Basics of Mesh Generation

Computational Methods in Engineering Applications

IIT Kanpur, 12-16 April 2016

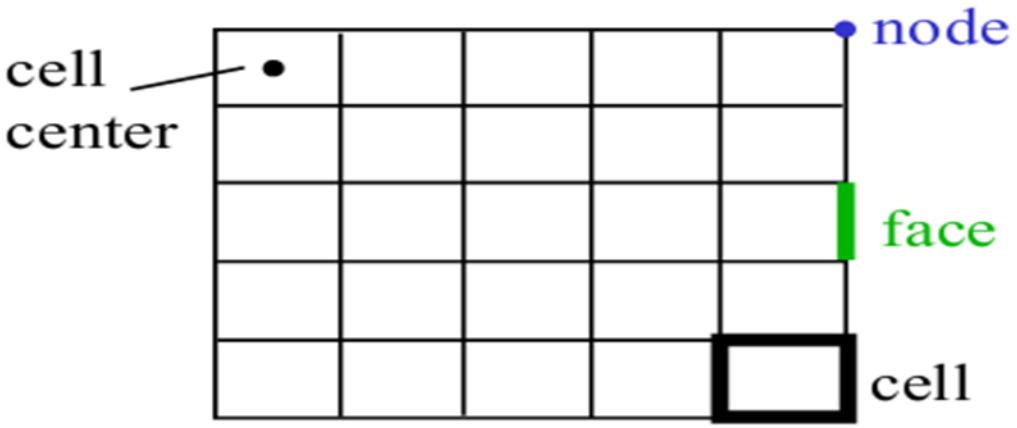
Syed Fahad Anwer
Fluid Mechanics Group
Department of Mechanical Engineering
Aligarh Muslim University, Aligarh

Outline

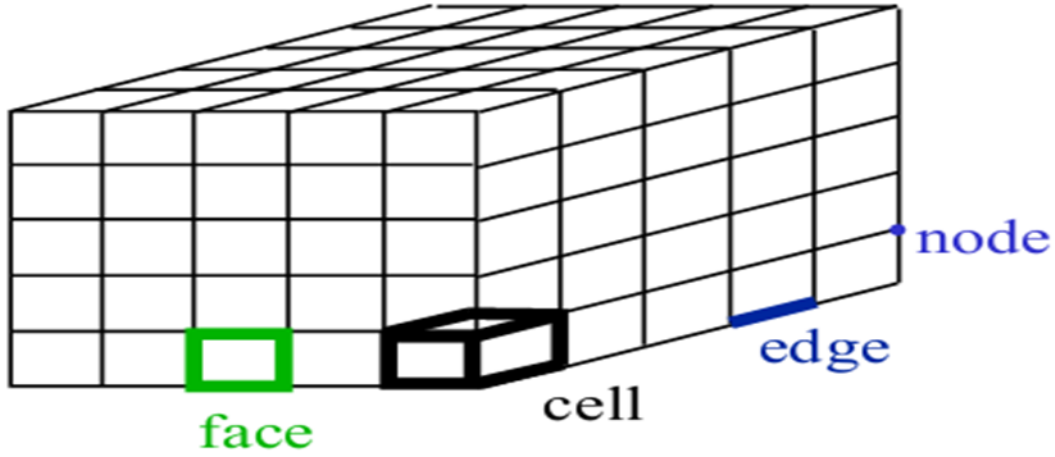
1. Introduction
2. Choice of grid
   - 2.1. Simple geometries
   - 2.2. Complex geometries
3. Grid generation
   - 3.1. Conformal mapping
   - 3.2. Algebraic methods
   - 3.3. Differential equation methods
   - 3.4. Commercial software

Introduction

- Flow domains are split into smaller sub-domains (hexahedra and tetrahedra in 3D and quadrilaterals and triangles in 2D)



2D computational grid

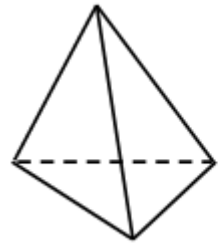


3D computational grid

Introduction

The grid has a significant impact on

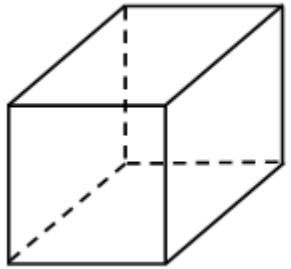
- Rate of convergence
- Solution accuracy
- CPU time required



tetrahedron
(**"tet"**)

Parameters of mesh quality

- Grid density.
- Adjacent cell length/volume ratios.(<20%)
- Skewness.(<0.85)
- Tet vs. hex.
- Mesh refinement through adaption.
- Orthogonality

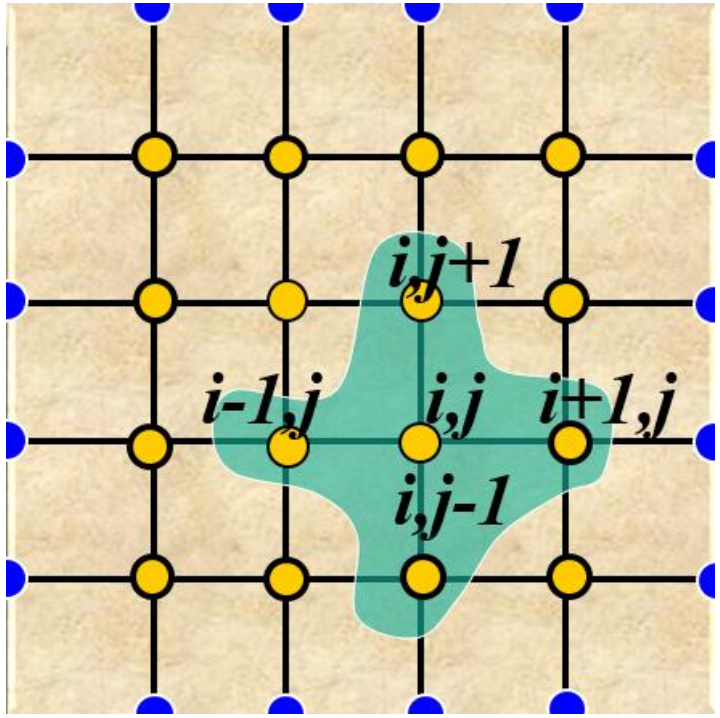


prism with
quadrilateral base
(**hexahedron** or **"hex"**)

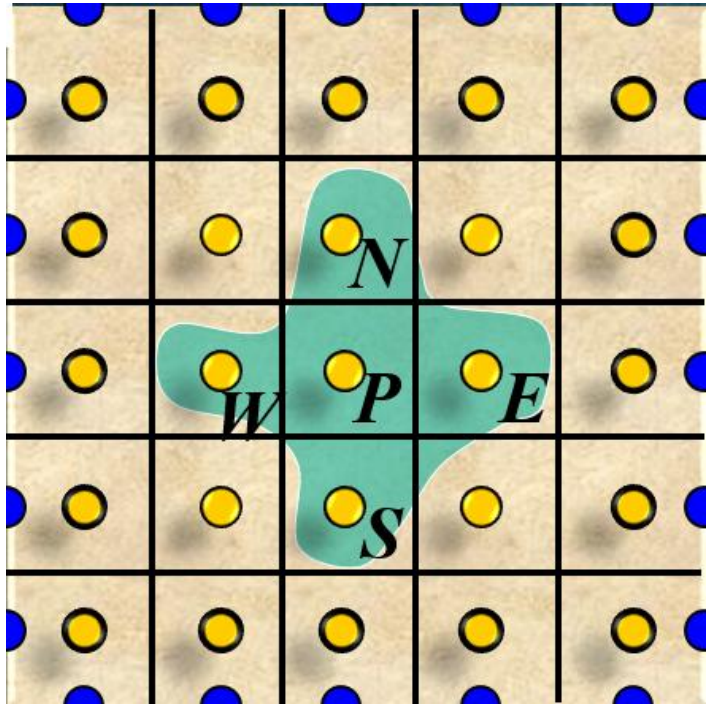
$$\left[\frac{\theta_{max}-90}{90}, \frac{90-\theta_{min}}{90}\right]$$

Simplest Grid Generation

Finite Difference method



Finite Volume method



- Finite Difference Method: Grid points are obtained by the intersection of grid lines (corresponding to the Cartesian or cylindrical or spherical coordinate system)

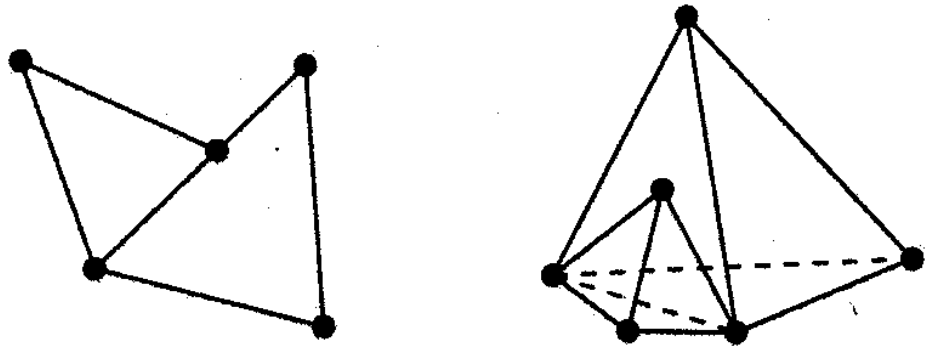
- Finite Volume Method (Cell centered): Grid points are defined at the centroids of the cells/CVs generated by the intersection of grid lines

Desirable Mesh Properties

1. **Compatible with solver**

- Finite-difference solver requires mesh to follow lines of constant coordinate
- Most finite-element and finite-volume codes are written only for grid elements of certain shapes (e.g., tetrahedron, hexahedral, etc.)

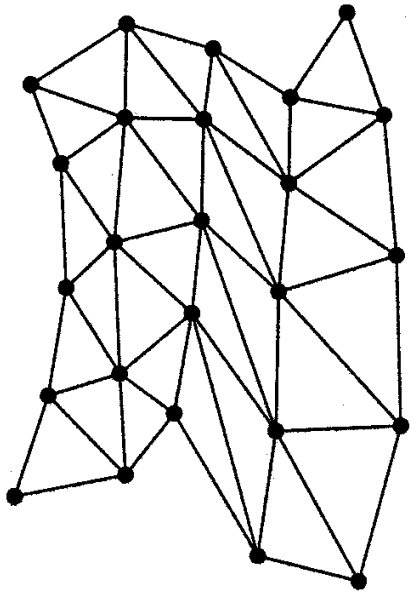
2. **Nodes of adjacent mesh elements are the same**



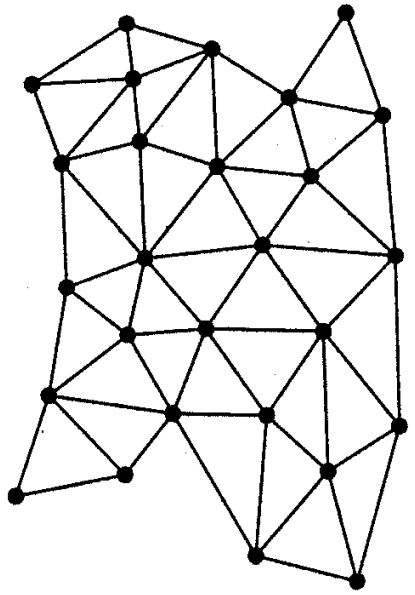
Examples of meshes that are not allowed

3. Element angles close to 90 degrees (Orthogonality)

- Meshes with angles that are too small or too large lead to inaccurate solutions, ill-conditioned matrices, and slow (or no) convergence of iterative solvers



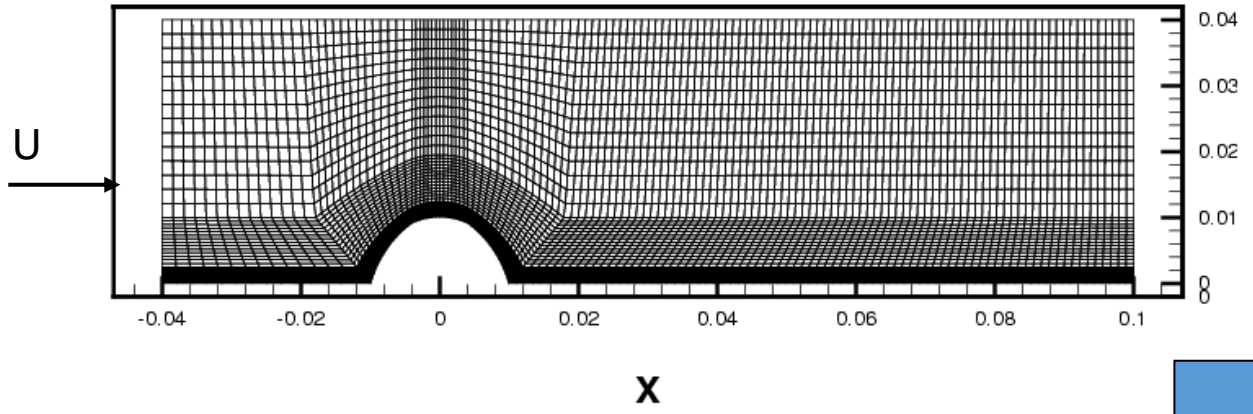
Poor Mesh

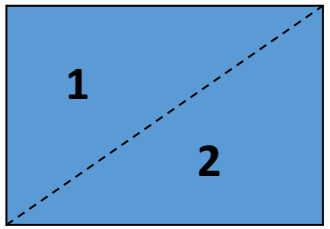


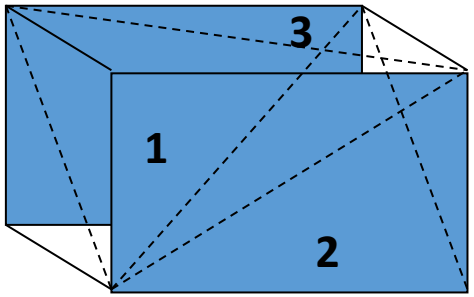
Good Mesh

4. Provides adequate resolution of computed fields

- Meshes must be finer in fluid/thermal boundary layers, near cracks in solids, near joints, within vortex cores, etc.







5. Uses minimum number of elements

- Triangles use twice as many elements as quadrilaterals
- Tetrahedral use six times as many elements as hexahedral

6. Easily refinable

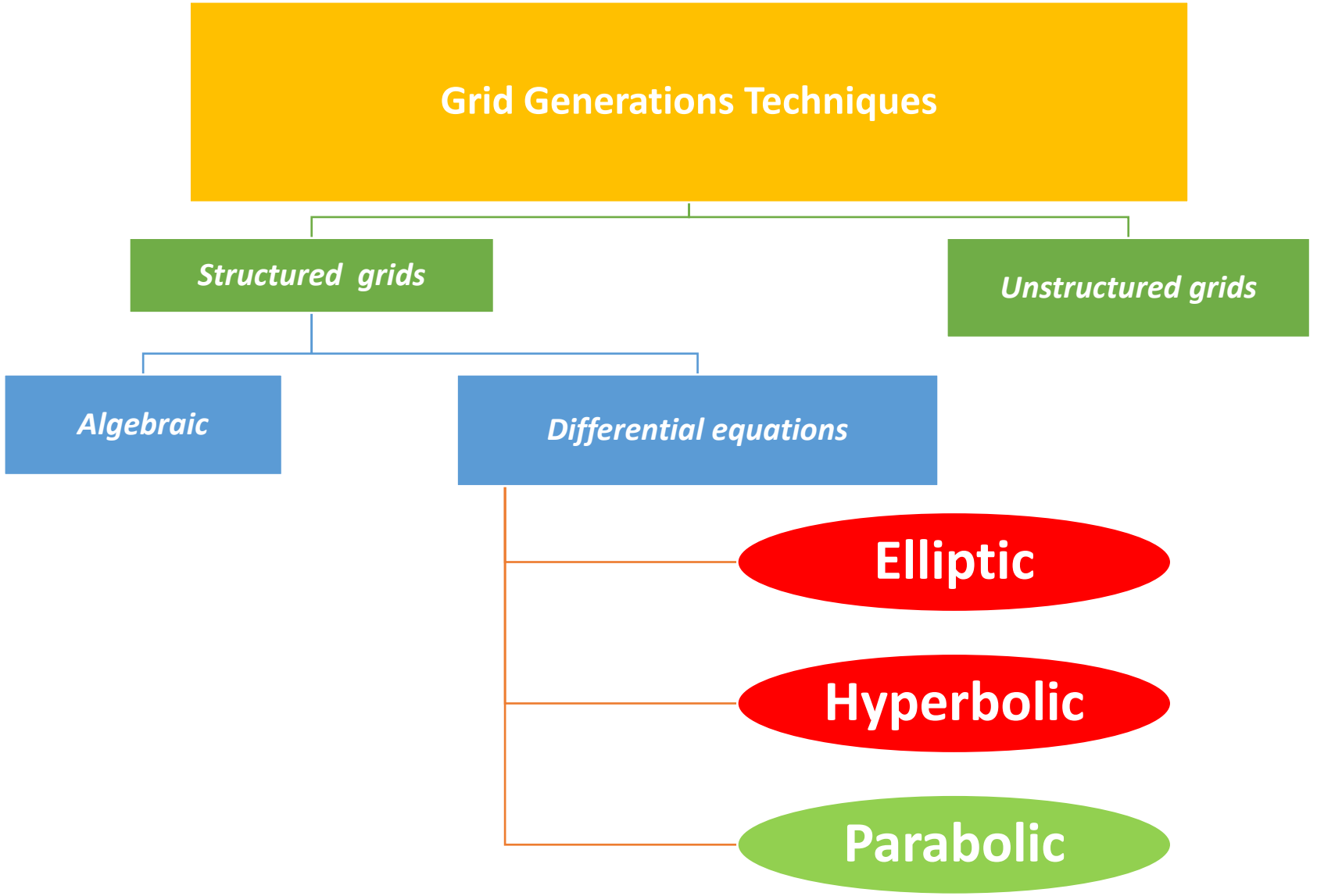
- We often want to do tests of resolution by varying the grid size in some systematic manner
- Useful property for multi-grid matrix iteration solvers and multi-scale computational approaches

