REVIEW Lecture 7:

• Direct Methods for solving Linear Equation Systems
  – Determinants and Cramer’s Rule (and other methods for a small number of equations)
  – Gauss Elimination
    • Algorithm
      – Forward Elimination/Reduction to Upper Triangular System
      – Back-Substitution
      – Number of Operations: $O(n^3)$
    • Numerical implementation and stability
      – Partial Pivoting
      – Equilibration
      – Full Pivoting
Gauss Elimination: Review

**Reduction/Elimination: Step k**

\[
\begin{align*}
    m_{ik} &= \frac{a_{ik}^{(k)}}{a_{kk}^{(k)}} \\
    a_{ij}^{(k+1)} &= a_{ij}^{(k)} - m_{ik}a_{kj}^{(k)}, \quad j = k, \ldots, n \quad i = k+1, \ldots, n \\
    b_{i}^{(k+1)} &= b_{i}^{(k)} - m_{ik}b_{k}^{(k)}
\end{align*}
\]

**Back-Substitution**

\[x_n = \frac{b_n^{(n)}}{a_{nn}^{(n)}}\]
\[x_{n-1} = \left(b_{n-1}^{(n-1)} - a_{n-1,n}^{(n-1)}x_n\right)/a_{n-1,n-1}^{(n-1)}\]
\[\vdots\]
\[x_k = \left(b_k^{(k)} - \sum_{j=k+1}^{n} a_{kj}^{(k)}x_j\right)/a_{kk}^{(k)}\]
\[x_1 = \left(b_1^{(1)} - \sum_{j=2}^{n} a_{1j}^{(1)}x_j\right)/a_{11}^{(1)}\]

Result after step (n-1) is an Upper triangular system!
### Gauss Elimination: Pivoting Review

**Reduction Step k**

\[
\begin{align*}
m_{ik} &= \frac{a_{ik}^{(k)}}{a_{kk}^{(k)}} \\
a_{ij}^{(k+1)} &= a_{ij} - m_{ik} a_{kj}^{(k)}, \quad j = k, \ldots, n \\
b_i^{(k+1)} &= b_i^{(k)} - m_{ik} b_k^{(k)}
\end{align*}
\]

\[i = 2, \ldots, n\]

**Partial Pivoting by Columns**

Two Solutions:

- **A. Partial Pivoting**
  - i. Search for largest available coefficient in column below pivot element
  - ii. Switch rows \( k \) and \( i \)

- **B. Complete Pivoting**
  - i. As for Partial, but search both rows and columns
  - ii. Rarely done since column re-ordering changes order of \( x \)'s

**Pivot Elements**

\[a_{11}^{(1)}, a_{22}^{(2)}, \ldots, a_{nn}^{(n)}\]

**Required at each step!**

\[a_{kk}^{(k)} \neq 0\]
Gauss Elimination: Review

Numerical Stability

- **Partial Pivoting**
  - Equilibrate system of equations (Normalize or scale variables)
  - Pivoting by Columns
  - Simple book-keeping
    - Solution vector in original order

- **Full Pivoting**
  - Does not necessarily require equilibration
  - Pivoting by both row and columns
  - More complex book-keeping
    - Solution vector re-ordered

Partial Pivoting is simplest and most common
Neither method guarantees stability due to large number of recursive computations (round-off error)
TODAY’s Lecture: Systems of Linear Equations II

• **Direct Methods**
  – Gauss Elimination
    • Algorithm
      – Forward Elimination/Reduction to Upper Triangular System
      – Back-Substitution
      – Number of Operations: $O(n^3)$
    • Numerical implementation and stability
      – Well suited for dense matrices,
      – **Issues**: round-off, cost, does not vectorize/parallelize well
    • Special cases, Multiple right hand sides, Operation count
      – LU decomposition/factorization
      – Error Analysis for Linear Systems
        – Condition Number
      – Special Matrices: Tri-diagonal systems

• **Iterative Methods**
  – Jacobi’s method, Gauss-Seidel iteration, etc
  – Convergence
Reading Assignment

• Chapter 10 of “Chapra and Canale, Numerical Methods for Engineers, 2006/2009.”
Special Applications of Gauss Elimination

• **Complex Systems**
  - Replace all numbers by complex ones, or,
  - Re-write system of $n$ complex equations into $2n$ real equations

