# GRID TYPES (1)

Intimately Linked to Approximation Theory

Aim: Obtain Most Accurate Solution For Given Cost

Considerations:

- Geometry of the Problem (Simple, Complex, Mappable)
- Field to be Approximated (Shear/Boundary Layers, Vortices, Far-Field)
- Numerical Algorithm Used (ADI, MG, etc.)
- Personnel Time

# GRID TYPES (2)

Mesh Taxonomy:

A) Alignment With Surfaces

a) Aligned/Boundary Conforming/Body Fitted

- Keeps Boundary Conditions Simple

- Provides Good Resolution Near Body (NavSto)

b) Non-Aligned/Boundary Conforming/Body Fitted

C) Conformal

a) Conformal/Consistent

- No Special Points in Field

b) Non-Conformal

T) Topology

a) Fully Structured

b) Mini Structured Macro Unstructured

c) Fully Unstructured

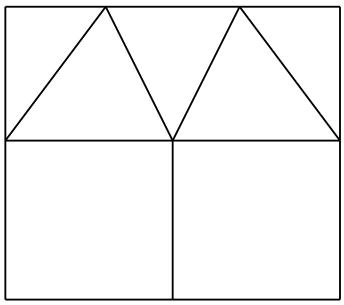
d) Mixed

E) Element or Cell Types

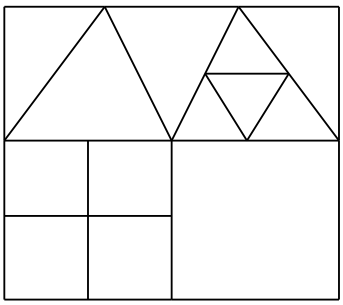
a) Uniform

b) Mixed (E.g. Tets/Prisms/Bricks)

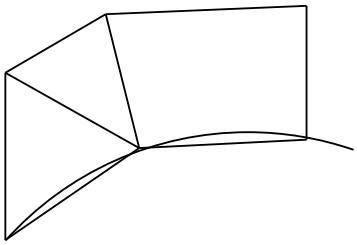
# GRID TYPES (3)



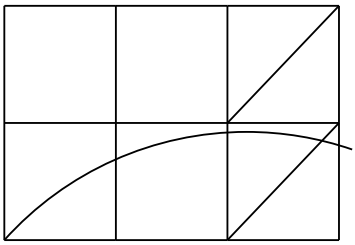
Conforming



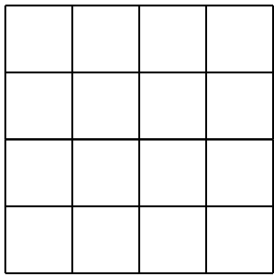
Non−conforming



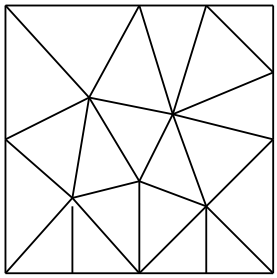
Surface aligned



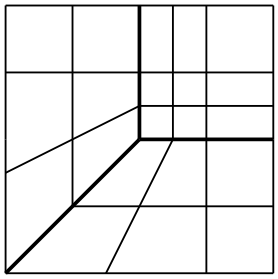
Non−surface aligned



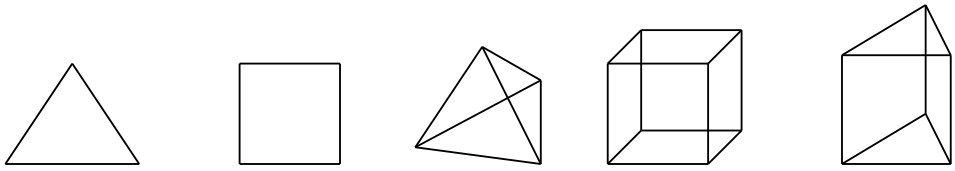
Micro−Structured



Micro−Unstructured



Macro−Unstructured
Micro−Structured



Element Type

Taxonomy of Grid Types

# GRID TYPES (4)

| | |
|---|---|
| Cartesian | A.b+C.a+T.a+E.a |
| Adaptive Cartesian | A.b+C.b+T.c+E.a |
| Mapped $I, J, K$-Grids | A.a+C.a+T.a+E.a |
| Chimera | A.a+C.b+T.b+E.a |
| Unstructured Uni-Element | A.a+C.a+T.c+E.a |
| General FEM Grids | A.a+C.a+T.c+E.b |