7. Easy to generate

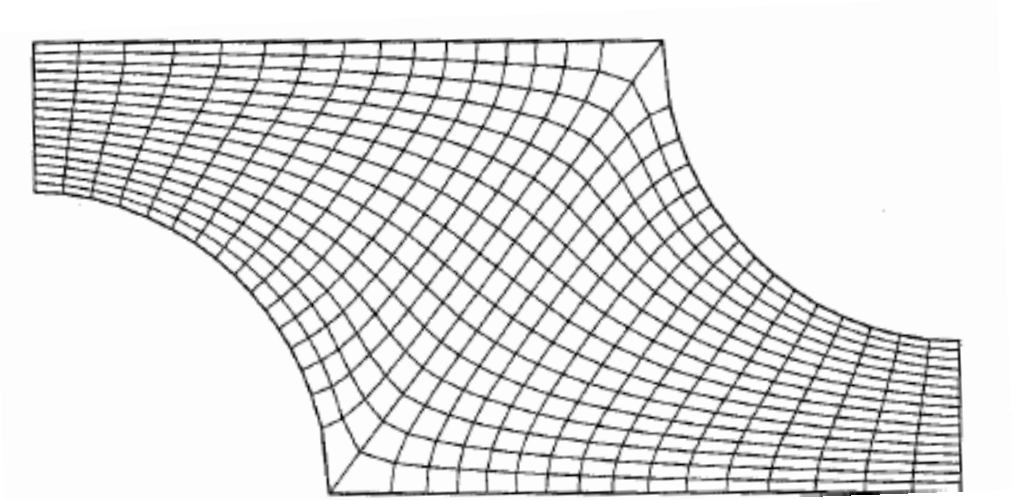
- Triangles and tetrahedrons are easy to generate using automatic grid generators
- Depends on capabilities of grid generators

8. One to One Correspondence

A mapping which guarantees one-to-one correspondence ensuring grid lines of the same family do not cross each other

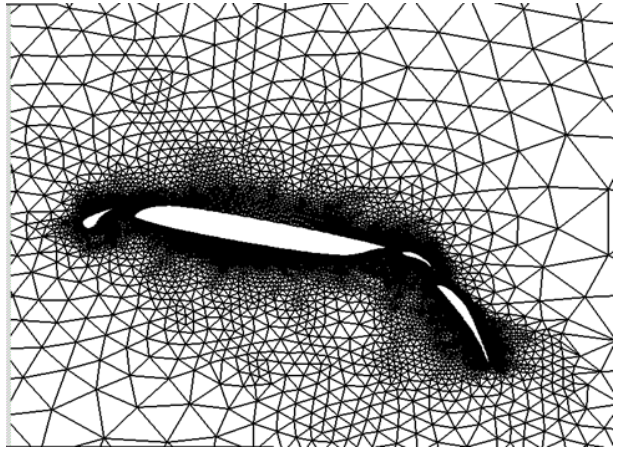

Grid Generations Techniques
Structured grids
Unstructured grids
Algebraic
Differential equations
Elliptic
Hyperbolic
Parabolic

Grid Generations Techniques



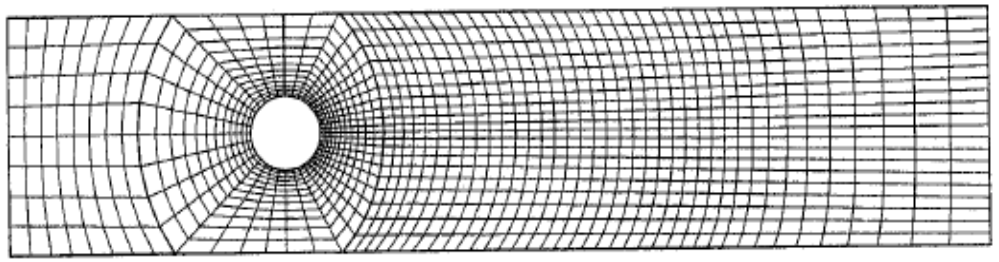
STRUCTURED GRID

- Regular connectivity
- Represented by i, j, k indices
- Conserves space



UNSTRUCTURED GRID

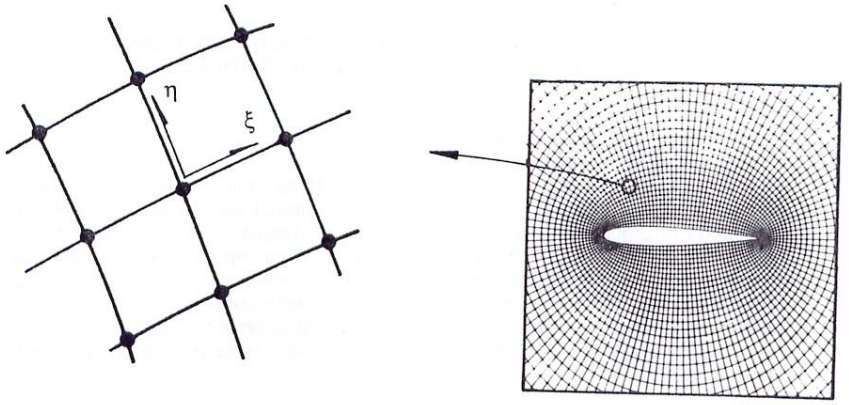
- Non-uniform pattern
- Represented by Node numbers
- Large storage requirement

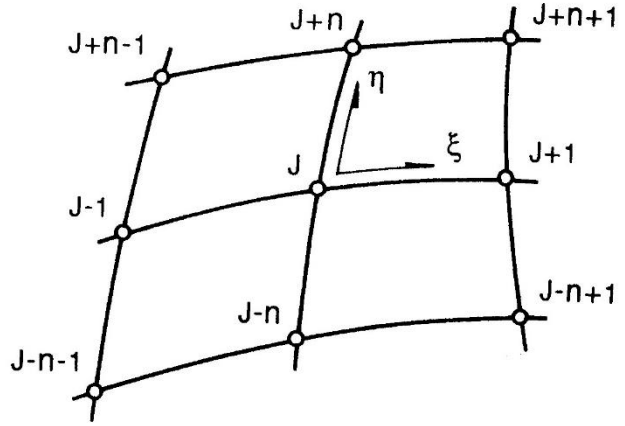


Block Structured Grid

Mesh Generation Concept

Structured Grids





Finite difference, finite volume and finite element discretisations

Mesh structure:

- Domain divided into a structured assembly of quadrilateral cells
- Each interior nodal points is surrounded by exactly the same number of mesh cells (or elements)
- Directions within the mesh can be immediately identify by associating a curvilinear co-ordinates system
- it is possible to immediately identify the nearest neigthbours of any node j on the mesh

Advantages:

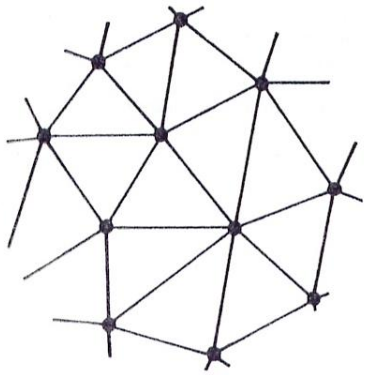
- Large number of algorithms for discretisation are available
- The algorithms can be normally implemented in a computationallly efficient manner

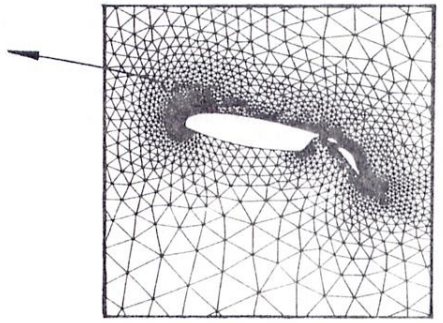
Disadvantages:

- Difficulty to generate grids of regions of general shapes → multi-block grids
- Very high elapsed time necessary to produce a grid for domains of extremely complex shape

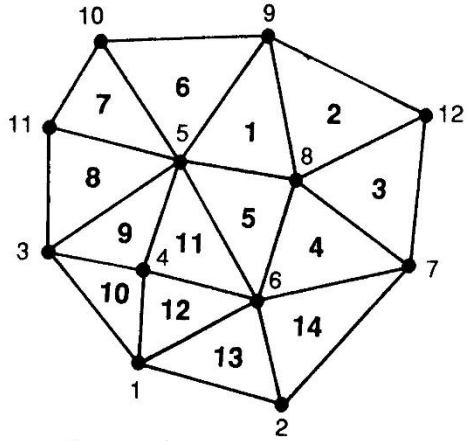
Mesh Generation Concept

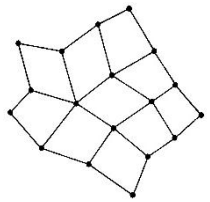
Unstructured Grids





Element	Nodes		
1	5	8	9
2	9	8	12
3	8	7	12
4	6	7	8
5	5	6	8
6	5	9	10
7	11	5	10
8	3	5	11
9	3	4	5
10	1	4	3
11	4	6	5
12	1	6	4
13	1	2	6
14	2	7	6





Finite volume and finite element discretisations

Mesh structure:

- Computational domain divided into an unstructured assembly of computational cells
- The number of cells surrounding a typical interior node is not necessarily constant
- The nodes and the elements has to be numbered
- To get the necessary information on the neightbours the numerotation of the nodes wich belong to each element has to be stored
- The concept of directionality does not exist anymore

Advantages:

- Powerful tool for discretising domains of complex shapes
- Unstructured mesh methods naturally offer the possibility of incorporating adaptivity

Disadvantages:

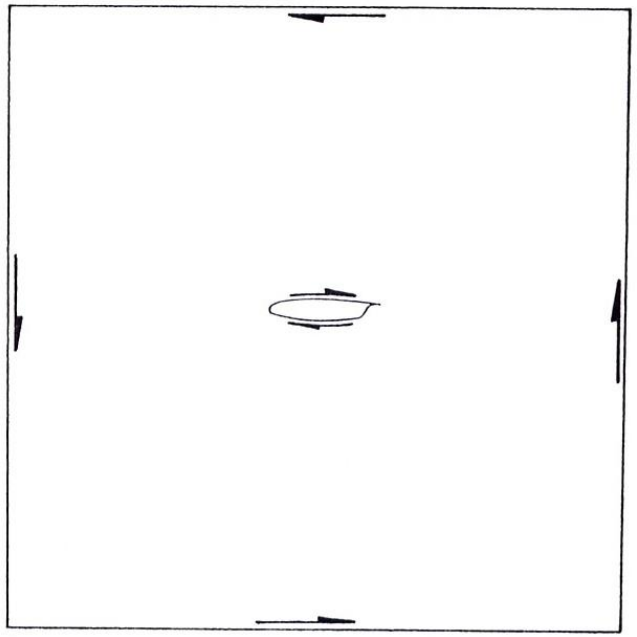
- Alternative solution algorithms are more limited
- Computational implementation places large demands on both computer memory and CPU

Geometry Modelling

The boundary of the domain to be discretised needs to be represented in a suitable manner before the generation can start. If the automatic discretisation of an arbitrary domain is to be achieved, the mathematical description of the domain topology has to possess the greatest possible generality.
The computer implementation of this description must provide means for automatically computing any geometrical quantity relevant to the generation procedure.

Mathematical description of the geometry → CAD model

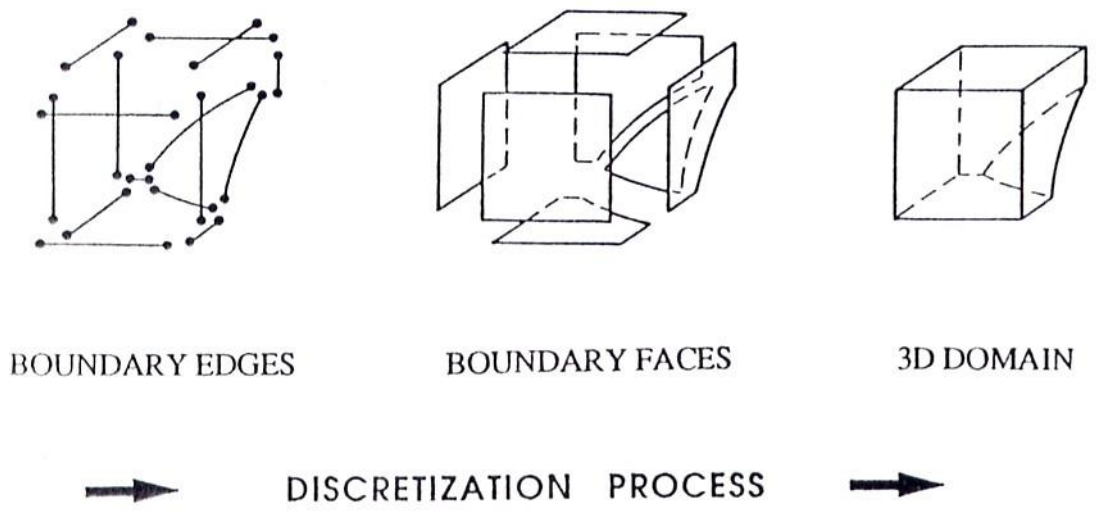
<u>Planar two-dimensional case</u>
In the planar two dimensional case, the boundary is represented by closed loops of orientated composite curves. For simple connected domains these curves are orientated in a counter-clockwise sense while for multiple connected regions the exterior boundary curves are given a counter-clockwise orientation and all the interior boundary curves are orientated in a clockwise sense



Geometry Modelling

Three dimensional case

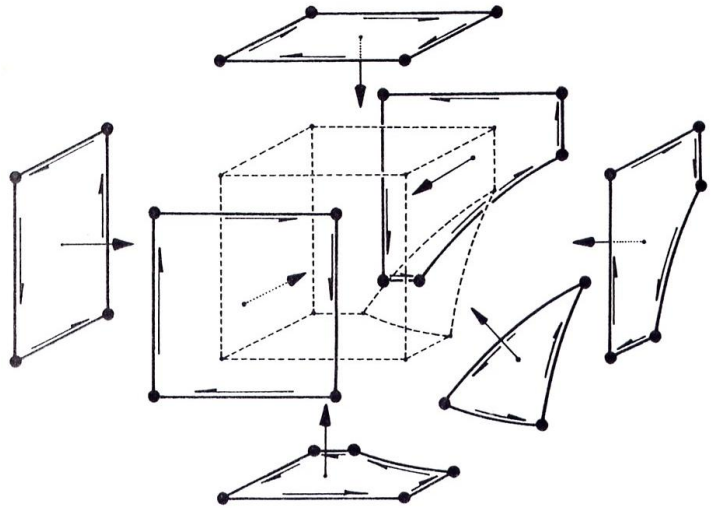
In three dimensions, the domain to be discretised is viewed as a region bounded by surfaces which intersect along curves. The portions of these curves and surfaces needed to define the three dimensional domain of interest are called *curve and surface components*, respectively.