• **Nonlinear Systems of equations**
  - Newton-Raphson: 1st order term kept, use 1st order derivatives
  - Secant Method: Replace 1st order derivatives with finite-difference
  - In both cases, at each iteration, this leads to a linear system, which can be solved by Gauss Elimination

• **Gauss-Jordan**: variation of Gauss Elimination
  - Elimination
    - Eliminates each unknown completely (both below and above the pivot row) at each step
    - Normalizes all rows by their pivot
  - Elimination leads to diagonal unitary matrix (identity): no back-substitution needed
  - Number of Ops: about 50% more expensive than Gauss Elimination ($n^{3/2}$ vs. $n^{3/3}$ multiplications/divisions)
Gauss Elimination: Multiple Right-hand Sides

\[ A X = B \]

Reduction Step \( k \)

\[ \begin{bmatrix} \times \\ 0 & \times \\ . & . & \times \\ . & 0 & \times & \times & \times \\ . & . & \times \\ . & . & \times \\ . & . & \times \\ 0 & 0 & \times \end{bmatrix} \]

\[ X = \begin{bmatrix} \times & \times & \times \\ \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot \\ \times & \times & \times \end{bmatrix} \]

Total Computation Count = ?

Reduction: \( N_r \)

Back Substitution: \( N_b \)

If \( n >> p \), we expect \( N_r >> N_b \)

But, if \( n \sim p \)? (next slide)

\( X \) is a \([n \times p]\) matrix
Gauss Elimination: Multiple Right-hand Sides

Number of Ops

Reduction/Elimination: Step k

\[
\begin{align*}
m_{ik} &= \frac{a_{ik}^{(k)}}{a_{kk}^{(k)}} \\
a_{ij}^{(k+1)} &= a_{ij}^{(k)} - m_{ik} a_{kj}^{(k)}, \quad j = k, \ldots, n \\
b_i^{(k+1)} &= b_i^{(k)} - m_{ik} b_k^{(k)}
\end{align*}
\]

- n-k divisions
- 2 \((n-k) (n-k+1)\) additions/multiplications
- 2 \((n-k) p\) additions

For reduction, the number of ops is:

\[
\sum_{k=1}^{n-1} \left(2p+1\right)(n-k) + 2(n-k) \times (n-k+1) = 2 \left(\frac{n^3}{3} + n^2 - \frac{n}{3}\right) - n(n-1) + pn(n-1) = O(n^3 + pn^2)
\]

Back-Substitution

\[
x_k = \left(b_k^{(k)} - \sum_{j=k+1}^{n} a_{kj}^{(k)} x_j^{(k)}\right) / a_{kk}^{(k)}
\]

- \(p \times \left( (n-k-1)+(n-k)+2 \right) = p \times \left( 2(n-k) + 1 \right) \) add./mul./div.

Number of ops for back-substitution:

\[
p + p \sum_{k=1}^{n-1} 2(n-k) + 1 = p + p(n-1)(n+1) = pn^2
\]

Grand total number of ops is \(O(n^3 + pn^2)\) can be inefficient for large \(p\)

2.29 Numerical Fluid Mechanics PFJL Lecture 8
Gauss Elimination: Multiple Right-hand Sides

Number of Ops, Cont’d

Reduction at end of step $k$

\[ \begin{bmatrix}
  \times & \times & \times \\
  0 & \times & \times \\
  \cdot & \cdot & \times \\
  \cdot & 0 & \times \\
  \cdot & \cdot & \times \\
  \cdot & \cdot & \times \\
  0 & 0 & \times
\end{bmatrix} \]

\[ \bar{x} = \begin{bmatrix}
  \times & \times & \times \\
  \cdot & \cdot & \cdot \\
  \cdot & \cdot & \cdot \\
  \cdot & \cdot & \cdot \\
  \cdot & \cdot & \cdot \\
  \cdot & \cdot & \cdot \\
\end{bmatrix} \]

1. Reduction for each right-hand side can be inefficient if $p \gg\gg$

2. In addition, one may need to redo the Reduction each time if RHS is result of iterations and unknown a priori

\[ A \mathbf{x}_1 = \mathbf{b}_1, \quad A \mathbf{x}_2 = \mathbf{b}_2, \text{ etc, where vector } \mathbf{b}_2 \text{ is a function of } \mathbf{x}_1, \text{ etc} \]

\[
\Rightarrow \text{ LU Factorization / Decomposition}
\]
LU Decomposition/Factorization:

LU Decomposition: Separates time-consuming elimination for $A$ from that for $b / B$

The coefficient Matrix $\overline{A}$ is decomposed as

$$\overline{A} = \overline{L} \cdot \overline{U}$$

where $\overline{L}$ is a lower triangular matrix and $\overline{U}$ is an upper triangular matrix

Then the solution is performed in two simple steps

1. $\overline{L} \overline{y} = \overline{b}$  
   Forward substitution

2. $\overline{U} \overline{x} = \overline{y}$  
   Back substitution

How to determine $\overline{L}$, $\overline{U}$, and...
LU Decomposition / Factorization via Gauss Elimination, assuming no pivoting needed

Gauss Elimination (GE): iteration eqns. for the reduction at step \( k \) are

\[
A_i^{(k+1)} = A_i^{(k)} - m_{ik} a_{kj}, \quad m_{ik} = a_{ik}^{(k)} / a_{kk}^{(k)}
\]

This gives the final changes occurring in reduction steps \( k = 1 \) to \( k = i-1 \)

After reduction step \( i-1 \):

- Above and on diagonal: \( i \leq j \)
- Unchanged after step \( i-1 \): \( a_{ij}^{(n)} = \cdots = a_{ij}^{(i)} \)
- Below diagonal: \( j < i \)
- Become and remain 0 in step \( j \): \( a_{ij}^{(n)} = \cdots = a_{ij}^{(j+1)} = 0 \)
LU Decomposition / Factorization
via Gauss Elimination, assuming no pivoting needed

After reduction step \( i-1 \):

- Above and on diagonal: \( i \leq j \)
- Unchanged after step \( i-1 \): \( a_{ij}^{(n)} = \cdots a_{ij}^{(i)} \)
- Below diagonal: \( j < i \)
- Become and remain 0 in step \( j \): \( a_{ij}^{(n)} = \cdots a_{ij}^{(j+1)} = 0 \)

Gauss Elimination (GE): iteration eqns. for the reduction at step \( k \) are

\[
A_{ij}^{(k+1)} = A_{ij}^{(k)} - m_{ik} A_{kj}^{(k)}, \quad m_{ik} = A_{ik}^{(k)}/A_{kk}^{(k)}
\]

This gives the final changes occurring in reduction steps \( k = 1 \) to \( k = i-1 \)

Now, to evaluate the changes that accumulated from when one starts the elimination, let’s try to sum this iteration equation, from:

- 1 to \( i-1 \) for above and on diagonal
- 1 to \( j \) for below diagonal

As done in class, you can also sum up to an arbitrary \( r \) and see which terms remain.
LU Decomposition / Factorization
via Gauss Elimination, assuming no pivoting needed

Gauss Elimination (GE): iteration eqns. for the reduction at step $k$ are

$$a_{ij}^{(k+1)} = a_{ij}^{(k)} - m_{ik}a_{kj}^{(k)}, \quad m_{ik} = a_{ik}^{(k)}/a_{kk}^{(k)}$$

This gives the final changes occurring in reduction steps $k = 1$ to $k = i-1$

$$\sum \text{ these step-}k \text{ eqns. from } (k=1 \text{ to } i-1) \Rightarrow$$
Gives the total change above diagonal:

$$i \leq j : \quad a_{ij}^{(i)} = a_{ij} - \sum_{k=1}^{i-1} m_{ik}a_{kj}^{(k)}$$

$$\sum \text{ this step-}k \text{ eqns. from } (k=1 \text{ to } j) \Rightarrow$$
Gives the total change below diagonal:

$$i > j : \quad a_{ij}^{(i)} = 0 = a_{ij} - \sum_{k=1}^{j} m_{ik}a_{kj}^{(k)}$$

After reduction step $i-1$:

Above and on diagonal: $i \leq j$

Unchanged after step $i-1$: $a_{ij}^{(n)} = \cdots a_{ij}^{(i)}$

Below diagonal: $j < i$

Become and remain 0 in step $j$: $a_{ij}^{(n)} = \cdots a_{ij}^{(j+1)} = 0$
LU Decomposition / Factorization via Gauss Elimination, assuming no pivoting needed

After reduction step $i-1$:

Above and on diagonal: $i \leq j$

Unchanged after step $i-1$: $a_{ij}^{(n)} = \cdots a_{ij}^{(i)}$

Below diagonal: $j < i$

Become and remain $0$ in step $j$: $a_{ij}^{(n)} = \cdots a_{ij}^{(j+1)} = 0$

Change in reduction steps $k = 1$ to $k = i-1$:

$$a_{ij}^{(k+1)} = a_{ij}^{(k)} - m_{ik}a_{kj}^{(k)}, \quad m_{ik} = a_{ik}^{(k)}/a_{kk}^{(k)}$$

We obtained: Total change above diagonal

$$i \leq j : \quad a_{ij}^{(i)} = a_{ij} - \sum_{k=1}^{i-1} m_{ik}a_{kj}^{(k)} \quad (1)$$

We obtained: Total change below diagonal

$$i > j : \quad a_{ij}^{(i)} = 0 = a_{ij} - \sum_{k=1}^{j} m_{ik}a_{kj}^{(k)} \quad (2)$$

Now, if we define:

$$m_{ii} = 1, \quad i = 1, \ldots, n$$

and use them in equations (1) and (2) =>

$$i \leq j : \quad a_{ij} = \sum_{k=1}^{i} m_{ik}a_{kj}^{(k)}$$

$$i > j : \quad a_{ij} = \sum_{k=1}^{j} m_{ik}a_{kj}^{(k)}$$

$$\Rightarrow a_{ij} = \sum_{k=1}^{\min(i,j)} m_{ik}a_{kj}^{(k)}$$
LU Decomposition / Factorization via Gauss Elimination, assuming no pivoting needed

Result seems to be a 'Matrix product':

$$a_{ij} = \sum_{k=1}^{\min(i,j)} m_{ik} a_{kj}$$

Sum stops at diagonal

Below diagonal

$$i > j :$$

$$a_{ij} = \sum_{k=1}^{j} m_{ik} a_{kj}$$

Above diagonal

$$i \leq j :$$

$$a_{ij} = \sum_{k=1}^{i} m_{ik} a_{kj}$$

Lower triangular

Upper triangular
LU Decomposition / Factorization via Gauss Elimination, assuming no pivoting needed

GE Reduction directly yields LU factorization

\[
\bar{A} = \bar{L} \cdot \bar{U}
\]

Lower triangular

\[
\bar{L} = l_{ij} = \begin{cases} 
0 & i < j \\
1 & i = j \\
m_{ij} & i > j 
\end{cases}
\]

Upper triangular

\[
\bar{U} = u_{ij} = \begin{cases} 
\alpha_{ij} & i \leq j \\
0 & i > j 
\end{cases}
\]

Compact storage:
no need for additional memory (the unitary diagonal of L does not need to be stored)

Number of Operations for LU?

Lower diagonal implied

\[
m_{ii} = 1, \quad i = 1, \ldots, n
\]

(referred to as the Doolittle decomposition)
LU Decomposition / Factorization via Gauss Elimination, assuming no pivoting needed

GE Reduction directly yields LU factorization

\[
\bar{A} = \bar{L} \cdot \bar{U}
\]

- Lower triangular
  \[
  \bar{L} = l_{ij} = \begin{cases} 
  0 & i < j \\
  1 & i = j \\
  m_{ij} & i > j 
  \end{cases}
  \]

- Upper triangular
  \[
  \bar{U} = u_{ij} = \begin{cases} 
  a_{ij}^{(i)} & i \leq j \\
  0 & i > j 
  \end{cases}
  \]

Compact storage

Number of Operations for LU?

Same as Gauss Elimination: less in Elimination phase (no RHS operations), but more in double back-substitution phase

\[
m_{ii} = 1, \quad i = 1, \ldots, n
\]
Pivoting in LU Decomposition / Factorization

Before reduction, step $k$

\[
\begin{bmatrix}
    a_{11}^{(1)} & a_{12}^{(1)} & \cdots & \cdots & a_{1n}^{(1)} \\
    m_{21} & a_{22}^{(2)} & \cdots & \cdots & a_{2n}^{(2)} \\
    \vdots & \vdots & \ddots & \vdots & \vdots \\
    m_{n1} & m_{n,k-1} & a_{nk}^{(k)} & \cdots & a_{n,n}^{(k)} \\
\end{bmatrix}
\]

Pivoting if

\[|a_{ik}^{(k)}| \gg |a_{kk}^{(k)}|, \quad i > k\]

To do this interchange of rows $i$ and $k$,

use a pivot vector:

\[
\begin{cases}
    p_k = i \\
    p_k = k
\end{cases}
\]

