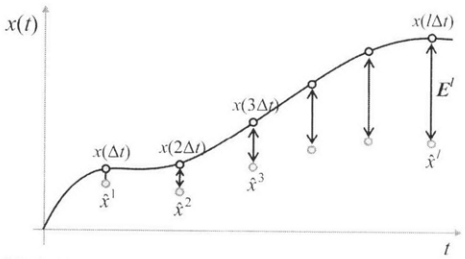
If $|u^{l+1}| \le (1+\varepsilon)|u^l| + b,\ u^0 = 0,\ \varepsilon > 0$

Then $|u^l| \le \dfrac{e^{\varepsilon l}}{\varepsilon}|b|$

$$u^1 \le (1+\varepsilon)(u^0) + b = b$$

$$u^2 \le (1+\varepsilon)b + b$$

$$u^3 \le (1+\varepsilon)^2 b + (1+\varepsilon)b + b$$



One-Step Methods

❖ Hints for the Proof

To prove, first write u^l as a power series and sum

$$|u^l| \le \sum_{j=0}^{l-1}(1+\varepsilon)^j|b| = \frac{1-(1+\varepsilon)^l}{1-(1+\varepsilon)}|b|$$

To finish, note $(1+\varepsilon) \le e^{\varepsilon} \Rightarrow (1+\varepsilon)^l \le e^{\varepsilon l}$

$$|u^l| \le \frac{1-(1+\varepsilon)^l}{1-(1+\varepsilon)}|b| = \frac{(1+\varepsilon)^l - 1}{\varepsilon}|b| \le \frac{e^{\varepsilon l}}{\varepsilon}|b|$$

$$|u^0| = 0 \quad b = 1 \quad \varepsilon = 0.1$$

$$|u^1| \le b = 1$$

$$|u^2| \le 1.1\cdot 1 + 1$$

$$|u^7| \le 1.1\cdot|u^6| + 1 = 1.1\cdot(1.1\cdot|u^5|) + 1 = 1.1\cdot 1.1\cdot(1.1\cdot|u^4|)$$

$$|u^l| \le \underbrace{(1+\varepsilon)^l b}_{\text{contribution from first step}} + (1+\varepsilon)^{l-1}b + \ldots + b$$

❖ Back to Forward Euler Convergence Analysis

Applying the lemma and canceling terms

$$|E^{l+1}| \le \left(1 + \underbrace{\Delta t|\lambda|}_{\varepsilon}\right)|E^l| + \underbrace{C(\Delta t)^2}_{b} \le \frac{e^{l\Delta t|\lambda|}}{\Delta t|\lambda|}C(\Delta t)^2$$

Finally noting that $l\Delta t \le T$,

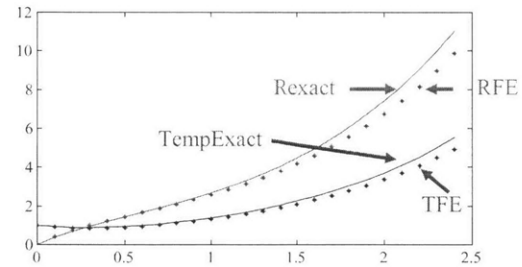
$$\max_{l\in[0,L]}|E^l| \le e^{\lambda T}\frac{C}{|\lambda|}\Delta t$$

Convergence Analysis

- Observations about the forward-Euler analysis

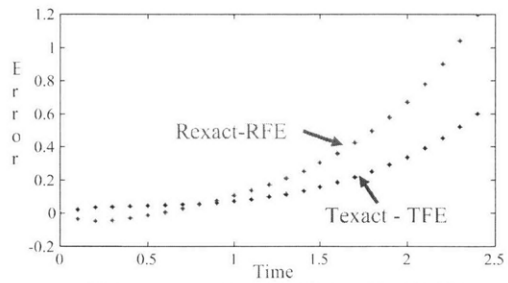
$$\max_{t\in[0,T]} |E^t| \le e^{|\lambda| T} \frac{C}{|\lambda|} \Delta t$$

  - Forward-Euler is order 1 convergent
  - Bound grows exponentially with time interval
  - C related to exact solution's second derivative
- Exact and Forward-Euler (FE) Plots for Unstable Reactions



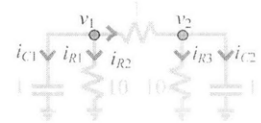
Forward-Euler Errors appear to grow with time

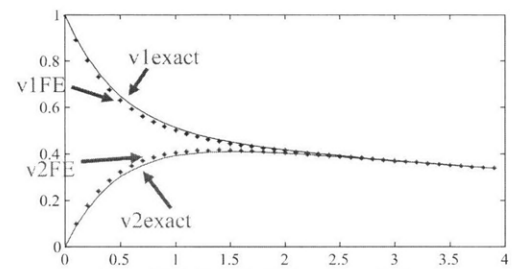
- Forward-Euler Errors for Solving Reaction Equation



Note error grows exponentially with time, as bound predicts

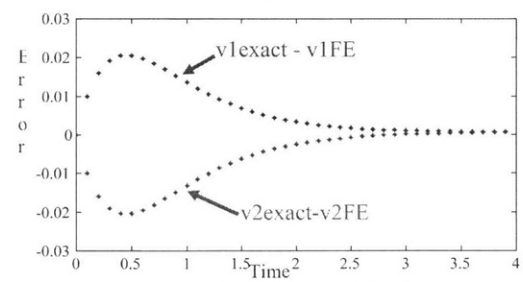
- Exact and forward-Euler (FE) plots for circuit.





Forward-Euler Errors don't always grow with time

- Forward-Euler (FE) errors for solving circuit equation.

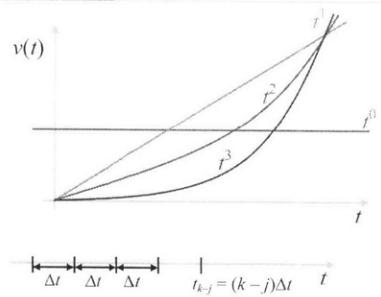


Error does not always grow exponentially with time!

Bound is conservative

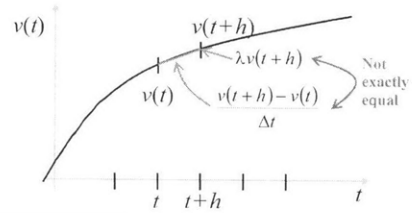
MINIMIZING LTE (CONSISTENCY)

Exactness Constraints



$$\dot{v}(t) = \lambda v(t) \qquad = \Delta t \lambda v(t+h) + e^l(t+h)$$

$$v(t+h) - v(t) \cong \Delta t \lambda v(t+h)$$



$$v(t_l) - v(t_{l-1}) - \Delta t \lambda v(t_l) = e^l$$

$$\hat{v}^l - \hat{v}^{l-1} - \Delta t \lambda \hat{v}^l = 0$$

Local Truncation Error : $\sum_{j=0}^{k} \alpha_j v(t_{l-j}) - \Delta t \sum_{j=0}^{k} \beta_j \frac{d}{dt} v(t_{l-j}) = \boxed{e^l}$

Local Truncation Error (LTE)

If $v(t) = t^p \Rightarrow \frac{d}{dt} v(t) = p t^{p-1}$

$$\sum_{j=0}^{k} \alpha_j \underbrace{((k-j)\Delta t)^p}_{v(t_{k-j})} - \Delta t \sum_{j=o}^{k} \beta_j \underbrace{p((k-j)\Delta t)^{p-1}}_{\frac{d}{dt} v(t_{k-j})} = e^k$$

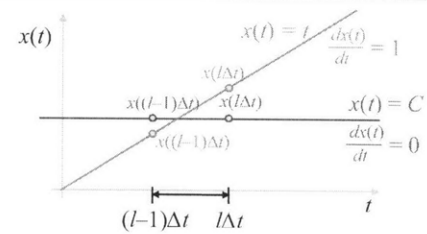
$$\Rightarrow (\Delta t)^p \left[\sum_{j=0}^{k} \alpha_j (l-j)^p - \sum_{j=o}^{k} \beta_j p (l-j)^{p-1} \right] = e^k$$

If $\sum_{j=0}^{k} \alpha_j (k-j)^p - \sum_{j=o}^{k} \beta_j p (k-j)^{p-1} = 0$ then $e^k = 0$ for $v(t) = t^p$

As any smooth $v(t)$ has a locally accurate Taylor series in t:

If $\sum_{j=0}^{k} \alpha_j (k-j)^p - \sum_{j=o}^{k} \beta_j p (k-j)^{p-1} = 0$ for all $p \le p_0$

Then $\sum_{j=0}^{k} \alpha_j v(t_{l-j}) - \sum_{j=o}^{k} \beta_j \frac{d}{dt}(v_{l-j}) = e^l = C(\Delta t)^{p_0+1}$



$$\hat{x}^l - \hat{x}^{l-1} + \Delta t \underbrace{f\left(\hat{x}^l\right)}_{\dot{\hat{x}}^l} = 0$$

$x(t) = 1$ $\qquad\qquad$ $x(t) = t$

$1 - 1 + 0 = 0$ $\qquad\qquad$ $l\Delta t - (l-1)\Delta t + \Delta t \cdot 1 = 0$

Error is zero for both constant and linear functions

- Example – Exactness Constraint $k = 2$

Exactness Constraints: $\sum_{j=0}^{k} \alpha_j (k-j)^p - \sum_{j=0}^{k} \beta_j p (k-j)^{p-1} = 0$

For $k = 2$, yields a 5×6 system of equations for Coefficients

$$\begin{array}{c} p=0 \\ p=1 \\ p=2 \\ p=3 \\ p=4 \end{array} \begin{bmatrix} 1 & 1 & 1 & 0 & 0 & 0 \\ 2 & 1 & 0 & -1 & -1 & -1 \\ 4 & 1 & 0 & -4 & -2 & 0 \\ 8 & 1 & 0 & -12 & -3 & 0 \\ 16 & 1 & 0 & -32 & -4 & 0 \end{bmatrix} \begin{bmatrix} \alpha_0 \\ \alpha_1 \\ \alpha_2 \\ \beta_0 \\ \beta_1 \\ \beta_2 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

Note $\sum \alpha_i = 0$ Always

Forward-Euler $\alpha_0 = 1, \alpha_1 = -1, \alpha_2 = 0, \beta_0 = 0, \beta_1 = 1, \beta_2 = 0,$
FE satisfies $p = 0$ and $p = 1$ but not $p = 2 \Rightarrow$ LTE $= C(\Delta t)^2$

Backward-Euler $\alpha_0 = 1, \alpha_1 = -1, \alpha_2 = 0, \beta_0 = 1, \beta_1 = 0, \beta_2 = 0,$
BE satisfies $p = 0$ and $p = 1$ but not $p = 2 \Rightarrow$ LTE $= C(\Delta t)^2$

Trapezoidal Rule $\alpha_0 = 1, \alpha_1 = -1, \alpha_2 = 0, \beta_0 = 0.5, \beta_1 = 0.5, \beta_2 = 0,$
Trapezoidal satisfies $p = 0, 1,$ or 2 but not $p = 3 \Rightarrow$ LTE $= C(\Delta t)^3$

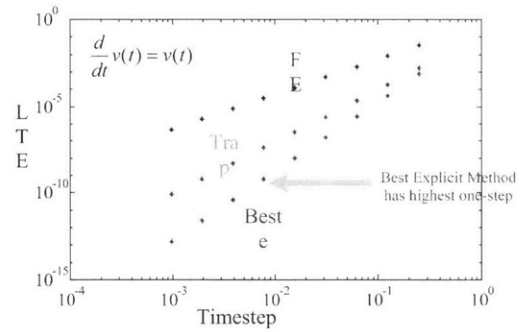
First introduce a normalization, for example $\alpha_0 = 1$

Solve for the 2-step method with lowest LTE
$\alpha_0 = 1, \alpha_1 = 0, \alpha_2 = -1, \beta_0 = 1/3, \beta_1 = 4/3, \beta_2 = 1/3$
Satisfies all five exactness constraints $\Rightarrow$ LTE $= C(\Delta t)^5$

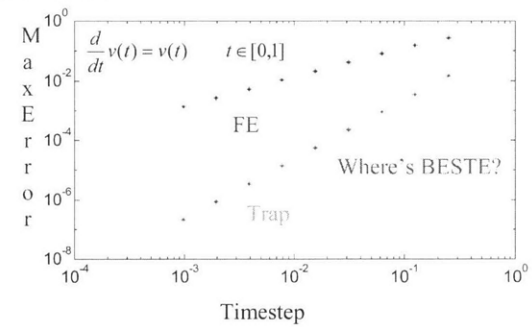
Solve for the 2-step explicit method with lowest LTE
$\alpha_0 = 1, \alpha_1 = 4, \alpha_2 = -5, \beta_0 = 0, \beta_1 = 4, \beta_2 = 2$
Satisfies all five exactness constraints $\Rightarrow$ LTE $= C(\Delta t)^4$

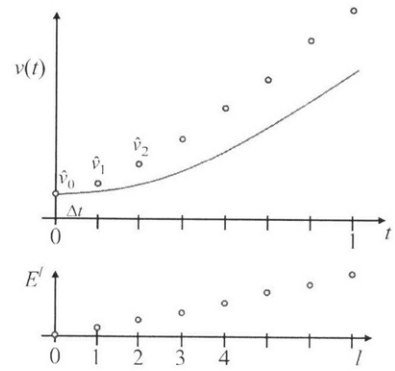
Plots for the FE, Trap, and "Best" Explicit (BESTE)

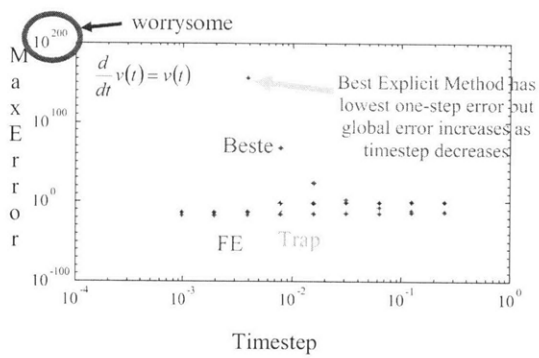
- Local Truncation Error



- Global Error







For a "good" method

$$E^l \cong k\sum_{i=1}^{l} e^i \qquad l \propto \frac{T}{\Delta t}$$

FE LTE Δt^2

$$|E^l| \le kl \max_i |e^i|$$

MINIMIZING THE ACCUMULATION OF ERRORS (STABILITY)

Difference Equation

Why did the "best" 2-step explicit method fail to converge?

Multistep Method Difference Equation

$$(\alpha_0 - \lambda\Delta t\beta_0)E^l + (\alpha_1 - \lambda\Delta t\beta_1)E^{l-1} + \cdots + (\alpha_k - \lambda\Delta t\beta_k)E^{l-k} = e^l$$

$v(l\Delta t) - \hat{v}^l$ Global Error LTE

We made the LTE so small, how come the Global error is so large?

$\alpha_0 = 1 \quad \beta_0 = 0$ (Explicit)

$$E^l = -(\alpha_1 - \lambda\Delta t\beta_1)E^{l-1} - (\alpha_2 - \lambda\Delta t\beta_2)E^{l-2} - \cdots - (\alpha_k - \lambda\Delta t\beta_k)E^{l-k} + e^l$$

$$\begin{bmatrix} E^l \\ E^{l-1} \\ E^{l-2} \\ \vdots \\ E^{l-k+1} \end{bmatrix} = \begin{bmatrix} -(\alpha_1 + \lambda\beta_1) & -(\alpha_2 + \lambda\beta_2) & -(\alpha_3 + \lambda\beta_3) & \cdots & -(\alpha_k + \lambda\beta_k) \\ 1 & & & & \\ & 1 & & & \\ & & \ddots & & \\ & & & 1 & 0 \end{bmatrix} \begin{bmatrix} E^{l-1} \\ E^{l-2} \\ \vdots \\ E^{l-k+1} \\ E^{l-k} \end{bmatrix} + \begin{bmatrix} e^l \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

Stability Definition

Multistep Method Difference Equation

$$(\alpha_0 - \lambda\Delta t\beta_0)E^l + (\alpha_1 - \lambda\Delta t\beta_1)E^{l-1} + \cdots + (\alpha_k - \lambda\Delta t\beta_k)E^{l-k} = e^l$$

Definition: A multistep method is stable if as $\Delta t \to 0$

$$\max_{l \in \left[0, \frac{T}{\Delta t}\right]} |E^l| \le \underbrace{C(T)}_{\substack{\text{interval} \\ \text{dependent}}} \frac{T}{\Delta t} \max_{l \in \left[0, \frac{T}{\Delta t}\right]} |e^l|$$

Stability means: Global error is bounded by a constant times the sum of the L.T.E.'s

Solving Difference Equations

- Convolution

  Consider a general k^{th} order difference equation

  $$a_0 x^l + a_1 x^{l-1} + \cdots + a_k x^{l-k} = u^l$$

  Which must have k initial conditions

  $$x^0 = x_0,\ x^{-1} = x_1,\ \cdots,\ x^{-k} = x_k$$

  As is clear when the equation is in update form

  $$x^1 = -\frac{1}{a_0}\left(a_1 x^0 + \cdots + a_k x^{-k+1} - u^1\right)$$

  It can be shown that the solution of a difference equation is simply a convolution sum:

  x can be related to u by $x^l = \sum_{j=0}^{l} h^{l-j} u^j$

- Calculating h

  If $a_0 z^k + a_1 z^{k-1} + \cdots + a_k = 0$ has distinct roots $\zeta_1, \zeta_2, \ldots, \zeta_k$

  Then $x^l = \sum_{j=0}^{l} h^{l-j} u^j$ where $h^l = \sum_{j=1}^{k} \gamma_j (\zeta_j)^l$

  To understand how h is derived, first a simple case

  Suppose $x^l = \varsigma x^{l-1} + u^l$ and $x^{-1} = 0$

  $$x^l = \varsigma x^0 + u^1 = u^1, \quad x^2 = \varsigma x^1 + u^2 = \varsigma u^1 + u^2$$

  $$x^l = \sum_{j=0}^{l} \varsigma^{l-j} u^j$$

  time index; exponent; time index

- Three Important Observations

  If $|\varsigma_i| < 1$ for all i, then $|x^l| \le C \max_j |u^j|$ where C does not depend on l

  If $|\varsigma_i| > 1$ for any i, then there exists a bounded u^j such that $|x^l| \to \infty$

  If $|\varsigma_i| \le 1$ for all i, and if $|\varsigma_i| = 1$, ς_i is distinct then $|x^l| \le Cl \max_j |u^j|$

$$x^l = \varsigma x^{l-1} + u^l \qquad x^0 = 0$$

$$x^1 = \underbrace{\varsigma x^0}_{0} + u^1$$

$$x^2 = \varsigma x^1 + u^2 = \varsigma u^1 + u^2$$

$$x^l = \varsigma^{l-1} u^1 + \varsigma^{l-2} u^2 + \varsigma^{l-3} u^3 + \ldots u^l$$

- Convolution Sum
  - When roots are not distinct

$$h^l = \sum_{q=1}^{Q} \sum_{m=0}^{M_q - 1} \gamma_{q,m} (l)^m (\varsigma_q)^l \qquad x^l = \sum_{j=0}^{l} h^{l-j} u^j$$

Root multiplicity

Roots of $a_0 z^k + a_1 z^{k-1} + \cdots + a_k = 0$

  - Bounding Terms

$$x^l = \sum_{q=1}^{Q} \sum_{m=0}^{M_q - 1} \underbrace{\left(\sum_{j=0}^{l} \gamma_{q,m} (l-j)^m (\varsigma_q)^{l-j} u^j \right)}_{R_{q,m}}$$

If $|\varsigma_i| < 1$, then $|R_{q,m}| \le C \max_j |u^j|$ — Independent of l

If $|\varsigma_q| < (1+\varepsilon)$, then $|R_{q,0}| \le C \frac{e^{\varepsilon l}}{\varepsilon} \max_j |u^j|$

Lemma: bounds **distinct** roots

Stability Theorem

Theorem: A multistep method is stable if and only if
Roots of $\alpha_0 z^k + \alpha_1 z^{k-1} + \ldots + \alpha_k = 0$ either:

1. Have magnitude less than one
2. Have magnitude equal to one and are distinct

Note: from the previous slides it is easy to see that a multistep method is stable if the same conditions hold for:

$$(\alpha_0 - \lambda\Delta t\beta_0) z^k + (\alpha_1 - \lambda\Delta t\beta_1) z^{k-1} + \cdots + (\alpha_k - \lambda\Delta t\beta_k) = 0$$

But the theorem says: forget about the β's…

- "Proof"

  Given the Multstep Method Difference Equation

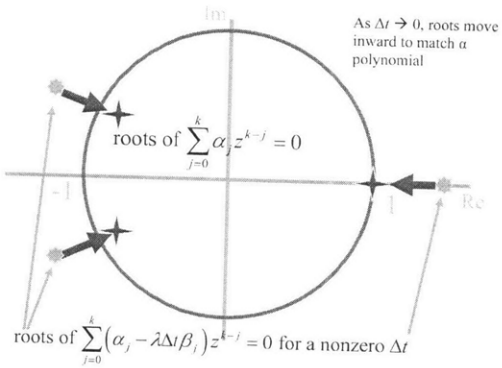
  $$(\alpha_0 - \lambda\Delta t\beta_0)E^l + (\alpha_1 - \lambda\Delta t\beta_1)E^{l-1} + \cdots + (\alpha_k - \lambda\Delta t\beta_k)E^{l-k} = e^l$$

  If, as $\Delta t \to 0$, roots of $(\alpha_0 - \lambda\Delta t\beta_0)z^l + \cdots + (\alpha_k - \lambda\Delta t\beta_k) = 0$
  - less than one in magnitude or
  - are distinct and bounded by $1 + \kappa\Delta t$, $\kappa > 0$

  Then from the aside on difference equations

  $$\max_{l \in \left[0, \frac{T}{\Delta t}\right]} \left|E^l\right| \le C\frac{e^{\kappa l/\Delta t}}{\Delta t} \max_{l \in \left[0, \frac{T}{\Delta t}\right]} \left|e^l\right| \le \underbrace{\frac{Ce^{\kappa T}}{T}}_{C(T)} \frac{T}{\Delta t} \max_{l \in \left[0, \frac{T}{\Delta t}\right]} \left|e^l\right|$$

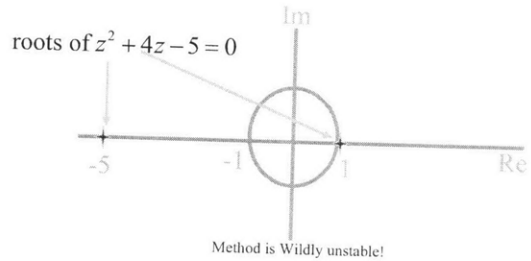
- Picture



BESTE Method

Best explicit 2-step method

$\alpha_0 = 1$, $\alpha_1 = 4$, $\alpha_2 = -5$, $\beta_0 = 0$, $\beta_1 = 4$, $\beta_2 = 2$



Method is Wildly unstable!

Dahlquist's First Stability Barrier

For a stable, explicit k-steps method, the maximum degree of a polynomial t^p that can be integrated exactly is less than or equal to k: $p_0 \le k$ (note there are $2k$ coefficients).

For a stable, implicit k-steps method, (with $2k + 1$ coeff)

$p_0 \le k + 2$ if k is even

$p_0 \le k + 1$ if k is odd

STABLE:	EXPLICIT	IMPLICIT
$k = 1$ step	$p_0 \le 1$ (FE)	$p_0 \le 2$ (TR)
$k = 2$ steps	$p_0 \le 2$	$p_0 \le 4$
$k = 3$ steps	$p_0 \le 3$	$p_0 \le 4$

Conditions for Convergence, Stability, and Consistency

1. Local Condition: One step errors are small (consistency)

   Exactness Constraints up to p_0 (p_0 must be > 0)

   $$\Rightarrow \max_{l \in \left[0, \frac{T}{\Delta t}\right]} \left\|e^l\right\| \le C_1(\Delta t)^{p_0+1} \quad \text{for } \Delta t < \Delta t_0$$

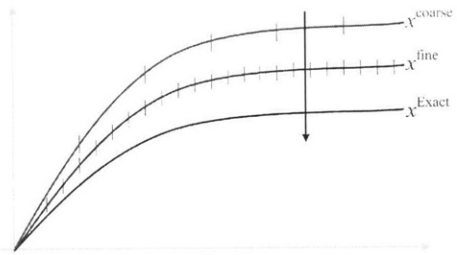
2. Global Condition: One step errors grow slowly (stability)

INTRODUCTION TO NUMERICAL SIMULATION

LECTURE 16.
Multistep Methods II: Small and Large Time-Steps Stability

TODAY'S OUTLINE:

- SMALL Timestep Issues for Multistep Methods
  - Stability
    - Difference Equations
  - Stability & Consistency implies Convergence
- LARGE Timestep Issues
  - Absolute Stability for two time-scale examples.
  - Oscillators



STABILITY FOR SMALL TIMESTEPS

Difference Equation

Why did the "best" two-step explicit method fail to converge?
Multistep Method Difference Equation:

$$(\alpha_0 - \lambda\Delta t\beta_0)E^l + (\alpha_1 - \lambda\Delta t\beta_1)E^{l-1} + \cdots + (\alpha_k - \lambda\Delta t\beta_k)E^{l-k} = e^l$$

$v(l\Delta t) - \hat{v}^l$ Global Error

LTE

We made the LTE so small, how come the Global error is so large?

Stability Definition

Multistep Method Difference Equation

$$(\alpha_0 - \lambda\Delta t\beta_0)E^l + (\alpha_1 - \lambda\Delta t\beta_1)E^{l-1} + \cdots + (\alpha_k - \lambda\Delta t\beta_k)E^{l-k} = e^l$$

Definition: A multistep method is stable if as $\Delta t \to 0$

$$\max_{l\in\left[0,\frac{T}{\Delta t}\right]} \left|E^l\right| \le \underbrace{C(T)}_{\substack{\text{interval}\\ \text{dependent}}} \frac{T}{\Delta t} \max_{l\in\left[0,\frac{T}{\Delta t}\right]} \left|e^l\right|$$

Stability means: Global Error is bounded by a constant times the sum of the LTE's

Solving Difference Equations

- Convolution

Consider a general k^{th} order difference equation

$$a_0 x^l + a_1 x^{l-1} + \cdots + a_k x^{l-k} = u^l$$

Which must have k initial conditions

$$x^0 = x_0, \quad x^{-1} = x_1, \quad \cdots \quad x^{-k} = x_k$$

as is clear when the equation is in update form

$$x^l = -\frac{1}{a_0}\left(a_1 x^0 + \cdots + a_k x^{-k+1} - u^l\right)$$

it can be shown that the solution of a difference equation is simply a convolution sum:

x can be related to u by $x^l = \sum_{j=0}^{l} h^{l-j} u^j$

BE $\dot{v}(t) = \lambda v$

$$\hat{v}^l - \hat{v}^{l-1} - \Delta t\lambda\hat{v}^l = 0$$

$$v(l\Delta t) - v((l-1)\Delta t) - \Delta t\lambda v(l\Delta t) = e^l$$

$$(1 - \Delta t\,\lambda)\ E^l - E^{l-1} = e^l$$

$$E^1 = \frac{1}{1-\Delta t\lambda}E^0 + \frac{1}{1-\Delta t\lambda}e^1$$

$$E^2 = \frac{1}{1-\Delta t\lambda}E^1 + \frac{1}{1-\Delta t\lambda}e^2 = \left(\frac{1}{1-\Delta t\lambda}\right)^2 e^1 + \frac{1}{1-\Delta t\lambda}e^2$$

$$E^l = \left(\frac{1}{1-\Delta t\lambda}\right)^l e^1 + \left(\frac{1}{1-\Delta t\lambda}\right)^{l-1} e^2 + \cdots + \frac{1}{1-\Delta t\lambda}e^l$$

$$E^l = \sum_{j=1}^{l}\left(\frac{1}{1-\Delta t\lambda}\right)^{l-j}\underbrace{\left(\frac{1}{1-\Delta t\lambda}\right)e^j}_{u}$$

- Calculating h

  If $a_0 z^k + a_1 z^{k-1} + \cdots + a_k = 0$ has distinct roots $\varsigma_1, \varsigma_2, \ldots, \varsigma_k$

  Then $x^l = \sum_{j=0}^{l} h^{l-j} u^j$ where $h^l = \sum_{j=1}^{k} \gamma_j (\varsigma_j)^l$

  To understand how h is derived, first a simple case

  Suppose $x^l = \varsigma x^{l-1} + u^l$ and $x^{-1} = 0$

  $x^1 = \varsigma x^0 + u^1 = u^1, \quad x^2 = \varsigma x^1 + u^2 = \varsigma u^1 + u^2$

  $x^l = \sum_{j=0}^{l} \varsigma^{l-j} u^j$ ← time index

  time index; exponent

- Three Important Observations

  If $|\varsigma_i| < 1$ for all i, then $|x^l| \le C \max_j |u^j|$ where C does not depend on l

  If $|\varsigma_i| > 1$ for all i, then there exists a bounded u^j such that $|x^l| \to \infty$

  If $|\varsigma_i| \le 1$ for all i, and i $|\varsigma_i| = 1$, ς_i is distinct then $|x^l| \le Cl \max_j |u^j|$

$$x^l = \lambda x^{l-1} + u^l$$

$$x^{-1} = 0$$

$$x^0 = u^0$$

$$x^1 = \lambda x^0 + u^1 = \lambda u^0 + u^1$$

$$x^2 = \lambda x^1 + u^2 = \lambda^2 u^0 + \lambda u^1 + u^2$$

$$x^l = \sum_{i=0}^{l} \lambda^{l-i} u^i$$

- Convolution Sum
  - When roots are not distinct

$$h^l = \sum_{q=1}^{Q} \sum_{m=0}^{M_q - 1} \gamma_{q,m} (l)^m (\varsigma_q)^l \qquad x^l = \sum_{j=0}^{l} h^{l-j} u^j$$

Root Multiplicity

Roots of $a_0 z^k + a_1 z^{k-1} + \ldots + a_k = 0$

  - Bounding Terms

$$x^l = \sum_{q=1}^{Q} \sum_{m=0}^{M_q - 1} \underbrace{\left[\sum_{j=0}^{l} \gamma_{q,m} (l-j)^m (\varsigma_q)^{l-j} u^j \right]}_{R_{q,m}}$$

If $|\varsigma_i| < 1$, then $|R_{q,m}| \le C \max_j |u^j|$

Independent of l

If $|\varsigma_q| < (1+\varepsilon)$, then $|R_{q,0}| \le C \frac{e^{\varepsilon l}}{\varepsilon} \max_j |u^j|$

Lemma: bounds **distinct** roots

Stability Theorem

Theorem: A multistep method is stable if and only if

Roots of $\alpha_0 z^k + \alpha_1 z^{k-1} + \cdots + \alpha_k = 0$ either:

- Have magnitude less than one
- Have magnitude equal to one and are distinct

Note: from the previous slides it is easy to see that a multistep method is stable if the same conditions hold for:

$(\alpha_0 - \lambda\Delta t\beta_0) z^k + (\alpha_1 - \lambda\Delta t\beta_1) z^{k-1} + \cdots + (\alpha_k - \lambda\Delta t\beta_k) = 0$

But the theorem says: forget about the betas…

- "Proof"

  Given the Multistep Method Difference Equation

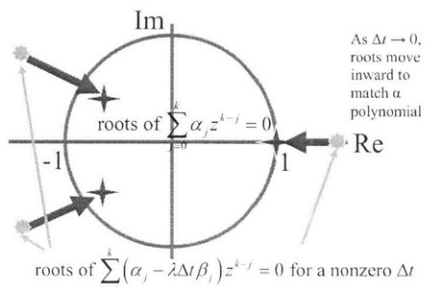
  $(\alpha_0 - \lambda\Delta t\beta_0) E^l + (\alpha_1 - \lambda\Delta t\beta_1) E^{l-1} + \cdots + (\alpha_k - \lambda\Delta t\beta_k) E^{l-k} = e^l$

  If, as $\Delta t \to 0$, roots of $(\alpha_0 - \lambda\Delta t\beta_0) z^l + \ldots + (\alpha_k - \lambda\Delta t\beta_k) = 0$
  - Less than one in magnitude or
  - Are distinct and bounded by $1 + \kappa\Delta t$, $\kappa > 0$

  Then from the aside on difference equations

$$\max_{l \in \left[0, \frac{T}{\Delta t}\right]} |E^l| \le C \frac{e^{\kappa l \Delta t}}{\Delta t} \max_{l \in \left[0, \frac{T}{\Delta t}\right]} |e^l| \le \frac{C e^{\kappa T}}{\underbrace{\frac{T}{\Delta t}}_{C(T)}} \frac{T}{\Delta t} \max_{l \in \left[0, \frac{T}{\Delta t}\right]} |e^l|$$

❖ Picture



roots of $\sum_{j=0}^{k} \left(\alpha_j - \lambda \Delta t \beta_j\right) z^{k-j} = 0$ for a nonzero Δt

$z^2 - 2z + 1 = 0$ — Roots = 1,1

$z + 1 = 0$ — Root = 1

$$x^{l+1} = 2x^l - x^{l-1} + 1 \qquad x^{l+1} = 2x^l + 1$$

$$x^{-1} = 0 \quad x^0 = 0$$

$x^1 = 1$
$x^2 = 3$
$x^3 = 6$
$x^4 = 10$
$x^5 = 15$
$\vdots$
$x^l = l$

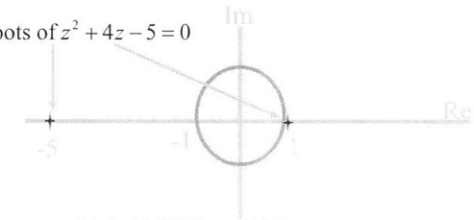
$x^1 = 1$
$x^2 = 2$
$\vdots$
$x^l = l$

BESTE Method

Best explicit 2-step method

$\alpha_0 = 1, \quad \alpha_1 = 4, \quad \alpha_2 = -5, \quad \beta_0 = 0, \quad \beta_1 = 4, \quad \beta_2 = 2$

roots of $z^2 + 4z - 5 = 0$



Method is Wildly unstable!

Dahlquist's First Stability Barrier

For a stable, explicit k-steps method, the maximum degree of a polynomial t^{p_0} tat can be integrated exactly is less than or equal to k: $p_0 \le k$ (note there are $2k$ coefficients).

For a stable, implicit k-steps method, (with $2k$+1 coeff)

$p_0 \le k + 2$ if k is even

$p_0 \le k + 1$ if k is odd

STABLE:	EXPLICIT:	IMPLICIT:
$k = 1$ step	$p_0 \le 1$ (FE)	$p_0 \le 2$ (TR)
$k = 2$ steps	$p_0 \le 2$	$p_0 \le 4$
$k = 3$ steps	$p_0 \le 3$	$p_0 \le 4$

Conditions for Convergence, Stability, and Consistency

❖ Local Condition: One step errors are small (consistency)

Exactness Constraints up to p_0 (p_0 must be > 0)

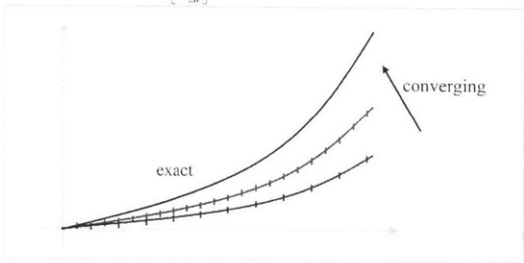
$$\Rightarrow \max_{l \in \left[0, \frac{T}{\Delta t}\right]} \left\| E^l \right\| \le C_1 (\Delta t)^{p_0 + 1} \quad \text{for } \Delta t < \Delta t_0$$

❖ Global Condition: One step errors grow slowly (stability)

roots of $\sum_{j=0}^{k} \alpha_j z^{k-j} = 0$ — Inside the unit circle or on the unit circle and distinct

$$\Rightarrow \max_{l \in \left[0, \frac{T}{\Delta t}\right]} \left\| E^l \right\| \le C_2 \frac{T}{\Delta t} \max_{l \in \left[0, \frac{T}{\Delta t}\right]} \left\| e^l \right\|$$

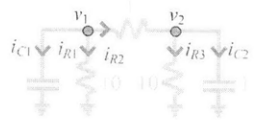
Convergence Result : $\max_{l \in \left[0, \frac{T}{\Delta t}\right]} \left\| E^l \right\| \le CT(\Delta t)^{p_0}$

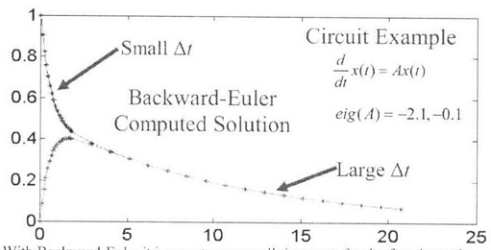


LARGE TIMESTEP ISSUES FOR MULTISTEP METHODS

Absolute Stability for two time-scale examples

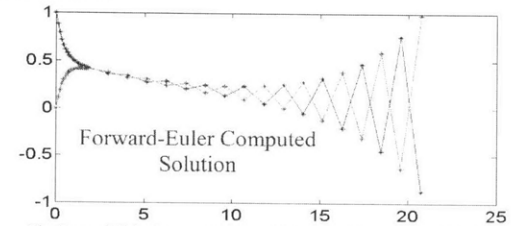
- Backward-Euler





With Backward-Euler it is easy to use small timesteps for the fast dynamics and then switch to large timesteps for the slow decay.

- Forward-Euler



The Forward-Euler is accurate for small timesteps, but goes unstable when the timestep is enlarged.

- FE, BE, and Trap on the scalar ODE problem

Scalar ODE : $\frac{d}{dt}v(t) = \lambda v(t),\ v(0) = v_0 \quad \lambda \in C$

Forward - Euler : $\hat{v}^{l+1} = \hat{v}^l + \Delta t \lambda \hat{v}^l = (1 + \Delta t \lambda)\hat{v}^l$

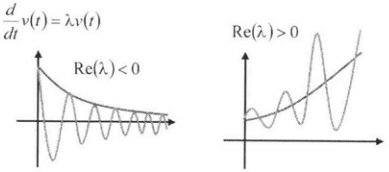
If $|1 + \Delta t \lambda| > 1$ the solution grows even if $\lambda < 0$

Backward - Euler : $\hat{v}^{l+1} = \hat{v}^l + \Delta t \lambda \hat{v}^{l+1} = \frac{1}{(1 - \Delta t \lambda)}\hat{v}^l$

If $\left|\frac{1}{1 + \Delta t \lambda}\right| < 1$ the solution decays even if $\lambda < 0$

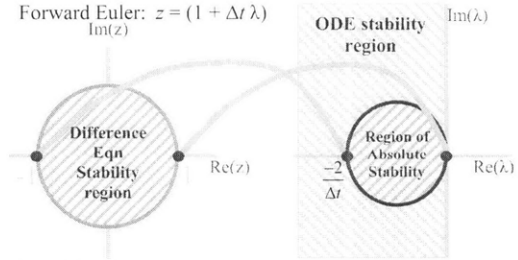
Trapezoidal Rule : $\hat{v}^{l+1} = \hat{v}^l + \frac{1}{2}\Delta t \lambda\left(\hat{v}^{l+1} + \hat{v}^l\right) \Rightarrow \hat{v}^{l+1} = \frac{\left(1 + \frac{1}{2}\Delta t \lambda\right)}{\left(1 - \frac{1}{2}\Delta t \lambda\right)}\hat{v}^l$

$\frac{d}{dt}v(t) = \lambda v(t)$

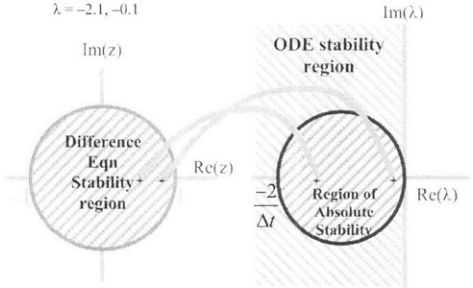


- Forward-Euler large timestep region of absolute stability

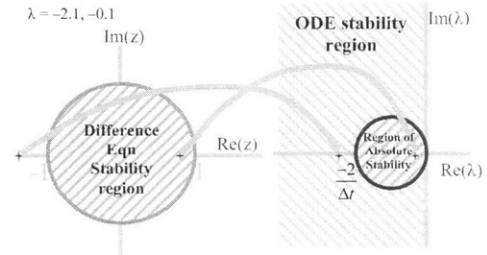
Forward Euler: $z = (1 + \Delta t\, \lambda)$

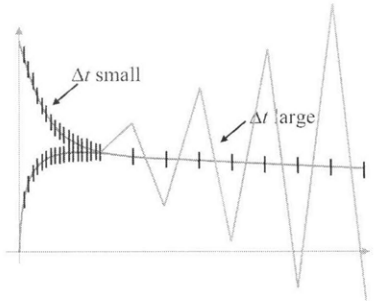


- Circuit Example
  - $\Delta t = 0.1$



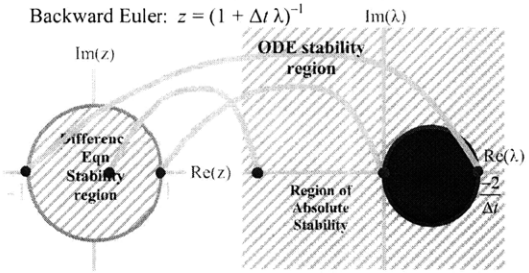
- $\Delta t = 1.0$



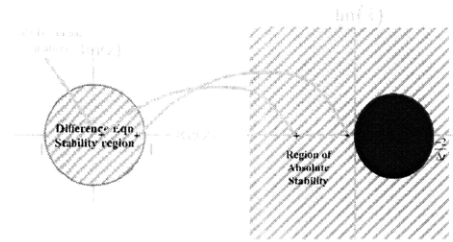


- Backward-Euler large timestep region of absolute stability

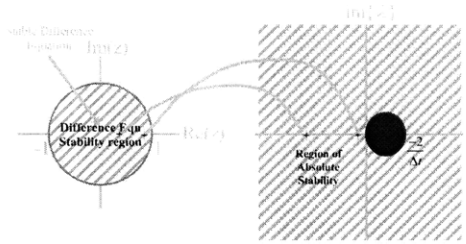
Backward Euler: $z = (1 + \Delta t\,\lambda)^{-1}$



- Circuit Example
  - $\Delta t = 0.1, \lambda = -2.1, -0.1$



- $\Delta t = 1.0, \lambda = -2.1, -0.1$



- Stability Definitions

  Region of Absolute Stability for a Multistep method:

  Values of $\lambda\Delta t$ where roots of $\sum_{j=0}^{k}(\alpha_j - \lambda\Delta t\beta_j)z^{k-j} = 0$ are inside the unit circle.

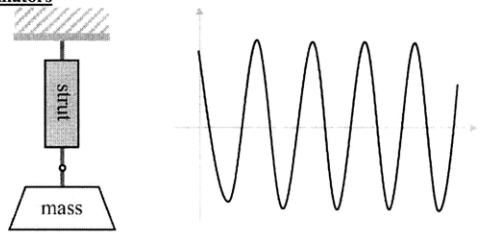
  A-Stable:

  A method is A-stable if its region of absolute stability includes the entire left-half of the complex plane

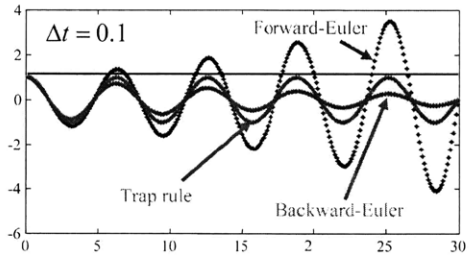
  Dahlquist's Second Stability Barrier:

  There are no A-stable multistep methods of convergence order greater than 2, and the trap rule is the most accurate.

Oscillators

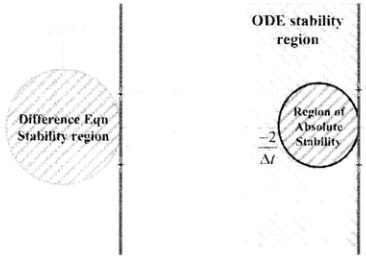


- Numerical Experiments – oscillating strut and mass

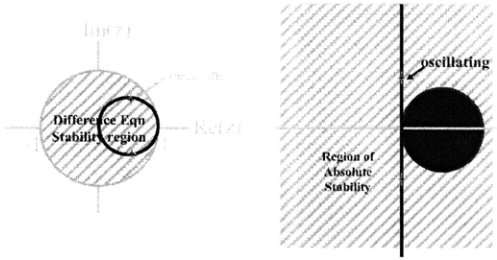


Why does FE result grow, BE result decay and the Trap rule preserve oscillations

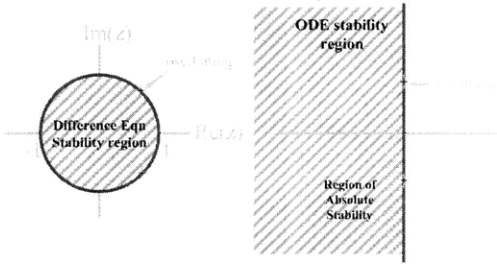
- FE Large Timestep Oscillator Example



- BE Large Timestep Oscillator Example



- Trap Large Timestep Oscillator Example



- Large Timestep Issues
  - Two Time-Constant Stable Problem (Circuit)
    - FE: stability, not accuracy, limited timestep size
    - BE: was A-stable, any timestep could be used.
    - Trap Rule most accurate A-stable m-step method
  - Oscillator Problem
    - Forward-Euler generated an unstable difference equation regardless of timestep size.

- Backward-Euler generated a stable (decaying) difference equation regardless of timestep size.
- Trapezoidal rule mapped the imaginary axis to the unit circle, regardless of timestep size:
  - Decaying ODE are mapped to stable difference equations
  - Unstable ODE are mapped to unstable difference equations
  - Osciallating ODE are mapped to oscillating difference equations

SUMMARY

- Small Timestep Issues for Multistep Methods
  - Local truncation error and exactness
  - Difference equation stability
  - Stability + consistency implies convergence
- Large Timestep Issues
  - Absolute stability for two time-scale examples
  - Oscillators

3.17 ODE - Periodic Steady State Analysis

INTRODUCTION TO NUMERICAL SIMULATION

LECTURE 17.
Methods for Computing Periodic Steady-State

TODAY'S OUTLINE:

- Periodic Steady-State Problems
  - Application examples and simple cases
- Review Full Time Integration Methodss
- Finite-Difference Methods for Periodic Steady-State
- Shooting Methods
  - State Transition Function
  - Sensitivity Matrix
  - Matrix Free Approach
- Spectral Methods (Harmonic Balance)
  - Galerkin Methods
  - Collocation Methods

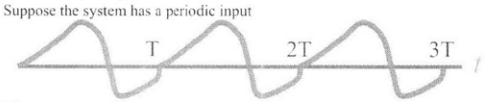
PERIODIC STEADY-STATE PROBLEMS

Basics

- Definition

$$\frac{dx(t)}{dt} = F\left(\underbrace{x(t)}_{state}\right) + \underbrace{u(t)}_{input}$$

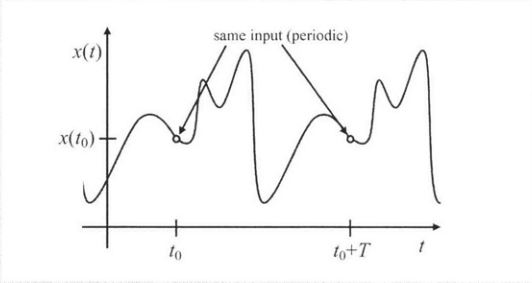
  - Suppose the system has a periodic input



  - Many systems eventually respond periodically
    $x(t+T) = x(t)$ for $t >> 0$
- Interesting Property

$$\dot{x} = F(x(t), u(t)) \qquad x(0) = x_0 \Rightarrow x(t) \text{ is unique}$$

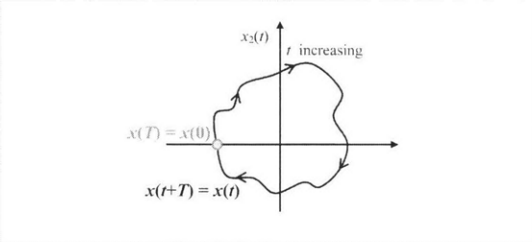
Assuming F is "nice"
smooth enough

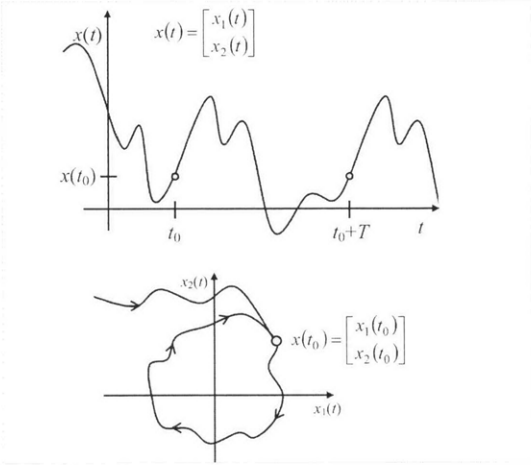


- If x satisfies a differential equation which has a unique solution for any initial condition

$$\frac{dx(t)}{dt} = F(x(t)) + u(t)$$

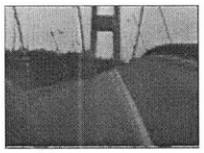
- Then if u is periodic with period T and $x(t_0 + T) = x(t)$ for some t_0
  $\Rightarrow x(t+T) = x(t)$ for all $t > t_0$



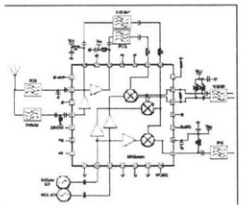


Application Examples

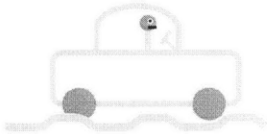
- Swaying Bridge
  - Periodic Input
    - Wind
  - Response
    - Oscillating Platform
  - Desired Info
    - Oscillation Amplitude

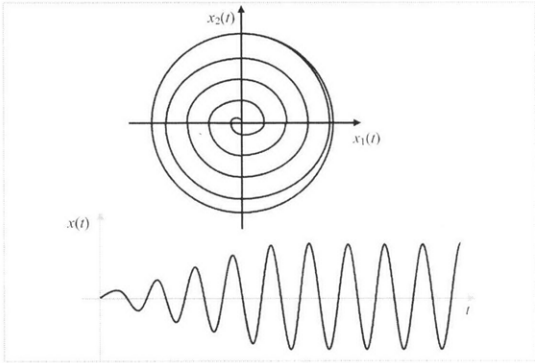


- Communication Integrated Circuit
  Periodic Input
    - Received signal at 900MHz
  - Response
    - Filtered Demodulated Signal
  - Desired Info
    - Distortion

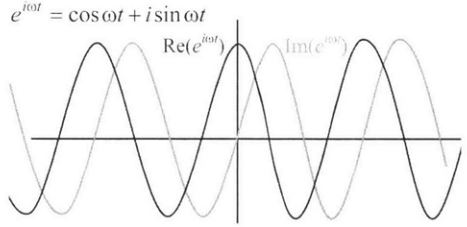


- Automobile Vibration
  - Periodic Input
    - Regularly Spaced Road Bumps
  - Response
    - Car Shakes
  - Desired Info
    - Shake amplitude





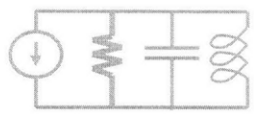
$$e^{i\omega t} = \cos\omega t + i\sin\omega t$$



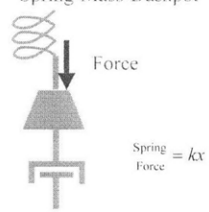
$x(t) = ke^{i\omega t}$ for linear systems

Simple Example [RLC Filter, Spring+Mass+Dashpot]

RLC Circuit



Spring-Mass-Dashpot

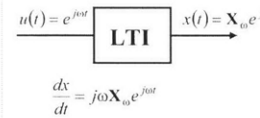


- Both described by second-order ODE

$$\mathbf{M}\frac{d^2x}{dt^2} + \mathbf{D}\frac{dx}{dt} + x = \underbrace{u(t)}_{input}$$

- $u(t) = 0$ lightly damped ($\mathbf{D} << \mathbf{M}$) response

$$x(t) \approx \mathbf{K}e^{-\frac{\mathbf{D}t}{2\mathbf{M}}}\cos\left(\frac{t}{\sqrt{\mathbf{M}}} + \phi\right)$$



$$\frac{dx}{dt} = j\omega\mathbf{X}_\omega e^{j\omega t}$$

$$\mathbf{X}_\omega = ?$$

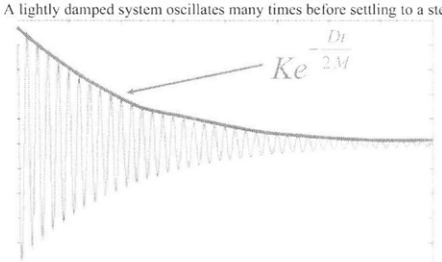
$$j\omega\mathbf{X}_\omega e^{j\omega t} = \mathbf{A}\mathbf{X}_\omega e^{j\omega t} + be^{j\omega t} \quad \forall t$$

$$(j\omega\mathbf{I} - \mathbf{A})\mathbf{X}_\omega = b$$

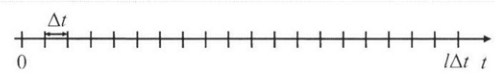
$$\mathbf{X}_\omega = (j\omega\mathbf{I} - \mathbf{A})^{-1}b$$

$$x(t) = (j\omega\mathbf{I} - \mathbf{A})^{-1}be^{j\omega t} = \mathbf{X}_\omega e^{j\omega t}$$

- A lightly damped system oscillates many times before settling to a steady-state



Computing Steady-State



$$\left.\frac{dx}{dt}\right|_{t=l\Delta t} \cong \frac{\overbrace{x(l\Delta t)}^{\hat{x}^l} - \overbrace{x((l-1)\Delta t)}^{\hat{x}^{l-1}}}{\Delta t} \qquad \text{B.E. formula}$$

$$\left.\frac{dx}{dt}\right|_{t=l\Delta t} \cong \underbrace{F\left(\overbrace{x(l\Delta t)}^{\hat{x}^l}\right) + u(l\Delta t)}_{f(x(l\Delta t),u(l\Delta t))} \qquad \text{O.D.E.}$$

❖ Frequency Domain Approach

- Sinusoidally excited linear time-invariant system

$$\frac{dx}{dt} = \mathbf{A}x(t) + bu(t) \qquad \text{input } u(t) = e^{j\omega t}$$

- Steady-state solution simple to determine

$$x(t) = (j\omega\mathbf{I} - \mathbf{A})^{-1} b e^{j\omega t}$$

Not useful for nonlinear or time-varying systems

Phaser Analysis

input $= e^{j\omega t}$

$$\dot{x} = \mathbf{A}x + bu(t)$$

Assume $x(t) = He^{j\omega t}$ — complex constant; Jumping to periodic steady state

$$j\omega He^{j\omega t} = \mathbf{A}He^{j\omega t} + be^{j\omega t}$$

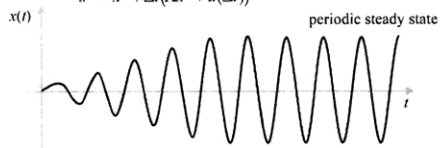
Works well for linear problems

Brut Force

$x(0) = 0 \qquad u(t) = \sin t$

Forward Euler $\quad \hat{x}^1 = \underbrace{x(0)}_{0} + \Delta t(\mathbf{A}x(0) + u(0))$

$$\hat{x}^2 = \hat{x}^1 + \Delta t(\mathbf{A}\hat{x}^1 + u(\Delta t))$$



REVIEW FULL TIME INTEGRATION METHODS

Computing Steady State

$$\frac{\hat{x}^l - \hat{x}^{l-1}}{\Delta t} = F(\hat{x}^l) + u(l\Delta t) \quad \text{Solve for } \hat{x}^l \text{ using Newton}$$

$$\alpha_0 = 1 \quad \alpha_1 = -1 \quad \beta_0 = 1$$

❖ Time-Integrate Until Steady-State Achieved

$$\frac{dx(t)}{dt} = F(x(t)) + u(t) \Rightarrow \hat{x}^l = \hat{x}^{l-1} + \Delta t\left[F(\hat{x}^l) + u(l\Delta t)\right]$$

❖ Need many timepoints for lightly damped case!

Solve with Backward-Euler

❖ Nonlinear System

$$\frac{dx(t)}{dt} = F\left(\underbrace{x(t)}_{state}\right) + \underbrace{u(t)}_{input} \qquad \underbrace{x(0) = x_0}_{\text{Initial Condition}}$$

❖ Backward Euler Equation for timestep l

$$\hat{x}^l - \hat{x}^{l-1} = \Delta t\left[F(\hat{x}^l) + u(l\Delta t)\right]$$

How do we solve the backward-Euler Equation?