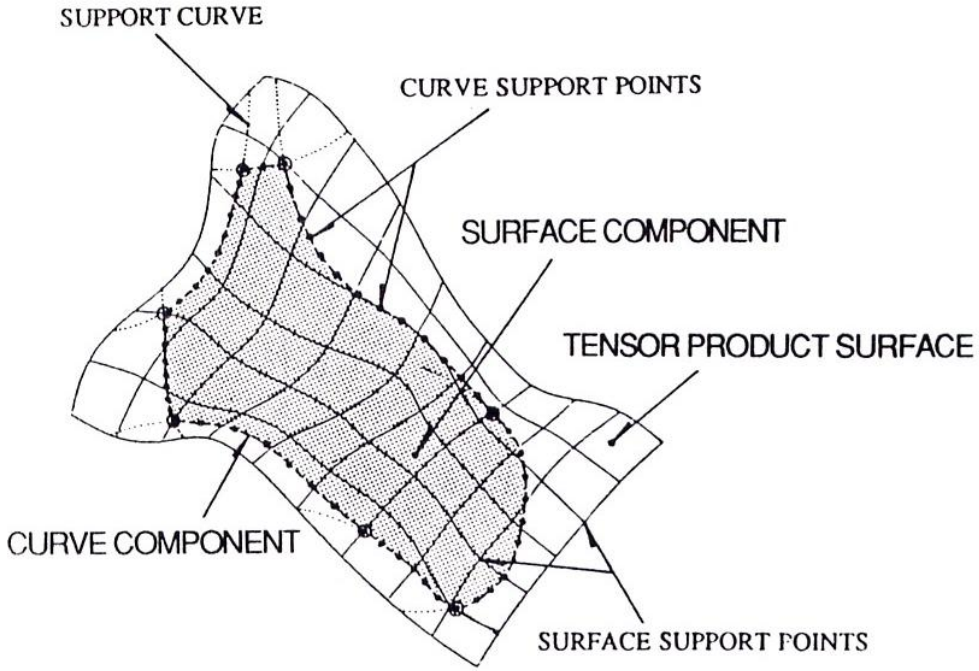
Decomposition of the boundary of a three dimensional domain into its surfaces and curve components

Geometry Modelling

In addition, boundary curves and surfaces are oriented. This is important in the generating process as it is used to define the location of the region to be discretised. The orientation of a boundary surface is defined by the direction of the inward normal. The orientation of the boundary curves is defined with respect to the boundary surfaces that contain them. Each boundary curve will be common to two boundary surfaces and will have opposite orientations with respect to each of them.



An example of the approximated geometry of the surfaces component and its curve components is here presented.



Curve Representation

A support curve may be described by a piecewise parametric representation.
In those representation, the curve is subdivided into n_a arcs, and the position vector r of a generic point on each arc is expressed as a function of a single real parameter u, which by convention varies into the interval [0,1].

$$\mathbf{r} \equiv [x, y, z]$$

In general, this function is represented by a polynomial whose rank may change from arc to arc and which can be expressed as

$$\mathbf{r} = \mathbf{r}(u) = \mathbf{a}_0 + \mathbf{a}_1 u + \mathbf{a}_2 u^2 + \ldots + \mathbf{a}_n u^n$$

where each $\mathbf{a}_k$ is a vector formed by three coefficients

$$\mathbf{a}_k = \left[a_k^x, a_k^y, a_k^z\right]$$

which represents the components of $\mathbf{a}_k$ with respect to a cartesian reference system (x,y,z), whereas n is the rank of the polynomial.
The position vector can be rewritten as follows

$$\mathbf{r}(u) = \mathbf{AU}$$

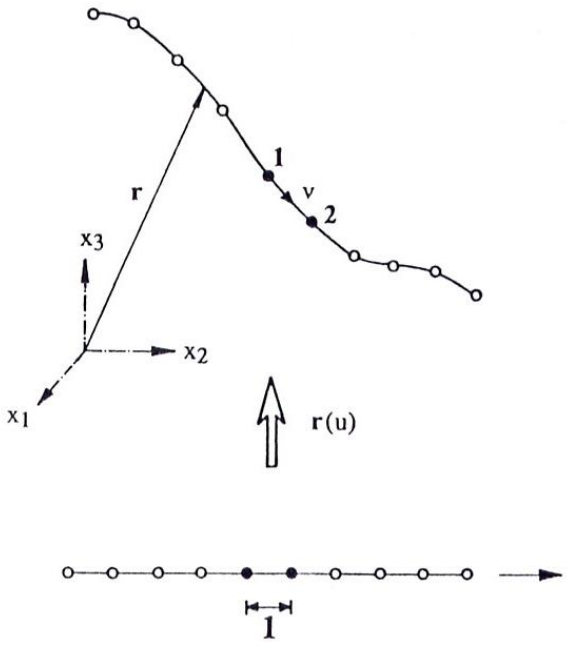
where

$$\mathbf{A} = [\mathbf{a}_0, \mathbf{a}_1, \mathbf{a}_2, \ldots, \mathbf{a}_n]$$

and

$$\mathbf{U} = \left[1, u, u^2, \ldots, u^n\right]^T$$

A point on the curve can then be identified by the number of the arc on which it is lying and the value of the parametric coordinates u, which is usually called local parametric coordinate.



Surface Representation

The surfaces are represented by using a similar scheme. They are subdivided into patches which forms a regular grid on the so-called parametric plane. On this plane, we can define with respect to each patch the local parametric coordinates u and v with varies on the interval [0,1].

The position r of a node on the surface can then be expressed as a polynomial expansion in u and v on each patch :

$$\mathbf{r}(u,v) = \mathbf{a}_{0,0} + \mathbf{a}_{1,0}u + \mathbf{a}_{0,1}v + \mathbf{a}_{2,0}u^2 + \mathbf{a}_{1,1}uv + \ldots + \mathbf{a}_{n,m}u^n v^m$$

The total rank of the polynomial is nxm, whereas n and m represent the rank of the polynomial with respect to the cartesian reference system (x,y,z):

$$\mathbf{a}_{k,l} = \left[a^x_{k,l}, a^y_{k,l}, a^z_{k,l}\right]$$

The position vector can be rewritten in the following form

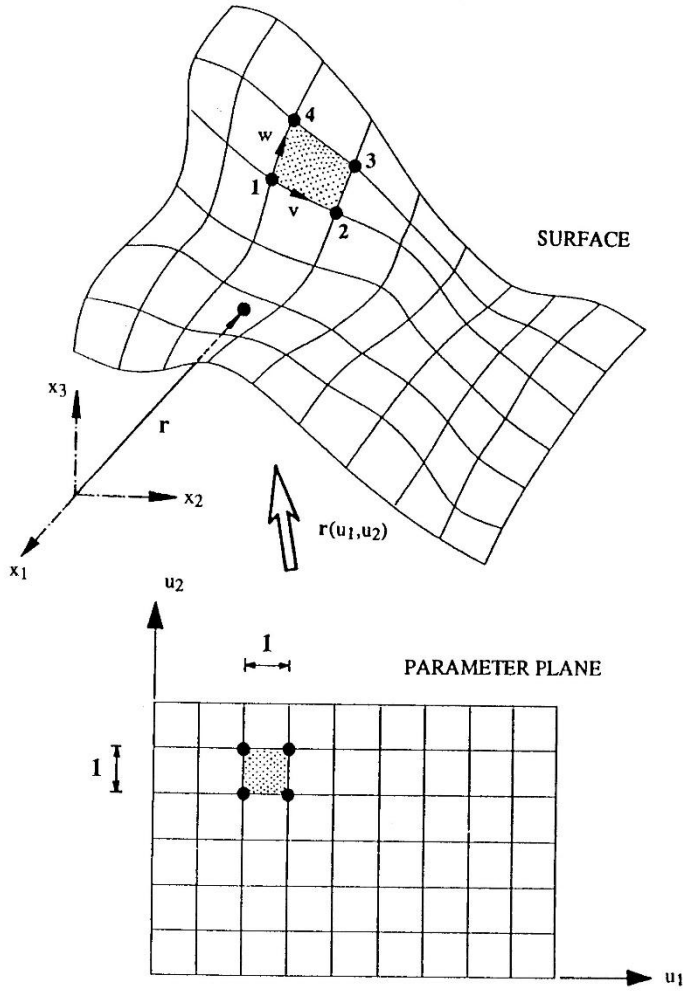
$$\mathbf{r}(u,v) = \mathbf{UAV}$$

where

$$\mathbf{A} = \left[\mathbf{a}_{k,l}\right] \quad , \quad \begin{matrix} k = (0,\ldots,n) \\ l = (0,\ldots,m) \end{matrix}$$

The quantities U and V are expressed as

$$\mathbf{U} = \left[1, u, u^2, \ldots, u^n\right]^T \quad , \quad \mathbf{V} = \left[1, v, v^2, \ldots, v^m\right]^T$$



Structured Grid generation techniques

- Conformal mapping:

  Based on complex variable theory, which is limited to two dimensions.

- Algebraic methods:

  1. 1D: polynomials, Trigonometric functions, Logarithmic functions
  2. 2D: Orthogonal one-dimensional transformation, normalizing transformation, connection functions
  3. 3D: Stacked two-dimensional transformations, elliptical boundaries

- Differential equation methods:

  Step 1: Determine the grid point distribution on the boundaries of the physical space.

  Step 2: Assume the interior grid point is specified by a differential equation that satisfies the grid point distributions specified on the boundaries and yields an acceptable interior grid point distribution.

Mappings and Jacobians

Consider the continuous mapping of the form $x = x(\xi)$. If a function f exists for which

$$f = f(x)$$

then

$$\frac{\partial f}{\partial \xi} = \frac{\partial f}{\partial x}\frac{\partial x}{\partial \xi} = J\frac{\partial f}{\partial x} \quad , \quad J = \frac{\partial x}{\partial \xi}$$

For any differential function $f(x)$, it is clear that the change of f with respect to ξ is equal to the change of f with respect to x multiplied by a scaling parameter J which is the change of x with respect to the new variable ξ .
Using this equation to solve for $\partial f / \partial x$ gives

$$\frac{\partial f}{\partial x} = \frac{1}{J}\frac{\partial f}{\partial \xi}$$

The term $J = \partial x / \partial \xi$ is called the jacobian of the mapping and can be seen to be an effective scaling introduced through the change of variables. The quantity $\left|\partial x / \partial \xi\right|$ is the ratio of the length in the physical space to the length in the logical space. If the Jacobian vanishes within the domain, then the transformation is invalid

In two dimensions, let consider a continuous transformation of the form

$$x = x(\xi,\eta) \ , \ y = y(\xi,\eta)$$

If we consider a function f with

$$f = f(x,y)$$

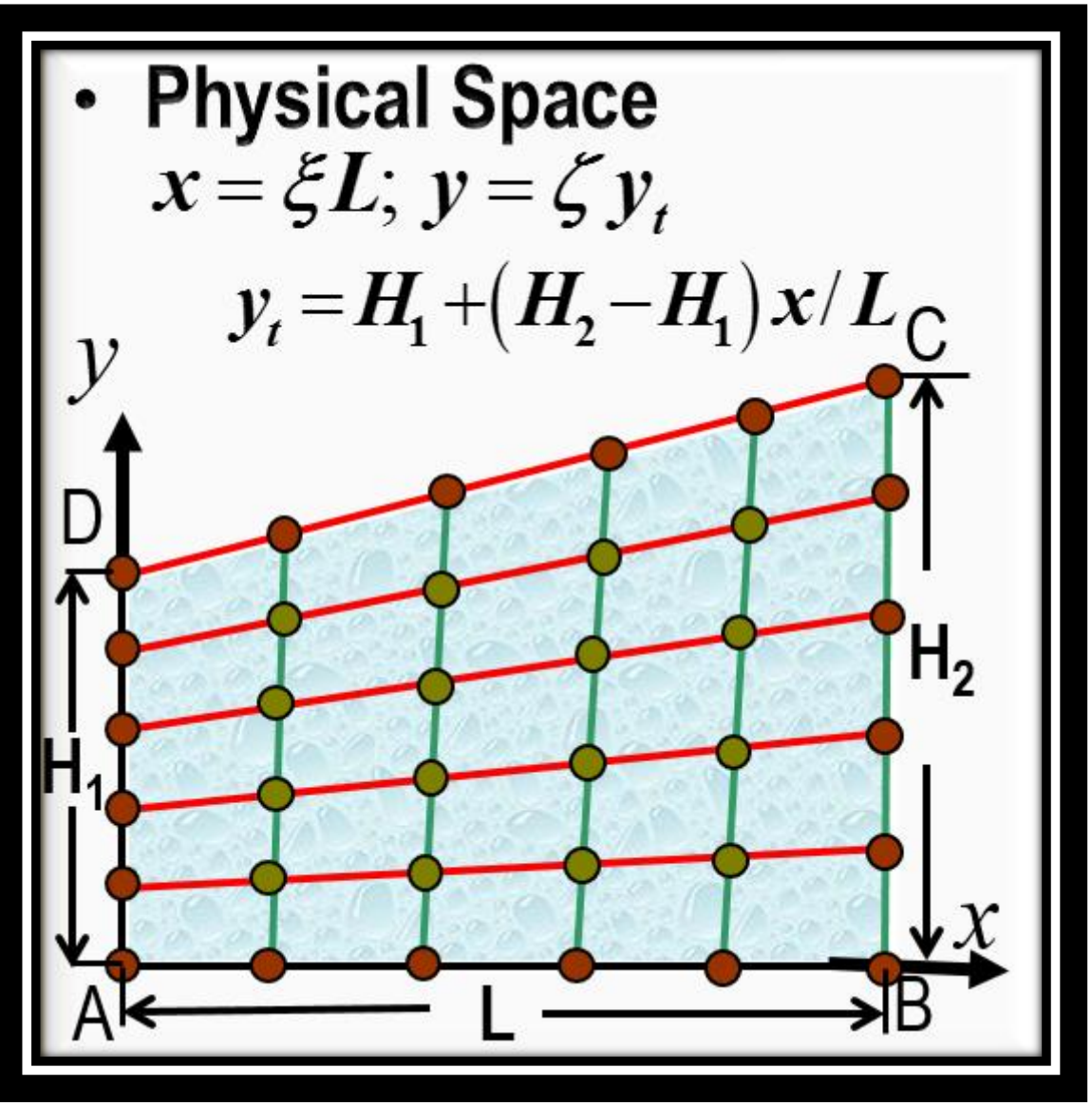
then

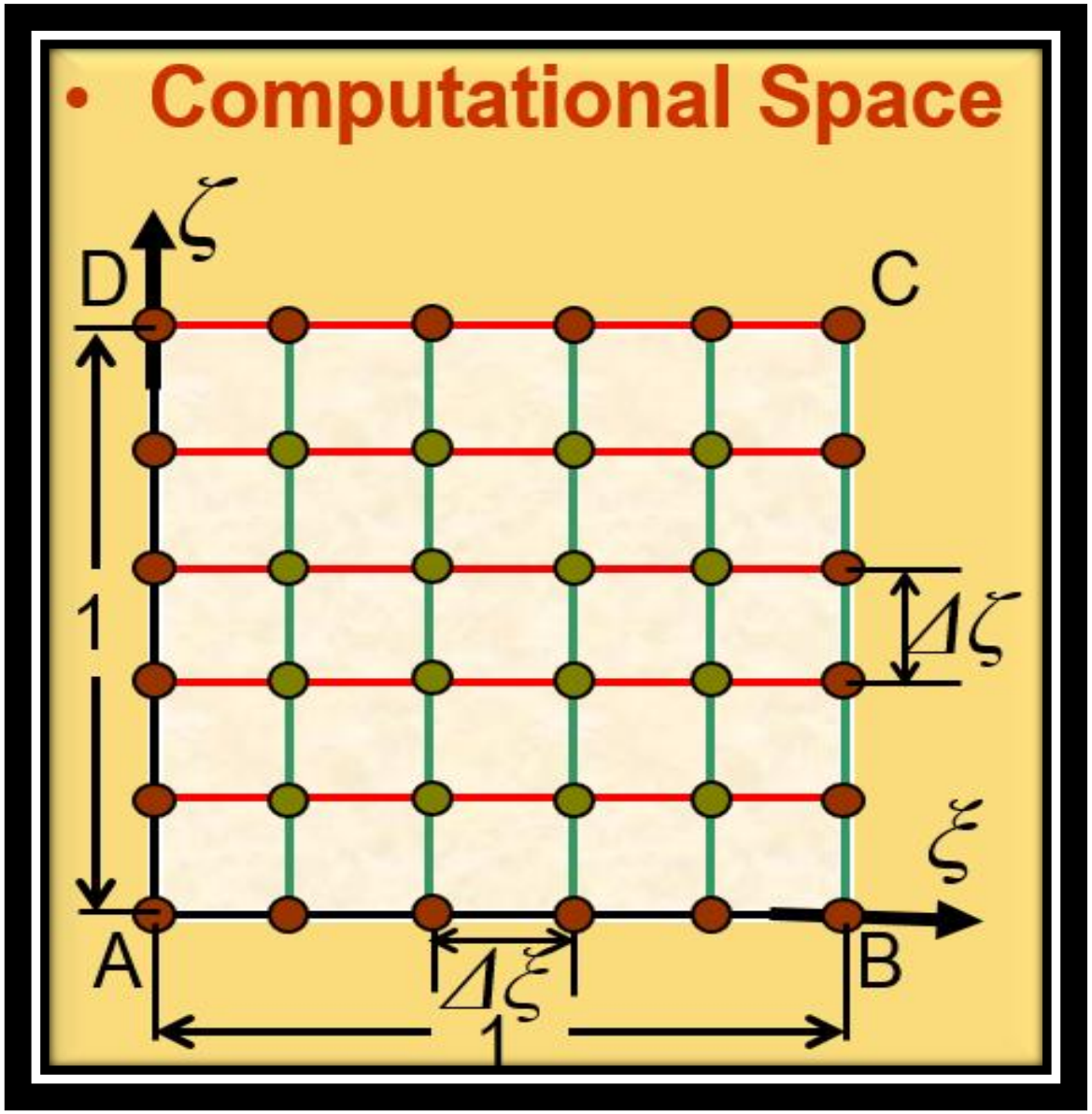
$$\frac{\partial f}{\partial \xi} = \frac{\partial f}{\partial x}\frac{\partial x}{\partial \xi} + \frac{\partial f}{\partial y}\frac{\partial y}{\partial \xi}$$

$$\frac{\partial f}{\partial \eta} = \frac{\partial f}{\partial x}\frac{\partial x}{\partial \eta} + \frac{\partial f}{\partial y}\frac{\partial y}{\partial \eta}$$

