REVIEW Lecture 21:

- End of Time-Marching Methods: higher-order methods
  - Runge-Kutta Methods
    - Additional points are between t_n and t_{n+1}
  - Multistep/Multipoint Methods: Adams Methods
    - Additional points are at past time steps
  - Practical CFD Methods
  - Implicit Nonlinear systems
  - Deferred-correction Approach

$$\phi^{n+1} - \phi^n = \int_{t_n}^{t_{n+1}} f(t,\phi)\, dt$$

- Complex Geometries
  - Different types of grids
  - Choice of variable arrangements
    - Velocity vectors: Cartesian or Grid-oriented
    - Staggered or Collocated variables
- Grid Generation
  - Basic concepts and structured grids

TODAY (Lecture 22): Grid Generation and Intro to FV Complex Geometries

- Complex Geometries
- Grid Generation
  - Basic concepts and structured grids
    - Stretched grids
    - Algebraic methods (strecthed grids)
    - General coordinate transformation
    - Differential equation methods
    - Conformal mapping methods
  - Unstructured grid generation
    - Delaunay Triangulation
    - Advancing Front method
- Finite Volume on Complex geometries
  - Computation of convective fluxes
  - Computation of diffusive fluxes
  - Comments on 3D
- Solution of the Navier-Stokes Equations

References and Reading Assignments

- Chapter 8 on “Complex Geometries” of “J. H. Ferziger and M. Peric, *Computational Methods for Fluid Dynamics*. Springer, NY, 3rd edition, 2002”
- Chapter 9 on “Grid Generation” of T. Cebeci, J. P. Shao, F. Kafyeke and E. Laurendeau, *Computational Fluid Dynamics for Engineers*. Springer, 2005.
- Ref on Grid Generation only:
  - Thompson, J.F., Warsi Z.U.A. and C.W. Mastin, “Numerical Grid Generation, Foundations and Applications”, North Holland, 1985

Classes of Grid Generation

- An arrangement of discrete set of grid points or cells needs to be generated for the numerical solution of PDEs (fluid conservation equations)
  - Finite volume methods:
    - Can be applied to uniform and non-uniform grids
  - Finite difference methods:
    - Require a coordinate transformation to map the irregular grid in the spatial domain to a regular one in the computational domain
    - Difficult to do this in complex 3D spatial geometries
    - So far, only used with structured grid (could be used with unstructured grids with polynomials ϕ defining the shape of ϕ around a grid point)
- Three major classes of grid generation: i) algebraic methods, ii) differential equation methods and iii) conformal mapping methods
- Grid generation and solving PDE can be independent
  - A numerical (flow) solver can in principle be developed independently of the grid
  - A grid generator then gives the metrics (weights) and the one-to-one correspondence between the spatial-grid and computational-grid

Grid Generation: Basic Concepts for Structured Grids

- Structured Grids (includes curvilinear or non-orthogonal grids)
  - Often utilized with FD schemes
  - Methods based on coordinate transformations
- Consider irregular shaped physical domain (x, y) in Cartesian coordinates and determine its mapping to the computational domain in the (ξ, η) Cartesian coordinates
  - Increase ξ *or* η monotonically in physical domain along "curved lines"
  - Coordinate lines of the same family do not cross
  - Lines of different family don't cross more than once
  - Physical grid refined where large errors are expected

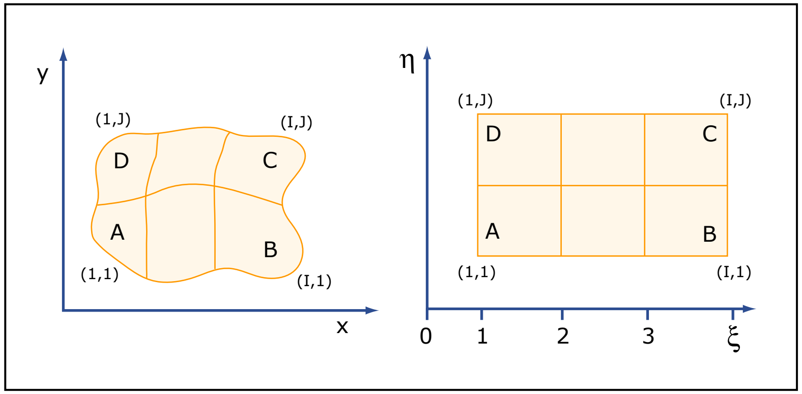


Image by MIT OpenCourseWare.

A simply-connected irregular shape in the physical plane is mapped as a rectangle in the computational plane.