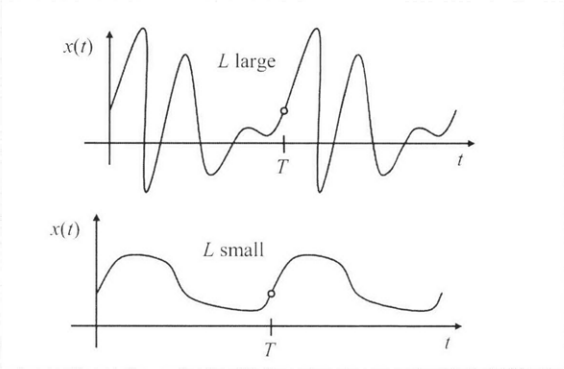
$$\hat{x}^l - \Delta t\left[F(\hat{x}^l) + \underbrace{u(l\Delta t)}_{\text{known}}\right] - \underbrace{\hat{x}^{l-1}}_{\text{known}} = 0$$

unknown

$$H(\hat{x}^l) = \hat{x}^l - \Delta t\left[F(\hat{x}^l) + u(l\Delta t)\right] - \hat{x}^{l-1}$$

Newton Iteration

$$\underbrace{\left[I - \Delta t\frac{\partial F(\hat{x}^{l,k})}{\partial x}\right]}_{J(\hat{x}^{l,k})}\left[\hat{x}^{l,k} - \hat{x}^{l,k+1}\right] = -H(\hat{x}^{l,k})$$



$$\left.\begin{aligned}\hat{x}^1 &= x(0)+\Delta t\left[F(x(0))+u(0)\right]\\ \hat{x}^2 &= \hat{x}^1+\Delta t\left[F\left(\hat{x}^1\right)+u(\Delta t)\right]\\ &\vdots\end{aligned}\right\} F.E.$$

$$\hat{x}^l-\hat{x}^{l-1}-\Delta t\left[F\left(\hat{x}^l\right)+u(l\Delta t)\right]=0$$

$$\Rightarrow \quad \underbrace{\hat{x}^l-\Delta t F\left(\hat{x}^l\right)-\underbrace{\hat{x}^{l-1}-\Delta t\cdot u(l\Delta t)}_{known}=0}$$

$$\left(\mathbf{I}-\Delta t\frac{\partial F}{\partial x}\right)\hat{x}^{l,j+1}-\hat{x}^{l,j}=-\left(\hat{x}^{l,j}-\Delta t F\left(\hat{x}^{l,j}\right)-\hat{x}^{l,j-1}-\Delta t u(l\Delta t)\right)$$

$$\mathbf{J}_H\Delta x=-\mathbf{H}$$

Implicit Methods

- Backward-Euler Example

Forward – Euler	Backward – Euler
$x(t_1)\approx\hat{x}^1=x(0)+\Delta t\cdot f(x(0),u(0))$	$x(t_1)\approx\hat{x}^1=x(0)+\Delta t\cdot f\left(\hat{x}^1,u(t_1)\right)$
$x(t_2)\approx\hat{x}^2=\hat{x}^1+\Delta t\cdot f\left(\hat{x}^1,u(t_1)\right)$	$x(t_2)\approx\hat{x}^2=\hat{x}^1+\Delta t\cdot f\left(\hat{x}^2,u(t_2)\right)$
$\vdots$	$\vdots$

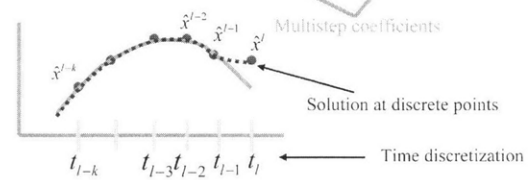
Forward – Euler	Backward – Euler
$x(t_L)\approx\hat{x}^L=\hat{x}^{L-1}+\Delta t\cdot f\left(\hat{x}^{L-1},u(t_{L-1})\right)$	$x(t_L)\approx\hat{x}^L=\hat{x}^{L-1}+\Delta t\cdot f\left(\hat{x}^L,u(t_L)\right)$
Requires just function evaluations	Nonlinear equation solution at each step

Stepwise Nonlinear equation solution needed whenever $\beta_0\neq 0$

- Multi-Step Methods

Nonlinear Differential Equation : $\frac{d}{dt}x(t)=f(x(t),u(t))$

k - Step Multistep Approach : $\sum_{j=0}^{k}\alpha_j\hat{x}^{l-j}=\Delta t\sum_{j=0}^{k}\beta_j f\left(\hat{x}^{l-j},u(t_{l-j})\right)$



- Solution with Newton

$$\underbrace{\left[\quad\right]}_{Jacobian}\Delta\hat{x}^{l,j+1}=-\left[\quad\right]$$

Rewrite the multistep equation

$$\alpha_0\hat{x}^l-\Delta t\beta_0 f\left(\hat{x}^l,u(t_l)\right)+\underbrace{\sum_{i=1}^{k}\alpha_i\hat{x}^{l-i}-\Delta t\sum_{i=1}^{k}\beta_i f\left(\hat{x}^{l-i},u(t_{l-i})\right)}_{b}=0$$

Independent of $\hat{x}^l$

Solve with Newton

$$\underbrace{\left[\alpha_0 I-\Delta t\beta_0\frac{\partial f\left(\hat{x}^{l,j},u(t_l)\right)}{\partial x}\right]}_{Jacobian}\left(\hat{x}^{l,j+1}-\hat{x}^{l,j}\right)=\underbrace{-\left[\alpha_0\hat{x}^{l,j}-\Delta t\beta_0 f\left(\hat{x}^{l,j},u(t_l)\right)+b\right]}_{F\left(\hat{x}^{l,j}\right)}$$

Here j is the Newton iteration index

Newton Iteration : $\left[\alpha_0 I-\Delta t\beta_0\frac{\partial f\left(\hat{x}^{l,j},u(t_l)\right)}{\partial x}\right]\left(\hat{x}^{l,j+1}-\hat{x}^{l,j}\right)=-F\left(x^{l,j}\right)$

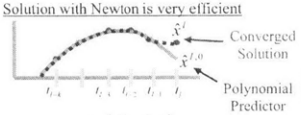
$$\text{Newton}\begin{cases} F(x)=0 \\ \dfrac{\partial F}{\partial x}\left(x^{j}\right)\Delta x^{j+1} = -F\left(x^{j}\right) \end{cases}$$

$$\lambda F(x) + (1-\lambda)x(\lambda) = 0$$

$$\left.\begin{aligned} F(x) &= \alpha_0 \hat{x}^{l} - \Delta t \beta_0 f\left(\hat{x}^{l}, u(t_l)\right) + b \\ \left[\frac{\partial F}{\partial x}\right] &= \alpha_0 \mathbf{I} - \Delta t \beta_0 \left[\frac{\partial f}{\partial x}\right] \end{aligned}\right\} \text{Multistep}$$

$$\frac{dx}{dt} = f(x(t), u(t))$$

Solution with Newton is very efficient



$$\alpha_0 I - \Delta t \beta_0 \frac{\partial f\left(\hat{x}^{l,j}, u(t_l)\right)}{\partial x} \Rightarrow \alpha_0 I \text{ as } \Delta t \to 0$$

Easy to generate a good initial guess using polynomial fitting

Jacobian becomes very diagonally dominant for small timesteps

- easy to factor
- or easy to solve iteratively

$l = 0 \quad x^{0} = x(t=0)$

$l \leftarrow l+1$

REPEAT (Multistep Integration Loop: time)

$\hat{x}^{l,0} \leftarrow \hat{x}^{l-1}$ OR F.E. $\hat{x}^{l,0} \leftarrow \hat{x}^{l-1} + \Delta t f\left(\hat{x}^{l-1}, u(t_l)\right)$

$b \leftarrow \alpha_1 \hat{x}^{l-1} + \ldots + \alpha_k \hat{x}^{l-k} - \Delta t\left[\beta_1 f\left(\hat{x}^{l-1}, u^{l-1}\right) + \ldots + \beta_k f\left(\hat{x}^{l-k}, u^{l-k}\right)\right]$

REPEAT (Newton Loop)

solve

$$\left[\alpha_0 \mathbf{I} - \Delta t \beta_0 \frac{\partial f}{\partial x}\left(\hat{x}^{l,j}, u(t_l)\right)\right]\Delta \hat{x}^{l,j+1} = -\left[\alpha_0 \hat{x}^{l,j} - \Delta t \beta_0 f\left(\hat{x}^{l,j}, u(t_l)\right) + b\right]$$

Solve for $\Delta \hat{x}^{l,j+1}$ with LU or GCR

$\hat{x}^{l,j+1} \leftarrow \hat{x}^{l,j} + \Delta \hat{x}^{l,j+1}$

until $\left\|\Delta \hat{x}^{l,j+1}\right\| \cdot \left\|\alpha_0 \hat{x}^{l,j+1} - \Delta t \beta_0 f\left(\hat{x}^{l,j+1}\right) + b\right\|$ small

$\hat{x}^{l} \leftarrow \hat{x}^{l,j+1}$

$l \leftarrow l+1$

$t = t + \Delta t$

until $t \geq t_{end}$

LU or GCR?

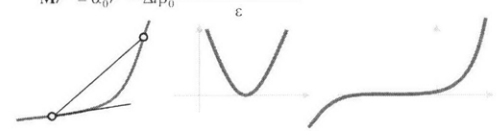
In GCR: at each step need matrix vector product

$$\left[\alpha_0 I - \Delta t \beta_0 \frac{\partial f}{\partial x}\left(\hat{x}^{l,j}, u(t_l)\right)\right] r^{k} \cong$$

$$\cong \alpha_0 r^{k} - \frac{\Delta t \beta_0}{\varepsilon}\left[f\left(\hat{x}^{l,j} + \varepsilon r^{k}, u(t_l)\right) - f\left(\hat{x}^{l,j}, u(t_l)\right)\right]$$

GCR

$$\mathbf{M}\bar{r}^{k} = \alpha_0 \bar{r}^{k} - \Delta t \beta_0 \frac{f\left(x + \varepsilon \bar{r}^{k}\right) - f(x)}{\varepsilon}$$



Using GCR for

$$\underbrace{\left[\alpha_0 I - \Delta t \beta_0 \frac{\partial f}{\partial x}\left(\hat{x}^{l,j}, u(t_l)\right)\right]}_{\text{Jacobian}} \Delta\hat{x}^{l,j+1} = -[\quad \cdots \quad]$$

At each step of GCR need matrix Vector product

$$\underbrace{\left[\alpha_0 I - \Delta t \beta_0 \frac{\partial f}{\partial x}\left(\hat{x}^{l,j}, u(t_l)\right)\right]}_{\text{Jacobian}} \bar{r}^k \cong$$

k-residual

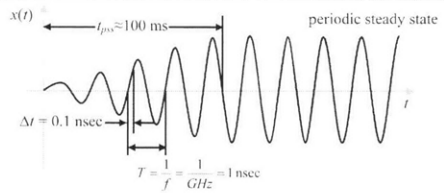
$$\cong \alpha_0 \bar{r}^k - \Delta t \beta_0 \frac{f\left(\hat{x}^{l,j} + \varepsilon \bar{r}^k, u(t_l)\right) - f\left(\hat{x}^{l,j}, u(t_l)\right)}{\varepsilon}$$

M

For small ε

$$\frac{\partial f}{\partial x}\left(\hat{x}^{l,j}, u(t_l)\right)\Delta\bar{x} \cong f\left(\hat{x}^{l,j} + \Delta\bar{x}, u(t_l)\right) - f\left(\hat{x}^{l,j}, u(t_l)\right)$$

for small $\Delta\bar{x} = \varepsilon\bar{r}^k$



$$\text{Total number of time steps} \cong \frac{100\ \text{msec}}{0.1\ \text{nsec}} = 10^9\ \text{steps}$$

Each step solve Newton (very fast e.g. 5 iterations)

Each Newton iteration solve system (e.g. $N = 1000$, sparse $O(N)$ say 10,000 FLOPS)

Each time step 50,000 FLOPS

say 1 GFLOP/sec

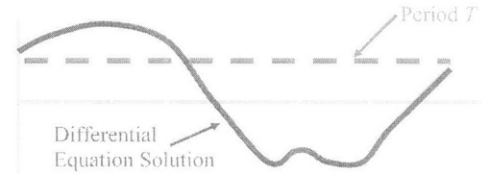
$$\frac{50{,}000\text{FLOPS}}{1\times10^9\ \frac{\text{FLOPS}}{\text{sec}}} = 50\mu\text{sec}_{\text{per time step}}$$

$50\mu\text{sec} \times 10^9 = 50{,}000\text{sec} \cong 14h \approx \frac{1}{2}$ day!!!

FINITE-DIFFERENCE METHODS FOR PERIODIC STEADY-STATE

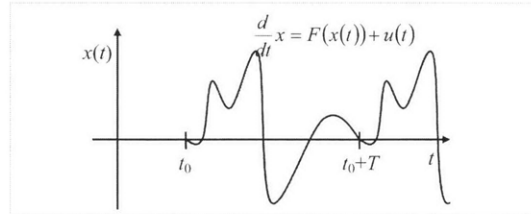
Boundary-Value Problem

- Basic Formulation



N Differential Equations: $\frac{d}{dt}x_i(t) = F_i(x(t))$

N Periodicity Constraints: $x_i(T) = x_i(0)$



- Finite Difference Methods
  - Linear Example Problem

$$\frac{dx(t)}{dt} = Ax(t) + \underbrace{u(t)}_{\text{input}} \quad t \in [0, T] \quad \underbrace{x(T) = x(t)}_{\text{periodicity constraint}}$$

Discretize e.g. with Backward-Euler

$$\hat{x}^1 = x(0) + \Delta t \cdot \left(A\hat{x}^1 + u(\Delta t)\right)$$
$$\hat{x}^2 = \hat{x}^1 + \Delta t \cdot \left(A\hat{x}^2 + u(2\Delta t)\right) \qquad \Delta t = \frac{T}{L}$$
$$\vdots$$
$$\hat{x}^L = \hat{x}^{L-1} + \Delta t \cdot \left(A\hat{x}^L + u(L\Delta t)\right)$$

Periodicity implies $\hat{x}^L = \hat{x}^0$

$$
\begin{array}{c}
\longleftarrow N \times L \longrightarrow \\
N \times L \updownarrow
\end{array}
\begin{bmatrix}
\mathbf{I}-\Delta t\mathbf{A} & 0 & \cdots & 0 & -\mathbf{I} \\
-\mathbf{I} & \mathbf{I}-\Delta t\mathbf{A} & 0 & \cdots & 0 \\
0 & -\mathbf{I} & \mathbf{I}-\Delta t\mathbf{A} & \cdots & 0 \\
\vdots & \vdots & \ddots & \ddots & \vdots \\
0 & 0 & \cdots & -\mathbf{I} & \mathbf{I}-\Delta t\mathbf{A}
\end{bmatrix}
\begin{bmatrix} \hat{x}^1 \\ \hat{x}^2 \\ \hat{x}^3 \\ \vdots \\ \hat{x}^L \end{bmatrix}
= \Delta t
\begin{bmatrix} u(\Delta t) \\ u(2\Delta t) \\ u(3\Delta t) \\ \vdots \\ u(L\Delta t) \end{bmatrix}
$$

Matrix is almost lower triangular

$$
\begin{aligned}
&\hat{x}^0 = \hat{x}^L \\
&\hat{x}^1 = \hat{x}^0 + \Delta t\left(\mathbf{A}\hat{x}^1 + u(\Delta t)\right) \\
&\Rightarrow \hat{x}^1 = \hat{x}^L + \Delta t\left(\mathbf{A}\hat{x}^1 + u(\Delta t)\right) \\
&\Rightarrow (I - \Delta t\mathbf{A})\hat{x}^1 - \hat{x}^L = \Delta t u(\Delta t) \\
&\hat{x}^2 = \hat{x}^1 + \Delta t\left(\mathbf{A}\hat{x}^2 + u(2\Delta t)\right)
\end{aligned}
$$

Backward Euler

$$
\begin{bmatrix}
\mathbf{I}-\Delta t\mathbf{A} & & & -\mathbf{I} \\
-\mathbf{I} & \mathbf{I}-\Delta t\mathbf{A} & & \\
 & \ddots & \ddots & \\
 & & -\mathbf{I} & \mathbf{I}-\Delta t\mathbf{A}
\end{bmatrix}
\begin{bmatrix} \hat{x}^1 \\ \hat{x}^2 \\ \vdots \\ \hat{x}^L \end{bmatrix}
= \Delta t
\begin{bmatrix} u(\Delta t) \\ u(2\Delta t) \\ \vdots \\ u(L\Delta t) \end{bmatrix}
$$

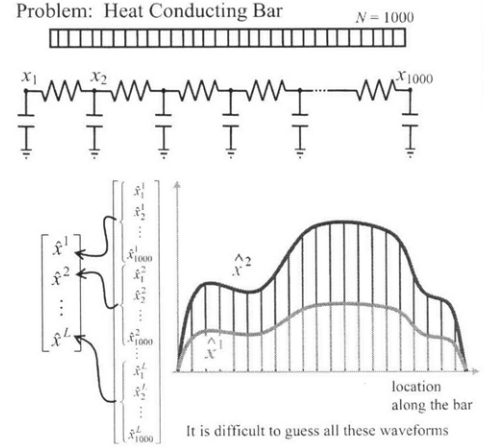
$$
\hat{x}^L = \hat{x}^{L-1} + \Delta t\left(\mathbf{A}\hat{x}^L + u(L\Delta t)\right)
$$

$$
\begin{aligned}
&\hat{x}^1 = \overbrace{\hat{x}^0}^{\hat{x}^L} + \Delta t\left(\mathbf{A}\hat{x}^0 + u(\Delta t)\right) \\
&\hat{x}^2 = \hat{x}^1 + \Delta t\left(\mathbf{A}\hat{x}^2 + u(2\Delta t)\right)
\end{aligned}
$$

Forward Euler

$$
\begin{bmatrix}
\mathbf{I} & & & -\mathbf{I}-\Delta t\mathbf{A} \\
-\mathbf{I}-\Delta t\mathbf{A} & \mathbf{I} & & \\
 & \ddots & \ddots & \\
 & & -\mathbf{I}-\Delta t\mathbf{A} & \mathbf{I}
\end{bmatrix}
\begin{bmatrix} \hat{x}^1 \\ \hat{x}^2 \\ \vdots \\ \hat{x}^L \end{bmatrix}
= \Delta t
\begin{bmatrix} u(\Delta t) \\ u(2\Delta t) \\ \vdots \\ u(L\Delta t) \end{bmatrix}
$$

Problem: Heat Conducting Bar



It is difficult to guess all these waveforms

Problem: Huge matrix to solve with $N \times L$ unknowns

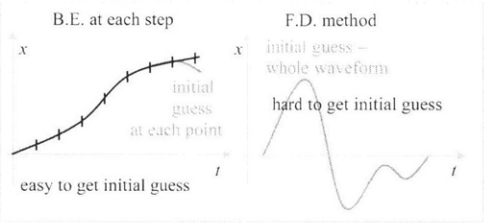
- Nonlinear Problem

$$
\frac{dx(t)}{dt} = F(x(t)) + \underbrace{u(t)}_{\text{input}} \quad t \in [0, T] \quad \underbrace{x(T) = x(t)}_{\text{periodicity constraint}}
$$

Discretize e.g. with Backward-Euler

$$
H_{FD}\left(\begin{bmatrix} \hat{x}^1 \\ \hat{x}^2 \\ \vdots \\ \hat{x}^L \end{bmatrix}\right) = \begin{bmatrix} \hat{x}^1 - \hat{x}^L - \Delta t\left[F\left(\hat{x}^1\right) + u(\Delta t)\right] \\ \hat{x}^2 - \hat{x}^1 - \Delta t\left[F\left(\hat{x}^2\right) + u(2\Delta t)\right] \\ \vdots \\ \hat{x}^L - \hat{x}^{L-1} - \Delta t\left[F\left(\hat{x}^L\right) + u(L\Delta t)\right] \end{bmatrix} = 0
$$

Solve the huge system using Newton's Method



$$
\begin{bmatrix}
\mathbf{I}-\Delta t\left.\frac{\partial F}{\partial x^1}\right|_{\hat{x}^{1,j}} & & & -\mathbf{I} \\
-\mathbf{I} & \mathbf{I}-\Delta t\left.\frac{\partial F}{\partial x^2}\right|_{\hat{x}^{2,j}} & & \\
 & -\mathbf{I} & \ddots & \\
 & & -\mathbf{I} \quad \mathbf{I} & \mathbf{I}-\Delta t\left.\frac{\partial F}{\partial x^L}\right|_{\hat{x}^{L,j}}
\end{bmatrix}
\begin{bmatrix}
\Delta\hat{x}^{1,j+1} \\ \Delta\hat{x}^{2,j+1} \\ \vdots \\ \Delta\hat{x}^{L,j+1}
\end{bmatrix}
=
$$

Jacobian

$$
= -\begin{bmatrix}
\hat{x}^{1,j}-\hat{x}^{L,j}-\Delta t\left[F\left(\hat{x}^{1,j}\right)+u(\Delta t)\right] \\
\hat{x}^{2,j}-\hat{x}^{1,j}-\Delta t\left[F\left(\hat{x}^{2,j}\right)+u(2\Delta t)\right] \\
\vdots \\
\hat{x}^{L,j}-\hat{x}^{L-1,j}-\Delta t\left[F\left(\hat{x}^{L,j}\right)+u(L\Delta t)\right]
\end{bmatrix}
$$

Size $N \times L = 1{,}000 \times 10 = 10{,}000$

Sparse → $O(10{,}000)$ for each solve → 100,000 FLOPS

→ Say 10 iterations → $10 \times 100{,}000$ FLOPS = 10^6 FLOPS

$$\Rightarrow \frac{10^6}{10^9} = 1 \text{ msec} << \frac{1}{2} \text{ day}$$

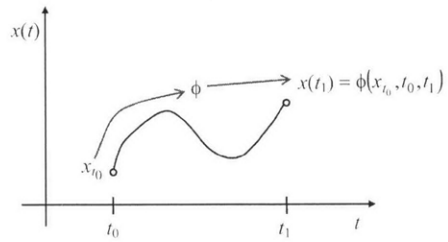
How many Newton operations?

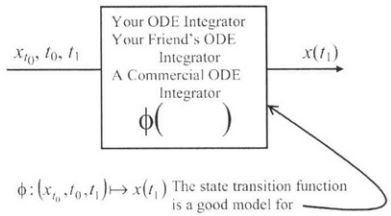
Trade Off
- Δt small – converges faster
  – but large matrix to solve
- Δt large – converges slower
  – but smaller matrix to solve

Full Time Domain Integration
(e.g. 10^9 Time Steps | 5 Newton Iterations | N=1,000 Sparse | → ½ day)

Finite Difference Method
(e.g. 10 Newton Iterations | L=10 Time Steps per Period | N=1,000 | → 1 msec)

Problem:
1. Initial Condition
2. Need internal access to simulator equations





$\phi : (x_{t_0}, t_0, t_1) \mapsto x(t_1)$ The state transition function is a good model for

$$\phi(x_{t_0}, t_0, t_1) \rightarrow x(t_1) = x_{t_0} e^{\lambda(t_1 - t_0)}$$

$$\frac{d}{dt} x(t) = \lambda x(t)$$

$$x(t) = \mathbf{A} e^{\lambda t}$$

$$x_{t_0} = \mathbf{A} e^{\lambda t_0} \rightarrow \mathbf{A} = x_{t_0} e^{-\lambda t_0}$$

$$x(t) = x_{t_0} e^{\lambda(t - t_0)} \qquad x(t_1) = x_{t_0} e^{\lambda(t_1 - t_0)}$$

SHOOTING METHODS

State Transition Function

- Basic Definitions

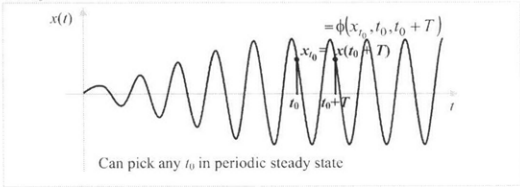
Start with $\dfrac{dx(t)}{dt} = F(x(t)) + u(t)$

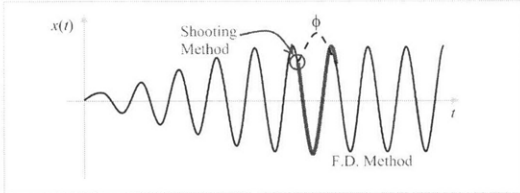
And assume $x(t)$ is unique given $x(0)$.

The O.D.E. defines a State-Transition Function

$$\Phi(x_{t_0}, t_0 t_1) = x(t_1)$$

Where $x(t_1)$ is the O.D.E. solution at time t_1 when starting from state $x_{t_0} = x(t_0)$ at time t_0

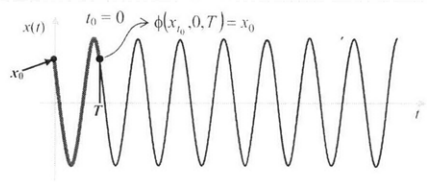




- State Transition Function Example

$$\frac{dx(t)}{dt} = \lambda x(t)$$

$$\Phi(x_{t_0}, t_0, t_1) = e^{\lambda(t_1 - t_0)} x_{t_0}$$



Assume we are already in periodic steady state

❖ Abstract Formulation

Solve $\underbrace{x(t_0 + T)}_{\Phi(x_{t_0}, t_0, t_0+T) = x_{t_0}} = x_{t_0}$ for a t_0 of your choice

e.g. pick $t_0 = 0$ $\quad H(x_0) = \Phi(x_0, 0, T) - x_0 = 0$

Evaluate $H(x^k)$

Integrate $\frac{d}{dt}x(t) = F(x(t)) + u(t)$, $\quad x(0) = x^k$

Over the interval $[0, T]$

x_0 unknown

$H(x_0) = 0$

$\underbrace{\phi(x_0, 0, T) - x_0}_{H(x_0)} = 0$

$\frac{\partial H}{\partial x_0}$

REPEAT

$$\left[\left.\frac{\partial \phi}{\partial x_0}\right|_{x_0^k} - \mathbf{I}\right]\Delta x_0^{k+1} = -\left[\phi(x_0^k, 0, T) - x_0^k\right]$$

UNTIL $\|\Delta x_0^{k+1}\|, \|H(x_0^k)\|$ small

Solve $H(x_0) = 0$ for x_0 using Newton's method

$$J_H(x_0^k)(x_0^{k+1} - x_0^k) = -H(x_0^k)$$

where $J_H(x_0^k) = \frac{\partial \Phi(x_0^k, 0, T)}{\partial x_0} - I$

$k \leftarrow 0$

$x^k \leftarrow x_0$

Note this is not so different to guess anymore; it is just <u>one</u> distribution of temperatures.

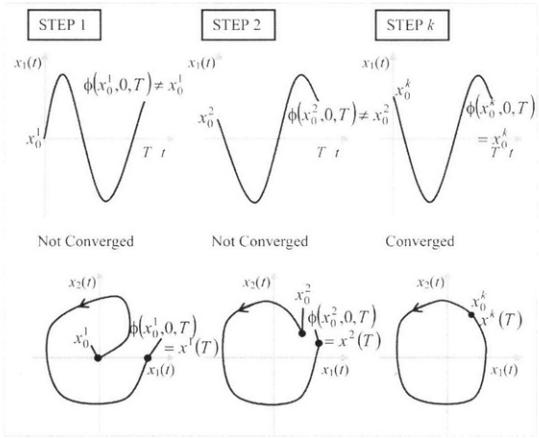
REPEAT

$$\left[\left.\frac{\partial \phi}{\partial x_0}\right|_{x_0^k} - \mathbf{I}\right]\Delta x_0^{k+1} = -\left[\phi(x_0^k, 0, T) - x_0^k\right]$$

$x_0^{k+1} \leftarrow x_0^k + \Delta x_0^{k+1}$

$k \leftarrow k + 1$

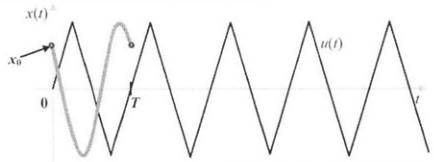
UNTIL $\|\Delta x_0^k\|, \|\phi(x_0^{k-1}, 0, T) - x_0^{k-1}\|$ small



Not Converged | Not Converged | Converged

$$\left[\left.\frac{\partial F}{\partial x}\right|_{x_0}\right]\bar{\varepsilon} \cong F(x_0+\bar{\varepsilon})-F(x_0) \qquad \Delta x \text{ small}$$

$$\underbrace{\left[\frac{\partial \phi}{\partial x_0}(x_0,0,T)\right]}_{\text{Matrix}}\bar{\varepsilon} \cong \underbrace{\phi(x_0+\bar{\varepsilon},0,T)}_{x^{\varepsilon}(T)}-\underbrace{\phi(x_0,0,T)}_{x(T)}$$

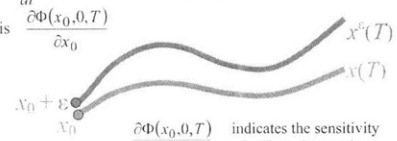


❖ Computing Newton

In order to calculate $H(x_0)=\Phi(x_0,0,T)-x_0$

Integrate $\dfrac{dx(t)}{dt}=F(x(t))+u(t)$ on $[0,T]$

But what is $\dfrac{\partial \Phi(x_0,0,T)}{\partial x_0}$

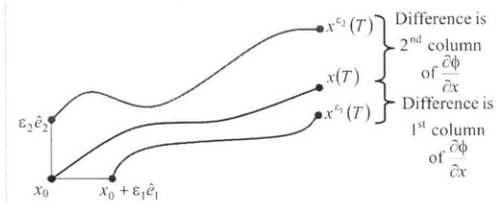


$\dfrac{\partial \Phi(x_0,0,T)}{\partial x_0}$ indicates the sensitivity of $x(T)$ to changes in x_0

$$\left[\frac{\partial \phi}{\partial x}\right]\begin{bmatrix} x_1^p(0)-x_1(0) \\ \vdots \\ x_N^p(0)-x_N(0)\end{bmatrix} \cong \begin{bmatrix} x_1^0(T)-x_1(T) \\ \vdots \\ x_N^0(T)-x_N(T)\end{bmatrix}$$

If we choose $\bar{\varepsilon}=\varepsilon_1\hat{e}_1=\begin{bmatrix}\varepsilon_1\\0\\0\\0\\\vdots\end{bmatrix}$ we get the first column of $\dfrac{\partial \phi}{\partial x}(x_0,0,T)$

First column of $\dfrac{\partial \phi}{\partial x} \cong \begin{bmatrix} \dfrac{x_1^{\varepsilon_1}(T)-x_1(T)}{\varepsilon_1} \\ x_2^{\varepsilon_1}(T)-x_2(T) \\ \vdots \\ x_N^{\varepsilon_1}(T)-x_N(T)\end{bmatrix}$



This is not efficient we need to do a lot of integrations to get the entire matrix (e.g. $M=1000$ columns)

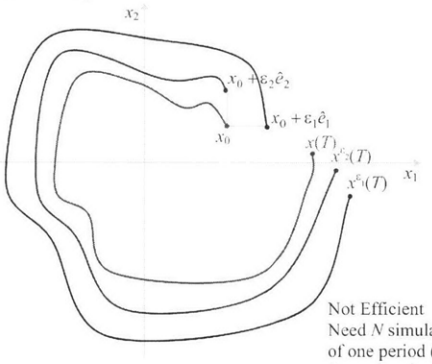
$$\left[\frac{\partial\phi}{\partial x_0}(x_0,0,T)\right]\bar{\varepsilon} \cong \phi(x_0+\bar{\varepsilon},0,T)-\phi(x_0,0,T)=x^{\varepsilon}(T)-x(T)$$

First Column use $\bar{\varepsilon}=\varepsilon_1\hat{e}_1$

$$\left[\frac{\partial\phi}{\partial x_0}(x_0,0,T)\right]_1 \varepsilon_1\hat{e}_1 \cong \phi(x_0+\varepsilon_1\hat{e}_1,0,T)-\phi(x_0,0,T)=x^{\varepsilon_1}(T)-x(T)$$

Second Column use $\bar{\varepsilon}=\varepsilon_2\hat{e}_2$

$$\left[\frac{\partial\phi}{\partial x_0}(x_0,0,T)\right]_2 \varepsilon_2\hat{e}_2 \cong \phi(x_0+\varepsilon_2\hat{e}_2,0,T)-\phi(x_0,0,T)=x^{\varepsilon_2}(T)-x(T)$$



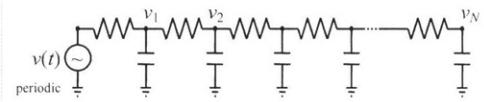
Not Efficient
Need N simulations
of one period (one
for each column)

Sensitivity Matrix

- Perturbation

$$\frac{\partial\Phi(x,0,T)}{\partial x} \approx \begin{bmatrix} \frac{x_1^{\varepsilon_1}(T)-x_1(T)}{\varepsilon_1} & \cdots & \frac{x_1^{\varepsilon_N}(T)-x_1(T)}{\varepsilon_N} \\ \vdots & \ddots & \vdots \\ \frac{x_N^{\varepsilon_1}(T)-x_N(T)}{\varepsilon_1} & \cdots & \frac{x_N^{\varepsilon_N}(T)-x_N(T)}{\varepsilon_N} \end{bmatrix}$$

Is $\frac{\partial\phi}{\partial x}(x_0,0,T)$ DENSE or SPARSE?



Try $x_0=\begin{bmatrix}0\\0\\\vdots\\0\end{bmatrix}$ Integrate over one period get $\phi(x_0,0,T)=x(T)$ $\begin{bmatrix}0\\0\\\vdots\\0\end{bmatrix}$

Try perturbing initial condition $x_0+\varepsilon_1\hat{e}_1=\begin{bmatrix}\varepsilon_1\\0\\\vdots\\0\end{bmatrix}$

Integrate over one period get

$$\phi(x_0+\varepsilon_1\hat{e}_1,0,T)=x^{\varepsilon_1}(T)=\begin{bmatrix}*\\*\\\vdots\\*\end{bmatrix}$$ all nonzero

So the matrix $\frac{\partial\phi}{\partial x}$ is DENSE.

For $i=1$ to N

$x_{0_i}^{\varepsilon_i}=x_{0_i}+\varepsilon$

Compute $x(T)=\Phi\left(x_0^{\varepsilon_i},0,T\right)$

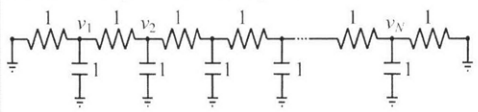
Column of $\frac{\partial\Phi}{\partial x}$

N Columns $\rightarrow$ N Integrations

Integrate $x=F(x)$

$x(0)=x_0^{\varepsilon_i}$ on $[0,T]$

Is the sensitivity matrix dense or sparse?



$$\frac{d}{dt}\begin{bmatrix} v_1 \\ v_2 \\ \vdots \\ v_N \end{bmatrix} = \begin{bmatrix} 2 & -1 & & \\ -1 & 2 & \ddots & \\ & \ddots & \ddots & -1 \\ & & -1 & 2 \end{bmatrix}\begin{bmatrix} v_1 \\ v_2 \\ \vdots \\ v_N \end{bmatrix}$$

In general the sensitivity matrix will be dense

$\frac{\partial \Phi}{\partial x}$ by perturbation

For $i = 1$ to N

Perturb $x_i(0)$

Compute $x^{(i)}(T)$ $\longrightarrow$ $\frac{\partial \Phi}{\partial x}$

❖ Efficient Sensitivity Evaluation

Differentiate the first step of Backward-Euler

$$\frac{\partial}{\partial x(0)}\left(\hat{x}^1 - x(0) - \Delta t\left(F\left(\hat{x}^1\right) + u(\Delta t)\right) = 0\right)$$

$$\Rightarrow \frac{\partial \hat{x}^1}{\partial x(0)} - \frac{\partial x(0)}{\partial x(0)} - \Delta t \frac{\partial F\left(\hat{x}^1\right)}{\partial x}\frac{\partial \hat{x}^1}{\partial x(0)} = 0$$

$$\Rightarrow \left(I - \Delta t \frac{\partial F\left(\hat{x}^1\right)}{\partial x}\right)\frac{\partial \hat{x}^1}{\partial x(0)} = \overbrace{\frac{\partial x(0)}{\partial x(0)}}^{I}$$

Applying the same trick on the lth step

$$\Rightarrow \left(I - \Delta t \frac{\partial F\left(\hat{x}^l\right)}{\partial x}\right)\frac{\partial \hat{x}^l}{\partial x(0)} = \frac{\partial \hat{x}^{l-1}}{\partial x(0)}$$

$$\frac{\partial \Phi(x,0,T)}{\partial x} \approx \prod_{l=1}^{L}\left(I - \Delta t \frac{\partial F\left(\hat{x}^l\right)}{\partial x}\right)^{-1}$$

B.E. $\quad \hat{x}^1 - x(0) - \Delta t\left(F\left(\hat{x}^1\right) + u(\Delta t)\right) = 0$

Apply Newton $\quad \left(I - \Delta t \frac{\partial F\left(\hat{x}^{1,k}\right)}{\partial x}\right)\Delta \hat{x}^{1,k+1} = \ldots$

$$\frac{d}{dt}x(t) = \mathbf{A}x(t) \Rightarrow x(t) = e^{\mathbf{A}t}x(0)$$

$$\frac{d\phi}{dx(0)} \cong (\mathbf{I} - \Delta t\mathbf{A})^{-l}$$

$$\frac{d\phi}{dx(0)} = e^{\mathbf{A}T} = e^{\mathbf{A}(l\Delta t)}$$

$$\begin{aligned}
\frac{\partial \phi}{\partial x} = \frac{\partial \hat{x}^l}{\partial x_0} &= \left[I - \Delta t \frac{\partial F}{\partial x}\left(\hat{x}^l\right)\right]^{-1}\frac{\partial \hat{x}^{l-1}}{\partial x(0)} \\
&= \left[I - \Delta t \frac{\partial F}{\partial x}\left(\hat{x}^l\right)\right]^{-1}\left[I - \Delta t \frac{\partial F}{\partial x}\left(\hat{x}^{l-1}\right)\right]^{-1}\frac{\partial \hat{x}^{l-2}}{\partial x(0)} \\
&\quad \vdots \\
&= \underbrace{\left[I - \Delta t \frac{\partial F}{\partial x}\left(\hat{x}^l\right)\right]^{-1}\left[I - \Delta t \frac{\partial F}{\partial x}\left(\hat{x}^{l-1}\right)\right]^{-1}\cdots\left[I - \Delta t \frac{\partial F}{\partial x}\left(\hat{x}^1\right)\right]^{-1}}_{l} I
\end{aligned}$$

In practice we solve l systems (each with size N)

❖ Observations on Sensitivity Matrix

Newton at each timestep uses same matrices

$$\frac{\partial \Phi(x,0,T)}{\partial x} \approx \prod_{l=1}^{L}\underbrace{\left(I - \Delta t \frac{\partial F\left(\hat{x}^l\right)}{\partial x}\right)^{-1}}_{\substack{\textit{Timestep} \\ \textit{Newton} \\ \textit{Jacobian}}}$$

Formula simplifies in the linear case

$$\frac{\partial \Phi(x,0,T)}{\partial x} \approx (I - \Delta t A)^{-L}$$

$$\frac{dx}{dt} = \mathbf{A}x(t) + u(t) \qquad F(x(t)) = \mathbf{A}x(t)$$

$$x(t) = \underbrace{e^{\mathbf{A}t}}_{\downarrow} x(0) \qquad T = \Delta t \cdot l$$

$$\begin{aligned}
\left[\frac{\partial \phi}{\partial x}\right]\Delta x(0) &\cong \phi(x(0) + \Delta x(0), 0, T) - \phi(x(0), 0, T) \\
&\cong e^{\mathbf{A}\Delta t}[x(0) + \Delta x] - e^{\mathbf{A}\Delta t}x(0) \\
&= \left[e^{\mathbf{A}\Delta t}\right]\Delta x(0) \\
&= \left[e^{-\mathbf{A}\Delta t}\right]^{-1}\Delta x(0)
\end{aligned}$$

$$\left[\frac{\partial \phi}{\partial x}\right] \cong [\mathbf{I} - \mathbf{A}\Delta t]^{-1} \quad \text{First Order Expansion (B.E.)}$$

Matrix-Free Approach

- Basic Setup

REPEAT

$$\underbrace{\left[\frac{\partial \phi}{\partial x_0}\Big|_{x_0^k} - \mathbf{I}\right]}_{\mathbf{M}} \underbrace{\Delta x_0^{k+1}}_{x} = \underbrace{-\left[\phi(x_0^k, 0, T) - x_0^k\right]}_{b}$$

$$x_0^{k+1} \leftarrow x_0^k + \Delta x_0^{k+1}$$

$$k \leftarrow k+1$$

UNTIL $\left\|\Delta x_0^k\right\|, \left\|\phi(x_0^{k-1}, 0, T) - x_0^{k-1}\right\|$ small

Start with $\dfrac{dx(t)}{dt} = F(x(t)) + u(t)$

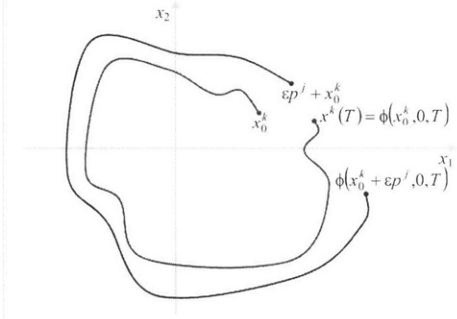
$$H(x_0) = \Phi(x_0, 0, T) - x_0 = 0$$

Solve $H(x_0) = 0$ for x_0 using Newton's method

$$J_H(x_0^k)(x_0^{k+1} - x_0^k) = -H(x_0^k)$$

where $J_H(x_0^k) = \dfrac{\partial \Phi(x_0^k, 0, T)}{\partial x_0} - I$

$$\left[\frac{d\phi}{dx}(x_0^k, 0, T) - I\right]p^j = \frac{\left[\phi(x_0^k + \delta p^j, 0, T) - \phi(x_0^k, 0, T)\right]}{\varepsilon} - p^j$$



- Matrix-Vector Product

Solve Newton equation with Krylov-subspace method

$$\underbrace{\left(\frac{\partial \Phi(x_0^k, 0, T)}{\partial x_0} - I\right)}_{\mathbf{M}} \underbrace{(x_0^{k+1} - x_0^k)}_{\tilde{x}} = \underbrace{x_0^k - \Phi(x_0^k, 0, T)}_{\tilde{b}}$$

Matrix-Vector Product Computation

$$\left(\frac{\partial \Phi(x_0^k, 0, T)}{\partial x_0} - I\right)p^j \approx \frac{\Phi(x_0^k + \varepsilon p^j, 0, T) - \Phi(x_0^k, 0, T)}{\varepsilon} - p^j$$

Krylov method search direction

$$\mathbf{A}p = \left[\frac{\partial \phi}{\partial x} - I\right]p = \frac{\partial \phi}{\partial x}p - p$$

Newton Iteration

$$\left[\frac{\partial\phi}{\partial x} - I\right]\left[\Delta x^{j+1}(0)\right] = -\underbrace{H\left(x^{j}(0)\right)}_{\text{call integration}}$$

Use Krylov

$$\underbrace{\left[\frac{\partial\phi}{\partial x} - I\right]p}_{\text{Matrix-Vector Product}} \cong \frac{\phi\left(x^{j}(0) + \varepsilon p\right) - \phi\left(x^{j}(0)\right)}{\varepsilon} - p$$

1.)	Computing P.S.S.	$x(T) = x(0)$
2.)	Use Shooting	$\phi(x(0), 0, T) - x(0) = 0$
3.)	Use Newton	$\left(\frac{\partial\phi}{\partial x} - I\right)\Delta x^{k-1}(0) = -\left(\phi\left(x^{k}, 0, T\right) - x^{k}\right)$
4.)	Use Krylov to Solve Newton	$\left(\frac{\partial\phi}{\partial x} - I\right)p$

search direction

- Convergence for GCR

  Example

  $$\frac{dx}{dt} - Ax = 0 \quad eig(A) \text{ real and negative}$$

  Shooting-Newton Jacobian

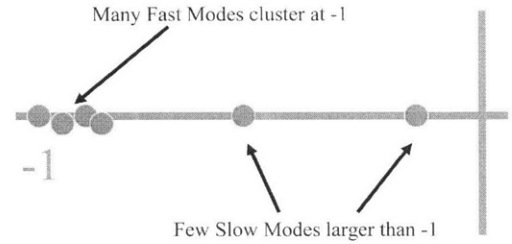
  $$\frac{\partial\Phi(x, 0, T)}{\partial x} - I = e^{AT} - I$$

$$eig\left(e^{\mathbf{A}T} - I\right) \quad \text{versus} \quad eig(\mathbf{A})$$

$$e^{\lambda T} - 1$$

$eig(\mathbf{A})$

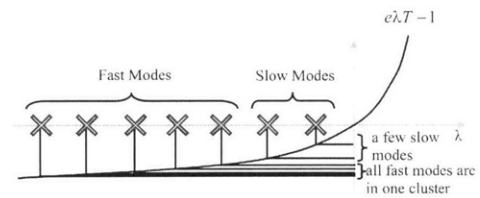
$$e^{AT} - I = S\begin{bmatrix} e^{\lambda_1 T} - 1 & & \\ & \ddots & \\ & & e^{\lambda_N T} - 1 \end{bmatrix} S^{-1}$$



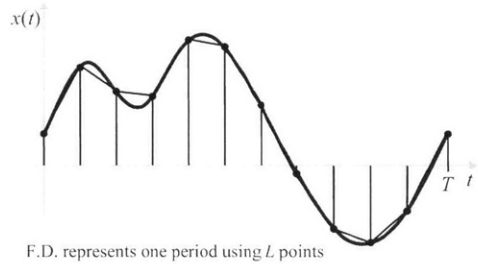
Use Spectral Mapping Theorem

$$eig\left(e^{\mathbf{A}t} - \mathbf{I}\right)$$

if λ_i is eigenvalue of $\mathbf{A}$ $\rightarrow$ $e^{\lambda_i t} - 1$ is eigenvalue of $e^{\mathbf{A}T} - I$



Typically there are many fast modes (each strut) and few slow modes (the whole frame)



F.D. represents one period using L points

Maybe we can use other ways to represent the shape

Periodic → use Fourier Coefficients

$$x(t) = \cdots + x_{-2}e^{-i2\pi\frac{t}{T}2} + x_{-1}e^{-i2\pi\frac{t}{T}} + x_0 + x_1 e^{i2\pi\frac{t}{T}} + x_2 e^{i2\pi\frac{t}{T}2} + \cdots$$

frequency $\frac{1}{T}$ fundamental

frequency $\frac{2}{T}$ fundamental

2nd Harmonic etc…

Works well (need few coefficients) if function smooth
harmonics

Let's solve for x_l

SPECTRAL METHODS (HARMONIC BALANCE)

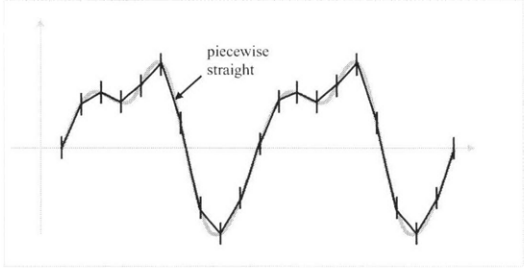
Fourier Representation

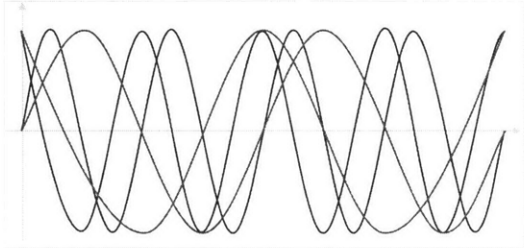
- Truncation Approximation
  - Periodic Function → Fourier Series

$$x(t) = \sum_{l=-\infty}^{\infty} X_l e^{-i2\pi l\frac{t}{T}}$$

  - Approximate a function with truncated series

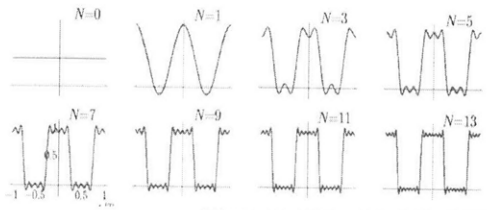
$$x(t) \approx \sum_{l=-L}^{L} X_l e^{-i2\pi l\frac{t}{T}}$$





❖ Square Wave Example

$$x_N(t) = \frac{1}{2} + \sum_{n=1}^{N} \left(\frac{\sin(n\pi/2)}{n\pi/2} \right) \cos(2\pi n t/T).$$



❖ Annoyance for Real Functions

- Real $x \rightarrow$ Fourier Coefficients complex conjugate

  $X_{-l} = X_l^*$

- Can rewrite series with fewer unknowns

$$x(t) = \sum_{l=1}^{\infty} \underbrace{X_l e^{-j2\pi l \frac{t}{T}} + X_l^* e^{+j2\pi l \frac{t}{T}}}_{\text{Real}} + \underbrace{X_0}_{l=0}$$

$$\langle f(t), g(t) \rangle_T = \int_0^T g(t)^* f(t) dt$$

$$l = m \quad \int_0^T e^{j2\pi m \frac{t}{T}} e^{-j2\pi m \frac{t}{T}} dt = \int_0^T dt = T = \left\| e^{j2\pi m \frac{t}{T}} \right\|^2$$

$$\|f(t)\|^2 = \langle f(t), g(t) \rangle_T$$

$$x(t) = \ldots + X_{-1} e^{-j2\pi \frac{t}{T}} + X_0 + X_1 e^{j2\pi \frac{t}{T}} + \cdots$$

$\{\hat{Q}_1, \hat{Q}_2, \ldots, \hat{Q}_n\}$ orthonormal vectors

$$\bar{b} = y_1 \hat{Q}_1 + y_2 \hat{Q}_2 + \cdots + y_n \hat{Q}_n \qquad y_i = \langle \bar{b}, \hat{Q}_i \rangle$$

$$y_1 = \langle \bar{b}, \hat{Q}_1 \rangle \qquad y_2 = \langle \bar{b}, \hat{Q}_2 \rangle$$

Orthogonal Projection

$$\hat{Q}_1 = \frac{\bar{v}_1}{\|\bar{v}_1\|}$$

$\{\bar{v}_1, \bar{v}_2, \ldots, \bar{v}_n\}$ orthogonal vectors

$$\alpha_i = \frac{\langle \bar{b}, \bar{v}_i \rangle}{\|\bar{v}_i\|}$$

$$\bar{b} = \alpha_1 \bar{v}_1 + \alpha_2 \bar{v}_2 + \cdots + \alpha_n \bar{v}_n$$

$$\alpha_1 = \frac{\langle \bar{b}, \bar{v}_1 \rangle}{\|\bar{v}_1\|^2} \qquad \alpha_2 = \frac{\langle \bar{b}, \bar{v}_2 \rangle}{\|\bar{v}_2\|^2}$$

Orthogonal Projection

Compare

$$x(t) = \ldots + x_{-2} e^{-j\frac{2\pi}{T}2t} + x_{-1} e^{-j\frac{2\pi}{T}t} + x_0 + x_1 e^{j\frac{2\pi}{T}t} + x_2 e^{j\frac{2\pi}{T}2t} + \ldots$$

$$b = \ldots + \alpha_{-2} v_{-2} + \alpha_{-1} v_{-1} + \alpha_0 + \alpha_1 v_1 + \alpha_2 v_2 + \ldots$$

$$b \leftrightarrow x(t)$$

$$\bar{v}_m \leftrightarrow e^{jm\frac{2\pi}{T}t}$$

$$\alpha_m \leftrightarrow x_m$$

$$x(t) = X_1 e^{-j2\pi \frac{t}{T}} + X_2 e^{-j2\pi \frac{t}{T}2} + \cdots$$

$$X_1 = \frac{1}{T} \int_0^T e^{j2\pi \frac{t}{T}} x(t) dt \qquad X_2 = \frac{1}{T} \int_0^T e^{j2\pi \frac{t}{T}2} x(t) dt$$

$$x_m = \frac{1}{T} \int_0^T e^{j2\pi m \frac{t}{T}} x(t) dt$$

- Orthogonality
  - Terms in Fourier Series are orthogonal

$$\int_0^T e^{i2\pi l \frac{t}{T}} e^{-i2\pi m \frac{t}{T}} dt = 0 \qquad l \neq m$$

  - Simple formula for computing coefficients

$$\int_0^T e^{-i2\pi m \frac{t}{T}} x(t) dt = \int_0^T e^{i2\pi m \frac{t}{T}} \sum_{l=-\infty}^{\infty} X_l e^{-i2\pi l \frac{t}{T}} dt = T X_m$$

- Advantages
  - For smooth functions (infinitely cont. diff)

$$\lim_{m \to \infty} \frac{1}{T} \int_0^T e^{-i2\pi m \frac{t}{T}} x(t) dt = \lim_{m \to \infty} X_m = O\left(c^m\right)$$

  - Automatically satisfies periodicity

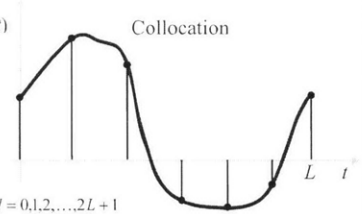
$$x(t+T) = \sum_{l=-L}^{L} X_l e^{-i2\pi l \left(\frac{t+T}{T}\right)} = \sum_{l=-L}^{L} X_l e^{-i2\pi l \frac{t}{T}} = x(t)$$

Computing Coefficients

$$\frac{dx}{dt} = F(x(t)) + u(t)$$

$$x(t) = \sum_{l=-L}^{L} X_l e^{-i2\pi l t}$$

unknowns!



$$R(X, t_L) = 0 \quad l = 0,1,2,\ldots,2L+1$$

- Residual
  - Plug representation in differential equation

$$\underbrace{R(\bar{X}, t)}_{\text{Residual}} = \frac{d}{dt}\left(\sum_{l=-L}^{L} X_l e^{-i2\pi l \frac{t}{T}}\right) - F\left(\sum_{l=-L}^{L} X_l e^{-i2\pi l \frac{t}{T}}\right) - u(t)$$

$$\frac{d}{dt}\underbrace{x(t)} = f(x(t)) + u(t)$$

$$\sum X_l e^{i2\pi l \frac{t}{T}}$$

$$\frac{d}{dt}(\ ) - F(\ ) - U(t) = 0$$

  - Simplify by differentiating representation

$$\underbrace{R(\bar{X}, t)}_{\text{Residual}} = \sum_{l=-L}^{L} -\frac{i2\pi l}{T} X_l e^{-i2\pi l \frac{t}{T}} - F\left(\sum_{l=-L}^{L} X_l e^{-i2\pi l \frac{t}{T}}\right) - u(t)$$

- Collocation & Galerkin
  - Collocation: residual = 0 at test points

$$\underbrace{R(\bar{X}, t)}_{\text{Residual}} = 0 \qquad l = \{1, \ldots, 2L+1\}$$

$$x(t) = \sum_{l=-L}^{L} X_l e^{i2\pi l \frac{t}{T}}$$

complex

  - Galerkin: residual orthogonal to Fourier Terms

$$\int_0^T e^{i2\pi m \frac{t}{T}} \underbrace{R(\bar{X}, t)}_{\text{Residual}} dt = 0 \qquad m = \{-L, \ldots, 0, \ldots, L\}$$

$$R(x, t) = 0$$

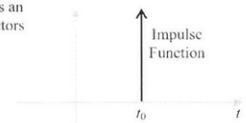
$$\langle R, ? \rangle = 0$$

Galerkin $\langle R, v_m \rangle = 0 \quad m = \cdots$

Actually also collocation can be written as an orthogonality condition with different vectors

$$\langle R, ? \rangle = 0 \qquad \int_0^T \delta(t - t_0) dt = 1$$

$$\int_0^T R(x, t) \underbrace{\delta(t - t_l)}_{impulse} dt = R(x, t_0) = 0$$



$$\delta(t - t_l) = \begin{cases} 0 & t \neq t_l \\ \int_0^T \delta(t - t_l) dt = 1 & t = t_l \end{cases}$$

Galerkin Methods

- Computing Coefficients
  - Linear Galerkin $F(x) = Ax$

$$-\left\{\int_0^T e^{i2\pi m\frac{t}{T}}\left[\sum_{l=-L}^{L} -\frac{i2\pi l}{T}X_l e^{-i2\pi l\frac{t}{T}} - F\left(\sum_{l=-L}^{L} X_l e^{-i2\pi l\frac{t}{T}}\right) - u(t)\right]dt\right\} =$$

$$i2\pi m X_l + \int_0^T e^{i2\pi m\frac{t}{T}} F\left(\sum_{l=-L}^{L} X_l e^{-i2\pi l\frac{t}{T}}\right)dt + \int_0^T e^{i2\pi m\frac{t}{T}} u(t)dt = 0$$

$$m = \{-L,\ldots,0,\ldots,L\}$$

Solve the system using Newton

$$i2\pi m X_l + \int_0^T e^{i2\pi m\frac{t}{T}} A\left(\sum_{l=-L}^{L} X_l e^{-i2\pi l\frac{t}{T}}\right)dt + \underbrace{\int_0^T e^{i2\pi m\frac{t}{T}} u(t)dt}_{TU_m} = 0$$

$$\begin{bmatrix} \frac{i2\pi L}{T} + \mathbf{A} & & & \\ & \frac{i2\pi(L-1)}{T} + \mathbf{A} & & \\ & & \ddots & \\ & & & -\frac{i2\pi L}{T} + \mathbf{A} \end{bmatrix} \begin{bmatrix} X_{-L} \\ X_{-(L-1)} \\ \vdots \\ X_L \end{bmatrix} = -\begin{bmatrix} U_{-L} \\ U_{-(L-1)} \\ \vdots \\ U_L \end{bmatrix}$$

$$j2\pi(-L)x_L + Ax_{-L}T + TU_{-L} = 0$$
$$j2\pi(-(L-1))x_{L-1} + Ax_{-L-1}T + TU_{-(L-1)} = 0$$
$$\frac{j2\pi(-(L-1))}{T}x_{L-1} + Ax_{-L-1} + U_{-(L-1)} = 0$$

Collocation Methods

- Computing Coefficients
  - Collocation – residual zero at test times

$$\underbrace{R(\tilde{X},t)}_{\text{Residual}} = 0 = \sum_{l=-L}^{L} -\frac{i2\pi l}{T}X_l e^{-i2\pi l\frac{t_l}{T}} - F\left(\sum_{l=-L}^{L} X_l e^{-i2\pi l\frac{t_l}{T}}\right) - u(t_l)$$

$$l = \{1,\ldots,2L+1\}$$

$$\underbrace{\begin{bmatrix} e^{i2\pi\frac{L}{T}t_1} & \cdots & e^{-i2\pi\frac{L}{T}t_1} \\ \vdots & & \vdots \\ e^{i2\pi\frac{L}{T}t_{(2L+1)}} & \cdots & e^{-i2\pi\frac{L}{T}t_{(2L+1)}} \end{bmatrix}}_{\substack{\text{What is this?} \\ \text{DFT Matrix} \\ \text{(Converts to time)}}} \begin{bmatrix} \frac{j2\pi L}{T} & & \\ & \ddots & \\ & & -\frac{j\pi L}{T} \end{bmatrix} \begin{bmatrix} X_L \\ \vdots \\ X_{-L} \end{bmatrix} - \begin{bmatrix} F(x(t_1)) \\ \vdots \\ F(x(t_{2L+1})) \end{bmatrix} = \begin{bmatrix} u(t_1) \\ \vdots \\ u(t_{2L+1}) \end{bmatrix}$$

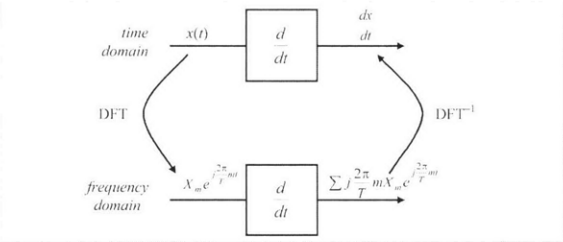
$$\begin{bmatrix} \frac{dx}{dt}(t_1) \\ \frac{dx}{dt}(t_2) \\ \vdots \\ \frac{dx}{dt}(t_{2L+1}) \end{bmatrix}$$

