A brief introduction to mesh generation

Josep Sarrate

Laboratori de Càlcul Numèric (LaCàN)

Departament d'Enginyeria Civil i Ambiental

Universitat Politècnica de Catalunya (Spain)

http://www.lacan.upc.edu

Layout of the course

1. Why do we need meshes?
2. Geometry description
3. Classification of mesh generation methods
4. Structured mesh generation methods
5. Unstructured mesh generation methods
6. Mesh optimization and mesh adaption
7. Concluding remarks

Layout of the course

1. Why do we need meshes?
2. Geometry description
3. Classification of mesh generation methods
4. Structured mesh generation methods
5. Unstructured mesh generation methods
6. Mesh optimization and mesh adaption
7. Concluding remarks

Why do we need meshes?

No mesh

No simulation

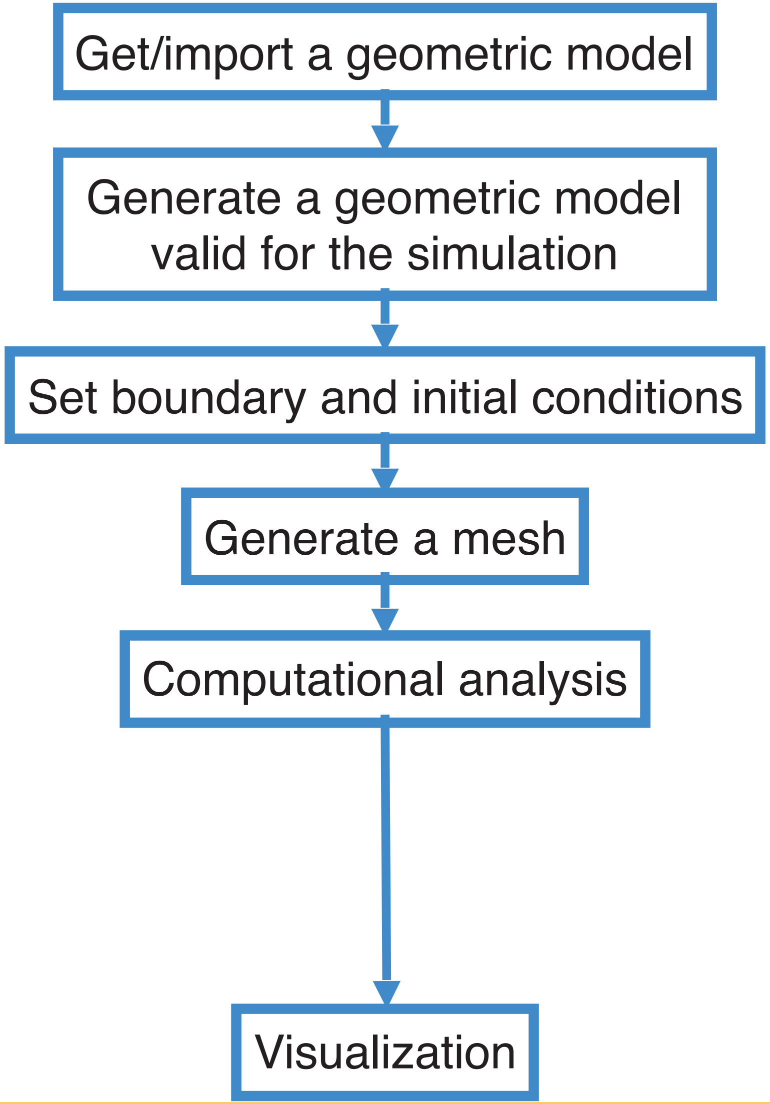
Low quality mesh

Low accuracy in the simulation

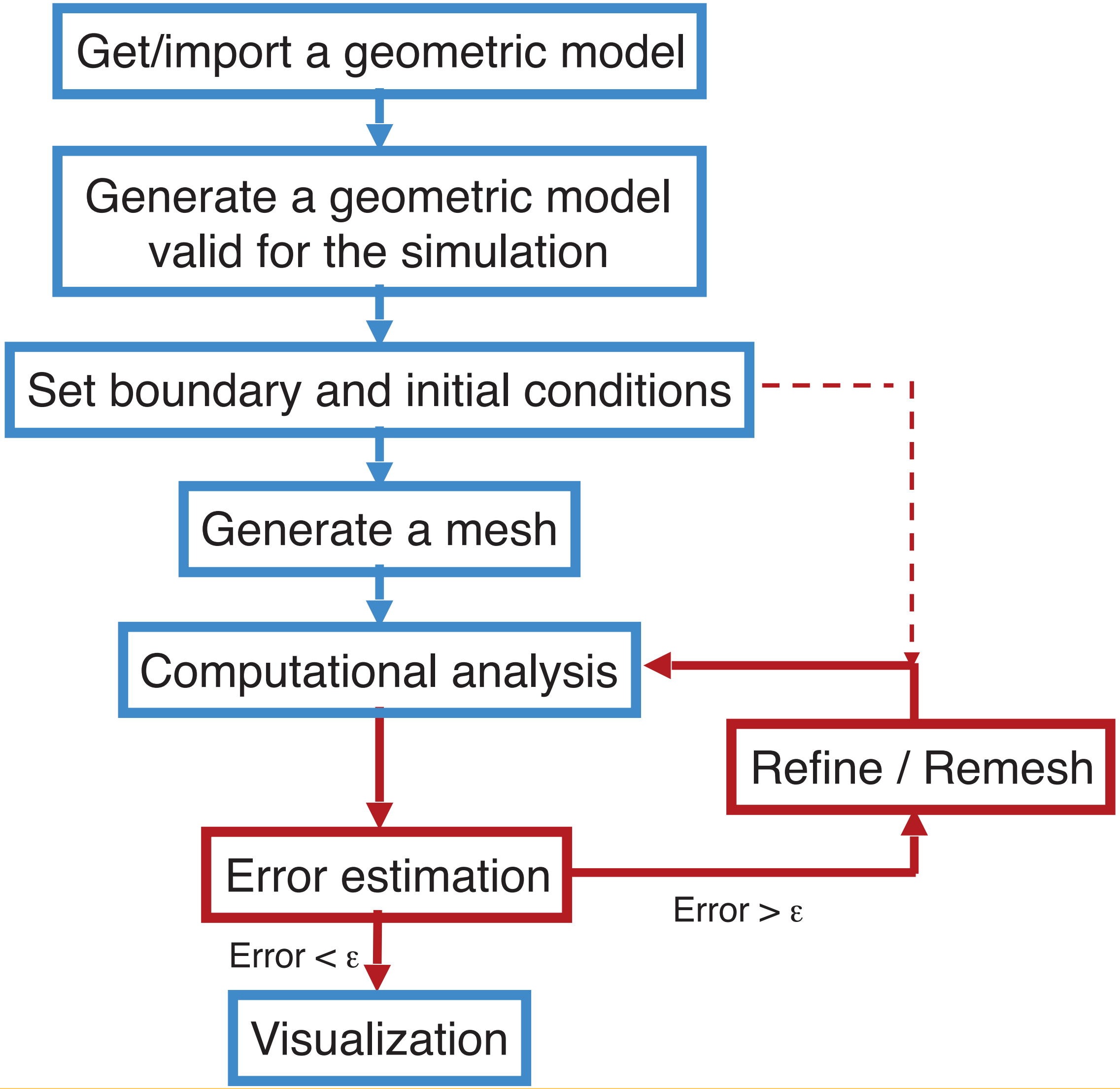
1. Why do we need meshes?

- The simulation process

Standard simulation process



Adaptive simulation process



Layout of the course

1. Why do we need meshes?
2. Geometry description
3. Classification of mesh generation methods
4. Structured mesh generation methods
5. Unstructured mesh generation methods
6. Mesh optimization and mesh adaption
7. Concluding remarks

2. Geometry description

- **How are geometries described?**
- CAD (brep)

- Tessellation

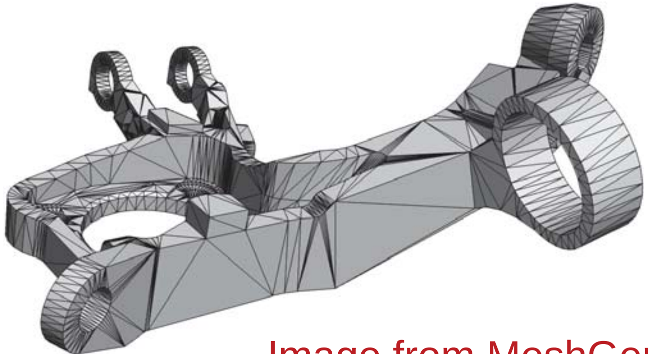


Image from MeshGems

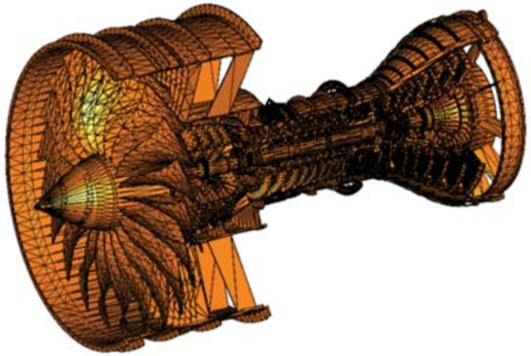
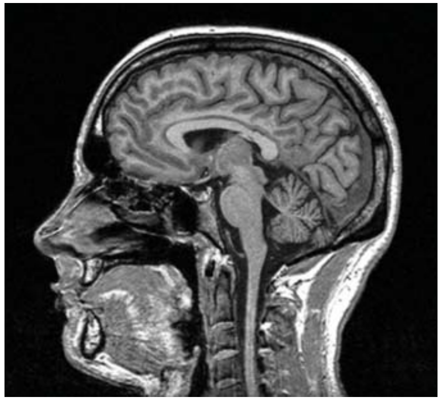
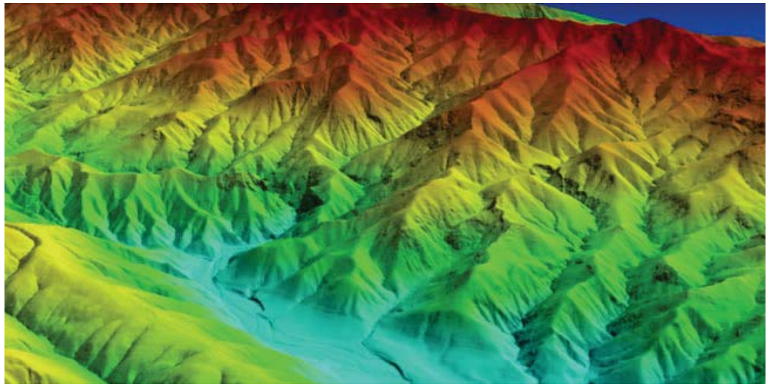


Image from CADfix

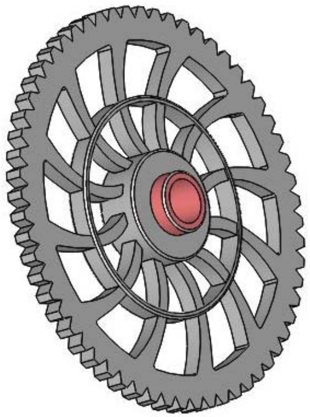
- Images

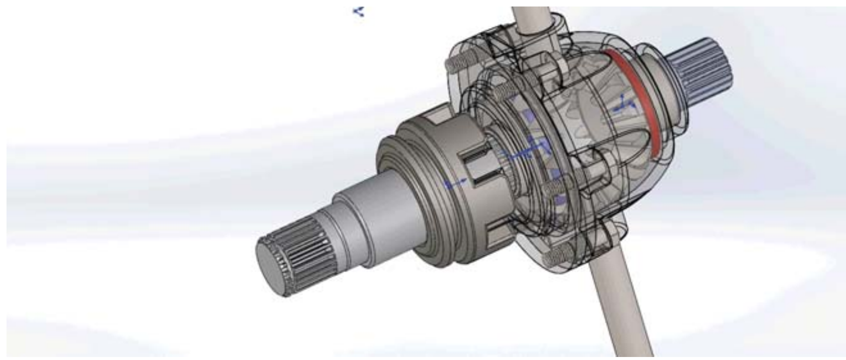




2. Geometry description

- Geometric models are developed mostly taking into account visualization, design, prototype and manufacture constraints

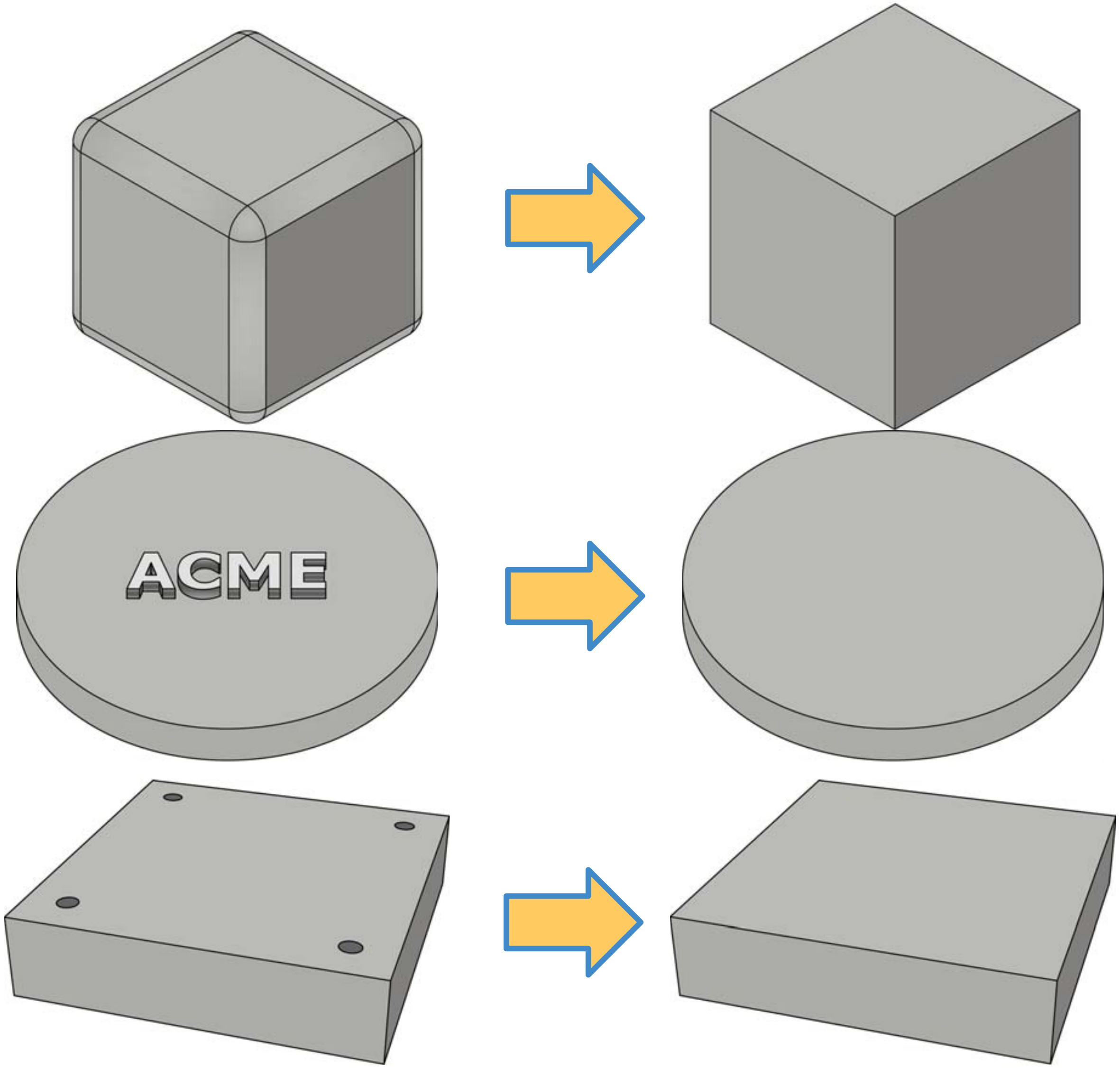




- Can not be directly applied into a simulation process
- They must be adapted to the simulation process:
  - Requirements prescribed by the physics of the problem
  - Requirements prescribed by the numerical method
- Two major types of actions
  - De-featuring
  - Healing

2. Geometry description

- **De-featuring**: removing geometrical details that are not relevant for the simulation process



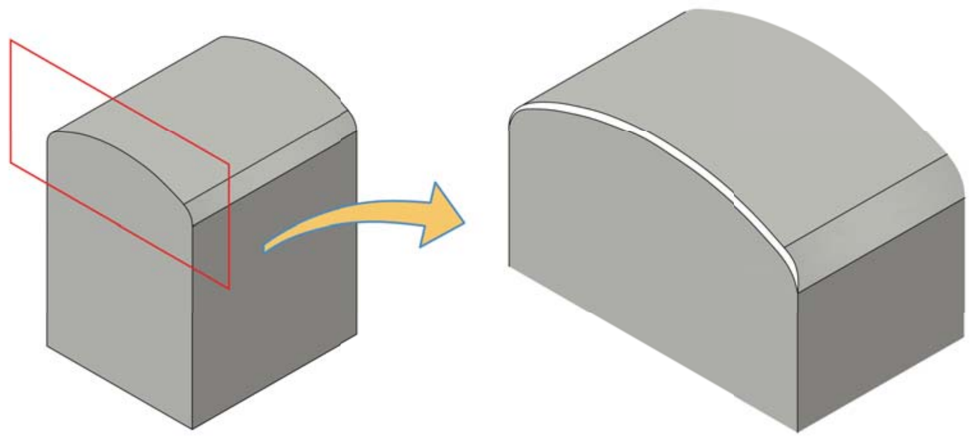
Remove fillets, chambers, blend surfaces…