Pivot element vector

\[p_i, \quad i = 1, \ldots n\]

Forward substitution, step $k$

\[
\overline{L}\vec{y} = \vec{b}
\]

Interchange rows $i$ and $k$

If $p_k = i$ \[\begin{cases}
    b_i^{(k)} = b_k \\
    b_k = b_i \\
    b_i = b_i^{(k)}
\end{cases}\]
LU Decomposition / Factorization: Variations

- **Doolittle decomposition:**
  - $m_{ii}=1$, stored in $L$

- **Crout decomposition:**
  - Directly impose diagonal of $U$ equal to 1’s.
  - Sweeps both by columns and rows
  - Reduce storage needs
  - Each element of $A$ only employed once

- **Matrix inverse:** $AX=I \Rightarrow (LU)X=I$

  - Numbers of ops: $O\left(\frac{2n^3}{3} + \frac{pn^2}{\text{LU Decomp.}} + \frac{pn^2}{\text{Backward Substitution}}\right)$ for $p=n$, $\Rightarrow \frac{2n^3}{3} + 2n^3 = \frac{8n^3}{3}$
Recall Lecture 2: The Condition Number

• The *condition* of a mathematical problem relates to its sensitivity to changes in its input values.

• A computation is *numerically unstable* if the uncertainty of the input values are magnified by the numerical method.

• Considering \( x \) and \( f(x) \), the *condition number* is the ratio of the relative error in \( f(x) \) to that in \( x \).

• Using first-order Taylor series:
  \[
  f(x) = f(x) + f'(x)(x - x) 
  \]

• Relative error in \( f(x) \):
  \[
  \frac{f(x) - f(x)}{f(x)} \approx \frac{f'(x)(x - x)}{f(x)} 
  \]

• Condition Nb = Ratio of relative errors:
  \[
  K_p = \frac{x f'(x)}{f(x)} 
  \]
Linear Systems of Equations
Error Analysis

Function of one variable

\[ y = f(x) \]

Condition number

\[ \left| \frac{f(\bar{x}) - f(x)}{f(x)} \right| = K \left| \frac{\bar{x} - x}{x} \right|, \quad \bar{x} = x + \delta x \]

\[ \left| \frac{\delta y}{y} \right| = K \left| \frac{\delta x}{x} \right| \]

The condition number \( K \) is a measure of the amplification of the relative error by the function \( f(x) \)

Linear systems

How is the relative error of \( \bar{x} \) dependent on errors in \( \bar{b} \)?

\[ \overline{A} \bar{x} = \bar{b} \]

Example

\[ \overline{A} = \begin{bmatrix} 1.0 & 1.0 \\ 1.0 & 1.0001 \end{bmatrix}, \quad \text{det}(\overline{A}) = 0.0001 \]

Using MATLAB with different \( \bar{b} \)'s (see tbt8.m):

\[ \bar{b} = \begin{bmatrix} 2 \\ 2 \end{bmatrix} \Rightarrow \bar{x} = \begin{bmatrix} 2 \\ 0 \end{bmatrix} \]

\[ \bar{b} = \begin{bmatrix} 2 \\ 2.0001 \end{bmatrix} \Rightarrow \bar{x} = \begin{bmatrix} 1 \\ 1 \end{bmatrix} \]

Small changes in \( \bar{b} \) give large changes in \( \bar{x} \)

The system is ill-Conditioned
Linear Systems of Equations: Norms

Evaluation of Condition Numbers requires use of Norms

Vector and Matrix Norms:

\[
\begin{align*}
|x|_\infty &= \max_i |x_i| \\
|A|_\infty &= \max_i \sum_{j=1}^n |a_{ij}|
\end{align*}
\]

Properties:

\[A \neq 0 \Rightarrow |A| > 0\]

\[|\alpha A| = |\alpha| |A|\]

\[|A + B| \leq |A| + |B|\]

Sub-multiplicative / Associative Norms (Banach Algebra/space)

\[|AB| \leq |A||B|\]

\[|Ax| \leq |A||x|\]
Examples of Matrix Norms

\[ \|A\|_1 = \max_{1 \leq j \leq n} \sum_{i=1}^{m} |a_{ij}| \]

“Maximum Column Sum”

\[ \|A\|_\infty = \max_{1 \leq i \leq m} \sum_{j=1}^{n} |a_{ij}| \]