- Discrete Fourier Transform

$$\underbrace{\begin{bmatrix} e^{i2\pi\frac{L}{T}t_1} & \cdots & e^{-i2\pi\frac{L}{T}t_1} \\ \vdots & \ddots & \vdots \\ e^{i2\pi\frac{L}{T}t_{(2L+1)}} & \cdots & e^{-i2\pi\frac{L}{T}t_{(2L+1)}} \end{bmatrix}}_{\text{Discrete Fourier Transform (DFT)}} \begin{bmatrix} X_{-L} \\ X_{-(L-1)} \\ \vdots \\ X_L \end{bmatrix} = \begin{bmatrix} x(t_1) \\ x(t_2) \\ \vdots \\ x(t_{(2L+1)}) \end{bmatrix}$$

$$x(t) \approx \sum_{l=-L}^{L} X_l e^{-i2\pi l\frac{t}{T}}$$

If $t_l = \frac{1}{2L+1}T$ then DFTMatrix has orthogonal columns



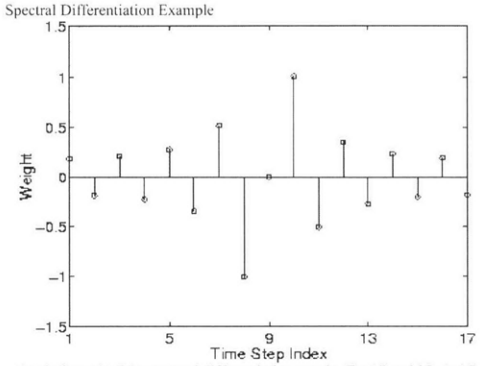
❖ Collocation using Timepoints

$$\underbrace{DFT\begin{bmatrix} \frac{i2\pi L}{T} & & & \\ & \frac{i2\pi(L-1)}{T} & & \\ & & \ddots & \\ & & & -\frac{i2\pi L}{T} \end{bmatrix}(DFT)^{-1}}_{\text{Spectral Differentiation}} \begin{bmatrix} x(t_1) \\ x(t_2) \\ \vdots \\ x(t_{(2L+1)}) \end{bmatrix} - \begin{bmatrix} F(x(t_1)) \\ F(x(t_2)) \\ \vdots \\ F(x(t_{(2L+1)})) \end{bmatrix} = \begin{bmatrix} u(t_1) \\ u(t_2) \\ \vdots \\ u(t_{(2L+1)}) \end{bmatrix}$$

Converting timepoint into Fourier Coeffs, Differentiating, and then returning to time

$$\begin{bmatrix} \frac{dx}{dt}(t_1) \\ \frac{dx}{dt}(t_2) \\ \vdots \\ \frac{dx}{dt}(t_{2L+1}) \end{bmatrix} = DFT\begin{bmatrix} \ddots & & \\ & \ddots & \\ & & \ddots \end{bmatrix} DFT^{-1}\begin{bmatrix} x(t_1) \\ x(t_2) \\ \vdots \\ x(t_{2L+1}) \end{bmatrix}$$

❖ Spectral Differentiation Example



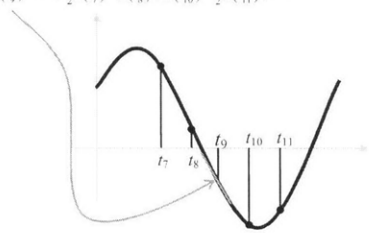
Coef of row 9 of the spectral differentiation matrix, $T = 17$ and $2L+1=17$

Backward Euler

$$\begin{bmatrix} \frac{dx}{dt}(t_1) \\ \vdots \\ \frac{dx}{dt}(t_9) \\ \vdots \\ \frac{dx}{dt}(t_{17}) \end{bmatrix} \approx \begin{bmatrix} & & & & \\ \cdots & -\frac{1}{\Delta t} & \frac{1}{\Delta t} & 0 & \cdots \\ & & & & \end{bmatrix} \begin{bmatrix} x(t_1) \\ \vdots \\ x(t_8) \\ x(t_9) \\ x(t_{10}) \\ \vdots \\ x(t_{17}) \end{bmatrix} \quad \frac{dx}{dt}(t_9) = \frac{-x(t_8)+x(t_9)}{\Delta t}$$

$$\begin{bmatrix} \frac{dx}{dt}(t_1) \\ \vdots \\ \frac{dx}{dt}(t_9) \\ \vdots \\ \frac{dx}{dt}(t_{17}) \end{bmatrix} \approx \underbrace{\begin{bmatrix} & & & & & & \\ \cdots & \frac{1}{2} & -1 & 0 & 1 & -\frac{1}{2} & \cdots \\ & & & & & & \end{bmatrix}}_{DFT \begin{bmatrix} \ddots & & \\ & \ddots & \\ & & \ddots \end{bmatrix} DFT^{-1}} \begin{bmatrix} x(t_1) \\ \vdots \\ x(t_8) \\ x(t_9) \\ x(t_{10}) \\ \vdots \\ x(t_{17}) \end{bmatrix}$$

$$\frac{dx}{dt}(t_9) = \cdots + \tfrac{1}{2}x(t_7) - x(t_8) + x(t_{10}) - \tfrac{1}{2}x(t_{11}) + \cdots$$



Backward Euler $\begin{bmatrix} \frac{1}{\Delta t} & & & & -\frac{1}{\Delta t} \\ -\frac{1}{\Delta t} & \ddots & & & \\ & \ddots & \frac{1}{\Delta t} & & \\ & & \ddots & \ddots & \\ & & & -\frac{1}{\Delta t} & \frac{1}{\Delta t} \end{bmatrix}$

❖ Spectral Collocation versus F-D

Spectral Differentiation:

$$\underbrace{DFT \begin{bmatrix} \frac{i2\pi L}{T} & & & \\ & \frac{i2\pi(L-1)}{T} & & \\ & & \ddots & \\ & & & -\frac{i2\pi L}{T} \end{bmatrix} (DFT)^{-1}}_{\text{Spectral Differentiation}} \begin{bmatrix} x(t_1) \\ x(t_2) \\ \vdots \\ x(t_{(2L+1)}) \end{bmatrix} - \begin{bmatrix} F(x(t_1)) \\ F(x(t_2)) \\ \vdots \\ F(x(t_{(2L+1)})) \end{bmatrix} = \begin{bmatrix} u(t_1) \\ u(t_2) \\ \vdots \\ u(t_{(2L+1)}) \end{bmatrix}$$

Backward Euler Differentiation:

$$\begin{bmatrix} \frac{1}{\Delta t} & & & & & -\frac{1}{\Delta t} \\ -\frac{1}{\Delta t} & \frac{1}{\Delta t} & & & & \\ & -\frac{1}{\Delta t} & \frac{1}{\Delta t} & & & \\ & & \ddots & \ddots & & \\ & & & -\frac{1}{\Delta t} & \frac{1}{\Delta t} & \end{bmatrix} \begin{bmatrix} \hat{x}^1 \\ \hat{x}^2 \\ \vdots \\ \hat{x}^{2L+1} \end{bmatrix} + \begin{bmatrix} F(x(t_1)) \\ F(x(t_2)) \\ \vdots \\ F(x(t_{(2L+1)})) \end{bmatrix} = \begin{bmatrix} u(t_1) \\ u(t_2) \\ \vdots \\ u(t_{(2L+1)}) \end{bmatrix}$$

3.18 PDE - Finite Difference Methods 1D

INTRODUCTION TO NUMERICAL SIMULATION

LECTURE 18.
Finite Difference Methods for Boundary Value Problems

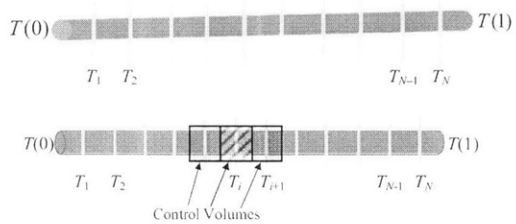
TODAY'S OUTLINE:

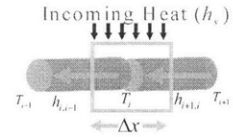
- Informal Finite Difference Methods
  - Heat Conducting Bar
- Comparing FEM and FD in 1-D
- Convergence Analysis for Heat Equation
  - Local Truncation Error and Consistency
  - Global Error and Stability
- Formal Generalization to any PDE
  - Consistency + Stability yields Convergence

INFORMAL FINITE DIFFERENCE METHODS

- Heat Conducting Bar

1. Cut the bar into short sections
2. Assign each cut a temperature





$$h_{i+1,i} = \text{heat flow} = \kappa \frac{T_{i+1} - T_i}{\Delta x}$$

$$h_{i+1,i} - h_{i,i-1} = -h_s \Delta x$$

Heat in from left — Heat out from right — Incoming heat per unit length

Limit as the sections become vanishingly small

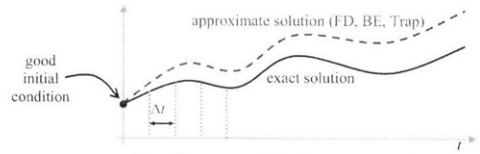
$$\lim_{\Delta x \to 0} -h_s(x) = \frac{\partial h(x)}{\partial x} = \frac{\partial}{\partial x} \kappa \frac{\partial T(x)}{\partial x}$$

Normalized Equation

$$\frac{\partial}{\partial x} \kappa \frac{\partial T(x)}{\partial x} = -h_s \Rightarrow -\frac{\partial^2 u(x)}{\partial x^2} = f(x)$$

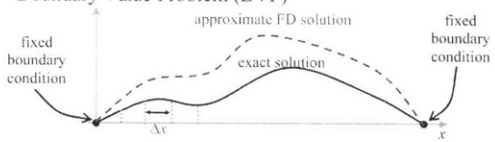
$$-u_{xx}(x) = f(x)$$

Initial Value Problem (IVP)



Errors keep accumulating from step to step forever.

Boundary Value Problem (BVP)

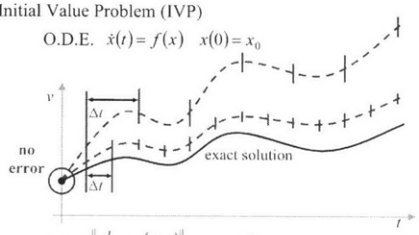


Errors accumulate but they need to decrease back to zero on the other side thanks to boundary conditions (B.C.)
(*This is probably an easier problem!*)

Convergence?

Initial Value Problem (IVP)

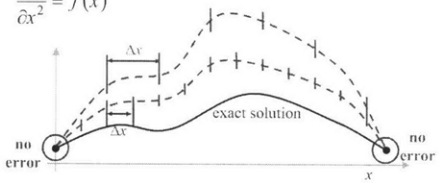
O.D.E. $\dot{x}(t) = f(x) \quad x(0) = x_0$



$$\max_{l \in \left[0, \frac{T}{\Delta t}\right]} \left\| \hat{v}^l - v(l\Delta t) \right\| \xrightarrow[\Delta t]{} 0$$

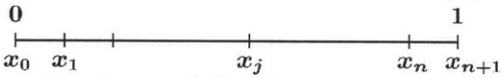
Boundary Value Problem (BVP)

$$\frac{\partial^2 u}{\partial x^2} = f(x)$$



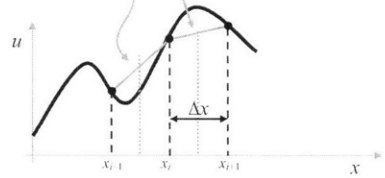
❖ Numerical Solution

Subdivide interval $(0,1)$ into $n+1$ equal subintervals : $\Delta x = \dfrac{1}{n+1}$



$x_j = j\Delta x, \quad \hat{u}_j \approx u_j \equiv u(x_j) \quad \text{for } 0 \le j \le n+1$

$$u_x\left(x_{i-\frac{1}{2}}\right) \cong \frac{u(x_i)-u(x_{i-1})}{\Delta x} \qquad u_x\left(x_{i+\frac{1}{2}}\right) \cong \frac{u(x_{i+1})-u(x_i)}{\Delta x}$$

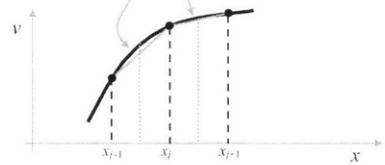


For example…

$$\begin{aligned} v''(x_j) &\approx \frac{1}{\Delta x}\left[v'\left(x_{j+\frac{1}{2}}\right) - v'\left(x_{j-\frac{1}{2}}\right)\right] \\ &\approx \frac{1}{\Delta x}\left[\frac{v_{j+1}-v_j}{\Delta x} - \frac{v_j - v_{j-1}}{\Delta x}\right] \\ &= \frac{v_{j+1}-2v_j+v_{j-1}}{\Delta x^2} \qquad \text{for } \Delta x \text{ small} \end{aligned}$$

$$v''(x_j) \cong \frac{v'\left(x_{j-\frac{1}{2}}\right) - v'\left(x_{j-\frac{1}{2}}\right)}{\Delta x}$$

$$v'\left(x_{j-\frac{1}{2}}\right) \cong \frac{v(x_j)-v(x_{j-1})}{\Delta x} \qquad v'\left(x_{j+\frac{1}{2}}\right) \cong \frac{v(x_{j+1})-v(x_j)}{\Delta x}$$

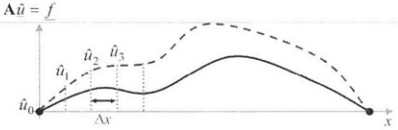


$-u_{xx} = f$ suggests...

$$-\frac{\hat{u}_{j+1} - 2\hat{u}_j + \hat{u}_{j-1}}{\Delta x^2} = f(x_j) \quad 1 \le j \le n$$

$$\hat{u}_0 = \hat{u}_{n+1} = 0$$

$$\Rightarrow \mathbf{A}\underline{\hat{u}} = \underline{f}$$



$$\begin{aligned} -\frac{\hat{u}_2 - 2\hat{u}_1 + \cancelto{0}{\hat{u}_0}}{\Delta x^2} &= f(x_1) \\ -\frac{\hat{u}_3 - 2\hat{u}_2 + \hat{u}_1}{\Delta x^2} &= f(x_2) \\ &\vdots \\ -\frac{\cancelto{0}{\hat{u}_{n+1}} - 2\hat{u}_n + \hat{u}_{n-1}}{\Delta x^2} &= f(x_n) \end{aligned} \qquad \frac{1}{\Delta x^2}\begin{bmatrix} 2 & -1 & & & \\ -1 & 2 & -1 & & \\ & -1 & 2 & \ddots & \\ & & \ddots & \ddots & -1 \\ & & & -1 & 2 \end{bmatrix}\begin{bmatrix} \hat{u}_1 \\ \hat{u}_2 \\ \vdots \\ \hat{u}_{n-1} \\ \hat{u}_n \end{bmatrix} = \begin{bmatrix} f(x_1) \\ f(x_2) \\ \vdots \\ f(x_{n-1}) \\ f(x_n) \end{bmatrix}$$

Initial Value Problem – Backward Euler

$$\hat{x}^l - \hat{x}^{l-1} = \Delta t\left[\lambda \hat{x}^l + u^l\right] \longleftarrow \text{test problem}$$

solve difference equation

no end point to match

Boundary Value Problem – Finite Difference Method

$$-\hat{u}_{j+1} + 2\hat{u}_j - \hat{u}_{j-1} = \Delta x^2 f(x_j)$$

solve difference equation

an end point to match

Backward Euler

$$\hat{x}^1 - \hat{x}^0 = \Delta t\lambda\left[\lambda\hat{x}^1 + u(t_1)\right] \Rightarrow \hat{x}^1 - x(0) = \Delta t\lambda\left[\lambda\hat{x}^1 + u(t_1)\right]$$

$$\underbrace{\begin{bmatrix} 1-\lambda\Delta t & & & \\ -1 & 1-\lambda\Delta t & & \\ & \ddots & \ddots & \\ & & -1 & 1-\lambda\Delta t \end{bmatrix}}_{\text{Lower Triangular}} \begin{bmatrix} \hat{x}^1 \\ \hat{x}^2 \\ \vdots \\ \hat{x}^l \end{bmatrix} = \underbrace{\begin{bmatrix} u(t_1)+x(0) \\ u(t_2) \\ \vdots \\ u(t_l) \end{bmatrix}}_{\text{input}}$$

$$\mathbf{A} = \frac{1}{\Delta x^2}\begin{pmatrix} 2 & -1 & & & \\ -1 & 2 & -1 & & \\ & \ddots & \ddots & \ddots & \\ & & -1 & 2 & -1 \\ & & & -1 & 2 \end{pmatrix}, \quad \hat{u} = \begin{pmatrix} \hat{u}_1 \\ \hat{u}_2 \\ \vdots \\ \hat{u}_{n-1} \\ \hat{u}_n \end{pmatrix}, \quad f = \begin{pmatrix} f(x_1) \\ f(x_2) \\ \vdots \\ f(x_{n-1}) \\ f(x_n) \end{pmatrix}$$

(Symmetric)

$\mathbf{A} \in \Re^{n\times n} \quad \hat{u}, \underline{f} \in \Re^n$

Is $\mathbf{A}$ nonsingular?

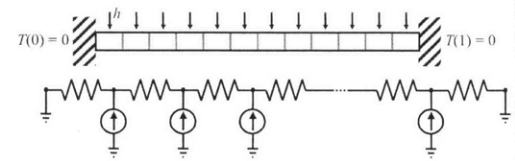
For any $\underline{v} = \{v_1, v_2, \ldots, v_n\}^T$

$$\underline{v}^T\mathbf{A}\underline{v} = \frac{1}{\Delta x^2}\left(v_1^2 + \sum_{i=2}^{n}(v_i - v_{i-1})^2 + v_n^2\right)$$

Hence $\underline{v}^T\mathbf{A}\underline{v} > 0$, for any $\underline{v} \neq 0$ ($\mathbf{A}$ is SPD)

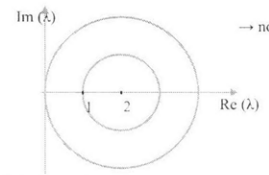
$\mathbf{A}\hat{u} = \underline{f}$: $\hat{u}$ exists and is unique

1.) How to visualize that $\mathbf{A}$ is non-singular.



If I put in some current (=heat) into the network of resistors (=bar) I get a unique set of voltages (=temperatures).

2.) Gershgorin Circle Theorem



→ not useful in this case

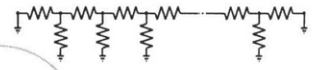
We know

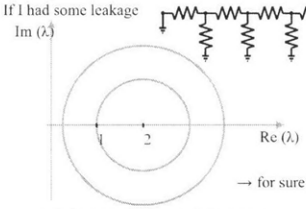
$\lambda(\mathbf{A}) \geq 0$

not

$\lambda(\mathbf{A}) > 0$

If I had some leakage





→ for sure non-singular

Note if $\lambda(\mathbf{A}) > 0 \ \forall\lambda$ positive definite $\Rightarrow$ $\lambda(\mathbf{A}) \neq 0 \ \forall\lambda$ non - singular

Check $v^T\mathbf{A}v$ = sum of squares always positive unless : $\begin{cases} v_1 = 0 \\ v_i = v_{i-1} \ \ \forall i \mapsto v = 0 \\ v_n = 0 \end{cases}$

COMPARING FEM AND FD IN 1-D

- Residual Equation

  Partial Differential Equation Form

$$-\frac{\partial^2 u}{\partial x^2} = f \quad u(0)=0 \quad u(1)=0$$

  Step 1: Choose Basis Functions to represent the solution

$$u(x) \approx u_h(x) = \sum_{i=1}^{n} \omega_i \underbrace{\varphi_i(x)}_{\text{Basis Functions}}$$

  Step 2: Generate equations for the basis functions weights setting residual orthogonal to some test functions

$$\int_0^1 \underbrace{\varphi_l(x)}_{\substack{\text{Test}\\ \text{Functions}}} R(x)dt = 0 \quad R(x) = \sum_{i=1}^{n} \omega_i \frac{d^2\varphi_i(x)}{dx^2} + f(x)$$

- FEM – Basis Weights
  - Galerkin Scheme

    Force the residual to be "orthogonal" to the basis functions

$$\int_0^1 \varphi_l(x) R(x) dt = 0$$

    Generates n equations in n unknowns

$$\int_0^1 \varphi_l(x)\left[\sum_{i=1}^{n} \omega_i \frac{d^2\varphi_i(x)}{dx^2} + f(x)\right]dx = 0 \quad l \in \{1,\ldots,n\}$$

  - Linear System

$$\sum_{i=1}^{n} \omega_i \underbrace{\int_0^1 \frac{d\varphi_l}{dx}\frac{d\varphi_i}{dx}dx}_{A_{l,i}} = \underbrace{\int_0^1 \varphi_l f(x)dx}_{F_l}$$

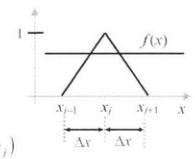
$$\frac{1}{\Delta x}\begin{bmatrix} 2 & -1 & & & \\ -1 & 2 & -1 & & \\ & -1 & 2 & \ddots & \\ & & \ddots & \ddots & -1 \\ & & & -1 & 2 \end{bmatrix}\begin{bmatrix} \omega_1 \\ \omega_2 \\ \vdots \\ \omega_n \end{bmatrix} = \begin{bmatrix} F_1 \\ F_2 \\ \vdots \\ F_n \end{bmatrix}$$

- Comparing – FD & FEM (hat basis) – 1D problem

$$\text{FEM} \quad \frac{1}{\Delta x}\begin{bmatrix} 2 & -1 & & \\ -1 & 2 & \ddots & \\ & \ddots & \ddots & -1 \\ & & -1 & 2 \end{bmatrix}\begin{bmatrix} \omega_1 \\ \omega_2 \\ \vdots \\ \omega_n \end{bmatrix} = \begin{bmatrix} \int_0^1 \varphi_1(x) f(x)dx \\ \int_0^1 \varphi_2(x) f(x)dx \\ \vdots \\ \int_0^1 \varphi_n(x) f(x)dx \end{bmatrix}$$

$$\text{FD} \quad \frac{1}{\Delta x}\begin{bmatrix} 2 & -1 & & \\ -1 & 2 & \ddots & \\ & \ddots & \ddots & -1 \\ & & -1 & 2 \end{bmatrix}\begin{bmatrix} \hat{u}_1 \\ \hat{u}_2 \\ \vdots \\ \hat{u}_n \end{bmatrix} = \begin{bmatrix} f(x_1) \\ f(x_2) \\ \vdots \\ f(x_n) \end{bmatrix}$$

F.D. $\Delta x \cdot f(x_i)$ F.E.M. $\int_0^1 \varphi_j(x) f(x)dx$



If $f(x)$ is constant:

$$\int_0^1 \varphi_j(x) f(x)dx = \frac{1}{2}\cdot 2 \cdot \Delta x \cdot f(x_j)$$

$\rightarrow$ same as F.D.

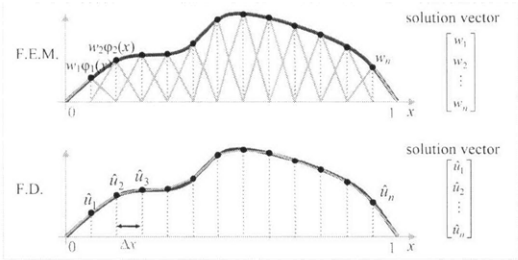
If $f(x)$ is not constant the RHS is still an approximation

$$\int_0^1 \varphi_i(x) f(x)dx \cong \frac{1}{2} = \Delta x \cdot f(x_i)$$

$$\text{F.E.M.} \quad \frac{1}{\Delta x}\begin{bmatrix} 2 & -1 & & \\ -1 & 2 & \ddots & \\ & \ddots & \ddots & -1 \\ & & -1 & 2 \end{bmatrix}\begin{bmatrix} \\ \\ \\ \\ \end{bmatrix} = \Delta x \begin{bmatrix} f(x_1) \\ \vdots \\ \vdots \\ f(x_n) \end{bmatrix}$$

Same as F.D. system for 1D hat basis

However F.E.M. gives more flexibility (can choose other basis)

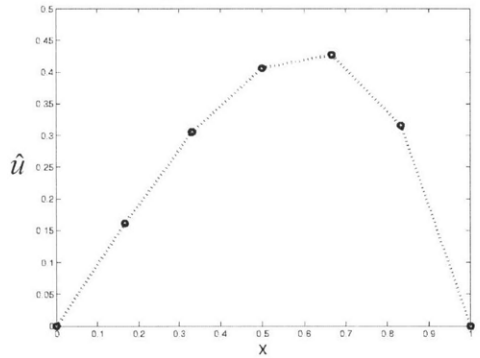


CONVERGENCE ANALYSIS FOR HEAT EQUATION

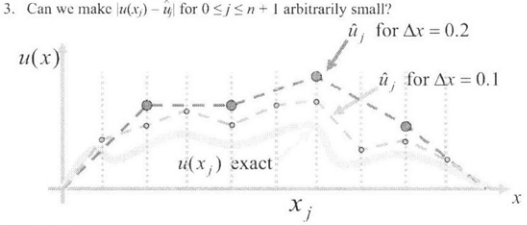
- Local Truncation Error and Consistency
  - Example

$$-u_{xx} = (3x + x^2)e^x, \quad x \in (0,1) \quad \text{with} \quad u(0) = u(1) = 0$$

Take $n = 5$, $\Delta x = 1/6$ …



- Convergence?
  1. Does the discrete solution $\hat{u}$ retain the qualitative properties of the continuous solution $u(x)$?
  2. Does the solution become more accurate when $\Delta x \to 0$?
  3. Can we make $|u(x_j) - \hat{u}_j|$ for $0 \le j \le n+1$ arbitrarily small?



- Truncation Error

For any $v \in C^4$ we can show that

$$\frac{v(x_{j+1}) - 2v(x_j) + v(x_{j-1})}{\Delta x^2} = v''(x_j) + \frac{\Delta x^2}{12} v^{(4)}(x_j + \theta\Delta x) \quad -1 \le \theta \le 1$$

Take $u \equiv v$, $(-u'' = f)$

$$-\frac{u(x_{j+1}) - 2u(x_j) + u(x_{j-1})}{\Delta x^2} = f(x_j) \underbrace{- \frac{\Delta x^2}{12} u^{(4)}(x_j + \theta_j \Delta x)}_{\tau_j}$$

$$e_i = e(x_i) = \underbrace{u(x_i)}_{\substack{\text{exact}\\ \text{solution}}} - \underbrace{\hat{u}_i}_{\substack{\text{F.D.}\\ \text{solution}}}$$

$u(x)$ exact solution solves $-u'' = f$ exact equation

$\hat{u}_i$ F.D. solution solves $-\dfrac{\hat{u}_{i-1} - 2\hat{u}_i + \hat{u}_{i+1}}{\Delta x^2} = f_i$ F.D. equation

Local truncation error (LTE) tells me how well the exact solution solves the F.D. equation.

$$-\left[\frac{u(x_{i-1}) - 2u(x_i) + u(x_{i+1})}{\Delta x^2}\right] - f_i = \tau_i \qquad \text{LTE}$$

hopefully small!!

How do we calculate the L.T.E. τ_i?

Use: Taylor Series

$$\frac{u(x_{j+1})-2u(x_j)+u(x_{j-1})}{\Delta x^2} \cong \frac{\partial^2 u}{\partial x^2}$$

$$u(x_{j+1}) = u(x_j)+\Delta x\cdot u'(x_j)+\frac{\Delta x^2}{2}u''(x_j)+\frac{\Delta x^3}{3!}u'''(x_j)+\frac{\Delta x^4}{4!}u^{(4)}(x)\Big|_{x\in[x_{j-1},x_{j+1}]}$$

$$-2u(x_j) = -2u(x_j)$$

$$u(x_{j-1}) = u(x_j)-\Delta x\cdot u'(x_j)+\frac{\Delta x^2}{2}u''(x_j)-\frac{\Delta x^3}{3!}u'''(x_j)+\frac{\Delta x^4}{4!}u^{(4)}(x)\Big|_{x\in[x_{j-1},x_{j+1}]}$$

$$u(x_{j+1})-2u(x_j)+u(x_{j-1}) = \Delta x^2 u''(x_j)+\frac{\Delta x^4}{12}u^{(4)}(x)_{x\in[x_{j-1},x_{j+1}]}$$

Hence $$u''(x_j) = \frac{u(x_{j+1})-2u(x_j)+u(x_{j-1})}{\Delta x^2}+\underbrace{\frac{\Delta x^2}{12}u^{(4)}(x)\Big|_{x\in[x_{j-1},x_{j+1}]}}_{\text{Approximation error}}$$

Substitute back and get τ_j L.T.E.

$$-\left[u''(x_j)-\frac{\Delta x^2}{12}u^{(4)}(x)_{x\in[x_{j-1},x_{j+1}]}\right]-f_j=\tau_j \Rightarrow \tau_j=\frac{\Delta x^2}{12}u^{(4)}(x)\Big|_{x\in[x_{j-1},x_{j+1}]}$$

Since $u''=f$

- Global Error and Stability
  - Discretization Error Analysis – Error Equation

    Let $e_j = u(x_j)-\hat{u}_j$ be the discretization error.

$$-\frac{u(x_{j+1})-2u(x_j)+u(x_{j-1})}{\Delta x^2}=f(x_j)+\tau_j$$

$$-\frac{\hat{u}_{j+1}-2\hat{u}_j+\hat{u}_{j-1}}{\Delta x^2}=f(x_j)$$

Subtracting

$$-\frac{e_{j+1}-2e_j+e_{j+1}}{\Delta x^2}=\tau_j \quad 1\le j\le n$$

and $e_0=e_{n+1}=0$

What we really want is the global error

$e_j = u(x_j)-\hat{u}_j$

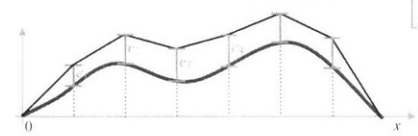
$$-\frac{u(x_{j-1})-2u(x_j)+u(x_{j+1})}{\Delta x^2}=f(x_j)+\underbrace{\tau_j}_{\text{LTE}}$$ Exact Solution into F.D. Equation

$$-\frac{\hat{u}_{j-1}-2\hat{u}_j+\hat{u}_{j+1}}{\Delta x^2}=f(x_j)$$ F.D. Solution solves F.D. Equation

$$-\frac{e_{j-1}-2e_j+e_{j+1}}{\Delta x^2}=\tau_j$$ Subtract and get a F.D. Equation for the Global Error

$e_0=u(x_0)-\hat{u}_0=0$ B.C.

This generates a linear system that can be solved for $\begin{bmatrix} e_1 \\ e_2 \\ \vdots \\ e_n \end{bmatrix}$



$\mathbf{A}e=\tau$

$$e=\begin{bmatrix} e_1 \\ e_2 \\ \vdots \\ e_N \end{bmatrix},\quad \tau=\frac{\Delta x^2}{12}\begin{bmatrix} u^{(4)}(x_1+\theta_1\Delta x) \\ u^{(4)}(x_2+\theta_2\Delta x) \\ \vdots \\ u^{(4)}(x_N+\theta_N\Delta x) \end{bmatrix}$$

- Properties of $\mathbf{A}^{-1}$

  Let $\mathbf{A}^{-1}=\{\alpha_{ij}\}_{1\le i,j\le n}$

  - Non-negativity $\alpha_{ij}\ge 0$, for $1\le i,j\le n$
  - Boundedness $0\le\sum_{j=1}^{N}\alpha_{ij}\le\frac{1}{8}$, for $1\le i\le n$

$e = \mathbf{A}^{-1}\tau$

$\mathbf{A}^{-1} = \{\alpha_{ij}\}$ α_{ij} are all positive... WHY?

How do we visualize the inverse of a matrix?

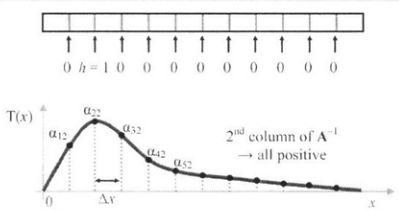
$$\mathbf{A}\begin{bmatrix} T_1 \\ T_2 \\ \vdots \\ T_n \end{bmatrix} = \begin{bmatrix} h_1 \\ h_2 \\ \vdots \\ h_n \end{bmatrix} \qquad \mathbf{A}^{-1}\begin{bmatrix} h_1 \\ h_2 \\ \vdots \\ h_n \end{bmatrix} = \begin{bmatrix} T_1 \\ T_2 \\ \vdots \\ T_n \end{bmatrix}$$

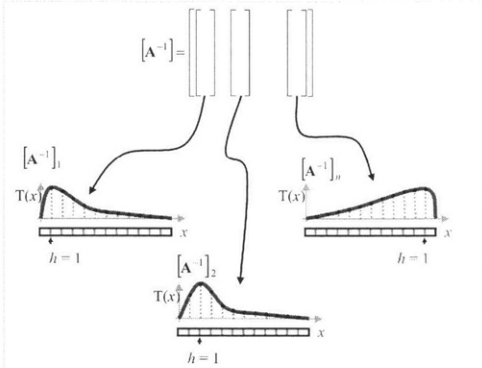
How do we get column j of $\mathbf{A}^{-1}$?

Just pick an appropriate set of heat sources.

e.g.

$$\begin{bmatrix} h_1 \\ \vdots \\ h_j \\ \vdots \\ h_n \end{bmatrix} = \begin{bmatrix} 0 \\ \vdots \\ 0 \\ 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \longleftarrow j \Rightarrow \left[\mathbf{A}^{-1}\right]_j = \begin{bmatrix} T_1 \\ T_2 \\ \vdots \\ T_n \end{bmatrix}$$





- Qualitative Properties of $\hat{u}$

  $f \geq 0 \rightarrow \hat{u} \geq 0$

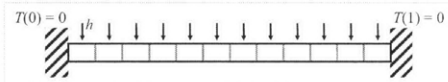
  $\underline{\hat{u}} = \mathbf{A}^{-1}\underline{f}$

  If $f_j = f(x_j) \geq 0$, for $1 \leq j \leq n$

  Then $\hat{u}_i = \sum_j \alpha_{ij} f_j \geq 0$, for $1 \leq i \leq n$

$\hat{u}$ is just a linear combination of columns of $\mathbf{A}^{-1}$

The columns are all positive if the coefficients are positive
→ the combination is positive.



$$\mathbf{A}\begin{bmatrix} T_1 \\ \vdots \\ T_n \end{bmatrix} = f \Rightarrow \begin{bmatrix} T_1 \\ \vdots \\ T_n \end{bmatrix} = \mathbf{A}^{-1} f \qquad \bar{e}_1 \qquad \Rightarrow \begin{bmatrix} T_1 \\ \vdots \\ T_n \end{bmatrix} = \mathbf{A}^{-1}\bar{e}_1$$

Discrete Stability

$\underline{\hat{u}} = \mathbf{A}^{-1}\underline{f}$

$$\|\hat{u}\|_\infty = \max_i \hat{u}_i| = \max_i \left|\sum_j \alpha_{ij} f_j\right| \le \max_i \left(\sum_j \alpha_{ij}\right) \max_i |f_i| \le \frac{1}{8}\|\underline{f}\|_\infty$$

- Convergence

  Using the discrete stability estimate on $\mathbf{A}\underline{e} = \underline{\tau}$

$$\|e\|_\infty \le \frac{1}{8}\|\tau\|_\infty$$

or

$$\max_{1\le i\le n} |u(x_i) - \hat{u}_i| \le \frac{\Delta x^2}{96} \max_{0\le x\le 1} |u^{(4)}(x)|$$

A- priori Error Estimate

$$\mathbf{A}e = \tau$$

$$e = \mathbf{A}^{-1}\tau$$

Global Error

Local Truncation Error

$$\|\tau\|_\infty \le \frac{\Delta x^2}{12} u^{(4)}(x)$$

bounded

$$\|e\|_\infty = \max_{i\in[1,n]} |e_i| = \|\mathbf{A}^{-1}\tau\|_\infty \le \underbrace{\|\mathbf{A}^{-1}\|_\infty}_{\text{Stability}} \underbrace{\|\tau\|_\infty}_{\text{L.T.E.}}$$

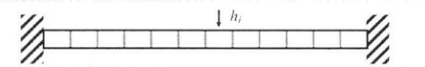
$$\|\mathbf{A}^{-1}\|_\infty = \max \frac{|\mathbf{A}^{-1}x|_\infty}{|x|_\infty} = \max_{\|x\|_\infty=1} \|\mathbf{A}^{-1}x\|_\infty = \max_{\text{all rows } i} \sum_j |\alpha_{ij}| \qquad \text{max row sum}$$

$$\left\|\mathbf{A}\underbrace{\begin{bmatrix}\pm 1\\ \pm 1\\ \vdots\\ \pm 1\end{bmatrix}}_{\|\mathbf{1}\|_\infty = 1}\right\|_\infty \qquad \mathbf{A} = \frac{1}{\Delta x^2}\begin{bmatrix} 2 & -1 & & \\ -1 & 2 & \ddots & \\ & \ddots & \ddots & -1 \\ & & -1 & 2\end{bmatrix}$$

We know $\mathbf{A}^{-1}$ has all positive entries

What x should we pick to maximize $\|\mathbf{A}^{-1}x\|_\infty$? $\quad x = \begin{bmatrix}1\\1\\\vdots\\1\end{bmatrix}$

Can look at max column sum since $\mathbf{A}$, $\mathbf{A}^{-1}$ symmetric



$$\|\vec{A}_i^{-1}\|_1 = \|\mathbf{A}^{-1}\vec{e}_i\|_1$$

$$\frac{1}{\Delta x^2}\begin{bmatrix} 2 & -1 & & \\ -1 & 2 & \ddots & \\ & \ddots & \ddots & -1 \\ & & -1 & 2\end{bmatrix}\begin{bmatrix}T_1\\T_2\\\vdots\\T_n\end{bmatrix} = \begin{bmatrix}0\\\vdots\\0\\1\\0\\\vdots\\0\end{bmatrix} \leftarrow h_i$$

$$\Rightarrow \begin{bmatrix}T_1\\T_2\\\vdots\\T_n\end{bmatrix} = \Delta x^2 \begin{bmatrix} 2 & -1 & & \\ -1 & 2 & \ddots & \\ & \ddots & \ddots & -1 \\ & & -1 & 2\end{bmatrix}^{-1} \begin{bmatrix}0\\\vdots\\0\\1\\0\\\vdots\\0\end{bmatrix} \Rightarrow \text{column } i^{th} \text{ of } \Delta x^2 \begin{bmatrix} 2 & -1 & & \\ -1 & 2 & \ddots & \\ & \ddots & \ddots & -1 \\ & & -1 & 2\end{bmatrix}^{-1}$$

	Lossy	Insulated
LTE	same	same
$\|\mathbf{A}^{-1}\|$	smaller	larger

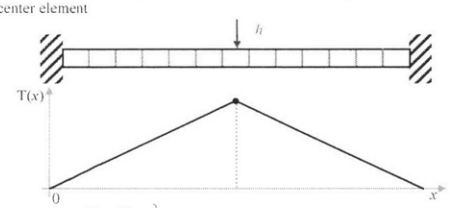
$$\mathbf{A} = \frac{1}{\Delta x^2}\begin{bmatrix} 2 & -1 & & \\ -1 & 2 & \ddots & \\ & \ddots & \ddots & -1 \\ & & -1 & 2\end{bmatrix} = \frac{1}{(n+1)^2}\begin{bmatrix} 2 & -1 & & \\ -1 & 2 & \ddots & \\ & \ddots & \ddots & -1 \\ & & -1 & 2\end{bmatrix}$$

$$\mathbf{A}^{-1} = (n+1)^2 \begin{bmatrix} 2 & -1 & & \\ -1 & 2 & \ddots & \\ & \ddots & \ddots & -1 \\ & & -1 & 2\end{bmatrix}^{-1} \qquad \left\|\begin{bmatrix} 2 & -1 & & \\ -1 & 2 & \ddots & \\ & \ddots & \ddots & -1 \\ & & -1 & 2\end{bmatrix}^{-1}\right\|_\infty$$

$$\|\mathbf{A}^{-1}\|_\infty = \max \frac{\|\mathbf{A}^{-1}x\|_\infty}{\|x\|_\infty} = \max_{\text{all rows } i} \sum_j |\alpha_{ij}| \qquad \text{max row sum}$$

$$x = \begin{bmatrix}\pm 1\\\pm 1\\\vdots\\\pm 1\end{bmatrix}$$

Maximum sum of temperatures will occur when heat is put into the center element

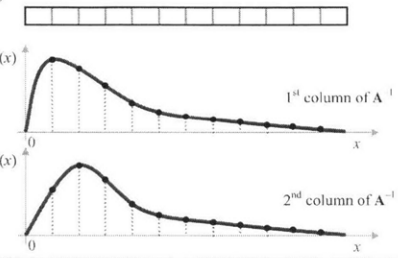


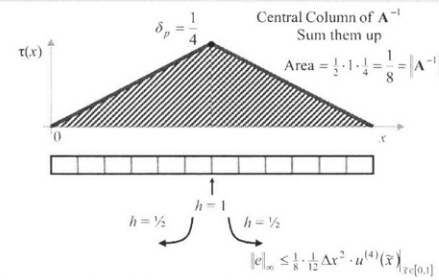
$$\left.\begin{array}{l} T_1 = \tfrac{1}{2} \\ T_2 = \tfrac{2}{2} \\ T_3 = \tfrac{3}{2} \\ \vdots \\ T_{n/2} = \tfrac{n}{(2\cdot 2)} \\ \vdots \\ T_n = \tfrac{1}{2} \end{array}\right\} \sum_{i=1}^{n} T_i = 2\cdot\frac{1+2+\ldots+\frac{n}{2}}{2} = \frac{\frac{n}{2}\left(\frac{n}{2}+1\right)}{2} \approx \frac{n^2}{8}$$

$$\left\| \begin{bmatrix} 2 & -1 & & \\ -1 & 2 & \ddots & \\ & \ddots & \ddots & -1 \\ & & -1 & 2 \end{bmatrix}^{-1} \right\|_\infty \le \frac{n^2}{8}$$

$$\mathbf{A}^{-1} = (n+1)^2 \begin{bmatrix} 2 & -1 & & \\ -1 & 2 & \ddots & \\ & \ddots & \ddots & -1 \\ & & -1 & 2 \end{bmatrix}^{-1} \le \frac{1}{8}$$

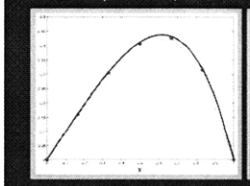
$\frac{1}{8}$ bound on $\|\mathbf{A}^{-1}\|_\infty$



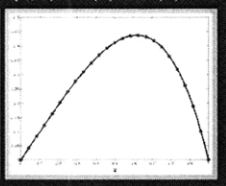


- Numerical Example

$$-u_{xx} = (3x + x^2)e^x, \quad x \in (0,1), \quad u(0) = u(1) = 0$$



$\Delta x = 1/6$



$\Delta x = 1/24$

$n+1$	$\|\underline{u} - \hat{\underline{u}}\|_\infty$
3	0.0227
6	0.0059
12	0.0015
24	$3.756e-04$
48	$9.404e-05$
96	$2.350e-05$
192	$5.876e-06$

Asymptotically,

$$\|\underline{u} - \hat{\underline{u}}\|_\infty \approx C\Delta x^\alpha$$

$C = 0.216623$

$\alpha = 2.000$

- Summary
  - For a simple model problem we can produce numerical approximations of **arbitrary accuracy**.
  - An **a-priori error estimate** gives the asymptotic dependence of the solution error on the discretization size Δx.

FORMAL GENERALIZATION TO ANY PDE

- Definitions

Consider a linear elliptic differential equation $\mathcal{L}u = f$

and a difference scheme $\hat{\mathcal{L}}\hat{u} = \hat{f}$

e.g. Different PDE

$\mathcal{L}u = f \quad -\dfrac{\partial^2 u}{\partial x^2} + \alpha u(x) = f(x)$ Helmholtz

$\hat{\mathcal{L}}\hat{u} = \hat{f} \quad -\dfrac{\hat{u}_{j-1} - 2\hat{u}_j + \hat{u}_{j+1}}{\Delta x^2} + \alpha\hat{u}_j = f_j$ F.D. eq on F.D. solution $\hat{u}$

- Consistency + Stability yields Convergence
  - Consistency

The difference scheme is **consistent** with the differential equation if:

For **all** smooth functions v

$$\left(\hat{\mathcal{L}}\underline{v} - \hat{\underline{f}}\right)_j \to 0, \quad \text{for} \quad j = 1,\ldots,n \quad \text{when } \Delta x \to 0.$$

$$\left(\hat{\mathcal{L}}\underline{v} - \hat{\underline{f}}\right)_j \to 0, \quad \text{for} \quad j = 1,\ldots,n \quad \text{when } \Delta x \to 0.$$

$$\left(\hat{\mathcal{L}}\underline{v} - \hat{\underline{f}}\right)_j = \mathcal{O}\left(\Delta x^p\right) \quad \text{for all } j \quad \Rightarrow p \text{ is } \textbf{order of accuracy}$$

Consistency LTE $\tau \to 0$ if $\Delta x \to 0$

$\hat{\mathcal{L}}u - \hat{f} = \tau$ Exact solution u into F.D. equation

$$-\frac{u(x_{j-1}) - 2u(x_j) + u(x_{j+1})}{\Delta x^2} + \alpha u(x_j) - f(x_j) = \tau_j$$

In this case note $\tau_j = \dfrac{1}{12}\Delta x^2 u^{(4)}(x)\Big|_{x\in[\ ,\ ]}$

As for Poisson Eq., $p = 2$

  - Truncation Error

$$\left(\hat{\mathcal{L}}\underline{u} - \hat{\underline{f}}\right)_j = \tau_j, \quad \text{for} \quad j = 1,\ldots,n \quad \text{or,} \quad \hat{\mathcal{L}}\underline{u} - \hat{\underline{f}} = \underline{\tau}.$$

The truncation error results from inserting the exact solution into the difference scheme.

Consistency $\Rightarrow \|\tau\|_\infty = \mathcal{O}\left(\Delta x^p\right)$

  - Error Equation

Original Scheme $\hat{\mathcal{L}}\hat{\underline{u}} = \hat{\underline{f}}$

Consistency $\hat{\mathcal{L}}u = \hat{\underline{f}} + \tau$

The error $e = u - \hat{u}$ satisfies $\hat{\mathcal{L}}e = \tau$

$\hat{\mathcal{L}}e = \tau \qquad e = u - \hat{u}$

$-\dfrac{u(x_{j-1}) - 2u(x_j) + u(x_{j+1})}{\Delta x^2} + \alpha u(x_j) - f(x_j) = \tau_j$	Exact Solution into F.D. Equation
$-\dfrac{\hat{u}_{j-1} - 2\hat{u}_j + \hat{u}_{j+1}}{\Delta x^2} + \alpha\hat{u}_j - f_j = 0$	F.D. Solution $\hat{u}$ into F.D. Equation
$-\dfrac{e_{j-1} - 2e_j + e_{j+1}}{\Delta x^2} + \alpha e_j = \tau_j$	Subtract

- Stability

Matrix Norm $\|\mathbf{M}\|_\infty = \sup_{v \in \Re^n} \frac{\|\mathbf{M}v\|_\infty}{\|v\|_\infty}$

The difference scheme is **stable** if $\|\hat{\mathcal{L}}^{-1}\|_\infty \le C$ (independent of Δx)

$$\hat{\mathcal{L}}_{\text{Helm}}\hat{u} = \left(\alpha \begin{bmatrix} 1 & & & \\ & 1 & & \\ & & \ddots & \\ & & & 1 \end{bmatrix} + \frac{1}{(\Delta x)^2} \begin{bmatrix} 2 & -1 & & \\ -1 & 2 & \ddots & \\ & \ddots & \ddots & -1 \\ & & -1 & 2 \end{bmatrix} \right) \begin{bmatrix} \hat{u}_1 \\ \hat{u}_1 \\ \vdots \\ \hat{u}_n \end{bmatrix}$$

$$\hat{f}e = \tau$$
$$e = \left[\hat{f}^{-1}\right]\tau$$

$$\hat{f} = \begin{bmatrix} 2+* & -1 & & & \\ -1 & 2+* & -1 & & \\ & -1 & 2+* & \ddots & \\ & & \ddots & \ddots & -1 \\ & & & -1 & 2+* \end{bmatrix} \qquad \begin{matrix} \|\mathbf{A}^{-1}\|_\infty < \frac{1}{8} \\ \|\hat{f}^{-1}\|_\infty < C \end{matrix}$$

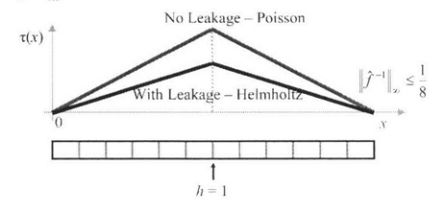
$$\|\mathbf{M}\|_\infty = \sup_{\|v\|_\infty = 1} \|\mathbf{M}v\|_\infty = \sup_{\|v\|_\infty = 1} \left(\max_i \sum_{j=1}^n m_{ij} v_j \right) \qquad v_j = \text{sign}(m_{ij})$$
$$= \max_i \left(\sup_{\|v\|_\infty = 1} \left| \sum_{j=1}^n m_{ij} v_j \right| \right) = \max_i \sum_{j=1}^n |m_{ij}| \qquad \textbf{(max row sum)}$$

- Convergence

Error Equation $e = \hat{\mathcal{L}}^{-1}\tau$

Taking Norms $\|e\|_\infty = \|\hat{\mathcal{L}}^{-1}\tau\|_\infty \le \|\hat{\mathcal{L}}^{-1}\|_\infty \|\tau\|_\infty \le \underbrace{\|\hat{\mathcal{L}}^{-1}\|_\infty C}_{C_1} \Delta x^p = C_1 \Delta x^p$

$\|\hat{f}^{-1}\|_\infty \le ?$



$\|\hat{f}^{-1}\|_\infty \le \frac{1}{8}$

- Summary

Consistency + Stability ⇨ Convergence

Convergence		Stability		Consistency
$\|e\|_\infty$	$\le$	$\|\hat{\mathcal{L}}^{-1}\|_\infty$	$\cdot$	$\|\tau\|_\infty$

I.V.P.	B.V.P.
Consistency + Stability L.T.E. $= O(\Delta x^p)$ $p > 1$	Consistency + Stability L.T.E. $= O(\Delta x^p)$ $p > 0$, $\|\hat{f}^{-1}\| < c$
$\|e\| \le \kappa(\Delta x)^{p-1}$ Global Error (we loose one order)	$\|e\| \le \kappa(\Delta x)^p$ We do not loose one order
	Problem easier!

3.19 PDE - Finite Difference Methods 3D

INTRODUCTION TO NUMERICAL SIMULATION

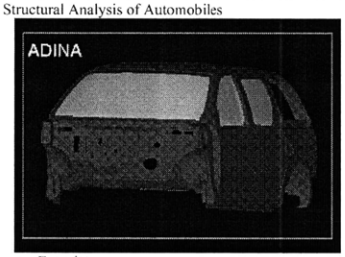
LECTURE 19.
Boundary Value Problems – Solving 3-D Finite-Difference Problems

TODAY'S OUTLINE:

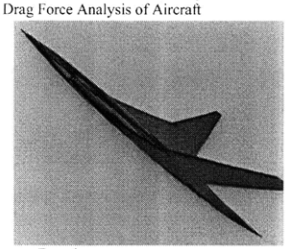
- Finite Difference Matrices in 1-D, 2-D, and 3-D
- Gaussian Elimination Cost:
  - Bandlimited GE
  - Sparse GE
- Krylov Method Cost
  - Counting iterations: Communication Lower Bound

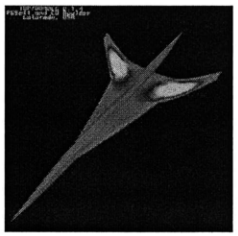
FINITE DIFFERENCE MATRICES IN 1-D, 2-D, AND 3-D

- Structural Analysis of Automobiles

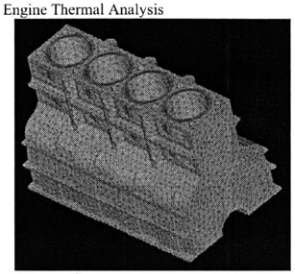


  - Equations
    - Force-displacement relationships for mechanical elements (beams, plates, shells) and sum of forces = 0.
    - Partial Differential Equations of Continuum Mechanics
- Drag Force Analysis of Aircraft

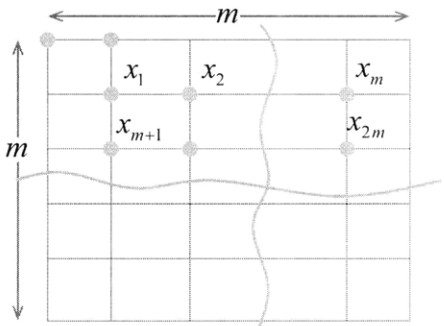




  - Equations
    - Navier-Stokes Partial Differential Equations.
- Engine Thermal Analysis

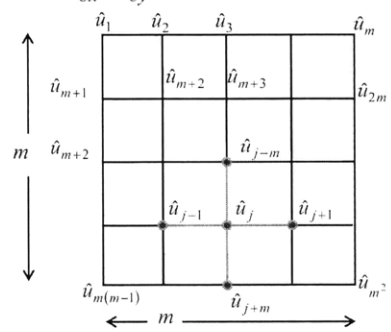


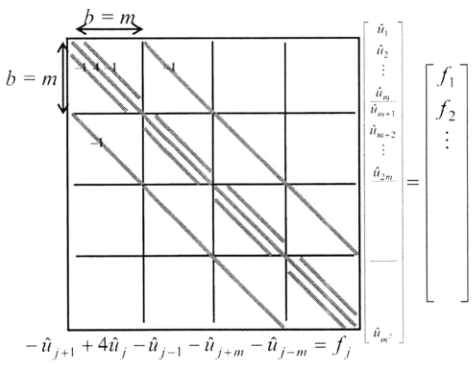
  - Equations
    - The Poisson Partial Differential Equation
- FD Matrix Properties
  - 2-D Discretized Problem
    - Discretized Poisson



$$\underbrace{-\frac{u_{j+1}-2u_j+u_{j-1}}{\Delta x^2}}_{u_{xx}}\underbrace{-\frac{u_{j+m}-2u_j+u_{j-m}}{\Delta y^2}}_{u_{yy}}=f(x_j)$$

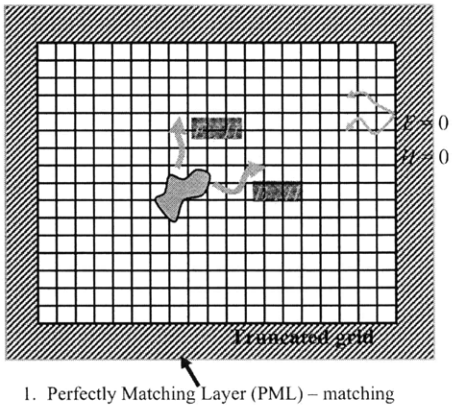
$$2D: \quad -\frac{\partial^2 u}{\partial x^2} - \frac{\partial^2 u}{\partial y^2} = f(x) \qquad \Delta x = \Delta y = 1$$





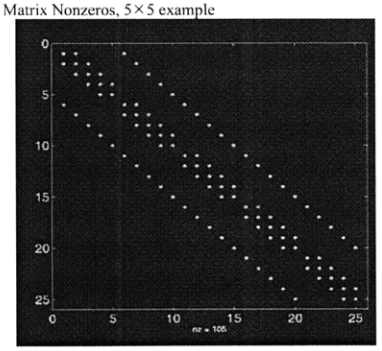
$$-\hat{u}_{j+1} + 4\hat{u}_j - \hat{u}_{j-1} - \hat{u}_{j+m} - \hat{u}_{j-m} = f_j$$

$$\#\ nonzero\ entries = O(5n) = O(5m^2)$$



1. Perfectly Matching Layer (PML) – matching impedence
2. Absorbing Boundary Condition
3. Changing loss & PML properties gradually

– Matrix Nonzeros, 5×5 example

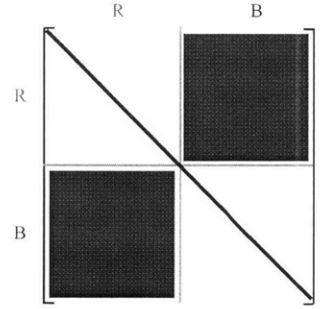


Red – Black Ordering

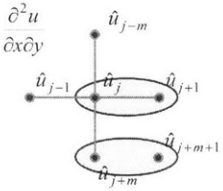
versus "Natural" Ordering

1	33	2	34	3	35	4	36
37	5	38	6	39	7	40	8
9	41	10	42	11	43	12	44
45	13	46	14	47	15	48	16
17	49	18	50	19	51	20	52
53	21	54	22	55	23	56	24
25	57	26	58	27	59	28	60
61	29	62	30	63	31	64	32

Red – Black Ordering

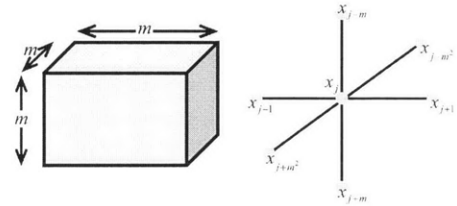


$$\frac{\partial^2 u}{\partial x \partial y}$$



$$\frac{\partial^2 u}{\partial x \partial y} \approx \frac{\dfrac{\hat{u}_{j+1} - \hat{u}_j}{\Delta x} - \dfrac{\hat{u}_{j+m+1} - \hat{u}_{j+m}}{\Delta x}}{\Delta y}$$

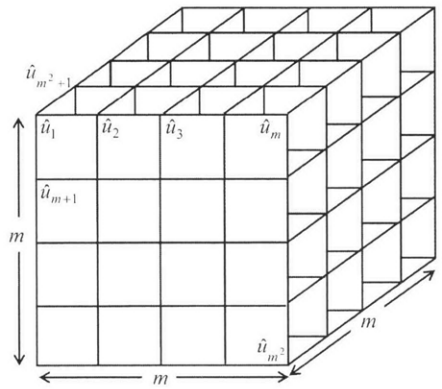
- 3-D Discretized Problem
  - Discretized Poisson

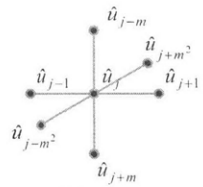


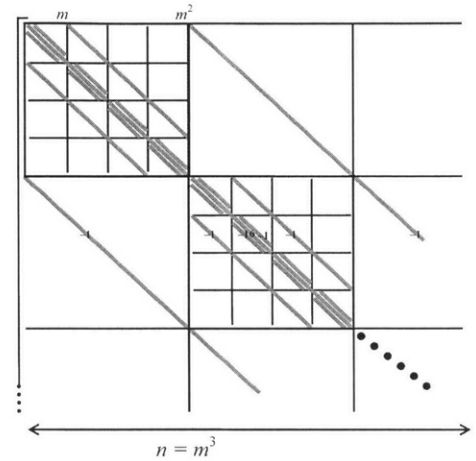
$$\underbrace{-\frac{\hat{u}_{j+1} - 2\hat{u}_j + \hat{u}_{j-1}}{\Delta x^2}}_{u_{xx}} \underbrace{-\frac{\hat{u}_{j+m} - 2\hat{u}_j + \hat{u}_{j-m}}{\Delta y^2}}_{u_{yy}} \underbrace{-\frac{\hat{u}_{j+m^2} - 2\hat{u}_j + \hat{u}_{j-m^2}}{\Delta z^2}}_{u_{zz}} = f(x_j)$$

FD 3D Domain Grid

3D: $-\frac{\partial^2 u}{\partial x^2} - \frac{\partial^2 u}{\partial y^2} - \frac{\partial^2 u}{\partial z^2} = f(x)$





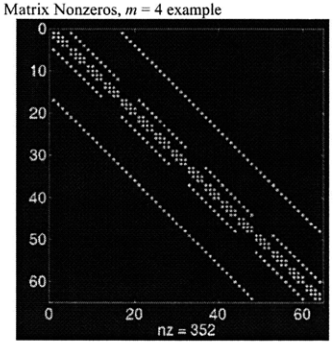


$$-\hat{u}_{j+1} + 6\hat{u}_j - \hat{u}_{j-1} - \hat{u}_{j+m} - \hat{u}_{j-m} - \hat{u}_{j+m^2} - \hat{u}_{j-m^2} = f_j$$

 # nonzero entries $= O(7n) = O(7m^3)$

1D: $b = 1$ $\quad n = m$
2D: $b = m$ $\quad n = m^2$
3D: $b = m^2$ $\quad n = m^3$

- Matrix Nonzeros, $m = 4$ example



- Summary
  - Numerical Properties

    Matrix is irreducibly diagonally dominant

    $$\left(|A_{ii}| \geq \sum_{j \neq i} |A_{ij}| \right)$$

    Matrix is symmetric positive definite

    Assuming uniform discretization, diagonal is

    $$1-D \rightarrow \frac{2}{\Delta^2}, \quad 2-D \rightarrow \frac{4}{\Delta^2}, \quad 3-D \rightarrow \frac{6}{\Delta^2},$$
  - Structural Properties

    Matrices in 3D are LARGE

    $$1-D \rightarrow m \times m, \quad 2-D \rightarrow m^2 \times m^2, \quad 3-D \rightarrow m^3 \times m^3$$ 10

    $0 \times 100 \times 100$ grid in 3D = 1 million × 1 million matrix

    Matrices are very sparse

    Nonzeros per row 1-D: 3, 2-D: 5, 3-D: 7

    Matrices are banded

    $$1-D \quad A_i = 0 \quad |i-j| > 1 = b$$

    $$2-D \quad A_{ij} = 0 \quad i-j > m = b$$

    $$3-D \quad A_{ij} = 0 \quad i-j| > m^2 = b$$

GAUSSIAN ELIMINATION COST

- Bandlimited GE
  - Dense GE
    - Picture

$$\begin{bmatrix} A_{11} & A_{12} & A_{13} & A_{14} \\ 0 & A_{22} & A_{23} & A_{24} \\ 0 & 0 & A_{33} & A_{34} \\ 0 & 0 & 0 & A_{44} \end{bmatrix}$$

- Algorithm

  For $i = 1$ to $n-1$ { "For each row"

  For $j = i+1$ to n { "For each row below pivot"

  Form n-1 reciprocals (pivots)

  $\widetilde{A}_{ji} \leftarrow \frac{A_{ji}}{A_{ii}}$ Form $\sum_{i=1}^{n-1}(n-i) = \frac{n^2}{2}$ multipliers

  For $k = i+1$ to n { "For each element beyond Pivot"

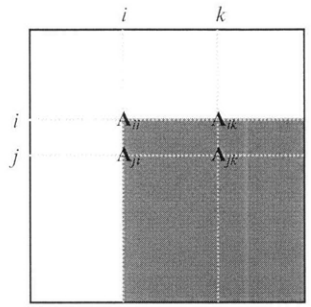
  $\widetilde{A}_{jk} \leftarrow A_{jk} - \widetilde{A}_{ji} A_{ik}$

  Perform $\sum_{i=1}^{n-1}(n-i)^2 \approx \frac{2}{3} n^3$ multiply-adds

  }

  }

  }



- Complexity

1-D	$O(n^3) = O(m^3)$	← 100 pt grid	$O(10^6)$ ops	10 msec
2-D	$O(n^3) = O(m^6)$	← 100×100 grid	$O(10^6)$ ops	2h 46min
3-D	$O(n^3) = O(m^9)$	← 100×100×100 grid	$O(10^{18})$ ops	317 years

e.g. on a 1 GFlops computer using $m = 100$ and a constant $c = 10$

For 2-D and 3-D problems – need a faster solver!