Remove imprints or extrusions

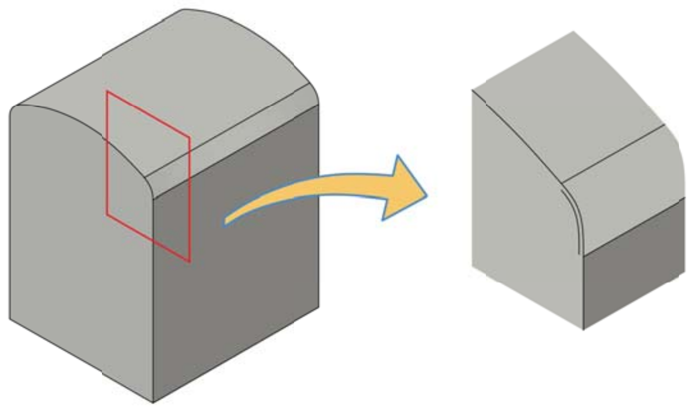
Remove holes

- **Healing**: repairing errors in the geometrical model

  Watertight: all the volumes must be completely closed



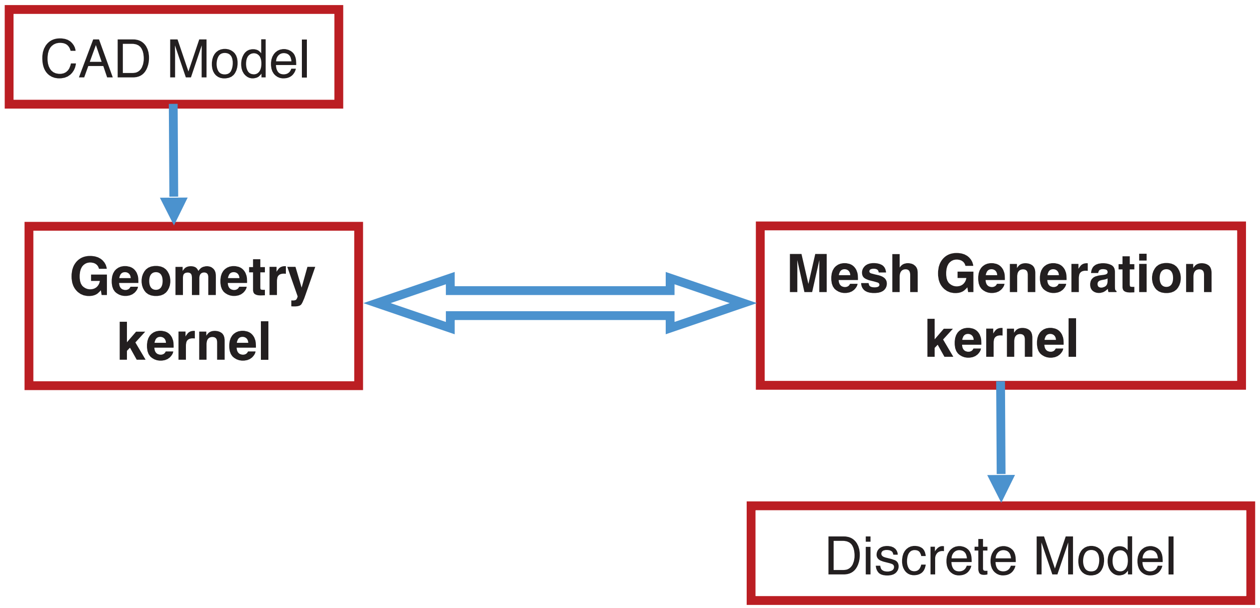
  Remove repeated entities



  Other issues: lost entities, detached entities, …

- **How we can address theses issues?**
  - Using CAD packages
    - CadFix (ITI TranscenData)
    - Creo Elements/Pro (ProEngineer)
    - Catia (Dassault Systems)
    - Solid Works (Dassault Systems)
    - Solid Edge (Siemens)
    - Rhinoceros
  - Some meshing environments provides tools
    - Catia (Dassault Systems)
    - CD-Adapco (Siemens)
    - Ansys
    - Abaqus (Dassault Systems)
    - GiD

- **How CAD models are linked to a meshing environment?**



- The geometry and the mesh of the model are represented by entities.
- In both cases it is of the major importance to take into account:
  - The topological relationship between entities
  - The geometrical relationship between entities

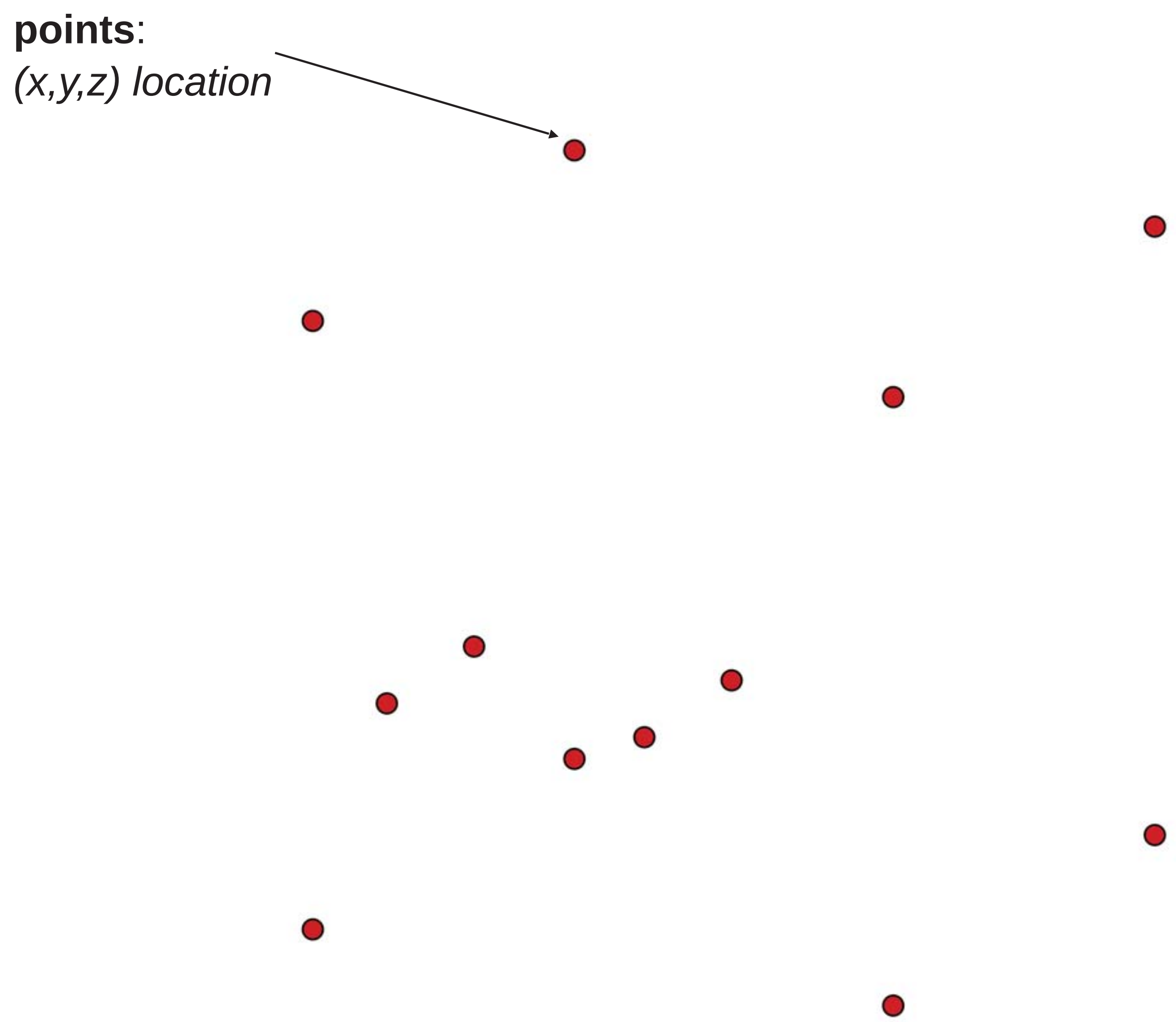
2. Geometry description

- Geometry engines (kernels)
  - Parasolid (Siemens)
  - Acis (Dassault Systems)
  - Open Cascade (Principia, open source engine)

- Some engines provide a file format to import / export the model representation
  - FBX/X_T (Parasolid)
  - SAT (Acis)