Algebraic Methods

Transformation from Physical to computational domain





Algebraic Grid Generation: Clustering at y=0

- **Transformation:**

$$\xi = x / L$$

$$\zeta = 1 - \frac{\ln\left\{\beta + 1 - (y/H)/\left[\beta - 1 + (y/H)\right]\right\}}{\ln\left[(\beta+1)/(\beta-1)\right]}$$

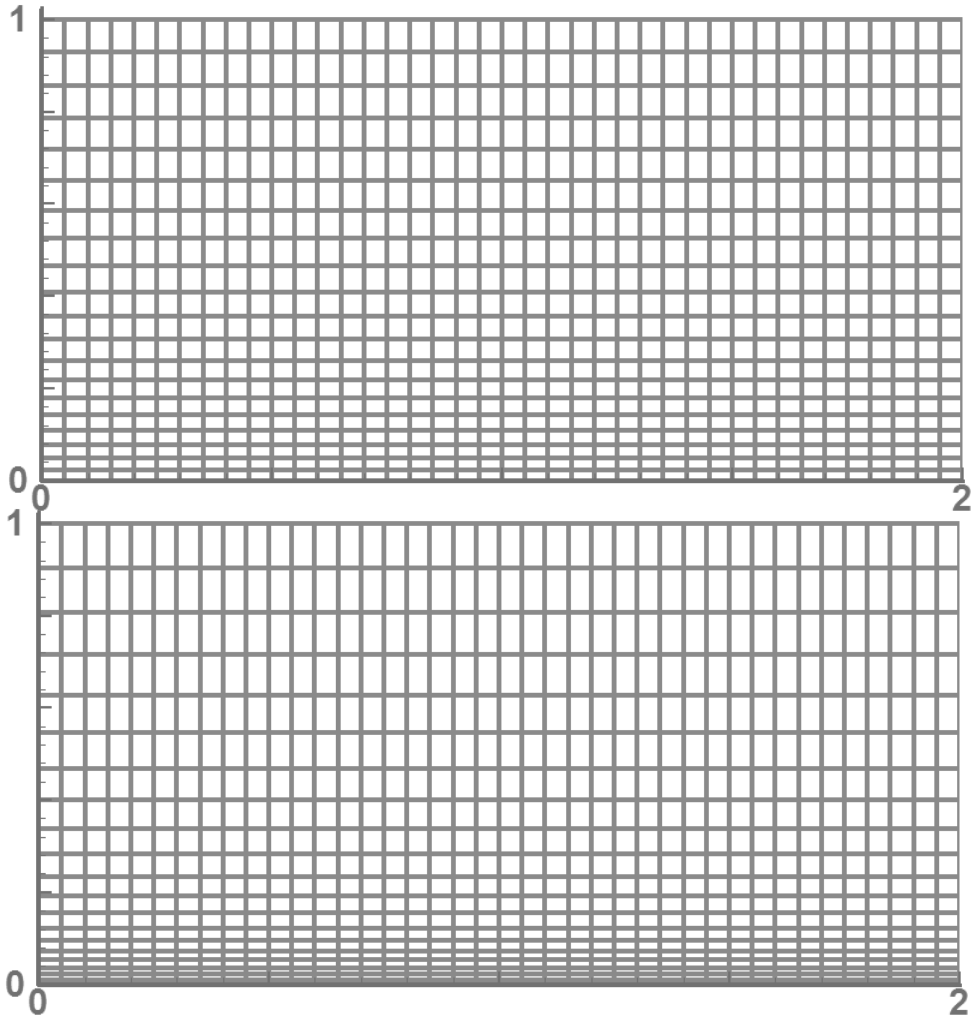
- **Inverse Transformation:**

$$x = L\,\xi$$

$$y = H\frac{(\beta+1) - (\beta-1)\left\{\left[(\beta+1)/(\beta-1)\right]^{1-\zeta}\right\}}{\left[(\beta+1)/(\beta-1)\right]^{1-\zeta} + 1}$$

Algebraic Grid Generation: Clustering at y=0

- β=1.2
- β=1.05



L=2, H=1
imax=21 & jmax=41

Algebraic Grid Generation: Equal Clustering at y=0 & y=H

- **Transformation:**

$$\xi = x / L$$

$$2\zeta = 1 + \frac{\ln\left\{\left[\beta + 2y/H - 1\right]/\left[\beta - 2y/H + 1\right]\right\}}{\ln\left[\left(\beta+1\right)/\left(\beta-1\right)\right]}$$

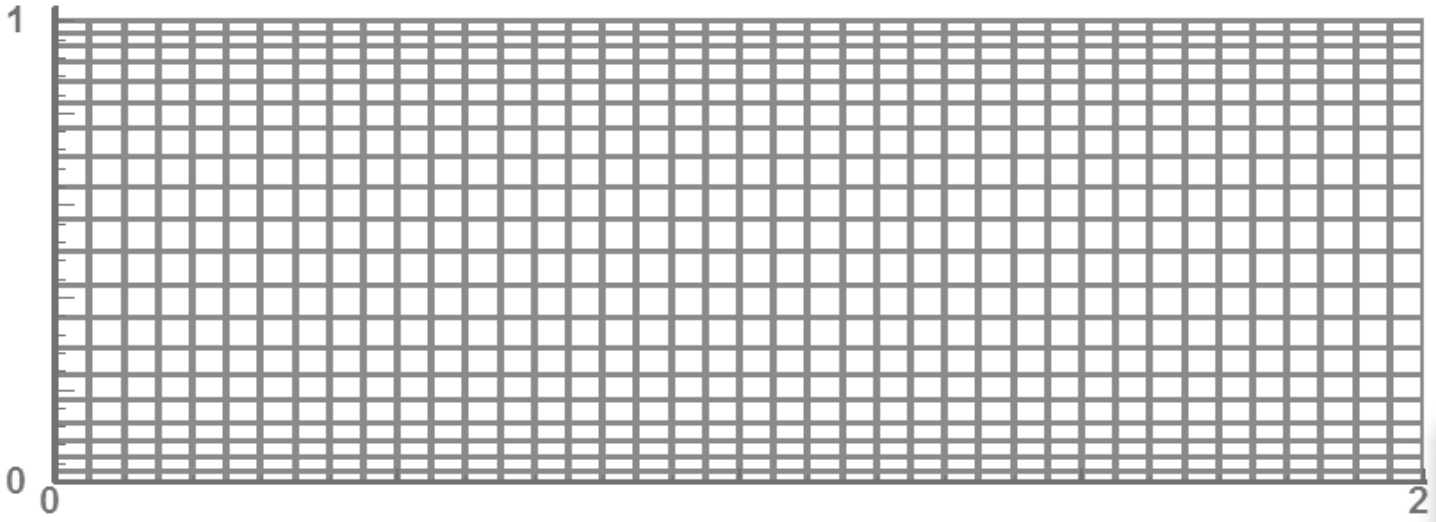
- **Inverse Transformation:**

$$x = L\ \xi$$

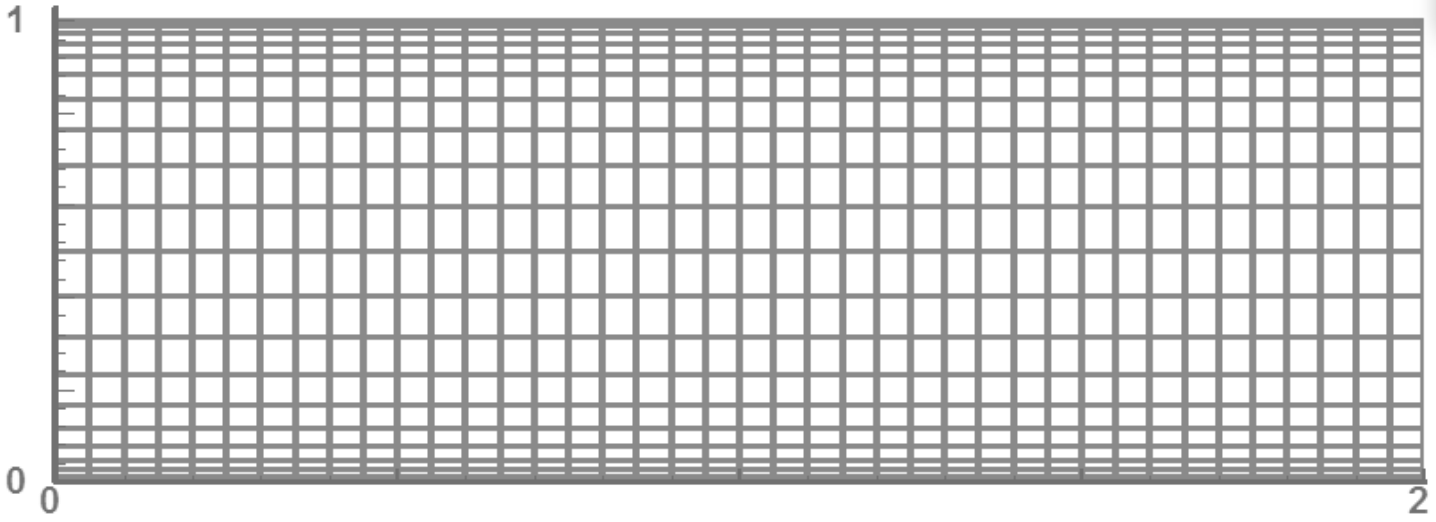
$$y = H\,\frac{\left(1+\beta\right)\left[\left(\beta+1\right)/\left(\beta-1\right)\right]^{\left(2\zeta-1\right)} + 1 - \beta}{2\left\{1+\left[\left(\beta+1\right)/\left(\beta-1\right)\right]^{\left(2\zeta-1\right)}\right\}}$$

Algebraic Grid Generation: Equal Clustering at y=0 & y=H

- **β=1.2**



- **β=1.05**



L=2, H=1
imax=21 & jmax=41

Algebraic Grid Generation
Clustering in the Interior of the Domain

- **Transformation:**

$$\xi = x/L$$

$$\zeta = A + \frac{1}{\beta}\sinh^{-1}\left[\left(\frac{y}{D} - 1\right)\sinh(\beta A)\right] \quad \left\{A = \frac{1}{2\beta}\ln\left[\frac{1+\left(e^{\beta}-1\right)(D/H)}{1+\left(e^{-\beta}-1\right)(D/H)}\right]\right\}$$

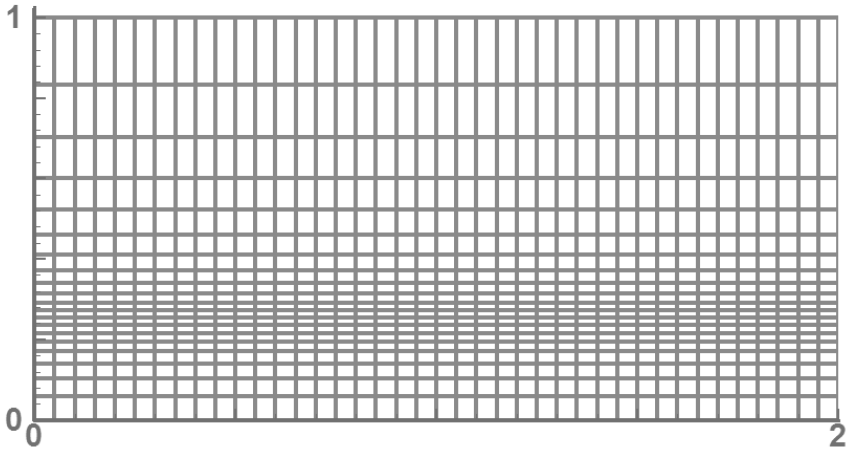
- **Inverse Transformation:**

$$x = L\ \xi$$

$$y = D\left\{1 + \frac{\sinh\left[\beta\left(\zeta - A\right)\right]}{\sinh\left(\beta A\right)}\right\}$$

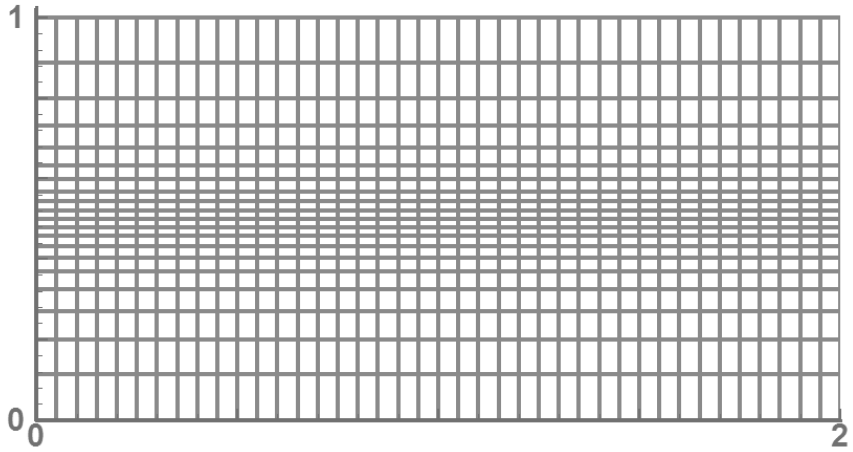
Algebraic Grid Generation: Clustering in the interior of the Domain