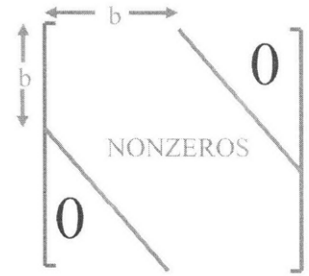
- Banded GE
  - Triangularizing Algorithm

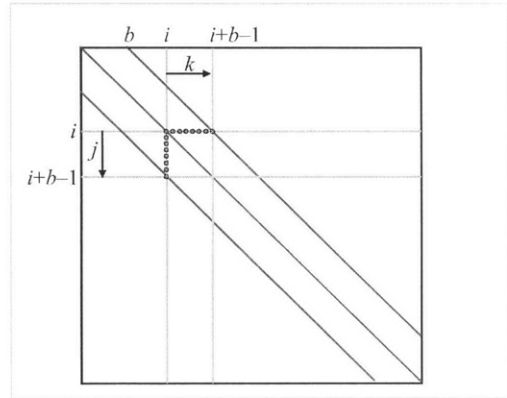
For $i = 1$ to $n - 1$ {
  For $j = i + 1$ to $i + b - 1$ {
    For $k = i + 1$ to $i + b - 1$ {

$$A_{jk} \leftarrow A_{jk} - \frac{A_{ji}}{A_{ii}} A_{ik}$$

    }
  }
}

Perform $\sum_{i=1}^{n-1} [\min(b-1, n-i)]^2 \approx O(b^2 n)$ Multiply - Adds





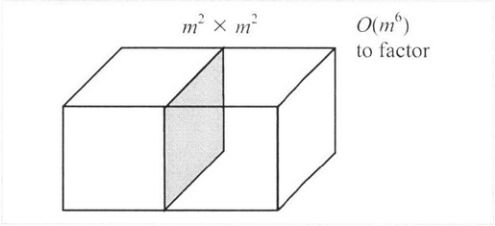
- Complexity

	matrix size	band
$1-D$	$n = m$	$b = 1$
$2-D$	$n = m^2$	$b = m$
$3-D$	$n = m^3$	$b = m^2$

domain size $= m$

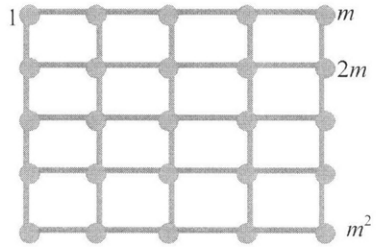
1-D	$O(b^2 n) = O(m)$	← 100 pt grid	$O(100)$ ops	1 μsec
2-D	$O(b^2 n) = O(m^4)$	← 100×100 grid	$O(10^8)$ ops	1 sec

3-D $O(b^2n) = O(m^7) \leftarrow 100 \times 100 \times 100$ grid $O(10^{14})$ ops 11d 13h
e.g. on a 1 GFlops computer using $m = 100$ and a constant $c = 10$
For 3-D problems – still need a faster solver!



- Sparse GE
  - Matrix Graphs – Grid Example

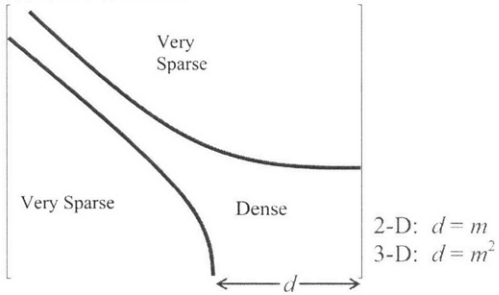
    How long does it take to factor an $m \times m$ grid

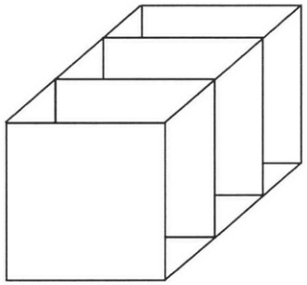


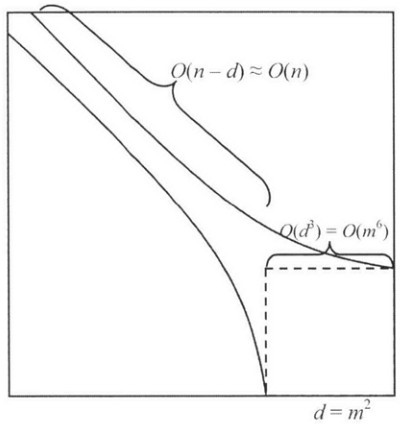
Suppose the center column is eliminated last?

- Fill-In

  Pattern of a Filled-in Matrix







G.E. Complexity
$O(n) + O(d^3)$

- Complexity of Sparse GE

	matrix size	dense block
$1-D$	$n = m$	$d = 1$
$2-D$	$n = m^2$	$d = m$
$3-D$	$n = m^3$	$d = m^2$

domain size $= m$

1-D	$O(n) + O(d^3) = O(m)$	$\leftarrow$ 100 pt grid	$O(100)$ ops	1 μsec
2-D	$O(n) + O(d^3) = O(m^3)$	$\leftarrow 100 \times 100$ grid	$O(10^6)$ ops	10 msec
3-D	$O(n) + O(d^3) = O(m^6)$	$\leftarrow 100 \times 100 \times 100$ grid	$O(10^{12})$ ops	2h 46m

e.g. on a 1 GFlops computer using $m = 100$ and a constant $c = 10$
For 3-D problems – can we do any better with GCR?

Krylov Method Cost

- The Generalized Conjugate Residual Algorithm
  - Algorithm Complexity

Compute $\mathbf{A}r^{k-1}$ — Sparse Matrix-vector product costs $O(n)$

$$\tilde{p}_{k-1} \leftarrow r^{k-1} - \sum_{j=0}^{k-1} (\mathbf{A}p_j)^T (\mathbf{A}r^{k-1}) p_j$$

inner product : $O(n)$

$$p_{k-1} \leftarrow \frac{\tilde{p}_{k-1}}{\|\mathbf{A}\tilde{p}_{k-1}\|}$$

$$y_{k-1} \leftarrow (r^{k-1})^T \mathbf{A}p_{k-1}$$

inner product : $O(n)$

$$x^k \leftarrow x^{k-1} + y_{k-1} p_{k-1}$$

$$r^k \leftarrow r^{k-1} - y_{k-1} \mathbf{A}p_{k-1}$$

multiplications : $O(n)$

Algorithm is $O(kn)$ where k = number of iterations

  - Complexity of sparse GCR (general worst case iterations)

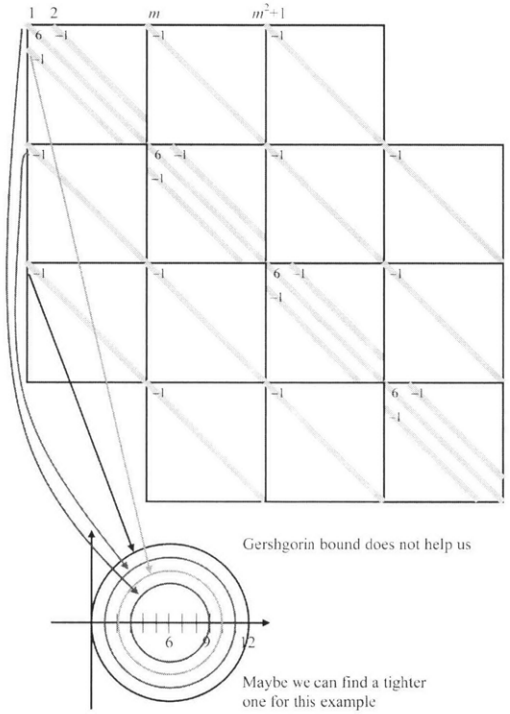
	matrix size	worst case iterations ($k = n$)
$1-D$	$n = m$	$k = m$
$2-D$	$n = m^2$	$k = m^2$
$3-D$	$n = m^3$	$k = m^3$

domain size $= m$

1-D	$O(kn) = O(m^2)$	$\leftarrow$	100 pt grid	$O(10^4)$ ops	0.1 ms
2-D	$O(kn) = O(m^4)$	$\leftarrow$	100×100 grid	$O(10^8)$ ops	1 sec
3-D	$O(kn) = O(m^6)$	$\leftarrow$	$100 \times 100 \times 100$ grid	$O(10^{12})$ ops	2h 46m

e.g. on a 1 GFlops computer using $m = 100$ and a constant $c = 10$
But how many iterations k does it really take?

Let's estimate k # of iterations

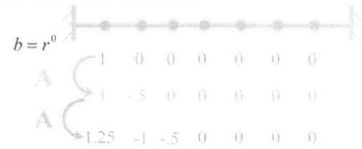


Gershgorin bound does not help us

Maybe we can find a tighter one for this example

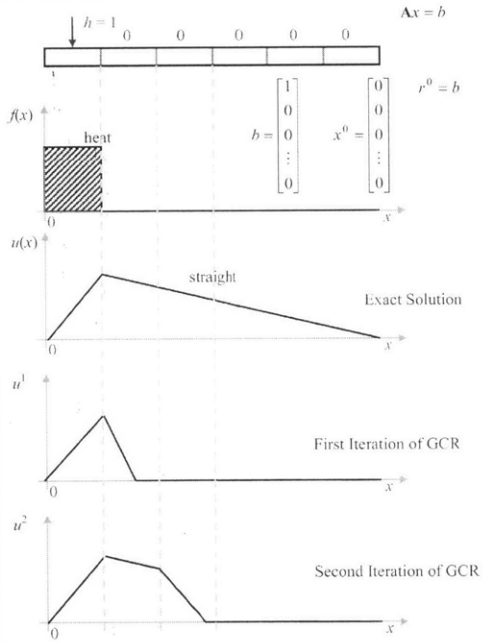
- Counting iterations
  - 1-D Case

1-D Discretized PDE

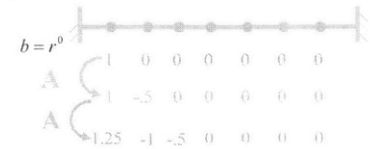


$$\underbrace{\begin{bmatrix} 1 & -\frac{1}{2} & & & & \\ -\frac{1}{2} & 1 & -\frac{1}{2} & & & \\ & -\frac{1}{2} & 1 & \ddots & & \\ & & \ddots & \ddots & \ddots & -\frac{1}{2} \\ & & & & -\frac{1}{2} & 1 \end{bmatrix}}_{A} \begin{bmatrix} 1 \\ -\frac{1}{2} \\ \vdots \\ 0 \end{bmatrix} = \begin{bmatrix} 1.25 \\ -1 \\ -\frac{1}{2} \\ 0 \end{bmatrix}$$

Maximum sum of temperatures will occur when heat is put into the center element



- Communication Lower Bound

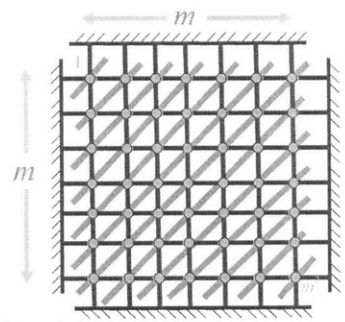


Communication Lower Bound for m gridpoints

$\mathbf{A}^k r^0$ is nonzero in m^{th} entry after $k = m$ iterations

Need at least m iterations for $\left(x^{k+1}\right)_m = \left(x^0 + \sum_{j=1}^{k} \alpha_j r^j\right)_m \neq 0$

- 2-D Case



For an $m \times m$ Grid

If $r^0 = \begin{bmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix}$

Takes $\approx \sqrt{2}m = O(m)$ iters for $\left(x^{k+1}\right)_{m^2} \neq 0$

- Eigenanalysis

3D Condition number: $\kappa \equiv \frac{\lambda_{max}}{\lambda_{min}} = \left(1 - \cos\left(\frac{m\pi}{m+1}\right)\right)\left(1 - \cos\left(\frac{\pi}{m+1}\right)\right)^{-1}$

Number of Iterations for GCR to achieve convergence

$\frac{\|r^k\|}{\|r^0\|} \leq 2\left(\frac{\sqrt{\kappa} - 1}{\sqrt{\kappa} + 1}\right)^k \leq \gamma$ (convergence tolerance)

$$k = \frac{\log \gamma}{\log\left(\frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1}\right)} \underset{m\to\infty}{\cong} O(m)$$

GCR achieves communication lower bound $O(m)$!

❖ Complexity of sparse GCR (no precond) for 3D FD methods

	matrix size	iterations ($k = n$)	
$1-D$	$n = m$	$k = m$	domain size $= m$
$2-D$	$n = m^2$	$k = m$	
$3-D$	$n = m^3$	$k = m$	

1-D	$O(kn) = O(m^2)$	←	100 pt grid	$O(10^4)$ ops	0.1 ms
2-D	$O(kn) = O(m^3)$	←	100×100 grid	$O(10^6)$ ops	10 ms
3-D	$O(kn) = O(m^4)$	←	$100\times100\times100$ grid	$O(10^8)$ ops	1 sec

e.g. on a 1 GFlops computer using $m = 100$ and a constant $c = 10$

But how many iterations k does it really take?

❖ Work for Banded Gaussian Elimination, Relaxation and GCR

Dimension	Banded GE	Sparse GE	GCR
1	$O(m)$	$O(m)$	$O(m^2)$
2	$O(m^4)$	$O(m^3)$	$O(m^3)$
3	$O(m^7)$	$O(m^6)$	$O(m^4)$

GCR faster than banded in GE in 2 and 3 dimensions

Could be faster, 3-D matrix only m^3 nonzeros.

Must defeat the communication lower bound!

$m = 100$

3–D SPG – $O(10^{12})$ ops ← 10,000 sec

GCR – $O(10^8)$ ops ← 1 sec

Dimension	Banded GE	Sparse GE	GCR (no preconditioner)
1	1 μs	1 μs	0.1 ms
2	1 sec	10 ms	10 ms
3	11days 6h	2h 46m	1 sec

e.g. on a 1 GFlops computer using $m = 100$, and constant $c = 10$

GCR faster than banded GE in 2 and 3 dimensions

Could be faster, 3-D matrix only m^3 nonzeros.

Must defeat the communication lower bound!

Dimension	Banded GE	Sparse GE	GCR (no preconditioner)
1	10 μs	10 μs	10 ms
2	2h 46m	10 sec	10 sec
3	317,000 years	317 years	2h 46m

e.g. on a 1 GFlops computer using $\underline{m = 1000}$, and constant $c = 10$

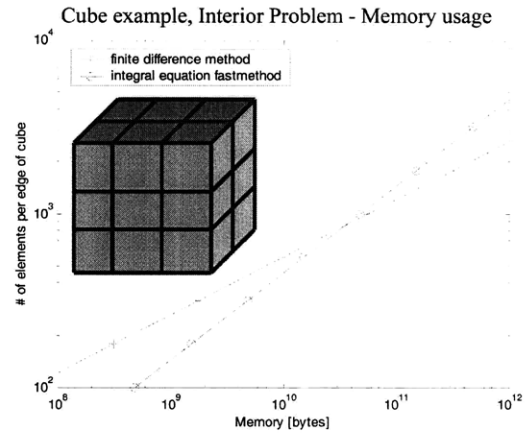
Dimension	Banded GE	Sparse GE	GCR	GCR + preconditioner
1	$O(m)$	$O(m)$	$O(m^2)$	$O(km)$
2	$O(m^4)$	$O(m^3)$	$O(m^3)$	$O(km^2)$
3	$O(m^7)$	$O(m^6)$	$O(m^4)$	$O(km^3)$

GCR+good preconditioner (i.e. $k = 5$–10) is always the best choice for 2D & 3D problems!

Dimension	Banded GE	Sparse GE	GCR	GCR + preconditioner
1	10 μs	10 μs	10 ms	0.1 ms
2	2h 46m	10 sec	10 sec	0.1 sec
3	317,000 years	317 years	2h 46m	1m 20s

e.g. on a 1 GFlops computer using $\underline{m} = 1000$, k=10, and constant c=10

❖ Cube Example, Interior Problem – Memory Usage



3.20 PDE - Finite Element Methods & GCR Pre-conditioners

INTRODUCTION TO NUMERICAL SIMULATION

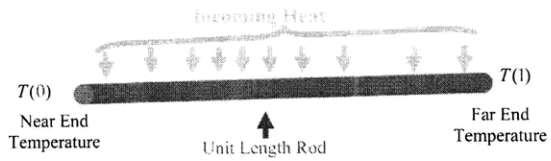
LECTURE 21.
Finite Element Methods & GCR Preconditioners

TODAY'S OUTLINE:

- Finite Element Methods
  - The FEM Basis Functions
  - The Galerkin scheme
  - The FEM linear system
  - An FEM Example
  - Energy minimization view of FEM
- GCR Preconditioners for FD and FEM
  - Diagonal Preconditioners
  - Block Diagonal Preconditioners
  - Incomplete Factorization Preconditioners
  - Communication Improving Preconditioners: Gauss-Seidel
  - Examples of GCR + Preconditioners for 3D FD

Finite Element Methods

HEAT FLOW 1-D EXAMPLE



Question: What is the temperature distribution along the bar

- Normalized Poisson Equation

$$\frac{\partial}{\partial x}\kappa\frac{\partial T(x)}{\partial x} = -h_s \Rightarrow -\frac{\partial^2 u(x)}{\partial x^2} = f(x)$$

$$-u_{,xx}(x) = f(x)$$

THE FEM BASIS FUNCTIONS

- Residual Equation

Partial Differential Equation Form

$$-\frac{\partial^2 u}{\partial x^2} = f \quad u(0)=0 \quad u(1)=0$$

Step 1: Choose Basis Functions to represent the solution

$$u(x) \approx u_h(x) = \sum_{i=1}^{n} \omega_i \underbrace{\varphi_i(x)}_{\text{Basis Functions}}$$

$$-\frac{\partial^2 u}{\partial x^2} = f(x) \qquad u(x) = \sum_{j=1}^{n} w_j \varphi_j(x) \qquad u(0)=0 \quad u(1)=0$$

What are good basis functions?

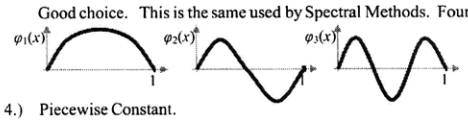
1.) $\varphi(x) = a_0 + a_1 x \rightarrow$ No, second derivative is always zero

2.) $\varphi(x) = a_0 + a_1 x + a_2 x^2 \rightarrow$ Could work

3.) I can try to build into the basis function the boundary conditions

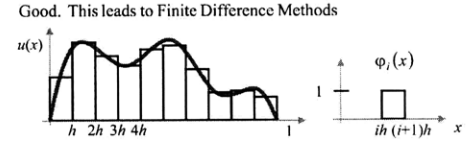
e.g. $\varphi_i(x) = \sin 2\pi i x$ [and second derivative is nonzero]

Good choice. This is the same used by Spectral Methods. Fourier Series.



4.) Piecewise Constant.

Good. This leads to Finite Difference Methods



5.) Piecewise Linear.

Step 2: Generate equations for the basis functions weights setting residual orthogonal to some test functions

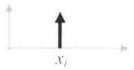
$$\int_0^1 \underbrace{\varphi_j(x)}_{\text{Test Functions}} R(x)\,dt = 0 \qquad R(x) = \sum_{i=1}^{n} \omega_i \frac{d^2\varphi_j(x)}{dx^2} + f(x)$$

Test Functions $\quad -\frac{\partial^2}{\partial x^2}\left(\sum_{i=1}^{n} w_i \varphi_i(x)\right) - f(x) = R(x) \overset{?}{=} 0$

1.) Collocation $\quad R(x_i) = 0 \quad i = 1,\dots,n \quad \Psi_i(x) = \delta(x - x_i)$

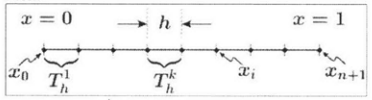
2.) Galerkin $\quad \varphi_i \perp R(x) \quad \int_0^1 \varphi_i(x) R(x)\,dx = 0 \quad \Psi_i(x) = \varphi_i(x)$

3.) Other Test Functions $\quad \Psi_i \perp R(x) \quad \int_0^1 \Psi_i(x) R(x)\,dx = 0$



$$\int_0^1 \delta(x - x_i) R(x) = R(x_i)$$

- Rayleigh-Ritz Approach – Approximation
  - Mesh

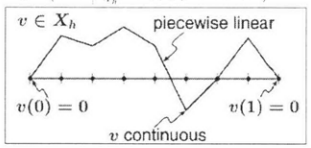


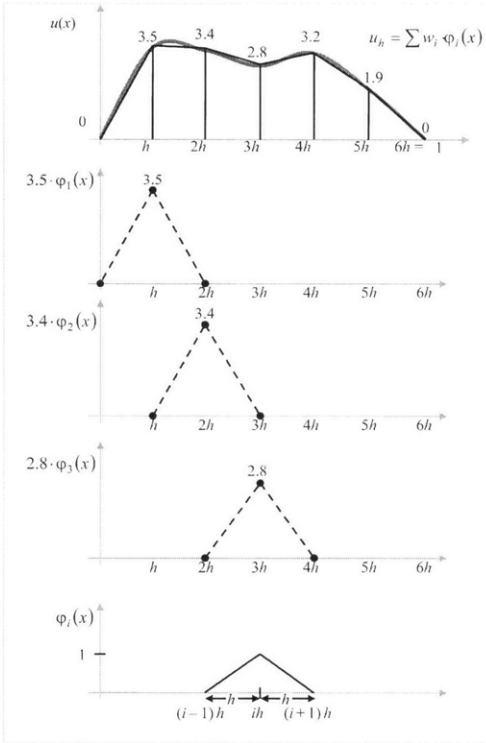
$\Omega = \bigcup_{k=1}^{K} T_h^k \quad T_h^k, k = 1,\dots K = n+1: \quad$ elements

$x_i, i = 0,\dots,n+1: \quad$ nodes

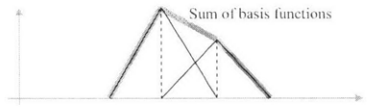
  - Space $X_h \subset X$

$X_h = \left\{ v \in X \,\middle|\, v|_{T_h^k} \in \mathrm{P}_1\left(T_h^k\right), k = 1,\dots,K \right\}$



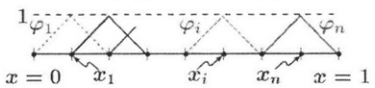


Adding two basis functions



- Basis

  Nodal basis for $\mathbf{X}_h$: $\varphi_j, j = 1,\ldots, n = \dim(\mathbf{X}_h)$



  φ_i nonzero only on $\overline{T}_h^i \cup \overline{T}_h^{i+1}$

❖ Example Basis Functions

Introduce basis representation $u(x) \approx u_h(x) = \sum_{i=1}^{n} \omega_i \underbrace{\varphi_i(x)}_{\text{Basis Functions}}$

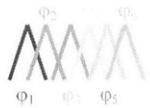
$\Rightarrow u_h(x)$ is a weighted sum of basis functions

The basis functions define a space

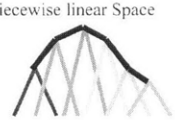
$$X_h = \left\{ v \in X_h \,\middle|\, v = \sum_{i=1}^{n} \beta_i \varphi_i \text{ for some } \beta_i\text{'s} \right\}$$

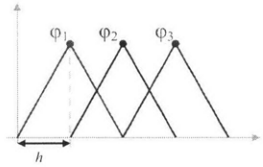
<u>Example</u>

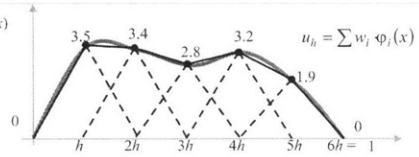
"Hat" basis functions



Piecewise linear Space







THE COLLOCATION SCHEME

Assume the initial condition is satisifed

$$\sum_{i=1}^{n} \omega_i \varphi_i(0) = x_0$$

Define the Residual

$$R(t) = \sum_{i=1}^{n} \omega_i \frac{d\varphi_i(t)}{dt} - A\sum_{i=1}^{n} \omega_i \varphi_i(t) - u(t)$$

Select weights to "minimize" the residual

Force the residual to be zero at n test points

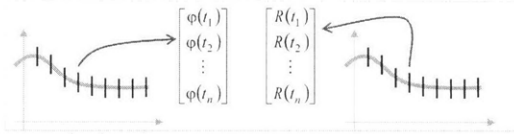
$$R(t_l) = \sum_{i=1}^{n} \omega_i \frac{d\varphi_i(t_l)}{dt} - A\sum_{i=1}^{n} \omega_i \varphi_i(t_l) - u(t_l) = 0 \qquad l \in \{1,\ldots,n\}$$

Generates n equations in n unknowns

THE GALERKIN SCHEME

Force the residual to be "orthogonal" to the basis functions

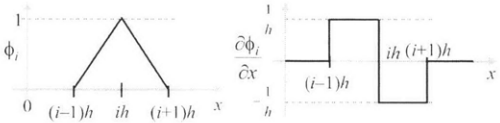
$$\int_0^1 \varphi_l(x) R(x) dt = 0$$



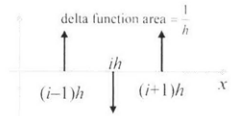
Generates n equations in n unknowns

$$\int_0^1 \varphi_l(x)\left[\sum_{i=1}^{n} \omega_i \frac{d^2\varphi_i(x)}{dx^2} + f(x)\right]dx = 0 \qquad l \in \{1,\ldots,n\}$$

$$\frac{\partial^2 \phi_i}{\partial x^2} = ?$$



$$\frac{\partial^2 \phi_i}{\partial x^2}$$



We don't want to work with these!!!

$$\int_0^1 \varphi_i(x)\left[\frac{\partial^2}{\partial x^2}\sum_{j=1}^{n} w_j \varphi_j(x) + f\right]dx = 0$$

$$\sum_{j=1}^{n} w_j \underbrace{\left[\int_0^1 \varphi_i(x)\frac{\partial^2}{\partial x^2}\varphi_j(x)dx\right]}_{A_{ij}} = -\int_0^1 \varphi_i(x) f(x)dx$$

$$\mathbf{A}x = b$$

$$\begin{bmatrix} & & \\ & A_{ij} = \int_0^1 \varphi_j(x)\frac{\partial^2}{\partial x^2}\varphi_i(x)dx & \\ & & \end{bmatrix} \begin{bmatrix} w_1 \\ \vdots \\ w_i \\ \vdots \\ w_n \end{bmatrix} = \begin{bmatrix} \int_0^1 \varphi_1(x) f(x)dx \\ \vdots \\ \int_0^1 \varphi_i(x) f(x)dx \\ \vdots \\ \int_0^1 \varphi_n(x) f(x)dx \end{bmatrix}$$

Integrate by parts to avoid having to deal with second derivatives of hat basis functions.

$$\left[\varphi_l(x)\sum_{i=1}^{n} w_i \frac{d\varphi_i}{dx}\right]_0^1 - \int_0^1 \frac{d\varphi_l}{dx}\sum_{i=1}^{n} w_i \frac{d\varphi_i}{dx}dx + \int_0^1 \varphi_l(x) f(x)dx = 0 \qquad l = \{1,2,\ldots n\}$$

$$\underbrace{\overset{0}{\varphi_l(1)}\sum_{i=1}^{n} w_i \frac{d\varphi_i}{dx}\Big|_{x=1} - \overset{0}{\varphi_l(0)}\sum_{i=1}^{n} w_i \frac{d\varphi_i}{dx}\Big|_{x=0}}_{\text{Impose the Boundary Conditions}}$$

$$\underbrace{-\int_0^1 \frac{d\varphi_l}{dx}\sum_{i=1}^{n} w_i \frac{d\varphi_i}{dx}dx + \int_0^1 \varphi_l(x) f(x)dx = 0}_{\text{Note: Do } \textbf{NOT} \text{ need } 2^{\text{nd}} \text{ derivatives}} \qquad l = \{1,2,\ldots n\}$$

Integration by Parts

$$\sum w_i \int_0^1 \varphi_l(x) \frac{\partial^2 \varphi_i(x)}{\partial x^2} dx = \sum w_i \left[\varphi_l(x) \frac{\partial \varphi_i(x)}{\partial x} \right]_0^1 - \sum w_i \int_0^1 \frac{\partial \varphi_l(x)}{\partial x} \frac{\partial \varphi_i(x)}{\partial x} dx$$

$$= \cancel{\sum_{\{\}} w_i \left[\varphi_l(1) \frac{\partial \varphi_i(1)}{\partial x} \right]} - \cancel{\sum_{\{\}} w_i \left[\varphi_l(0) \frac{\partial \varphi_i(0)}{\partial x} \right]}$$

$$- \sum w_i \int_0^1 \frac{\partial \varphi_l(x)}{\partial x} \frac{\partial \varphi_i(x)}{\partial x} dx$$

$$= -\sum w_i \int_0^1 \frac{\partial \varphi_l(x)}{\partial x} \frac{\partial \varphi_i(x)}{\partial x} dx$$

$$\mathbf{A}x = b$$

$$\begin{bmatrix} & & \\ & A_{ij} = -\int_0^1 \frac{\partial \varphi_l(x)}{\partial x} \frac{\partial \varphi_i(x)}{\partial x} dx & \\ & & \end{bmatrix} \begin{bmatrix} w_1 \\ \vdots \\ w_i \\ \vdots \\ w_n \end{bmatrix} = \begin{bmatrix} \int_0^1 \varphi_1(x) f(x) dx \\ \vdots \\ \int_0^1 \varphi_i(x) f(x) dx \\ \vdots \\ \int_0^1 \varphi_n(x) f(x) dx \end{bmatrix}$$

Only first derivatives of basis functions

$$\int_0^1 \frac{d\varphi_l(x)}{dx} \frac{d \sum_{i=1}^n \omega_i \varphi_i(x)}{dx} dx - \int_0^1 \varphi_l(x) f(x) dx = 0 \qquad l \in \{1, \ldots, n\}$$

$$\sum_i \omega_i \int_0^1 \frac{d\phi_1}{dx} \cdot \frac{d\phi_i}{dx} dx = \int_0^1 \phi_1(x) f(x) dx$$

$$\omega_1 \frac{1}{(\Delta x)^2} (2 \cdot \Delta x) + \omega_2 \frac{1}{(\Delta x)^2} (-1 \cdot \Delta x) = \int_0^1 \phi_1(x) f(x) dx$$

$$\omega_1 \frac{1}{(\Delta x)^2} (-1 \cdot \Delta x) + \omega_2 \frac{1}{(\Delta x)^2} (2 \cdot \Delta x) - \omega_3 \frac{1}{(\Delta x)^2} (-1 \cdot \Delta x) = \int_0^1 \phi_1(x) f(x) dx$$

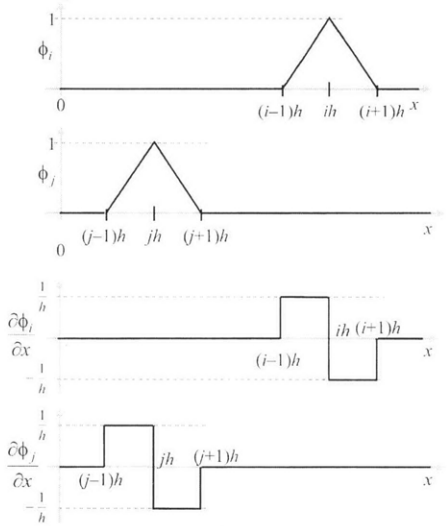
$$\frac{1}{\Delta x} \begin{bmatrix} 2 & -1 & & \\ -1 & 2 & \ddots & \\ & \ddots & \ddots & -1 \\ & & -1 & 2 \end{bmatrix} \begin{bmatrix} \omega_n \\ \omega_n \\ \vdots \\ \omega_n \end{bmatrix} = \begin{bmatrix} \int \phi_1 f(x) dx \\ \int \phi_2 f(x) dx \\ \vdots \\ \int \phi_n f(x) dx \end{bmatrix}$$

The FEM Linear System

$$\sum_{i=1}^n \omega_i \underbrace{\int_0^1 \frac{d\varphi_l}{dx} \frac{d\varphi_i}{dx} dx}_{A_{l,i}} = \underbrace{\int_0^1 \varphi_l(x) f(x) dx}_{F_l}$$

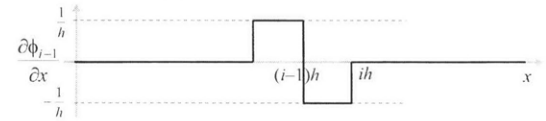
$$\begin{bmatrix} & & \\ & \mathbf{A} & \\ & & \end{bmatrix} \begin{bmatrix} \omega_1 \\ \omega_2 \\ \vdots \\ \omega_n \end{bmatrix} = \begin{bmatrix} F_1 \\ F_2 \\ \vdots \\ F_n \end{bmatrix}$$

$$A_{ij} = \int_0^1 \frac{\partial \varphi_i}{\partial x} \frac{\partial \varphi_j}{\partial x}\, dx$$

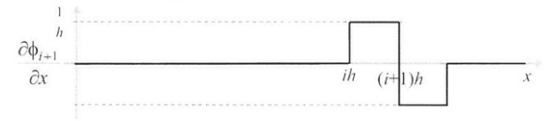


They do **not** overlap unless $j = i-1,\ i,\ i+1$

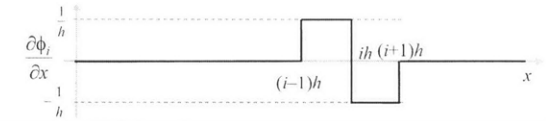
$$A_{i,i-1} = \tfrac{1}{h} \cdot \left(-\tfrac{1}{h}\right) \cdot h = -\frac{1}{h}$$



$$A_{i,i+1} = \left(-\tfrac{1}{h}\right) \cdot \tfrac{1}{h} \cdot h = -\frac{1}{h}$$



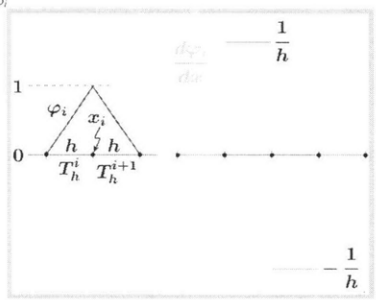
$$A_{i,i} = \tfrac{1}{h} \cdot \tfrac{1}{h} \cdot h + \left(-\tfrac{1}{h}\right) \cdot \left(-\tfrac{1}{h}\right) \cdot h = \frac{2}{h}$$

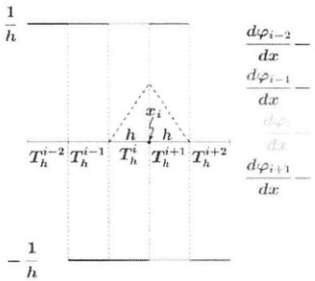


$$\mathbf{A} = \frac{1}{h} \cdot \begin{bmatrix} 2 & -1 & & & & & \\ -1 & \ddots & \ddots & & & \mathbf{0} & \\ & \ddots & \ddots & \ddots & & & \\ & & -1 & 2 & -1 & & \\ & & & \ddots & \ddots & \ddots & \\ & \mathbf{0} & & & \ddots & \ddots & -1 \\ & & & & & -1 & 2 \end{bmatrix}$$

Discrete Equations

❖ φ_i





❖ Typical Row

$$A^1_{h\,i\,j} = \int_\Omega \frac{d\varphi_i}{dx}\frac{d\varphi_j}{dx}\,dx = \int_{T_h^i} \frac{d\varphi_i}{dx}\frac{d\varphi_j}{dx}\,dx + \int_{T_h^{i-1}} \frac{d\varphi_i}{dx}\frac{d\varphi_j}{dx}\,dx$$

is nonzero only for $i = j-1, j, j+1$

$$A^1_{h\,i\,i} = \frac{1}{h^2}(h) + \frac{1}{h^2}(h) = \frac{2}{h}$$

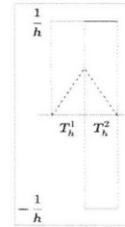
$$A^1_{h\,i\,i-1} = \frac{1}{h}\left(-\frac{1}{h}\right)(h) = -\frac{1}{h}$$

$$A^1_{h\,i\,i} = \left(-\frac{1}{h}\right)\frac{1}{h}(h) = -\frac{1}{h}$$

❖ Boundary Rows

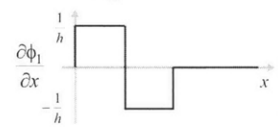
$$A^1_{h\,11} = \frac{2}{h} \quad A^1_{h\,12} = -\frac{1}{h}$$

$$A^1_{h\,nn} = \frac{2}{h} \quad A^1_{h\,n\,n-1} = -\frac{1}{h}$$

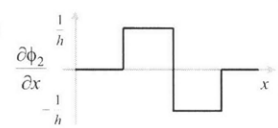


$$\mathbf{A} = \frac{1}{h}\cdot\begin{bmatrix} 2 & -1 & & & & \\ -1 & \ddots & \ddots & & & \\ & \ddots & \ddots & \ddots & & \\ & & -1 & 2 & -1 & \\ & & & \ddots & \ddots & \ddots \\ & & & & \ddots & \ddots & -1 \\ & & & & & -1 & 2 \end{bmatrix}$$

$$A_{11} = \frac{2}{h} = A_{n,n}$$



$$A_{12} = \left(-\frac{1}{h}\right)\cdot\frac{1}{h}\cdot h = -\frac{1}{h} = A_{21}$$

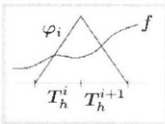


- Properties of $\mathbf{A}_h$

$$\underline{A}_h^1 = \frac{1}{h}\begin{pmatrix} 2 & -1 & & & & & \\ -1 & 2 & -1 & & & \mathbf{0} & \\ & -1 & 2 & -2 & & & \\ & & & & \ddots & & \\ & \mathbf{0} & & & -1 & 2 & -1 \\ & & & & & -1 & 2 \end{pmatrix}$$

$\underline{A}_h^1$ is SPD [Symmetric Positive Definite]; and
diagonally dominant; and
sparse; and
tridiagonal.

- "Load" Vector Elements: F_h



$$F_{hi} = \int_0^1 f\varphi_i dx$$

$$F_{hi} = \int_{T_h^i} f\varphi_i dx + \int_{T_h^{i+1}} f\varphi_i dx, \quad i = 1,\ldots,n;$$

Comparing FEM and FD in 1-D

- Residual Equation

Partial Differential Equation Form

$$-\frac{\partial^2 u}{\partial x^2} = f \quad u(0) = 0 \quad u(1) = 0$$

Step 1: Choose Basis Functions to represent the solution

$$u(x) \approx u_h(x) = \sum_{i=1}^{n} \omega_i \underbrace{\varphi_i(x)}_{\text{Basis Functions}}$$

Step 2: Generate equations for the basis functions weights setting residual orthogonal to some test functions

$$\int_0^1 \underbrace{\varphi_l(x)}_{\text{Test Functions}} R(x) dt = 0 \quad R(x) = \sum_{i=1}^{n} \omega_i \frac{d^2\varphi_i(x)}{dx^2} + f(x)$$

- FEM – Basis Weights
  - Galerkin Scheme

    Force the residual to be "orthogonal" to the basis functions

$$\int_0^1 \varphi_l(x) R(x) dt = 0$$

Generates n equations in n unknowns

$$\int_0^1 \varphi_l(x)\left[\sum_{i=1}^{n} \omega_i \frac{d^2\varphi_i(x)}{dx^2} + f(x)\right] dx = 0 \quad l \in \{1,\ldots,n\}$$

  - Linear System

$$\sum_{i=1}^{n} \omega_i \underbrace{\int_0^1 \frac{d\varphi_l}{dx}\frac{d\varphi_i}{dx} dx}_{A_{l,i}} = \underbrace{\int_0^1 \varphi_l f(x) dx}_{F_l}$$

$$\frac{1}{\Delta x}\begin{bmatrix} 2 & -1 & & & \\ -1 & 2 & -1 & & \\ & -1 & 2 & \ddots & \\ & & \ddots & \ddots & -1 \\ & & & -1 & 2 \end{bmatrix}\begin{bmatrix} \omega_1 \\ \omega_2 \\ \vdots \\ \omega_n \end{bmatrix} = \begin{bmatrix} F_1 \\ F_2 \\ \vdots \\ F_n \end{bmatrix}$$

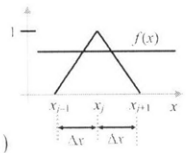
- Comparing – FD & FEM (hat basis) – 1D problem

$$\text{FEM} \quad \frac{1}{\Delta x}\begin{bmatrix} 2 & -1 & & \\ -1 & 2 & \ddots & \\ & \ddots & \ddots & -1 \\ & & -1 & 2 \end{bmatrix}\begin{bmatrix} \omega_1 \\ \omega_2 \\ \vdots \\ \omega_n \end{bmatrix} = \begin{bmatrix} \int_0^1 \varphi_1(x) f(x) dx \\ \int_0^1 \varphi_2(x) f(x) dx \\ \vdots \\ \int_0^1 \varphi_n(x) f(x) dx \end{bmatrix}$$

$$\text{FD} \quad \frac{1}{\Delta x}\begin{bmatrix} 2 & -1 & & \\ -1 & 2 & \ddots & \\ & \ddots & \ddots & -1 \\ & & -1 & 2 \end{bmatrix}\begin{bmatrix} \hat{u}_1 \\ \hat{u}_2 \\ \vdots \\ \hat{u}_n \end{bmatrix} = \begin{bmatrix} f(x_1) \\ f(x_2) \\ \vdots \\ f(x_n) \end{bmatrix}$$

F.D. $\Delta x \cdot f(x_i)$

F.E.M. $\int_0^1 \varphi_j(x) f(x) dx$



If $f(x)$ is constant:

$$\int_0^1 \varphi_j(x) f(x) dx = \frac{1}{2} \cdot 2 \cdot \Delta x \cdot f(x_j)$$

→ same as F.D.

If $f(x)$ is not constant the RHS is still an approximation

$$\int_0^1 \varphi_i(x) f(x) dx \cong \frac{1}{2} = \Delta x \cdot f(x_i)$$

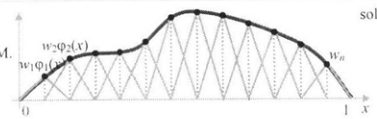
F.E.M.

$$\frac{1}{\Delta x}\begin{bmatrix} 2 & -1 & & \\ -1 & 2 & \ddots & \\ & \ddots & \ddots & -1 \\ & & -1 & 2 \end{bmatrix} = \Delta x \begin{bmatrix} f(x_1) \\ \vdots \\ \vdots \\ f(x_n) \end{bmatrix}$$

Same as F.D. system for 1D hat basis

However F.E.M. gives more flexibility (can choose other basis)

F.E.M.



solution vector

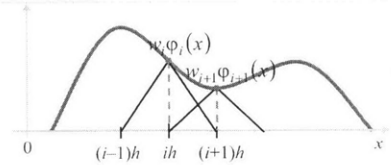
$$\begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_n \end{bmatrix}$$

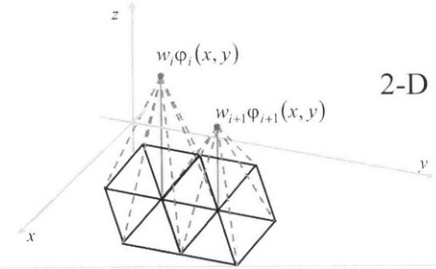
F.D.

solution vector

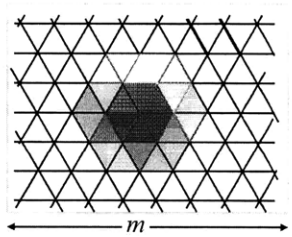
$$\begin{bmatrix} \hat{u}_1 \\ \hat{u}_2 \\ \vdots \\ \hat{u}_n \end{bmatrix}$$

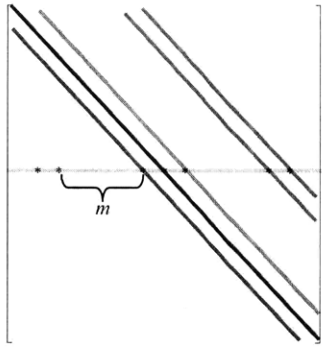
AN FEM EXAMPLE





Top View

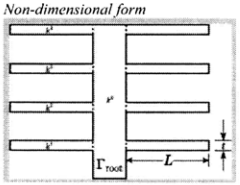




A

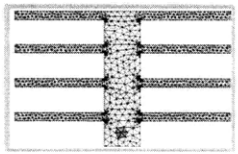
Heat Transfer Problem

Non-dimensional form

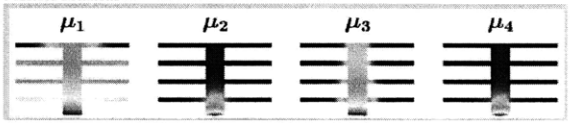


k^i: thermal conductivity for Ω_i, $i = 0,1,\ldots,4$
Bi: heat transfer coefficient
t, L: geometric parameters

$X_h = \text{span}\{\varphi_1,\ldots,\varphi_n\}$
$\varphi_i(x)$: Nodal basis functions
– first order elements $\dim(X_h) = n$



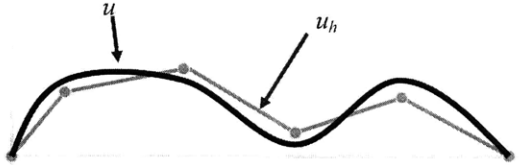
Possible Solutions



CONVERGENCE ANALYSIS OF FEM

❖ Convergence Analysis

The question is



How does $\underbrace{\|u - u_h\|}_{error}$ decrease with refinement?

❖ Review of FEM

Introduce basis representation $u(x) \approx u_h(x) = \sum_{i=1}^{n} \omega_i \underbrace{\varphi_i(x)}_{\text{Basis Functions}}$

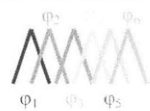
$\Rightarrow u_h(x)$ is a weighted sum of basis functions

The basis functions define a space

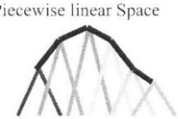
$$X_h = \left\{ v \in X_h \middle| v = \sum_{i=1}^{n} \beta_i \varphi_i \text{ for some } \beta_i\text{'s} \right\}$$

<u>Example</u>

"Hat" basis functions



Piecewise linear Space



Partial Differential Equation Form

$$-\frac{\partial^2 u}{\partial x^2} = f \quad u(0) = 0 \quad u(1) = 0$$

"Nearly" Equivalent Weak Form

$$\underbrace{\int_\Omega \frac{\partial u}{\partial x}\frac{\partial v}{\partial x} dx}_{a(u,v)} = \underbrace{\int_\Omega f v dx}_{l(v)} \quad \text{for all } v$$

Introduced an abstract notation for the equation u must satisfy

$a(u,v) = l(v)$ for all v

$a(u,u)$ defines a norm $a(u,u) = \|u\|^2$

u is restricted to be 0 at 0 and 1!!!

<u>Key Idea:</u>

Using the norm properties, it is possible to show

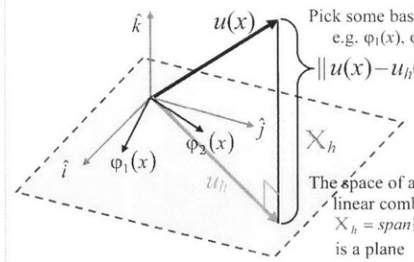
If $a(u_h, \varphi_i) = l(\varphi_i)$ for all $\varphi_i \in \{\varphi_1, \varphi_2, \ldots, \varphi_n\}$

Note: this is basically Galerkin

Then $\underbrace{\|u - u_h\|}_{\text{Solution Error}} = \min_{w_h \in X_h} \underbrace{\|u - w_h\|}_{\text{Projection Error}}$

Or in other words: given some basics, Galerkin finds the best coefficients

$u(x) \in \Re^\infty$ Here $u(x) \in \Re^3$ for visualization



Pick some basis functions e.g. $\varphi_1(x)$, $\varphi_2(x)$

$\| u(x) - u_h(x) \|$

The space of all their linear combinations $X_h = span\{\varphi_1(x), \varphi_2(x)\}$ is a plane

Here, for visualization, assume the $\{\hat{i}, \hat{j}\}$ plane

The key idea says that the error

$$\|u(x) - u_h(x)\| = \|e_h\| = \sqrt{a(e,e)} = \sqrt{a(u(x) - u_h(x), u(x) - u_h(x))}$$

$$= \boxed{\left[\int_0^1 \frac{\partial(u - u_h)}{\partial x}\frac{\partial(u - u_h)}{\partial x} dx\right]^{\frac{1}{2}}}$$

in 1D example

is minimized by choosing $u_h(x)$ such that

$$\begin{cases} a(u_h, \varphi_1) - \ell(\varphi_1) = 0 \\ a(u_h, \varphi_2) - \ell(\varphi_2) = 0 \end{cases}$$ the solution is obtained by projection

⇨ for a given set of basis, Galerkin gives the projection and hence gives the smallest possible error (best coefficients)

⇨ Note: to do better than that I need to change X_h the set of basis

For instance hat basis

As I decrease $h = \frac{1}{n}$ the error decreases linearly

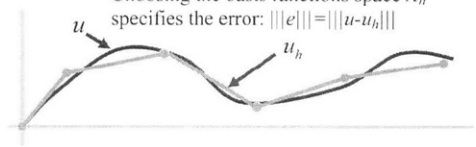
$$\|e_h\| = \|u - u_h\| = \mathcal{O}\left(\frac{1}{n}\right) \xrightarrow[n \to \infty]{} 0$$

- Error Analysis – Energy Norm

  In words: even if you ***knew*** u,
  you could not find a w_h in X_h
  more accurate than u_h ***in the energy norm.***

- FEM Convergence Analysis

Choosing the basis functions space X_h specifies the error: $|||e|||=|||u-u_h|||$



Question: How does that error change when I change basis?