“Maximum Row Sum”

\[ \|A\|_F = \left( \sum_{i=1}^{m} \sum_{j=1}^{n} |a_{ij}|^2 \right)^{1/2} \]

“The Frobenius norm” (also called Euclidean norm), which for matrices differs from:

\[ \|A\|_2 = \sqrt{\lambda_{\max}(A^*A)} \]

“The l-2 norm” (also called spectral norm)
Linear Systems of Equations

Error Analysis: Perturbed Right-hand Side

Vector and Matrix Norms

\[ \|\mathbf{x}\|_\infty = \max_i |x_i| \]
\[ \|\mathbf{A}\|_\infty = \max_i \sum_{j=1}^n |a_{ij}| \]

Properties

\[ \mathbf{A} \neq \mathbf{0} \Rightarrow \|\mathbf{A}\| > 0 \]

\[ \|\alpha \mathbf{A}\| = |\alpha| \|\mathbf{A}\| \]
\[ \|\mathbf{A} + \mathbf{B}\| \leq \|\mathbf{A}\| + \|\mathbf{B}\| \]
\[ \|\mathbf{AB}\| \leq \|\mathbf{A}\| \|\mathbf{B}\| \]
\[ \|\mathbf{Ax}\| \leq \|\mathbf{A}\| \|\mathbf{x}\| \]

Perturbed Right-hand Side implies

\[ \mathbf{Ax} = \mathbf{b} \]
\[ \mathbf{A}(\mathbf{x} + \delta\mathbf{x}) = \mathbf{b} + \delta\mathbf{b} \]

Subtract original equation

\[ \mathbf{A} \delta\mathbf{x} = \delta\mathbf{b} \]
\[ \delta\mathbf{x} = \mathbf{A}^{-1} \delta\mathbf{b} \]

\[ \begin{align*}
\|\delta\mathbf{x}\| & \leq \|\mathbf{A}^{-1}\| \|\delta\mathbf{b}\| \\
\|\mathbf{b}\| &= \|\mathbf{Ax}\| \leq \|\mathbf{A}\| \|\mathbf{x}\| 
\end{align*} \] \Rightarrow

Relative Error Magnification

\[ \frac{\|\delta\mathbf{x}\|}{\|\mathbf{x}\|} \leq \|\mathbf{A}^{-1}\| \frac{\|\delta\mathbf{b}\|}{\|\mathbf{b}\|} \]

Condition Number

\[ K(\mathbf{A}) = \|\mathbf{A}^{-1}\| \|\mathbf{A}\| \]
Linear Systems of Equations

Error Analysis: Perturbed Coefficient Matrix

Vector and Matrix Norm

\[ ||\mathbf{x}||_\infty = \max_i |x_i| \]

\[ ||\mathbf{A}||_\infty = \max_i \sum_{j=1}^{n} |a_{ij}| \]

Properties

\[ \mathbf{A} \neq 0 \Rightarrow ||\mathbf{A}|| > 0 \]

\[ ||\alpha \mathbf{A}|| = |\alpha||\mathbf{A}| \]

\[ ||\mathbf{A} + \mathbf{B}|| \leq ||\mathbf{A}|| + ||\mathbf{B}|| \]

\[ ||\mathbf{A} \mathbf{B}|| \leq ||\mathbf{A}|| ||\mathbf{B}|| \]

\[ ||\mathbf{A} \mathbf{x}|| \leq ||\mathbf{A}|| ||\mathbf{x}|| \]

Perturbed Coefficient Matrix implies

\[ \left( \mathbf{A} + \delta \mathbf{A} \right) (\mathbf{x} + \delta \mathbf{x}) = \mathbf{b} \]

Subtract unperturbed equation

\[ \mathbf{A} \delta \mathbf{x} + \delta \mathbf{A} (\mathbf{x} + \delta \mathbf{x}) = \mathbf{0} \]

(Neglect 2nd order)

\[ \delta \mathbf{x} = -\mathbf{A}^{-1} \delta \mathbf{A} (\mathbf{x} + \delta \mathbf{x}) \sim -\mathbf{A}^{-1} \delta \mathbf{A} \mathbf{x} \]

Relative Error Magnification

\[ \left| \left| \delta \mathbf{x} \right| \right| \leq \left| \left| \mathbf{A}^{-1} \right| \right| ||\delta \mathbf{A}|| \left| \left| \mathbf{x} \right| \right| \]

\[ \left| \left| \frac{\delta \mathbf{x}}{\mathbf{x}} \right| \right| \leq \left| \left| \mathbf{A}^{-1} \right| \right| \left| \left| \mathbf{A} \right| \right| \frac{||\delta \mathbf{A}||}{||\mathbf{A}||} \]