# GRID TYPES (5)

For Complex Geometries, Main Contenders Are:

a) Multiblock Mesh Systems:

- Older, More Developed
- Large Support Industry
- Not Automatic (1-2 Months) ⇒ User Intervention
- Chimera Very General

b) Unstructured Mesh Systems:

- Automatic (1 Week) ⇒ No User Intervention
- Most General
- Newer, Less Developed

c) Cartesian Mesh Systems:

- Automatic (1 Week) ⇒ No User Intervention
- Adaptivity Essential (Non-Conforming)
- Special Grids for NavSto
- Newest, Least Developed

# EFFICIENCY OF ELEMENT TYPES

Desired: Element Size $h \Rightarrow$ Element Volumes:

- Hexahedra: $h^3$
- Tetrahedra: $0.118\ h^3$ $(= 0.866 * 0.817 * h^3/6)$

**Elements** Per Unit Volume:

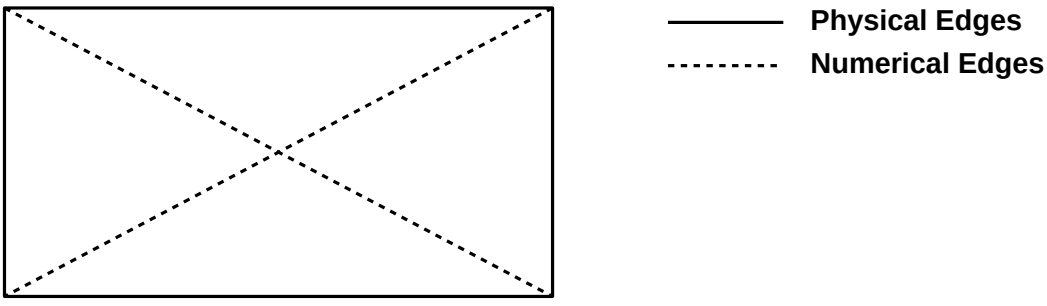
- Hexahedra: $h^{-3}$
- Tetrahedra: $8.5\ h^{-3}$

**Points** Per Unit Volume:

- Hexahedral Grid: $h^{-3}$
- Tetrahedral Grid: $1.54\ h^{-3}$ $(= 8.5\ h^{-3}/5.5)$

**Edges** (Work) Per Unit Volume:

- Simple FD Hex: $3 * h^{-3}$ ( 3 Edges per Point)
- Trilin FE Hex: $13 * h^{-3}$ (13 Edges per Point)
- Linear FE Tet: $11 * h^{-3}$ ( 7 Edges per Point $*1.54$)