For piecewise linear: $\underbrace{|||u-u_h|||}_{error}=O\left(\frac{1}{n}\right)$

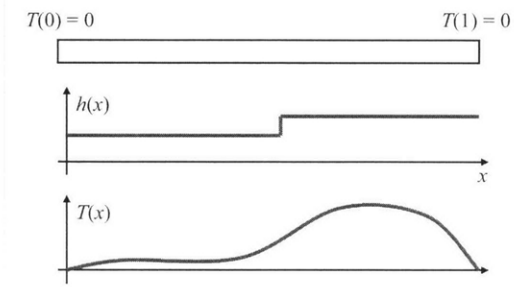
Energy Minimization View of FEM

- Problem of Interest
  - Helmholtz Equation in 1D

    Boundary Value Problem (BVP) – Strong Form

    $$-u''(x)+\alpha u(x)=f(x) \qquad \alpha \geq 0$$
    $$x \in (0,1) \quad u(0)=u(1)=0$$

    Describes many physical phenomena (e.g.):
    - Temperature distribution in a bar
    - Deformation of an elastic bar
    - Deformation of a string under tension
  - Solution Properties
    - The solution $u(x)$ always ***exists***
    - $u(x)$ is always smoother than the data $f(x)$
    - given $f(x)$ the solution $u(x)$ is ***unique***



- Minimization Principle Statement

  Find
  $$u=\arg\min_{w\in X} J(w)$$
  where
  $$X=\{v \text{ sufficiently smooth } v(0)=v(1)=0\}$$
  and
  $$J(w)=\frac{1}{2}\int_0^1 (w_x w_x+\alpha w w)dx-\int_0^1 fw dx$$

$$J_{\text{Poisson}}(w)=\frac{1}{2}\int_0^1 w_x w_y dx-\int_0^1 fw dx$$

$$\int_0^1 \underbrace{R(x)}_{u''-f(x)}\varphi dx=\int_0^1 \frac{\partial\varphi}{\partial x}\frac{\partial\varphi}{\partial x}dx-\int_0^1 f\varphi dx$$

$$\int_0^1 (u''-f(x))v=0 \qquad \forall v$$

In words:

Over all functions w in X,

u that satisfies

$$-u_{xx} + \alpha u = f \quad \text{in } \Omega$$

$$u(0) = u(1) = 0$$

makes $J(w)$ as small as possible

- Proof

Let $w = u + v$

Then

$$J\Big(\overbrace{\underbrace{u}_{\in X} + \underbrace{v}_{\in X}}^{w \in X}\Big) = \frac{1}{2}\int_0^1 (u+v)_x (u+v)_x \, dx + \frac{\alpha}{2}\int_0^1 (u+v)(u+v)dx - \int_0^1 f(u+v)dx$$

$$\begin{aligned} J(u+v) = \; & \frac{1}{2}\int_0^1 (u_x u_x + \alpha uu)dx - \int_0^1 fudx && J(u) \\ & + \int_0^1 (u_x v_x + \alpha uv)dx - \int_0^1 fvdx && \underset{\text{first variation}}{\delta J_v(u)} \\ & + \frac{1}{2}\int_0^1 (v_x v_x + \alpha vv)dx && > 0 \text{ for } v \neq 0 \end{aligned}$$

$$\delta J_{\text{Poisson}}(w) = \int_0^1 u_x v_x dx - \int_0^1 fvdx$$

If $v_x = \varphi_i$ then Galerkin

$$\begin{aligned} \delta J_v(u) &= \int_0^1 (u_x v_x + \alpha uv)dx - \int_0^1 fvdx \\ &= \overbrace{v(0)}^{0} u_x(0) - \overbrace{v(1)}^{0} u_x(1) - \int_0^1 u_{xx} v dx + \alpha \int_0^1 uvdx - \int_0^1 fvdx \end{aligned}$$

$$J(\underbrace{u+v}_{w}) = J(u) + \frac{1}{2}\underbrace{\int_0^1 (v_x v_x + \alpha vv)dx}_{>0 \text{ unless } v=0}, \quad \forall v \in X$$

$$\Rightarrow \boxed{\begin{array}{c} J(w) > J(u), \quad \forall w \in X, w \neq u \\ \Updownarrow \\ u \text{ is } \textbf{THE} \text{minimizer of } J(w) \end{array}}$$

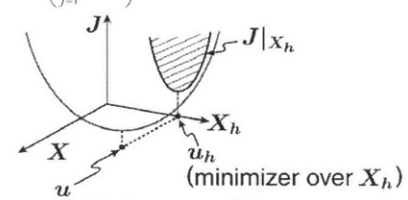
- Rayleigh-Ritz Approach – "Projection"

Let

$$\underbrace{u_h(\in X_h)}_{\text{RR/FE approximation}} = \sum_{j=1}^{n} u_{hj}\varphi_j(x)$$

Set $u_{hj} = w_j$ that minimize

$$J\left(\sum_{j=1}^{n} w_j \varphi_j\right)$$



$$J\left(\sum_{j=1}^{n} w_j \varphi_j\right) = \frac{1}{2}\int_0^1 \frac{d}{dx}\left(\sum_{i=1}^{n} w_i \varphi_i\right)\frac{d}{dx}\left(\sum_{j=1}^{n} w_j \varphi_j\right) + \cdots$$

$$\cdots + \frac{\alpha}{2}\int_0^1 \sum_{i=1}^{n}(w_i\varphi_i)\sum_{j=1}^{n}(w_j\varphi_j) - \int_0^1 f\sum_{j=1}^{n} w_j\varphi_j$$

$$= \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} w_i w_j \int_0^1 \left(\frac{d\varphi_i}{dx}\frac{d\varphi_j}{dx} + \alpha\varphi_i\varphi_j\right)dx - \sum_{j=1}^{n} w_j \int_0^1 f\varphi_j dx$$

By ***bilinearity*** and ***linearity***.

$$u_h(x) = \sum w_i \varphi_i(x)$$

$$-u_{xx} + \alpha u - f(x) = 0$$

$$\underbrace{-\frac{\partial^2}{\partial x^2}\left(\sum w_i \varphi_i(x)\right) + \alpha \sum w_i \varphi_i(x) - f(x)}_{R(x)} \cong 0$$

$$\int_0^1 \varphi_l(x) R(x) = 0 \qquad \forall l$$

$$-\int_0^1 \varphi_l(x) \sum_{i=1}^{n} w_i \frac{\partial^2}{\partial x^2} \varphi_i(x) dx + \int_0^1 \varphi_l(x) \alpha \sum_{i=1}^{n} w_i \varphi_i(x) dx - \int_0^1 \varphi_l(x) f(x) = 0$$

$$J^R\left(\underline{w} \in \Re^n\right) \equiv J\left(\sum_{j=1}^{n} w_j \varphi_j\right) = \tfrac{1}{2} \underline{w}^T \underline{A}_h \underline{w} - \underline{w}^T \underline{F}_h$$

$$\underline{F}_h \in \Re^n : \quad F_{hi} = \int_0^1 f \varphi_i dx$$

$$\underline{A}_h \in \Re^{n \times n} : \quad A_{hij} = \int_0^1 \left(\frac{d\varphi_i}{dx} \frac{d\varphi_j}{dx} + \alpha \varphi_i \varphi_j \right) dx$$

$$J(\vec{w}) = \begin{bmatrix} w_1 \\ \vdots \\ w_n \end{bmatrix}^T \begin{bmatrix} \cdots & \vdots & \cdots \\ & \frac{1}{2}\int_0^1 \left(\frac{\partial \varphi_i}{\partial x} \frac{\partial \varphi_j}{\partial x} + \alpha \varphi_i \varphi_j \right) dx & \\ & \vdots & \end{bmatrix} \begin{bmatrix} w_1 \\ \vdots \\ w_n \end{bmatrix} = \begin{bmatrix} w_1 \\ \vdots \\ w_n \end{bmatrix}^T \begin{bmatrix} F_1 \\ \vdots \\ F_n \end{bmatrix}$$

<u>For Poisson</u>

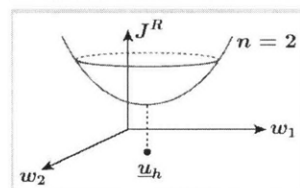
$$J(\vec{w}) = \frac{1}{2} \sum \sum w_i w_j \underbrace{\int \frac{\partial \varphi_i}{\partial x} \frac{\partial \varphi_j}{\partial x} dx}_{A_{ij}} - \sum w_j \int f \varphi_j dx$$

$$= \frac{1}{2} \vec{w}^T \mathbf{A} \vec{w} - \vec{w}^T \vec{F}_h$$

$$\nabla_w J(\vec{w}) = 0 \quad \text{to minimize}$$

$$\Rightarrow \mathbf{A}\vec{w} - \vec{F}_h = 0$$

If $\mathbf{A}$ is SPD $\quad \min_{\vec{w}} \frac{1}{2} \vec{w}^T \mathbf{A} \vec{w} - \vec{w}^T \vec{F}_h \quad$ $\vec{w}$ satisfies $\mathbf{A}\vec{w} = \vec{F}_h$



$\underline{u}_h = \arg\min_{w \in \Re^n} J^R(\underline{w})$

Expand $J(w = u_h + v)$; require $J(\underline{w}) > J(\underline{u}_h)$ unless $\underline{v} = 0$.

$$J^R(u_h+v) = \frac{1}{2}(u_h+v)^T A_h(u_h+v) - (u_h+v)^T F_h$$

$$= \tfrac{1}{2}u_h^T \underline{A}_h u_h - u_h^T \underline{F}_h + \tfrac{1}{2}\underline{v}^T A_h u_h + \tfrac{1}{2}u_h^T \underline{A}_h \underline{v} - \underline{v}^T \underline{F}_h + \tfrac{1}{2}\underline{v}^T \underline{A}_h \underline{v}$$

$$J^R(\underline{u}_h+\underline{v}) = J(\underline{u}) + \underbrace{(\underline{A}_h u_h - \underline{F}_h)}_{\nabla J^R(u_h)}{}^T \underline{v}\,\delta J_v^R(\underline{u}_h) \quad \text{SPD}$$

$$+ \tfrac{1}{2}\underbrace{\underline{v}^T \underline{A}_h \underline{v}}_{>0,\forall v\neq 0} \quad \text{SPD}$$

If (and only if)

$$\delta J_v^R(u_h) = 0, \quad \forall v \in \Re^n$$

$$\Updownarrow$$

$$\nabla J^R(u_h) = \underline{A}_h u_h - \underline{F}_h = 0$$

then

$$J(\underline{w} = \underline{u}_h + \underline{v}) > J(\underline{u}_h), \quad \forall \underline{v} \neq 0$$

$$\text{Poisson} \rightarrow \min_u J(u) \xrightarrow[u=\sum w_i\varphi_i]{} \min_{\vec{w}} J(\vec{w})$$

min min

$$\text{Hope } u \approx \sum w_i \varphi_i$$

Find $u_h \in \Re^n$ such that

$$A_h u_h = F_h \quad \Rightarrow \quad u_h(x) = \sum_{j=1}^{n} u_{hj}\varphi_j(x)$$

SPD ⇨ existence and uniqueness

❖ Error Analysis – Energy Norm

$$J(u_h) < J(w_h), \quad \forall w_h \in X_h, \quad w_h \neq u_h$$

If $e = u - u_h$

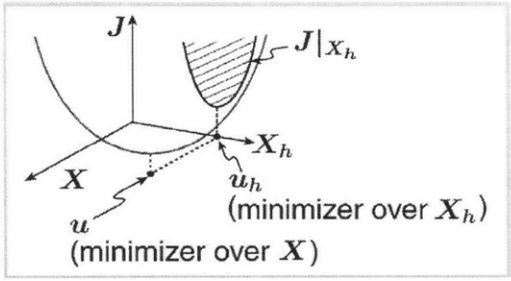
$$\left\| \underbrace{u - u_h}_{e} \right\| < \|u - w_h\|, \quad \forall w_h \in X_h, \quad w_h \neq u_h$$

and

$$\|e\| = \inf_{w_h \in X_h} \|u - w_h\|$$

where the energy norm is defined as: $\|e\| = \sqrt{\int_0^1 (e_x e_x + \alpha e e)dx}$

❖ Geometric Picture



GCR Preconditioners

$\mathbf{A}x = b$

$\mathbf{PA}x = \mathbf{P}b$ *preconditioner*

$\mathbf{P} \approx \mathbf{A}^{-1}$ Choose $\tilde{\mathbf{A}}$ s.t.

1.) $\tilde{\mathbf{A}} \approx \mathbf{A}$ e.g. $\tilde{\mathbf{A}} = diag(\mathbf{A}) = \mathbf{D}$

2.) $\mathbf{P} = \tilde{\mathbf{A}}^{-1}$ e.g. $\tilde{\mathbf{A}} = tridiag(\mathbf{A})$

easy to solve

$$\tilde{\mathbf{A}}^{-1}\mathbf{A}x = \tilde{\mathbf{A}}^{-1}b$$

$$\tilde{\mathbf{A}}^{-1}b = \tilde{b} \Longrightarrow b = \tilde{\mathbf{A}}\tilde{b}$$

$$\tilde{\mathbf{A}}^{-1}\mathbf{A}x = \tilde{\mathbf{A}}^{-1}b = \tilde{b}$$

$$\tilde{\mathbf{A}}^{-1}\mathbf{A}r_{k-1} = r_k \qquad \text{Solve } \tilde{\mathbf{A}}r_k = \mathbf{A}r_{k-1}$$

$\mathbf{P} = \mathbf{A}^{-1}$ ⇨ $\mathbf{A}^{-1}\mathbf{A}\,x = \mathbf{A}^{-1}b$ ⇨ $\mathbf{I}\,x = \mathbf{A}^{-1}b$ ← GCR converges in 1 iteration

Diagonal Preconditioners

Let $\mathbf{A} = \mathbf{D} + \mathbf{A}_{nd}$

Apply GCR to

$$\left(\mathbf{D}^{-1}\mathbf{A}\right)\vec{x} = \left(I + \mathbf{D}^{-1}\mathbf{A}_{nd}\right)\vec{x} = \mathbf{D}^{-1}\vec{b}$$

- The inverse of a diagonal is cheap to compute
- Usually improves convergence

Krylov Methods – Convergence Analysis

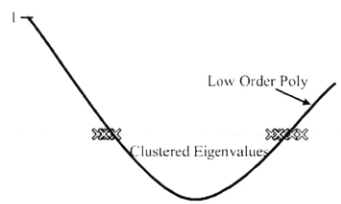
GCR Optimality Property:

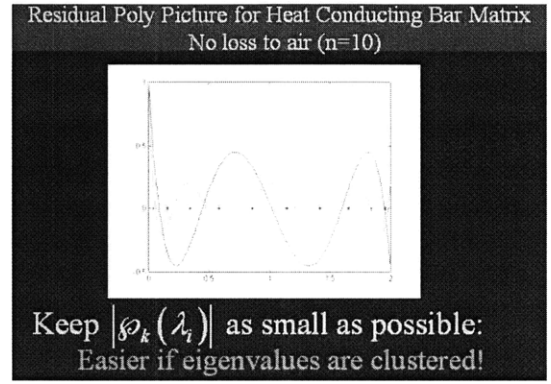
$$\left\|\vec{r}^{\,k}\right\| = \left\|\wp_k(\mathbf{M})\vec{r}^{\,0}\right\| \le \left\|\tilde{\wp}_k(\mathbf{M})\vec{r}^{\,0}\right\| \le \left\|\tilde{\wp}_k(\mathbf{M})\right\|\left\|\vec{r}^{\,0}\right\|$$

$\tilde{\wp}_k$ is any k^{th} order poly such that $\tilde{\wp}_k(0) = \mathbf{I}$

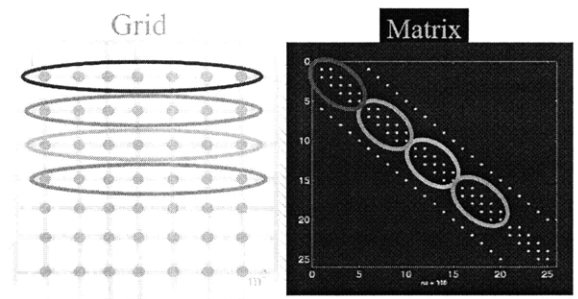
Therefore

Any polynomial which satisfies the constraint can be used to get an upper bound on $\dfrac{\left\|\vec{r}^{\,k}\right\|}{\left\|\vec{r}^{\,0}\right\|} \le \left|\tilde{\wp}_k(\mathbf{M})\right|$



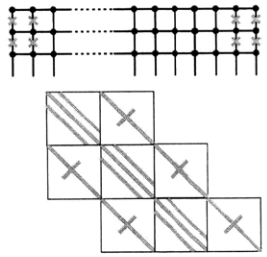


Block Diagonal Preconditioners



Tridiagonal Matrices factor quickly

Block Diagonal Preconditioners



$\tilde{\mathbf{A}}_B \approx \mathbf{A}$ (blocks)

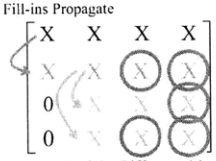
$\tilde{\mathbf{A}}_B$ is a different problem (no vertical interaction)

$\tilde{\mathbf{A}}_B$ is easy to solve (tridiagonal)

$\mathbf{P} = \tilde{\mathbf{A}}_B^{-1}$

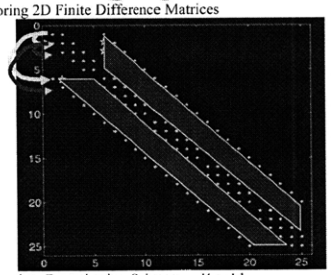
INCOMPLETE FACTORIZATION PRECONDITIONERS

- Sparse Matrices – Fill-in
  Fill-ins Propagate



Fill-ins from Step 1 results in Fill-ins in Step 2

- Factoring 2D Finite Difference Matrices



Generated Fill-in Makes Factorization Expensive

- Incomplete Factorization Schemes – Key Idea
  Throw away all fill-ins
  Throw away only fill-ins with small values

Throw away fill-ins produced by other fill-ins
Throw away fill-ins produced by fill-ins of other fill-ins, etc.

$$\begin{bmatrix} 4 & -1 & & -1 & & \\ -1 & 4 & \ddots & & \ddots & \\ & \ddots & \ddots & & & \\ -1 & & & 4 & -1 & \\ & \ddots & & -1 & 4 & \ddots \\ & & & & \ddots & \ddots \end{bmatrix}$$

m

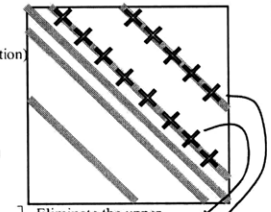
$\tilde{\mathbf{A}}_L = \mathbf{L} + \mathbf{D}$
Lower triangular plus diagonal

$\tilde{\mathbf{A}}_L$ is easy to solve (Forward Elimination)

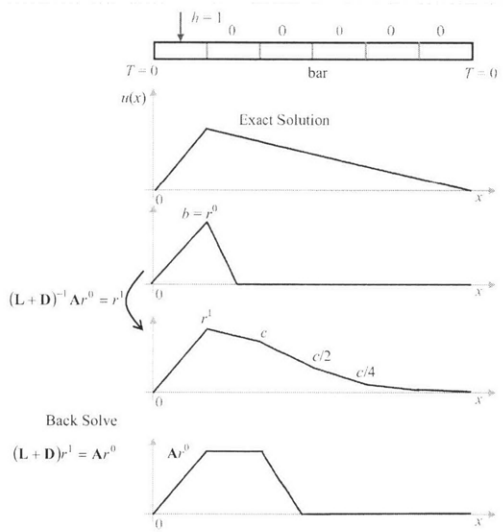
$\mathbf{P} = \tilde{\mathbf{A}}_L^{-1} = (L + D)^{-1}$

$$\mathbf{A} = \begin{bmatrix} 2 & -1 & & \\ -1 & 2 & -1 & \\ & -1 & 2 & \ddots \\ & & \ddots & \ddots \end{bmatrix} = \mathbf{L} + \mathbf{D} + \mathbf{U}$$

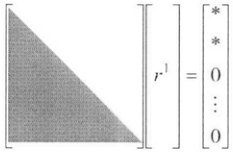
$$\rightarrow \mathbf{L} + \mathbf{D} = \begin{bmatrix} 2 & & & \\ -1 & 2 & & \\ & -1 & 2 & \\ & & \ddots & \ddots \end{bmatrix}$$



Eliminate the upper triangular part of the matrix



$$(\mathbf{L}+\mathbf{D})r^1 = \mathbf{A}r^0$$



$$\begin{bmatrix} \\ \\ \\ \end{bmatrix}\begin{bmatrix} r^1 \end{bmatrix} = \begin{bmatrix} * \\ * \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

Forward Elimination

Communication Improving Preconditioners: Gauss-Seidel

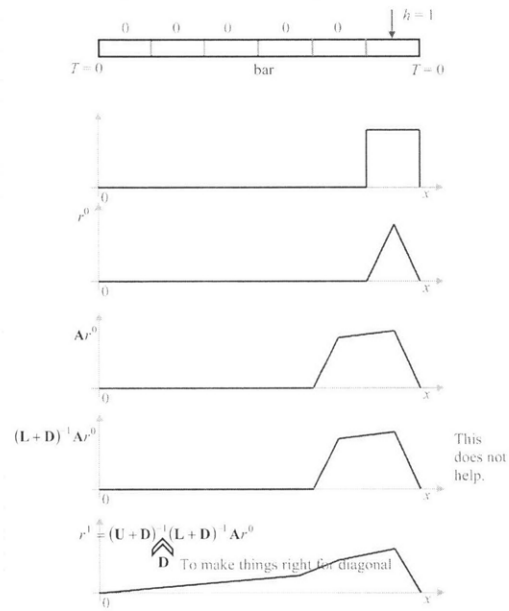
- Lower Bound

Communication Lower Bound for m gridpoints

$\mathbf{A}^k r^0$ is nonzero in m^{th} entry after $k = m$ iterations

Need at least m iterations for $\left(x^{k+1}\right)_m = \left(x^0 + \sum_{j=1}^{k} \alpha_j r^j\right)_m \neq 0$

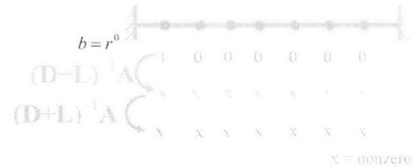
$$X_k \in span\{b, \mathbf{A}b, \mathbf{A}^2 b, \mathbf{A}^3 b, \ldots, \mathbf{A}^k b\}$$



- Faster Converging GCR
  GCR already achieves Communication Lower bound
  <u>Preconditioner must accelerate communication</u>
  Multiplying by Preconditioner must move values by more than one grid point.

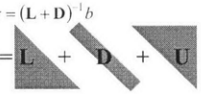
- Gauss-Seidel Preconditioning
  - Krylov Vectors

1-D Discretized PDE



$\mathbf{A}x = b$

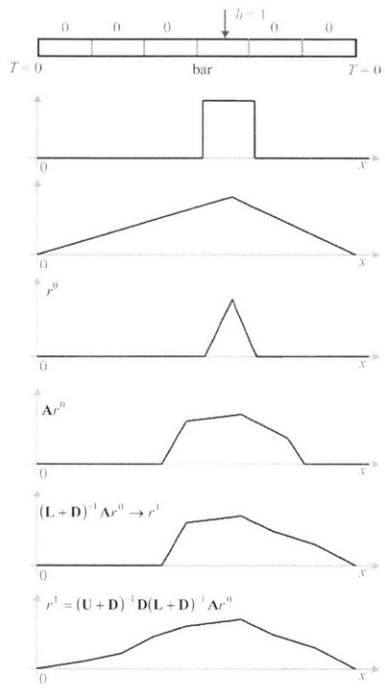
$(\mathbf{L}+\mathbf{D})^{-1}\mathbf{A}x = (\mathbf{L}+\mathbf{D})^{-1}b$



1.) $\widetilde{\mathbf{A}} \approx \mathbf{A} = \mathbf{L} + \mathbf{D} + \mathbf{U}$

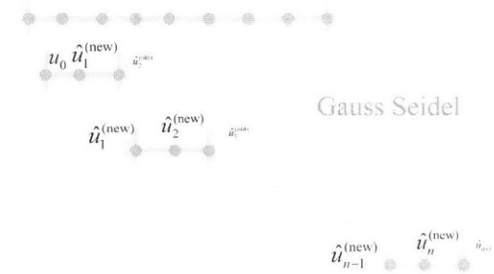
$\widetilde{\mathbf{A}} = \mathbf{L} + \mathbf{D}$ $\quad \widetilde{\mathbf{A}} = \begin{bmatrix} 2 & & & \\ -1 & 2 & & \\ & \ddots & \ddots & \\ & & -1 & 2 \end{bmatrix}$

2.) Easy to solve for $\widetilde{\mathbf{A}} = \mathbf{L} + \mathbf{D}$
Forward Elimination



- Physical View

1-D Discretized PDE



Gauss Seidel

Each Iteration of Gauss-Seidel Relaxation moves data across the grid

$$\mathbf{A}x = [\mathbf{L} + \mathbf{D} + \mathbf{U}]x = b$$

$$(\mathbf{L} + \mathbf{D})^{-1}\mathbf{A}x = (\mathbf{L} + \mathbf{D})^{-1}b$$

Lower Triangular
Apply with back solve

$$\mathbf{A}x = b$$

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix}$$

Lower Triangular
Apply with back solve

$$a_{11}u_1^{new} + a_{12}u_2^{old} + a_{13}u_3^{old} = b_1$$

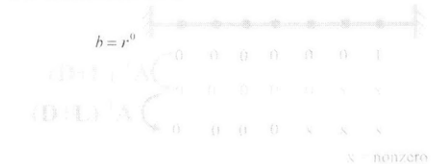
$$a_{21}u_1^{new} + a_{22}u_2^{new} + a_{23}u_3^{old} = b_2 \quad \Rightarrow (\mathbf{L} + \mathbf{D})u^{new} + \mathbf{U}u^{old} = b$$

$$a_{31}u_1^{new} + a_{32}u_2^{new} + a_{33}u_3^{new} = b_3$$

$$(\mathbf{L} + \mathbf{D})^{-1}\mathbf{A}x = (\mathbf{L} + \mathbf{D})^{-1}b$$

- Krylov Vectors

1-D Discretized PDE

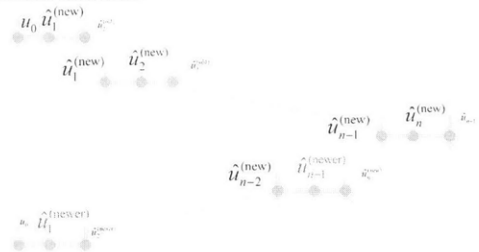


Gauss-Seidel communicates quickly in only one direction.

$$\underbrace{(\mathbf{L}+\mathbf{D})^{-1}}_{\tilde{\mathbf{M}}}\mathbf{A}x = (\mathbf{L}+\mathbf{D})^{-1}b$$

$$\overbrace{(\mathbf{U}+\mathbf{D})^{-1}\mathbf{D}(\mathbf{L}+\mathbf{D})^{-1}}^{P}\mathbf{A}x = (\mathbf{U}+\mathbf{D})^{-1}\mathbf{D}(\mathbf{L}+\mathbf{D})^{-1}b$$

- Symmetric Gauss-Seidel



This symmetric Gauss-Seidel Preconditioner Communicates both directions

Derivation of the SGS Iteration Equations

Forward Sweep (half step): $(\mathbf{D}+\mathbf{L})x^{k+\frac{1}{2}} + \mathbf{U}x^k = b$

Backward Sweep (half step): $(\mathbf{D}+\mathbf{U})x^k + \mathbf{L}x^{k+\frac{1}{2}} = b$

$$\Rightarrow x^{k+1} = (\mathbf{D}+\mathbf{U})^{-1}\mathbf{L}(\mathbf{D}+\mathbf{L})^{-1}\mathbf{U}x^k + (\mathbf{D}+\mathbf{U})^{-1}b\ldots$$
$$-(\mathbf{D}+\mathbf{U})^{-1}\mathbf{L}(\mathbf{D}+\mathbf{L})^{-1}b$$
$$\Rightarrow x^{k+1} = x^k - (\mathbf{D}+\mathbf{U})^{-1}\mathbf{D}(\mathbf{D}+\mathbf{L})^{-1}\mathbf{A}x^k + (\mathbf{D}+\mathbf{U})^{-1}\mathbf{D}(\mathbf{D}+\mathbf{L})^{-1}b$$

Step 1.) Pick $\tilde{\mathbf{A}}$ s.t.

a.) $\tilde{\mathbf{A}} \approx \mathbf{A}$

b.) $\tilde{\mathbf{A}}$ easy to factor (or solve) $\tilde{\mathbf{A}} = \tilde{\mathbf{L}}\tilde{\mathbf{U}}$

$\Longrightarrow \mathbf{P} = \tilde{\mathbf{A}}^{-1}$

e.g. incomplete factorization

Step 2.) $\mathbf{A}x = b$ becomes $\underbrace{\tilde{\mathbf{A}}^{-1}\mathbf{A}}x = \underbrace{\tilde{\mathbf{A}}^{-1}b}$

Solve $\tilde{\mathbf{A}}\tilde{b} = b$ use $\tilde{\mathbf{L}}\tilde{\mathbf{U}}\tilde{b} = b$

Step 3.) Do not calculate $\tilde{\mathbf{A}}^{-1}\mathbf{A}$

Instead at each iteration of GCR method

$[\tilde{\mathbf{A}}^{-1}\mathbf{A}]_{r_k} \rightarrow r_{k+1}$

Set-up system $\tilde{\mathbf{A}}r_{k+1} = \mathbf{A}r_k$ Use $\tilde{\mathbf{L}}\tilde{\mathbf{U}}r_{k+1} = \mathbf{A}r_k$

get r_{k+1}

Examples of GCR + Preconditioners for 3D FD

- Complexity of Sparse GCR + Preconditioner for 3D FD Methods

	matrix size	dense block	
$1-D$	$n = m$	$k = 5-10$	domain size $= m$
$2-D$	$n = m^2$	$k = 5-10$	
$3-D$	$n = m^3$	$k = 5-10$	

1-D	$O(kn) = 10\ O(m)$	$\leftarrow$	100 pt grid	$O(10^3)$ ops	1 μs
2-D	$O(kn) = 10\ O(m^2)$	$\leftarrow$	100×100 grid	$O(10^5)$ ops	0.1 ms
3-D	$O(kn) = 10\ O(m^3)$	$\leftarrow$	$100 \times 100 \times 100$ grid	$O(10^7)$ ops	10 ms

e.g. on a 1 GFlops computer using $m = 100$ and a constant $c = 10$

- Krylov – Work for Banded Gaussian Elimination, Relaxation, and GCR

Dimension	Banded GE	Sparse GE	GCR	GCR + preconditioner
1	$O(m)$	$O(m)$	$O(m^2)$	$O(km)$
2	$O(m^4)$	$O(m^3)$	$O(m^3)$	$O(km^2)$
3	$O(m^7)$	$O(m^6)$	$O(m^4)$	$O(km^3)$

GCR+good preconditioner is always the best choice for 2D & 3D problems!

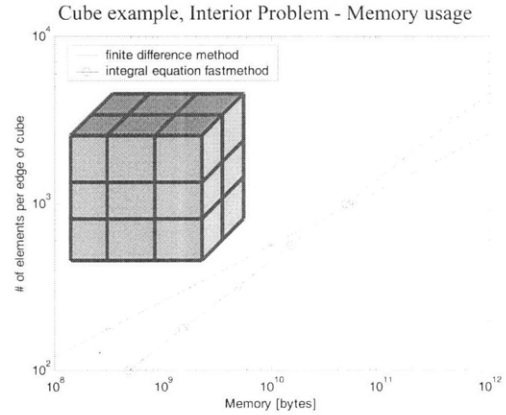
Dimension	Banded GE	Sparse GE	GCR	GCR + preconditioner
1	1 μs	1 μs	0.1 ms	1 μs
2	1 sec	10 ms	10 ms	0.1 sec
3	11d 6h	2h 46m	1 sec	10 ms

e.g. on a 1 GFlops computer using $\underline{m} = 1000$, k=10, and constant c=10
GCR+good preconditioner is always the best choice for 2D & 3D problems!

Dimension	Banded GE	Sparse GE	GCR	GCR + preconditioner
1	10 μs	10 μs	10 ms	0.1 ms
2	2h 46m	10 sec	10 sec	0.1 sec
3	317,000 years	317 years	2h 46m	1m 20s

e.g. on a 1 GFlops computer using $\underline{m} = 1000$, k=10, and constant c=10
GCR+good preconditioner (i.e. k = 5–10) is always the best choice for 2D & 3D problems!

- Cube Example, Interior Problem – Memory Usage



3.21 PDE - BEM Integral Equation Method I

INTRODUCTION TO NUMERICAL SIMULATION

LECTURE 22.
PDE – BEM Integral Equation Method I

TODAY'S OUTLINE:

- Poisson and Laplace Equations: examples
- Exterior Versus Interior Problems
- Green's Function
- Basis Functions
- Collocation Method
- Galerkin Method

OVERVIEW OF COURSE

Linear Systems: $\mathbf{M}x = b$ → Direct Methods – L.U. (Sparse); Iterative Solvers – G.C.R. (Dense)

Non-Linear Systems: $F(x) = 0$ Newton
- Damped
- Continuation
- Newton + GCR
- Matrix Implicit

Dynamics (ODE): $\frac{dx}{dt} = F(x(t)) + u(t)$ → B.E. T.R.; Periodic $u(t)$
- Shooting (time)
- Harmonic Balance (frequency)

PDE:
- $\nabla^2\Psi = 0$ Laplace
- $\nabla^2\Psi = -f(x)$ Poisson
- $\nabla^2\Psi + \kappa^2\Psi = 0$ Helmholtz

- BEM
- FEM
- FD

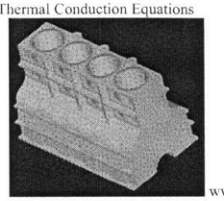
MOR:

$$\nabla^2\Psi + \kappa^2\Psi = -f \xrightarrow{\text{discretize}} \underbrace{\frac{dx}{dt} = \mathbf{A}x(t) + bu(t)}_{\text{ODE}} \xrightarrow{\text{M.O.R.}} \frac{d\hat{x}}{dt} = \hat{\mathbf{A}}\hat{x}(t) + \hat{b}\hat{u}(t)$$

1M eqns → 10 eqns

POISSON AND LAPLACE EQUATIONS: EXAMPLES

- Thermal Conduction Equations



www.adina.com

- Thermal Conduction Equations
  - The Poisson Partial Differential Equation.

$$\nabla^2\Psi(x) = \nabla\cdot\nabla\Psi(x) = -f(x) \qquad f(x) = \begin{cases} \neq 0 & \text{Poisson} \\ = 0 & \text{Laplace} \end{cases}$$

1D

$$\frac{d^2\Psi}{dx^2} = -f(x)$$

3D

$$\frac{\partial^2\Psi(x,y,z)}{\partial x^2} + \frac{\partial^2\Psi(x,y,z)}{\partial y^2} + \frac{\partial^2\Psi(x,y,z)}{\partial z^2} = -f(x,y,z)$$

Heat Flow (conduction)

$\Psi(x) \leftrightarrow$ temperature

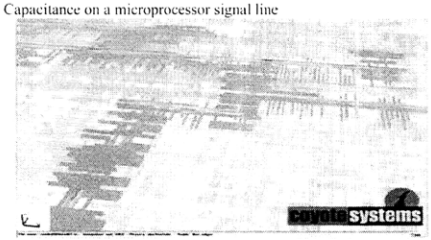
$\nabla\Psi(x) \propto$ heat flow

$$\nabla\cdot h = \nabla\cdot\underbrace{\nabla\Psi}_{\substack{\text{branch}\\ \text{equation}}} = 0 \leftrightarrow \text{Conservation Law}$$

$$\Leftrightarrow \sum h = 0$$

$$\int_V \nabla\cdot h\,dV = \oint_S h\cdot n\,dS$$

- Capacitance on a microprocessor signal line

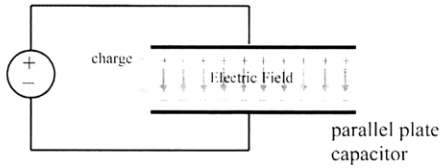


  - Electrostatic Analysis
    - The Laplace Partial Differential Equation.

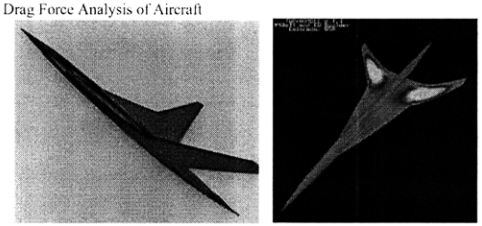
    Electrostatic Analysis

    $\Psi(x) \leftrightarrow$ electrostatic potential

    $\nabla\Psi(x) \propto$ electric field



- Drag Force Analysis of Aircraft



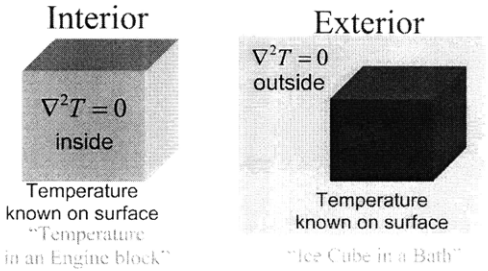
  - Potential Flow Equations

    - Poisson Partial Differential Equations

    Potential Flow

    $\nabla\Psi$ = air velocity

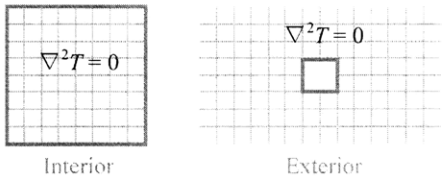
EXTERIOR VERSUS INTERIOR PROBLEMS

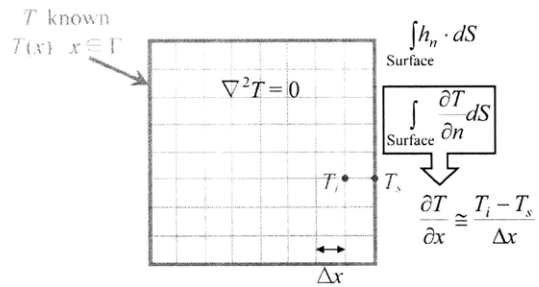


What is the heat flow?

$$\text{Heat Flow} = \text{Thermal Conductivity} \int_{surface} \frac{\partial T}{\partial n}$$

Finite Difference/Finite Element Method Meshing Problem



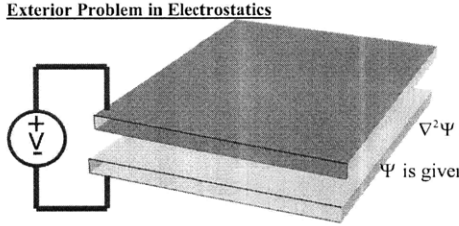


$$\int_{\text{Surface}} h_n \cdot dS$$

$$\int_{\text{Surface}} \frac{\partial T}{\partial n} dS$$

$$\frac{\partial T}{\partial x} \cong \frac{T_i - T_s}{\Delta x}$$

Exterior Problem in Electrostatics



potential

$\nabla^2 \Psi = 0$ Outside

Ψ is given on Surface

What is the capacitance?

$$\text{Capacitance} = \frac{\text{Dielectric}}{\text{Permitivity}} \int_{surface} \frac{\partial \Psi}{\partial n}$$

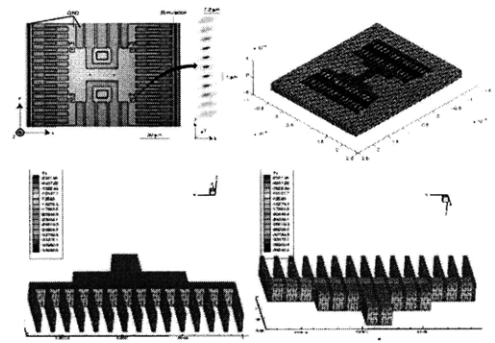
$\Psi(x) \leftrightarrow$ electrostatic potential

$\nabla \Psi(x) = \frac{\partial \Psi}{\partial n} \leftrightarrow$ electric field $\propto$ charge on surface $\leftrightarrow Q$

$$\text{capacitance} = \frac{Q}{V} = \frac{\text{charge}}{\text{voltage}} = Q_{|V=1} = \int_{\text{surface}} \frac{d\Psi}{dn}_{V=1} dS$$

$$\nabla^2 \Psi = -\frac{q}{\varepsilon}$$

Drag Force in a Microresonator



Comb Drive

Exterior Problem

Complicated fluid flow problem

Interested in Quality Factor at Resonance

Used as – sensors (e.g. gyroscope)

– RF resonator

Stokes Equation – Linear

– High viscosity

– Size of structure

Electromagnetic Coupling in a Package

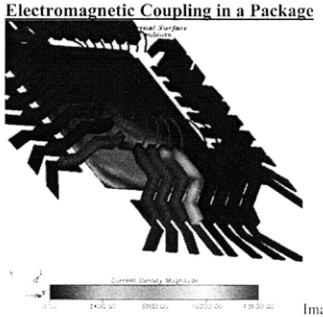


Image Thanks to Conventor

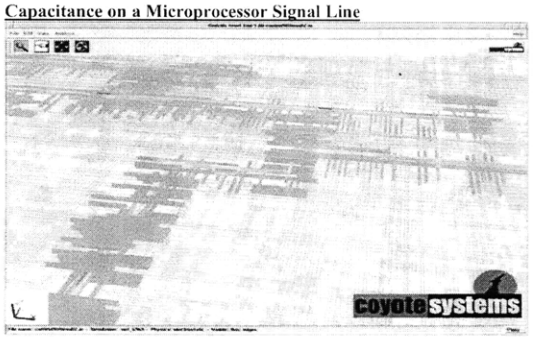
Full Wave

$\nabla^2\Psi + \kappa^2\Psi = 0$ Helmholtz Equation

$\nabla^2\Psi = 0$ Laplace (or Poisson) MQS or EMQS

Ψ = magnetic vector potential

Capacitance on a Microprocessor Signal Line



What is common about these problems

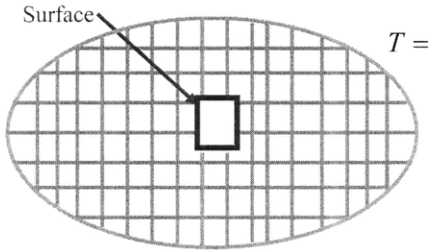
- Exterior Problems
  - Drag Force in MEMs device – fluid (air) creates drag.
  - Coupling in a Package – Field sin exterior create coupling
  - Capacitance of a Signal Line – Fields in exterior.
- Quantities of Interest are on the Surface
  - MEMs device – just want surface traction force
  - Package – just want coupling between conductors
  - Signal Line – just want surface charge
- Exterior Problem is Linear and Space-Invariant
  - MEMs – Exterior Stokes Flow equation (linear)
  - Package – Maxwell's equation in free space (linear)
  - Signal Line – Laplace's equation in free space (linear)

But problems are geometrically very complex!!

Exterior Problems

- Why not use Finite-Difference or FEM methods?

  2D Heat Flow Example



$T = 0$ at ∞

But, must truncate the mesh

Only need $\frac{\partial T}{\partial n}$ on the surface, but T is computed everywhere

Must truncate the mesh $\Rightarrow T(\infty) = 0$ becomes $T(R) = 0$

Green's Function

Laplace's Equation

- Green's Function

  Definition: Green's Function $G(x,x')$ is the solution at test location x produced by a unit point source at location x'

  e.g. for Laplace operator:

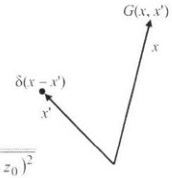
  $\nabla^2 G(x,x') = \delta(x - x')$

  - 2–D

$$\Psi(x,y) = \log\sqrt{(x-x_0)^2 + (y-y_0)^2}$$

$$\frac{\partial^2\Psi}{\partial x^2} + \frac{\partial^2\Psi}{\partial y^2} = 0 \quad \text{for all } (x,y) \neq (x_0, y_0)$$

$$G(x,x') = \frac{1}{2\pi}\log|x-x'|$$



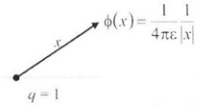
- 3–D

$$\Psi(x,y,z) = \frac{1}{\sqrt{(x-x_0)^2 + (y-y_0)^2 + (z-z_0)^2}}$$

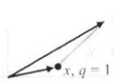
$$\frac{\partial^2\Psi}{\partial x^2} + \frac{\partial^2\Psi}{\partial y^2} + \frac{\partial^2\Psi}{\partial z^2} = 0 \quad \text{for all } (x,y,z) \neq (x_0, y_0, z_0)$$

$$G(x,x') = -\frac{1}{4\pi}\frac{1}{|x-x'|}$$

Proof: Just differentiate and see!



$$\phi(x) = \frac{1}{4\pi\varepsilon}\frac{1}{|x|}$$



$$\phi(x) = \frac{1}{4\pi\varepsilon}\frac{1}{|x-x'|}$$

$$\nabla^2\phi = \frac{-\delta(x-x')}{\varepsilon}$$

$$G(x,x') = -\frac{1}{4\pi}\frac{1}{|x-x'|}$$

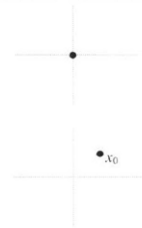
$$\nabla^2 G = \delta(x-x')$$

$$\nabla^2\Psi = \delta(x)$$

$$\nabla^2\Psi = 0 \quad x \neq 0$$

Green Function

$$\Psi(x) = \begin{cases} \log|x| & \text{2D} \\ 1 & \text{3D} \\ |x| & \end{cases}$$



$$\Psi(x) = \log|x-x_0|$$

$$\Psi(x) = \frac{1}{|x-x_0|}$$

Think Electrostatics

$$\nabla^2\phi = \delta(x)$$

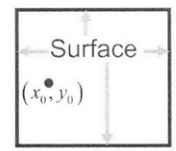
$$\phi(x) = \frac{1}{4\pi\varepsilon|x|}$$

$$\nabla^2\phi = \delta(x-x_0)$$

$$\phi(x) = \frac{1}{4\pi\varepsilon|x-x_0|} = \frac{1}{4\pi\varepsilon}\frac{1}{\sqrt{(x-x_0)^2 + (y-y_0)^2 + (z-z_0)^2}}$$

- Simple Idea (in 2-D)
  - One Point

$\Psi(x,y)$ is given on surface



$$\frac{\partial^2\Psi}{\partial x^2} + \frac{\partial^2\Psi}{\partial y^2} = 0 \quad \text{outside}$$

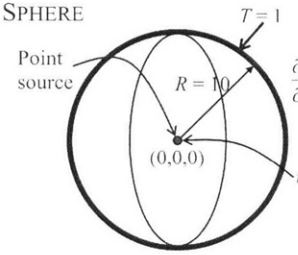
Let $\Psi(x,y) = \log\sqrt{(x-x_0)^2 + (y-y_0)^2}$

$$\frac{\partial^2\Psi}{\partial x^2} + \frac{\partial^2\Psi}{\partial y^2} = 0 \quad \text{outside}$$

~~Problem Solved~~

Does not match boundary conditions!

SPHERE

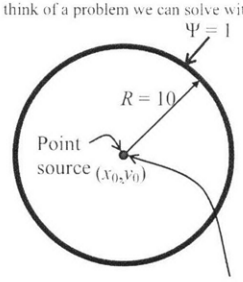


$$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} + \frac{\partial^2 u}{\partial z^2} = 0$$

$$u = \frac{10}{\sqrt{x^2 + y^2 + z^2}}$$

CIRCLE

Can you think of a problem we can solve with just one point source?



Outside:

$$\nabla^2 \Psi = 0$$

$$\Leftrightarrow \frac{\partial^2 \Psi}{\partial x^2} + \frac{\partial^2 \Psi}{\partial y^2} = 0$$

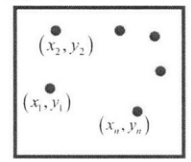
$$\Psi = A \log \sqrt{(x - x_0)^2 + (y - y_0)^2}$$

$$A = \frac{1}{\log R}$$ Weighted point source

Note: with one point source I can match only one B.C. For more complicated B.C. I could use more point sources, how many point sources do I need? What do I put them?

- "More Points"

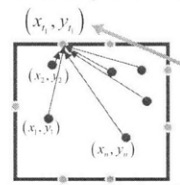
$\Psi(x, y)$ is given on surface



$$\frac{\partial^2 \Psi}{\partial x^2} + \frac{\partial^2 \Psi}{\partial y^2} = 0 \quad \text{outside}$$

Let $\Psi(x, y) = \sum_{j=1}^{n} \omega_j \log \sqrt{(x - x_j)^2 + (y - y_j)^2} = \sum_{j=1}^{n} \omega_j G(x - x_j, y - y_j)$

Pick the ω_j's to match the boundary conditions!



Source Strengths selected to give correct potential at test points.

$$\begin{bmatrix} G(x_{t_1} - x_1, y_{t_1} - y_1) & \cdots & \cdots & G(x_{t_1} - x_n, y_{t_1} - y_n) \\ \vdots & & & \vdots \\ G(x_{t_n} - x_1, y_{t_n} - y_1) & \cdots & \cdots & G(x_{t_n} - x_n, y_{t_n} - y_n) \end{bmatrix} \begin{bmatrix} \omega_1 \\ \vdots \\ \omega_n \end{bmatrix} = \begin{bmatrix} \Psi(x_{t_1}, y_{t_1}) \\ \vdots \\ \Psi(x_{t_n}, y_{t_n}) \end{bmatrix}$$

$$\Psi(x_t, y_t) = w_1 G(x_t - x_1, y_t - y_1) + w_2 G(x_t - x_2, y_t - y_2) + \cdots$$

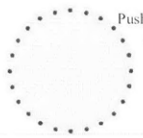
Source Point Locations

If Cluster of Charges in Center:

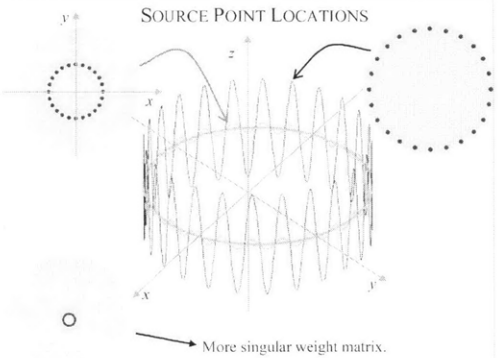
- Smaller oscillations
- Less change to B.C.
- Matrix is ill conditioned
  (if all on top of one another – singular matrix)

Push Charges to Surface:

- To avoid ill conditioning
- Use many points to reduce oscillation amplitude



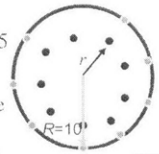
Source Point Locations



More singular weight matrix.

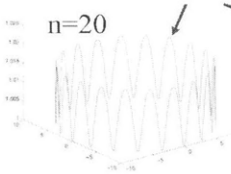
- Computational Results using Points Approach

Circle with Charges $r=9.5$

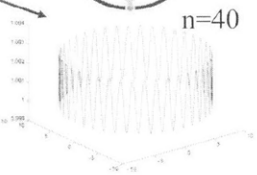


Potentials on the Circle

n=20



n=40



$$\Psi(x) = \sum_i \overset{\text{unknown}}{w_i} G(x, x_i)$$

$$\underbrace{\Psi(x)_{surface}}_{\text{given}} = \int_{surface} \underbrace{\sigma(x')}_{\text{unknown}} G(x, x')\, dx' \quad \Leftrightarrow \nabla^2 \Psi = 0 + \underbrace{\text{B.C}}_{\Psi(x)=\Psi(x)_{surface}}$$

- Charge Density

  Want to smear point charges to the surface

Results in an Integral Equation

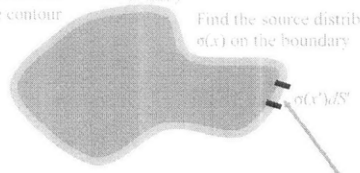
$$\Psi(x) = \int_{surface} G(x, x')\sigma(x')\, dS'$$

How do we solve the integral equation?

❖ Integral Formulation

Given solution on the contour $\Psi(x)_{\text{boundary}}$

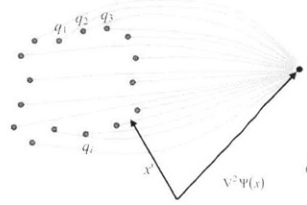
Find the source distribution $\sigma(x)$ on the boundary



Using linear superposition results in an Integral Equation

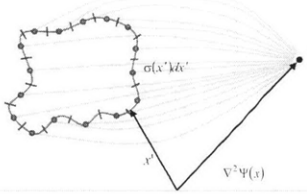
$$\Psi(x)_{\text{boundary}} = \int_{\text{boundary}} G(x,x')\sigma(x')dx'$$

How do we solve the integral equation?



$$\Psi(x) = \sum_{i=1}^{n} G(x,x')q_i$$

$$G(x,x') = \begin{cases} \frac{1}{2\pi}\log|x-x'| & 2D \\ \frac{1}{4\pi}\frac{1}{|x-x'|} & 3D \end{cases}$$



$$\Psi(x) = \int_{\text{boundary}} G(x,x')\sigma(x')dx'$$

$$\nabla^2\Psi + \kappa^2\Psi = 0$$

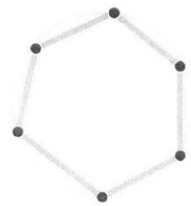
$$G(x,x') = \frac{1}{4\pi}\frac{e^{i\kappa|x-x'|}}{|x-x'|}$$

PDE Problem	IE Problem
$\nabla^2\Psi = 0$ $\Psi(x)_{\text{Boundary}} = \text{Given}$ (find $\sigma(x)$)	$\underbrace{\Psi(x)_{\text{Boundary}}}_{\text{Given}} = \int_{\text{boundary}} G(x,x')\underbrace{\sigma(x')}_{\text{unknown}}dx'$ $G(x,x') = \frac{1}{4\pi}\frac{1}{\lvert x-x'\rvert}$ in 3D

BASIS FUNCTIONS

Basic Idea

Represent $\sigma(x) = \sum_{i=1}^{n} \omega_i \underbrace{\varphi_i(x)}_{\text{Basis Functions}}$

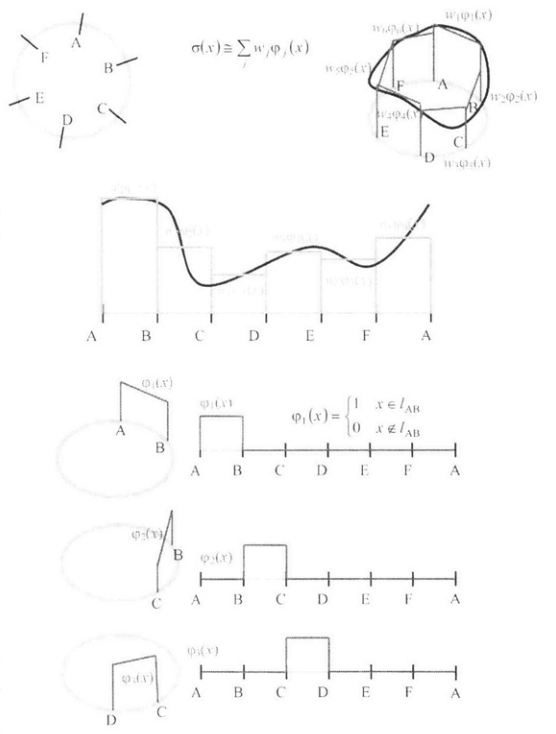


Example Basis

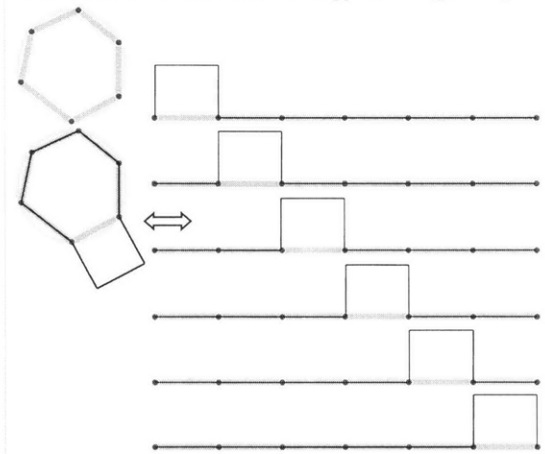
- Represent a circle with straight lines
- Assume σ is constant along each line

The basis functions are "on" the surface

- Can be used to approximate the density
- May also approximate the geometry

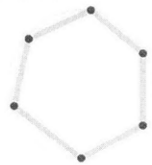


Basis Functions for a Circle – *unwrapped circle geometry*

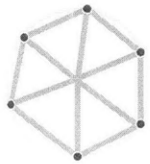


Geometric Approximation

- Not a New Idea



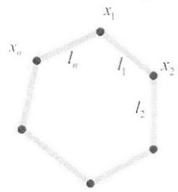
Piecewise Straight surface basis functions approximate the circle



Triangles for 2-D FEM approximate the circle too!

$$\Psi(x) = \int_{\text{approx surface}} G(x,x') \sum_{i=1}^{n} \omega_i \varphi_i(x') dS'$$

- Piecewise Constant Straight Sections Example



Represent $\sigma(x) = \sum_{i=1}^{n} \omega_i \underbrace{\varphi_i(x)}_{\text{Basis Functions}}$

1.) Pick a set of n points on the surface
2.) Define a new surface by connecting points with n lines
3.) Define $\varphi_j(x) = 1$ if x is on line l_j otherwise, $\varphi_j(x) = 0$

$$Q = \int_l \sigma(x) dx = \int_l \sum_{i=1}^{n} \omega_i \varphi_i(x)$$

$$\Psi(x) = \int_l G(x,x') \sum_{i=1}^{n} \omega_i \varphi_i(x') dx'$$

$$\Psi(x)\big|_{\text{boundary}} = \int_{\text{boundary}} G(x,x')\sigma(x')dx'$$

$$= \int_{\text{approximate boundary}} G(x,x') \sum_{j=1}^{n} \omega_j \varphi_j(x') dx'$$

$$= \sum_{j=1}^{n} \omega_j \int_{\text{boundary}} G(x,x')\varphi_j(x')dx'$$

$$= \sum_{j=1}^{n} \omega_j \int_{\text{line } l_j} G(x,x')dx'$$

How do we determine the ω_j's?

$$\nabla^2 \Psi = 0 + \underbrace{\Psi(x)_{surface}}_{\text{B.C.}}$$

Basis Functions $\sigma(x) \cong \sum_{j=1}^{n} w_j \varphi_j(x)$

Integral Equation $\Psi(x)_{surface} = \int_{surface} G(x,x')\sigma(x')dS'$

Substitute Basis Functions $\Psi(x)\big|_{surface} \cong \int_{surface} G(x,x') \sum_{j=1}^{n} w_j \varphi_j(x') dS'$

$$\cong \sum_{j=1}^{n} w_j \int_{surface} G(x,x')\varphi_j(x')dS'$$

If basis functions are piecewise constant $\underbrace{\Psi(x)_{surface}}_{given} \cong \sum_{j=1}^{n} w_j \int_{l_j} G(x,x')dS'$

$$\varphi_j(x') = \begin{cases} 1 & \text{if } x' \in l_j \\ 0 & \text{if } x' \notin l_j \end{cases}$$

unknowns
We are going to try and find the weights w_j

Q1. How do I find w_j?

Q2. Given w_j, what is the charge on the surface?

$$\sigma(x) = \sum_{j=1}^{n} w_j \varphi_j(x)$$

Q3. Given w_j, what is the potential anywhere outside?

$$\varphi(x) = \sum_{j=1}^{n} w_j \int_{t_j} G(x, x')dS'$$

Can evaluate this anywhere outside if we evaluate on the surface we obtain the given B.C.

Residuals

- Residual Definition

$$R(x) \equiv \Psi(x)_{boundary} - \int_{\substack{approx \\ surface}} G(x,x') \sum_{i=1}^{n} \omega_i \varphi_i(x') dS'$$

- Residual Minimization

We will pick the ω_i's to make $R(x)$ small.

General Approach : Pick a set of test functions $\phi_1, \phi_2, \ldots, \phi_n$ and force $R(x)$ to be orthogonal to the set $\int \phi_i(x) R(x) dS = 0$ for all i.