Condition Number

\[ K(\mathbf{A}) = \left| \left| \mathbf{A}^{-1} \right| \right| \left| \left| \mathbf{A} \right| \right| \]
Using Cramer's rule:

\[
\begin{bmatrix}
  1.0 & 1.0 \\
  1.0 & 1.0001
\end{bmatrix}
\begin{bmatrix}
  a_{11} & a_{12} \\
  a_{21} & a_{22}
\end{bmatrix}
= \begin{bmatrix}
  1 & 0 \\
  0 & 1
\end{bmatrix}
\]

\[
\det(\overline{A}) = 0.0001
\]

\[
\begin{align*}
a_{11} &= \frac{1.0001}{0.0001} = 10,001 \\
a_{12} &= \frac{-1}{0.0001} = -10,000 \\
a_{21} &= \frac{-1}{0.0001} = -10,000 \\
a_{11} &= \frac{1.0}{0.0001} = 10,000
\end{align*}
\]

\[
\begin{bmatrix}
  1.0 & 1.0 \\
  1.0 & 1.0001
\end{bmatrix}
\begin{bmatrix}
  a_{11} & a_{12} \\
  a_{21} & a_{22}
\end{bmatrix}
= \begin{bmatrix}
  1 & 0 \\
  0 & 1
\end{bmatrix}
\Rightarrow K(\overline{A}) \approx 40,000
\]

4-digit Arithmetic

```matlab
tbt6.m

n=4
a = [ [1.0 1.0]' [1.0 1.0001]']
b= [1 2]'
ai=inv(a);
a_nrm=max( abs(a(1,1)) + abs(a(1,2)) ,
            abs(a(2,1)) + abs(a(2,2)) )
ai_nrm=max( abs(ai(1,1)) + abs(ai(1,2)) ,
            abs(ai(2,1)) + abs(ai(2,2)) )
k=a_nrm*ai_nrm
r=ai * b
x=[0 0];
m21=a(2,1)/a(1,1);
a(2,1)=0;
a(2,2) = radd(a(2,2),-m21*a(1,2),n);
b(2) = radd(b(2),-m21*b(1),n);
x(2) = b(2)/a(2,2);
x(1) = (radd(b(1), -a(1,2)*x(2),n))/a(1,1);x'
```
Example: Better-Conditioned System

Using Cramer's rule:

\[
\begin{bmatrix}
0.0001 & 1.0 \\
1.0 & 1.0 \\
\end{bmatrix}
\begin{bmatrix}
a_{11} & a_{12} \\
a_{21} & a_{22} \\
\end{bmatrix}
= 
\begin{bmatrix}
1 & 0 \\
0 & 1 \\
\end{bmatrix}
\]

\[
det(\mathbf{A}) = 0.9999
\]

\[
\begin{aligned}
a_{11} &= \frac{-1}{0.9999} = -1,0001 \\
a_{12} &= \frac{1}{0.9999} = 1.0001 \\
a_{21} &= \frac{1}{0.9999} = 1.0001 \\
a_{11} &= \frac{-0.0001}{0.9999} = -0.0001
\end{aligned}
\]

\[
\begin{bmatrix}
\mathbf{A} \\
\mathbf{A}^{-1}
\end{bmatrix}_\infty = 2.0 \\
\Rightarrow K(\mathbf{A}) \approx 4
\]

Relatively Well-conditioned system

4-digit Arithmetic

n=4
a = [[0.0001 1.0]' [1.0 1.0]']
b = [1 2]'
ai = inv(a);
a_nrm = max(abs(a(1,1)) + abs(a(1,2)), abs(a(2,1)) + abs(a(2,2)))
ai_nrm = max(abs(ai(1,1)) + abs(ai(1,2)), abs(ai(2,1)) + abs(ai(2,2)))
k = a_nrm * ai_nrm
r = ai * b
x = [0 0];
m21 = a(2,1)/a(1,1);
a(2,1) = 0;
a(2,2) = radd(a(2,2), -m21 * a(1,2), n);
b(2) = radd(b(2), -m21 * b(1), n);
x(2) = b(2)/a(2,2);
x(1) = (radd(b(1), -a(1,2) * x(2), n))/a(1,1);
x'