  Other open file formats
  - STEP
  - IGES

- CAD geometry representation

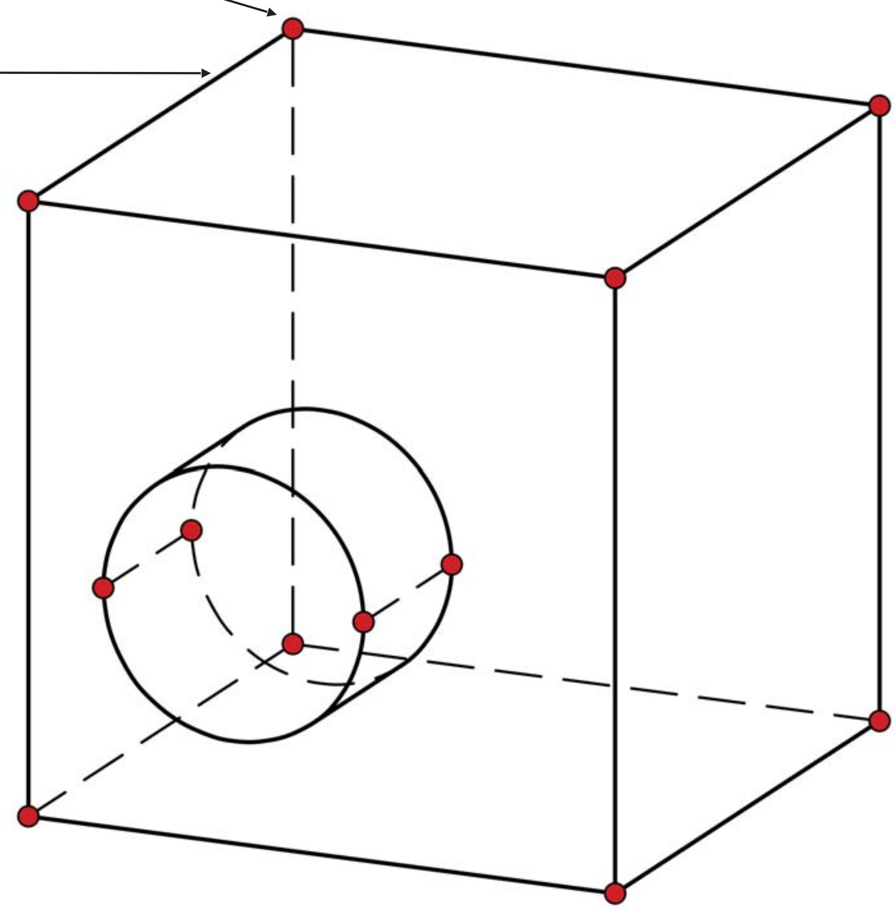


2. Geometry description

- CAD geometry representation

points:
(x,y,z) location

Curves:
bounded by two points

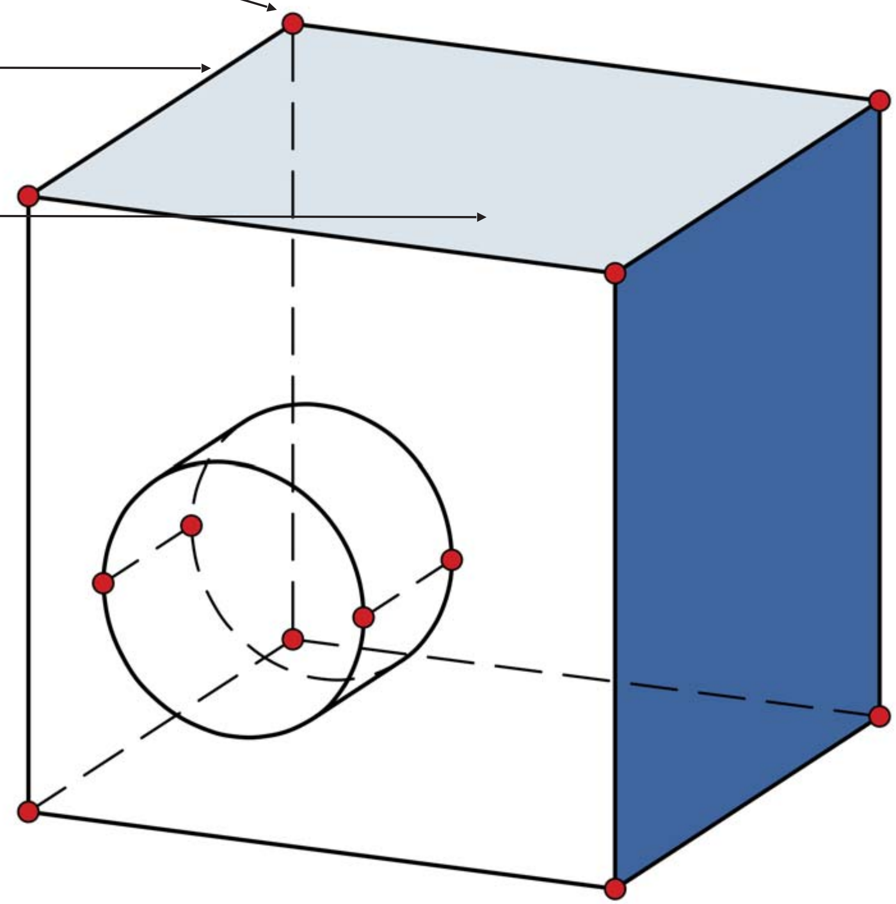


- CAD geometry representation

points:
(x,y,z) location

Curves:
bounded by two points

Surfaces:
closet set of curves



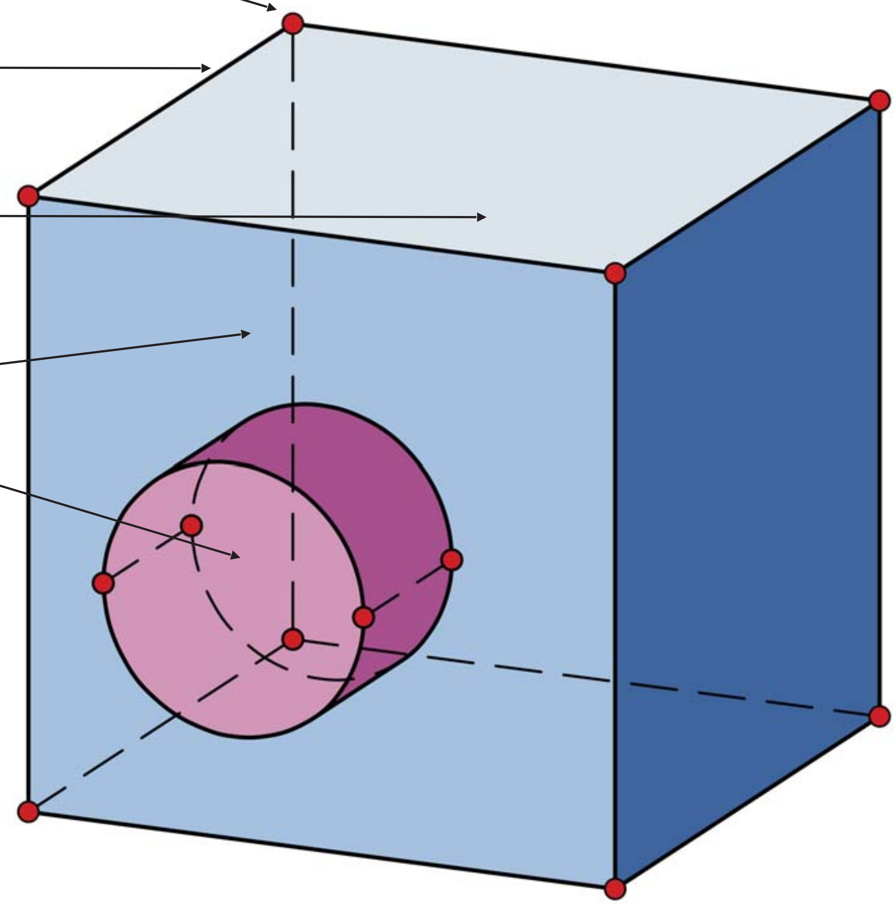
- CAD geometry representation

points:
(x,y,z) location

Curves:
bounded by two points

Surfaces:
closet set of curves

Volumes:
closet set of surfaces



2. Geometry description

- CAD geometry representation

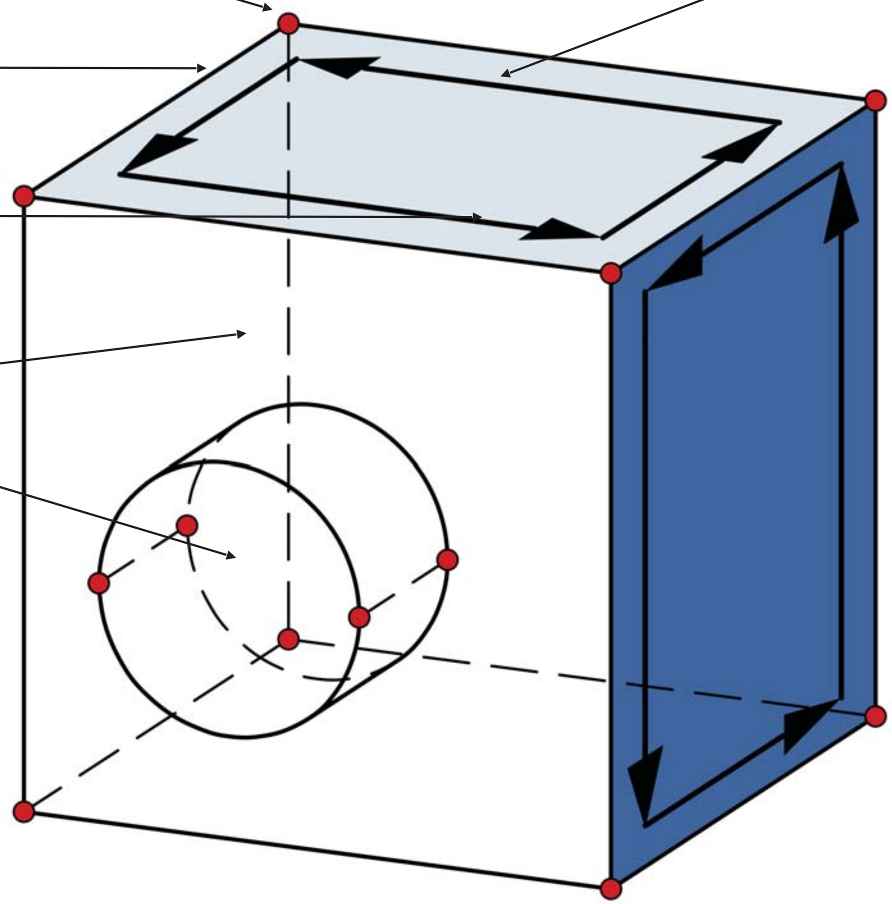
points:
(x,y,z) location

Wire (loop):
ordered set of curves on surface

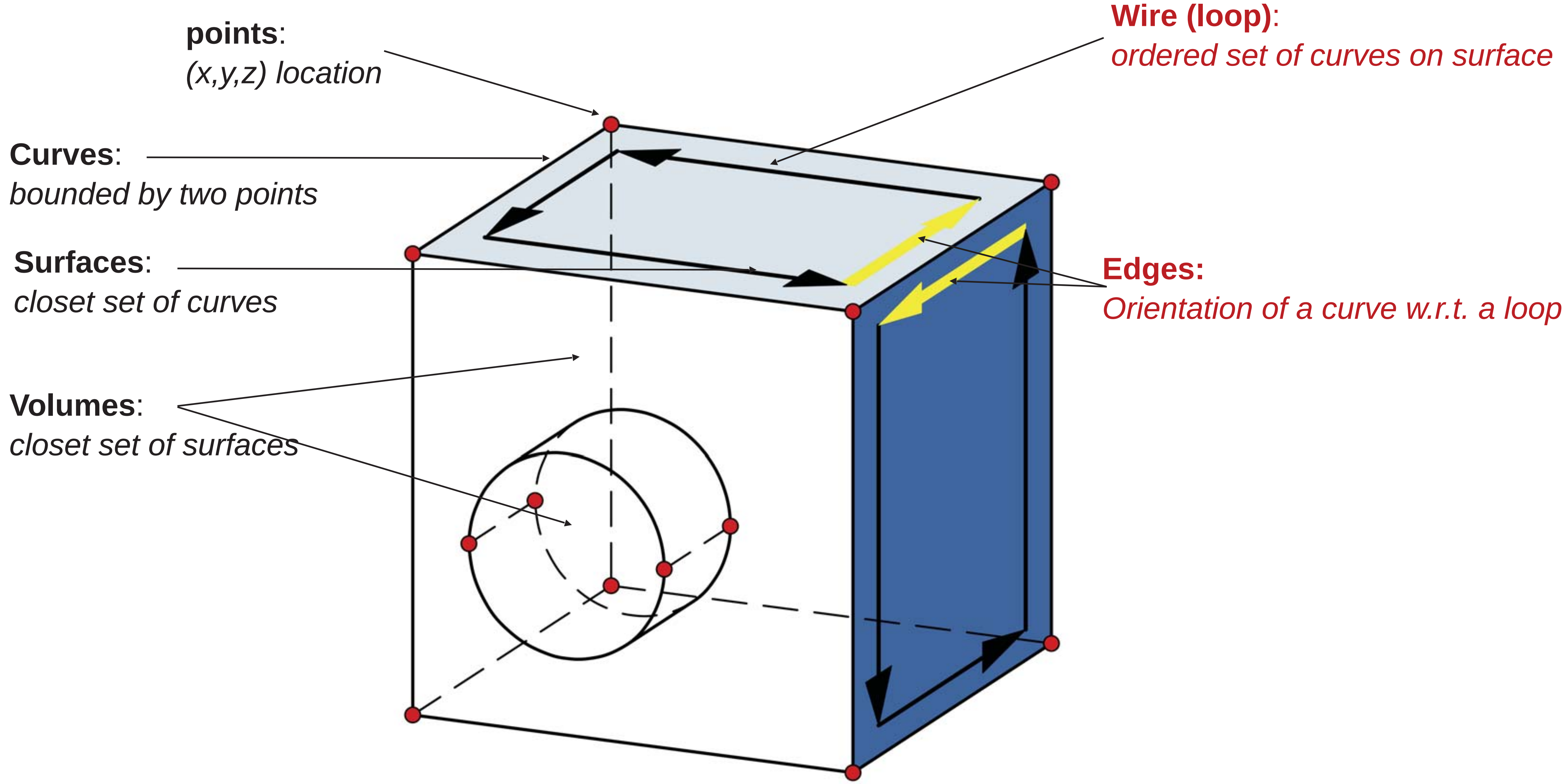
Curves:
bounded by two points

Surfaces:
closet set of curves

Volumes:
closet set of surfaces



- CAD geometry representation



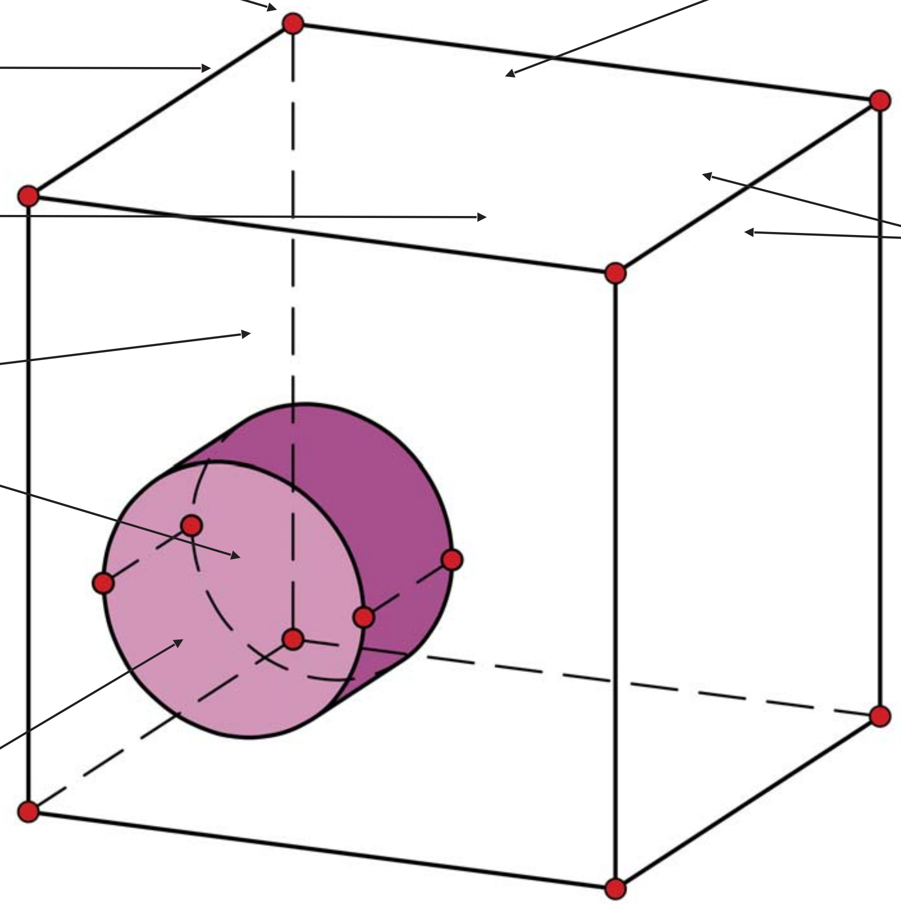
- CAD geometry representation

points:
(x,y,z) location

Curves:
bounded by two points

Surfaces:
closet set of curves

Volumes:
closet set of surfaces

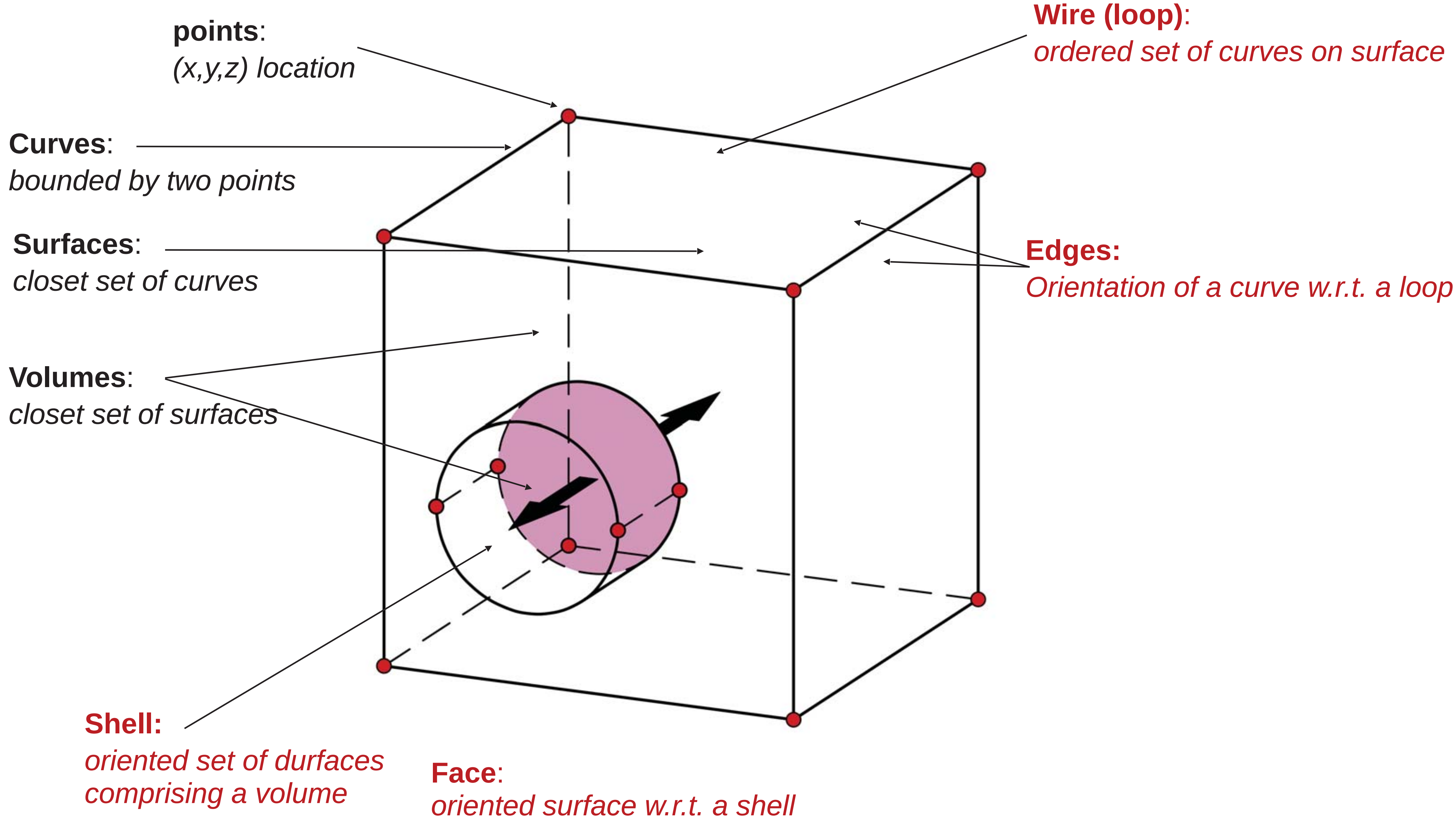


Wire (loop):
ordered set of curves on surface

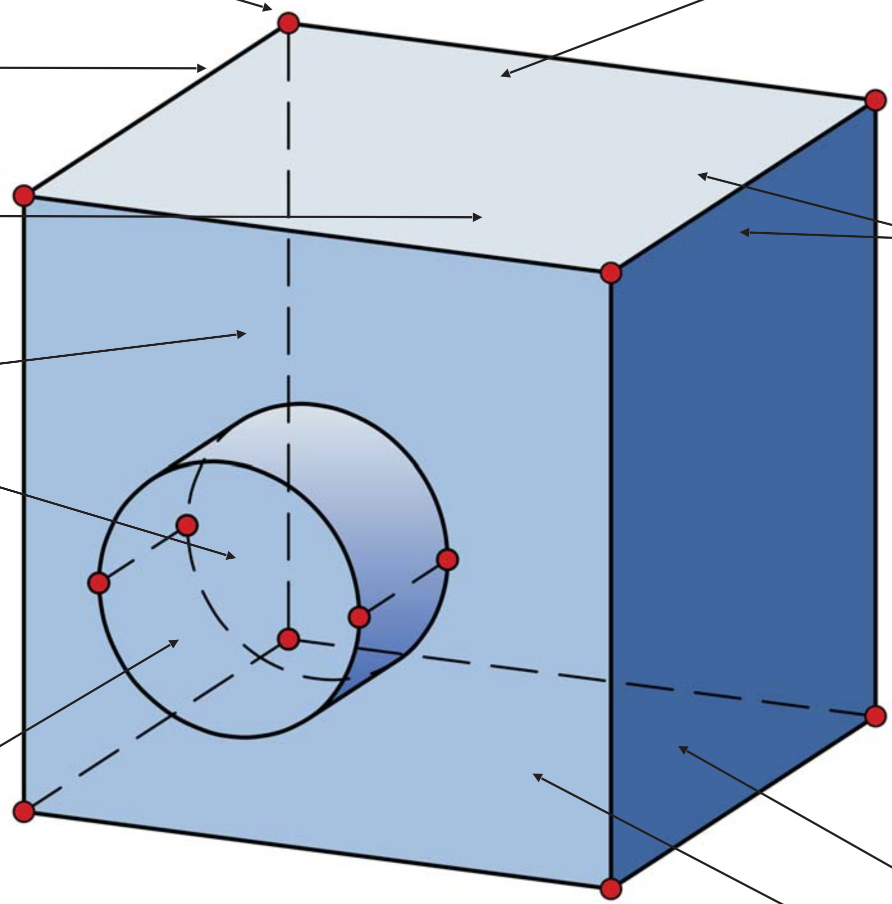
Edges:
Orientation of a curve w.r.t. a loop

Shell:
oriented set of surfaces comprising a volume

- CAD geometry representatio



- CAD geometry representation



points:
(x,y,z) location

Curves:
bounded by two points

Surfaces:
closet set of curves

Volumes:
closet set of surfaces

Wire (loop):
ordered set of curves on surface

Edges:
Orientation of a curve w.r.t. a loop

Solid:

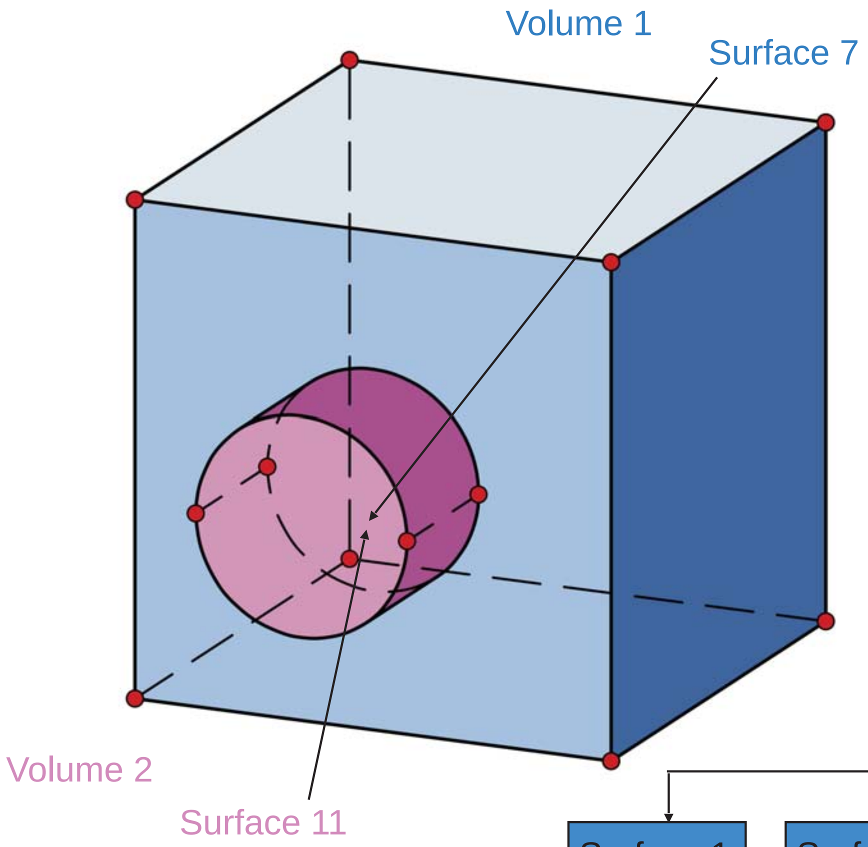
Shell:
oriented set of surfaces comprising a volume

Face:
oriented surface w.r.t. a shell

Body (group):
Collection of solids

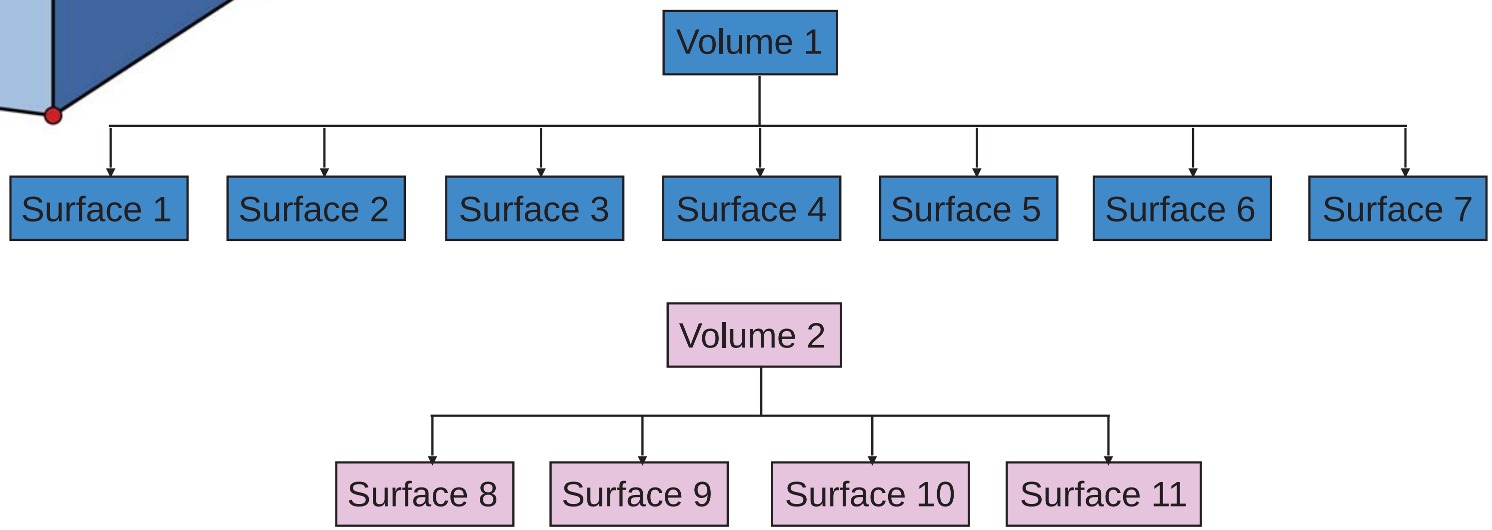
2. Geometry description

- CAD geometry description



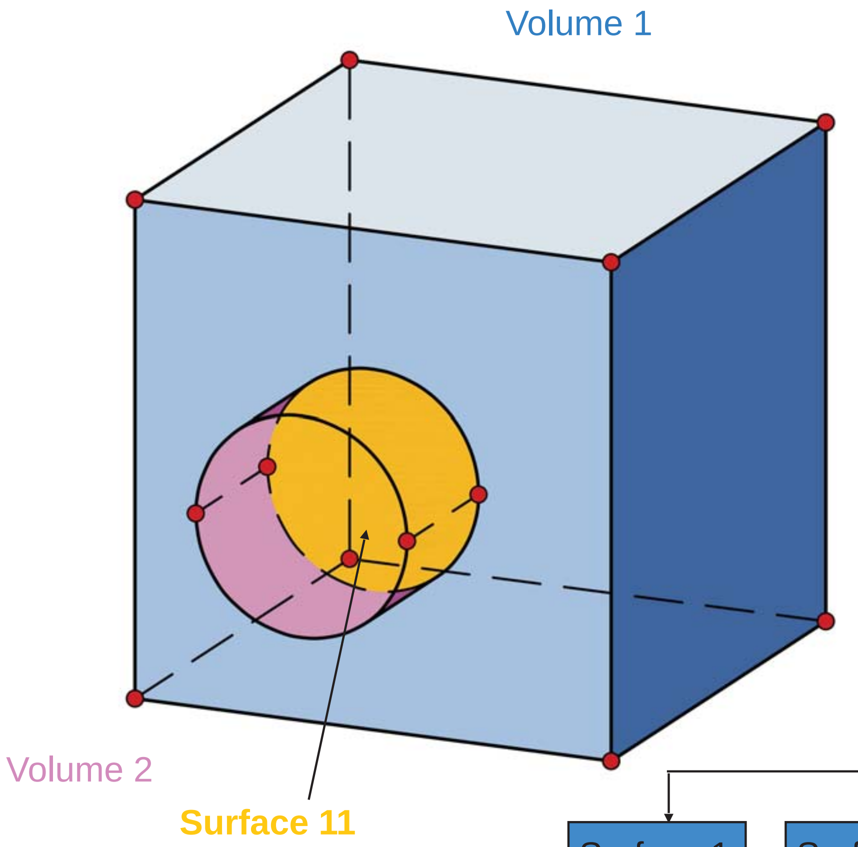
Manifold Geometry:

Each volume maintains its own set of unique surfaces



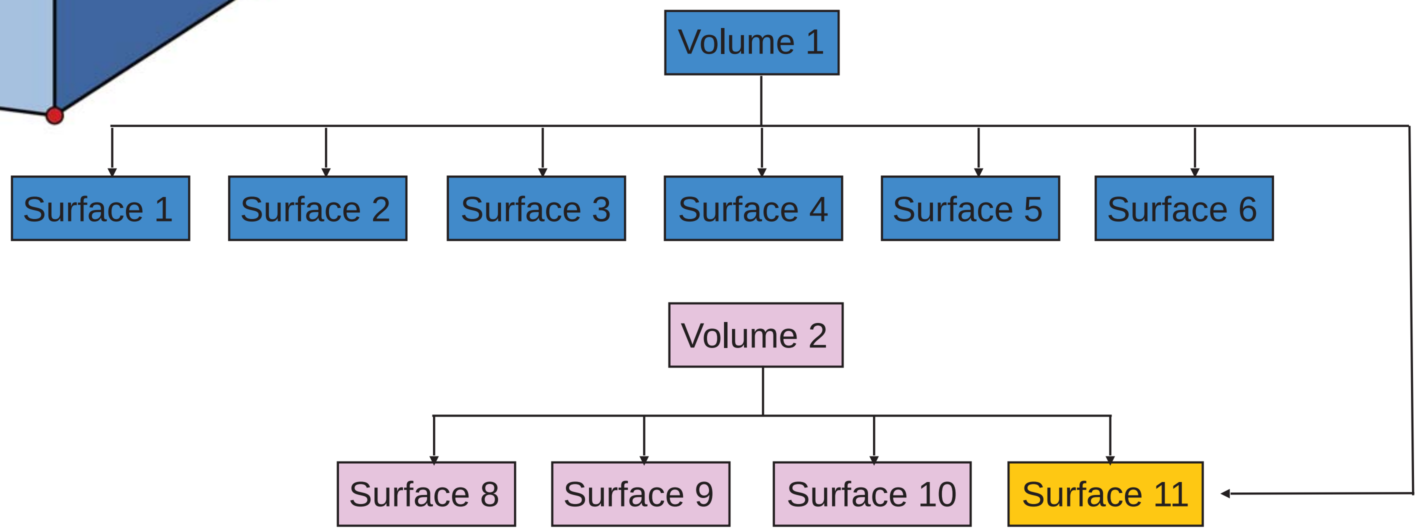
2. Geometry description

- CAD geometry description



Non-Manifold Geometry:

Volumes share matching surfaces



- CAD geometry description
  - **B**oundary **Rep**resentation of the geometry **(Brep)**
  - **Hierarchical** classification of geometrical and topological entities

Entity dimension	Topological classification	Geometrical classification
0-D	vertex	point
1-D	edge wire	curve
2-D	face shell	surface
3-D	solid body	volume

  - From a meshing point of view, we are interested in non-manifold representation of a geometry
  - The hierarchical description of a CAD model can be exploited by the meshing algorithms

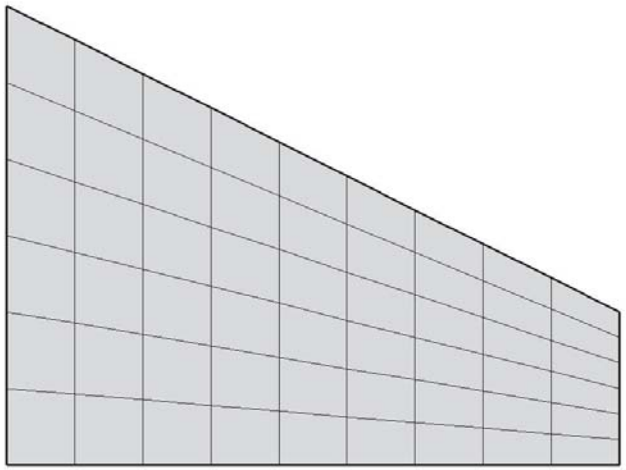
Layout of the course

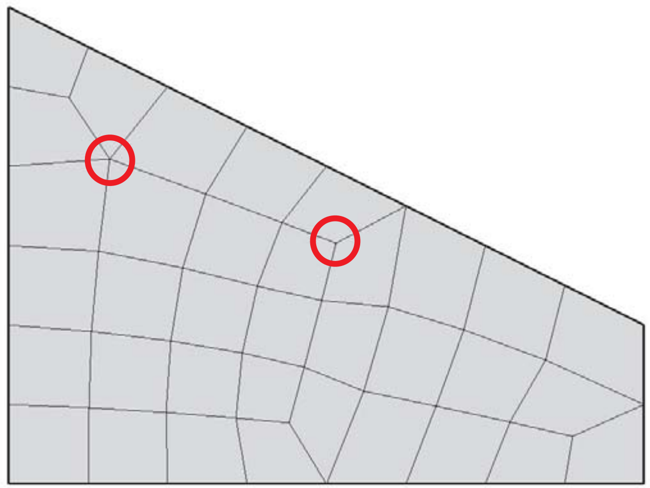
1. Why do we need meshes?
2. Geometry description
3. Classification of mesh generation methods
4. Structured mesh generation methods
5. Unstructured mesh generation methods
6. Mesh optimization and mesh adaption
7. Concluding remarks

3. Classification of mesh generation methods

- **Type of meshes**: depending on the number of adjacent elements to each inner node

Structured mesh (constant) **Unstructured mesh** (non-constant)





Structured vs unstructured	
Domain must verify some constraints	Valid for arbitrary domains
More restrictive for dealing with non-constant element size	More flexible for dealing with non-constant element size
Preferable for aligning elements with boundaries / material properties	Can be used for aligning elements with boundaries / material properties
Easier to develop	More complex to develop

3. Classification of mesh generation methods

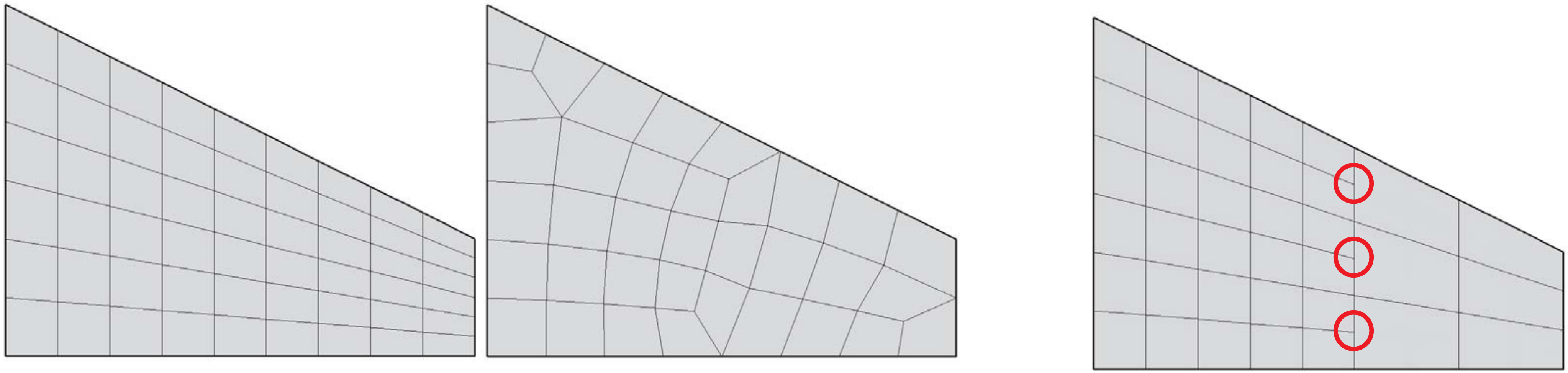
- **Type of meshes**: depending on the intersection between elements

Conformal mesh

- Empty set or an entity of inferior dimension
- There are not any hanging nodes

Non-conformal mesh

- Empty set or part of an entity of inferior dimension
- There are hanging nodes



Conformal vs non-conformal	
Flexible / restrictive for dealing with non-constant element size	More flexible for dealing with non-constant element size
More usual in industry	Less usual in industry

3. Classification of mesh generation methods

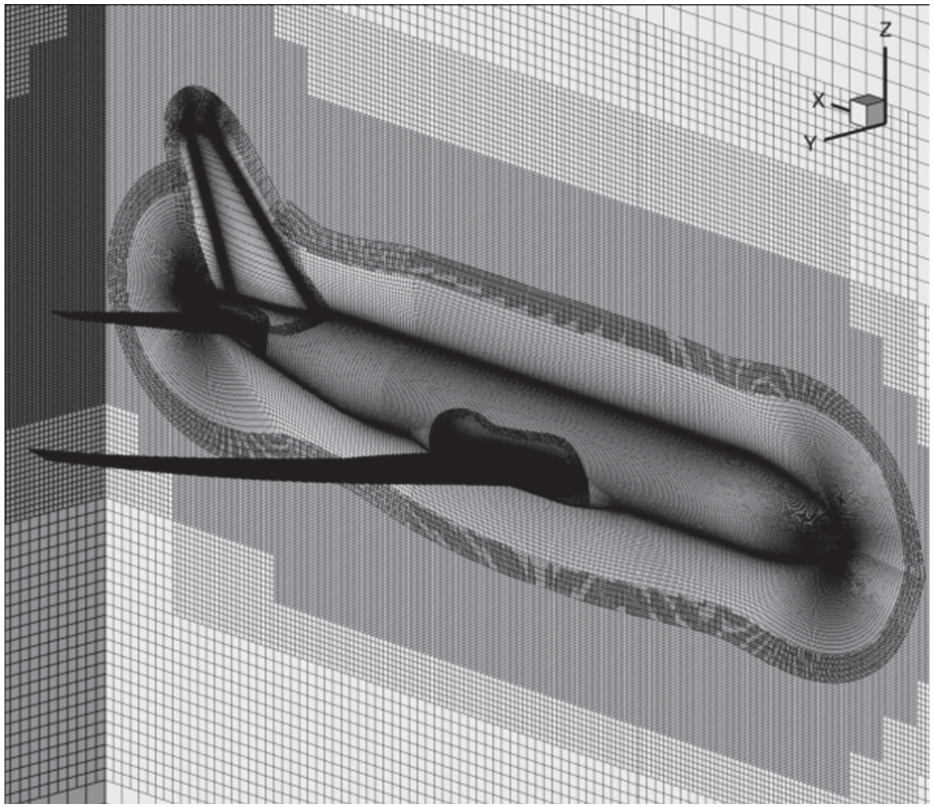
- **Type of meshes**: depending on the intersection between elements

Conformal mesh

- Empty set or an entity of inferior dimension
- There are not any hanging nodes

Non-conformal mesh

- Empty set or part of an entity of inferior dimension
- There are hanging nodes



Non-conformal meshes are useful for:

- structured meshes

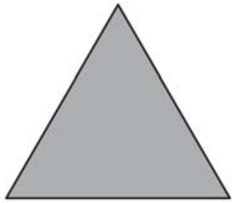
  +

- large gradients in the element size

Image from D. Hue *et.al.*

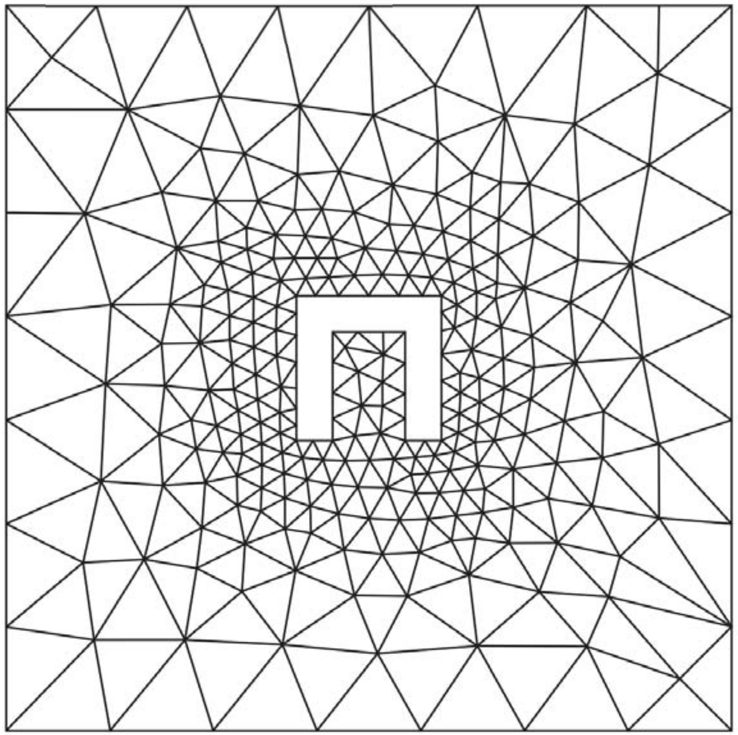
3. Classification of mesh generation methods

- 2D meshes are composed by polyhedral **elements**
- Most common 2D-element **types**:

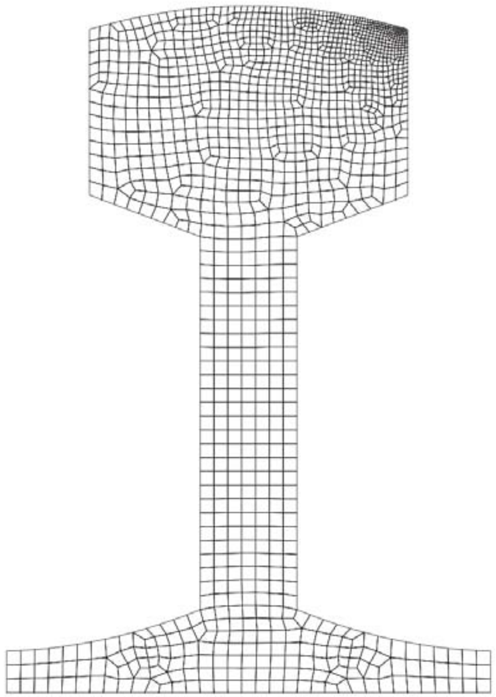


Triangle (3 edges)

Quadrilateral (4 edges)



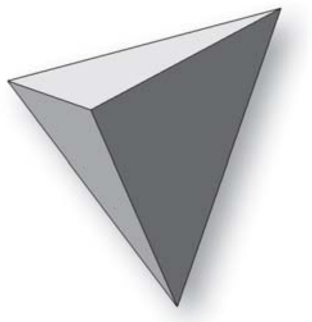
Mesh composed by 245 tris



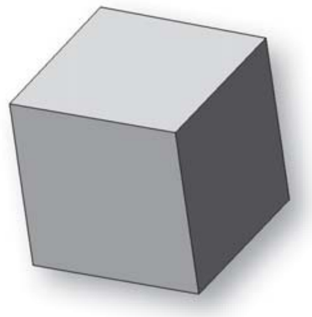
Mesh composed by 1413 quads

3. Classification of mesh generation methods

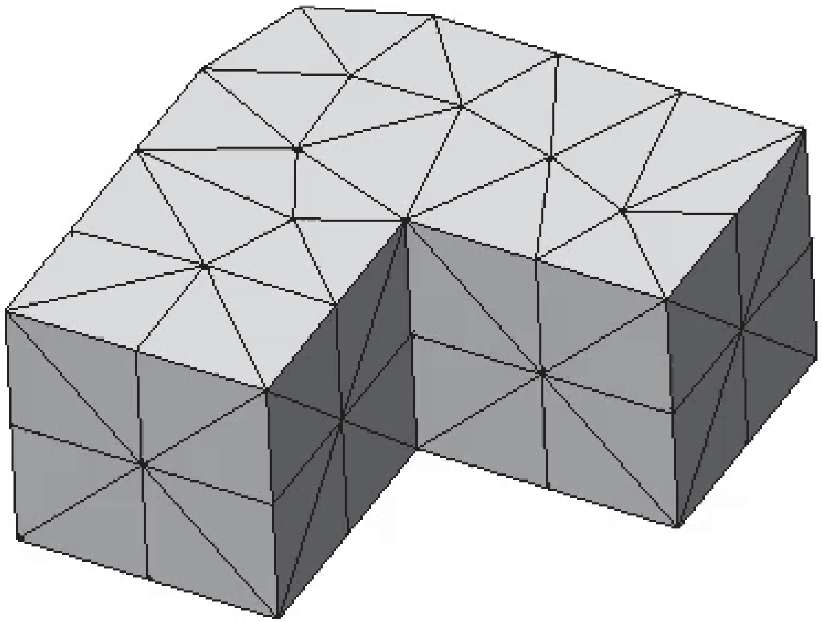
- 3D meshes are composed by polyhedral **elements**
- Most common 3D-element **types**:



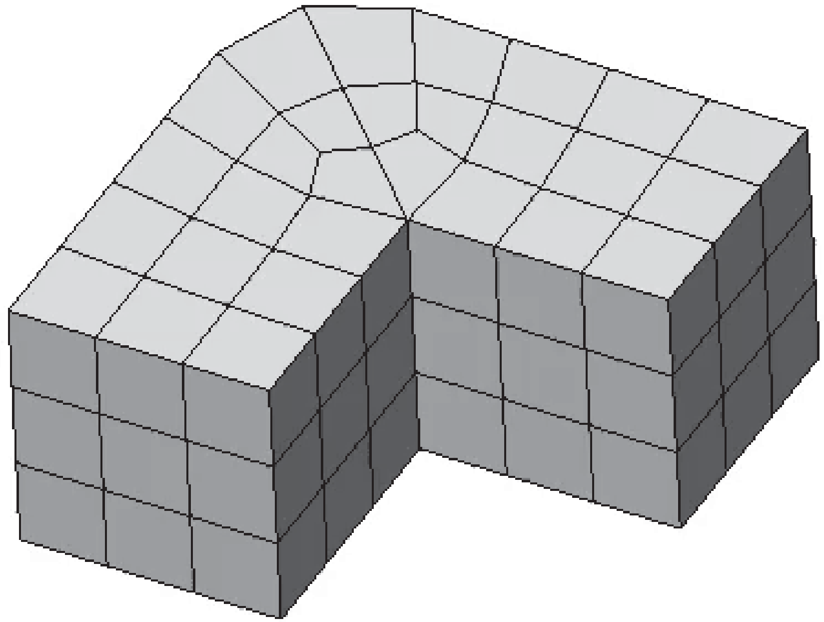
Tetrahedron (4 tris)



Hexahedron (6 quads)



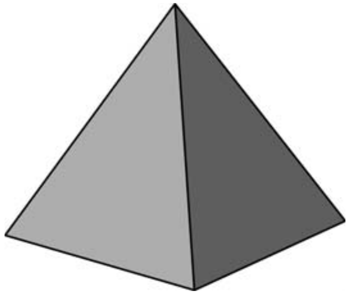
Mesh composed by 168 tetrahedra



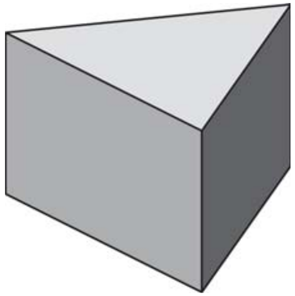
Mesh composed by 84 hexahedra

3. Classification of mesh generation methods

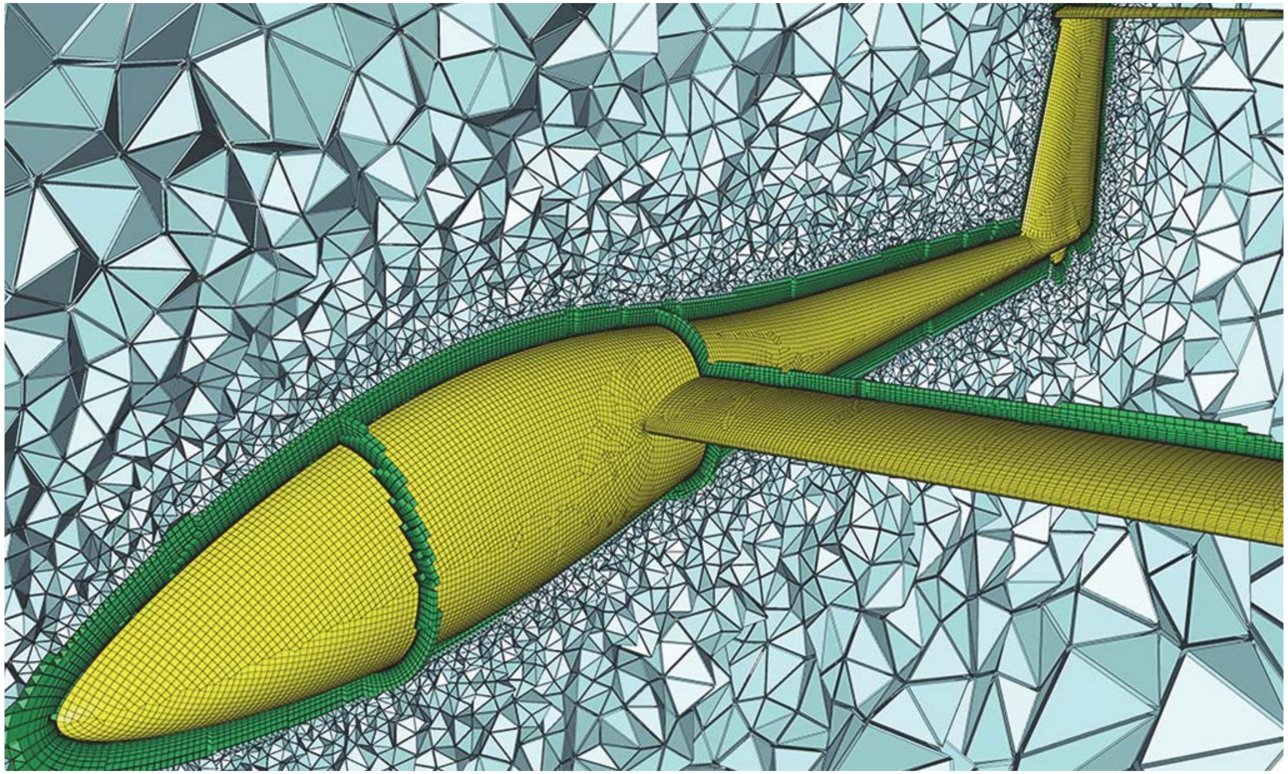
- 3D mixed meshes
- Other 3D elements



Pyramids (4 tris & 1 quad)



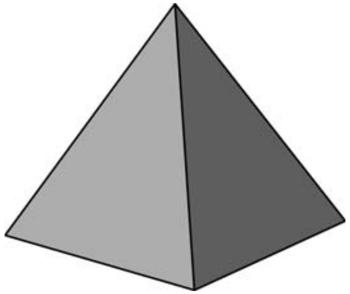
Prisms(2 tris & 3 quad)



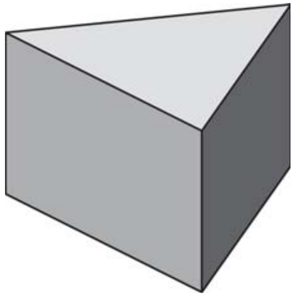
Images from Pointwise (http://www.pointwise.com /)

3. Classification of mesh generation methods

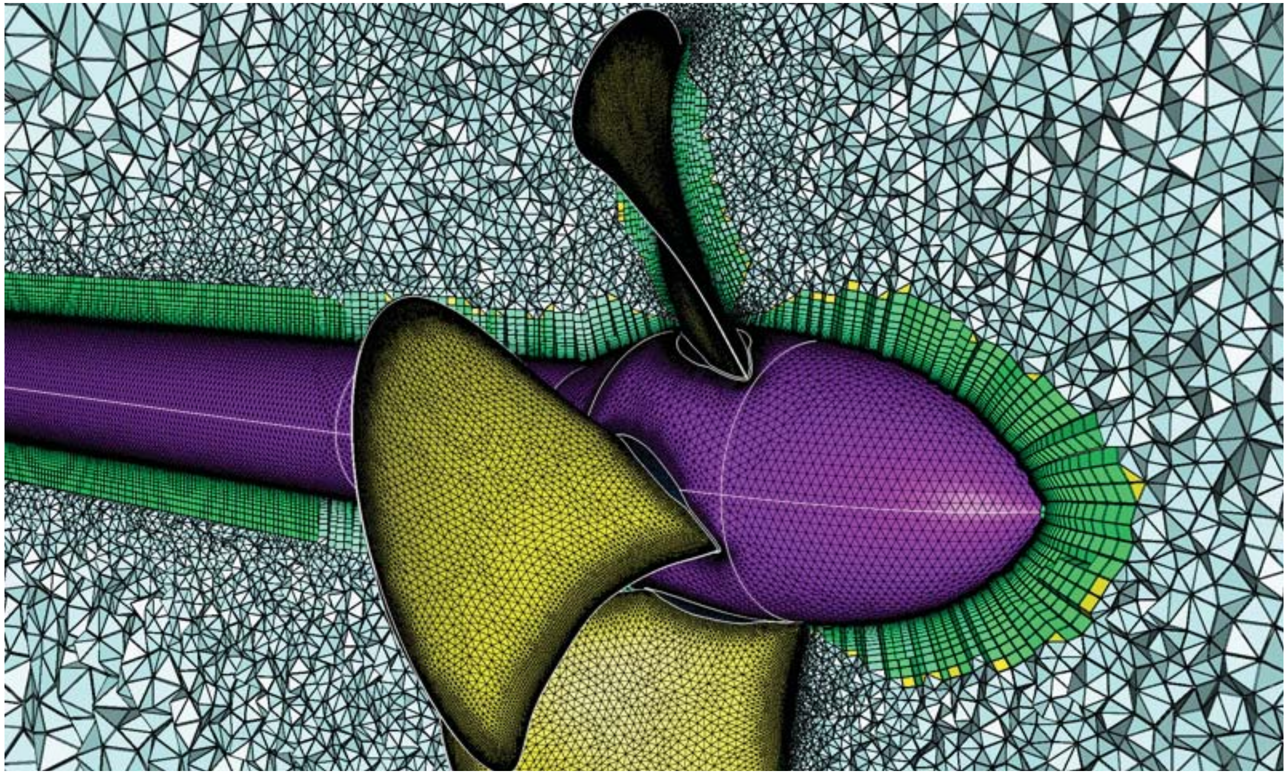
- 3D mixed meshes
- Other 3D elements



Pyramids (4 tris & 1 quad)



Prisms(2 tris & 3 quad)



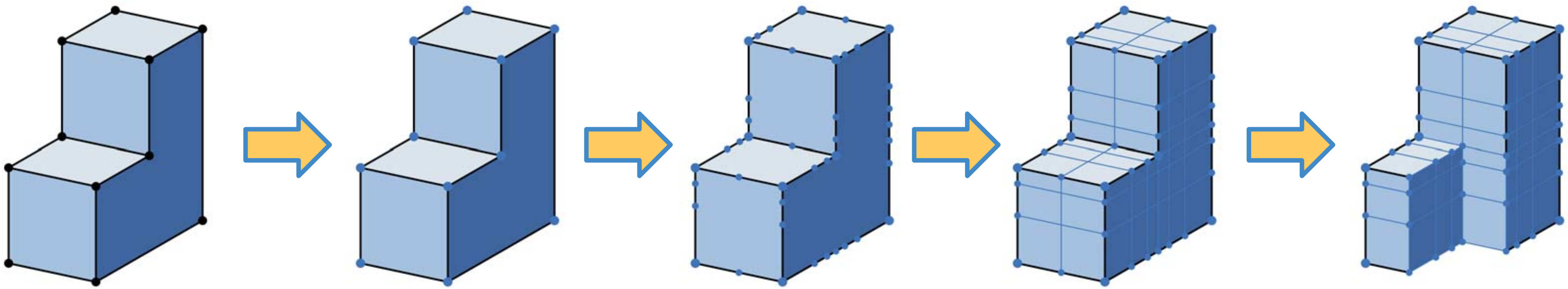
Images from Pointwise (http://www.pointwise.com /)

3. Classification of mesh generation methods

- Hierarchical mesh generation approach

Most of the meshing algorithms follow a bottom-up approach

Initial (adapted/healed) CAD model → Mesh points (0-D entities) → Mesh curves (1-D entities) → Mesh surfaces (2-D entities) → Mesh volumes (3-D entities)



3. Classification of mesh generation methods

- Classification of the meshing algorithms

Meshing Algorithms

- Structured
  - Tris/Tets
  - Quads/Hex
    - Primitives
    - Mapped
      - TFI
      - Elliptic
      - Hyperbolic
    - Decomposition
      - Sub-mapping
      - Sweeping
      - Block decomposition
- Unstructured
  - Tris/Tets
    - Delaunay
    - Advancing front
    - Octree based
  - Quads/Hex
    - Indirect
      - Tri / Tet combine
      - Qmorph / Hmorph
      - Blossom
    - Direct
      - Advancing front
        - Paving / Plastering
        - Unconstrained Plastering
      - Grid based
      - Cross-Field
      - Dual
        - Whisker Weaving
        - Sheet manipulation

Layout of the course

1. Why do we need meshes?
2. Geometry description
3. Classification of mesh generation methods
4. Structured mesh generation methods
5. Unstructured mesh generation methods
6. Mesh optimization and mesh adaption
7. Concluding remarks

4. Structured mesh generation methods

- ## Why Structured meshes

  Structured meshes are still preferred in a wide range of simulations where a strict alignment of elements are required by the analysis:

  - boundary layers in computational fluid dynamics
  - composites in structural dynamics.

- ## Basic property

  Extremely **fast and robust** for **specific** (but common in industry) geometries

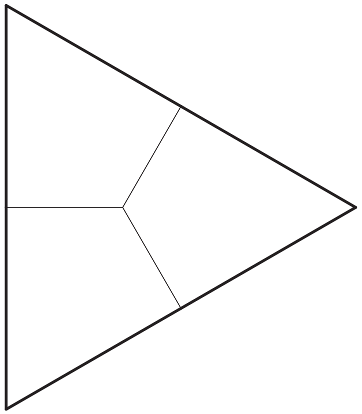
- ## Classification

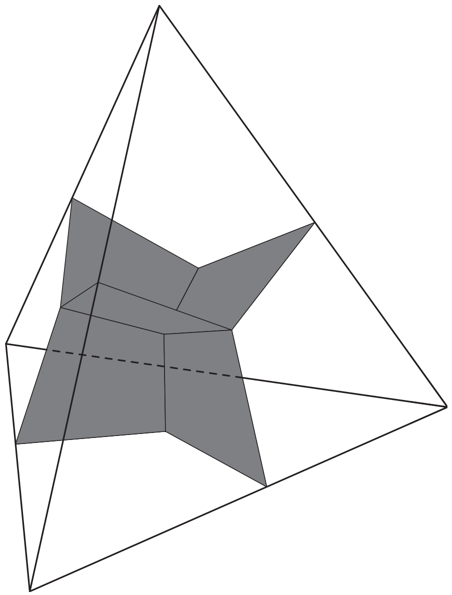
  - Kernel methods
    - Methods based on primitives
    - Methods based on Partial Differential Equations
    - Algebraic interpolation methods (Transfinite Interpolation, TFI)
  - Decomposition methods
    - Submapping
    - Sweeping

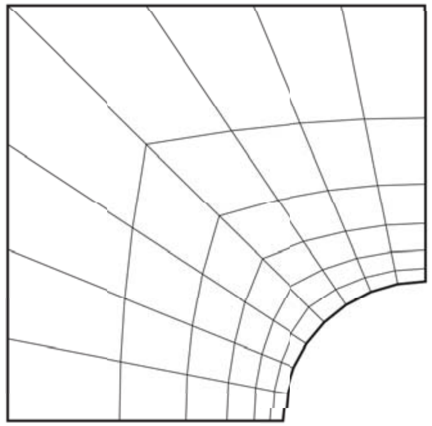
4. Structured mesh generation methods

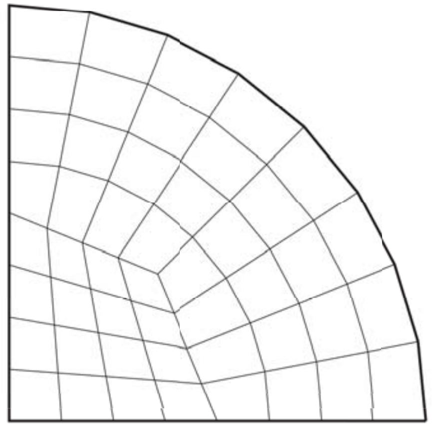
▪ Methods based on primitives

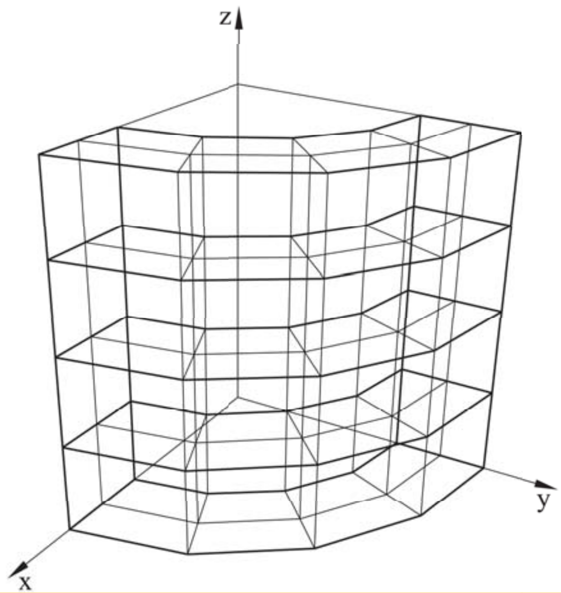
Basic idea: identify simple geometrical shapes and mesh them with a predetermined template











Properties:

- Limited applicability
- High quality meshes
- Implemented in several environments

4. Structured mesh generation methods

- **Methods based on Partial Differential Equations**

  Extremely used in Computational Fluid Dynamics (CFD)