$\varphi_i = ?$

$$\int_{surface} \varphi_i(x) R(x) = 0 \rightarrow R(x_{t_i}) = 0 \qquad i = 1,2,\ldots,N$$

$$\int_{surface} \delta(x - x_{t_i}) R(x) = R(x_{t_i})$$

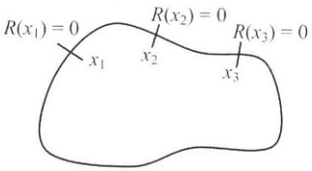
- Residual Minimization Using Test Functions

$$\int \phi_i(x) R(x) dS = \int \phi_i(x) \Psi(x) dS - \int_{\substack{approx \\ surface}} \int \phi_i(x) G(x,x') \sum_{j=1}^{n} \omega_j \varphi_j(x') dS' dS = 0$$

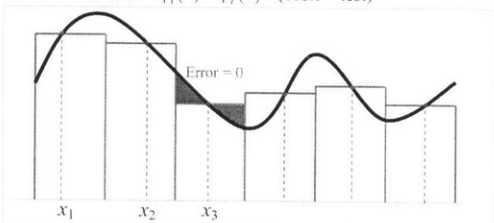
We will generate the different methods by choosing the $\phi_1, \phi_2, \ldots, \phi_n$

Collocation : $\phi_i(x) = \delta(x - x_{t_i})$ (point - matching)

$\varphi_i(x) = \delta(x - x_i) \quad \Rightarrow R(x_i) = 0$

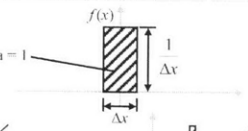


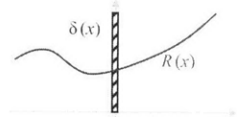
Galerkin Method : $\phi_i(x) = \varphi_i(x)$ (basis = test)

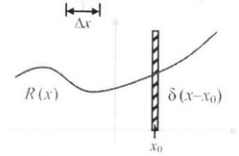


$$\delta(x) = \begin{cases} 0 & x \neq 0 \\ 1 & x = 1 \end{cases}$$

$$\int \delta(x) dx = 1$$







$$\int R(x)\delta(x) dx = R(0)$$

$$\int R(x)\delta(x - x_0) dx = R(x_0)$$

COLLOCATION METHOD

$w_j \quad j = 1, 2, \ldots, n$

$$\varphi(x_{t_i})_{surface} = \sum_{j=1}^{n} w_j \underbrace{\int_{i_j} G(x_{t_i}, x')dS'}_{A_{ij}}$$

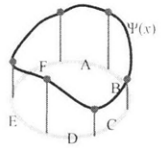
$$[\mathbf{A}][w] = [\varphi(x_{t_i})]$$

Collocation : $\phi_i(x) = \delta(x_i)$ (point - matching)

$$\int \delta(x - x_{t_i}) R(x) dS = R(x_{t_i}) = \Psi(x_{t_i}) - \int_{\substack{approx \\ surface}} G(x_{t_i}, x') \sum_{j=1}^{n} \omega_j \varphi_j(x') dS' = 0$$

$$\Rightarrow \Psi(x_{t_i}) = \sum_{j=1}^{n} \omega_j \underbrace{\int_{\substack{approx \\ surface}} G(x_{t_i}, x') \varphi_j(x') dS'}_{A_{i,j}}$$

$$\begin{bmatrix} A_{1,1} & \cdots & A_{1,n} \\ \vdots & \ddots & \vdots \\ A_{n,1} & \cdots & A_{n,n} \end{bmatrix} \begin{bmatrix} \omega_1 \\ \vdots \\ \omega_n \end{bmatrix} = \begin{bmatrix} \Psi(x_{t_1}) \\ \vdots \\ \Psi(x_{t_n}) \end{bmatrix}$$



$R(x_i) = 0 \qquad i = 1, 2, \ldots, N$

$$\Psi(x)_{surface} = \int_{surface} G(x, x')\sigma(x')dS'$$

Basis Functions
$\sigma(x) \cong \sum w_j \varphi_j(x)$

$$\Psi(x)_{surface} = \sum_{j=1}^{n} w_j \int_{surface} G(x, x')\sigma_j(x')dS'$$

Collocation

$$\underbrace{\Psi(x_{t_i})}_{b_i} = \sum_{j=1}^{n} w_j \underbrace{\int_{surface} G(x_{t_i}, x')\sigma_j(x')dS'}_{A_{ij}}$$

Piecewise constant basis functions

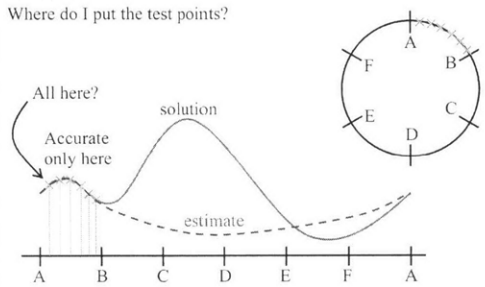
$$\underbrace{\Psi(x_{t_i})}_{b_i} = \sum_{j=1}^{n} w_j \underbrace{\int_{i_j} G(x_{t_i}, x')dS'}_{A_{ij}}$$

test source

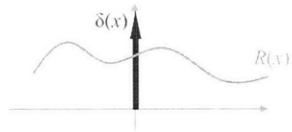
$$\sum_{j=1}^{n} w_j \underbrace{\int_{i_j} G(x_i, x')dx'}_{A_{ij}} = \underbrace{\Psi(x_i)}_{given} \quad i = 1, 2, \ldots, N$$

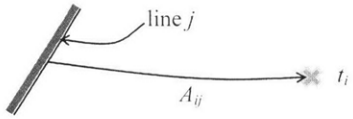
$$\left[A_{ij} = \int_{i_j} G(x_i, x')dx' \right] \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_n \end{bmatrix} = \begin{bmatrix} \Psi(x_1) \\ \Psi(x_2) \\ \vdots \\ \Psi(x_n) \end{bmatrix}$$

Where do I put the test points?

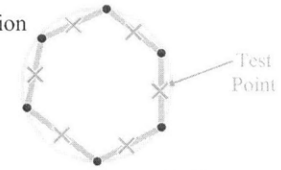


Also ill-conditioned

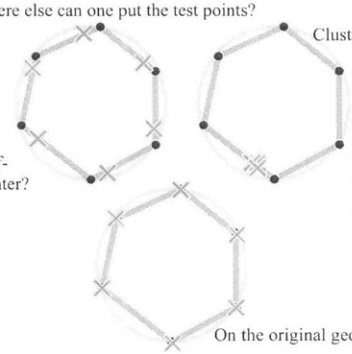




Collocation



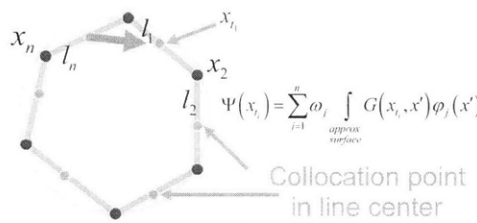
Where else can one put the test points?



Clustered in one area?
Test points should be spread out because you are solving the integral equation exactly at the test points – so clustered test points will solve the integral equation exactly at that location but nowhere else.

Centroid Collocation

- Piecewise Constant Bases



$$\Psi(x_{t_i}) = \sum_{j=1}^{n} \omega_j \int_{\substack{approx\\surface}} G(x_{t_i}, x')\varphi_j(x')dS'$$

$$\begin{bmatrix} A_{1,1} & \cdots & A_{1,n} \\ \vdots & \ddots & \vdots \\ A_{n,1} & \cdots & A_{n,n} \end{bmatrix} \begin{bmatrix} \omega_1 \\ \vdots \\ \omega_n \end{bmatrix} = \begin{bmatrix} \Psi(x_{t_1}) \\ \vdots \\ \Psi(x_{t_n}) \end{bmatrix}$$

$$\Psi(x_{t_i}) = \sum_{j=1}^{n} \omega_j \underbrace{\int_{line\,j} G(x_{t_i}, x')\, dS'}_{A_{i,j}}$$

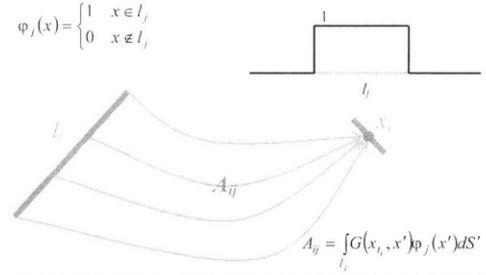
- Generates a Nonsymmetric A

$$\Psi(x_{t_i}) = \sum_{j=1}^{n} \omega_j \underbrace{\int_{line\,j} G(x_{t_i}, x')\, dS'}_{A_{i,j}}$$

$$A_{1,2} = \int_{line\,2} G(x_{t_1}, x')\, dS' \neq \int_{line\,1} G(x_{t_2}, x')\, dS' = A_{2,1}$$

test source test source

$$\varphi_j(x) = \begin{cases} 1 & x \in l_j \\ 0 & x \notin l_j \end{cases}$$



Galerkin Method

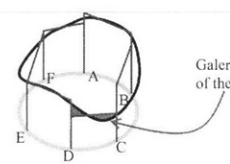
Galerkin : $\varphi_i(x) = \varphi_i(x)$ (test = basis)

$$\int \varphi_i(x)R(x)dS = \int \varphi_i(x)\Psi(x)dS - \int_{\substack{approx\\surface}} \int \varphi_i(x)G(x,x')\sum_{j=1}^{n}\omega_j\varphi_j(x')dS'dS = 0$$

$$\underbrace{\int_{\substack{approx\\surface}} \varphi_i(x)\Psi(x)dS}_{b_i} = \sum_{j=1}^{n}\omega_j \underbrace{\int_{\substack{approx\\surface}}\int_{\substack{approx\\surface}} G(x,x')\varphi_i(x)\varphi_j(x')dS'dS}_{A_{i,j}}$$

$$\begin{bmatrix} A_{1,1} & \cdots & A_{1,n} \\ \vdots & \ddots & \vdots \\ A_{n,1} & \cdots & A_{n,n} \end{bmatrix} \begin{bmatrix} \omega_1 \\ \vdots \\ \omega_n \end{bmatrix} = \begin{bmatrix} b_1 \\ \vdots \\ b_n \end{bmatrix}$$

If $G(x,x') = G(x',x)$ then $\mathbf{A}_{i,j} = \mathbf{A}_{j,i}$ $\mathbf{A}$ is symmetric



Galerkin sets the weighted average of the residual (error) to zero

$$\int R(x)\varphi_i(x)dx = 0 \quad i = 1,2,\ldots,N$$

$$\Psi(x)_{surface} = \int_{surface} G(x,x')\sigma(x')dS'$$

Basis Functions

$$\sigma(x') \cong \sum w_j \varphi_j(x')$$

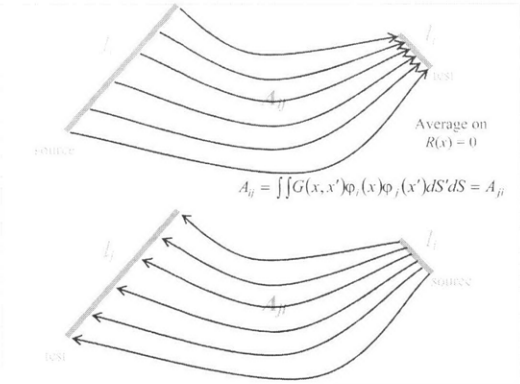
$$\Psi(x)_{surface} = \sum_{j=1}^{n} w_j \int_{surface} G(x,x')\sigma_j(x')dS'$$

Galerkin

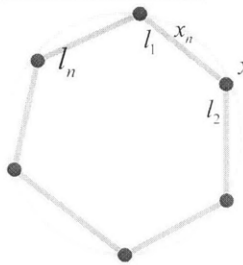
$$\int_s \Psi(x)_s \varphi_i(x)dS = \sum_{j=1}^{n} w_j \int_s\int_s G(x,x')\varphi_j(x')\varphi_j(x)dS'dS$$

$$\underbrace{\int_{l_i} \Psi(x)dS}_{b_i} = \sum_{j=1}^{n} w_j \underbrace{\int_{l_i}\int_{l_i} G(x,x')dS'dS}_{A_{ij}}$$

Piecewise constant basis functions



Piecewise Constant Bases



$$\underbrace{\int_{line_i} \Psi(x)dS}_{b_i} = \sum_{j=1}^{n} \omega_j \underbrace{\int_{line_i}\int_{line_j} G(x,x')\,dS'dS}_{A_{i,j}}$$

$$\begin{bmatrix} A_{1,1} & \cdots & A_{1,n} \\ \vdots & \ddots & \vdots \\ A_{n,1} & \cdots & A_{n,n} \end{bmatrix} \begin{bmatrix} \omega_1 \\ \vdots \\ \omega_n \end{bmatrix} = \begin{bmatrix} b_1 \\ \vdots \\ b_n \end{bmatrix}$$

3.22 PDE - BEM Integral Equation Method II

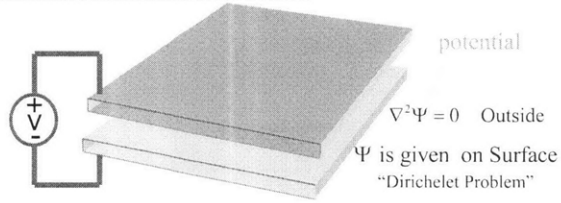
INTRODUCTION TO NUMERICAL SIMULATION

LECTURE 23.
PDE – BEM Integral Equation Method II

TODAY'S OUTLINE:

- Collocation Method
- Galerkin Method
- 3D Panel Integration
- Solving Discretized Integral Equations

EXTERIOR PROBLEM IN ELECTROSTATICS



- First Kind Integral Equation for Charge:

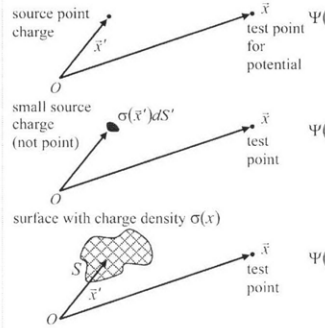
$$\Psi(x) = \int_{surface} \underbrace{\frac{1}{\|x - x'\|}}_{Green's\ Function} \underbrace{\sigma(x')}_{Charge\ Density} dS'$$

potential

First Kind Integral Equation for Charge

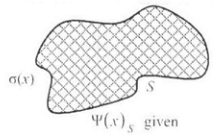
$$\nabla^2\Psi = 0 \qquad \Rightarrow \Psi(\vec{x}) = \int_S \frac{1}{\|\vec{x} - \vec{x}'\|}\sigma(\vec{x}')dS' \qquad \text{in 3D}$$

one way to view this is as a superposition integral



$$\nabla^2\Psi = 0 + \underset{B.C.}{\Psi(\vec{x})_S} \qquad \underbrace{\Psi(\vec{x})_S}_{\substack{\text{known} \\ \text{boundary condition}}} = \int_S \frac{\overset{\text{unknown}}{\sigma(\vec{x}')}}{\|\vec{x} - \vec{x}'\|} dS'$$

Problem.



$\nabla^2\Psi = 0$ everywhere outside

Find $\sigma(x)$ on S

Solve $\underbrace{\Psi(\vec{x})_{\vec{x}\in S}}_{\text{given}} = \int_S \frac{1}{\|\vec{x}-\vec{x}'\|}\,\sigma(\vec{x}')dS'$

Use Basis Functions $\sigma(\vec{x}') \cong \sum w_j \varphi_j(\vec{x}')$

$$\Psi(\vec{x})_{\vec{x}\in S} = \int_S \frac{1}{\|\vec{x}-\vec{x}'\|}\sum_{j=1}^{N} w_j \varphi_j(\vec{x}')dS'$$

e.g. piecewise constant basis (panels)

$$\Psi(\vec{x})_{\vec{x}\in S} = \sum_{j=1}^{N} w_j \int_{panel\ j} \frac{1}{\|\vec{x}-\vec{x}'\|}\,dS'$$

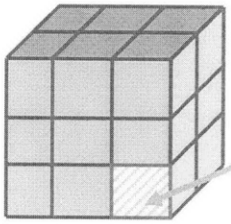
N unknowns

$$\varphi_j(\vec{x}') = \begin{cases} 1 & \text{if } \vec{x} \in panel\ j \\ 0 & \text{otherwise} \end{cases}$$

BASIS FUNCTION APPROACH

- Piecewise Constant Basis Function

Integral Equation: $\Psi(x) = \int_{surface} \frac{1}{\|x-x'\|}\sigma(x')\,dS'$



Discretize Surface into Panels

Represent $\sigma(x) \approx \sum_{j=1}^{n} \omega_j \varphi_j(x)$

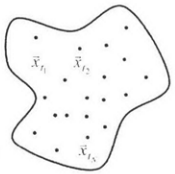
Panel j

$\varphi_j(x) = 1$ if x is on panel j

$\varphi_j(x) = 0$ otherwise

Collocation.

$$N \text{ equations} \begin{cases} \Psi(\vec{x}_{t_1})_{\vec{x}_{t_1}\in S} = \sum_{j=1}^{N} w_j \int_{S_j} \frac{1}{\|\vec{x}_{t_1}-\vec{x}\|}\,dS' \\ \Psi(\vec{x}_{t_2})_{\vec{x}_{t_2}\in S} = \sum_{j=1}^{N} w_j \int_{S_j} \frac{1}{\|\vec{x}_{t_2}-\vec{x}\|}\,dS' \\ \vdots \\ \Psi(\vec{x}_{t_N})_{\vec{x}_{t_N}\in S} = \sum_{j=1}^{N} w_j \int_{S_j} \frac{1}{\|\vec{x}_{t_N}-\vec{x}\|}\,dS' \end{cases}$$

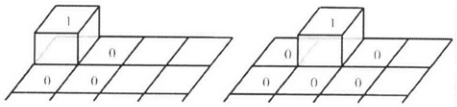


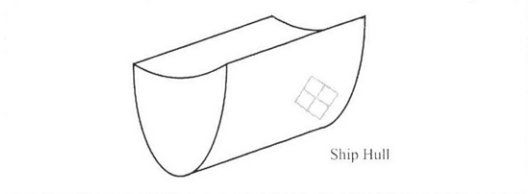
$$\begin{bmatrix} \int_{S_1} \frac{1}{\|x_{t_1}-x'\|}dS' & \int_{S_2} \frac{1}{\|x_{t_1}-x'\|}dS' & \\ & & \\ & & \end{bmatrix} \begin{bmatrix} w_1 \\ \vdots \\ w_N \end{bmatrix} = \begin{bmatrix} \Psi(\vec{x}_{t_1})_{\vec{x}_{t_1}\in S} \\ \vdots \\ \Psi(\vec{x}_{t_N})_{\vec{x}_{t_N}\in S} \end{bmatrix}$$

Piecewise Constant Basis Functions + Collocation.

$$A_{i,j} = \int_{S_j} \frac{1}{\|x_{t_i}-x'\|}\,dS'$$

Basis Functions





1.) Discretize Unknowns $\sigma(x) \rightarrow N$ unknowns using basis functions
2.) Discretize Equations → Collocation or Galerkin

- Residual Definition

$$R(x) \equiv \Psi(x) - \int_{\substack{approx \\ surface}} G(x,x') \sum_{i=1}^{n} \omega_i \varphi_i(x') dS'$$

- Residual Minimization

We will pick the ω_i's to make $R(x)$ small.

General Approach : Pick a set of test functions $\phi_1, \phi_2, \ldots, \phi_n$ and force $R(x)$ to be orthogonal to the set $\int \phi_i(x) R(x) dS = 0$ for all i.

- Residual Minimization Using Test Functions

$$\int \phi_i(x) R(x) dS = \int \phi_i(x) \Psi(x) dS - \int_{\substack{approx \\ surface}} \phi_i(x) G(x,x') \sum_{j=1}^{n} \omega_j \varphi_j(x') dS' dS = 0$$

We will generate the different methods by choosing the $\phi_1, \phi_2, \ldots, \phi_n$

Collocation : $\phi_i(x) = \delta(x - x_{t_i})$ (point - matching)

Collocation.

$\vec{x}_{t_1} \quad \vec{x}_{t_2} \quad \cdots \quad \vec{x}_{t_N}$

$$\Psi(\vec{x}_{t_i})_{/\vec{x}_{t_i} \in S} = \sum_{j=1}^{N} w_j \underbrace{\int_{S_j} \frac{1}{\|\vec{x}_{t_i} - \vec{x}\|} dS'}_{A_{ij}} \qquad i = 1,2,\ldots N$$

$$\Psi(\vec{x}_{t_i})_{/x_{t_i} \in S} = \sum_{j=1}^{N} w_j A_{ij}$$

$$\begin{bmatrix} & & \\ & \mathbf{A} & \\ & & \end{bmatrix} \begin{bmatrix} \\ \vec{w} \\ \\ \end{bmatrix} = \begin{bmatrix} \Psi(\vec{x}_{t_1}) \\ \vdots \\ \Psi(\vec{x}_{t_N}) \end{bmatrix}$$

Galerkin Method : $\phi_i(x) = \varphi_i(x)$ (basis = test)

COLLOCATION METHOD

- Basis Function Approach

Collocation : $\phi_i(x) = \delta(x_{t_i})$ (point - matching)

$$\int \delta(x - x_{t_i}) R(x) dS = R(x_{t_i}) = \Psi(x_{t_i}) - \int_{\substack{approx \\ surface}} G(x_{t_i}, x') \sum_{j=1}^{n} \omega_j \varphi_j(x') dS' = 0$$

$$\Rightarrow \Psi(x_{t_i}) = \sum_{j=1}^{n} \omega_j \underbrace{\int_{\substack{approx \\ surface}} G(x_{t_i}, x') \varphi_j(x') dS'}_{A_{i,j}}$$

$$\begin{bmatrix} A_{1,1} & \cdots & A_{1,n} \\ \vdots & \ddots & \vdots \\ A_{n,1} & \cdots & A_{n,n} \end{bmatrix} \begin{bmatrix} \omega_1 \\ \vdots \\ \omega_n \end{bmatrix} = \begin{bmatrix} \Psi(x_{t_1}) \\ \vdots \\ \Psi(x_{t_n}) \end{bmatrix}$$

Collocation.

$$\Psi(x_{t_1}) = \sum_{j=1}^{N} w_j \overbrace{\int_{\substack{panel\\ j}} \frac{1}{x_{t_1} - x'} dS'}^{A_{1j}}$$

$$\Psi(x_{t_2}) = \sum_{j=1}^{N} w_j \overbrace{\int_{\substack{panel\\ j}} \frac{1}{\left\| x_{t_2} - x' \right\|} dS'}^{A_{2j}}$$

$$\Psi(x_{t_i}) = \sum_{j=1}^{N} w_j A_{ij}$$

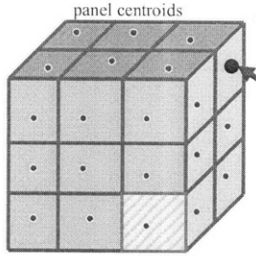
$$A_{ij} = \int_{panel\ j} \frac{1}{\left\| x_{t_i} - x' \right\|} dS'$$

$$\begin{bmatrix} A_{11} & A_{12} & \cdots & A_{1N} \\ & & & \end{bmatrix} \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_N \end{bmatrix} = \begin{bmatrix} \Psi(x_{t_1}) \\ \Psi(x_{t_2}) \\ \vdots \\ \Psi(x_{t_N}) \end{bmatrix}$$

- Centroid Collocation

Put collocation points at panel centroids

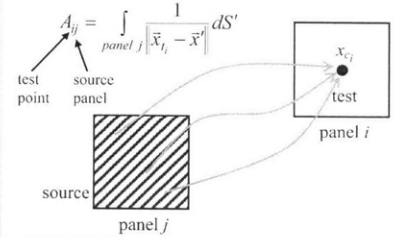
$$\Psi(x_{c_i}) = \sum_{j=1}^{n} \omega_j \underbrace{\int_{panel\ j} G(x_{c_i}, x') dS'}_{A_{i,j}}$$

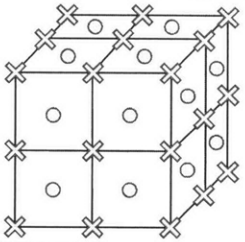


$$\begin{bmatrix} A_{1,1} & \cdots & A_{1,n} \\ \vdots & \ddots & \vdots \\ A_{n,1} & \cdots & A_{n,n} \end{bmatrix} \begin{bmatrix} \omega_1 \\ \vdots \\ \omega_n \end{bmatrix} = \begin{bmatrix} \Psi(x_{t_1}) \\ \vdots \\ \Psi(x_{t_n}) \end{bmatrix}$$

Collocation. Piecewise constant basis.

$$A_{ij} = \int_{panel\ j} \frac{1}{\left\| \vec{x}_{t_i} - \vec{x}' \right\|} dS'$$

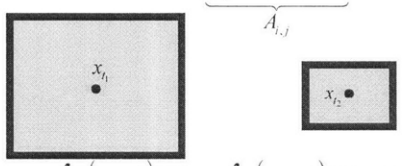




Use centroids (as opposed to vertices) in 3-D because you have as many unknowns as you have test points.

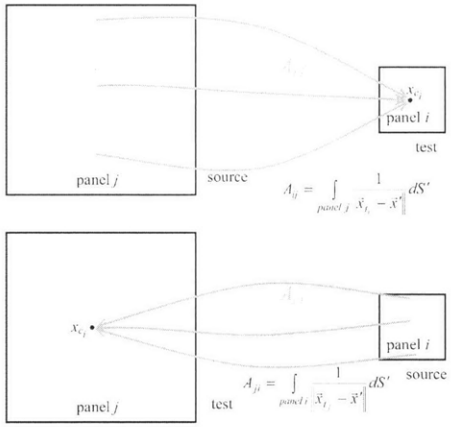
- Nonsymmetric **A**

$$\Psi\left(x_{t_i}\right)=\sum_{j=1}^{n}\omega_j \underbrace{\int_{line\ j} G\left(x_{t_i},x'\right)dS'}_{A_{i,j}}$$



$$A_{1,2}=\int_{panel\ 2} G\left(x_{t_1},x'\right)dS' \neq \int_{panel\ 1} G\left(x_{t_2},x'\right)dS'=A_{2,1}$$

test source test source



GALERKIN METHOD

- Piecewise Constant Basis Function

Galerkin : $\phi_i(x)=\varphi_i(x)$ (test = basis)

$$\int\varphi_i(x)R(x)dS=\int\varphi_i(x)\Psi(x)dS-\int_{\substack{approx\\surface}}\int\varphi_i(x)G(x,x')\sum_{j=1}^{n}\omega_j\varphi_j(x')dS'dS=0$$

$$\underbrace{\int_{\substack{approx\\surface}}\varphi_i(x)\Psi(x)dS}_{b_i}=\sum_{j=1}^{n}\omega_j\underbrace{\int_{\substack{approx\\surface}}\int_{\substack{approx\\surface}}G(x,x')\varphi_i(x)\varphi_j(x')dS'dS}_{A_{i,j}}$$

$$\begin{bmatrix}A_{1,1}&\cdots&A_{1,n}\\ \vdots&\ddots&\vdots\\ A_{n,1}&\cdots&A_{n,n}\end{bmatrix}\begin{bmatrix}\omega_1\\ \vdots\\ \omega_n\end{bmatrix}=\begin{bmatrix}b_1\\ \vdots\\ b_n\end{bmatrix}$$

$$\Psi(x)_{surface}=\int_{surface}\frac{1}{\|x-x'\|}\sigma(x')dS'$$

Basis Functions $\sigma(x')\cong\sum w_j\varphi_j(x')$

$$\Psi(x)_{surface}=\sum_{j=1}^{n}w_j\int_{surface}\frac{1}{\|x-x'\|}\varphi_j(x')dS'$$

Galerkin

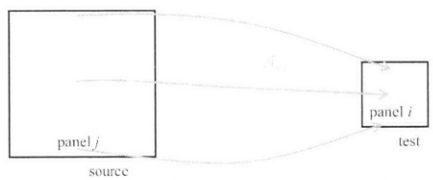
$$\int_s\Psi(x)_s\varphi_i(x)dS=\int_S\varphi_i(x)\sum_{j=1}^{n}w_j\int_s\frac{1}{\|x-x'\|}\varphi_j(x')dS'dS \qquad i=1,2,\ldots,n$$

$$\int_s\Psi(x)_s\varphi_i(x)dS=\sum_{j=1}^{n}w_j\int_s\int_s\frac{1}{\|x-x'\|}\varphi_j(x')\varphi_i(x)dS'dS \qquad i=1,2,\ldots,n$$

Piecewise constant basis functions

$$\underbrace{\int_{panel\ i}\Psi(x)_{x\in S}dS}_{b_i}=\sum_{j=1}^{n}w_j\underbrace{\int_{panel\ i}\int_{panel\ j}\frac{1}{\|x-x'\|}dS'dS}_{A_{ij}} \qquad i=1,2,\ldots,n$$

$$\begin{bmatrix} \leftarrow & \int_S\int_S \frac{\varphi_1(x)\varphi_j(x')}{\|x-x'\|}dx'dx = \int_{S_1}\int_{S_j}\frac{1}{\|x-x'\|}dx'dx & \rightarrow \\ \leftarrow & \int_S\int_S \frac{\varphi_2(x)\varphi_j(x')}{\|x-x'\|}dx'dx = \int_{S_2}\int_{S_j}\frac{1}{\|x-x'\|}dx'dx & \rightarrow \\ & \vdots & \end{bmatrix}\begin{bmatrix}\uparrow \\ \vec{w} \\ \downarrow\end{bmatrix} = \begin{bmatrix}\int_{S_1}\Psi(x)_{x\in S}dx \\ \int_{S_2}\Psi(x)_{x\in S}dx \\ \vdots\end{bmatrix}$$



Galerkin requires that the boundary conditions are satisfied "on averages" over each test panel i

General Galerkin: $A_{ij} = \int_S\int_S \varphi_i(x)\varphi_j(x)G(x,x')dx'dx$

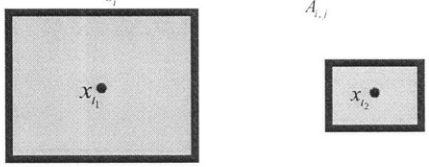
$$\underbrace{\int \varphi_i(x)\Psi(x)dS}_{b_i} = \sum_{j=1}^{n}\omega_j \underbrace{\int\int \varphi_i(x)G(x,x')\varphi_j(x')dS'dS}_{A_{i,j}}$$

$$\underbrace{\int \varphi_i(x)\Psi(x)dS}_{b_i} = \sum_{j=1}^{n}\omega_j \underbrace{\int_{panel\ i}\int_{panel\ j}\frac{1}{\|x-x'\|}dS'dS}_{A_{i,j}}$$

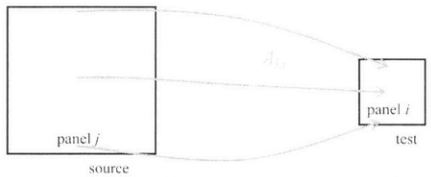
$$\begin{bmatrix} A_{1,1} & \cdots & A_{1,n} \\ \vdots & \ddots & \vdots \\ A_{n,1} & \cdots & A_{n,n}\end{bmatrix}\begin{bmatrix}\omega_1 \\ \vdots \\ \omega_n\end{bmatrix} = \begin{bmatrix} b_1 \\ \vdots \\ b_n\end{bmatrix}$$

❖ Symmetric **A**

$$\underbrace{\int \Psi(x)dS'}_{b_i} = \sum_{j=1}^{n}\alpha_j \underbrace{\int_{panel\ i}\int_{panel\ j}\frac{1}{\|x-x'\|}dS'dS}_{A_{i,j}}$$



If $G(x,x') = G(x',x)$ then $A_{i,j} = A_{j,i}$ **A** is symmetric



Galerkin requires that the boundary conditions are satisfied "on averages" over each test panel i

$$A_{ij} = \int_{panel\ i}\int_{panel\ j} \frac{1}{x - x'}\, dS'dS$$

$$A_{ji} = \int_{panel\ j}\int_{panel\ i} \frac{1}{x' - x}\, dS'dS$$

$A_{ij} = A_{ji}$ if

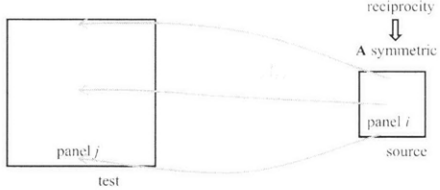
$$G(x,x') = G(x',x)$$

$$\frac{1}{\|x - x'\|} = \frac{1}{\|x' - x\|}$$

reciprocity

⇩

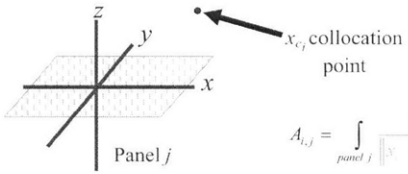
A symmetric



Even if size panel $i \neq$ size panel j

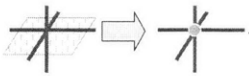
3D PANEL INTEGRATION

❖ Collocation Approach – Calculating Matrix Elements



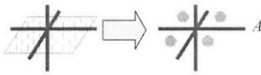
$$A_{i,j} = \int_{panel\ i} \frac{1}{\|x - x'\|}\, dS'$$

One point quadrature approximation



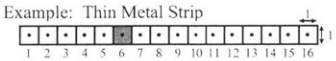
$$A_{i,j} \approx \frac{Panel\ Area}{\|x - x_{c_{panel\ i}}\|}$$

Four point quadrature approximation



$$A_{i,j} \approx \sum_{p=1}^{4} \frac{0.25 * Area}{\|x - x_{c_{panel\ i}}\|}$$

Example: Thin Metal Strip



Piecewise Constant Basis Function

Unknowns: w_j charges on each panel

Assume $\Psi(x) = 1$ given on the metal strip

$$\varphi_j(x') = \begin{cases} 1 & x' \in panel\ j \\ 0 & x' \notin panel\ j \end{cases}$$

$$\begin{bmatrix} \_ & 1 & \frac{1}{2} & \frac{1}{3} & \frac{1}{4} & \frac{1}{5} & \frac{1}{6} & \cdots & \frac{1}{15} \\ & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & & \\ \cdots & 1 & \_ & 1 & \frac{1}{2} & \frac{1}{3} & \frac{1}{4} & \frac{1}{5} & \cdots \\ & & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \end{bmatrix} \begin{bmatrix} w_1 \\ \vdots \\ w_j \\ \vdots \\ w_{16} \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \\ 1 \end{bmatrix}$$

Toeplitz Matrix

Known Potential

Collocation

$$A_{\substack{1,\ 6 \\ test\ source}} = \int_{panel\,6} \frac{1}{\|x_{c_1} - x'\|} dS' = \frac{\text{Area Panel 6}}{\|x_{c_1} - x_{c_6}\|} = \frac{1}{5}$$

$$A_{\substack{2,\ 6 \\ test\ source}} = \int_{panel\,6} \frac{1}{\|x_{c_2} - x'\|} dS' = \frac{\text{Area Panel 6}}{\|x_{c_1} - x_{c_2}\|} = \frac{1}{4}$$

$$A_{6,1} = \int_{panel\,1} \frac{1}{\|x_{c_6} - x'\|} dS' = \frac{\text{Area Panel 1}}{\|x_{c_6} - x_{c_1}\|} = \frac{1}{5}$$

$A_{6,1} = A_{1,6}$. Why? Collocation in general gives $A_{6,1} \neq A_{1,6}$

But if $\text{Area}_{panel\,1} = \text{Area}_{panel\,6}$, then $A_{6,1} = A_{1,6}$

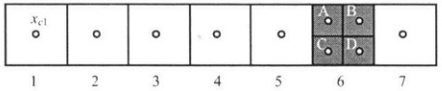
Self–term

What about $A_{1,1}$?

$$A_{1,1} = \int_{panel\,1} \frac{1}{\|x_{c_1} - x'\|} dS' = \frac{\text{Area Panel 1}}{\|x_{c_1} - x_{c_1}\|} = \frac{1}{0}$$

No.

Four Point Quadrature



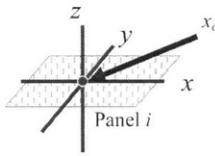
$$A_{\substack{1,\ 6 \\ test\ source}} = \int_{panel\,6} \frac{1}{\|x_{c_1} - x'\|} dS'$$

$$= \frac{\text{Area 6A}}{\|x_{c_1} - x_{c_{6A}}\|} + \frac{\text{Area 6B}}{\|x_{c_1} - x_{c_{6B}}\|} + \frac{\text{Area 6C}}{\|x_{c_1} - x_{c_{6C}}\|} + \frac{\text{Area 6D}}{\|x_{c_1} - x_{c_{6D}}\|}$$

Does $A_{6,1} = A_{1,6}$ with four point quadrature?

In general, yes, if shape panel j = shape panel i

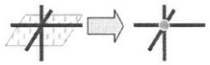
❖ Basis Function Approach – Calculating "Self-Term"



x_{c_i} collocation point

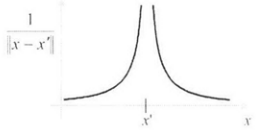
$$A_{i,i} = \int_{panel\ i} \frac{1}{\|x_{c_i} - x'\|} dS'$$

One point quadrature Approximation

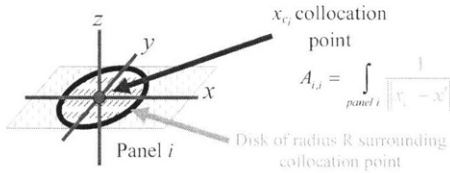


$$A_{i,i} \approx \frac{\text{Panel Area}}{\|x_{c_i} - x_{c_i}\|} \to 0$$

$$A_{i,i} = \int_{panel\ i} \frac{1}{\|x_{c_i} - x'\|} dS' \text{ is an integrable singularity}$$



$$A_{i,j} = \int_S \frac{1}{\|x-x'\|}\,dx'$$



$$A_{i,j} = \int_{panel\ i} \frac{1}{\|x_{c_i}-x'\|}\,dS'$$

Integrate in two pieces

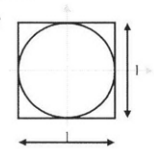
$$A_{i,j} = \int_{disk} \frac{1}{\|x_{c_i}-x'\|}\,dS' + \int_{rest\ of\ panel} \frac{1}{\|x_{c_i}-x'\|}\,dS'$$

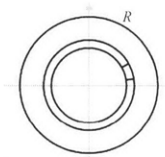
Disk Integral has singularity but has analytic formula

$$\int_{disk} \frac{1}{\|x_{c_i}-x'\|}\,dS' = \int_0^R \int_0^{2\pi} \frac{1}{r}\,r\,dr\,d\theta = 2\pi R$$

Self-Term

$A_{6,6} = ?$





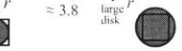
$dS' = dr\,r\,d\theta$

$$\int_{disk} \frac{1}{\|x-x'\|}\,dx' = \int_{disk} \frac{1}{r}\,dS' = \int_0^R \int_0^{2\pi} \frac{1}{r}\,r\,dr\,d\theta = 2\pi R$$

$$A_{6,6} = \int_{Panel6} \frac{1}{\|x_{c_6}-x'\|}\,dS' = \int_{disk} \frac{1}{r}\,dS' + \int_{rest} \frac{1}{r}\,dS'$$

$$2\pi \cdot \frac{1}{2} = \int_{disk} \frac{1}{r}\,dS' \le A_{6,6} \le \int_{large\ disk} \frac{1}{r}\,dS' = 2\pi \cdot \frac{\sqrt{2}}{2}$$

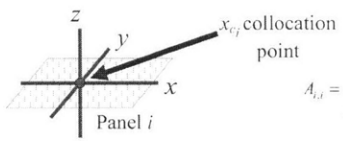
≈ 3.8



Toeplitz Matrix

$$\begin{bmatrix} 3.8 & 1 & \frac{1}{2} & \frac{1}{3} & \frac{1}{4} & \frac{1}{5} & \frac{1}{6} & \cdots & \frac{1}{15} \\ \cdots & 1 & 3.8 & 1 & \frac{1}{2} & \frac{1}{3} & \frac{1}{4} & \frac{1}{5} & \cdots & \frac{1}{16-i} \\ & & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & & \ddots \end{bmatrix}$$

- Not diagonally dominant
- Dense
- All positive entries



$$A_{i,i} = \int_{panel\ i} \frac{1}{\|x - x'\|}\, dS'$$

Integrand is singular

1. If panel is a flat polygon, analytical formulas exist.
2. Curve panels can be handled with projection.

Solving Discretized Integral Equations

- Basis Function – problem with dense matrices

  Integral Equation Method Generates Huge Dense Matrices

$$\begin{bmatrix} A_{1,1} & \cdots & A_{1,n} \\ \vdots & \ddots & \vdots \\ A_{n,1} & \cdots & A_{n,n} \end{bmatrix} \begin{bmatrix} \omega_1 \\ \vdots \\ \omega_n \end{bmatrix} = \begin{bmatrix} \Psi(x_{t_1}) \\ \vdots \\ \Psi(x_{t_n}) \end{bmatrix}$$

  Gaussian Elimination Much Too Slow!

  $n = 100{,}000 = 10^5$

  Storage for an $n \times n$ matrix = $\underbrace{8 \text{ bytes}}_{\text{double precision}} \times \underbrace{10^{10}}_{n^2} = 80$ gigabytes

- The Generalized Conjugate Residual Algorithm –
  - The kth step of GCR

compute $\mathbf{A}r^{k-1}$	For discretized integral equations, $\mathbf{A}$ is dense
$\tilde{p}_{k-1} \leftarrow r^{k-1} - \sum_{j=0}^{k-2} (\mathbf{A}p_j)^T (\mathbf{A}r^{k-1}) p_j$	$\mathbf{A}$ - Orthonormalize search direction
$p_{k-1} \leftarrow \frac{\tilde{p}_{k-1}}{\|\mathbf{A}\tilde{p}_{k-1}\|}$	
$y_{k-1} \leftarrow (r^{k-1})^T \mathbf{A}p_{k-1}$	Determine optimal stepsize in k^{th} search direction
$x^k \leftarrow x^{k-1} + y_{k-1}p_{k-1}$	Update the solution and the residual
$r^k \leftarrow r^{k-1} - y_{k-1}\mathbf{A}p_{k-1}$	

  - Complexity of symmetric GCR (e.g. for Galerkin)

compute $\mathbf{A}r^{k-1}$	Dense matrix - vector product costs $O(n^2)$
$\tilde{p}_{k-1} \leftarrow r^{k-1} - \sum_{j=0}^{k-2} (\mathbf{A}p_j)^T (\mathbf{A}r^{k-1}) p_j$	inner product : $O(n)$
$p_{k-1} \leftarrow \frac{\tilde{p}_{k-1}}{\|\mathbf{A}\tilde{p}_{k-1}\|}$	
$y_{k-1} \leftarrow (r^{k-1})^T \mathbf{A}p_{k-1}$	inner product : $O(n)$
$x^k \leftarrow x^{k-1} + y_{k-1}p_{k-1}$	$O(n)$ mult.
$r^k \leftarrow r^{k-1} - y_{k-1}\mathbf{A}p_{k-1}$	

3D Finite Difference Method — $100 \times 100 \times 100$ grid

Number of unknowns = 10^6

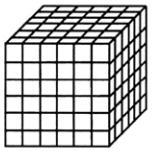
Number of nonzeros = $7 \cdot 10^6 \rightarrow 7$ million $\quad O(n)$

3D Integral Equation Method

Number of unknowns = 10^6

Number of nonzeros = $10^{12} \rightarrow 8{,}000$ gigabytes $\quad O(n^2)$

- Fast Matrix Vector Products
  - Computing exactly $\mathbf{A}r^{k-1}$
    Dense matrix-vector product costs $O(n^2)$
  - Computing approximately $\mathbf{A}r^{k-1}$
    Reduces matrix-vector product costs to $O(n)$ or $O(n \log n)$



$$\mathbf{A}\vec{q} = \Psi$$

unknown charge densities

Given a P.D.E. e.g. $\nabla^2\Psi = 0$ $\quad \nabla^2\Psi + \kappa^2\Psi = 0$

1.) Find a Green Function e.g. $G(x,x') = \dfrac{1}{|x-x'|}$ $\quad G(x,x') = \dfrac{e^{i\kappa|x-x'|}}{|x-x'|}$

2.) Discretize Surface Boundary Conditions Using Basis Functions

$$\sigma(x) = \sum_{j=1}^{d} w_j \varphi_j(x) \xrightarrow[\varphi_j(x) = \begin{cases} 1 & x \in S_j \\ 0 & x \notin S_j \end{cases}]{\text{Piecewise Constant}} \Psi(x) = \sum_{j=1}^{d} w_j \int_{S_j} G(x,x')dS'$$

3.) Impose Boundary Conditions

$$\mathbf{A}\begin{bmatrix} w_1 \\ \vdots \\ w_d \end{bmatrix} = b$$

e.g. Collocation $\quad \underbrace{\Psi(x_i)}_{b_i} = \sum_{j=1}^{N} w_j \underbrace{\int_{S_j} G(x,x')dS'}_{A_{ij}}$

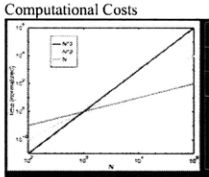
e.g. Galerkin $\quad \underbrace{\int_{S_i} \Psi(x_i)dS}_{b_i} = \sum_{j=1}^{N} w_j \underbrace{\int_{S_i}\int_{S_j} G(x,x')dSdS'}_{A_{ij}}$

4.) Solve system using Krylov iterative methods at each step need $\mathbf{A}r^{k-1}$

5.) Use Fast Matrix Vector Product for $\mathbf{A}r^{k-1}$ (e.g. Fast Multipole or PFFT)

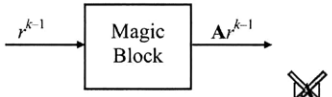
Cost $O(N \log N)$
$O(N)$ memory

- Computational Costs



	DEC 21164-333	
N	Gaussian Elim	"Fast" $O(N)$
	300 MFLOPS	30 MFLOPS
5e4	3 days,20GB	80sec, 130M
1e5	25 days,80GB	2.5min, 300M
5e5	8.8yrs,2TB	15min, 1.5GB

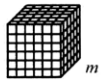
- Gaussian Elimination: $O(n^3)$ time, $O(n^2)$ memory
- GCR with direct M-V: $O(n^2)$ time, $O(n^2)$ memory
- Fast Methods: $O(n)$ time, $O(n)$ memory



Integral Equation Discretization

- Cube Example, Interior Problem – Memory Usage

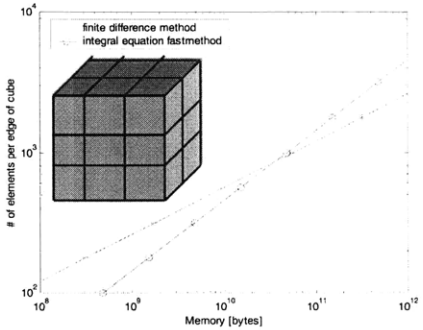
Solving a cube interior problem



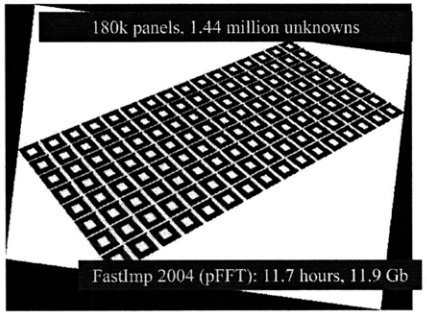
m

	F.D.			I.E. (BEM)		
	80's	2005	$O(\ )$	80's	2005	$O(\ )$
# panels on edge m	100	800 × 8		100	800 × 8	
# unknowns n	1M	× 512	$O(m^3)$	60,000	× 64	$O(6m^2)$
# nonzero entries	7M	× 512	$O(7n)=$ $O(7m^3)$	3.6 G	×4096	$O(n^2)=$ $O(36m^4)$
memory	56 MB	× 512 30 GB	$O(56n)=$ $O(56m^3)$	~~20 GB~~	~~×4096~~ ~~120 TB~~	$O(8n^2)=$ $O(290m^4)$
				0.5 GB	×64 30 GB	$O(8000n)$ $=$ $O(48000m^2)$

GCR + Fast Methods



- 128 3-turn spiral RF inductors array



$$\nabla^2\Psi = 0 \quad \Rightarrow \Psi(x) = \int G(x,x')\sigma(x)dS \quad \Rightarrow \mathbf{A}\vec{x} = \vec{b}$$

(1) Choice of basis functions
(2) Choice of evaluation – Collocation or Galerkin

3.23 PDE - Fast Methods for Integral Equation Solvers

INTRODUCTION TO NUMERICAL SIMULATION

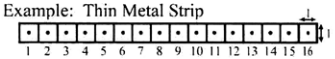
LECTURE 24.A.
PDE – BEM Integral Equation Method III
Fast Methods for Integral Equation Solvers

TODAY'S OUTLINE:

- Fast Matrix-Vector Product
  - Multipole Algorithm (for Laplace Green Function)
  - Precorrected FFT (Green Function Independent)

FAST MATRIX-VECTOR PRODUCT

Example: Thin Metal Strip



Assume $\Psi(x) = 1$ given everywhere on the metal strip

Find $\sigma(x)$ on strip $\leftrightarrow$ Find weights $w_1, w_2, \ldots w_{16}$ (charge on each panel)

Piecewise Constant Basis Function

$$\varphi_j(x') = \begin{cases} 1 & x' \in S_j \\ 0 & x' \notin S_j \end{cases}$$

$$A_{i,j} = \int_{panel\ j} \frac{1}{|x_{c_i} - x'|} dS'$$

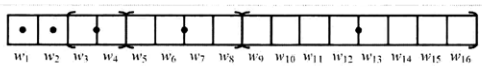
$$\approx \frac{1}{|x_{c_i} - x_{c_j}|} \approx \frac{1}{|i-j|}$$

$$A_{6,1} = \frac{\text{Area Panel 6}}{\|x_{c_1} - x_{c_6}\|} = \frac{1}{5} = A_{6,1}$$

$$\pi \le A_{1,1} \le \pi\sqrt{2} \quad A_{1,1} \approx 3.8$$

$$\begin{bmatrix} 3.8 & 1 & \frac{1}{2} & \frac{1}{3} & \frac{1}{4} & \frac{1}{5} & \frac{1}{6} & \cdots & \frac{1}{16} \\ 1 & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & & \\ \frac{1}{2} & \ddots & 3.8 & 1 & \frac{1}{2} & \frac{1}{3} & \frac{1}{4} & \frac{1}{5} & \\ \frac{1}{3} & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots \end{bmatrix}$$

Structure: Diagonals (Toeplitz)



$d = 16$

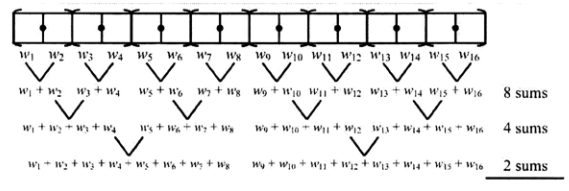
Matrix Vector Product $\quad \mathbf{A}\begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_{16} \end{bmatrix} = \Psi$

$$\Psi(x_1) = 3.8w_1 + 1w_2 + \tfrac{1}{2}w_3 + \tfrac{1}{3}w_4 + \ldots + \tfrac{1}{15}w_{16}$$

$$\Psi(x_1) = 3.8w_1 + \frac{w_2}{1} + \frac{w_3 + w_4}{2.5} + \frac{w_5 + w_6 + w_7 + w_8}{5.5} + \frac{w_9 + w_{10} + w_{11} + \ldots + w_{16}}{11.5}$$

4 products
4 sums
$= \log_2 d$

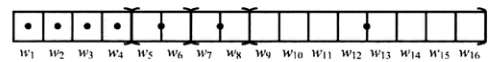
In order to do all the others $d \cdot \log_2 d$ $\vdash\!\!\rightarrow$ cost of the entire matrix vector product provided j can get the clusters cheaply.



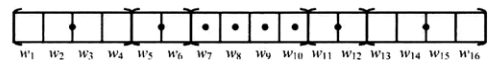
14 sums

$\log_2 d - 1$ sums $\approx O(d)$ sums

How do we use those sums?



$$\Psi(x_2) \cong \frac{1}{1}w_1 + 3.8w_2 + \frac{1}{1}w_3 + \frac{1}{2}w_4 + \frac{w_5 + w_6}{3.5} + \frac{w_7 + w_8}{5.5} + \frac{w_9 + w_{10} + w_{11} + \ldots + w_{16}}{10.5}$$



$$\Psi(x_8) \cong \frac{w_1 + \ldots + w_4}{5.5} + \frac{w_5 + w_6}{2.5} + w_7 + 3.8w_8 + w_9 + \frac{1}{2}w_{10} + \frac{w_{11} + w_{12}}{3.5} + \frac{w_{13} + \ldots + w_{16}}{6.5}$$

All are $O(\log_2 d) \rightarrow$ total cost $O(d \cdot \log_2 d)$

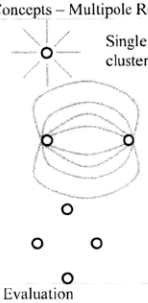
Multipole Algorithm (for Laplace Green Function)

❖ Basic Multipole Concepts – Multipole Representation

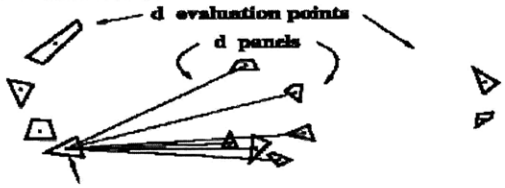
Monopole $j = 1$ — Single charge in center of cluster with sum of all charges — Potential $\propto \frac{1}{R}$

Dipole $j = 2$ — Two Charges — Potential $\propto \frac{1}{R^2}$

Quadrapole $j = 4$ — Four Charges — Potential $\propto \frac{1}{R^4}$



- Direct Potential Evaluation

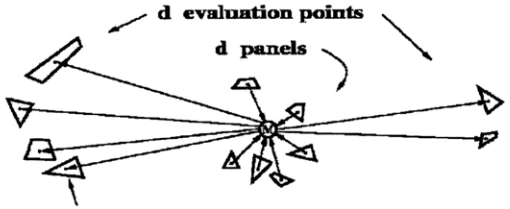


- Potential at point i:

$$v_i(r_i, \phi_i, \theta_i) = \sum_{j=1}^{d} q_j P_{ij}$$

- Complete evaluation at d points costs d^2 operations

- Multipole Potential Evaluation



- Approximate potential at point i:

$$v_i(r_i, \phi_i, \theta_i) \approx \sum_{j=0}^{order} \sum_{k=-j}^{j} \frac{M_j^k}{r_i^{j+1}} Y_j^k(\phi_i, \theta_i)$$

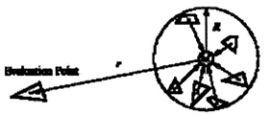
- Multipole coefficients function of panel charges:

$$M_j^k \triangleq \sum_{i=1}^{d} \frac{q_i}{A_i} \int_{\text{panel } i} \rho^i Y_j^{-k}(\alpha, \beta) dA$$

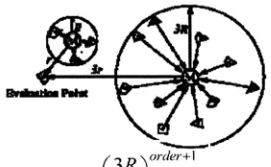
- Computing Multipole expansions costs order d operations.
- Each approximate potential evaluation costs order 1 operations.

d potential evaluation due to d panels in order d operations

- Scale Invariance of Error



$$\text{Error} \le K\left(\frac{R}{r}\right)^{order+1}$$

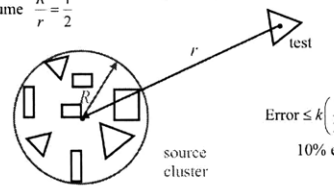


$$\text{Error} \le K\left(\frac{3R}{3r}\right)^{order+1}$$

e.g. assume $j = 2$

assume $\frac{R}{r} = \frac{1}{2}$

monopole, dipole

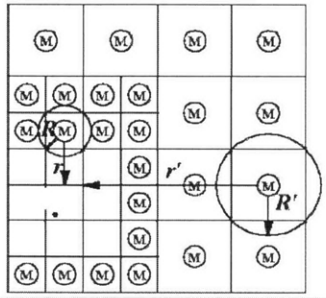


$$\text{Error} \le k\left(\frac{1}{2}\right)^{2+1} = k\frac{1}{8}$$

10% error bound

Key Point: If test is much further away (r larger) then the clusters can be larger (R larger by same amount)

- Multipole Algorithm Hierarchy

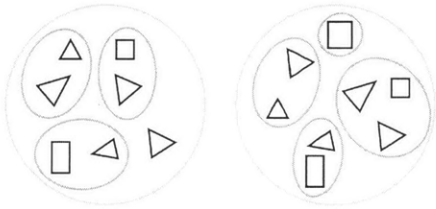


Hierarchy guarantees:
Bounded error:

$$\text{Error} \le K\left(\frac{R}{r}\right)^{order+1}$$

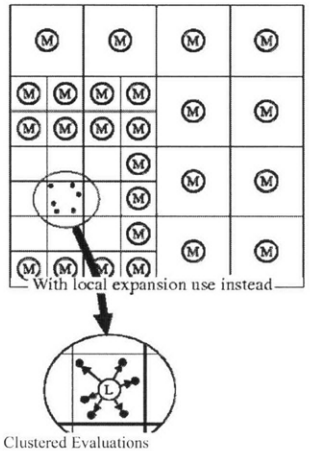
$$\le K\left(\frac{1}{2}\right)^{order+1}$$

order = 2 yields one percent accuracy.



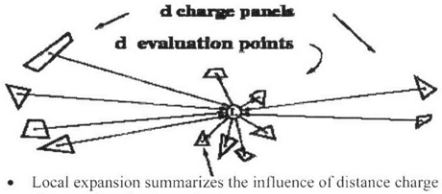
❖ Multipole Optimizations – Local Expansions

- Cost Reduction



- Construct a local expansion to represent distant charge potentials.
- Evaluate a single local expansion, rather than many mulitpole expansions, at each evaluations point.

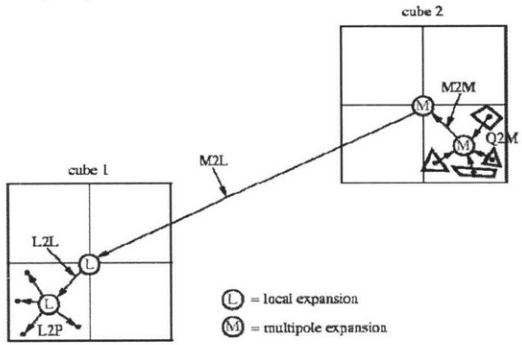
- Clustered Evaluations



- Local expansion summarizes the influence of distance charge or clusters of evaluation points.
- Gives $O(n)$ potential evaluation when combined with coalescing of charge done by multipole expansions.
- Approximate potential at point i:

$$v_i(r_i, \phi_i, \theta_i) \approx \sum_{j=0}^{order} \sum_{k=-j}^{j} L_j^k Y_j^k(\phi_i, \theta_i) r_i^j$$

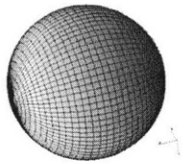
- Summary of Operations



  - Multipole and local expansions are built using complementary hierarchies.
  - Complete calculation consists of:
    1. Build multipoles (Upward Pass).
    2. Build locals (Downward Pass).
    3. Evaluation local expansions and nearby charge potential (Evaluation Pass).

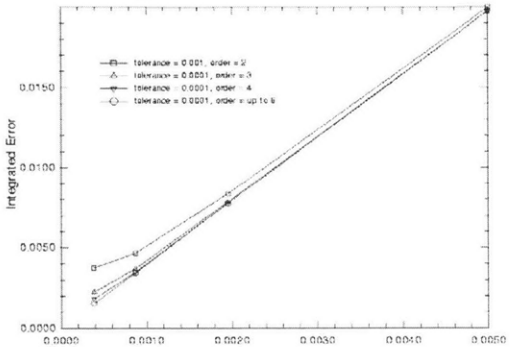
❖ Computational Examples

- Translating Sphere
  - Potential Distribution



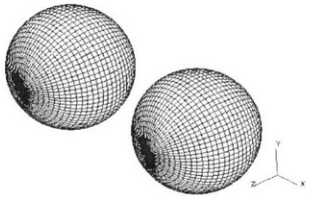
  - Potential given by $\psi(x) = -\frac{x_3}{2\|x\|^3}$
  - Charge given by $\sigma(x) = -\frac{3}{8\pi} x_3$

  - Discretization Convergence



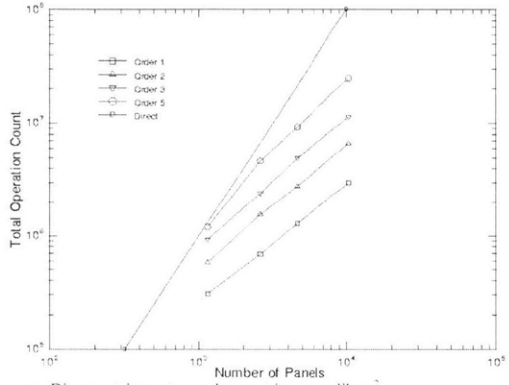
  - Error should decay like $1/n$
  - Multipole approximations eventually interfere
  - Higher-order multipole expansions needed for higher accuracy

- Two Sphere Example
  - Potential Distribution



  - Potential on each sphere : $\psi(x) = -\frac{x_3}{2\|x\|^3}$
  - Does not correspond to a simple physical problem.