  - Mapped (computational) region has a rectangular shape:
    - Coordinates (ξ, η) can vary from 1 to (I, J), with mesh sizes taken equal to 1
  - Boundaries are mapped to boundaries

Grid Generation:
Basic Concepts for Structured Grids, Cont'd

- The example just shown was the mapping of an irregular, simply connected, region into a rectangle.
- Other configurations are of course possible
  - For example, a L-shape domain can be mapped into:
    - a regular L-shape
    - or into a rectangular shape

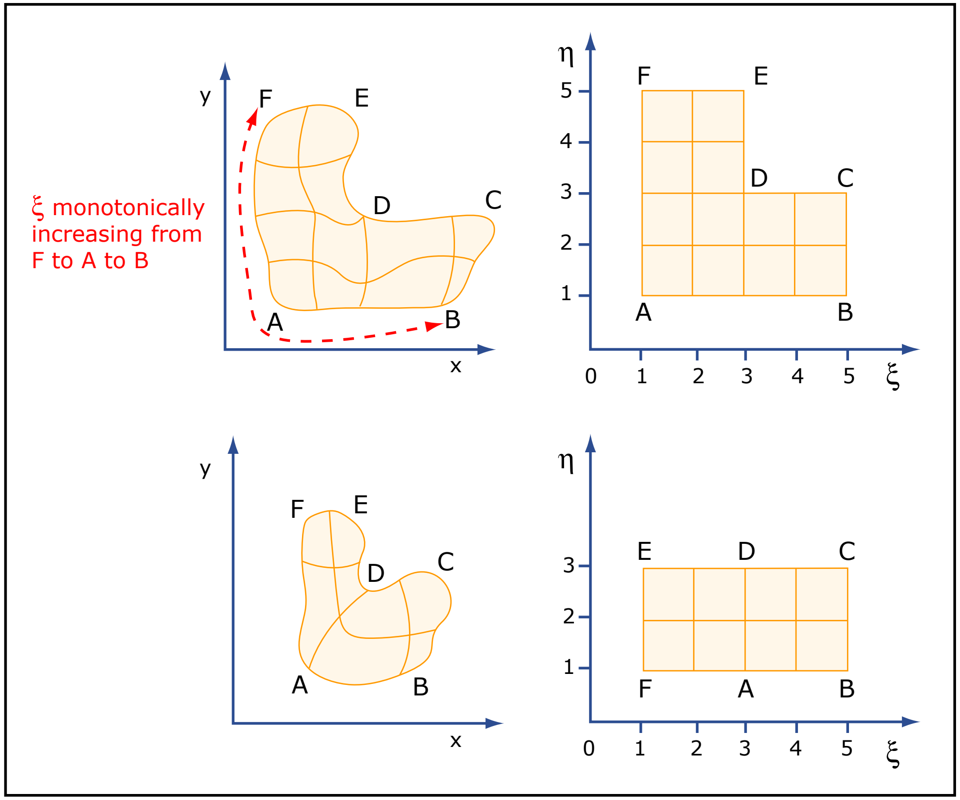


Image by MIT OpenCourseWare.

Grid Generation for Structured Grids: **Stretched Grids**

- Consider a viscous flow solution on a given body, where the velocity varies rapidly near the surface of the body (Boundary Layer)
- For efficient computation, a finer grid near the body and coarser grid away from the body is effective (aims to maintain constant accuracy)
- Possible coordinate transformation: a scaling "$\eta = \log(y)$" ↔ "$y = \exp(\eta)$"

$$\xi = x$$
$$\eta = 1 - \frac{\ln[A(y)]}{\ln B}$$

where $A(y) = \frac{\beta + (1 - y/h)}{\beta - (1 - y/h)}$ and $B = \frac{\beta + 1}{\beta - 1}$

The parameter β $(1 < \beta < \infty)$ is the stretching parameter. As β gets close to 1, more grid points are clustered to the wall in the physical domain.

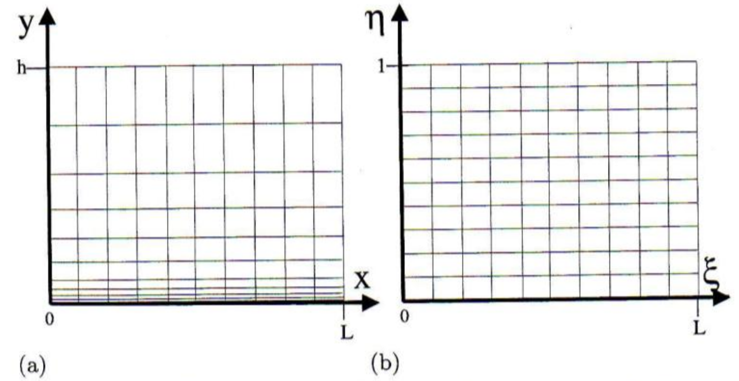


Fig. 9.4. One-dimensional stretching transformation. (**a**) Physical plane, (**b**) computational plane.

- Inverse transformation is needed to map solutions back from ξ, η domain:

$$x = \xi$$
$$\frac{y}{h} = \frac{(\beta + 1) - (\beta - 1)B^{1-\eta}}{1 + B^{1-\eta}}$$

Grid Generation for Structured Grids: **Stretched Grids, Cont'd**

- How do the conservation equations change?
- Consider the continuity equation for steady state flow in physical (x, y) space:

$$\nabla.(\rho\vec{v}) = 0 \quad \Rightarrow \quad \frac{\partial \rho u}{\partial x} + \frac{\partial \rho v}{\partial y} = 0$$