Physical and Numerical Edges

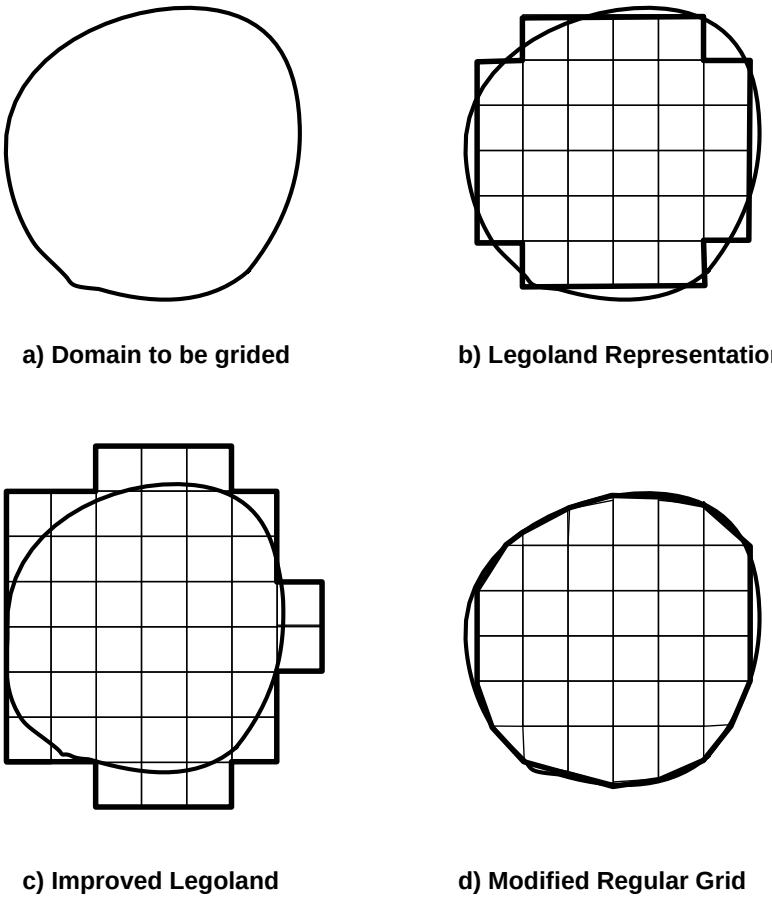
# CARTESIAN GRIDS (1)

References:

- Kuwahara, Boris, ...

Key Ideas:

- Set Up Uniform 'Voxel' Grid Over the Object
- Determine In/Out
- Keep Staircase Boundary ('Legoland Approximation')



**a) Domain to be grided** **b) Legoland Representation**

**c) Improved Legoland** **d) Modified Regular Grid**

Cartesian Grid

# CARTESIAN GRIDS (2)

Pros:

- Easy
- Direct Link to CAD Possible (In/Out Search)
- Lowest User Input
- Fast

Cons:

- Errors at Curved Boundaries
- Navier-Stokes Grids Difficult
- No Mesh Grading

Best For:

- Problems Without Geometrical/Physical Stiffness
- Problems Where Exact Geometry Irrelevant
- Quick Design Runs

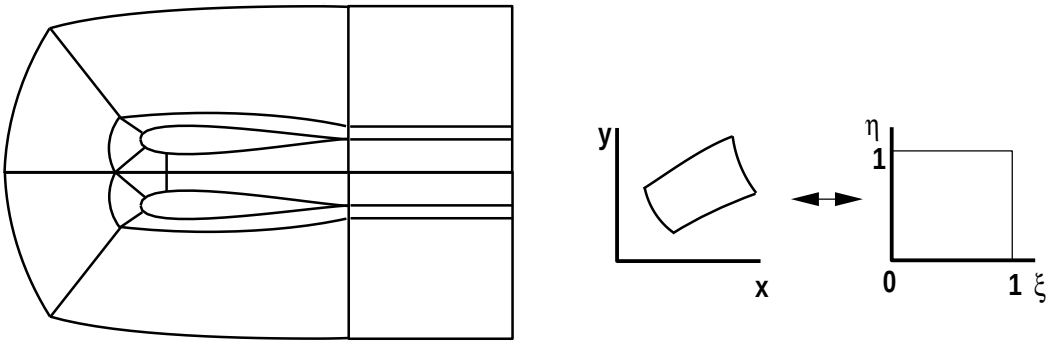
# MACRO-ELEMENT/MULTI-BLOCK (1)

References:

- Zienkiewicz & Phillips, Thompson, EAGLE, RAMBO, GRIDGEN, ...