- Matrix-Vector Product Cost



- Direct matrix-vector product cost increases like n^2.
- Multipole matrix-vector product cost increases like n.
- The slope for the multipole algorithm depends on accuracy.
- For order 2 expansions, breakpoint is about $n = 400$.

❖ Complexity Summary

For an integral equation discretized with n panels:

- Gaussian elimination: $O(n^3)$
- GCR, direct M-V $O(n^2)$.
- Multipole accelerated GCR $O(mn)$.

translation invariant

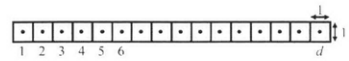
$$\Psi(x_i) = \sum_{j=1}^{n} f(i-j)\sigma_j \longleftarrow \text{Discrete Convolution} \quad \text{Computed in } n \log n \text{ using FFT}$$

$$\nabla^2\Psi + \kappa^2\Psi = 0 \qquad \text{Helmholtz Equation (Full Wave Equation)}$$

$$G(x,x') = \frac{e^{j\kappa|x-x'|}}{|x-x'|}$$

Can still use multipole expansions but every time I change the Green function I need to come up with multipole expansions.
Fast Multipole is Green Function dependent!

Example: Thin Metal Strip

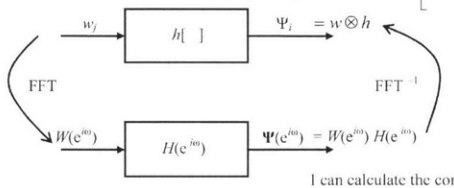


$$\Psi(x_i) = \sum_{j=1}^{d} w_j A_{i,j} \qquad A_{i,j} = \frac{Area}{|x_i - x_j|} = \frac{1}{|i-j|} = \underbrace{A_{i-j}}_{h_{i-j}} = A_{j,i}$$

$$\Psi(x_i) = \sum_{j=1}^{d} w_j h_{i-j} = w \otimes h$$

Toeplitz

Recognize Discrete Convolution Sum
h is the unit sample response.



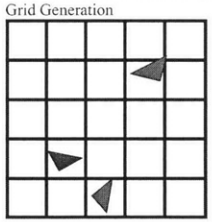
I can calculate the convolution efficiently using 2 FFT [$O(d \log_2 d)$]

This works only if $G(x,x') = G(|x-x'|)$ translation invariant

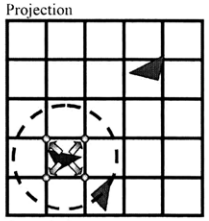
Precorrected FFT (Green Function Independent)

❖ P-FFT Matrix Vector Product

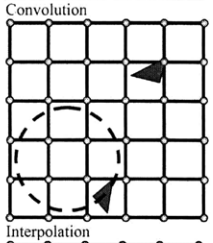
- Problem: Solve iteratively $\mathbf{A}\vec{w} = \Psi$
- At each iteration evaluate matrix-vector products $\mathbf{A}\vec{w}$ using the following steps:
  - Grid Generation



- Projection



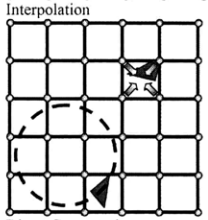
- Convolution



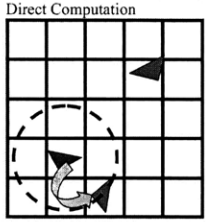
Calculate potentials on grid points due to charges on grid points with FFT

$O(n^2) \Rightarrow O(n \log n)$

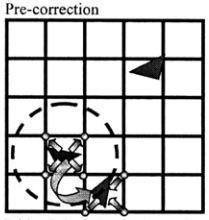
- Interpolation



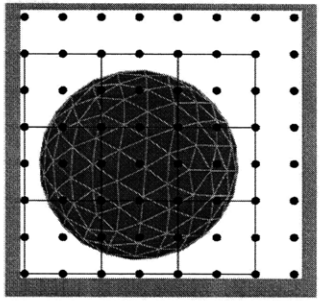
- Direct Computation



- Pre-correction

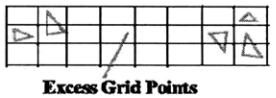


❖ FFT Grid – Selected to Balance Costs



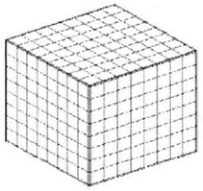
- Grid selected so direct cost equals FFT cost.
- Finer problem discretizations usually yield finer grids.

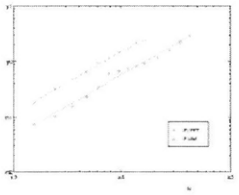
❖ Inhomogeneity Problem



Excess Grid Points

Inhomogeneity – Empty Grid due to FFT - Inefficiency
Refining Cube Discretization – Worsening Inhomogenity





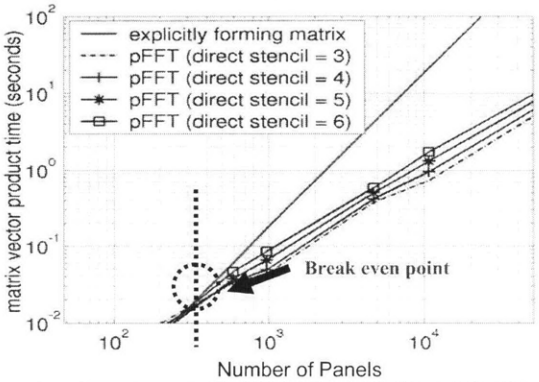
Trade off for a given discretization

Coarse Grid
- FFT cheap
- Direct Computation and Preconditioner – expensive

Fine Grid
- FFT expensive
- Direct Computation and Preconditioner – cheap

Balance
Typically produces
panels ≃ # grid points

❖ Computational Complexity of PFFT++ is nearly $O(N)$
PFFT++ is available at www.rle.mit.edu/cpg



EQS	FASTCAP	$\nabla^2\Psi=0$	Capacitance
MQS	FASTHENRY	$\longrightarrow$	L, R

Fullwave @UIUC
Fast Multipole

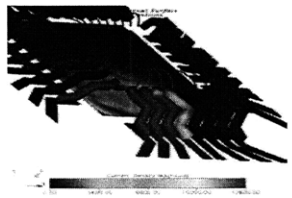
pFFT++

FASTIMP EMQS + Fullwave impedance extraction

FASTMAXWELL EMQS + Fullwave impedance extraction

- Example Maxwell Integral Equation Solver
  Kernel Not $1/R$ Green Functions

$$\nabla \times E = -\mu \frac{dH}{dt}$$
$$\nabla \times H = \varepsilon \frac{dE}{dt}$$
$$\nabla \bullet \varepsilon E = 0$$
$$\nabla \bullet \mu H = 0$$

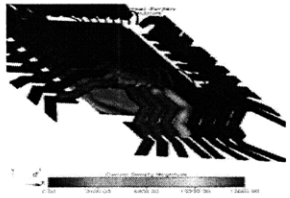


$$\frac{\mathbf{J}(\mathbf{r})}{\sigma} + j\omega \frac{\mu}{4\pi} \int_V \mathbf{J}(\mathbf{r}') \frac{e^{jk|\mathbf{r}-\mathbf{r}'|}}{|\mathbf{r}-\mathbf{r}'|} d\mathbf{r}' = -\nabla\phi$$

$$\frac{1}{4\pi\varepsilon} \int_S \rho(\mathbf{r}_s') \frac{e^{jk|\mathbf{r}_S-\mathbf{r}_S'|}}{|\mathbf{r}_S-\mathbf{r}_S'|} d\mathbf{r}_S' = \phi(\mathbf{r}_S)$$

Preconditioner for Maxwell Integral Equation Solver

$$\nabla \times E = -\mu \frac{dH}{dt}$$
$$\nabla \times H = \varepsilon \frac{dE}{dt}$$
$$\nabla \bullet \varepsilon E = 0$$
$$\nabla \bullet \mu H = 0$$



discretization

$$M \begin{bmatrix} L & 0 \\ 0 & P \end{bmatrix} M^T x = b$$

very sparse

very dense, symmetric
diagonally dominant

- Preconditioning
  - Krylov Methods – Diagonal Preconditioners

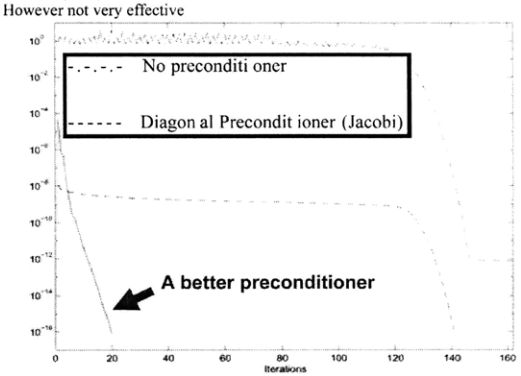
    Let $\mathbf{A} = \mathbf{D} + \mathbf{A}_{nd}$

    Apply GCR to $(\mathbf{D}^{-1}\mathbf{A})x = (\mathbf{I} + \mathbf{D}^{-1}\mathbf{A}_{nd})x = \mathbf{D}^{-1}b$
    - The inverse of a diagonal is cheap to compute
    - Usually improves convergence
  - Integral Equation Preconditioning

    Diagonal Preconditioner (Jacobi)

$$\mathbf{P} = \left(diag\left(\mathbf{M} \begin{bmatrix} \mathbf{L} & \mathbf{0} \\ \mathbf{0} & \mathbf{P} \end{bmatrix} \mathbf{M}^T \right) \right)^{-1}$$

    Extremely easy to invert
    However not very effective



    Diagonal Preconditioner (Jacobi)

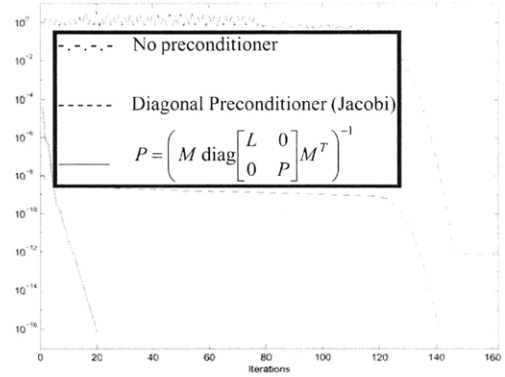
$$\mathbf{P} = \left(diag\left(\mathbf{M} \begin{bmatrix} \mathbf{L} & \mathbf{0} \\ \mathbf{0} & \mathbf{P} \end{bmatrix} \mathbf{M}^T \right) \right)^{-1}$$

    Extremely easy to invert
    However not very effective

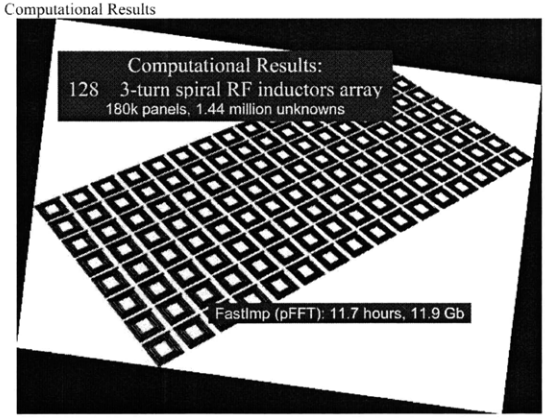
    A Better Preconditioner:

$$\mathbf{P} = \left(\mathbf{M}\, diag \begin{bmatrix} \mathbf{L} & \mathbf{0} \\ \mathbf{0} & \mathbf{P} \end{bmatrix} \mathbf{M}^T \right)^{-1}$$

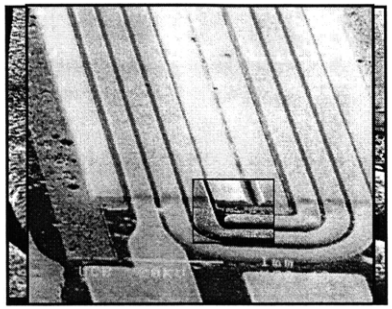
A bit more difficult to factor (but still duable since **M** is sparse)
However much more effective

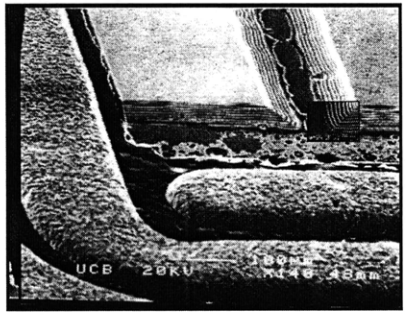


❖ Computational Results



3.24 Model Order Reduction I





INTRODUCTION TO NUMERICAL SIMULATION

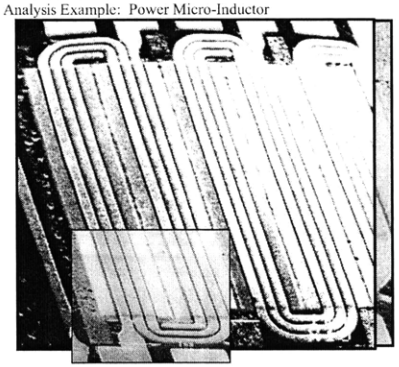
LECTURE 24.B.
Model-Order Reduction

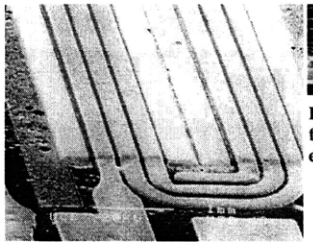
TODAY'S OUTLINE:

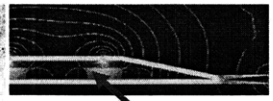
- Motivations and Examples
- From PDE to a State Space Model
  - Heat Conducting Bar

MOTIVATION AND EXAMPLES

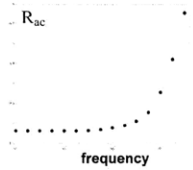
- Analysis Example: Power Micro-Inductor





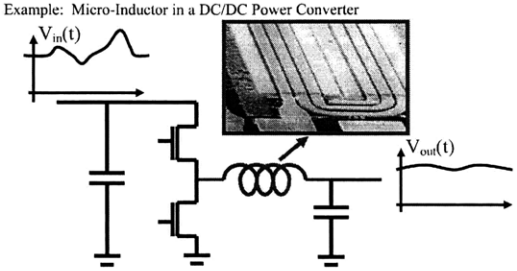


How is the magnetic fringing field from the core effecting eddy current losses?



Spattered laminated NiFe core, electroplated windings [Daniel96]

❖ Example: Micro-Inductor in a DC/DC Power Converter

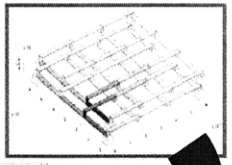


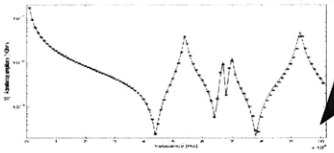
How is the frequency dependency of the power loss in the inductor affecting the dynamics of the power converter and its overall efficiency?

❖ Motivation: Analysis Produces Impedance vs. Frequency Curves

- How are parasitic and the resonances of the power distribution grid affecting the impedance?
- *Analysis tools* can produce for instance impedance vs. frequency curves.

PCB, package, IC interconnects

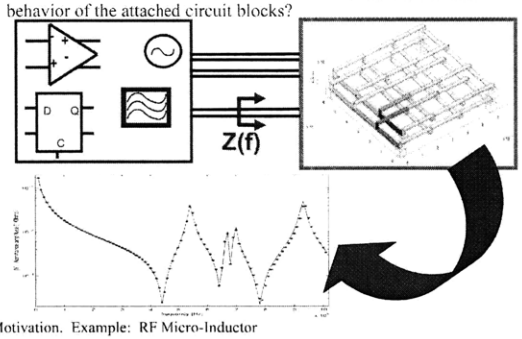




❖ Example: Power Grid used Feeding Circuit Blocks

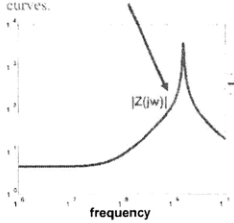
But what is the effect of Z(f) on the behavior of the attached circuit blocks?

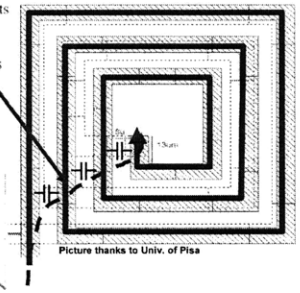
PCB, package, IC interconnects



❖ Motivation. Example: RF Micro-Inductor

- How are the substrate eddy currents affecting the quality factor of the inductor?
- How are the displacement currents affecting the resonance of the inductor?
- *Analysis tools* can produce for instance impedance vs. frequency curves.

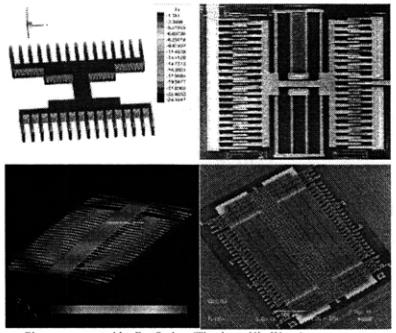




Picture thanks to Univ. of Pisa

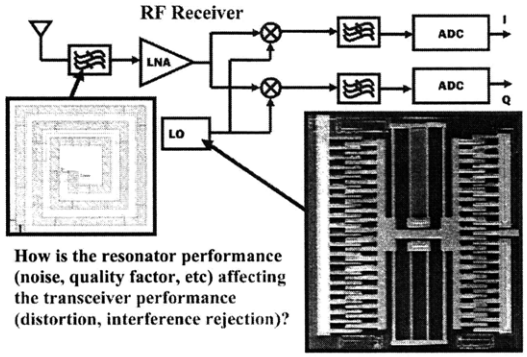
❖ Example: Accelerometer and RF Resonator

- What is the Drag force on the fingers of a resonator or accelerometer?
- How does it affect the quality factor?



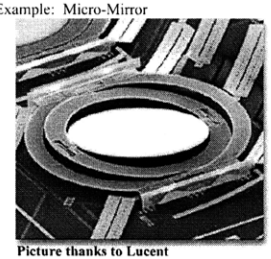
Pictures generated by FastStokes (Thanks to Xin Wang)

❖ Example: Micro-Inductor and Resonator in a Wireless Transceiver

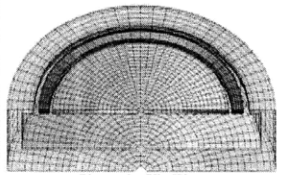


How is the resonator performance (noise, quality factor, etc) affecting the transceiver performance (distortion, interference rejection)?

❖ Example: Micro-Mirror



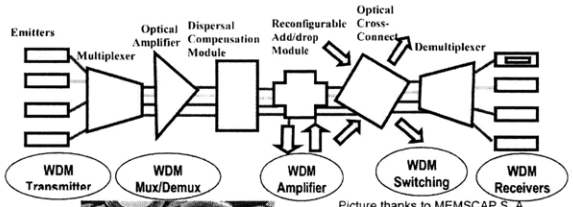
Picture thanks to Lucent



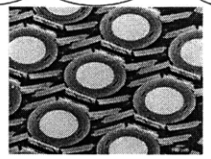
Picture by Xin Wang

What are the forces applied on the mirror, how do they affect the dynamic response of the mirror?

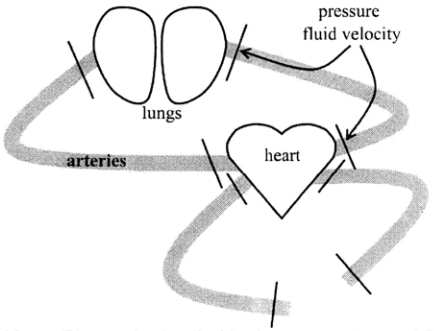
❖ Example: Micromirror Switch in a Dense Wavelength Division Multiplexing Optical Communication System



Picture thanks to MEMSCAP S. A.



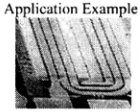
How is the micromirror performance affecting the communication system functionality?

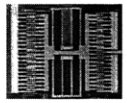


Problem: Want to simulate the blood transport system and find out if the heart is under stress

- Important to account for distributed nature of the arteries and veins
- But the final quantities we want to observe are only input/output e.g. pressure and fluid velocity

❖ Application Examples – Traditional Approach to Generating Models





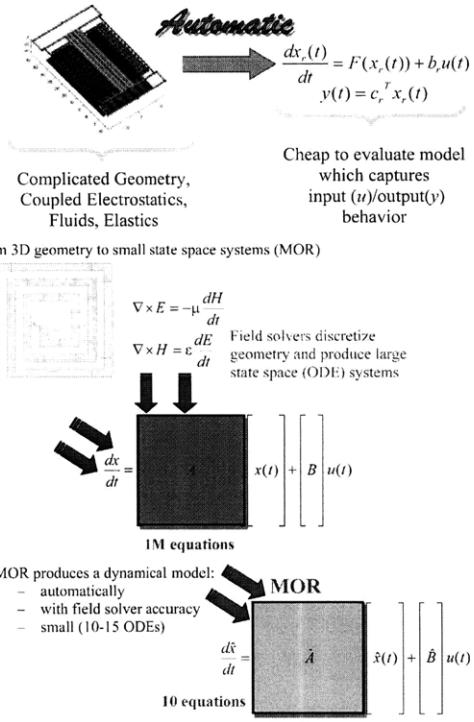
$$\frac{dx_r(t)}{dt} = F(x_r(t)) + b_r u(t)$$

$$y(t) = c_r^T x_r(t)$$

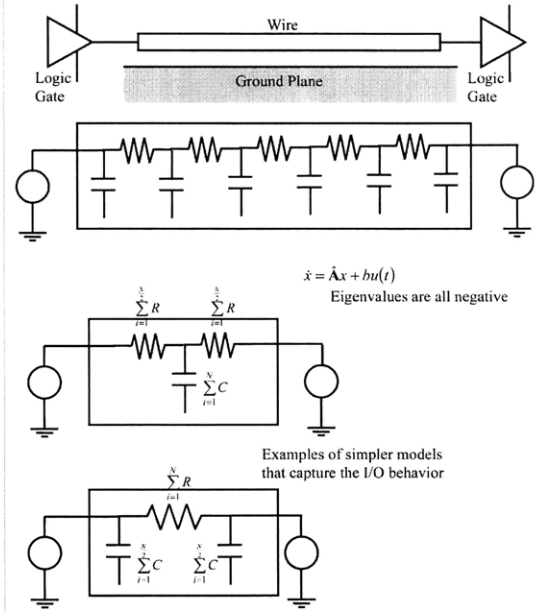
Model for the System Simulator

❖ The Numerical Macromodeling or Model Reduction Paradigm

Generate a Reduced-Order Model Directly from 3-D Geometry and Physics



Signal propagation along a wire



FROM PDE TO A STATE SPACE MODEL

- Compare PDE Solvers and Model Order Reduction
  - PDE Solvers:
    - Accurate
    - Relatively fast (minutes to hours)
    - Challenges:
      - very large matrices: sparse (FD or FEM), or dense (BEM)
    - Application: verification and characterization of component properties
  - Model Order Reduction
    - Preserve PDE solver accuracy
    - Model construction relatively fast (same as PDE solvers minutes to hours)
    - Challenges: same as PDE solvers but only in model construction
    - Only capture input/output behavior (don't show field distributions)
    - But model evaluation in msec (e.g. get dynamical response from any input)
    - Application: analysis of functionality and interaction with other components
- Model Order Reduction State of the Art

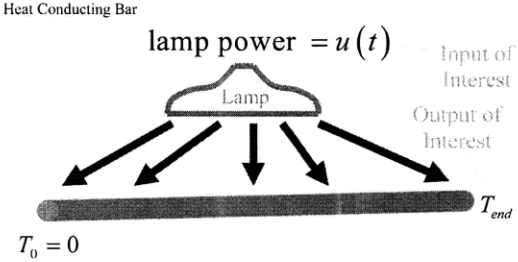
  Model Order Reduction for simple linear systems is well understood (e.g. interconnect, heat diffusion)

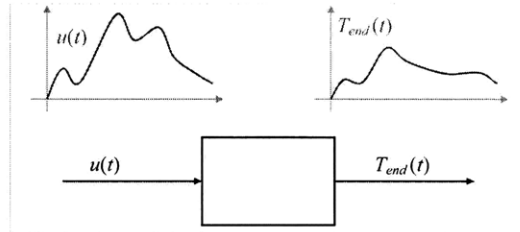
$$\frac{dx}{dt} = A\,x(t) + B\,u(t)$$
$$y(t) = C\,x(t)$$

  Not many techniques yet for NON-LINEAR systems

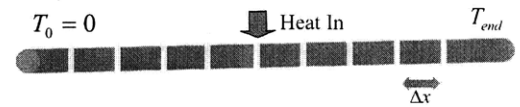
$$\frac{dx}{dt} = F[x(t)] + B\,u(t)$$
$$y(t) = C\,x(t)$$

- Heat Conducting Bar





- Basic Equations

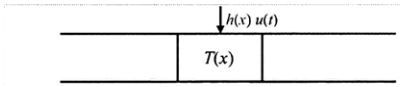


Temperature Differential Equation

$$\underbrace{\gamma}_{\text{specific heat}} \frac{\partial T(x,t)}{\partial t} - \underbrace{\kappa}_{\text{thermal conductivity}} \frac{\partial^2 T(x,t)}{\partial x^2} = h(x)\underbrace{u(t)}_{\text{scalar input}}$$

Spatial Discretization (except at end)

$$\gamma \frac{d\hat{T}_i}{dt} - \frac{\kappa}{(\Delta x)^2}\left(\hat{T}_{i+1} - 2\hat{T}_i + \hat{T}_{i-1}\right) = h(x_i)u(t)$$

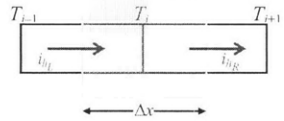


In steady-state $\sum h = 0$ Conservation Law $\Rightarrow -\kappa \frac{\partial^2 T}{\partial x^2} - h(x) = 0$

What if $\sum h \neq 0$? e.g. $h(x)u(t)$ increases suddenly

$$-\kappa \frac{\partial^2 T}{\partial x^2} - h(x) = -\gamma \frac{\partial T}{\partial t}$$

then the temperature will take some time to readjust



$$\Delta x \gamma \cdot \frac{dT_i}{dt} = \text{net heat flow}$$

$$= i_{h_L} - i_{h_R}$$

$$\kappa \frac{T_{i-1} - T_i}{\Delta x} \qquad \kappa \frac{T_i - T_{i+1}}{\Delta x}$$

PDE in time and space

$$\gamma \frac{\partial T}{\partial t} - \kappa \underbrace{\frac{\partial^2 T}{\partial x^2}}_{\substack{\text{1-D} \\ \text{in space}}} = h(x)u(t)$$

$$\frac{d}{dt}\begin{bmatrix} T_1 \\ T_2 \\ \vdots \\ T_m \end{bmatrix} = \begin{bmatrix} 2 & -1 & & & \\ -1 & 2 & -1 & & \\ & -1 & 2 & -1 & \\ & & \ddots & \ddots & \ddots \\ & & & \ddots & \ddots \end{bmatrix} \begin{bmatrix} T_1 \\ T_2 \\ \vdots \\ T_m \end{bmatrix} + \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_m \end{bmatrix} u(t)$$

$$\boxed{\frac{dx}{dt} = \mathbf{A}x(t) + bu(t)}$$

$$y(t) = T_N$$

- Input-Output Discrete Equations

$$\gamma \frac{d\hat{T}_i}{dt} - \frac{\kappa}{(\Delta x)^2}\left(\hat{T}_{i+1} - 2\hat{T}_i + \hat{T}_{i-1}\right) = h(x_i)u(t) \qquad i \in [1, \ldots, N-1]$$

$$\gamma \frac{d\hat{T}_i}{dt} - \frac{\kappa}{(\Delta x)^2}\left(\hat{T}_N - \hat{T}_{N-1}\right) = h(x_N)u(t)$$

$$T_{end} = \hat{T}_N$$

- State-Space Description

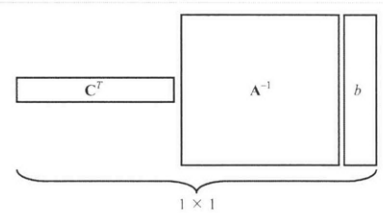
$$\frac{dx(t)}{dt} = \underbrace{A}_{NxN} x(t) + \underbrace{b}_{Nx1} \underbrace{u(t)}_{\substack{\text{scalar} \\ \text{input}}} \qquad \underbrace{y(t)}_{\substack{\text{scalar} \\ \text{output}}} = \underbrace{c^T}_{1xN} x(t)$$

Given the right scaling

$$A = -\begin{bmatrix} 2 & -1 & & & \\ -1 & 2 & -1 & & \\ & -1 & 2 & \ddots & \\ & & \ddots & 2 & -1 \\ & & & -1 & 2 \end{bmatrix} \quad b = \begin{bmatrix} h(x_1) \\ h(x_2) \\ \vdots \\ h(x_N) \end{bmatrix} \quad c = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 1 \end{bmatrix}$$

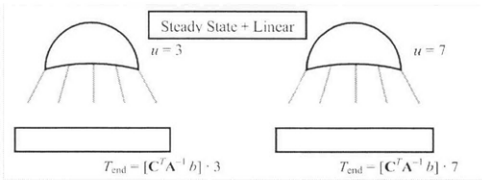
$$\frac{d}{dt}\underbrace{\begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}}_{\text{states}} = \begin{bmatrix} 2 & -1 & & & \\ -1 & 2 & -1 & & \\ & -1 & 2 & -1 & \\ & & \ddots & \ddots & \ddots \\ & & & \ddots & \ddots \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} + \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix} \underbrace{u(t)}_{\substack{\text{scalar} \\ \text{input}}}$$

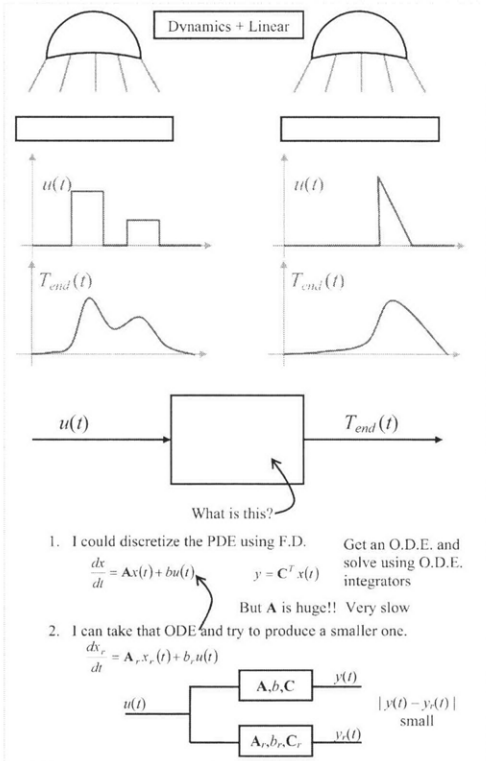
$$\underbrace{y(t)}_{\text{scalar}} = \begin{bmatrix} c_1 & c_2 & \cdots & c_n \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = \sum_{i=1}^{n} c_i x_i(t)$$



1. Small amount of information to give the user
2. Does not give access to interior information
3. But yet accounts for it and produces correct I/O behavior

Note: if someone gives me a different u input I do not need to rediscretize the bar and solve.





1. I could discretize the PDE using F.D.

$$\frac{dx}{dt} = \mathbf{A}x(t) + bu(t) \qquad y = \mathbf{C}^T x(t)$$

Get an O.D.E. and solve using O.D.E. integrators

But **A** is huge!! Very slow

2. I can take that ODE and try to produce a smaller one.

$$\frac{dx_r}{dt} = \mathbf{A}_r x_r(t) + b_r u(t)$$

$|y(t) - y_r(t)|$ small

❖ Dynamic Linear Case – State-Space Description

Original Dynamical System - Single Input/Output

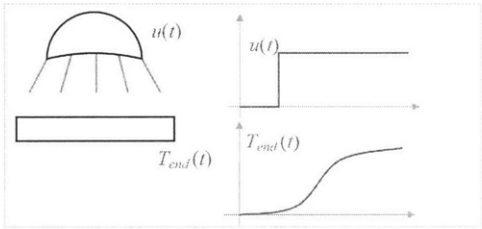
$$\frac{dx(t)}{dt} = \underbrace{A}_{N\times N} x(t) + \underbrace{b}_{N\times 1} \underbrace{u(t)}_{\substack{scalar\\ input}} \qquad \underbrace{y(t)}_{\substack{scalar\\ output}} = \underbrace{c^T}_{1\times N} x(t)$$

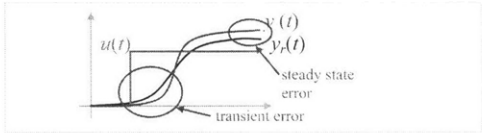
Reduced Dynamical System

$$\frac{dx_r(t)}{dt} = \underbrace{A_r}_{q\times q} x(t) + \underbrace{b_r}_{q\times 1} \underbrace{u(t)}_{\substack{scalar\\ input}} \qquad \underbrace{y_r(t)}_{\substack{scalar\\ output}} = \underbrace{c_r^T}_{1\times q} x_r(t)$$

$q << N$, but input/output behavior preserved

$$\underbrace{\bar{y} = \begin{bmatrix} c_1 & \cdots & c_N \end{bmatrix} \begin{bmatrix} \mathbf{A}^{-1} \end{bmatrix} \begin{bmatrix} b_1 \\ \vdots \\ b_N \end{bmatrix}}_{\text{scalar}} \bar{u}$$





INTRODUCTION TO NUMERICAL SIMULATION

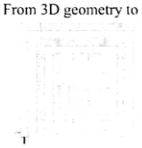
LECTURE 25.
Model-Order Reduction II

TODAY'S OUTLINE:

- Problem Setup: from PDE to large ODE
- Reduction via eigenmode truncation method
- Reduction via transfer function fitting
  - Point Matching
  - Least Square
  - Quasi-convex Optimization Method
- Reduction via Projection Framework

FROM PDE TO LARGE ODE

- From 3D geometry to small state space systems (MOR)



$$\nabla \times E = -\mu \frac{dH}{dt}$$

$$\nabla \times H = \varepsilon \frac{dE}{dt}$$

Field solvers discretize geometry and produce large state space (ODE) systems

$$\frac{dx}{dt} = A\,x(t) + B\,u(t)$$

1M equations

MOR produces a dynamical model:
- automatically
- with field solver accuracy
- small (10-15 ODEs)

MOR

$$\frac{d\hat{x}}{dt} = \hat{A}\,\hat{x}(t) + \hat{B}\,u(t)$$

10 equations

- Dynamic Linear Case – State-Space Description
  - Original Dynamical System – Single Input/Output

$$\frac{dx(t)}{dt} = \underbrace{\mathbf{A}}_{N\times N} x(t) + \underbrace{b}_{N\times 1}\underbrace{u(t)}_{\text{scalar input}} \qquad \underbrace{y(t)}_{\text{scalar output}} = \underbrace{c}_{N\times 1}{}^{T} x(t)$$

INTRODUCTION TO NUMERICAL SIMULATION

LECTURE 25.
Model-Order Reduction II

- Reduced Dynamical System

$$\frac{d\hat{x}(t)}{dt} = \underbrace{\hat{\mathbf{A}}}_{q\times q}\hat{x}(t) + \underbrace{\hat{b}}_{q\times 1}\underbrace{u(t)}_{\text{scalar input}} \qquad \underbrace{\hat{y}(t)}_{\text{scalar output}} = \underbrace{\hat{c}}_{q\times 1}{}^{T}\hat{x}(t)$$

- $q << N$, but input/output behavior preserved

REDUCTION VIA EIGENMODE TRUNCATION METHOD

- Reminder about Eigenanalysis

Consider an ODE : $\dfrac{dx(t)}{dt} = \mathbf{A}x(t) + bu(t), \quad x(0) = 0$

Eigendecomposition :

$$\mathbf{A} = \underbrace{\begin{bmatrix} \uparrow & \uparrow & & \uparrow \\ E_1 & E_2 & \cdots & E_N \\ \downarrow & \downarrow & & \downarrow \end{bmatrix}}_{\mathbf{E}} \begin{bmatrix} \lambda_1 & & \\ & \ddots & \\ & & \lambda_N \end{bmatrix} \begin{bmatrix} \uparrow & \uparrow & & \uparrow \\ E_1 & E_2 & \cdots & E_N \\ \downarrow & \downarrow & & \downarrow \end{bmatrix}^{-1}$$

Change of Variables : $\mathbf{E}w(t) = x(t) \Leftrightarrow w(t) = \mathbf{E}^{-1}x(t)$

Substituting : $\dfrac{d\mathbf{E}w(t)}{dt} = \mathbf{A}\mathbf{E}w(t) + bu(t), \quad \mathbf{E}w(0) = 0$

Multiply by $\mathbf{E}^{-1}$: $\dfrac{dw(t)}{dt} = \mathbf{E}^{-1}\mathbf{A}\mathbf{E}w(t) + \mathbf{E}^{-1}bu(t),$

$$\vec{E}_1 w_1(t) + \vec{E}_2 w_2(t) + \ldots + \vec{E}_N w_N(t) = x(t)$$

Decoupled Equations

$$\frac{d}{dt}\begin{bmatrix} w_1 \\ \vdots \\ w_N \end{bmatrix} = \begin{bmatrix} \lambda_1 & & \\ & \ddots & \\ & & \lambda_N \end{bmatrix}\begin{bmatrix} w_1 \\ \vdots \\ w_N \end{bmatrix} + \underbrace{\begin{bmatrix} (\mathbf{E}^{-1}b)_1 \\ \vdots \\ (\mathbf{E}^{-1}b)_N \end{bmatrix}}_{\hat{b}} u(t)$$

LECTURE 25.
Model-Order Reduction II

$$\frac{dw_i}{dt} = \lambda_i w_i(t) + \tilde{b}_i u(t)$$

how fast mode i responds

how much the input excites mode i
if $\tilde{b}_i = 0 \Rightarrow$ that mode is not controllable – eliminate it!

$$y(t) = \sum_{i=1}^{n} \tilde{c}_i w_i(t)$$

how much the mode i is seen at the output
if $\tilde{c}_i = 0 \Rightarrow$ the mode is not observable – eliminate it!

Output Equation

$$y(t) = c^T x(t) = c^T \mathbf{E} w(t) = \underbrace{\left(\mathbf{E}^T c\right)^T}_{\tilde{c}} w(t)$$

Solving Decoupled Equations

$$w_i(t) = \int_0^t e^{\lambda_i (t-\tau)} \tilde{b}_i u(\tau) d\tau \quad \text{Assuming Zero Initial Conditions}$$

Output Equation

$$y(t) = \sum_{i=1}^{N} \tilde{c}_i w_i(t)$$

$$\frac{d}{dt} w_1 = \lambda_1 w_1 + \tilde{b}_1 u(t)$$
$$\vdots$$
$$\frac{d}{dt} w_N = \lambda_N w_N + \tilde{b}_N u(t)$$

$$y(t) = \sum \tilde{c}_i w_i(t)$$

$\tilde{c}_i = 0$ i^{th} node unobservable!

$\tilde{b}_i = 0$ i^{th} node uncontrollable!

❖ Dynamic Linear Case – Reduced Models via Mode Truncation

$$\begin{bmatrix} \dot{w}_1 \\ \vdots \\ \dot{w}_q \\ \hline \vdots \\ \dot{w}_N \end{bmatrix} = \begin{bmatrix} \lambda_1 & & & & \\ & \ddots & & & \\ & & \lambda_q & & \\ \hline & & & \ddots & \\ & & & & \lambda_N \end{bmatrix} \begin{bmatrix} w_1 \\ \vdots \\ w_q \\ \hline \vdots \\ w_N \end{bmatrix} + \begin{bmatrix} \tilde{b}_1 \\ \vdots \\ \tilde{b}_q \\ \hline \vdots \\ \tilde{b}_N \end{bmatrix} u(t)$$

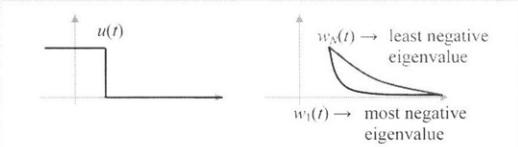
$$\begin{bmatrix} \dot{w}_1 \\ \vdots \\ \dot{w}_q \end{bmatrix} = \begin{bmatrix} \lambda_1 & & \\ & \ddots & \\ & & \lambda_q \end{bmatrix} \begin{bmatrix} w_1 \\ \vdots \\ w_q \end{bmatrix} + \begin{bmatrix} \tilde{b}_1 \\ \vdots \\ \tilde{b}_q \end{bmatrix} u(t)$$

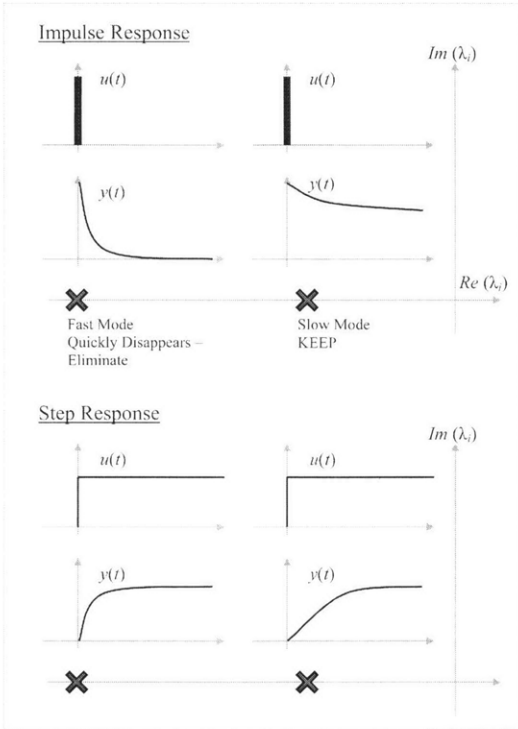
LECTURE 25.
Model-Order Reduction II

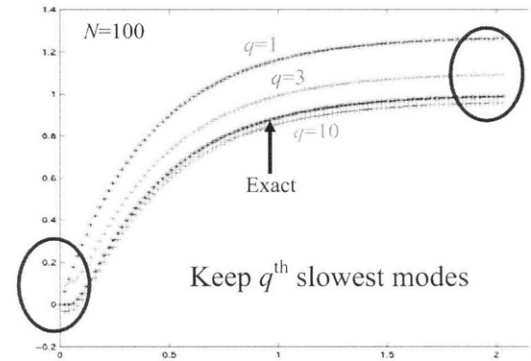
$$y(t) = \sum_{i=1}^{q} \tilde{c}_i w_i(t) \quad \text{Output Equation}$$

- Certain modes are not affected by the input
  $\tilde{b}_{k+1}, \ldots, \tilde{b}_N$ are all small
- Certain modes do not affect the output
  $\tilde{c}_{k+1}, \ldots, \tilde{c}_N$ are all small
- Keep least negative eigenvalues (slowest modes)
  - Look at response to a constant input

$$w_i(t) = \int_0^t e^{\lambda_i (t-\tau)} \tilde{b}_i u(\tau) d\tau = \underbrace{\frac{1}{\lambda_i}\left(\tilde{b}_i u - \tilde{b}_i u e^{\lambda_i t}\right)}_{\text{Small if } |\lambda_i| \text{ large}}$$







Final Value Theorem

For accuracy at steady state of step response need good accuracy for s → 0:

$$\lim_{s\to 0} H(s) = \lim_{s\to 0}\sum_{i=1}^{n}\frac{\tilde{c}_i\tilde{b}_i}{s-\lambda_i} = \sum_{i=1}^{n}\frac{\tilde{c}_i\tilde{b}_i}{\lambda_i}$$

eliminate terms where $\frac{\tilde{c}_i\tilde{b}_i}{\lambda_i}$ is small

$$H(s) = \frac{\tilde{c}_1\tilde{b}_1}{s-\lambda_1} + \cdots + \frac{\tilde{c}_N\tilde{b}_N}{s-\lambda_N}$$

- Transfer Functions
  - Laplace Transform

Consider an ODE : $\frac{dx(t)}{dt} = \mathbf{A}x(t) + bu(t)$

Bilateral Laplace Transform : $X(s) = \int_{-\infty}^{\infty} x(t)e^{-st}dt$

Key Transform Property : $sX(s) = \int_{-\infty}^{\infty}\frac{dx(t)}{dt}e^{-st}dt$

LECTURE 25.
Model-Order Reduction II

Rewrite the ODE in transformed variables

$$sX(s) = \mathbf{A}X(s) + bU(s) \qquad Y(s) = c^T X(s)$$

$$\Rightarrow Y(s) = \underbrace{c^T (sI - \mathbf{A})^{-1} b}_{\substack{H(s) \\ \text{Transfer Function}}} \mathbf{U}(s)$$

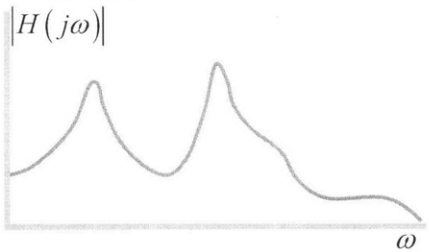
$$sX(s) = \mathbf{A}x(s) + bU(s)$$
$$(sI - \mathbf{A})X(s) = bU(s)$$

- Meaning of H(s)

  For Stable Systems, $H(j\omega)$ is the frequency response

  If $u(t) = e^{j\omega t}$ ← sinusoid

  then $y(t) = H(j\omega)e^{j\omega t}$ ← sinusoid with shifted phase and amplitude



- EigenAnalysis

  Transfer Function

  $$H(s) = c^T (sI - \mathbf{A})^{-1} b$$

  Apply Eigendecomposition $\mathbf{A} = \mathbf{E}\lambda\mathbf{E}^{-1}$

  $$H(s) = c^T \mathbf{E}(sI - \mathbf{A})^{-1} \mathbf{E}^{-1} b$$

  $$= \tilde{c}^T \begin{bmatrix} \frac{1}{s-\lambda_1} & & \\ & \ddots & \\ & & \frac{1}{s-\lambda_N} \end{bmatrix} \tilde{b} \Rightarrow H(s) = \sum_{i=1}^{N} \frac{\tilde{c}_i \tilde{b}_i}{s-\lambda_i}$$

Eliminate each mode for which this term is small.

LECTURE 25.
Model-Order Reduction II

$$H(s) = \frac{\tilde{b}_1\tilde{c}_1}{s-\lambda_1} + \frac{\tilde{b}_2\tilde{c}_2}{s-\lambda_2} + \cdots + \frac{\tilde{b}_N\tilde{c}_N}{s-\lambda_N}$$

leave out $w_i(t)$ or leave out $\frac{\tilde{b}_i\tilde{c}_i}{s-\lambda_i}$ from $H(s)$

$$\mathbf{A} = \mathbf{E}\Lambda\mathbf{E}^{-1}$$

$$\begin{aligned} H(s) &= \mathbf{C}^T\left(s\mathbf{I} - \mathbf{E}\Lambda\mathbf{E}^{-1}\right)^{-1} b \\ &= \mathbf{C}^T\left(s\mathbf{E}\mathbf{E}^{-1} - \mathbf{E}\Lambda\mathbf{E}^{-1}\right)^{-1} b \\ &= \mathbf{C}^T\mathbf{E}(s\mathbf{I} - \mathbf{\Lambda})^{-1}\mathbf{E}^{-1} b = \left[\ (\mathbf{E}^T\mathbf{C})^T\right] \begin{bmatrix} \frac{1}{s-\lambda_1} & & \\ & \ddots & \\ & & \frac{1}{s-\lambda_n} \end{bmatrix} \begin{bmatrix} (\mathbf{E}^{-1}b)_1 \\ \vdots \\ (\mathbf{E}^{-1}b)_n \end{bmatrix} \\ &= \sum_{i=1}^{n} \frac{\left[\mathbf{C}^T\mathbf{E}\right]_i \left[\mathbf{E}^{-1}b\right]_i}{s-\lambda_i} \end{aligned}$$

Pole-Residue Form

$$\mathbf{A} = \mathbf{E}\begin{bmatrix} \lambda_1 & & \\ & \ddots & \\ & & \lambda_n \end{bmatrix}\mathbf{E}^{-1} \longleftarrow \text{Assume } \mathbf{A} \text{ diagonalizable}$$

$$\underbrace{\mathbf{C}^T\mathbf{E}}_{\tilde{c}}\left(sI - \begin{bmatrix} \lambda_1 & & \\ & \ddots & \\ & & \lambda_n \end{bmatrix}\right)^{-1} \underbrace{\mathbf{E}^{-1}b}_{\tilde{b}} = \frac{\tilde{b}_1\tilde{c}_1}{s-\lambda_1} + \frac{\tilde{b}_2\tilde{c}_2}{s-\lambda_2} + \cdots$$

❖ Model Order Reduction via Eignmode Analysis

Pole - Residue Form

$$H(s) = \sum_{i=1}^{n} \frac{\tilde{c}_i\tilde{b}_i}{s-\lambda_i}$$

$$h(t) = \sum_{i=1}^{n} \tilde{c}_i\tilde{b}_i e^{\lambda_i t}$$

Pole - Zero Form (SISO)

$$H(s) = \frac{\prod_{i=1}^{n-1}(s-\zeta_i)}{\prod_{i=1}^{n}(s-\lambda_i)}$$

Ideas for reducing order:

- Drop terms with small residues $\tilde{c}_i\tilde{b}_i$
- Drop terms with large negative Re (λ_i) ("fast" modes)
- Remove pole/zero near-cancellations
- Cluster poles that are "together"

LECTURE 25.
Model-Order Reduction II

REDUCTION VIA TRANSFER FUNCTION FITTING

- Counting Degrees of Freedom

Dynamical System — **Degrees of Freedom**

$$\frac{d\hat{x}}{dt} = \hat{\mathbf{A}}\hat{x}(t) + \hat{b}u(t)$$
$$\hat{y}(t) = \hat{c}^T\hat{x}(t)$$

q^2+2q coefficients

Transfer Function — **Degrees of Freedom**

$$H(s) = \frac{r_1}{(s-\lambda_1)} + \cdots + \frac{r_n}{(s-\lambda_n)} = k\frac{(s-\zeta_1)\cdots(s-\zeta_{n-1})}{(s-\lambda_1)\cdots(s-\lambda_n)} = \frac{b_0 + b_1 s + \cdots + b_{n-1}s^{n-1}}{1 + a_1 s + \cdots + a_n s^n}$$

$2q$ coefficients

- Fully Invertible Change of Coordinates

Reduced Model Transfer Function

$$\frac{d\hat{x}}{dt} = \hat{\mathbf{A}}\hat{x}(t) + \hat{b}u(t) \quad \Rightarrow \quad H(s) = \hat{c}^T(sI - \hat{\mathbf{A}})^{-1}\hat{b}$$
$$\hat{y}(t) = \hat{c}^T\hat{x}(t)$$

Apply any invertible change of coordinates to the state $\hat{x}(t) = \mathbf{U}\tilde{x}(t)$

$$\underbrace{\mathbf{U}^{-1}\mathbf{U}}_{I}\frac{d\tilde{x}}{dt} = \mathbf{U}^{-1}\hat{\mathbf{A}}\mathbf{U}\tilde{x}(t) + \mathbf{U}^{-1}\hat{b}u(t) \quad \Rightarrow \quad \tilde{H}(s) = \hat{c}^T(sI - \hat{\mathbf{A}})^{-1}\hat{b} = H(s)$$
$$y(t) = \hat{c}^T\mathbf{U}\tilde{x}(t)$$

Many Dynamical System have the same transfer function!!!

$$\tilde{H}(s) = \mathbf{C}^T\mathbf{U}(s\mathbf{I} - \mathbf{U}^{-1}\mathbf{A}\mathbf{U})^{-1}\mathbf{U}^{-1}b = \mathbf{C}^T\mathbf{U}\mathbf{U}^{-1}(s\mathbf{I} - \mathbf{A})^{-1}\mathbf{U}\mathbf{U}^{-1}b = \mathbf{C}^T(s\mathbf{I} - \mathbf{A})^{-1}b$$

$$\mathbf{U} = \overbrace{\begin{bmatrix} \uparrow & \uparrow & & \uparrow \\ u_1 & u_2 & \cdots & u_q \\ \downarrow & \downarrow & & \downarrow \end{bmatrix}}^{N} \times$$

- Model Order Reduction via Rational Transfer Function Fitting

Original System Transfer Function:

$$H(s) = \frac{b_0 + b_1 s + \cdots + b_{N-1}s^{N-1}}{1 + a_1 s + \cdots + a_N s^N}$$

rational function

LECTURE 25.
Model-Order Reduction II

Model Reduction =Find a low order ($q << N$) rational function matching

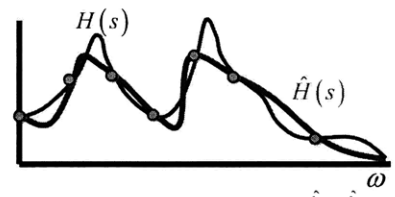
$$\hat{H}(s) = \frac{\hat{b}_0 + \hat{b}_1 s + \cdots + \hat{b}_{q-1}s^{q-1}}{1 + \hat{a}_1 s + \cdots + \hat{a}_q s^q}$$

reduced order rational function

$$(sI - \mathbf{A})\mathbf{X}(s) = b\mathbf{U}(s)$$
$$\mathbf{X}(s) = (sI - \mathbf{A})^{-1}b\mathbf{U}(s)$$
$$\mathbf{Y}(s) = \underbrace{\mathbf{C}^T(sI - \mathbf{A})^{-1}b}_{\mathbf{H}(s)}\mathbf{U}(s)$$

- Point Matching
  - Rational Transfer Function Fitting: via Point Matching

$$\underbrace{H(s_i)}_{\text{original}} = \underbrace{\frac{b_0^r + b_1^r s_i^1 + b_2^r s_i^2 + \ldots + b_{q-1}^r s_i^{q-1}}{1 + a_1^r s_i^1 + \cdots + a_q^r s_i^q}}_{H_r(s)}$$



- **Can match 2q points** $\hat{H}(s_i) = \dfrac{\hat{b}_0 + \hat{b}_1 s_i + \cdots + \hat{b}_{q-1}s_i^{q-1}}{1 + \hat{a}_1 s_i + \cdots + \hat{a}_q s_i^q}$
- **cross multiplying generates a linear system**

  For i = 1 to $2q$

$$\left(1 + \hat{a}_1 s_i + \cdots + \hat{a}_q s_i^q\right)\hat{H}(s_i) - \left(\hat{b}_0 + \hat{b}_1 s_i + \cdots + \hat{b}_{q-1}s_i^{q-1}\right) = 0$$

LECTURE 25.
Model-Order Reduction II

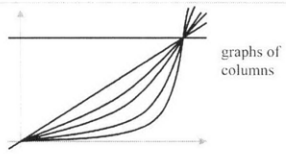
- Point Matching can be ill-conditioned

$$\begin{bmatrix} s_1 H(s_1) & \cdots & s_1^q H(s_1) & -1 & -s_1 & \cdots & -s_1^{q-1} \\ s_2 H(s_2) & \cdots & s_2^q H(s_2) & -1 & -s_2 & \cdots & -s_2^{q-1} \\ \vdots & & & & & & \vdots \\ s_{2q} H(s_{2q}) & \cdots & s_{2q}^q H(s_{2q}) & -1 & -s_{2q} & \cdots & -s_{2q}^{q-1} \end{bmatrix} \begin{bmatrix} a_1 \\ \vdots \\ a_q \\ b_0 \\ b_1 \\ \vdots \\ b_{q-1} \end{bmatrix} = \begin{bmatrix} H(s_1) \\ H(s_2) \\ \vdots \\ H(s_{2q}) \end{bmatrix}$$

  - Columns contain progressively higher powers of the test frequencies: problem is numerically ill-conditioned
  - Also… missing data can cause severe accuracy problems

$$s_1 H(s_1) a_1 + \ldots + s_1^q H(s_1) a_q^r - \left(b_0 + b_1 s_1 + \ldots + b_{q-1} s^{q-1}\right) = 0$$

$s_i \in [0,1]$

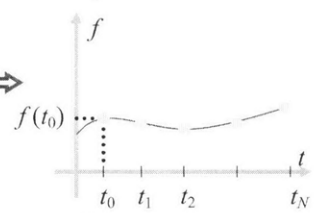


- Hard to Solve Systems – Fitting Example

Polynomial Interpolation

Table of Data

t_0	$f(t_0)$
t_1	$f(t_1)$
$\vdots$	$\vdots$
t_N	$f(t_N)$



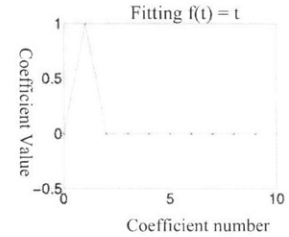
Problem fit data with an Nth order polynomial

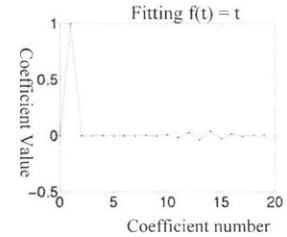
$$f(t) = \alpha_0 + \alpha_1 t + \alpha_2 t^2 + \ldots + \alpha_N t^N$$

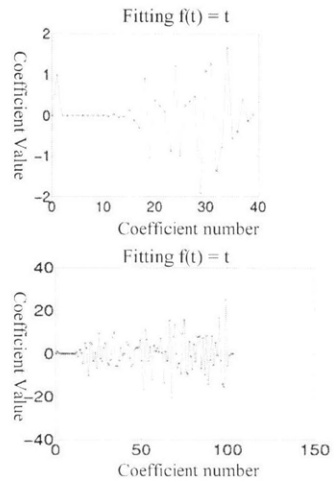
LECTURE 25.
Model-Order Reduction II

Matrix Form

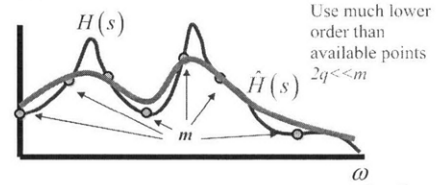
$$\underbrace{\begin{bmatrix} 1 & t_0 & t_0^2 & & t_0^N \\ 1 & t_1 & t_1^2 & & t_1^N \\ & & & & \\ 1 & t_N & t_N^2 & & t_N^N \end{bmatrix}}_{M_{interp}} \begin{bmatrix} \alpha_0 \\ \alpha_1 \\ \\ \alpha_N \end{bmatrix} = \begin{bmatrix} f(t_0) \\ f(t_1) \\ \\ f(t_N) \end{bmatrix}$$







- Least Square



$$\begin{bmatrix} s_1H(s_1) & \cdots & s_1^qH(s_1) & -1 & -s_1 & \cdots & -s_1^{q-1} \\ s_2H(s_2) & \cdots & s_2^qH(s_2) & -1 & -s_2 & \cdots & -s_2^{q-1} \\ \vdots & & & & & & \vdots \\ s_mH(s_m) & \cdots & s_m^qH(s_m) & -1 & -s_m & \cdots & -s_m^{q-1} \end{bmatrix} \overset{\omega}{\begin{bmatrix} a_1 \\ \vdots \\ a_q \\ b_0 \\ b_1 \\ \vdots \\ b_{q-1} \end{bmatrix}} = \begin{bmatrix} H(s_1) \\ H(s_2) \\ \vdots \\ H(s_m) \end{bmatrix}$$

- Cross-multiplying generates a linear TALL SKINNY system
- Use, for instance, QR to solve it or Gauss- Newton

- Quasi-Convex Optimization Method
  - Optimization Based Rational Fit Model Order Reduction Setup

$$\underset{p(s),q(s)}{\text{minimize}} \left\| H(s) - \frac{p(s)}{q(s)} \right\|$$

from field solver OR measurements

small stable and passive reduced order model

Least Square Method

- Cast as nonlinear least squares (solved by e.g. Gauss-Newton)
- Do not consider stability or passivity while finding poles (need post-processing)

Quasi Convex Method

- Cast as quasi-convex program (solved by convex optimization algorithm)
- Explicitly take care of stability and passivity while finding poles

LECTURE 25.
Model-Order Reduction II

- Relaxation of the H-inf norm MOR setup [Sou, Megretski, Daniel]

$$\underset{p,q,r}{\text{minimize}} \quad \left\| H(z) - \frac{p(z)}{q(z)} - \frac{r(z^{-1})}{q(z^{-1})} \right\|_\infty$$

Anti-stable term

$$\text{subject to} \quad \deg(q) = m, \quad \deg(p) \le m, \quad \deg(r) < m$$

Stability: $q(z)$ Schur polynomial (roots inside unit circle)
Passivity, and possibly other constraints
Benefit: Relaxation *equivalent* to a quasi-convex program
Drawback: May obtain suboptimal solutions

- How bad is our relaxation?