- In the computational plane, this equation becomes (chain rule)

$$\left.\begin{array}{l} \dfrac{\partial \rho u}{\partial x} = \dfrac{\partial \rho u}{\partial \xi}\dfrac{\partial \xi}{\partial x} + \dfrac{\partial \rho u}{\partial \eta}\dfrac{\partial \eta}{\partial x} \\ \dfrac{\partial \rho u}{\partial y} = \dfrac{\partial \rho v}{\partial \xi}\dfrac{\partial \xi}{\partial y} + \dfrac{\partial \rho v}{\partial \eta}\dfrac{\partial \eta}{\partial y} \end{array}\right\} \Rightarrow \frac{\partial \rho u}{\partial \xi}\xi_x + \frac{\partial \rho u}{\partial \eta}\eta_x + \frac{\partial \rho v}{\partial \xi}\xi_y + \frac{\partial \rho v}{\partial \eta}\eta_y = 0$$

- For our stretching transformation, one obtains:

$$\xi_x = 1, \quad \eta_x = 0, \quad \xi_y = 0, \quad \eta_y = \frac{2\beta}{h\ln(B)}\frac{1}{\beta^2 - (1 - y/h)^2}$$

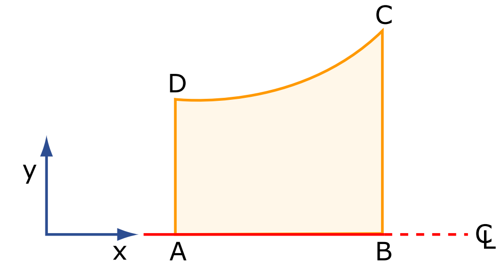
- Therefore, the continuity equation becomes:

$$\frac{\partial \rho u}{\partial \xi} + \frac{\partial \rho v}{\partial \eta}\eta_y = 0$$

  – This equation can be solved on a uniform grid (slightly more complicated eqn. system), and the solution mapped back to the physical domain using the inverse transform

Grid Generation for Structured Grids: Algebraic Methods

- Algebraic Method = Generalization of stretching method (2 & 3D)
- Consider fitting a diverging nozzle:
  - Let's assume a nozzle defined by: $y = x^2 \quad 1 \le x \le 2$



- Now, choose a curvilinear system:
  - Define the transformation:

$$\xi = x$$

$$\eta = \frac{y}{x^2}$$

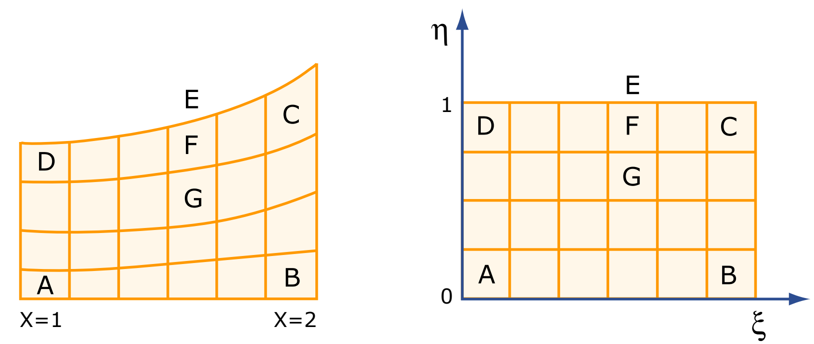


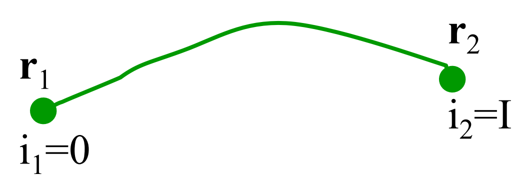
Image by MIT OpenCourseWare.

  - Metrics of the transformation: for the continuity eqn.

$$\frac{\partial \rho u}{\partial \xi}\xi_x + \frac{\partial \rho u}{\partial \eta}\eta_x + \frac{\partial \rho v}{\partial \xi}\xi_y + \frac{\partial \rho v}{\partial \eta}\eta_y = 0, \qquad \text{where} \quad \xi_x = 1, \quad \xi_y = 0, \quad \eta_x = -\frac{2\eta}{\xi}, \quad \eta_y = \frac{1}{\xi^2}$$

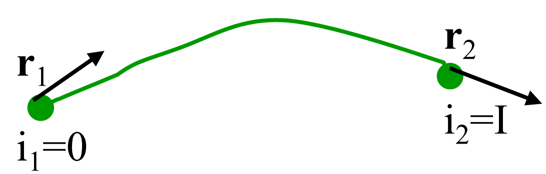
Algebraic Methods: Transfinite Interpolation

- Multi-directional interpolation (Transfinite Interpolation)
  - To generate algebraic grids within more complex domains or around more complex configurations, multi-directional interpolations can be used
    - They consist of a suite of unidirectional interpolations
- Unidirectional Interpolations (1D curve)
  - The Cartesian coordinate vector of each point on a curve **r**(x,y) is obtained as an interpolation between points that lie on the boundary curves
  - How to interpolate? the regulars:
    - Lagrange Polynomials: match function values



$$\vec{r}(i) = \sum_{k=0}^{n} L_k(i)\, \vec{r}_k \quad \text{with} \quad L_k(i) = \prod_{j=0, j\neq k}^{n} \frac{i - i_j}{i_k - i_j},$$