- **D=H/4 β=5**

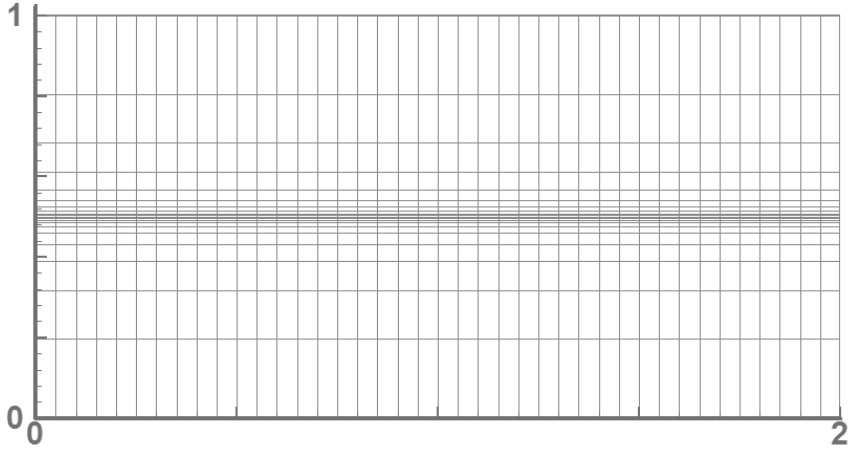


L=2, H=1
imax=21 & jmax=41

- **D=H/2;β=5**

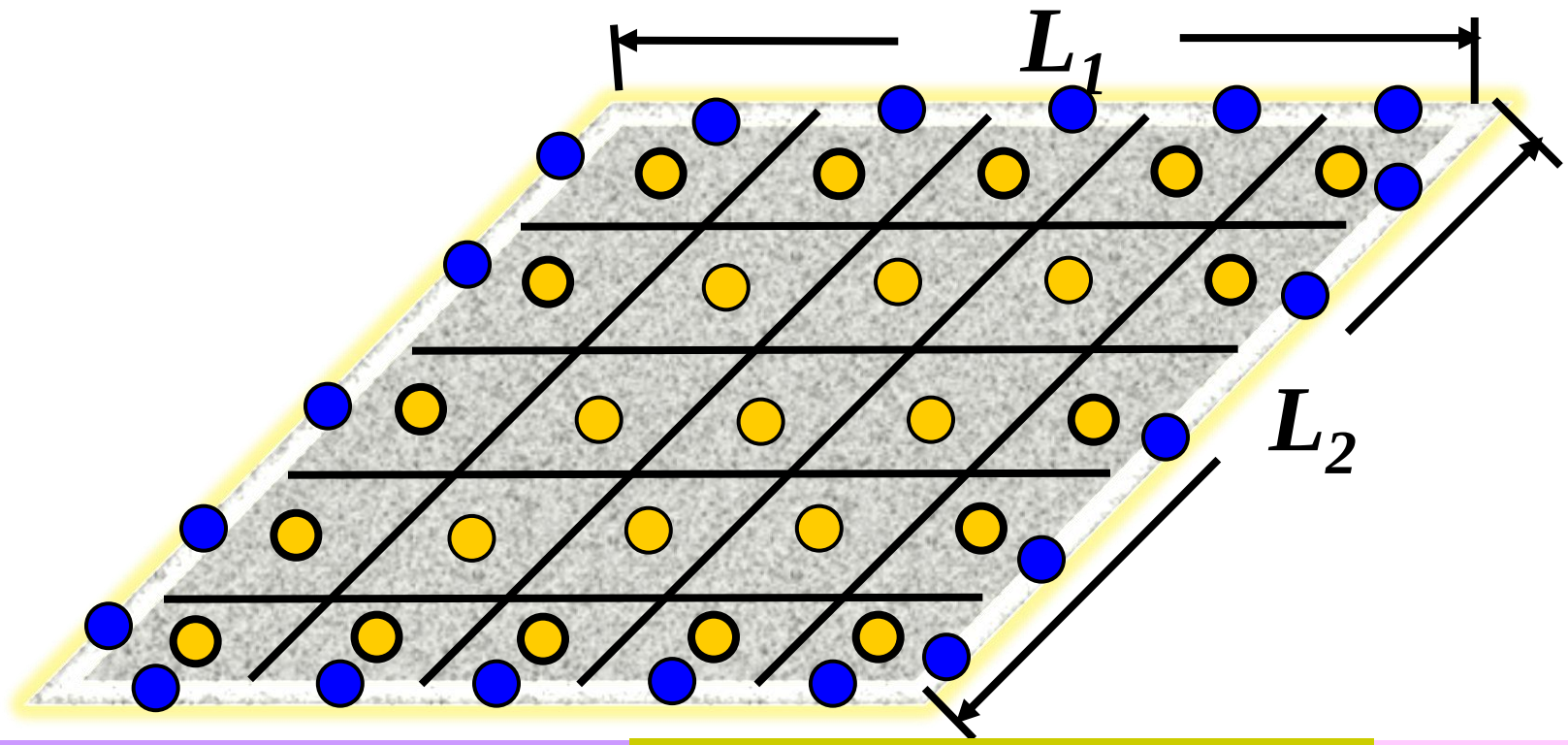


- **D=H/2;β=10**



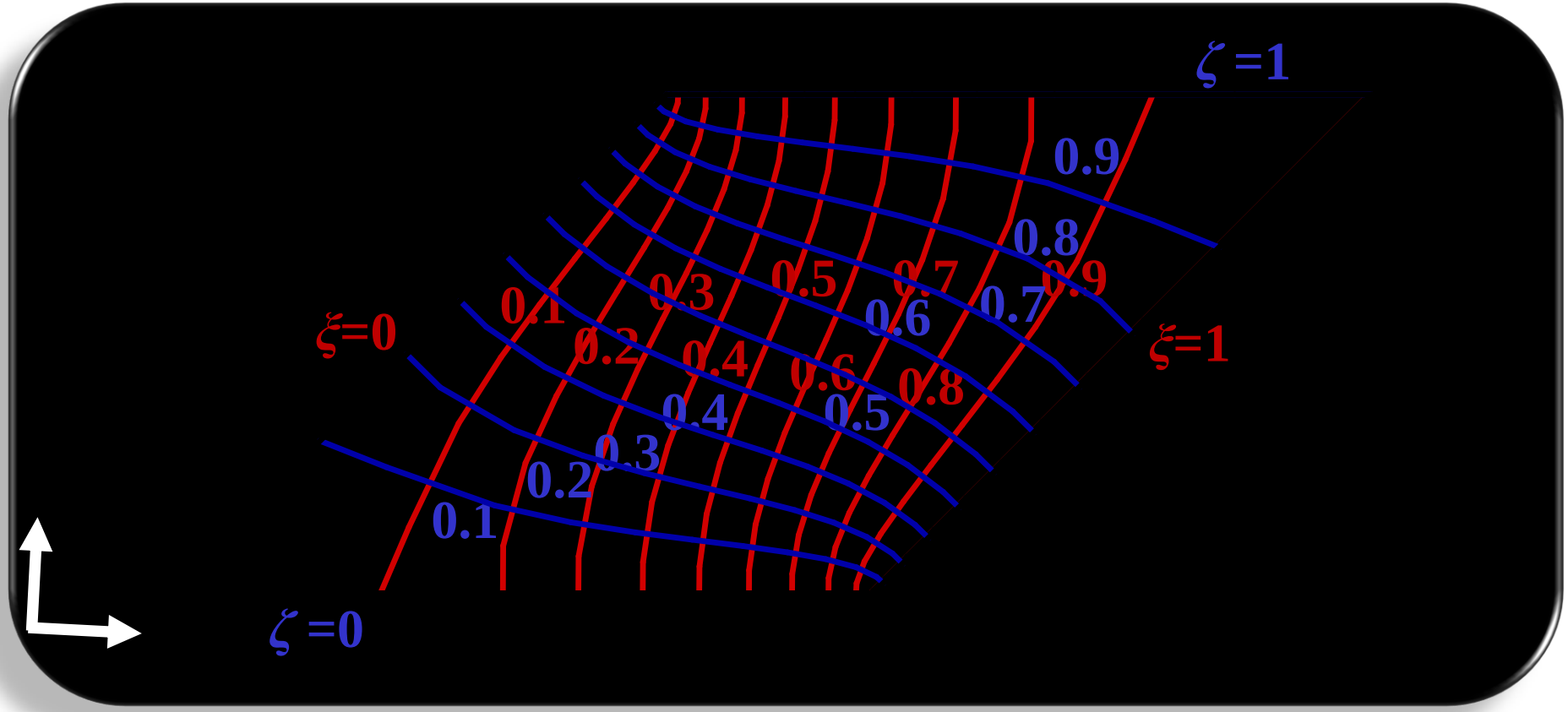
Elliptic Partial Differential Equations Methods

Grid Generation: Finite CV



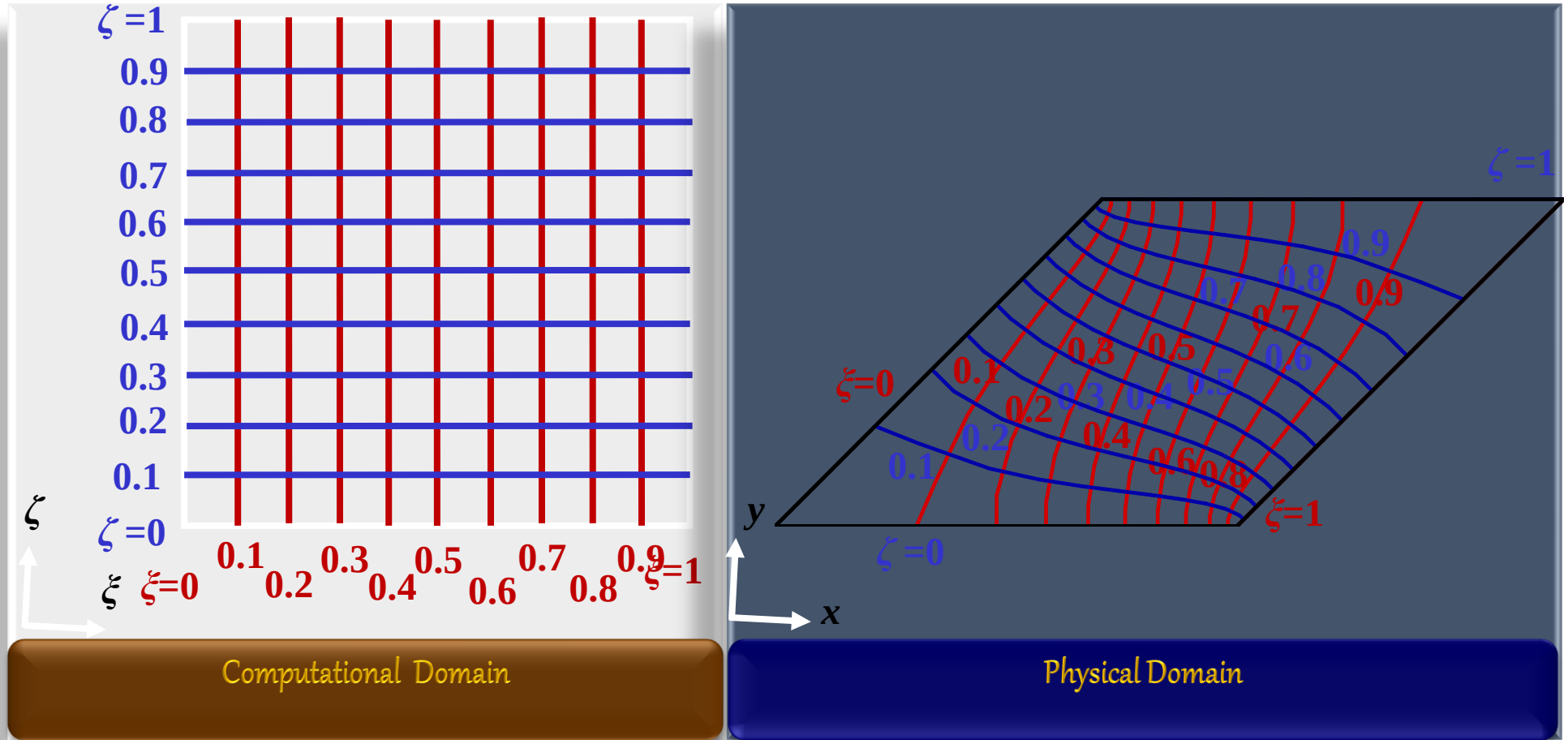
Boundary CVs: ●	Border CVs: ○	Interior CVs: ○
i=1 & imax; j=2 to jmax-1	*i=2 & imax-1; j=2 to jmax-1*	*i=3 to imax-2;*
j=1 & jmax; i=2 to imax-1	*j=2 & jmax-1; i=2 to imax-1*	*j=3 to jmax-2*

Grid Generation in the Physical Domain: Intersection of the Isotherms



Elliptic Grid Generation: Inverse Transformation

- The governing partial differential equations for x=f(ξ,ζ) and y=f(ξ,ζ) is solved in square computational domain using finite difference method with boundary conditions in terms for x and y.



Elliptic Grid Generation: Laplace Equation

$$\nabla^2 \phi = 0$$

$$\frac{\partial^2 \xi}{\partial x^2} + \frac{\partial^2 \xi}{\partial y^2} = 0 \qquad \frac{\partial^2 \zeta}{\partial x^2} + \frac{\partial^2 \zeta}{\partial y^2} = 0$$

- Satisfies both the maximum and minimum principles
- Excellent smoothness property of the Laplacian equation, the discontinuity in the prescribed boundary data is eliminated at a small distance inside the domain.
- Grid orthogonality can also be achieved

Elliptic Grid Generation:
Inverse-Transformation of Laplace-Equation

- G.E:

$$A\frac{\partial^2 x}{\partial \xi^2} - 2B\frac{\partial^2 x}{\partial \xi \partial \zeta} + C\frac{\partial^2 x}{\partial \zeta^2} = 0$$

where

$$A\frac{\partial^2 y}{\partial \xi^2} - 2B\frac{\partial^2 y}{\partial \xi \partial \zeta} + C\frac{\partial^2 y}{\partial \zeta^2} = 0$$

$$A = \left(\frac{\partial x}{\partial \zeta}\right)^2 + \left(\frac{\partial y}{\partial \zeta}\right)^2 ; C = \left(\frac{\partial x}{\partial \xi}\right)^2 + \left(\frac{\partial y}{\partial \xi}\right)^2$$

$$B = \left(\frac{\partial x}{\partial \xi}\right)\left(\frac{\partial x}{\partial \zeta}\right) + \left(\frac{\partial y}{\partial \xi}\right)\left(\frac{\partial y}{\partial \zeta}\right)$$

Grid Orthogonality at the Domain Boundary

- Condition

$$B = \left(\frac{\partial x}{\partial \xi}\right)\left(\frac{\partial x}{\partial \zeta}\right) + \left(\frac{\partial y}{\partial \xi}\right)\left(\frac{\partial y}{\partial \zeta}\right) = 0$$

- Top/Bottom Boundary: Horizontal Lines

$$\frac{\partial x}{\partial \xi} = \text{constant}; \quad \frac{\partial y}{\partial \xi} = 0 \Rightarrow \frac{\partial x}{\partial \zeta} = 0$$

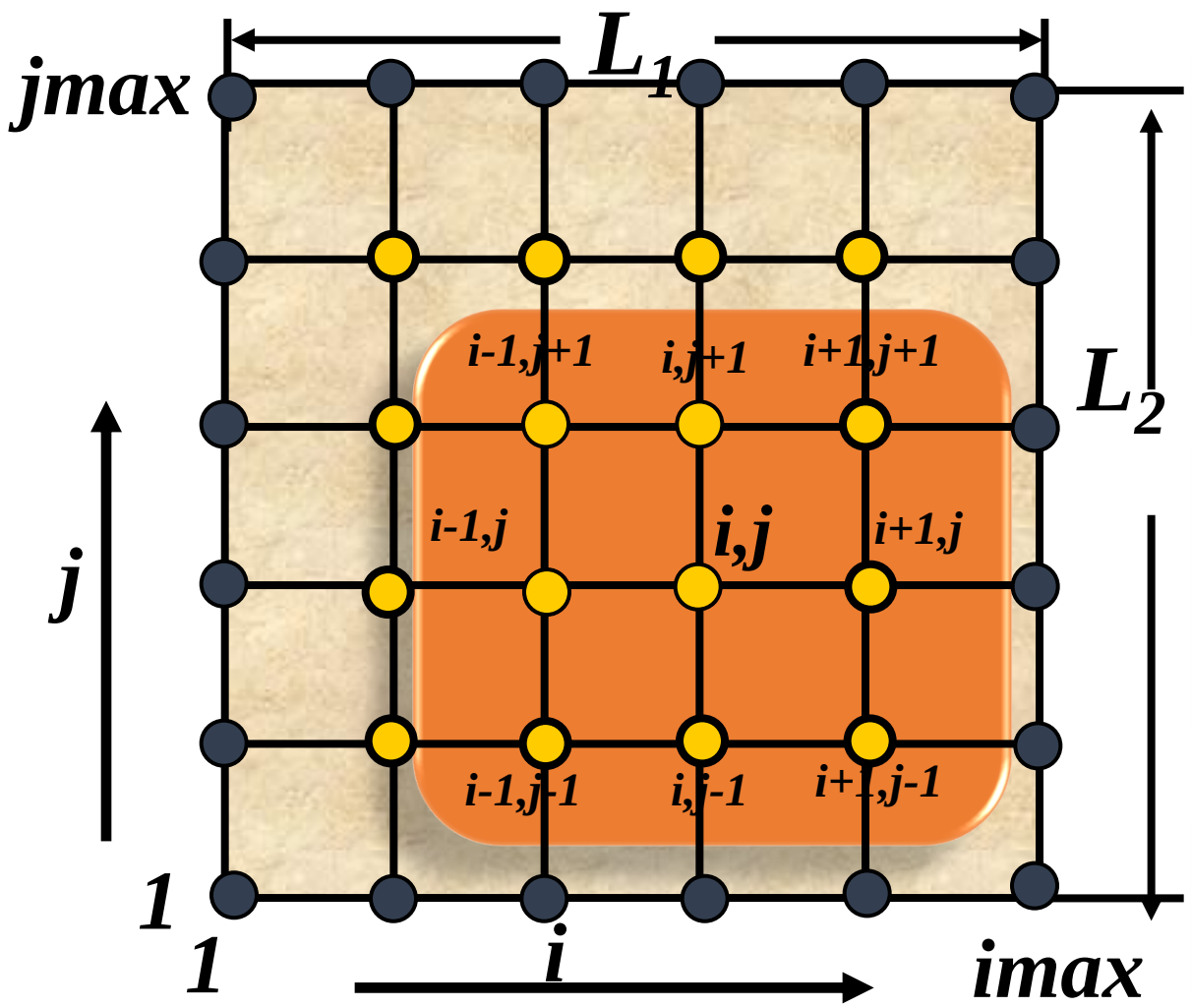
- Left/Right Boundary: Inclined Lines

$$\frac{\partial x}{\partial \zeta} = k; \quad \frac{\partial y}{\partial \zeta} = k \Rightarrow \frac{\partial x}{\partial \xi} + \frac{\partial y}{\partial \xi} = 0$$

PDE's Grid Generation

- A system of PDEs is solved for the location of grid points in the physical space
- The computational space/domain is a square shaped with uniform grid spacing.
- Types
  - Elliptic
  - Parabolic
  - Hyperbolic