THEOREM:

$$\text{Let } (q^*, p^*, r^*) = \arg\min_{q,p,r} \left\| H(z) - \frac{p(z)}{q(z)} - \frac{r(z^{-1})}{q(z^{-1})} \right\|_\infty$$

Such that $\deg(q) = m$, $q(z)$ is Schur polynomial

$$\text{Then } \left\| H(z) - \frac{p(z)^*}{q(z)^*} \right\|_\infty < m\sigma_{m+1}(H)$$

$m+1^{\text{th}}$ Hankel singular value

- Equivalent Quasi-Convex Setup

$$\underset{a,b,c}{\text{minimize}} \quad \left\| H(e^{j\omega}) - \frac{b(e^{j\omega}) + jc(e^{j\omega})}{a(e^{j\omega})} \right\|$$

quasi-convex function

$$\deg(a) = m, \quad \deg(b) \le m$$

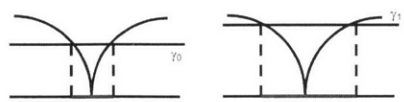
$$\deg(c) \le m,$$

subject to

Stability: $a(e^{j\omega}) > 0, \forall \omega \in [0, \pi]$

Passivity: $b(e^{j\omega}) > 0, \forall \omega \in [0, \pi]$

convex set

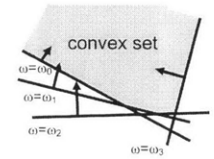


LECTURE 25.
Model-Order Reduction II

This is a *quasi-convex* program, because

$$a(e^{j\omega}) = 2\cos(m\omega) + 2\cos((m-1)\omega)a_{m-1} + \ldots + a_0 > 0$$

defines an intersection of halfspaces



This is a *quasi-convex* program, because

$$2\cos\left(\begin{bmatrix} m\omega_0 & (m-1)\omega_0 & \ldots & 0 \end{bmatrix}\right)\begin{bmatrix} 1 & a_{m-1} & \ldots & a_0 \end{bmatrix}' > 0$$
$$2\cos\left(\begin{bmatrix} m\omega_1 & (m-1)\omega_1 & \ldots & 0 \end{bmatrix}\right)\begin{bmatrix} 1 & a_{m-1} & \ldots & a_0 \end{bmatrix}' > 0$$
$$2\cos\left(\begin{bmatrix} m\omega_2 & (m-1)\omega_2 & \ldots & 0 \end{bmatrix}\right)\begin{bmatrix} 1 & a_{m-1} & \ldots & a_0 \end{bmatrix}' > 0$$
$$2\cos\left(\begin{bmatrix} m\omega_3 & (m-1)\omega_3 & \ldots & 0 \end{bmatrix}\right)\begin{bmatrix} 1 & a_{m-1} & \ldots & a_0 \end{bmatrix}' > 0$$

defines an intersection of halfspaces

- Solving the Quasi-Convex Program

$$\underset{a,b,c}{\text{minimize}} \quad \left\| H(e^{j\omega}) - \frac{b(e^{j\omega}) + jc(e^{j\omega})}{a(e^{j\omega})} \right\|$$

quasi-convex set

$$\deg(a) = m, \quad \deg(b) \le m$$

$$\deg(c) \le m,$$

subject to

Stability: $a(e^{j\omega}) > 0, \forall \omega \in [0, \pi]$

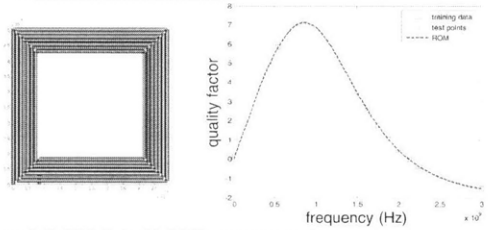
Passivity: $b(e^{j\omega}) > 0, \forall \omega \in [0, \pi]$

convex set

Standard problem.
Use for example by the ellipsoid algorithm

LECTURE 25.
Model-Order Reduction II

- Example 2: RF Inductor with Substrate (from field solver)
  - RF inductor with substrate effect captured by layered Green's function [Hu Dac 05]
  - System matrices are frequency dependent
  - Full model has infinite order
  - Reduced model has order 6

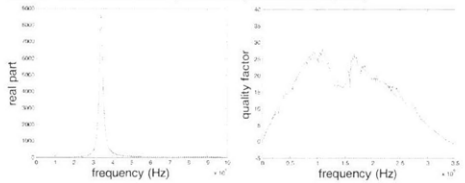


- Example 3: RF Inductor Model (from measurement)

  Fabricated 7 turn spiral inductor

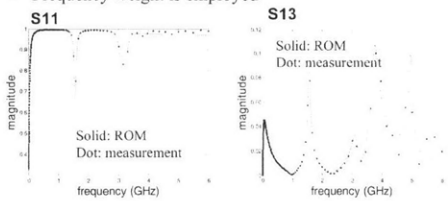
  Blue: measurement

  Red: 10th order reduced model (positive real part constraint imposed)

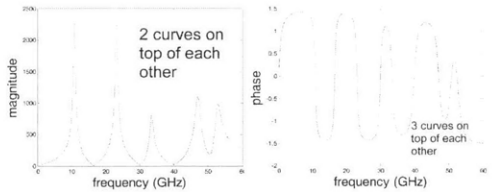


LECTURE 25.
Model-Order Reduction II

- Example 4: Model of graphic card package (from measurement)
  - Industry example of a multi-port device (390 frequency samples)
  - 12th order SISO reduced models are constructed
  - Bounded realness constraint is imposed
  - Frequency weight is employed



- Example 5: Large IC power distribution grid (from field solver)
  - Power distribution grid (dimension size = 7mm, wire width = 2 μm)
  - Blue: full model (order 2046)
  - Red: QCO 40th order reduced model (positive real)



LECTURE 25.
Model-Order Reduction II

REDUCTION VIA PROJECTION FRAMEWORK

❖ Counting Degrees of Freedom

Dynamical System — **Degrees of Freedom**

$$\frac{dx}{dt} = \mathbf{A}x(t) + bu(t)$$
$$y(t) = c^T x(t)$$

$n^2 + 2n$ coefficients

Transfer Function — **Degrees of Freedom**

$$H(s) = \frac{r_1}{(s-\lambda_1)} + \cdots + \frac{r_n}{(s-\lambda_n)}$$
$$= k\frac{(s-\zeta_1)\cdots(s-\zeta_{n-1})}{(s-\lambda_1)\cdots(s-\lambda_n)}$$
$$= \frac{b_0 + b_1 s + \cdots + b_{n-1}s^{n-1}}{1 + a_1 s + \cdots + a_n s^n}$$

$2n$ coefficients

❖ Fully Invertible Change of Coordinates

Reduced Model Transfer Function

$$\frac{dx}{dt} = \mathbf{A}x(t) + bu(t) \quad \Rightarrow \quad H(s) = c^T(sI - \mathbf{A})^{-1} b$$
$$y(t) = c^T x(t)$$

Apply any invertible change of coordinates to the state $x(t) = \mathbf{V}\tilde{x}(t)$

$$\underbrace{\mathbf{V}^{-1}\mathbf{V}}_{I}\frac{d\tilde{x}}{dt} = \mathbf{V}^{-1}\mathbf{A}\mathbf{V}\tilde{x}(t) + \mathbf{V}^{-1}bu(t) \quad \Rightarrow \quad \tilde{H}(s) = c^T(sI - \mathbf{A})^{-1} b = H(s)$$
$$y(t) = c^T\mathbf{V}\tilde{x}(t)$$

Many Dynamical System have the same transfer function!!!

$$(sI - \mathbf{V}^{-1}\mathbf{A}\mathbf{V})\tilde{\mathbf{X}}(s) = \mathbf{V}^{-1}b\mathbf{U}(s)$$
$$\tilde{\mathbf{X}}(s) = (sI - \mathbf{V}^{-1}\mathbf{A}\mathbf{V})^{-1}\mathbf{V}^{-1}b\mathbf{U}(s)$$
$$\tilde{\mathbf{X}}(s) = (s\mathbf{V}^{-1}\mathbf{V} - \mathbf{V}^{-1}\mathbf{A}\mathbf{V})^{-1}\mathbf{V}^{-1}b\mathbf{U}(s)$$
$$\tilde{\mathbf{X}}(s) = \mathbf{V}^{-1}(sI - \mathbf{A})^{-1}\mathbf{V}\mathbf{V}^{-1}b\mathbf{U}(s)$$
$$\mathbf{Y}(s) = \mathbf{C}^T\mathbf{V}\mathbf{V}^{-1}(sI - \mathbf{A})^{-1}\mathbf{V}\mathbf{V}^{-1}b\mathbf{U}(s)$$

LECTURE 25.
Model-Order Reduction II

❖ Eigenanalysis

Consider an ODE: $\frac{dx(t)}{dt} = \mathbf{A}x(t) + bu(t), \quad x(0) = 0$

Eigendecomposition:

$$\mathbf{A} = \underbrace{\begin{bmatrix} \uparrow & \uparrow & & \uparrow \\ E_1 & E_2 & \cdots & E_N \\ \downarrow & \downarrow & & \downarrow \end{bmatrix}}_{\mathbf{E}} \begin{bmatrix} \lambda_1 & & \\ & \ddots & \\ & & \lambda_N \end{bmatrix} \begin{bmatrix} \uparrow & \uparrow & & \uparrow \\ E_1 & E_2 & \cdots & E_N \\ \downarrow & \downarrow & & \downarrow \end{bmatrix}^{-1}$$

Change of Variables: $\mathbf{E}w(t) = x(t) \Leftrightarrow w(t) = \mathbf{E}^{-1}x(t)$

Substituting: $\frac{d\mathbf{E}w(t)}{dt} = \mathbf{A}\mathbf{E}w(t) + bu(t), \quad \mathbf{E}w(0) = 0$

Multiply by $\mathbf{E}^{-1}$: $\frac{dw(t)}{dt} = \mathbf{E}^{-1}\mathbf{A}\mathbf{E}w(t) + \mathbf{E}^{-1}bu(t).$

❖ Eigenvalue Method – Reduced Models via Mode Truncation

$$\begin{bmatrix} \dot{w}_1 \\ \vdots \\ \dot{w}_q \\ \vdots \\ \dot{w}_N \end{bmatrix} = \begin{bmatrix} \lambda_1 & & & & \\ & \ddots & & & \\ & & \lambda_q & & \\ & & & \ddots & \\ & & & & \lambda_N \end{bmatrix} \begin{bmatrix} w_1 \\ \vdots \\ w_q \\ \vdots \\ w_N \end{bmatrix} + \begin{bmatrix} \tilde{b}_1 \\ \vdots \\ \tilde{b}_q \\ \vdots \\ \tilde{b}_N \end{bmatrix} u(t)$$

$$\begin{bmatrix} \dot{w}_1 \\ \vdots \\ \dot{w}_q \end{bmatrix} = \begin{bmatrix} \lambda_1 & & \\ & \ddots & \\ & & \lambda_q \end{bmatrix} \begin{bmatrix} w_1 \\ \vdots \\ w_q \end{bmatrix} + \begin{bmatrix} \tilde{b}_1 \\ \vdots \\ \tilde{b}_q \end{bmatrix} u(t)$$

❖ Projection Framework: Noninvertible Change of Coordinates

Note: $q << N$

$$\begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_{N-1} \\ x_N \end{bmatrix} \approx \underbrace{\begin{bmatrix} \uparrow & & \uparrow \\ \bar{v}_1 & \cdots & \bar{v}_q \\ \downarrow & & \downarrow \end{bmatrix}}_{\hat{\mathbf{V}}_q} \begin{bmatrix} \hat{x}_1 \\ \vdots \\ \hat{x}_q \end{bmatrix}$$

Note: $q << N$

reduced state

original state

$$\frac{d}{dt}\begin{bmatrix} \\ x \\ \\ \end{bmatrix} = \begin{bmatrix} & & \\ & \mathbf{A} & \\ & & \end{bmatrix}\begin{bmatrix} \\ x \\ \\ \end{bmatrix} + \begin{bmatrix} \\ b \\ \\ \end{bmatrix} u(t)$$

$$\frac{d}{dt}\begin{bmatrix} \\ \hat{x} \\ \\ \end{bmatrix} = \begin{bmatrix} & & \\ & \hat{\mathbf{A}} & \\ & & \end{bmatrix}\begin{bmatrix} \\ \hat{x} \\ \\ \end{bmatrix} + \begin{bmatrix} \\ \hat{b} \\ \\ \end{bmatrix} u(t)$$

$$\begin{bmatrix} \\ x \\ \\ \end{bmatrix} = \begin{bmatrix} \begin{bmatrix} \\ \vec{v}_1 \\ \\ \end{bmatrix} & \begin{bmatrix} \\ \vec{v}_2 \\ \\ \end{bmatrix} & \cdots & \begin{bmatrix} \\ \vec{v}_q \\ \\ \end{bmatrix} \end{bmatrix}\begin{bmatrix} \\ \hat{x} \\ \\ \end{bmatrix}$$

- Original System

$$\frac{dx}{dt} = \mathbf{A}x(t) + bu(t)$$
$$y(t) = c^T x(t)$$

- Substitute $x \leftarrow \mathbf{V}_q \hat{x}(t)$

$$\mathbf{V}_q \frac{d\hat{x}}{dt} = \mathbf{A}\mathbf{V}_q \hat{x}(t) + bu(t)$$
$$\hat{y}(t) = c^T \mathbf{V}_q \hat{x}(t)$$

- Notes: now few variables ($q << N$) in the state, but still thousands of equations (N)

INTRODUCTION TO NUMERICAL SIMULATION

LECTURE 25.
Model-Order Reduction II

$$\hat{\mathbf{A}} = \underbrace{\begin{bmatrix} \mathbf{U}^T \\ q \times N \end{bmatrix}\begin{bmatrix} \mathbf{A} \\ N \times N \end{bmatrix}\begin{bmatrix} \mathbf{V} \\ N \times q \end{bmatrix}}_{q \times q} \qquad \mathbf{U}^T\mathbf{V} = I$$

$$V_q \frac{d\hat{x}(t)}{dt} = A V_q \hat{x}(t) + b\, u(t)$$
$$y(t) = c^T V_q \hat{x}(t)$$

Reduction of number of equations: test by multiplying by V_q^T

$$U_q^T V_q \frac{d\hat{x}(t)}{dt} = U_q^T A V_q \hat{x}(t) + U_q^T b\, u(t)$$
$$y(t) = c^T V_q \hat{x}(t)$$

$U_q^T V_q = I$

If U_q^T and V_q^T are chosen biorthogonal

$$\frac{d\hat{x}(t)}{dt} = \hat{A}\hat{x}(t) + \hat{b}\, u(t)$$
$$y(t) = \hat{c}^T \hat{x}(t)$$

$$V_q^T = \begin{bmatrix} 1 & 0 & 0 & & \vdots & \vdots & \vdots \\ 0 & 1 & 0 & & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & & 0 & 1 & 0 \\ \vdots & \vdots & \vdots & & 0 & 0 & 1 \end{bmatrix}$$

eliminate all the equations except for the first three and last three

$$V_q^T = \begin{bmatrix} 1 & 1 & 1 & & & & & \\ & & & 1 & 1 & 1 & & \\ & & & & & & 1 & 1 & 1 \\ & & & & & & & \end{bmatrix}$$

Add groups of three

$$\frac{dx}{dt} = \underbrace{A}_{\text{nxn}} x + bu$$

$$\underbrace{V_q}_{\text{nxq}} \frac{d\hat{x}}{dt} = \underbrace{A}_{\text{nxn}} \underbrace{V_q}_{\text{nxq}} \hat{x} + bu$$

$$\frac{d\hat{x}}{dt} = \underbrace{\underbrace{U_q^T}_{\text{qxn}} \underbrace{A}_{\text{nxn}} \underbrace{V_q}_{\text{nxq}}}_{\frac{d\hat{x}}{dt} = \underbrace{\hat{A}}_{\text{qxq}} \hat{x}(t) + \hat{b}u(t)} \hat{x} + U_q^T bu$$

$$\frac{dx}{dt} = A\,x(t) + b\,u(t)$$

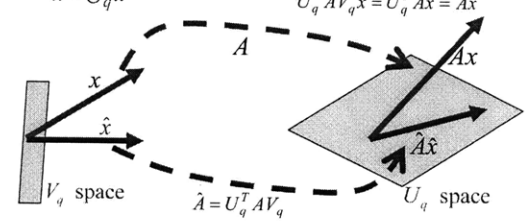
Non-invertible change of coordinates (Projection)

$$x = U_q \hat{x}$$

$$\frac{d\hat{x}}{dt} = \hat{A}\,\hat{x}(t) + \hat{b}u(t)$$

Equation Testing (Projection)

$$U_q^T A V_q \hat{x} = U_q^T A x = \hat{A}\hat{x}$$



LECTURE 25.
Model-Order Reduction II

- Approaches for picking V and U
  - Use Eigenvectors of the system matrix (modal analysis)

Eigenmode Analysis

$$\mathbf{A} = \mathbf{A}^T$$

$$\mathbf{A} = \mathbf{E}\Lambda\mathbf{E}^{-1} = \begin{bmatrix} \uparrow & \uparrow & & \uparrow \\ E_1 & E_2 & \cdots & E_N \\ \downarrow & \downarrow & & \downarrow \end{bmatrix} \begin{bmatrix} \lambda_1 & & & \\ & \lambda_2 & & \\ & & \ddots & \\ & & & \lambda_N \end{bmatrix} \begin{bmatrix} \uparrow & \uparrow & & \uparrow \\ E_1 & E_2 & \cdots & E_N \\ \downarrow & \downarrow & & \downarrow \end{bmatrix}^{-1}$$

$$x(t) = \mathbf{E}\tilde{x} = \begin{bmatrix} \uparrow & \uparrow & & \uparrow & \uparrow & & \uparrow \\ E_1 & E_2 & \cdots & E_q & E_{q+1} & \cdots & E_N \\ \downarrow & \downarrow & & \downarrow & \downarrow & & \downarrow \end{bmatrix} \tilde{x}$$

pick some of these

Invertible change of coordinates

$$x = \underbrace{\begin{bmatrix} \uparrow & \uparrow & & \uparrow \\ E_1 & E_2 & \cdots & E_q \\ \downarrow & \downarrow & & \downarrow \end{bmatrix}}_{\mathbf{V}_q} \hat{x}$$

q eigenvectors

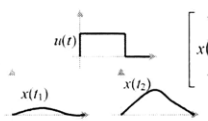
Non-invertible change of coordinates

$$\mathbf{V}^T = ? \qquad \mathbf{V}^T\mathbf{V} = I$$

$$\mathbf{V}_q = \begin{bmatrix} \uparrow & \uparrow & \uparrow & & \uparrow \\ E_1 & E_2 & E_3 & \cdots & E_q \\ \downarrow & \downarrow & \downarrow & & \downarrow \end{bmatrix}$$

$$\frac{dx}{dt} = \mathbf{A}x(t) + bu(t)$$

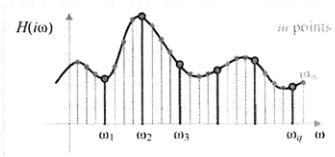
$$u(t) \to x(t)$$



$$\begin{bmatrix} \uparrow \\ x(t_1) \\ \downarrow \end{bmatrix} \begin{bmatrix} \uparrow \\ x(t_2) \\ \downarrow \end{bmatrix} \cdots \begin{bmatrix} \uparrow \\ x(t_q) \\ \downarrow \end{bmatrix}$$

$$\mathbf{V}_q = \underbrace{\begin{bmatrix} \uparrow & \uparrow & \uparrow & & \uparrow \\ x(t_1) & x(t_2) & x(t_3) & \cdots & x(t_m) \\ \downarrow & \downarrow & \downarrow & & \downarrow \end{bmatrix}}_{m} \xrightarrow[\text{Decomposition}]{\text{Singular Value}} \begin{bmatrix} \uparrow & & \uparrow \\ x(t_1) & \cdots & x(t_q) \\ \downarrow & & \downarrow \end{bmatrix}$$

LECTURE 25.
Model-Order Reduction II



$$\mathbf{V}_q = \underbrace{\begin{bmatrix} \uparrow & \uparrow & \uparrow & & \uparrow \\ V_1 & V_2 & V_3 & \cdots & V_m \\ \downarrow & \downarrow & \downarrow & & \downarrow \end{bmatrix}}_{m} \xrightarrow[\text{Decomposition}]{\text{Singular Value}} \underbrace{\begin{bmatrix} \uparrow & & \uparrow \\ V_1 & \cdots & V_q \\ \downarrow & & \downarrow \end{bmatrix}}_{q}$$

- Use Frequency Domain Data
  - Compute $x(s_1), x(s_2), \ldots, x(s_k)$
  - Use the SVD to pick $q < k$ important vectors
- Use Time Series Data
  - Compute $x(t_1), x(t_2), \ldots, x(t_k)$
  - Use the SVD to pick $q < k$ important vectors

Point Matching

SVD Singular Value Decomposition Method
other names:
- POD Proper Orthogonal Decomposition
- KLD Karhumen-Lo`eve Decomposition
- PCA Principal Component Analysis
- "Poor Man" TBR (Truncated Balance Relizations)

- Use Singular Vectors of System Grammians Product (Truncated Balance Realizations)
- Use Krylov Subspace Vectors (Moment Matching)

3.26 Model Order Reduction III

INTRODUCTION TO NUMERICAL SIMULATION

LECTURE 26.
Model-Order Reduction III

TODAY'S OUTLINE:

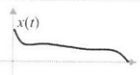
- Truncated Balance Realizations
- Krylov Subspace Moment Matching
  - Preserving passivity
  - Need for orthogonalization (Arnoldi process)
  - Overall computational complexity
- Comparing Truncated Balance Realizations and Krylov Subspace Moment Matching

TRUNCATED BALANCE REALIZATIONS

- Approaches for picking V and U
  - Use Eigenvectors of the system matrix (modal analysis)
  - Use Frequency Domain Data
    - Compute $x(s_1), x(s_2), \ldots, x(s_k)$
    - Use the SVD to pick $q < k$ important vectors
  - Use Time Series Data
    - Compute $x(t_1), x(t_2), \ldots, x(t_k)$
    - Use the SVD to pick $q < k$ important vectors
  - Use Singular Vectors of System Grammians Product (Truncated Balance Realizations)
  - Use Krylov Subspace Vectors (Moment Matching)
- Observability Gramian

$$u(t) = 0 \Rightarrow \begin{cases} \dfrac{dx}{dt} = \mathbf{A}x(t) & x(0) = x \\ y(t) = \mathbf{C}^T x(t) & y(t) = e^{\mathbf{A}t}\mathbf{C}x \end{cases}$$

$$x^T \underbrace{\left[\int \cdots\cdots\right]}_{N\times N} x$$



Energy of the output $y(t)$ starting from state x with no input:

$$\|y(t)\|_x^2 = \int_0^\infty y(t)^T y(t)dt = \int_0^\infty \left(\mathbf{C}e^{\mathbf{A}t}x\right)^T \mathbf{C}e^{\mathbf{A}t}x\,dt = x^T \underbrace{\left[\int_0^\infty e^{\mathbf{A}^T t}\mathbf{C}^T\mathbf{C}e^{\mathbf{A}t}dt\right]}_{\substack{W_O \\ \text{Observabilty Gramian}}} x$$

Note: it is also the solution of $\mathbf{A}^T W_O + W_O\mathbf{A} = -\mathbf{C}^T\mathbf{C}$

Note: If $x = x_i$ the ith eigenvector of W_O:

$$\|y(t)\|_{x_i}^2 = x_i^T W_O x_i = \lambda_{O,i}$$

Hence: eigenvectors of W_O corresponding to small eigenvalues do NOT produce much energy at the output (i.e. they are not very observable):

Idea: let's get rid of them!

$$W_O = \underbrace{\begin{bmatrix} \uparrow & & \uparrow \\ x_1 & \cdots & x_N \\ \downarrow & & \downarrow \end{bmatrix}}_{\text{eigenvectors}} \begin{bmatrix} \lambda_1 & & \\ & \ddots & \\ & & \lambda_N \end{bmatrix} \begin{bmatrix} \uparrow & & \uparrow \\ x_1 & \cdots & x_N \\ \downarrow & & \downarrow \end{bmatrix}^{-1}$$

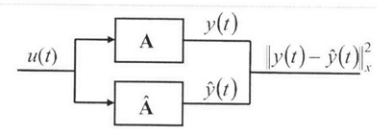
Note $\quad W_O^{-1} = W_O^T \Rightarrow \begin{bmatrix} \uparrow & & \uparrow \\ x_1 & \cdots & x_N \\ \downarrow & & \downarrow \end{bmatrix}^{-1} = \begin{bmatrix} \leftarrow & x_1 & \rightarrow \\ & \vdots & \\ \leftarrow & x_N & \rightarrow \end{bmatrix}$

Unitary Orthonormal Matrix

$$\|y(t)\|_{x_i}^2 = x_i^T W_O x_i$$

$$= \left[\leftarrow \; x_i^T \; \rightarrow\right] \underbrace{\begin{bmatrix} \uparrow & & \uparrow \\ x_1 & \cdots & x_N \\ \downarrow & & \downarrow \end{bmatrix}}_{\text{eigenvectors}} \begin{bmatrix} \lambda_{O,1} & & \\ & \ddots & \\ & & \lambda_{O,N} \end{bmatrix} \begin{bmatrix} \leftarrow & x_1 & \rightarrow \\ & \vdots & \\ \leftarrow & x_N & \rightarrow \end{bmatrix} \begin{bmatrix} \uparrow \\ x_i \\ \downarrow \end{bmatrix}$$

$$= \left[0 \;\cdots\; 0 \;\underset{i}{1}\; 0 \;\cdots\; 0\right] \begin{bmatrix} \lambda_{O,1} & & \\ & \ddots & \\ & & \lambda_{O,N} \end{bmatrix} \begin{bmatrix} 0 \\ \vdots \\ 0 \\ 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \begin{matrix} \\ \\ \\ \leftarrow i \\ \\ \\ \\ \end{matrix} = \lambda_{O,i}$$



- Controllability Gramian

  Minimum amount input energy required to drive the system to a specific state x:

$$\min_{u(t)} \int_{-\infty}^{0} u(t)^T u(t)dt = x^T \underbrace{\left[\int_0^{\infty} e^{\mathbf{A}t} \mathbf{B}\mathbf{B}^T e^{\mathbf{A}^T t} dt \right]}_{\substack{W_C^{-1} \\ \text{Inverse of} \\ \text{Controllability Gramian}}} x$$

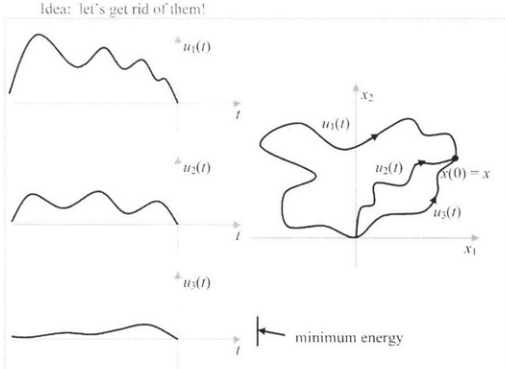
It is also the solution of $\mathbf{A}W_C + W_C\mathbf{A}^T = -\mathbf{B}\mathbf{B}^T$

Note: If $x = x_i$ the ith eigenvector of W_C:

$$\min_{u(t)} \|u(t)\|_{L_2}^2 = x_i^T W_C^{-1} x_i = \frac{1}{\lambda_{C,i}}$$

Hence: eigenvectors of W_C corresponding to small eigenvalues do NOT produce much energy at the output (i.e. they are not very controllable):

Idea: let's get rid of them!



eigenvectors

$$x_i^T W_C^{-1} x_i = x_i^T \left[\mathbf{X}_C \Lambda_C \mathbf{X}_C^T \right]^{-1} x_i$$

$$= x_i^T \mathbf{X}_C \Lambda_C^{-1} \mathbf{X}_C^T x_i$$

$$= x_i^T \begin{bmatrix} \uparrow & & \uparrow \\ x_1 & \cdots & x_N \\ \downarrow & & \downarrow \end{bmatrix} \begin{bmatrix} \frac{1}{\lambda_{c,1}} & & \\ & \ddots & \\ & & \frac{1}{\lambda_{c,N}} \end{bmatrix} \begin{bmatrix} \leftarrow & x_1 & \rightarrow \\ & \vdots & \\ \leftarrow & x_N & \rightarrow \end{bmatrix} x_i$$

$$= \begin{bmatrix} 0 & \cdots & 0 & \underset{i}{1} & 0 & \cdots & 0 \end{bmatrix} \begin{bmatrix} \frac{1}{\lambda_{c,1}} & & \\ & \ddots & \\ & & \frac{1}{\lambda_{c,N}} \end{bmatrix} \begin{bmatrix} 0 \\ \vdots \\ 0 \\ 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \leftarrow i = \frac{1}{\lambda_{c,i}}$$

- Native Controllability/Observability MOR
  - Suppose I could compute a basis for the strongly observable and/or strongly controllable spaces. Projection-based MOR can give a reduced model that deletes weakly observable and/or weakly controllable modes.
  - Problem:
    - What if the same mode is strongly controllable, but weakly observable?
    - Are the eignvalues of the respective Gramians even unique?
- Changing Coordinate System
  - Consider an invertible change of coordinates: $x(t) = U\tilde{x}(t)$
  - We know that the input/output relationship will be unchanged.
  - But what about the Gramians, and their eigenvalues?

    $\tilde{W}_O \leftarrow U^T W_O U \quad \tilde{W}_C \leftarrow U^{-1} W_C U^{-T}$
  - Gramians and their eigenvalues change! Hence the relative degrees of observability and controllability are properties of the coordinate system
  - A bad choice of coordinates will lead to bad reduced models if we look at controllability and observability separately.
  - What coordinate system should we use then?

$$W_O = \int_0^\infty e^{\mathbf{A}^T t}\mathbf{C}^T\mathbf{C}e^{\mathbf{A}t}dt \qquad x = \mathbf{U}\tilde{x}$$

$$\begin{cases} \dfrac{d\tilde{x}}{dt} = \mathbf{U}^{-1}\mathbf{A}\mathbf{U}\tilde{x}(t) + \mathbf{U}^{-1}bu(t) \\ y(t) = \mathbf{C}\mathbf{U}\tilde{x}(t) \end{cases}$$

$$\begin{aligned} \tilde{W}_O &= \int_0^\infty e^{(\mathbf{U}^{-1}\mathbf{A}\mathbf{U})^T t}(\mathbf{C}\mathbf{U})^T(\mathbf{C}\mathbf{U})e^{\mathbf{U}^{-1}\mathbf{A}\mathbf{U}t}dt \\ &= \int_0^\infty \mathbf{U}^T e^{\mathbf{A}^T t}\mathbf{U}^{-T}\mathbf{U}^T\mathbf{C}^T\mathbf{C}\mathbf{U}\mathbf{U}^{-1}e^{\mathbf{A}t}\mathbf{U}dt \\ &= \mathbf{U}^T W_O \mathbf{U} \end{aligned}$$

$$W_O = \mathbf{X}\Lambda\mathbf{X}^T \qquad \mathbf{U} \text{ invertible}$$

$$\tilde{W}_O = \underbrace{\mathbf{U}^T\mathbf{X}}\Lambda\mathbf{X}^T\mathbf{U}$$

these are <u>not</u> the new eigenvectors in general

$$\mathbf{X}^T\mathbf{U} \neq (\mathbf{U}^T\mathbf{X})^{-1}$$

$$\mathbf{U}^T\mathbf{X}\mathbf{X}^T\mathbf{U} = \mathbf{U}^T\mathbf{U} \neq I \qquad = I \text{ only if } \mathbf{U} \text{ is orthonormal}$$

- Balancing

  Fortunately the eigenvalues of the **product of the Gramians** (Hankel singular values) do not change when changing coordinates:

  Diagonal matrix with eigenvalues of the product

  The eigenvectors change

  But not the eigenvalues

$$W_C W_O = S\overbrace{\Sigma^2}S^{-1} \qquad x(t) = U\tilde{x}(t)$$

$$\underbrace{U^{-1}W_C U^{-T}}\ \underbrace{U^T W_O U} = U^{-1}W_C W_O U = \underbrace{(U^{-1}S)}\Sigma^2(U^{-1}S)^{-1}$$

And since W_C and W_O are symmetric a change of coordinate matrix U can be found that diagonalize both

$\Sigma\Sigma = \Sigma^2$ In Balanced Coordinates the Gramians are equal and diagonal

- Selection of Vectors for the Columns of the Reduced Order Projection Matrix.

  In balanced coordinates it is easy to select the best vectors for the reduced model: we want the subspace of vectors that are at the same time most controllable and observable:

$$U^{-1}W_C U^{-T}\ U^T W_O U = \Sigma^2$$

Simply pick the eigenvecors corresponding to the largest entries on the diagonal (Hankel singular values)

  In other words the ones corresponding to the largest eigenvalues of the controllability and observability Gramians product.

- Truncated Balance Realization Summary
  - The good news: $\|H(j\omega) - H_q(j\omega)\|_\infty \leq \cancel{\times}\ (\Sigma_{q+1,q+1} + \ldots + \Sigma_{N,N})$

  - We even have bounds for the error
  - Can do even a bit better with the optimal Hankel Reduction
  - The bad news: $A^T W_0 + W_o A = -C^T C$
    - It is expensive:
      - Need to compute the Gramians (solve Lyapunov equation)
      - Need to compute eigenvalues of the product: $O(N^3)$
  - The bottom line:
    - If the size of your system allows $O(N^3)$ computation, Truncation Balance Realization or Hankel Reduction are a much better choice than the any other reduction method.
    - But if you cannot afford $O(N^3)$ computation (e.g. dense matrix with $N > 5000$) then PRIMA or PVL or Quasi-Convex-Optimization are better choices.

- Approaches for picking V and U
  - Use Eigenvectors of the system matrix (modal analysis)
  - Use Frequency Domain Data
    - Compute $x(s_1), x(s_2), \ldots, x(s_k)$
    - Use the SVD to pick $q < k$ important vectors
  - Use Time Series Data
    - Compute $x(t_1), x(t_2), \ldots, x(t_k)$
    - Use the SVD to pick $q < k$ important vectors
  - Use Singular Vectors of System Grammians Product (Truncated Balance Realizations)
  - Use Krylov Subspace Vectors (Moment Matching)

Original System Transfer Function Moments

$$H(s) = c^T(sI - \mathbf{A})^{-1}b$$

$$= -c^T\underbrace{(I - s\mathbf{A}^{-1})^{-1}}_{\text{Taylor expand with respect to } s}\mathbf{A}^{-1}b \qquad \frac{1}{1-x} = 1 + x + x^2 + x^3 + \ldots$$

$$\left(I - \underbrace{s\mathbf{A}^{-1}}_{\text{small}}\right)^{-1} = I + s\mathbf{A}^{-1} + s^2\mathbf{A}^{-2} + s^3\mathbf{A}^{-3} + \ldots$$

$$= \sum_{k=0}^{\infty} c^T A^{-(k+1)} b s^k$$

moments

$$H_r(s) = \frac{b_0^r + b_1^r s + \ldots + b_{q-1}^r s^{q-1}}{1 + a_1^r s + \ldots + a_q^r s^q} = m_0 + m_1 s + \ldots + m_{2q-1} s + \ldots$$

$$c^T \underline{\mathbf{A}^{-1} b} \quad c^T \underline{\mathbf{A}^{-2} b} \quad c^T \underline{\mathbf{A}^{-3} b} \quad \cdots \quad c^T \underline{\mathbf{A}^{-k} b}$$

eigenvectors

$$\mathbf{A}^{-1} b = \mathbf{A}^{-1}\left[\alpha_1 \vec{e}_1 + \cdots + \alpha_N \vec{e}_N\right]$$
$$= \alpha_1 \lambda_1^{-1} \vec{e}_1 + \cdots + \alpha_N \lambda_N^{-1} \vec{e}_N$$

$$\mathbf{A}^{-k} b = \alpha_1 \lambda_1^{-k} \vec{e}_1 + \cdots + \underbrace{\alpha_N \lambda_N^{-k} \vec{e}_N}_{\text{dominates}}$$

Cross-Multiplying and Matching Terms

$$\begin{bmatrix} m_0 & m_1 & \cdots & m_{k-1} \\ m_1 & \ddots & & \vdots \\ \vdots & & \ddots & m_{2q-3} \\ m_{k-1} & \cdots & m_{2q-3} & m_{2q-2} \end{bmatrix} \begin{bmatrix} a_q \\ a_{q-1} \\ \vdots \\ a_1 \end{bmatrix} = \begin{bmatrix} m_q \\ m_{q+1} \\ \vdots \\ m_{2q-1} \end{bmatrix}$$

- A Canonical Form for Model Order Reduction

$$sx = Ax + \bar{b}\, u$$
$$y = c^T x$$

$$E = A^{-1}$$
$$b = A^{-1}\bar{b}$$

Assuming A is non-singular we can cast the dynamical linear system into a canonical form for moment matching model order reduction

$$sEx = x + b\, u$$
$$y = c^T x$$

Note: this step is not necessary, it just makes the notation simple for educational purposes

- The Moment Matching Idea [Grimme PhD97]

$$sEx = x + bu \quad \Rightarrow \quad x = -\left(I - sE\right)^{-1} bu$$

Taylor series expansion:

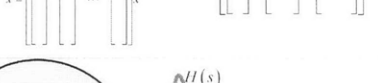
$$x = -\sum_{k=0}^{\infty} s^k \; E^k b \;\; u \quad \Rightarrow \quad x \in \operatorname{span}\left\{b, Eb, E^2 b, \cdots\right\}$$

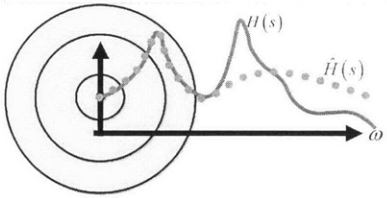
change basis: use only first q vectors of the Taylor series expansion matching first q derivatives around expansion point

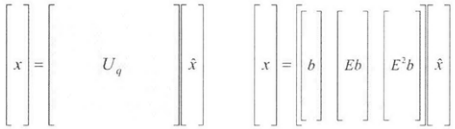
$$x = b + s\underbrace{\mathbf{E}b}_{\frac{\partial H}{\partial s}} + s^2 \underbrace{\mathbf{E}^2 b}_{\frac{1}{2}\frac{\partial^2 H}{\partial s^2}} + s^3 \underbrace{\mathbf{E}^3 b}_{\frac{1}{3!}\frac{\partial^3 H}{\partial s^3}} + \ldots$$

Moment

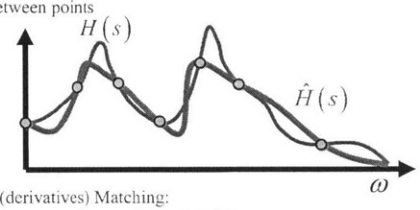
$$\underset{N}{x} \in \underbrace{\left\{b, \mathbf{E}b, \mathbf{E}^3 b, \ldots\right\}}_{N}$$



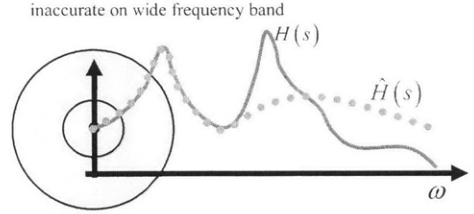




Point Matching:
can be very inaccurate
in between points

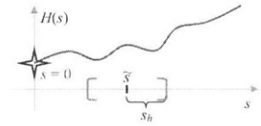


Moment (derivatives) Matching:
accurate around expansion point, but
inaccurate on wide frequency band



- Krylov Subspaces – Definition
  The order k Krylov subspace generated from matrix E and vector b is defined as
  $\kappa_k(E,b) \equiv span\{b, Eb, E^2b, \ldots, E^{k-1}b\}$
- Moment Matching around Non-zero Frequencies
  - Instead of expanding around only $s = 0$ we can expand around other points
    $s = 0 \quad s = s_1 \quad s = s_2 \quad \cdots \quad s = s_j$

$$sx = Ax + bu \qquad \Rightarrow \qquad (\tilde{s} + s_h)x = Ax + bu$$
$$y = c^T x \qquad s = \tilde{s} + s_h \qquad y = c^T x$$



$$\tilde{s}x = (\mathbf{A}x - s_h x) + bu$$
$$\tilde{s}\underbrace{(\mathbf{A} - s_h I)^{-1}}_{\mathbf{E}_h} x = x + \underbrace{(\mathbf{A} - s_h I)^{-1} bu}_{b_h}$$

$$x = \left[\left[b\right] \left[\mathbf{E}b\right] \left[\mathbf{E}^2 b\right] \left[b_h\right] \left[\mathbf{E}_h b_h\right] \left[b_k\right] \left[\mathbf{E}_k b_k\right] \left[\mathbf{E}_k^2 b_k\right]\right] \hat{x}$$

- For each expansion point the problem can then be put again in the canonical form

$$\boxed{\begin{aligned} E_h &= (A - s_h I)^{-1} \\ b_h &= (A - s_h I)^{-1} b \end{aligned}}$$

$$\tilde{s} E_h \; x = x + b_h \; u$$
$$y = c^T x$$

- Projection Framework: Moment Matching Theorem [E. Grimme 97]

If $\quad Range(U_q) \supseteq \bigcup_{h=1}^{J} \kappa_{k_h^h}(E_h, b_h)$

and $\quad Range(V_q) \supseteq \bigcup_{h=1}^{J} \kappa_{k_h^h}(E_h^T, c_h)$

Then $\left.\dfrac{\partial^l H}{\partial s^l}\right|_{s_h} = \left.\dfrac{\partial^l \hat{H}}{\partial s^l}\right|_{s_h}$ for $\begin{array}{l} l = 0, \ldots, k_h^h + k_h^h - 1 \\ h = 1, \ldots, J \end{array}$

Total of $2q$ moment of the transfer function will match

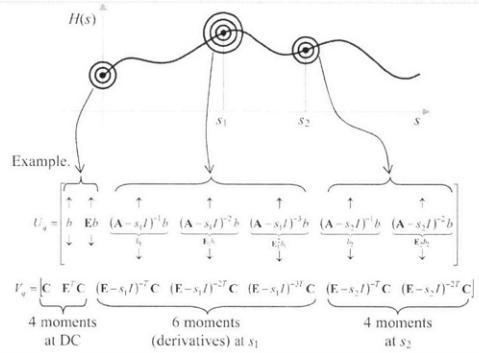
If $span\{\vec{u}_1,\ldots,\vec{u}_q\} \supseteq \bigcup_{j=1}^{J} \kappa_{k_j^b}\left((\mathbf{A}-s_j I)^{-1},(\mathbf{A}-s_j I)^{-1} b\right)$

and $span\{\vec{v}_1,\ldots,\vec{v}_q\} \supseteq \bigcup_{j=1}^{J} \kappa_{k_j^c}\left((\mathbf{A}-s_j I)^{-T},(\mathbf{A}-s_j I)^{-T} c\right)$

Then $\dfrac{\partial^l H(s_j)}{\partial s^l} = \dfrac{\partial^l H_r(s_j)}{\partial s^l}$ for $l = 0,\ldots,k_j^b + k_j^c - 1$

$$span\{\vec{v}_1,\ldots,\vec{v}_q\} = span\{\vec{u}_1,\ldots,\vec{u}_q\} \supseteq \mathbf{A}^{-1}b, \mathbf{A}^{-2}b, \mathbf{A}^{-3}b,\ldots$$

$$H_r(s) = \underbrace{m_0 + m_1 s + m_2 s^2 + \ldots}_{\text{original system moments}}$$



If $\mathbf{U}$ and $\mathbf{V}$ are such that

$$\mathbf{U} = \mathbf{V} = \{\vec{u}_1,\ldots,\vec{u}_q\} \text{ and } \mathbf{U}^T\mathbf{U} = I$$

$$span\{\vec{u}_1,\ldots,\vec{u}_q\} = span\{\mathbf{A}^{-1}b, \mathbf{A}^{-2}b,\ldots,\mathbf{A}^{-q}b\}$$

Then the first q moments of reduced system match

$$H(s) = -c^T\left(I - s\mathbf{A}^{-1}\right)^{-1}\mathbf{A}^{-1}b = \sum_{k=0}^{\infty} c^T\mathbf{A}^{-(k+1)}bs^k$$

$$H_r(s) = -c_r^T\left(I - s\mathbf{A}_r^{-1}\right)^{-1}\mathbf{A}_r^{-1}b_r = \sum_{k=0}^{\infty} c_r^T\mathbf{A}_r^{-(k+1)}b_r s^k$$

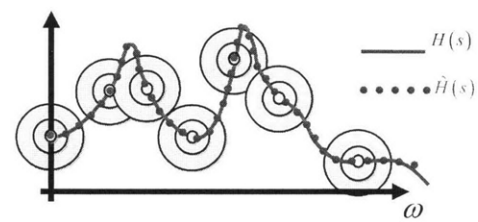
$$c^T\mathbf{A}^{-(k+1)}b = c^T\mathbf{U}_q\left(\mathbf{U}_q^T\mathbf{A}\mathbf{U}_q\right)^{-(k+1)}\mathbf{U}_q^T b \qquad k = \{0,\ldots,q-1\}$$

$$\mathbf{A}_r = \mathbf{U}_q^T\mathbf{A}\mathbf{U}_q$$

$$b_r = \mathbf{U}_q^T b$$

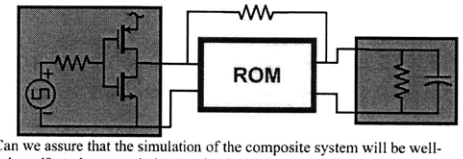
$$c_r = \mathbf{U}_q^T c$$

- Combine Point and Moment Matching: Multipoint Moment Matching
  - Multipole expansion points give larger band
  - Moment (derivates) matching gives more accurate behavior in between expansion points



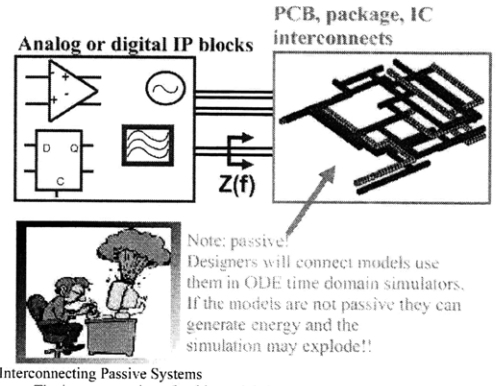
KRYLOV SUBSPACE MOMENT MATCHING

- Preserving Passivity
  - Interconnected Systems
    - In reality, reduced models are only useful when connected together with models of other components in a composite simulation
    - Consider a state-space model connected to external circuitry (possibly with feedback)

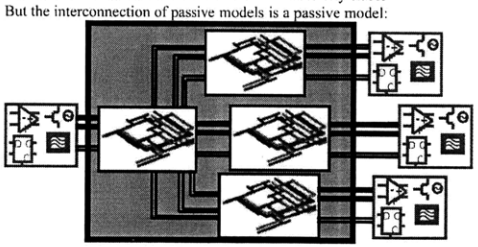


- Can we assure that the simulation of the composite system will be well-behaved? At least preclude non-physical behavior of the reduced model?

- Need to Preserve Passivity for Models of Passive Interconnect



- Interconnecting Passive Systems
  - The interconnection of stable models is not necessarily stable
  - But the interconnection of passive models is a passive model:



- Sufficient Conditions for Passivity

  $sEx = x + Bu$

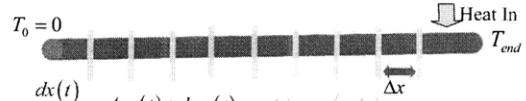
  $y = Cx$

  Sufficient conditions for passivity
  1. $C = B^T$
  2. $x^T E x \le 0$, for all x

     i.e. E is negative semidefinite

  Note that these are NOT necessary conditions (common misconceptions)

- Example Finite Difference on Poisson Equation

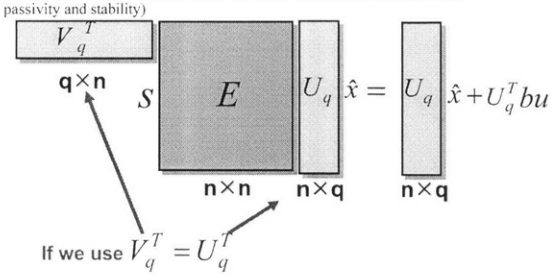


$$\frac{dx(t)}{dt} = \underbrace{A}_{N\times N} x(t) + \underbrace{b}_{N\times 1} \underbrace{u(t)}_{\text{scalar input}} \qquad \underbrace{y(t)}_{\text{scalar output}} = \underbrace{c^T}_{1\times N} x(t)$$

$$A = -\begin{bmatrix} 2 & -1 & & & \\ -1 & 2 & -1 & & \\ & -1 & 2 & \ddots & \\ & & \ddots & \ddots & -1 \\ & & & -1 & 2 \end{bmatrix} \qquad b = \begin{bmatrix} 0 \\ \vdots \\ \vdots \\ 0 \\ 1 \end{bmatrix} \qquad c = \begin{bmatrix} 0 \\ \vdots \\ \vdots \\ 0 \\ 1 \end{bmatrix}$$

We already know the Finite Difference matrices is positive semidefinite. Hence $E = -A^{-1}$ is negative semidefinite.

- Congruence Transformation Preserves Negative Definiteness of E (hence passivity and stability)



- Then we loose half of the degrees of freedom
  i.e. we match only q moments instead of 2q
- But if the original matrix E is negative semidefinite
  so is the reduced hence the system is passive and stable

$$x^T \mathbf{E} x \le 0 \qquad \forall x$$

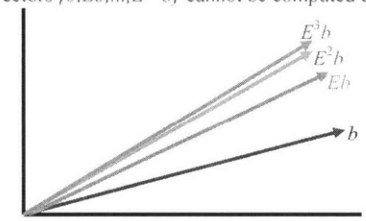
$$x^T \mathbf{U}_q^T \mathbf{E} \mathbf{U} x \overset{?}{\le} 0 \qquad \forall x$$

Pick x_0

$$\underbrace{x_0^T \mathbf{U}_q^T}_{x^T} \underbrace{\mathbf{E} \mathbf{U} x_0}_{x} \le 0 \qquad \text{use} \quad x^T \mathbf{E} x \le 0$$

- Need for Orthogonalization (Arnoldi process)
  - Need for Orthonormalization of U

Vectors $\{b, Eb, \ldots, E^{k-1}b\}$ cannot be computed directly



Vectors will quickly line up with dominant eigenspace!

$$\mathbf{E} = \mathbf{V}\Lambda\mathbf{U}^T \qquad \mathbf{U} = \{b, \mathbf{E}b, \mathbf{E}^2 b, \mathbf{E}^3 b, \ldots\}$$

$$b = \alpha_1 v_1 + \alpha_2 v_2 + \ldots + \alpha_N v_N \qquad \lambda_1 \ge \lambda_2 \ge \ldots \ge \lambda_N$$

$$\mathbf{E}b = \alpha_1 \mathbf{E} v_1 + \alpha_2 \mathbf{E} v_2 + \ldots + \alpha_N \mathbf{E} v_N$$

$$= \alpha_1 \lambda_1 v_1 + \alpha_2 \lambda_2 v_2 + \ldots + \alpha_N \lambda_N v_N$$

$$\mathbf{E}^2 b = \alpha_1 \lambda_1^2 v_1 + \alpha_2 \lambda_2^2 v_2 + \ldots + \alpha_N \lambda_N^2 v_N$$

$$\mathbf{E}^{k-1} b = \alpha_1 \lambda_1^{k-1} v_1 + \alpha_2 \lambda_2^{k-1} v_2 + \ldots + \alpha_N \lambda_N^{k-1} v_N \cong \alpha_1 \lambda_1^{k-1} v_1$$

very large

They all become linearly dependent and parallel to the eigenvector with the largest eigenvalue

- Overall Computational Complexity
  - Orthonormalization of U: The Arnoldi Algorithm Computational Complexity

$$\vec{u}_1 = b / \|b\|$$ Normalize first vector O(n)

For $i = 1$ to k

$$\vec{u}_{i+1} = E\vec{u}_i$$ Generates new Krylov subspace vector O(????????)

For $j = 1$ to i

$$\vec{u}_{i+1} \leftarrow \vec{u}_{i+1} - \underbrace{\left(\vec{u}_{i+1}^T \vec{u}_j\right)}_{\Gamma_{ji}} \vec{u}_j$$ Orthogonalize new vector $O(k^2 n)$

$$\vec{u}_{i+1} \leftarrow \underbrace{\frac{1}{\|\vec{u}_{i+1}\|}}_{\Gamma_{i+1,i}} \vec{u}_{i+1}$$ Normalize new vector O(n)

  - Generating Vectors for the Krylov Subspace
    - Most of the computation cost is spent in calculating:

$$\vec{u}_{i+1} = E_h \vec{u}_h = (A - s_h I)^{-1} \vec{u}_i$$

$$\vec{u}_{i+1} = (\mathbf{A} - s_h I)^{-1} \vec{u}_i \quad \Rightarrow (\mathbf{A} - s_h I)\vec{u}_{i+1} = \vec{u}_i \qquad O(n)$$

    - Set up and solve a linear system using GCR

$$(A - s_h I)^{-1} \vec{u}_{i+1} = \vec{u}_i$$

    - If we have a good preconditioner and a fast matrix vector product each new vector is calculated $O(n)$
    - The total complexity for calculating the projection matrix U_q is $O(qn)$

Comparing Truncated Balance Realizations and Krylov Subspace Moment Matching

$$\mathbf{U}_q^T \mathbf{E} \mathbf{U}_q \frac{d\hat{x}}{dt} = \hat{x}(t) + \mathbf{U}^T b \hat{x}(t) \quad \text{Reduced Model}$$

$$\hat{y}(t) = \mathbf{C}^T \mathbf{U}_q \hat{x}(t)$$

- What about computing the reduced matrix $\hat{\mathbf{E}} = \mathbf{U}_q^T \mathbf{E} \mathbf{U}_q$?

$$E\begin{bmatrix} \\ \vec{u}_i \\ \\ \end{bmatrix} = \begin{bmatrix} \\ \vec{u}_1 & \cdots & \vec{u}_q \\ \\ \end{bmatrix} \begin{bmatrix} \Gamma_{1,i} \\ \vdots \\ \Gamma_{q,i} \end{bmatrix}$$

Orthonormalization of the ith column of U_q

$$E\begin{bmatrix} \\ \vec{u}_1 & \ldots & \vec{u}_q \\ \\ \end{bmatrix} = \begin{bmatrix} \\ \vec{u}_1 & \ldots & \vec{u}_q \\ \\ \end{bmatrix} \begin{bmatrix} \\ \Gamma \\ \\ \end{bmatrix}$$

Orthonormalization of all columns of U_q

$$EU_q = U_q \Gamma$$

$$U_q^T E U_q = \Gamma$$

So we don't need to compute the reduced matrix. We have it already: $\hat{E} = \Gamma$

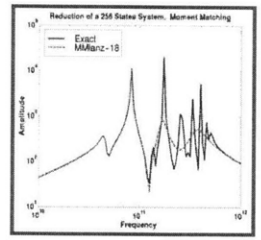
- Two Complementary Approaches

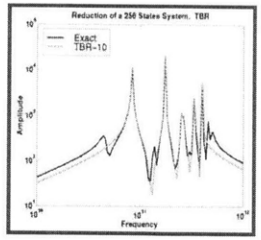
Moment Matching Approaches

- Accurate over a narrow band.
  - Matching function values and derivatives.
- Cheap: $O(qn)$
- Use it as a FIRST STAGE REDUCTION

Truncated Balanced Realization and Hankel Reduction

- Optimal (best accuracy for given size q, and apriori error bound.
- Expensive: $O(n^3)$
- USE IT AS A SECOND STAGE REDUCTION





- Model Order Reduction Computational Complexity (time and memory)

$$\frac{dx}{dt} = \mathbf{A}\,x(t) + B\,u(t)$$

A is DENSE !

MOR technique	Computational Complexity
TBR ('81) Hankel ('84) (have error bounds)	Bottleneck: SVD $O(N^3)$ e.g. 10months, 80GB, for N=100,000
Moment matching ('97) (no error bounds)	Bottleneck: matrix-vector product $O(qN^2)$ e.g. 7days, 80GB, for N=100,000 q=10
Moment Matching + pFFT matrix-vector	$O(qN \log N)$ e.g. 8hours, 0.3GB for N=100,000 q=10
QuasiConvex Optimization + pFFT field solver	$O(mN \log N)$ Same as above, but have error bound!

- And Remember, No Matter What…

Never Invert a Matrix!