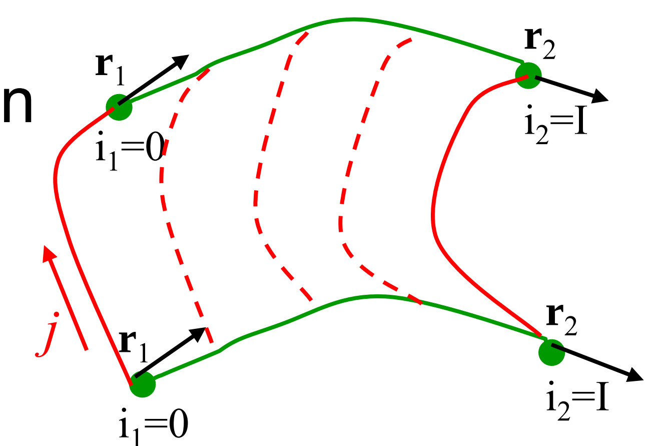
- Hermite Polynomials: match both function and 1st derivative values



$$\vec{r}(i) = \sum_{k=1}^{n} a_k(i)\, \vec{r}_k + \sum_{k=1}^{m} b_k(i)\, \vec{r}\,'_k$$

Grid Generation for Structured Grids: Algebraic Methods: Transfinite Interpolation, Cont'd

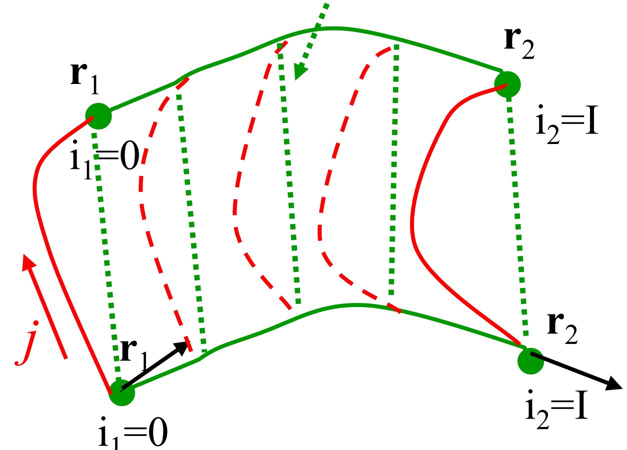
- Unidirectional Interpolations (1D curve), Cont'd
  - Lagrange and Hermite Polynomials fit a single polynomial from one boundary to the next => for long boundaries, oscillations may occur
  - Alternative, use set of lower order polynomials to form a piece-wise continuous interpolation:
    - Spline interpolation (match as many derivatives as possible at interior point junctions), Tension-spline (more localized curvature) and B-splines (allows local modification of the interpolation)
  - Use interpolation functions that are not polynomials, usually "stretching functions": exp, tanh, sinh, etc
- Multi-directional or Transfinite Interpolation
  - Extends 1D results to 2D or 3D by successive applications of 1D interpolations
  - For example, i then j.



Grid Generation for Structured Grids: Algebraic Methods: Transfinite Interpolation, Cont'd

- Multi-directional or Transfinite Interpolation, Cont'd
  - In 2D, the transfinite interpolation can be implemented as follows
    - Interpolate position vectors $\mathbf{r}$ in i-direction => leads to points $\mathbf{f}_1 = \square_i(\mathbf{r})$ and i-lines
    - Evaluate the difference between $\mathbf{r}$ and this result on the j-lines that will be used in the j-interpolation (e.g. difference with curved $i=0$ and $i=\mathrm{I}$): $\mathbf{r} - \mathbf{f}_1$
    - Interpolation of the discrepancy in the j-direction: $\mathbf{f}_2 = \square_j(\mathbf{r} - \mathbf{f}_1)$
    - Addition of the results of this j-interpolation to the results of the i-interpolation: $\mathbf{r}\,(i, j) = \mathbf{f}_1 + \mathbf{f}_2$

- Of course, Lagrange, Hermite Polynomials, Spline and non-polynomial (stretching) functions can be used for transfinite interpolations
- In 2D, inputs to program are 4 boundaries
- Issues: Propagates discontinuities in the interior and grid lines can overlap in some situations
- => needs to be refined by grid generator solving a PDE



Grid Generation for Structured Grids: Algebraic Methods: Transfinite Interpolation, Cont'd

- Examples:

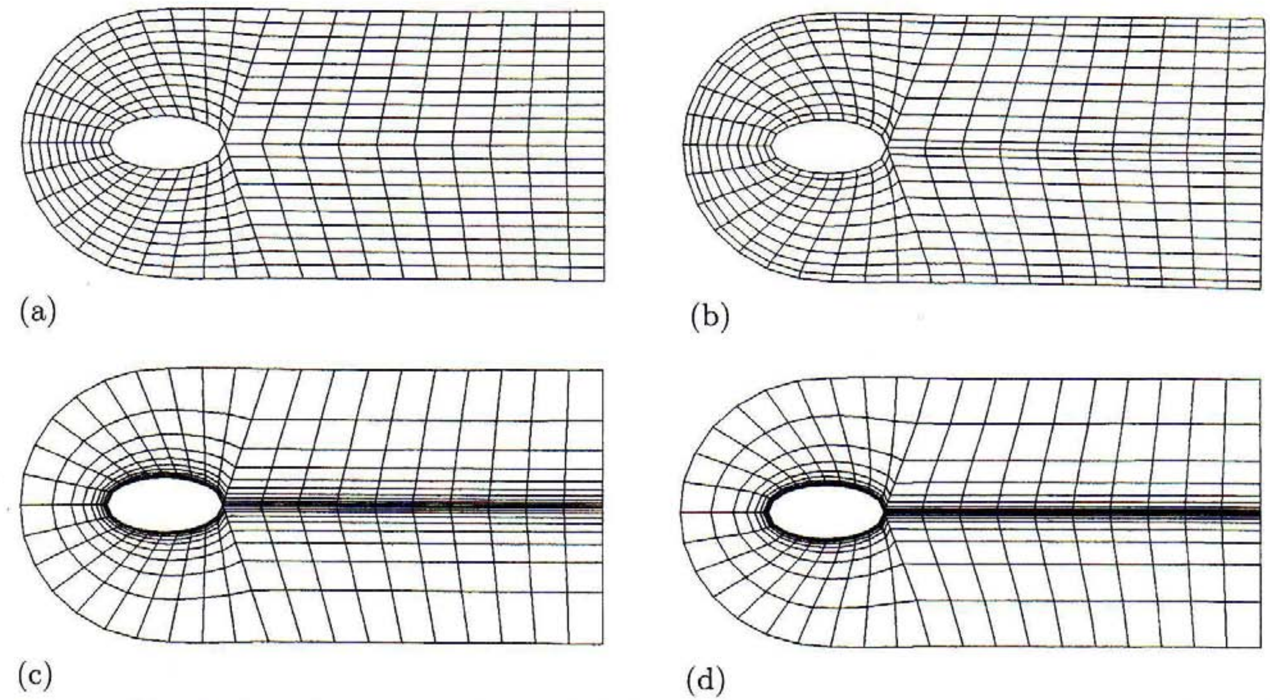


Fig. 9.12. (**a**) C-grid around ellipse: Unidirectional Lagrange Interpolation, (**b**) C-grid around ellipse: Unidirectional Hermite Interpolation, (**c**) C-grid around ellipse: Unidirectional Lagrange Interpolation with Hyperbolic Tangent Spacing, (**d**) C-grid around ellipse: Unidirectional Hermite Interpolation with Hyperbolic Tangent Spacing.

Grid Generation, structured Grids: General Coordinate transformation

- In general, coordinates are defined by a transformation: $x_i = x_i(\xi_j)$ $\quad$ (i and j = 1, 2, 3)
- All transformations are characterized by their Jacobian determinant J.

$$J = \det\left(\frac{\partial x_i}{\partial \xi_j}\right) = \begin{vmatrix} \frac{\partial x_1}{\partial \xi_1} & \frac{\partial x_1}{\partial \xi_2} & \frac{\partial x_1}{\partial \xi_3} \\ \frac{\partial x_2}{\partial \xi_1} & \frac{\partial x_2}{\partial \xi_2} & \frac{\partial x_2}{\partial \xi_3} \\ \frac{\partial x_3}{\partial \xi_1} & \frac{\partial x_3}{\partial \xi_3} & \frac{\partial x_3}{\partial \xi_3} \end{vmatrix}$$

  – For Cartesian vector components, one only needs to transform derivatives. One has:

$$\frac{\partial \phi}{\partial x_i} = \frac{\partial \phi}{\partial \xi_j}\frac{\partial \xi_j}{\partial x_i} = \frac{\partial \phi}{\partial \xi_j}\frac{\beta^{ij}}{J}, \quad \text{where } \beta^{ij} \text{ represents the cofactor of } \frac{\partial x_i}{\partial \xi_j} \text{ (element } i, j \text{ of Jacobian matrix)}$$

  – In 2D, $x = x(\xi,\eta)$ and $\phi = \phi(\xi,\eta)$, this leads to:

$$\frac{\partial \phi}{\partial x} = \frac{\partial \phi}{\partial \xi}\frac{\partial \xi}{\partial x} + \frac{\partial \phi}{\partial \eta}\frac{\partial \eta}{\partial x} = \frac{\partial \phi}{\partial \xi}\frac{\beta^{11}}{J} + \frac{\partial \phi}{\partial \eta}\frac{\beta^{12}}{J} = \frac{1}{J}\left(\frac{\partial \phi}{\partial \xi}\frac{\partial y}{\partial \eta} - \frac{\partial \phi}{\partial \eta}\frac{\partial y}{\partial \xi}\right)$$

Recall: the minor element m_{ij} corresponding to a_{ij} is the determinant of the submatrix that remains after the i^{th} row and the j^{th} column are deleted from $\mathbf{A}$. The cofactor c_{ij} of a_{ij} is: $c_{ij} = (-1)^{i+j}\, m_{ij}$

Structured Grids: Coordinate transformation, Cont'd

- How do the conservation equations transform?
  The generic conservation equation in Cartesian coordinates:

$$\frac{\partial \rho\phi}{\partial t} + \nabla.(\rho\phi\vec{v}) = \nabla.(k\nabla\phi) + s_\phi \quad \Leftrightarrow \quad \frac{\partial \rho\phi}{\partial t} + \frac{\partial}{\partial x_j}\left(\rho\phi v_j - k\frac{\partial \phi}{\partial x_j}\right) = s_\phi$$