Discrete Representation of Domain: Grid Generation



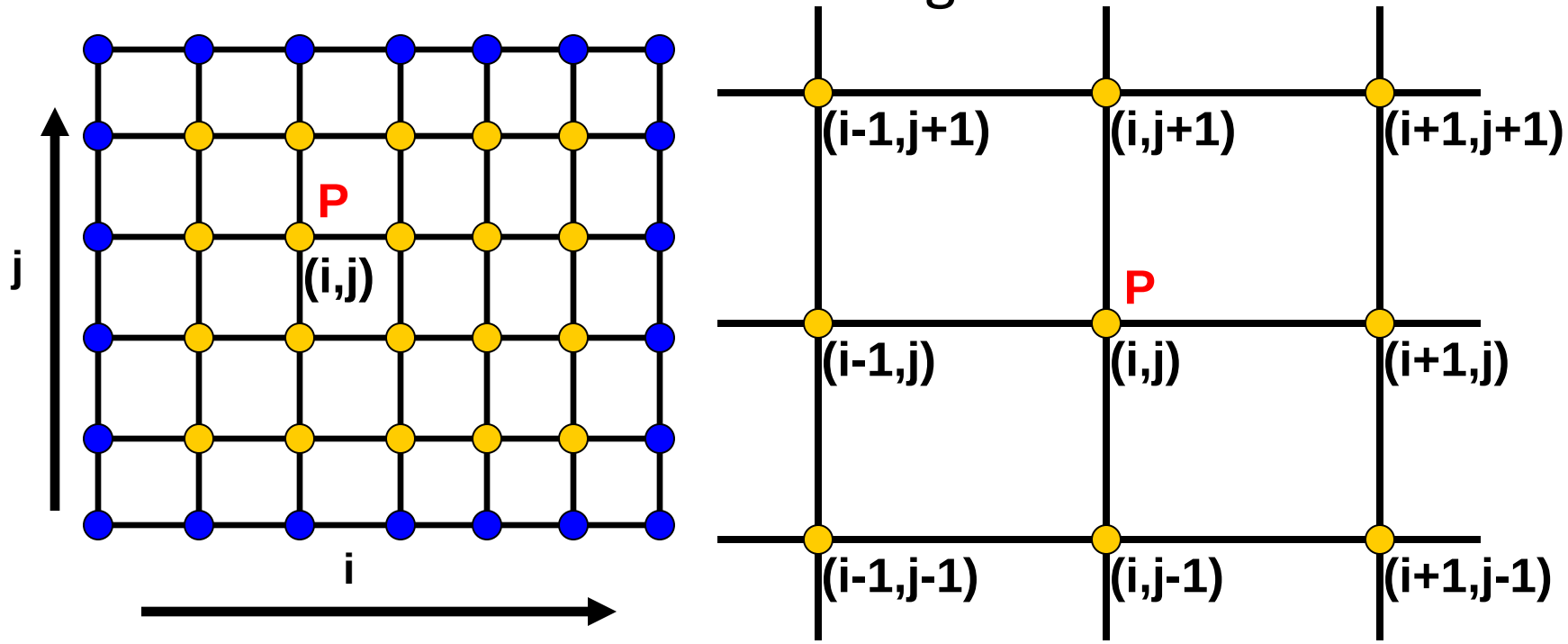
Boundary Grid Point:
i=1 & imax;
⇨ *j=2 to jmax-1*
j=1 & jmax;
⇨ *i=2 to imax-1*

Border Grid Point:
i=2 & imax-1;
⇨ *j=2 to jmax-1*
j=2 & jmax-1;
⇨ *i=2 to imax-1*

Interior Grid Point:
i=2 to imax-1;
j=2 to jmax-1

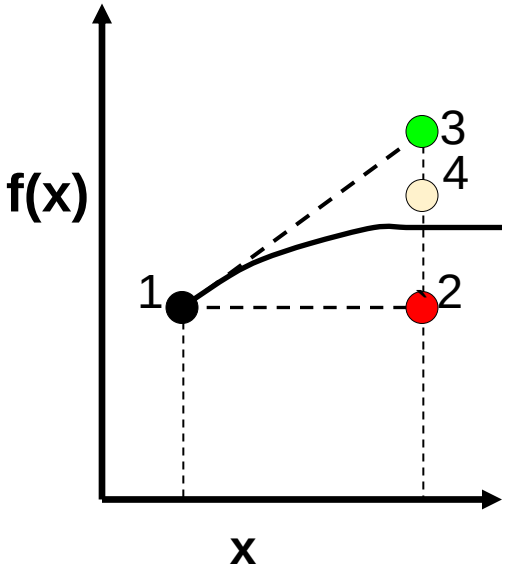
FDM: Grid Generation

- A Representative Grid Points and its Neighbors



Taylor Series

- Example:



$$f(x)=\sin 2\pi x$$

$$x=0.2,\Delta x=0.02 \Rightarrow f(x+\Delta x)=0.9823$$

$$f(x+\Delta x)=f(x)+\frac{\partial f(x)}{\partial x}\Delta x+\frac{\partial^2 f(x)}{\partial x^2}\frac{\Delta x^2}{\lfloor 2}+\cdots$$

$$f_2=f_1=0.9511\ (3.176\%\ \text{Error})$$

$$f_3=0.9899\ (0.775\%\ \text{Error})$$

$$f_4=0.9824\ (0.01\%\ \text{Error})$$

FDM: Taylor Series Expansion

$$\phi_{i+1,j} = \phi_{i,j} + \left(\frac{\partial \phi}{\partial x}\right)_{i,j} \Delta x + \left(\frac{\partial^2 \phi}{\partial x^2}\right)_{i,j} \frac{\Delta x^2}{\lfloor 2} + \left(\frac{\partial^3 \phi}{\partial x^3}\right)_{i,j} \frac{\Delta x^3}{\lfloor 3} + \cdots$$

$$\phi_{i-1,j} = \phi_{i,j} - \left(\frac{\partial \phi}{\partial x}\right)_{i,j} \Delta x + \left(\frac{\partial^2 \phi}{\partial x^2}\right)_{i,j} \frac{\Delta x^2}{\lfloor 2} - \left(\frac{\partial^3 \phi}{\partial x^3}\right)_{i,j} \frac{\Delta x^3}{\lfloor 3} + \cdots$$

FDM: I Order Forward Difference

$$\left(\frac{\partial \phi}{\partial x}\right)_{i,j} = \left\{\frac{\phi_{i+1,j} - \phi_{i,j}}{\Delta x}\right\}$$

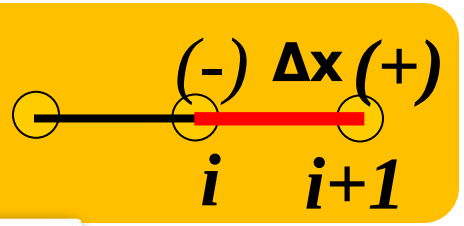
Finite Difference Quotient

Truncation Error

$$-\left[\left(\frac{\partial^2 \phi}{\partial x^2}\right)_{i,j} \frac{\Delta x}{\lfloor 2} + \left(\frac{\partial^3 \phi}{\partial x^3}\right)_{i,j} \frac{\Delta x^2}{\lfloor 3} + \cdots\right]$$

$$\left(\frac{\partial \phi}{\partial x}\right)_{i,j} = \left\{\frac{\phi_{i+1,j} - \phi_{i,j}}{\Delta x}\right\} + O(\Delta x)$$

I Order Accurate



FDM: I Order Backward Difference

$$\left(\frac{\partial \phi}{\partial x}\right)_{i,j} = \left\{\frac{\phi_{i,j} - \phi_{i-1,j}}{\Delta x}\right\}$$

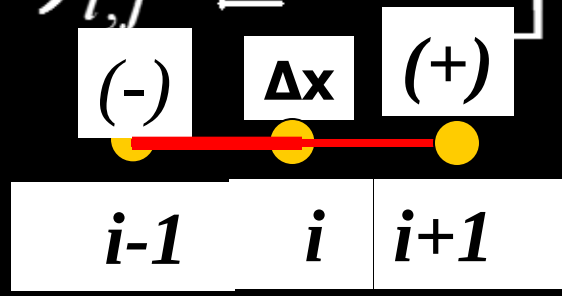
Finite Difference Quotient

$$+\left[\left(\frac{\partial^2 \phi}{\partial x^2}\right)_{i,j} \frac{\Delta x}{\lfloor 2} - \left(\frac{\partial^3 \phi}{\partial x^3}\right)_{i,j} \frac{\Delta x^2}{\lfloor 3} + \cdots\right]$$

Truncation Error

$$\left(\frac{\partial \phi}{\partial x}\right)_{i,j} = \left\{\frac{\phi_{i,j} - \phi_{i-1,j}}{\Delta x}\right\} + O(\Delta x)$$

I Order Accurate



FDM: II Order Central Difference

$$\left(\frac{\partial \phi}{\partial x}\right)_{i,j} = \left\{\frac{\phi_{i+1,j} - \phi_{i-1,j}}{2\Delta x}\right\}$$

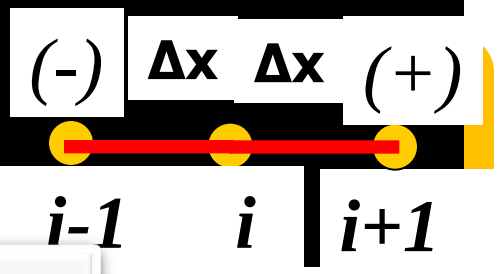
Finite Difference Quotient

Truncation Error

$$-\left[\left(\frac{\partial^3 \phi}{\partial x^3}\right)_{i,j} \frac{\Delta x^2}{\lfloor 3} + \left(\frac{\partial^5 \phi}{\partial x^5}\right)_{i,j} \frac{\Delta x^4}{\lfloor 5} \cdots\right]$$

$$\left(\frac{\partial \phi}{\partial x}\right)_{i,j} = \left\{\frac{\phi_{i+1,j} - \phi_{i-1,j}}{2\Delta x}\right\} + O\left(\Delta x^2\right)$$

II Order Accurate

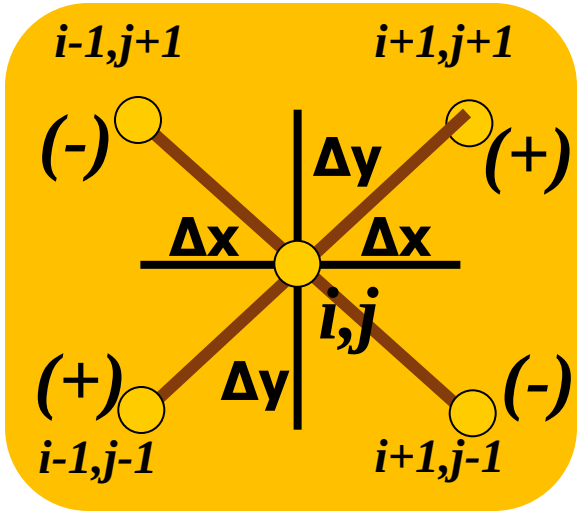


FDM: II Order Central Second Cross Derivative Difference

$$\left(\frac{\partial^2\phi}{\partial x\partial y}\right)_{i,j} = \left\{\frac{(\partial\phi/\partial y)_{i+1,j} - (\partial\phi/\partial y)_{i-1,j}}{2\Delta x}\right\}$$

$$\Rightarrow \frac{\left(\frac{\phi_{i+1,j+1} - \phi_{i+1,j-1}}{2\Delta y}\right) - \left(\frac{\phi_{i-1,j+1} - \phi_{i-1,j-1}}{2\Delta y}\right)}{2\Delta x}$$

$$\Rightarrow \frac{\phi_{i+1,j+1} + \phi_{i-1,j-1} - \phi_{i+1,j-1} - \phi_{i-1,j+1}}{4\Delta x\Delta y}$$



Elliptic Grid Generation

$$A\frac{\partial^2 x}{\partial \xi^2} - 2B\frac{\partial^2 x}{\partial \xi \partial \zeta} + C\frac{\partial^2 x}{\partial \zeta^2} = 0$$

$$A\frac{\partial^2 y}{\partial \xi^2} - 2B\frac{\partial^2 y}{\partial \xi \partial \zeta} + C\frac{\partial^2 y}{\partial \zeta^2} = 0$$

$$A = \left(\frac{\partial x}{\partial \zeta}\right)^2 + \left(\frac{\partial y}{\partial \zeta}\right)^2 ; C = \left(\frac{\partial x}{\partial \xi}\right)^2 + \left(\frac{\partial y}{\partial \xi}\right)^2$$

$$B = \left(\frac{\partial x}{\partial \xi}\right)\left(\frac{\partial x}{\partial \zeta}\right) + \left(\frac{\partial y}{\partial \xi}\right)\left(\frac{\partial y}{\partial \zeta}\right)$$

Finite Difference Discretization of Laplace Equation

$$A_{i,j}\frac{x_{i+1,j}-2x_{i,j}+x_{i-1,j}}{\Delta\xi^2}$$

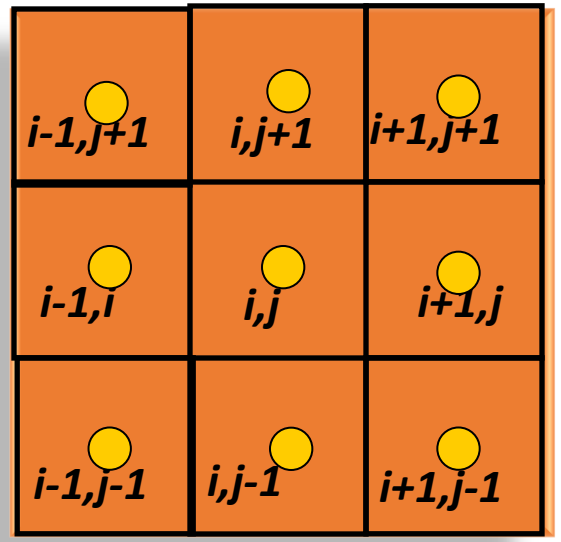
$$-2B_{i,j}\frac{x_{i+1,j+1}+x_{i-1,j-1}-x_{i-1,j+1}-x_{i+1,j-1}}{4\Delta\xi\Delta\zeta}$$

$$+C_{i,j}\frac{x_{i,j+1}-2x_{i,j}+x_{i,j-1}}{\Delta\zeta^2}=0$$

$D_{i,j}$

Taking $\Delta\xi=\Delta\zeta$,

$$x_{i,j}=\frac{A_{i,j}\left(x_{i+1,j}+x_{i-1,j}\right)+C_{i,j}\left(x_{i,j+1}+x_{i,j-1}\right)+D_{i,j}}{2\left(A_{i,j}+C_{i,j}\right)}$$

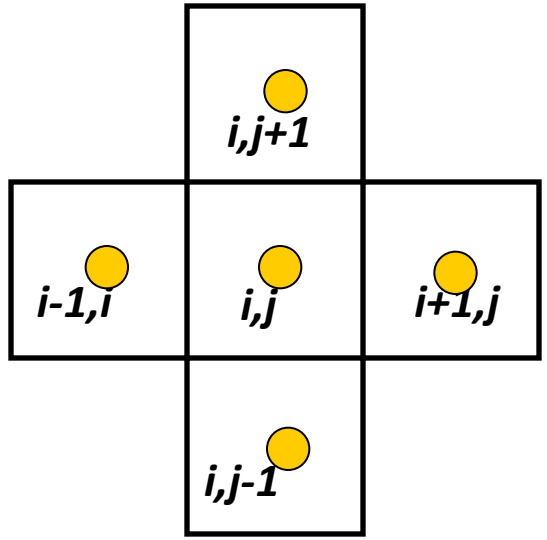


Finite Difference Discretization of Laplace Equation

$$A_{i,j} = \left(\frac{x_{i,j+1} - x_{i,j-1}}{2\Delta\zeta} \right)^2 + \left(\frac{y_{i,j+1} - y_{i,j-1}}{2\Delta\zeta} \right)^2 ;$$

$$B_{i,j} = \left(\frac{x_{i+1,j} - x_{i-1,j}}{2\Delta\xi} \right) \left(\frac{x_{i,j+1} - x_{i,j-1}}{2\Delta\zeta} \right) + \left(\frac{y_{i+1,j} - y_{i-1,j}}{2\Delta\xi} \right) \left(\frac{y_{i,j+1} - y_{i,j-1}}{2\Delta\zeta} \right)$$

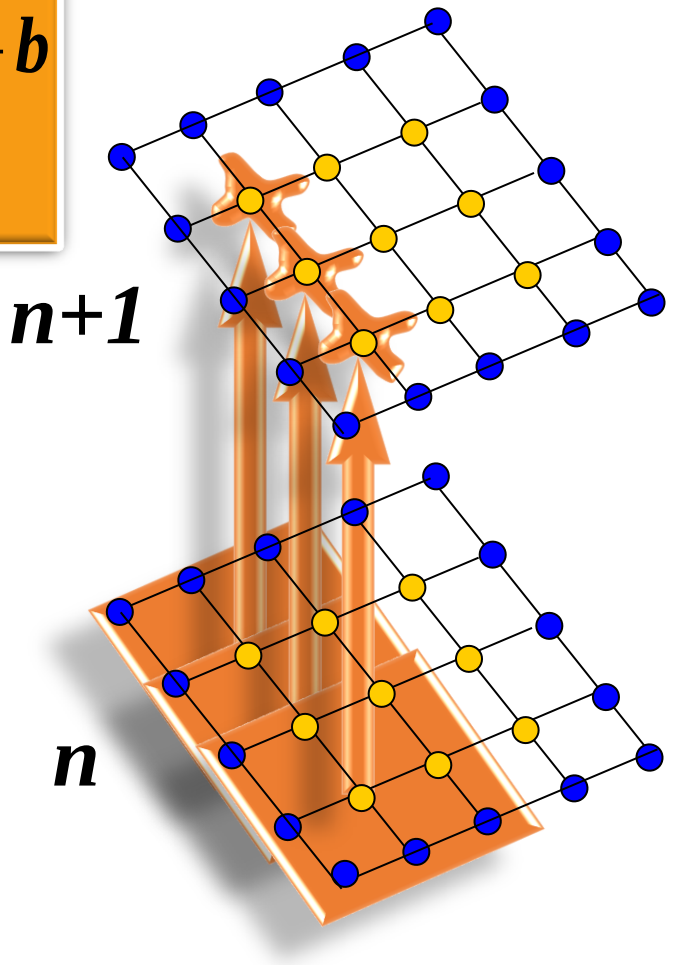
$$C_{i,j} = \left(\frac{x_{i+1,j} - x_{i-1,j}}{2\Delta\xi} \right)^2 + \left(\frac{y_{i+1,j} - y_{i-1,j}}{2\Delta\xi} \right)^2$$