  Used in industry as a kernel in multiblock decomposition

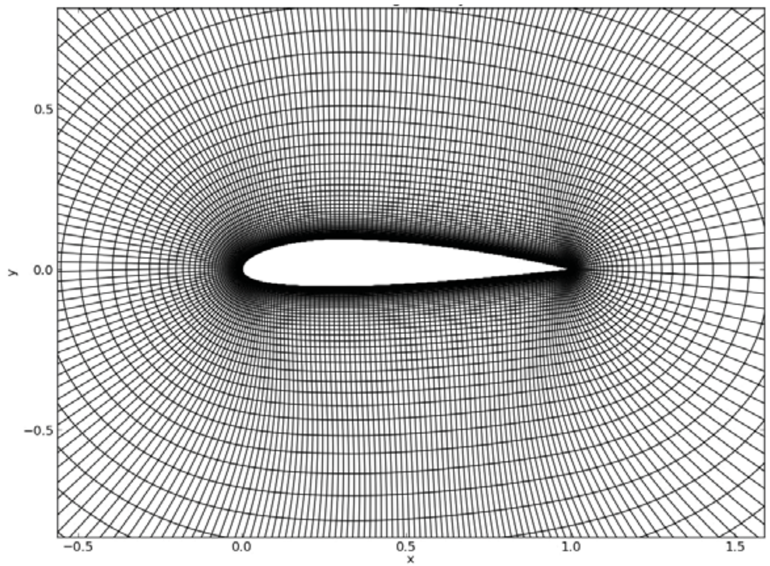


Image from https://sourceforge.net/projects/construct2d

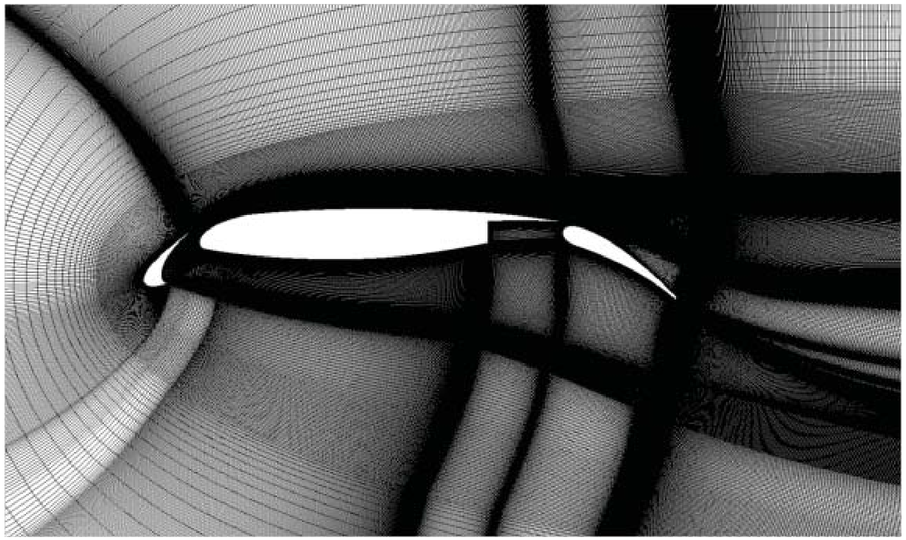


Image from https://bog.poinwise.com

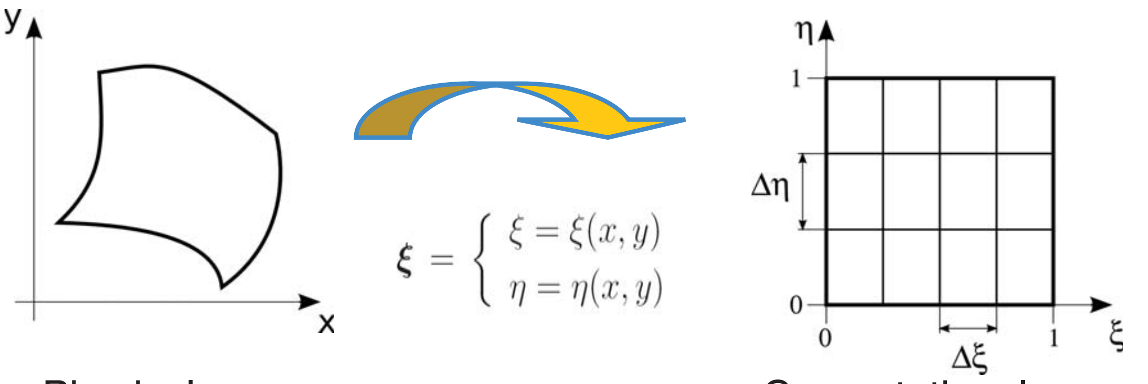
- **Properties**
  - Smooth meshes
  - Control on the orthogonality of the mesh edges
  - Require solving numerically a PDE (cost)

4. Structured mesh generation methods

- Determine a coordinate transformation that maps

  the body-fitted non-uniform non-orthogonal physical space (x,y,z)

  the transformed uniform orthogonal computational space (ξ,η,ξ)



Physical space

Computational space

We require that:

- Any point of the computational space is mapped to a unique point of the physical space (*one-to-one*)
- Each point of the physical space is the image of a point in the computational space (*onto*)

We assume that mapping is smooth and that the **Jacobian is not null**

$$\boxed{L = \begin{vmatrix} \xi_x & \xi_y \\ \eta_x & \eta_y \end{vmatrix} = \xi_x \eta_y - \xi_y \eta_x}$$ (mapping ξ(x,y) is invertible).

4. Structured mesh generation methods

The most common elliptic PDE used for grid generation is the Poisson equation:

$$\boxed{\begin{aligned} \nabla^2\xi &= \xi_{xx} + \xi_{yy} = P(\xi,\eta) \\ \nabla^2\eta &= \eta_{xx} + \eta_{yy} = Q(\xi,\eta) \end{aligned}}$$

where $P(\xi,\eta)$ and $Q(\xi,\eta)$ are used to control the distribution of points:

$P(\xi,\eta)>0$ the points are attracted to the "right"

$P(\xi,\eta)<0$ the points are attracted to the "left"

$Q(\xi,\eta)>0$ the points are attracted to the "top"

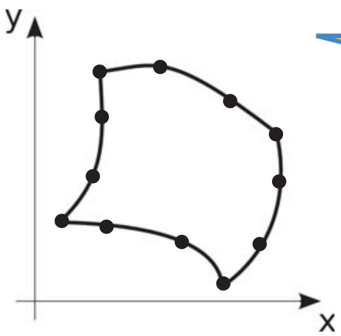
$Q(\xi,\eta)<0$ the points are attracted to the "bottom"

But this is not our objective !!!

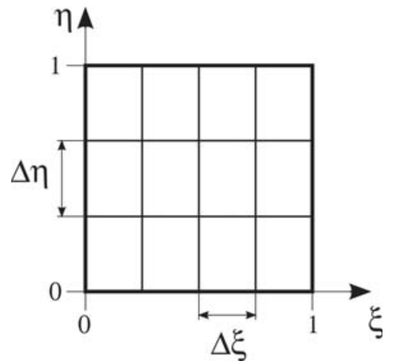
Our goal is to create a mesh in the physical domain by performing all the computations in the uniform rectangular space.

4. Structured mesh generation methods

- Our goal is to create a mesh in the physical domain by performing all the computations in the uniform rectangular space.
- Since function $\boldsymbol{\xi} = (\xi(x,y), \eta(x,y))$ is invertible we can define the inverse transformation $\boldsymbol{x} = (x(\xi,\eta), y(\xi,\eta))$



$$\boldsymbol{x} = \begin{cases} x = x(\xi,\eta) \\ y = y(\xi,\eta) \end{cases}$$



Physical space

Computational space

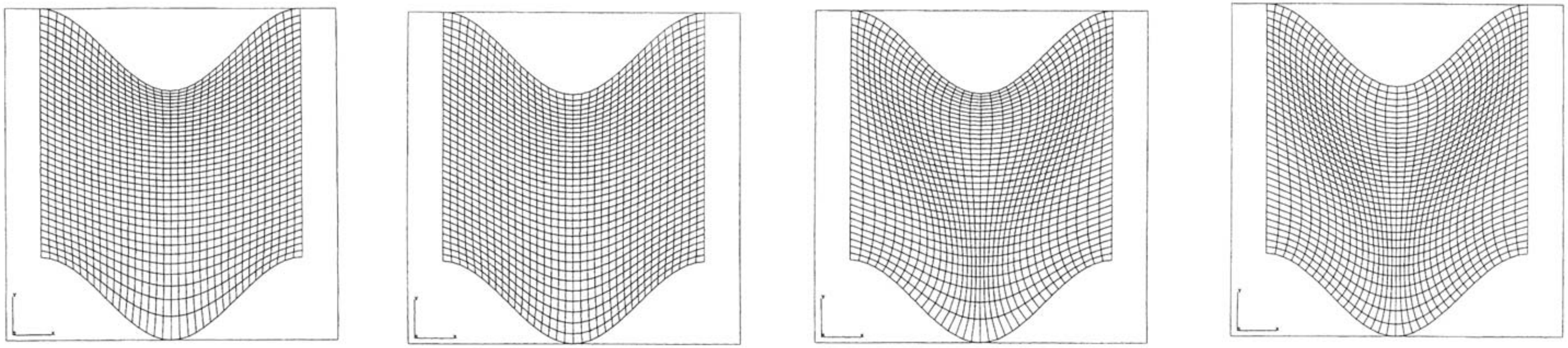
- After some analysis we get

$$\boxed{\begin{aligned} \alpha x_{\xi\xi} - 2\beta x_{\xi\eta} + \gamma x_{\eta\eta} + J^2(P x_\xi + Q x_\eta) = 0 \\ \alpha y_{\xi\xi} - 2\beta y_{\xi\eta} + \gamma y_{\eta\eta} + J^2(P y_\xi + Q y_\eta) = 0 \end{aligned}}$$

where $\left\{ \begin{aligned} \alpha &= x_\eta^2 + y_\eta^2 \\ \beta &= x_\xi x_\eta + y_\xi y_\eta \\ \gamma &= x_\xi^2 + y_\xi^2 \end{aligned} \right.$ and $J = \begin{vmatrix} x_\xi & x_\eta \\ y_\xi & y_\eta \end{vmatrix} = x_\xi y_\eta - x_\eta y_\xi$

4. Structured mesh generation methods

- Laplacian method can be modified in order to control the shape of the grid



Laplacian grid | Grid with cell height control | Grid with boundary orthogonality control | Grid with cell height and boundary orthogonality control

Images from “Handbook of grid generation”, Thompson et al.

- Other types of PDE are also used: hyperbolic equations
- Extended to 3D problems

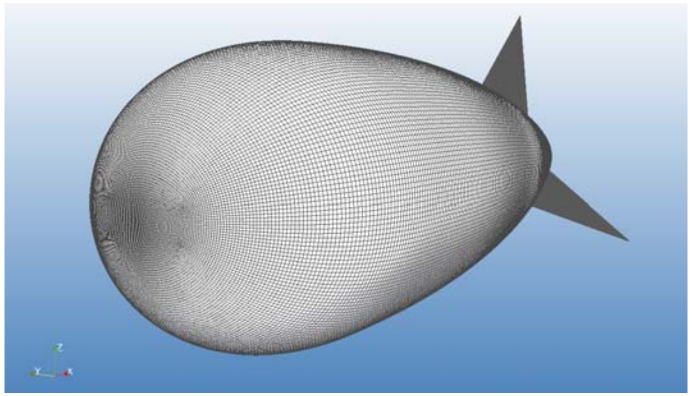


Image from http://rtech-engineering.com

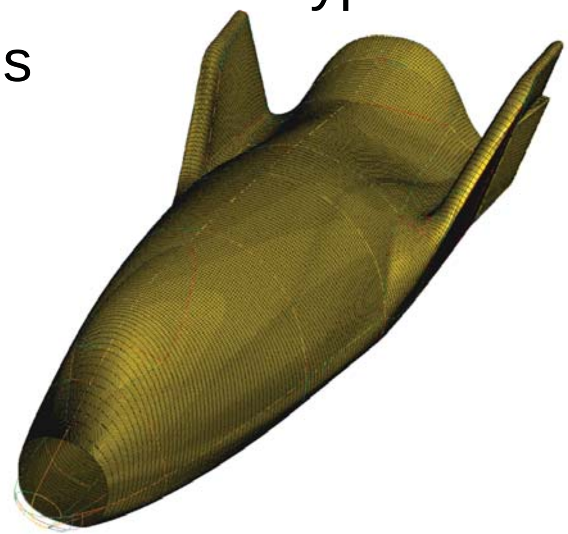


Image from J. P. Steinbrenner & J.R. Chawner

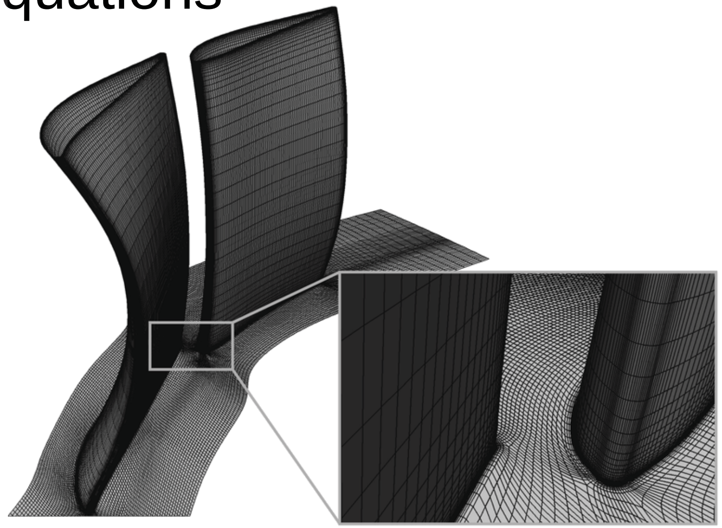
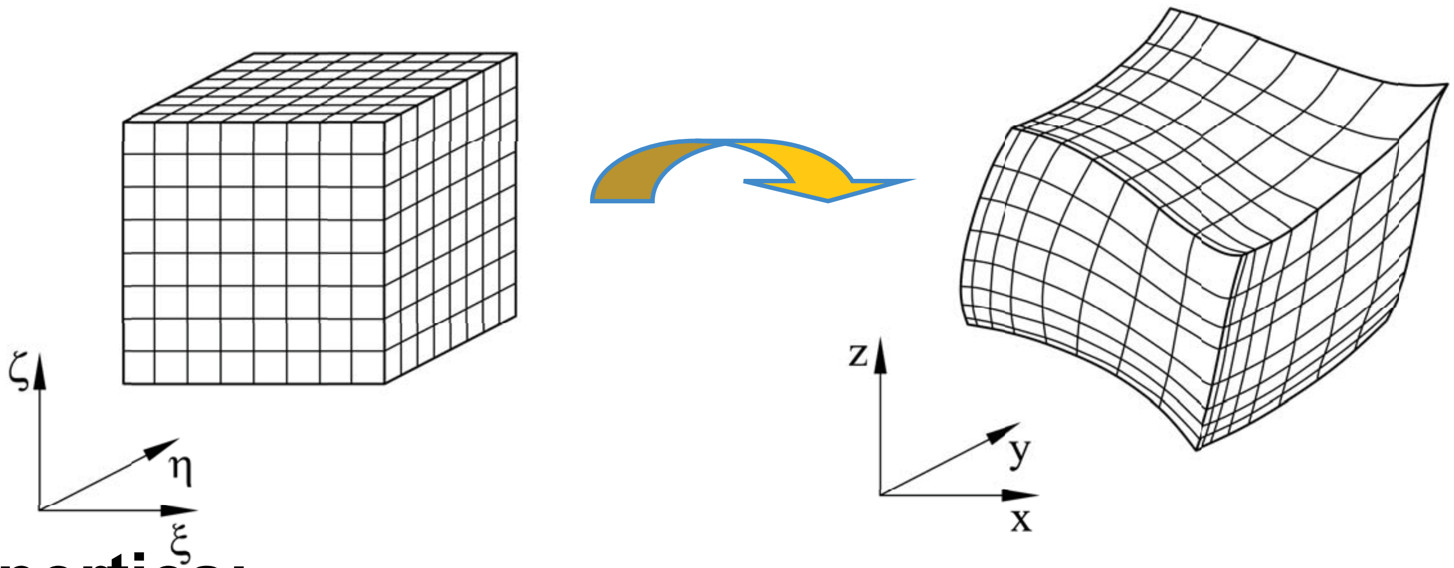


Image from R. Schalps , S. Shahpar & V. Gümmer

4. Structured mesh generation methods

- **Algebraic methods (Transfinite Interpolation, TFI)**

Transformations from a rectangular computational domain to an arbitrarily shaped physical domain delimited by 4-logical sides (2D) or 6-logical sides (3D)



- **Properties:**
  - Fast (compared with PDE based methods)
  - Direct control on the node location
  - Less control on the grid smoothness

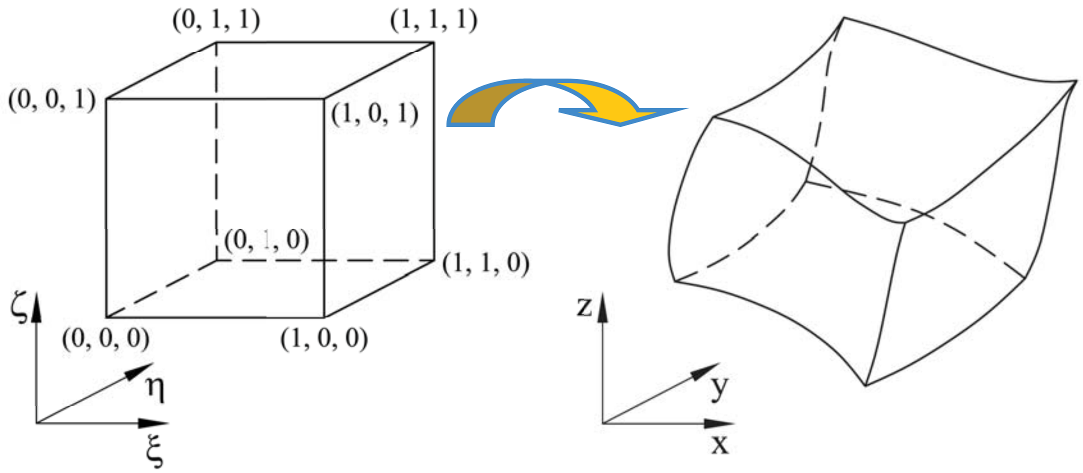
4. Structured mesh generation methods

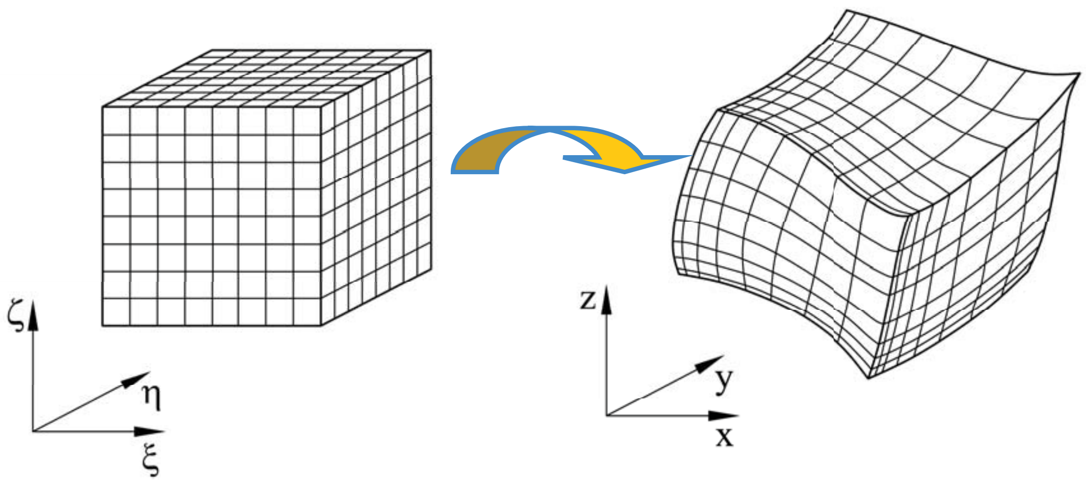
- Mapping definition

$$\mathbf{X}(\xi,\eta,\zeta) = \begin{bmatrix} x(\xi,\eta,\zeta) \\ y(\xi,\eta,\zeta) \\ z(\xi,\eta,\zeta) \end{bmatrix}$$

$$0 \le \xi \le 1 \qquad 0 \le \eta \le 1 \qquad 0 \le \zeta \le 1$$

Computational domain | Physical domain





The image of a discrete subset (ξ_I, η_J, ζ_K)