- becomes:

$$J\frac{\partial \rho\phi}{\partial t} + \frac{\partial}{\partial \xi_j}\left(\rho\phi U_j - \frac{k}{J}\left(\frac{\partial \phi}{\partial \xi_m}B^{mj}\right)\right) = J\ s_\phi$$

where:

$$U_j = v_k\beta^{kj} = v_1\beta^{1j} + v_2\beta^{2j} + v_3\beta^{3j}$$ is proportional to the velocity component aligned with $\vec{\xi}_j$ (normal to $\xi_j = \text{const.}$)

$$B^{mj} = \beta^{kj}\beta^{km} = \beta^{1j}\beta^{1m} + \beta^{2j}\beta^{2m} + \beta^{3j}\beta^{3m}$$ are coefficients, sum of products of cofactors β^{ij}

- As a result, each 1st derivative term is replaced by a sum of three terms which contains derivatives of the coordinates as coefficients
- Unusual features of conservation equations in non-orthogonal grids:
  – Mixed derivatives appear in the diffusive terms and metrics coefficients appear in the continuity eqn.

Structured Grids: Coordinate transformation, Cont'd
Some Comments

- Coordinate transformation often presented only as a means of converting a complicated non-orthogonal grid into a simple, uniform Cartesian grid (the computational domain, whose grid-spacing is arbitrary)
- However, simplification is only apparent:
  - Yes, the computational grid is simpler than the original physical one
  - But, the information about the complexity in the computational domain is now in the metric coefficients of the transformed equations
    - i.e. discretization of computational domain is now simple, but the calculation of the Jacobian and other geometric information is not trivial (the difficulty is hidden in the metric coefficients)
- As mentioned earlier, FD method can in principle be applied to unstructured grids: specify a local shape function, differentiate and write FD equations. Has not yet been done.

Grid Generation for Structured Grids: Differential Equation Methods

- Grid transformation relations determined by a finite-difference solution of PDEs
  - For 2D problems, two elliptic (Poisson) PDEs are solved
  - Can be done for any coordinate systems, but here we will use Cartesian coordinates. The 2D transformation is then:
    - From the physical domain (x, y) to the computational domain (ξ, η)
    - At physical boundaries, one of ξ, η *is* constant, the other is monotonically varying
    - At interior points:

$$\frac{\partial^2 \xi}{\partial x^2} + \frac{\partial^2 \xi}{\partial y^2} = P(\xi, \eta)$$

$$\frac{\partial^2 \eta}{\partial x^2} + \frac{\partial^2 \eta}{\partial y^2} = Q(\xi, \eta)$$

  where $P(\xi,\eta)$ and $Q(\xi,\eta)$ are called the "control functions"

    - Their selection allows to concentrate the ξ, η lines in specific regions
    - If they are null, coordinates will tend to be equally spaced away from boundaries
    - Boundary conditions: ξ, η specified on boundaries of physical domain

Differential Equation Methods, Cont'd

- Computations to generate the grid mapping are actually carried out in the computational domain (ξ, η) itself !
  - don't want to solve the elliptic problem in the complex physical domain!
- Using the general rule, the elliptic problem is transformed into:

$$\alpha \frac{\partial^2 x}{\partial \xi^2} - 2\beta \frac{\partial^2 x}{\partial \xi \partial \eta} + \gamma \frac{\partial^2 x}{\partial \eta^2} + J^2 \left(P \frac{\partial x}{\partial \xi} + Q \frac{\partial x}{\partial \eta} \right) = 0$$

$$\alpha \frac{\partial^2 y}{\partial \xi^2} - 2\beta \frac{\partial^2 y}{\partial \xi \partial \eta} + \gamma \frac{\partial^2 y}{\partial \eta^2} + J^2 \left(P \frac{\partial y}{\partial \xi} + Q \frac{\partial y}{\partial \eta} \right) = 0$$

where $\alpha = x_\eta^2 + y_\eta^2 \,; \quad \beta = x_\xi x_\eta + y_\xi y_\eta \,; \quad \gamma = x_\xi^2 + y_\xi^2 \,; \quad J = x_\xi y_\eta - x_\eta y_\xi \quad (\text{with } x_\xi = \frac{\partial x}{\partial \xi}, \text{ etc})$

  - Boundary conditions are now the transformed values of the BCs in (x, y) domain: they are the values of the positions (x, y) of the grid points on the physical domain mapped to their locations in the computational domain
  - Equations can be solved by FD method to determine values of every grid point (x, y) in the interior of the physical domain
- Method developed by Thomson et al, 1985 (see ref)