Grid Generation: Computational Stencil

$$a_{i,j}x_{i,j}^{n+1} = \sum_{nb=E,W,N,S,NE,NW,SE,SW} a_{nb}x_{nb}^{n} + b$$

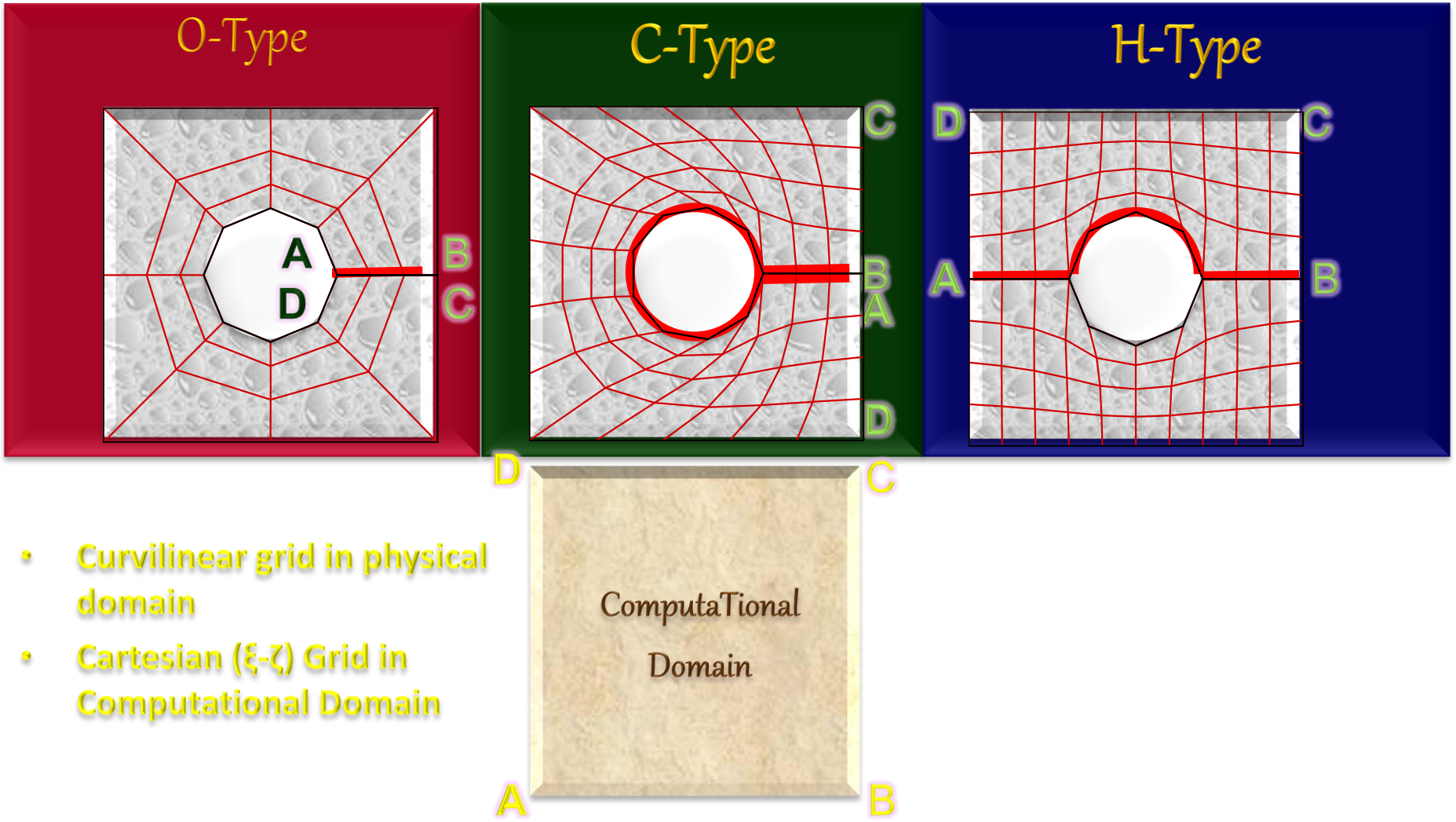
n: Iteration Number



Grid Generation: Solution Algorithm

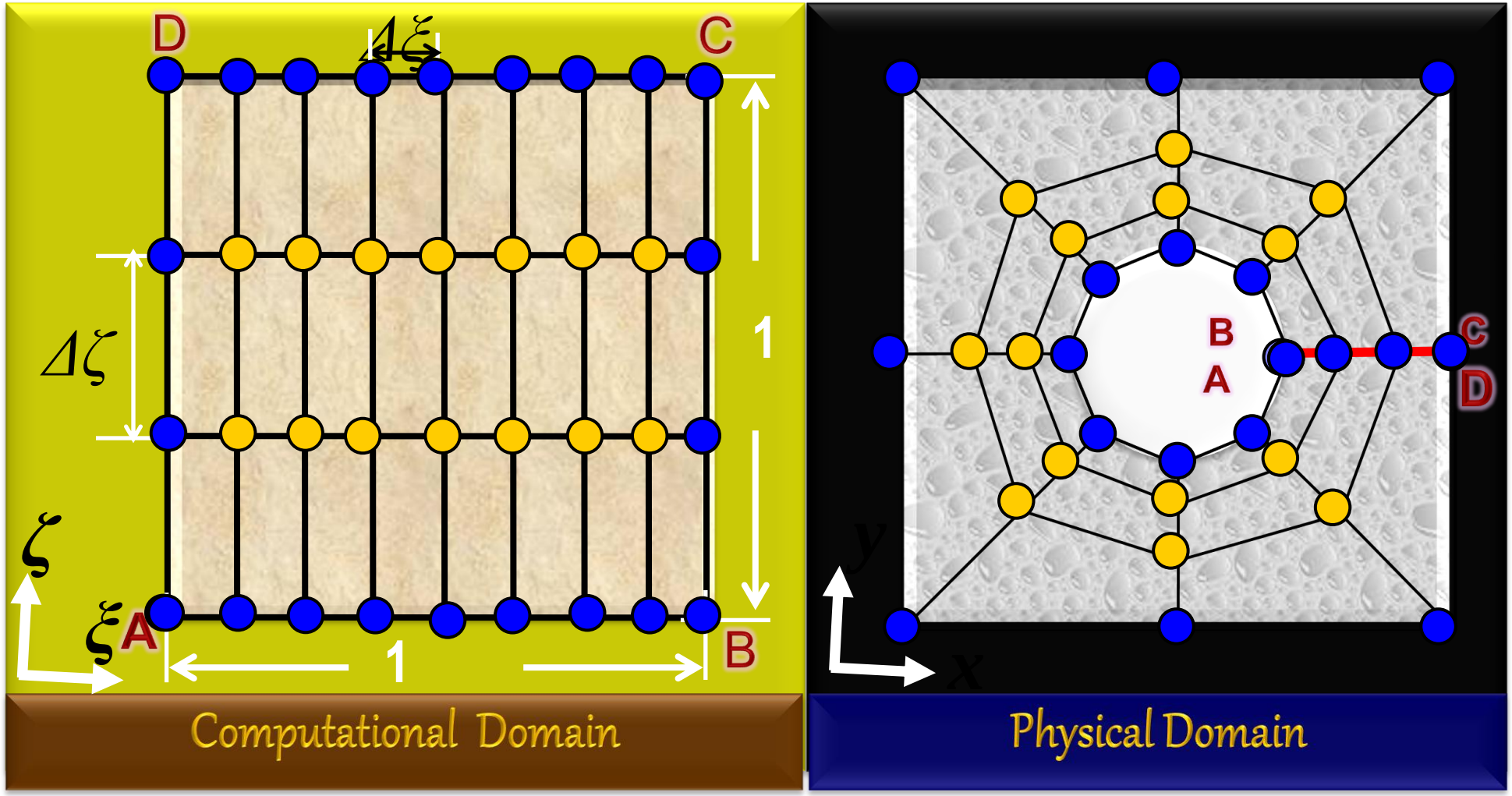
1. Enter imax, jmax and declare the matrix for x, y, xold and yold
2. Calculate dzi=1/(imax-1) and dzeta=1/(jmax-1)
3. Enter the initial guess for x and y
4. Enter the BC for x and y
5. xold=x and yold=y
6. Using $xold_{i,j}$ and $yold_{i,j}$, For i=2,imax-1 and j=2,jmax-1, Calculate
    1. $A_{i,j}$, $B_{i,j,}$ $C_{i,j}$, and $D_{i,j}$
    2. $x_{i,j}$ and $y_{i,j}$
7. Check whether the maximum of abs($x_{i,j}$-$xold_{i,j}$) and abs($y_{i,j}$-$yold_{i,j}$) for all interior grid points between two consecutive iteration is less than ε (for ex.: 10^{-3}, practically zero). If not, go to step 5 and continue till convergence

Grids for Multiply connected Domains: Hole in a Square Plate

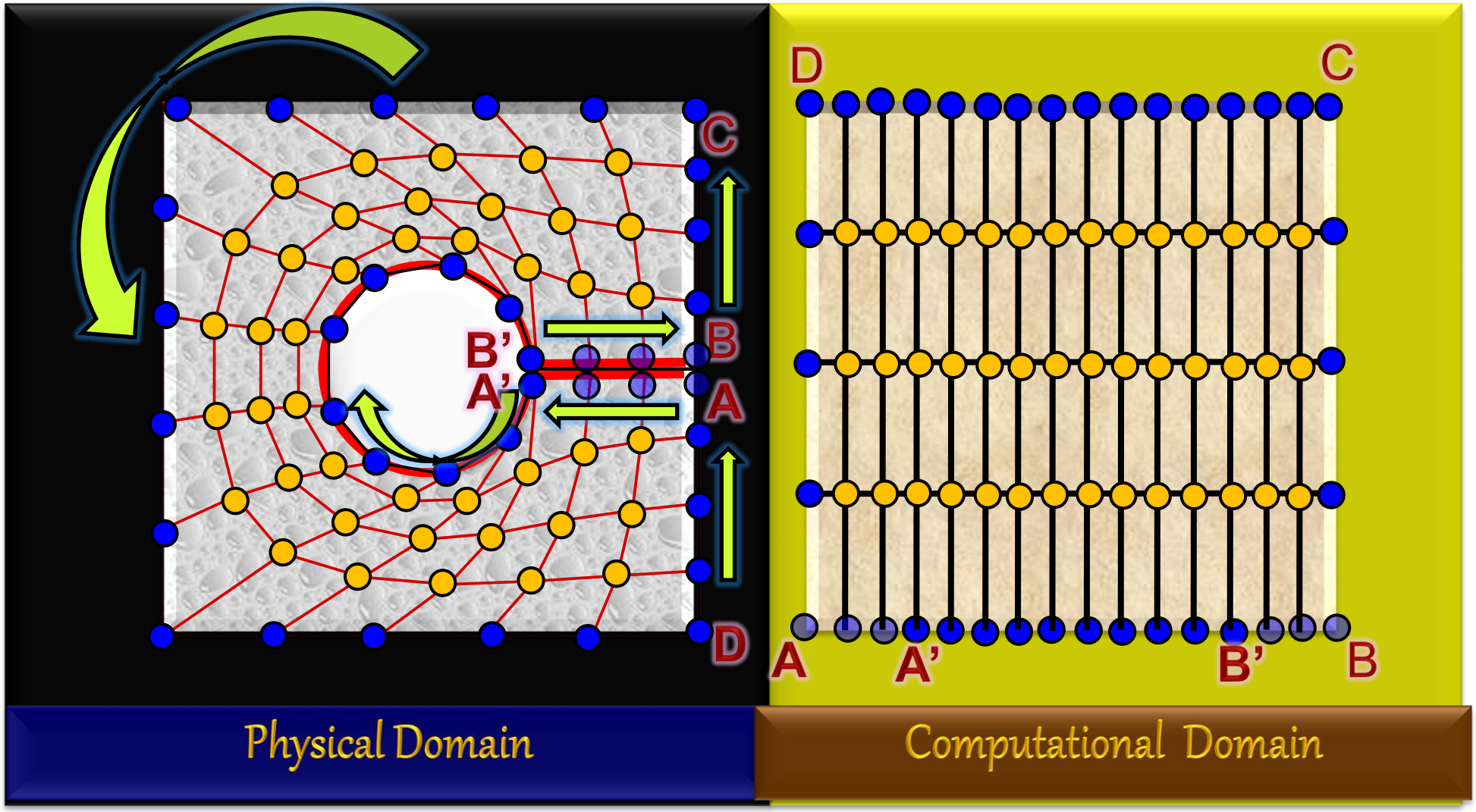


- Curvilinear grid in physical domain
- Cartesian (ξ-ζ) Grid in Computational Domain

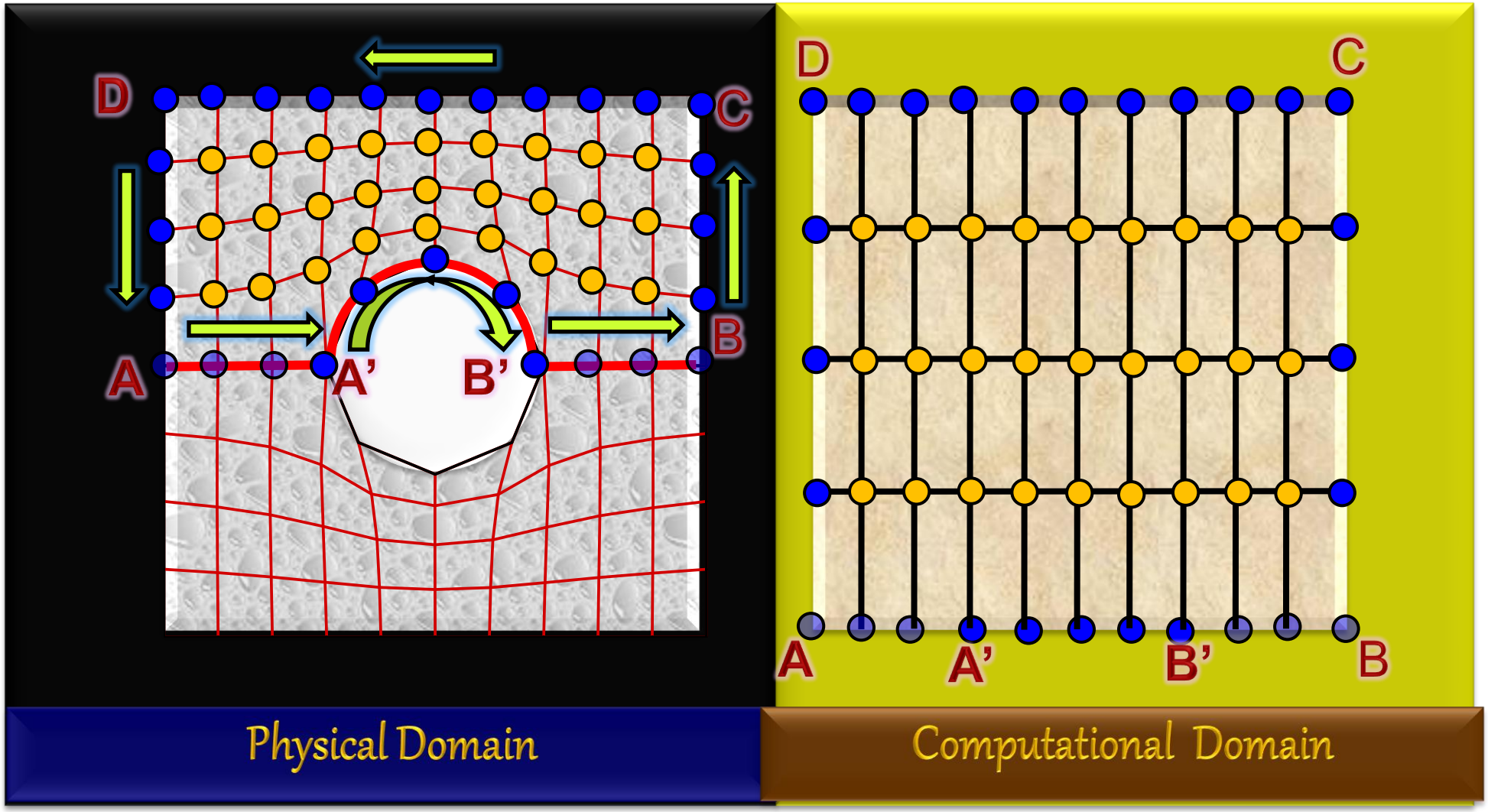
Grid Generation:
O-Type Structured Grids