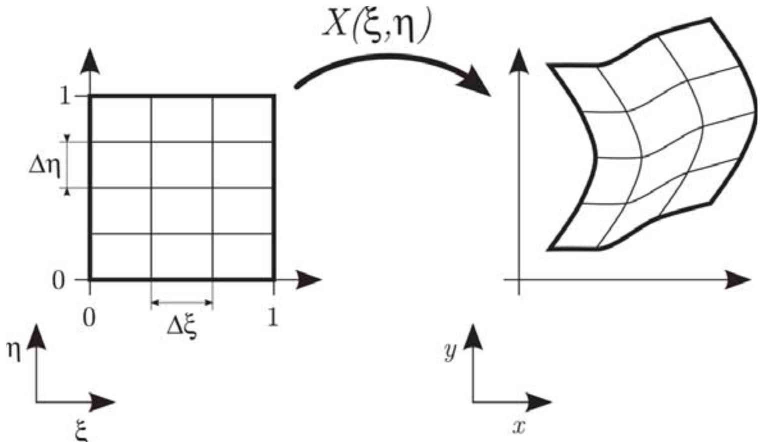
$$\mathbf{X}(\xi_I,\eta_J,\zeta_K) = \begin{bmatrix} x(\xi_I,\eta_J,\zeta_K) \\ y(\xi_I,\eta_J,\zeta_K) \\ z(\xi_I,\eta_J,\zeta_K) \end{bmatrix}$$

with
$$\begin{cases} 0 \le \xi_I = \dfrac{I-1}{\hat{I}-1} \le 1 \\ 0 \le \eta_J = \dfrac{J-1}{\hat{J}-1} \le 1 \\ 0 \le \zeta_K = \dfrac{K-1}{\hat{K}-1} \le 1 \end{cases}$$

being
$$I = 1, 2, \ldots, \hat{I}$$
$$J = 1, 2, \ldots, \hat{J}$$
$$K = 1, 2, \ldots, \hat{K}$$

is a structured mesh

- **Linear TFI in 2D**



TFI sets an univariate interpolation in each direction of the computational space

$$
\begin{aligned}
\mathbf{U}(\xi,\eta) &= \sum_{i=1}^{2} \alpha_i(\xi)\mathbf{X}(\xi_i,\eta) \\
\mathbf{V}(\xi,\eta) &= \sum_{j=1}^{2} \beta_j(\eta)\mathbf{X}(\xi,\eta_j)
\end{aligned}
$$

For the linear TFI the blending functions are (other types can be used: cubic Hermite,…)

$$
\begin{cases} \alpha_1(\xi) = 1-\xi \\ \alpha_2(\xi) = \xi \end{cases} \qquad \begin{cases} \beta_1(\eta) = 1-\eta \\ \beta_2(\eta) = \eta \end{cases}
$$

We also consider the tensor product on these univariate interpolation

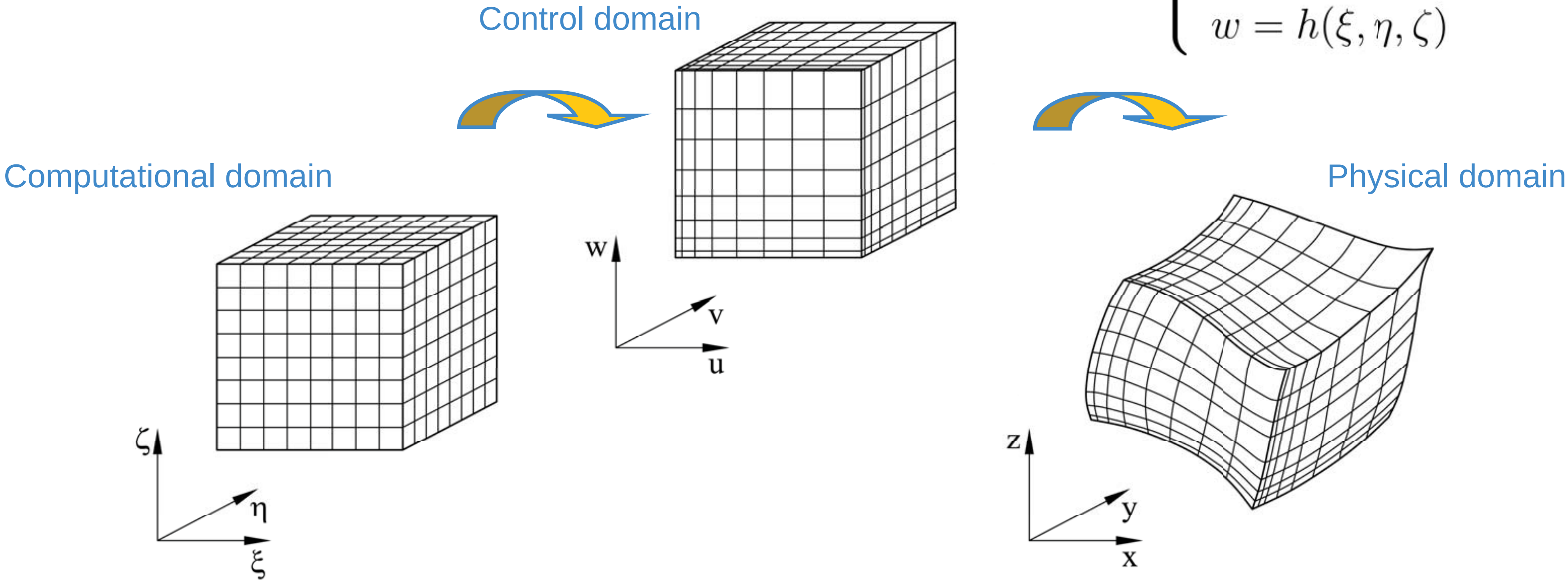
$$
\mathbf{UV}(\xi,\eta) = \sum_{i=1}^{2}\sum_{j=1}^{2} \alpha_i(\xi)\beta_j(\eta)\,\mathbf{X}(\xi_i,\eta_j)
$$

Finally, the transfinite mapping as the Boolean sum of the two interpolation

$$
\boxed{\mathbf{X}(\xi,\eta) = \mathbf{U}(\xi,\eta) \oplus \mathbf{V}(\xi,\eta) = \mathbf{U}(\xi,\eta) + \mathbf{V}(\xi,\eta) - \mathbf{UV}(\xi,\eta)}
$$

- **Grid spacing control**
  - The spacing between points in the physical domain is controlled by blending functions: $\alpha_i(\xi)$ and $\beta_j(\eta)$
  - Two approaches are used to control the spacing between points:
    - Design a **blending function** to generate the desired grid concentration
    - To define an **intermediate control domain** between the computational and physical domains
  - We define the intermediate control space according to

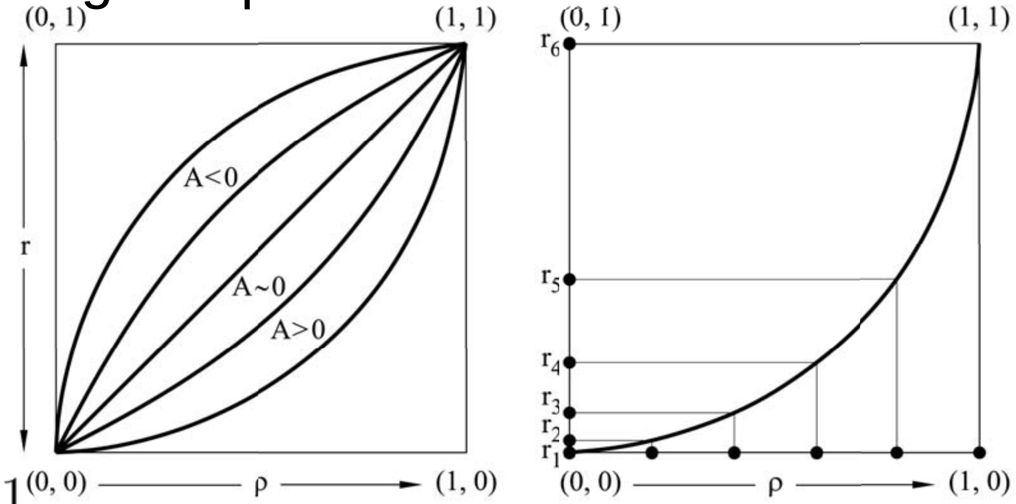
$$\begin{cases} u = f(\xi, \eta, \zeta) \\ v = g(\xi, \eta, \zeta) \\ w = h(\xi, \eta, \zeta) \end{cases}$$



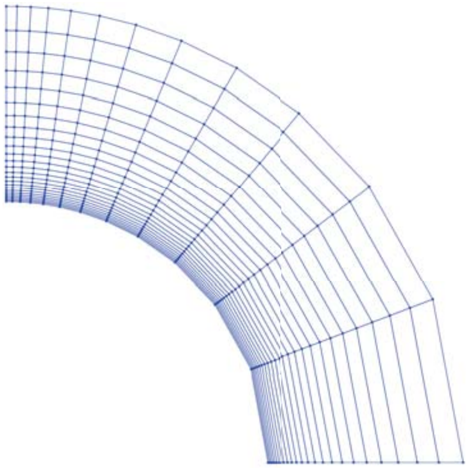
4. Structured mesh generation methods

- There exist a wide range of functions to define the intermediate space (the spacing of grid points).
- One of the simplest choices is the single-exponential function

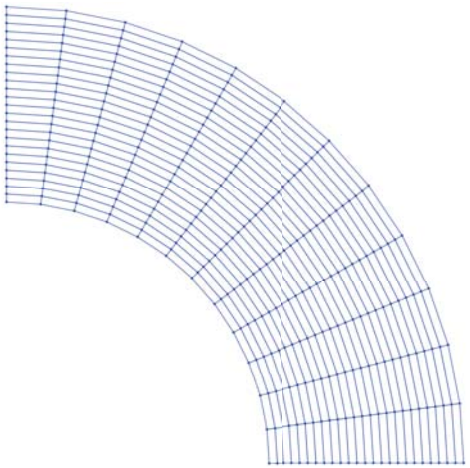
$$\boxed{r = \frac{e^{A\rho} - 1}{e^{A} - 1}}$$



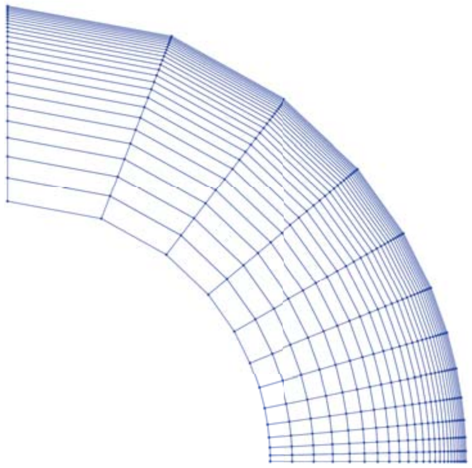
that maps $0 \leq \rho \leq 1$ onto $0 \leq r \leq 1$



A=-3



A=0

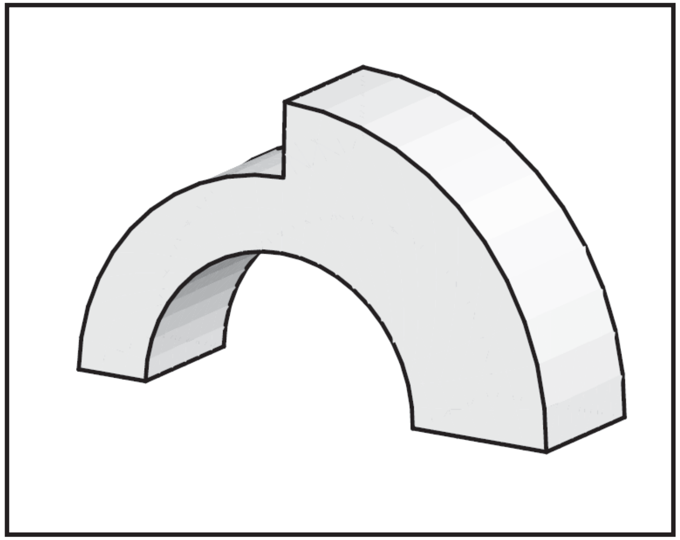


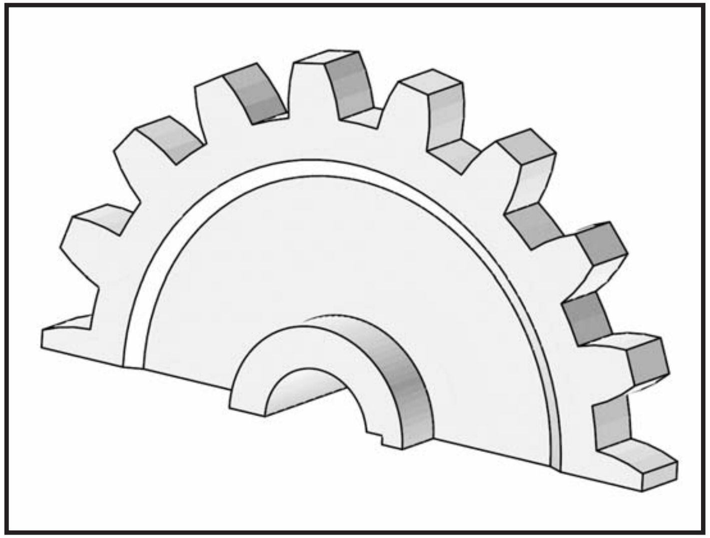
A= 3

4. Structured mesh generation methods

- **Submapping**

A method to decompose and mesh a “blocky” geometry into simple pieces that are equivalent to a quadrilateral (2D) or a box (3D)





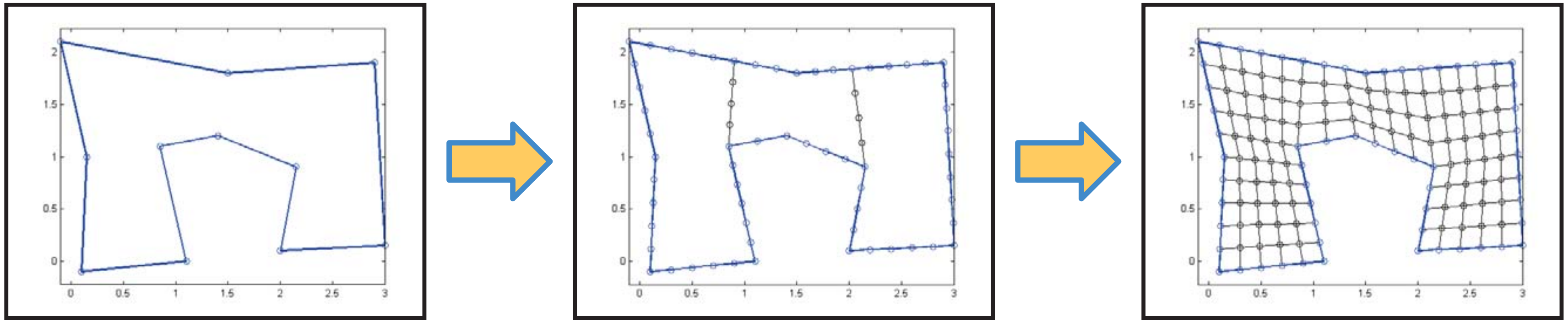
Properties:

- Full automatic decomposition
- Fast
- The decomposition leads to a compatible meshing of blocks

Overview of the algorithm

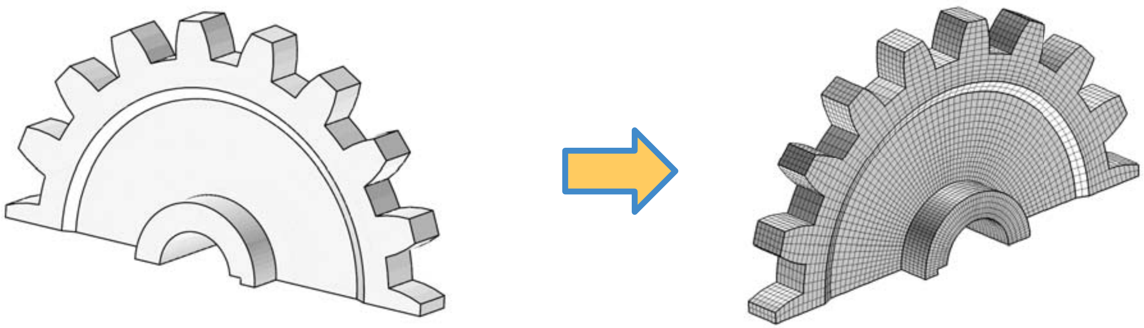
The submapping algorithm can be divided into two steps:

- **Decomposition** of the geometry into blocks ensuring a compatibility between patches
- **Discretization** of each patchusing kernel methods (PDE's, TFI,…)



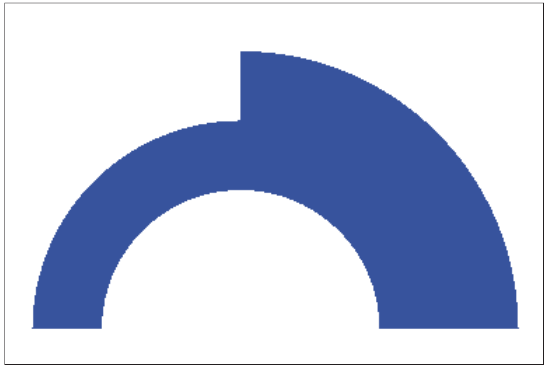
Initial blocky geometry | Decomposed geometry | Meshed geometry

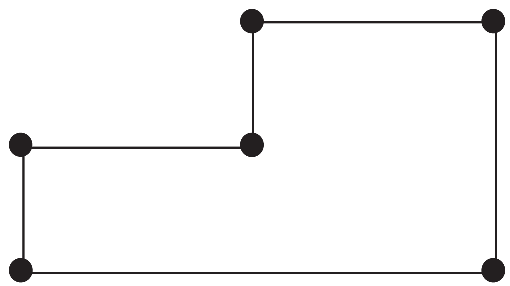
Important: fully automated process !!!
(the user only prescribes the element size)

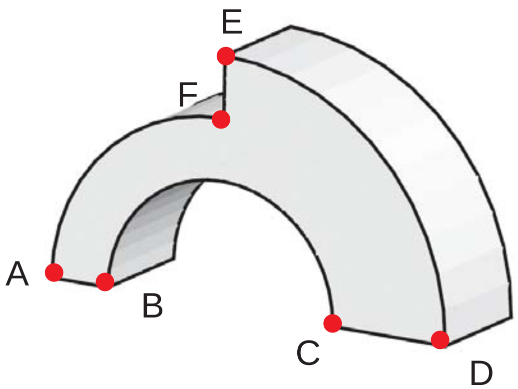


4. Structured mesh generation methods

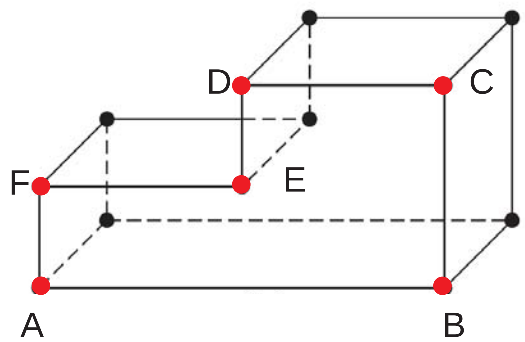
- Two representations of the geometry are used:
  - The **physical space** is the initial representation of the geometry
  - The **computational space** is a representation of the initial geometry in which each edge is parallel to the coordinate axes.