Grid Generation for Structured Grids: **Differential Equation Methods, Example**

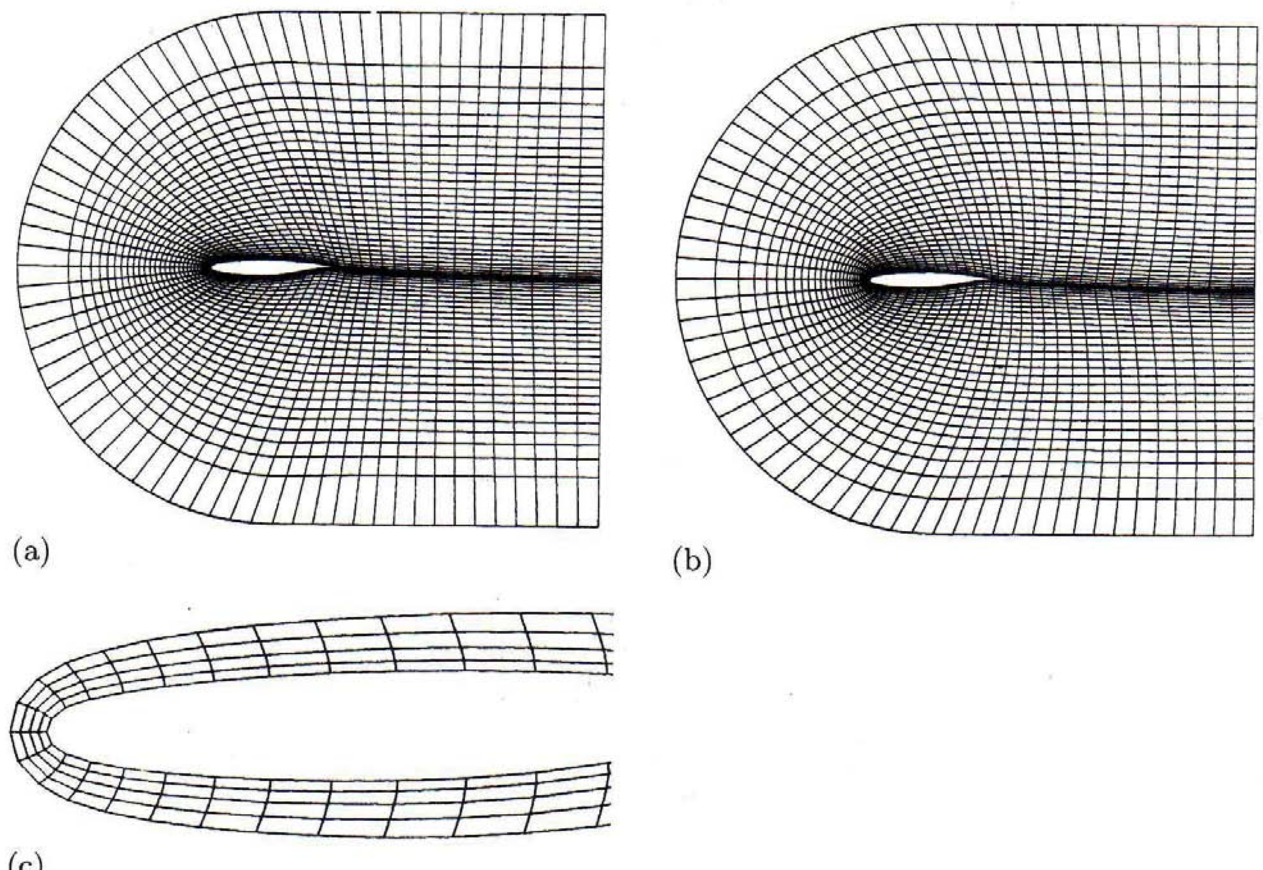


Fig. 9.13. (**a**) Starting algebraic C-grid around an airfoil section; 70 × 30 grid points; inner spacing $\Delta S_1 = 0.015c$, outer spacing $\Delta S_2 = 0.3c$, (**b**) Elliptic C-grid obtained after smoothing the algebraic grid of (**a**) by the solution of Poisson equations (50 iterations), (**c**) Close-up of the C-grid showing the application of orthogonality conditions near the leading edge region.

Grid Generation for Structured Grids: Conformal Mapping Methods

- Conformal mapping schemes are analytical or partially analytical (as opposed to differential equation methods)
- Restricted to two dimensional flows (based on complex variables): useful for airfoils
- Examples:

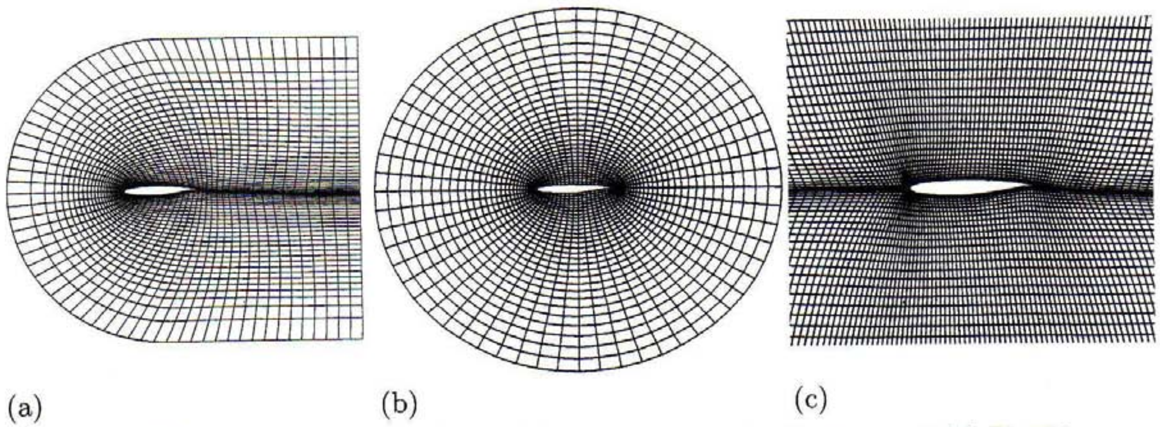


Fig. 9.14. Three common grids for airfoils. (**a**) C-grid, (**b**) O-grid, and (**c**) H-grid.