C-Type Curvilinear Grid



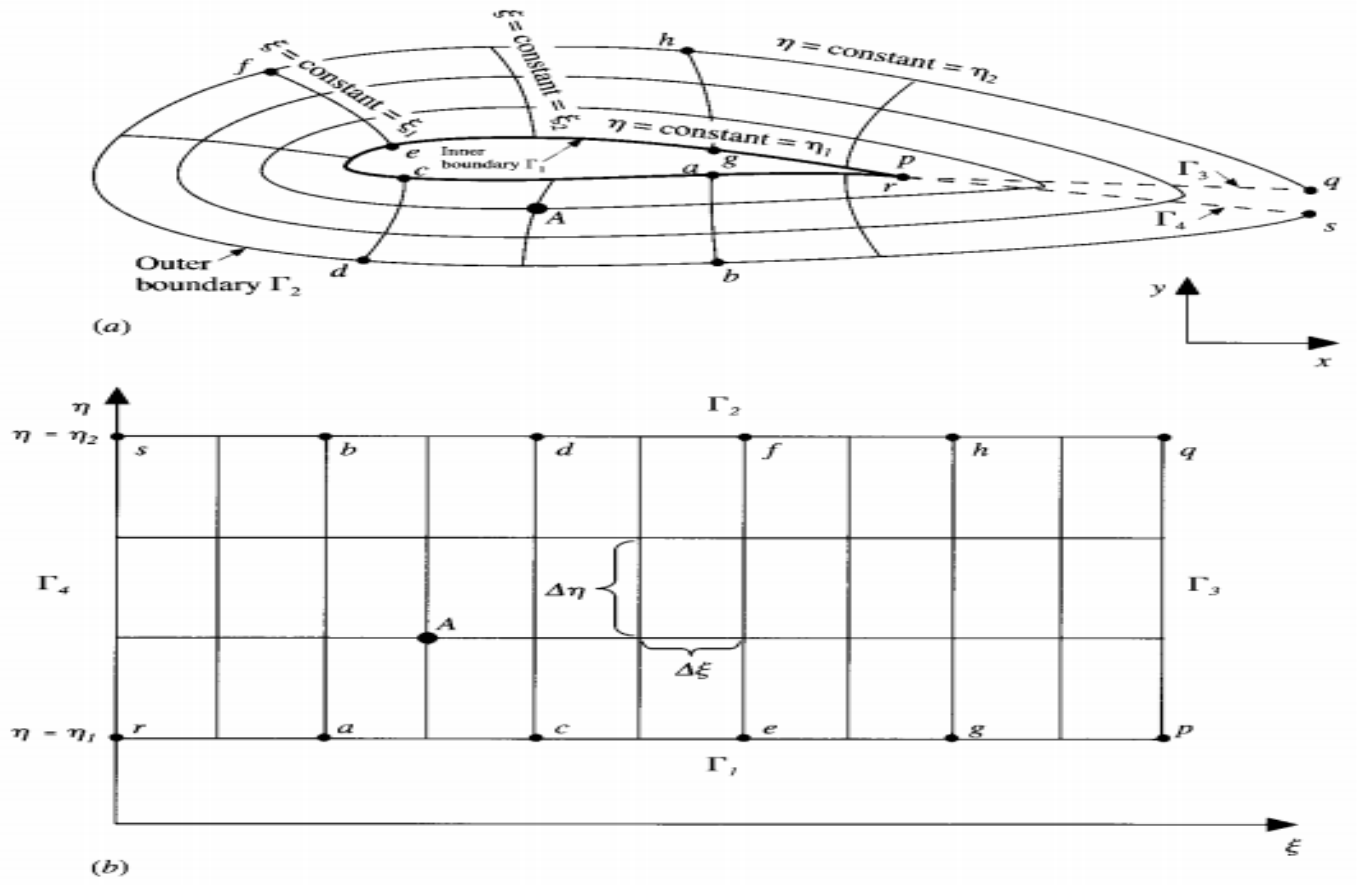
H-Type Curvilinear Grid



HYPERBOLIC Structured Grid

Hyperbolic Grid Generation

- Body conforming meshes
- Outer boundary is not specified
- Orthogonal grids
- Faster method



Hyperbolic Grid Generation

- Cell Volume scheme- Steger and Chausse

$$dx.dy = x_\xi y_\eta - x_\eta y_\xi = V(\xi,\eta) \rightarrow (1)$$

$$g_{11} = x_\xi^{\ 2} + y_\xi^{\ 2}$$

$$V = K\sqrt{g_{11}}e^{-\lambda(1-\eta)}$$

$$\lambda = 4$$

K=40,

$$\nabla\xi.\nabla\eta = x_\xi x_\eta + y_\xi y_\eta = 0 \rightarrow (2)$$

- Condition of orthogonality

$$y_\eta = \frac{V}{g_{11}} x_\xi \qquad x_\eta = -\frac{V}{g_{11}} y_\xi$$

Algorithm

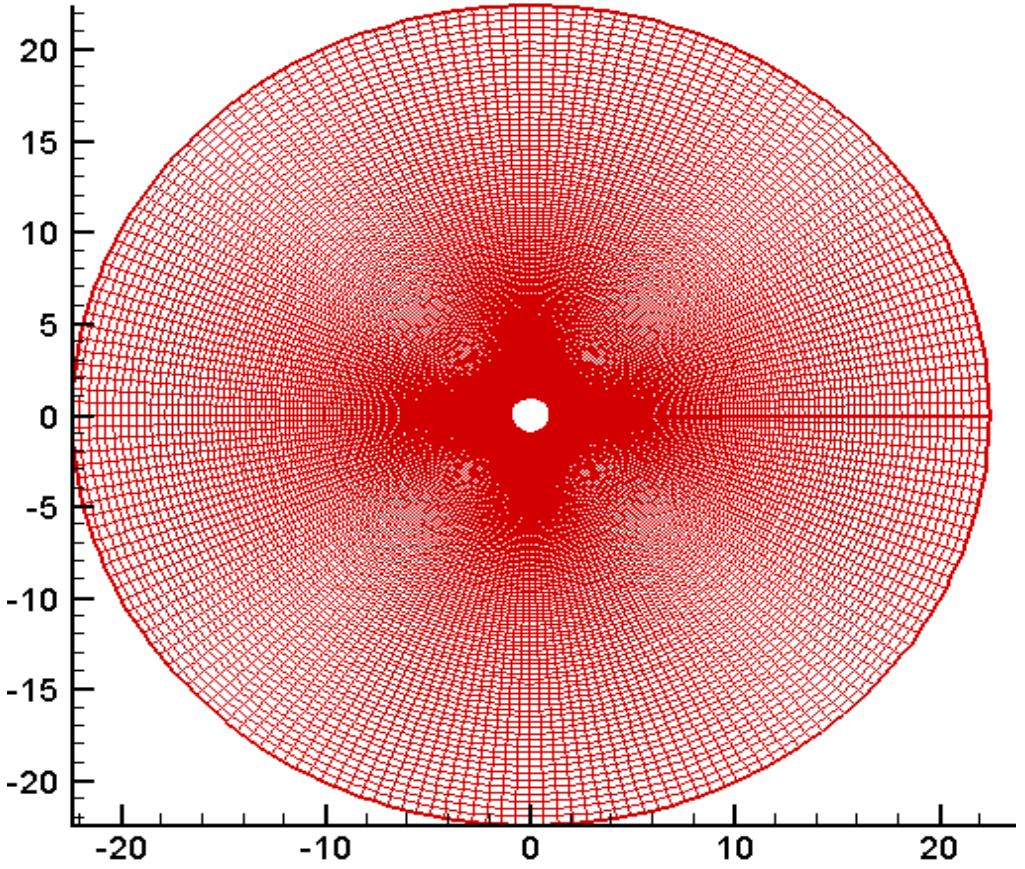
Step 1
- Input the body over which grid is to be generated

Step 2
- Calculate x_ξ and x_η at η =0 level(Central difference)

Step 3
- Calculate g11 and V

Step 4
- Calculate x_η and y_η

Step 5
- Integrate in η direction by 1 step
  - $x_{i,j+1} = x_{i,j} + x_\eta . \Delta\eta$
  - $y_{i,j+1} = y_{i,j} + y_\eta . \Delta\eta$

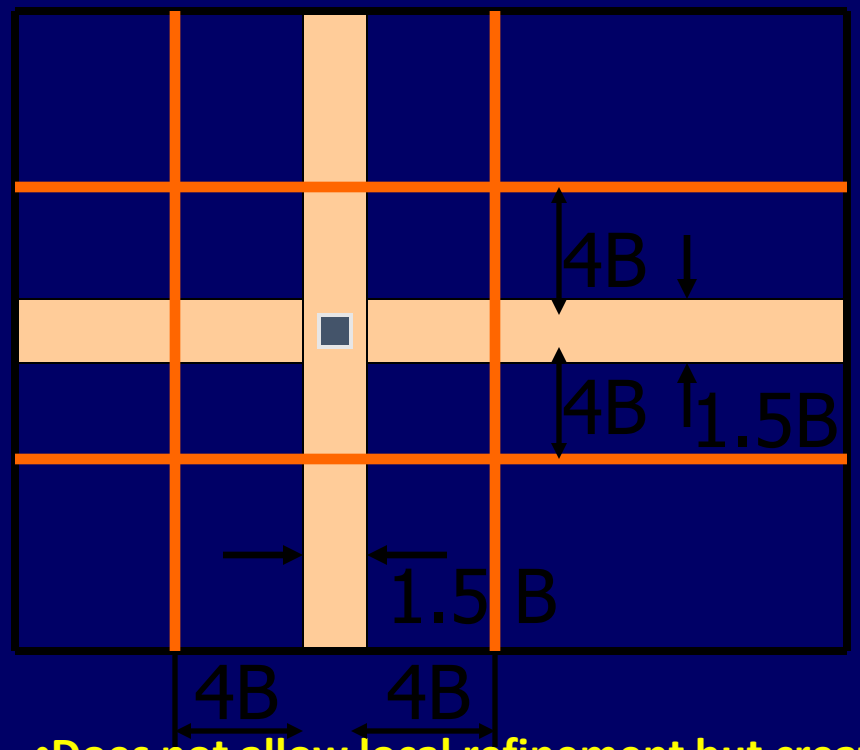


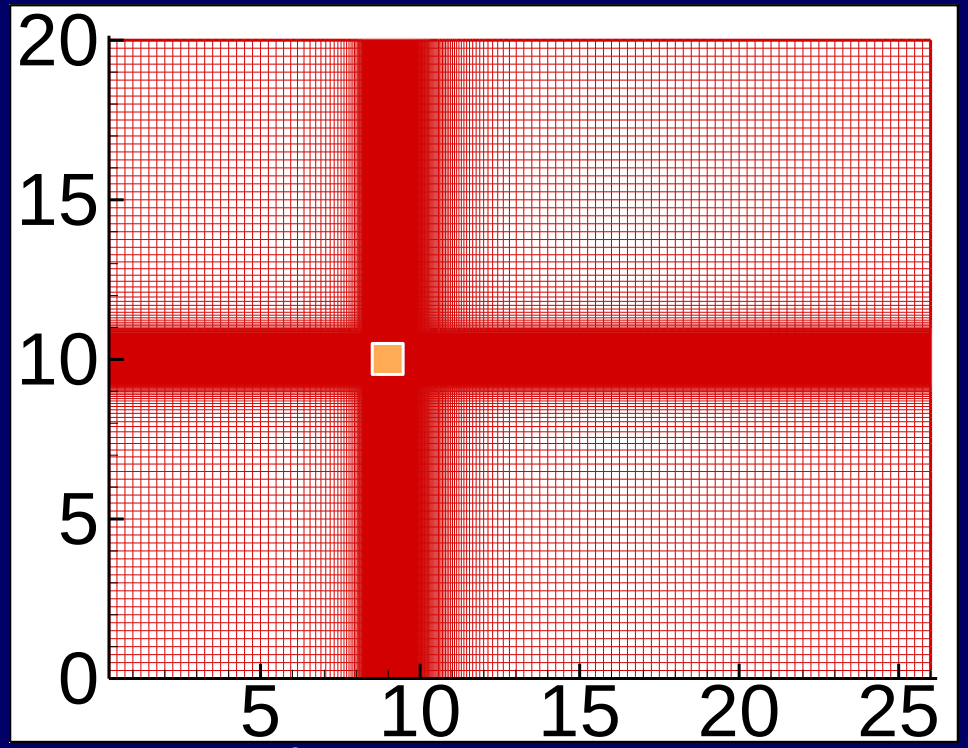
Hyperbolic grid over cylinder

Multi-Block Structured Grid

Grid Generation:
Flow across a Cylinder

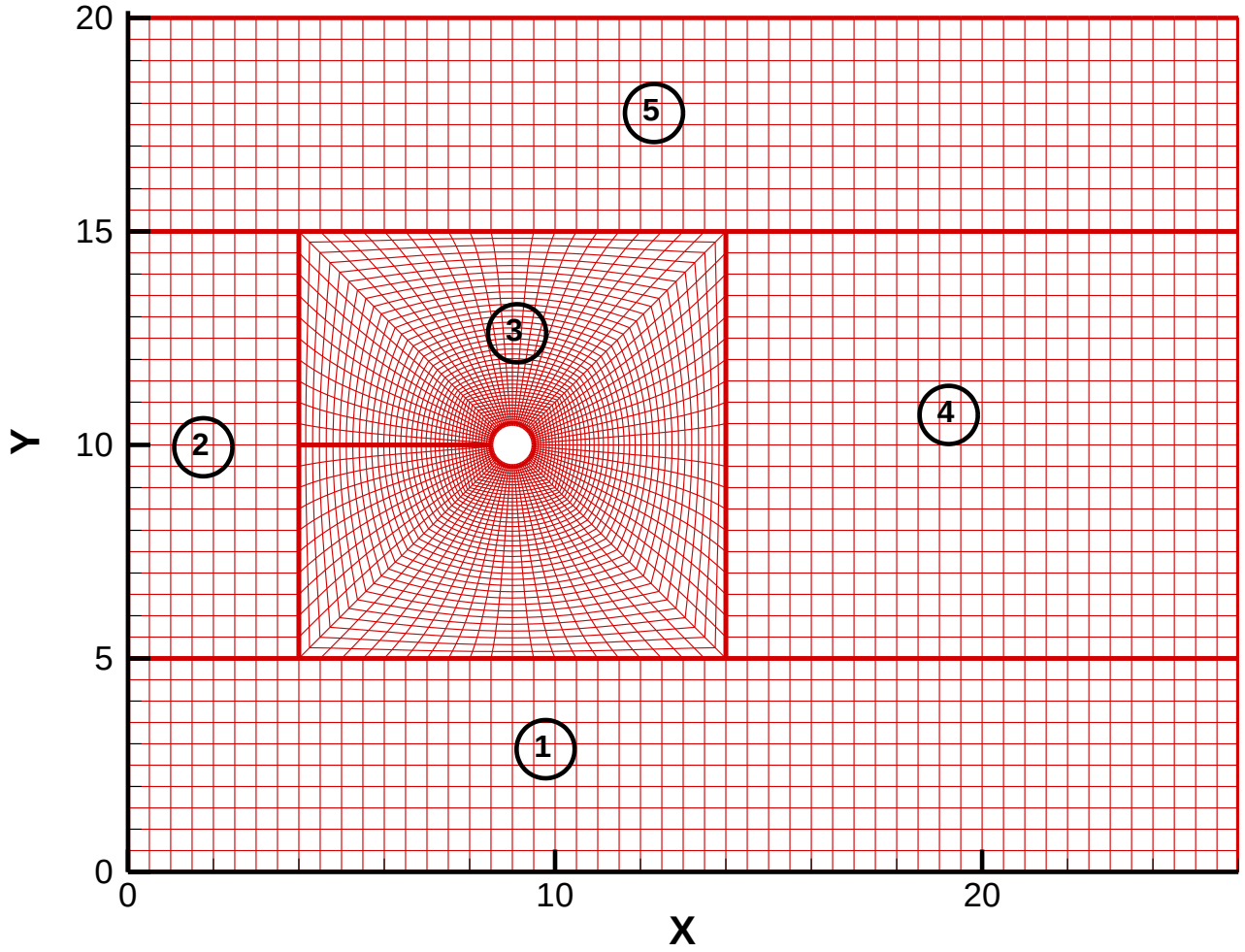
- Requirement: Local Fine Grid Clustering near the cylinder





- **Does not allow local refinement but creates a horizontal/vertical band of fine grid**

MultiBlock Structured Grid: Flow across a Circular Cylinder



Types of Grid Generation

- Structured Grid: Numerically efficient but less geometrical flexibility of local grid refinement
- Unstructured Grid: Greater geometrical flexibility but less numerically efficient.
- Multi-Block Structured Grid
  - The physical domain is divided into regions, called blocks.
  - Geometrical flexibility better than structured but worse than unstructured grid.
  - They are globally(in-between the blocks) unstructured but locally(within a block) structured
  - This block structuring can be viewed as compromise between high geometrical flexibility of fully unstructured grids and highly numerically efficient structured grids.
  - Natural Basis of Parallelization

Questions and Suggestions