Key Ideas:

- Subdivide Space Into Macro-Blocks (Macro-Bricks)
- Mesh Each Macro-Block
  - Algebraic/Transfinite
  - Elliptic




Multi-Block Grid

# MACRO-ELEMENT/MULTI-BLOCK (2)

Pros:

- Easy
- Algebraic (Fast)
- No Search
- Vectorizable/Parallelizable

Cons:

- High Manual Input
- Points Must Meet at Boundaries
  $\Rightarrow$ May Lead to Unnecessary Resolution
- Difficult to Automate
- No Guarantee of 'Good Macro-Blocks' for general Geometries
  $\Rightarrow$ May Lead to Bad Elements
- 'Not General'

Best For:

- Simple Domains With Brick Elements

# UNSTRUCTURED GRID GENERATION (1)

As More General, More Options Immediately Possible
⇒ Order According to:

1. Ways to Construct a Mesh:

   a) Fill 'empty', i.e. not yet gridded space
      ⇒ **Advancing Front** (AFT)
   b) Modify existing grid
      ⇒ **Delaunay Triangulation** (GVT)

2. Order of Creation for Points/Elements:

   a) Generate Points Ahead of Time
   b) Generate Points and Elements Simultaneously

3. Ways to Specify Element Size/Shape:

   a) Cartesian Point Distribution
   b) Quad/Octree Point Distribution
   c) Random Point Distribution
   d) Point Distribution from Overlapping Algebraic Grids
   e) From an Analytic Function
   d) From a Background Grid
   f) From an Internal Measure of Quality

# UNSTRUCTURED GRID GENERATION (2)

| | |
|---|---|
| Lo | 1.a+2.a+3.a |
| van Phai | 1.a+2.a+3.e |
| Morgan/Peraire/Löhner | 1.a+2.b+3.d |
| Huet | 1.a+2.b+3.f |
| Weatherhill/Mavriplis | 1.b+2.a+3.d |
| Baker (1985) | 1.b+2.a+3.d |
| Baker (1990) | 1.b+2.a+3.a+3.d |
| Modified Octree (Yerry/Shephard) | 1.b+2.a+3.b |
| Holmes/Weatherhill/George... | 1.b+2.b+3.f |

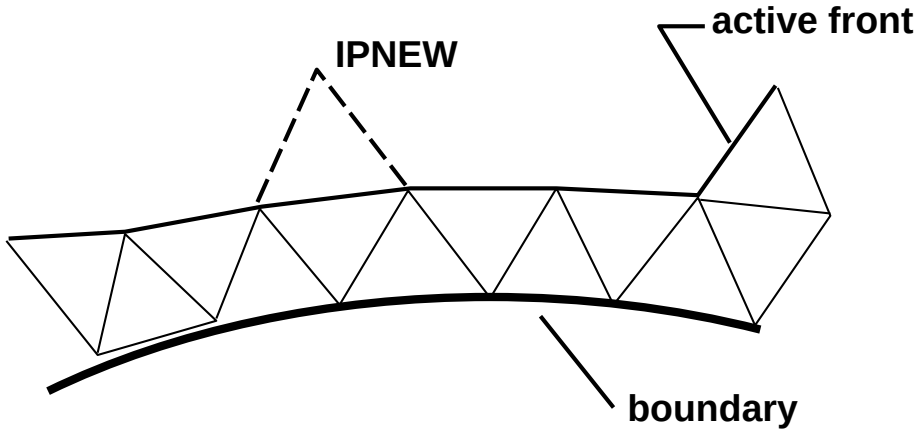
# ADVANCING FRONT (1)

References:

- Morgan, Peraire, Peiro, Löhner, ...

Key Ideas:

- Delete a Face From the Front
- Possibly Introduce a New Point
- Introduce a New Element
- Update Front of Faces




Advancing Front Technique

# ADVANCING FRONT (2)

Pros:

- 'General'
- Any Point Distribution Possible
- Stretching/Grading Possible
- No Problems With 'Voids'

Cons:

- 3-D Local Surface Search For Each Element
- Use of Quads/Bricks Alone ?
    - Quads Possible
- Scalar

Best For:

- Complicated Domains, Tetrahedra + Stretching
- Adaptive Remeshing

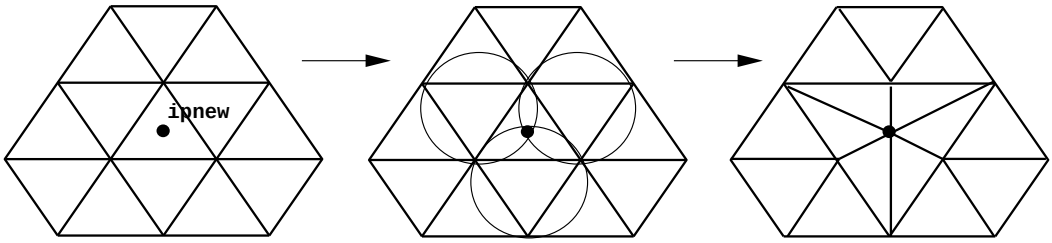
# DELAUNAY TRIANGULATION (1)

References:

- Watson, Weatherhill, Mavriplis, Baker, Holmes, George, ...

Key Ideas:

- Introduce a New Point
- Remove 'Convex Hull' of Elements Close to It
- Reconnect to Form New Elements




Delaunay Triangulation

# DELAUNAY TRIANGULATION (2)

Pros:

- 'General'
- Guarantee of 'Best' Tetrahedra
- Any Point Distribution Possible
- Stretching/Grading Possible
- Fast

Cons:

- Logic Intensive for 'Voids'
- Surface Recovery Necessary
- Scalar
- No Quads/Bricks

Best For:

- Complicated Domains, Tetrahedra + Stretching
- Adaptive Remeshing

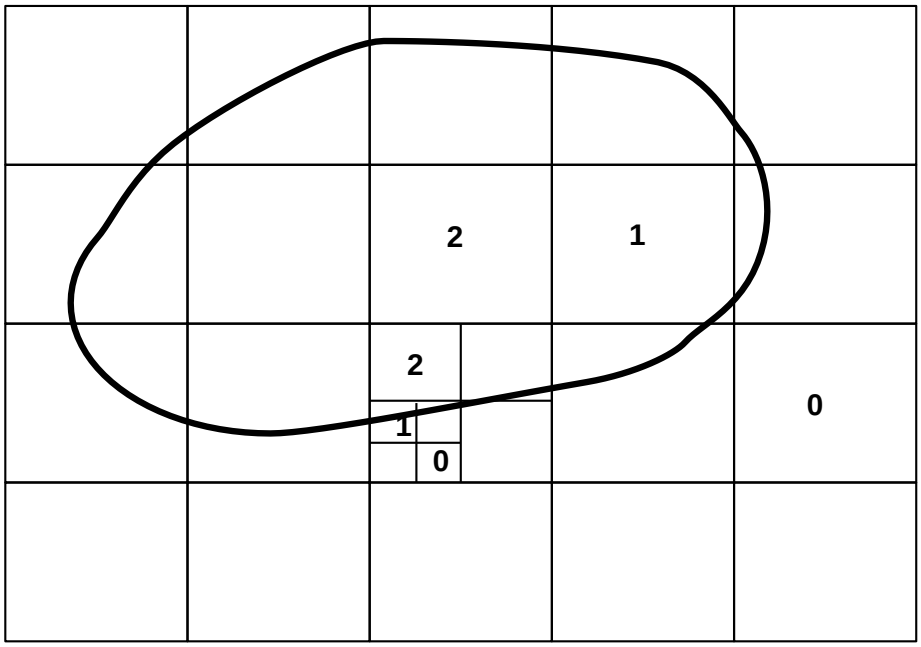
# MODIFIED QUADTREE (2-D) / OCTREE (3-D)

References:

- Yerry, Shephard, Nakahashi, ...

Key Ideas:

- From CAD: Obtain Quad/Octree
- Determine In/Out
- Modify Staircase Boundary
- Reconnect to Form Triangles/Tetrahedra




Modified Quadtree Technique

# MODIFIED QUADTREE (2-D) / OCTREE (3-D)

Pros:

- 'General'
- Connection to Solic Modeller
- Integer Logic Only
- Stencils for Interior Tets ⇒ Fast

Cons:

- Stretching ?
- Logic Intensive
- Cusps in Geometry Difficult
- Use of Quads/Bricks Alone ?
- Scalar In Boundary Reconnection Phase

Best For:

- Complicated Domains with Regular Tetrahedral Elements
- Adaptive Remeshing