  – C-mesh: high density near leading edge of airfoil and good wake
  – O-mesh: high density near leading and trailing edge of airfoil
  – H-mesh: two sets of mesh lines similar to a Cartesian mesh, which is easiest to generate. Its mesh lines are often well aligned with streamlines

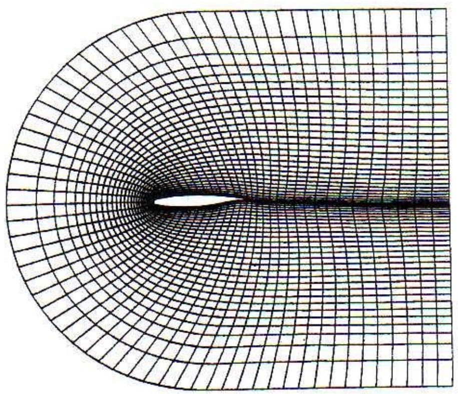
Conformal Mapping Methods: Example

- C-mesh example is generated by a parabolic mapping function
- It is essentially a set of confocal, orthogonal parabolas wrapping around the airfoil
- The mapping is defined by:

$$2(x+iy) = (\xi + i\eta)^2$$

or

$$2\,x = \xi^2 - \eta^2\,; \quad y = \xi\,\eta$$



- Inverse transformation:

$$\xi^2 = \sqrt{x^2+y^2} + x\,; \quad \eta^2 = \sqrt{x^2+y^2} - x$$

- Polar coordinates can be used for easier physical plane to computational plane transformation.
- In conformal mapping, singular point is point where mapping fails (here, it is the origin) => move it to half the distance from the nose radius

Grid Generation: Unstructured Grids

- Generating unstructured grid is complicated but now relatively automated in “classic” cases
- Involves succession of smoothing techniques that attempt to align elements with boundaries of physical domain
- Decompose domain into blocks to de-couple the problems
- Need to define point positions and connections
- Most popular algorithms:
  - Delaunay Triangulation Method
  - Advancing Front Method
- Two schools of thought: structured vs. unstructured, what is best for CFD?

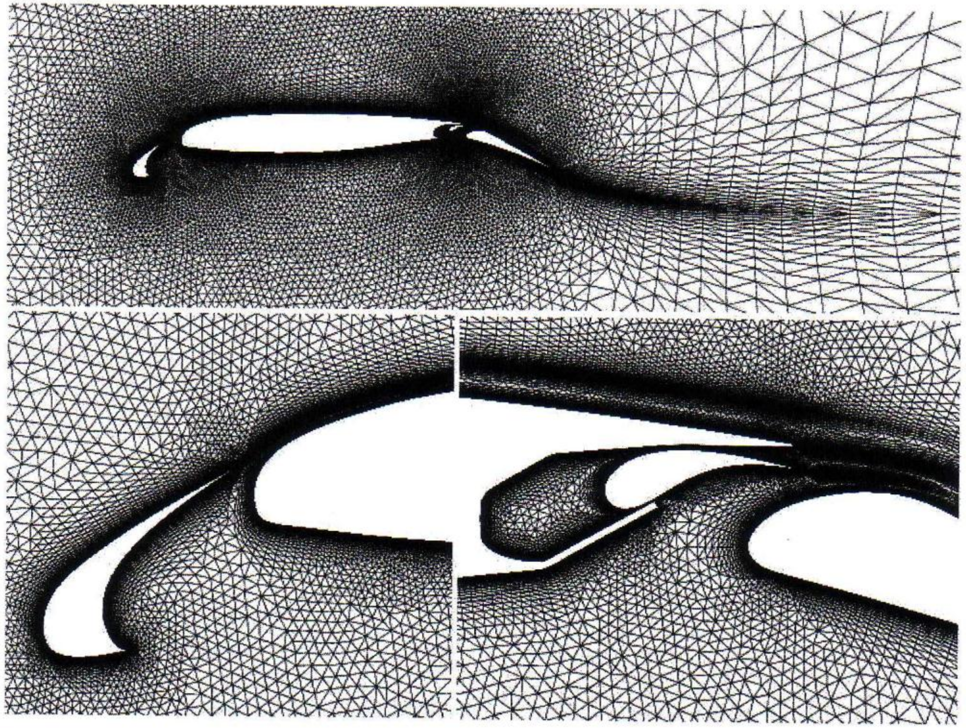


Fig. 9.16. 2D Unstructured grid for Navier–Stokes computations of a multi-element airfoil generated with the hybrid advancing front Delaunay method of Mavriplis [6].

- Structured grids: simpler grid and straightforward treatment of algebraic system, but mesh generation constraints on complex geometries
- Unstructured grids: generated faster on complex domains, easier mesh refinements, but data storage and solution of algebraic system more complex

2.29 Numerical Fluid Mechanics
Fall 2011