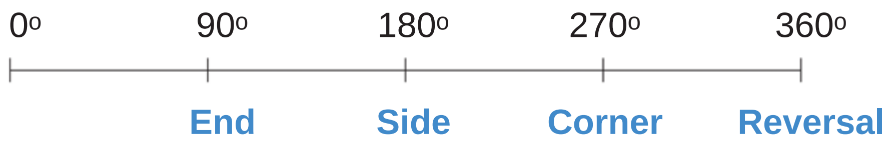


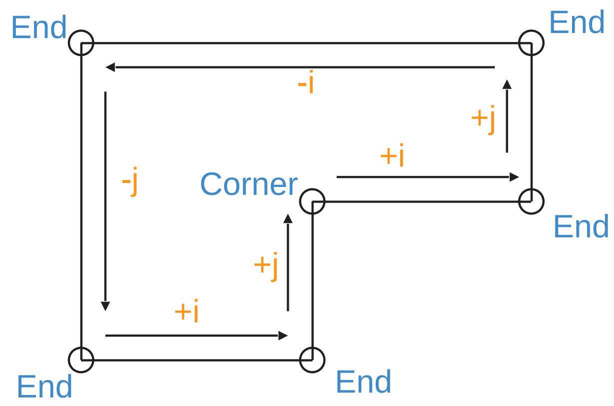
Physical domain

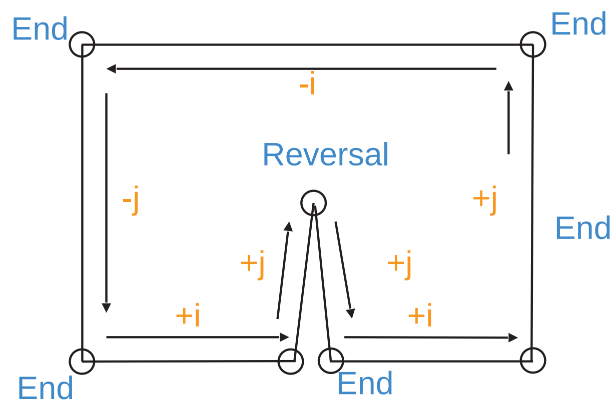


Computational domain

- ## Vertex classification:
  - **End**: Inner angle close to 90º.
  - **Side**: Inner angle close to 180º
  - **Corner**: Inner angle close to 270º.
  - **Reversal**: Inner angle close to 360º







- ## Edge classification
  - **+i**: The edge is horizontal and goes from left to right.
  - **-i** : The edge is horizontal and goes from right to left.
  - **+j**: The edge is vertical and goes from down to up.
  - **-j** : The edge is vertical and goes from up to down.

4. Structured mesh generation methods

- For a structured mesh, we impose the compatibility conditions:

$$\boxed{\begin{aligned} \sum_{e \in I^+} n_e &= \sum_{e \in I^-} n_e, \\ \sum_{e \in J^+} n_e &= \sum_{e \in J^-} n_e, \end{aligned}}$$

n_e is the number of elements on edge *e*.

- To keep the number of elements to a reasonable number, we will solve the following **integer linear programming** problem:

$$\begin{cases} \min \displaystyle\sum_{i=1}^{i=N_{edges}} n_e \\ \displaystyle\sum_{e \in I^+} n_e = \sum_{e \in I^-} n_e, \\ \displaystyle\sum_{e \in J^+} n_e = \sum_{e \in J^-} n_e, \\ n_e \geq N_e, \\ M \geq \dfrac{n_e}{N_e}. \end{cases}$$

More sophisticated objective function can be defined to generate more accurate node distributions

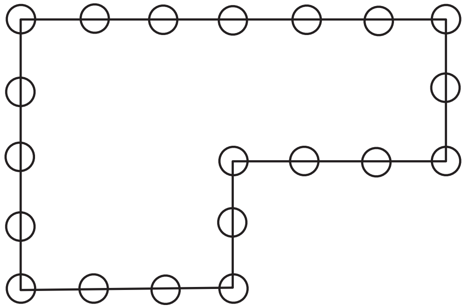
We have at least N_e elements on edge e

Upper limit on the number of elements on edge *e*

The solution to this problem provides a mesh for the boundary that accepts a structured mesh in the interior.

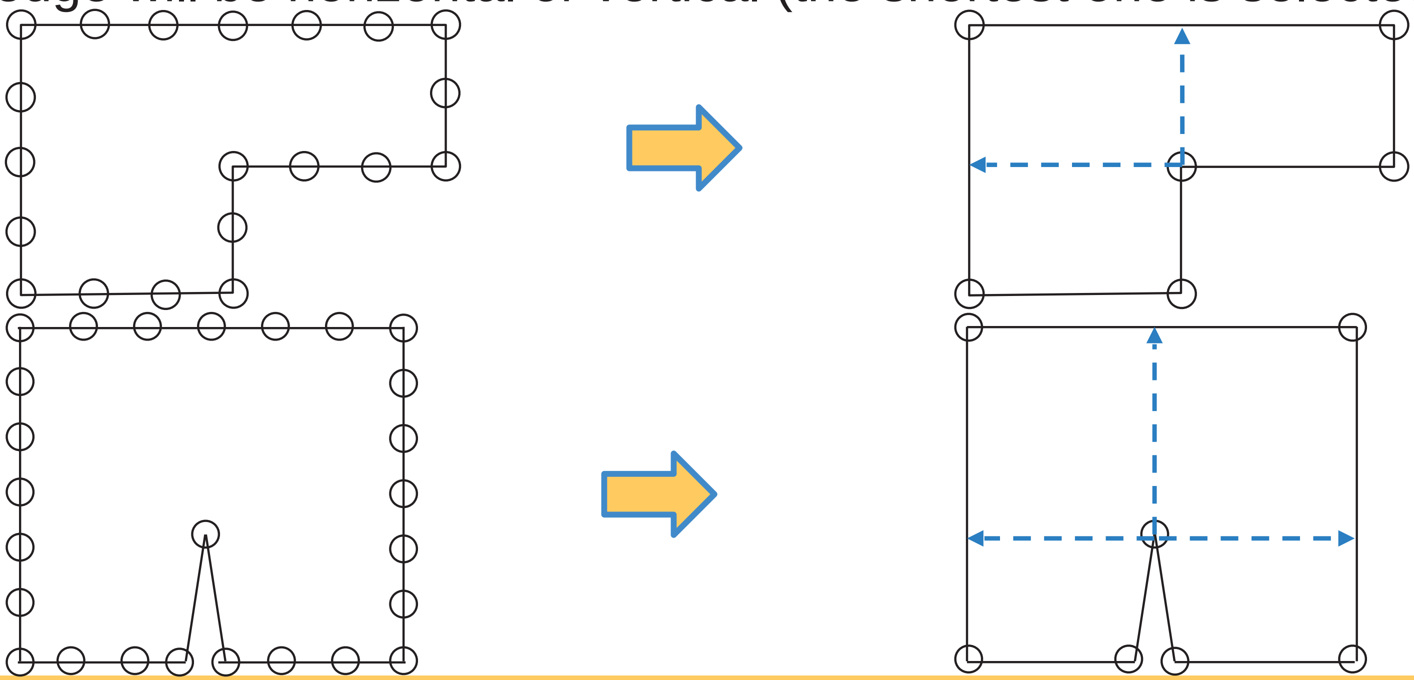
4. Structured mesh generation methods

- **Creation of the computational domain**: once the number of intervals on each edge is computed we have to create them in the computational space

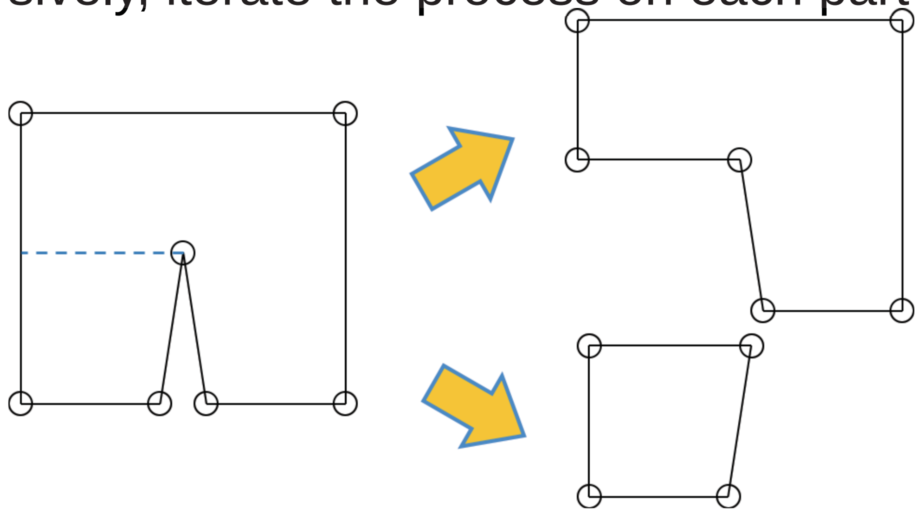


- **Selecting a cutting edge**: the start of a cutting edge will be a node classified as **corner** or **reversal**

The cutting edge will be horizontal or vertical (the shortest one is selected)

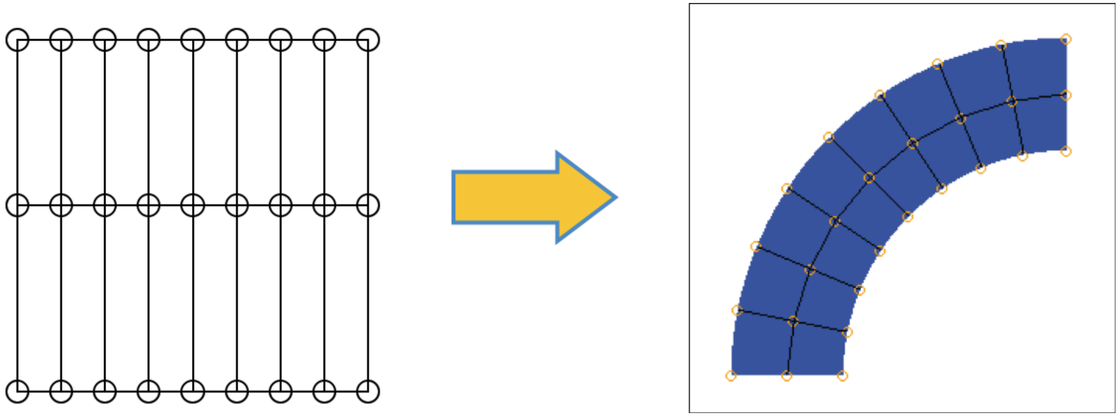


- **Splitting the domain**: once a cutting edge is found, we proceed to split the domain and, recursively, iterate the process on each part



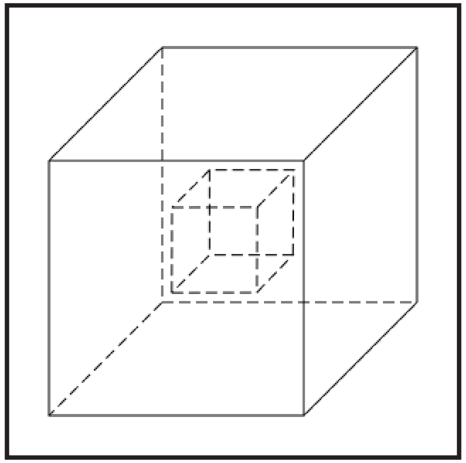
The process of decomposition ends when there are no corner nor reversal nodes

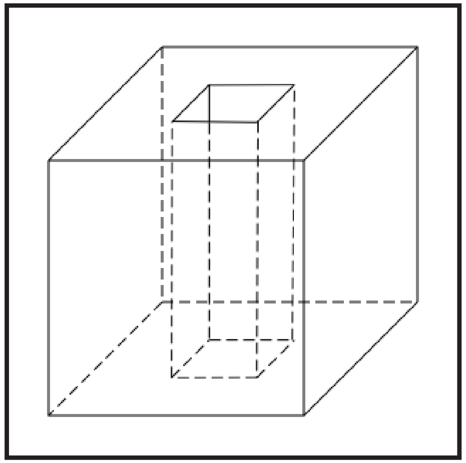
- **Discretization of each subdomain**: we mesh each subdomain of the geometry using a classical structured mesh generator, for example TFI



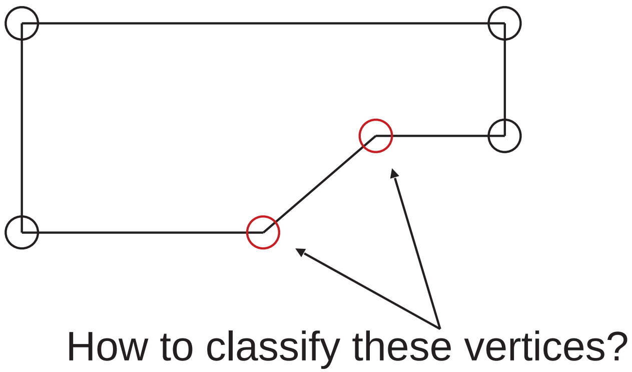
4. Structured mesh generation methods

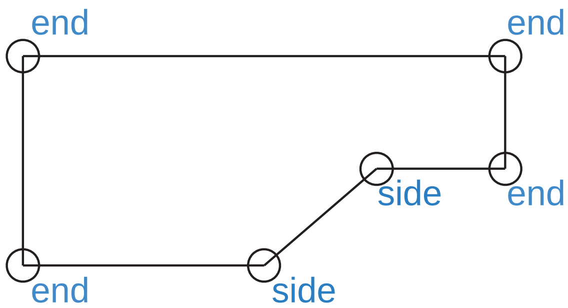
- **Advanced issues**
  - Domains with holes: how to locate them?





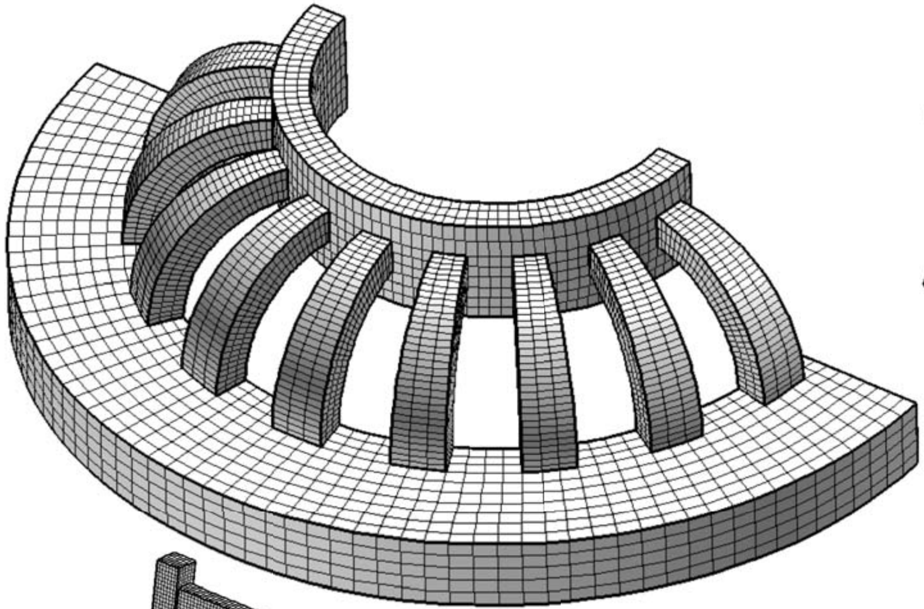
  - Ensuring a correct classification of the vertices

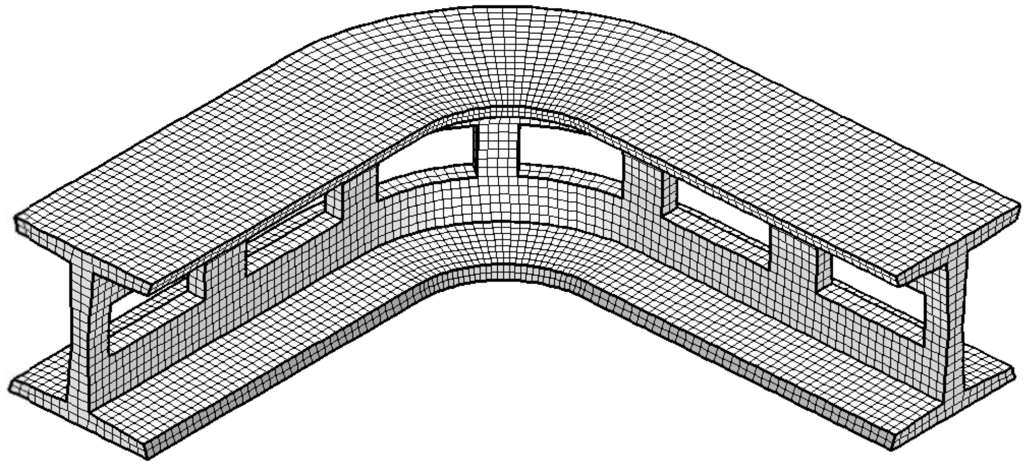


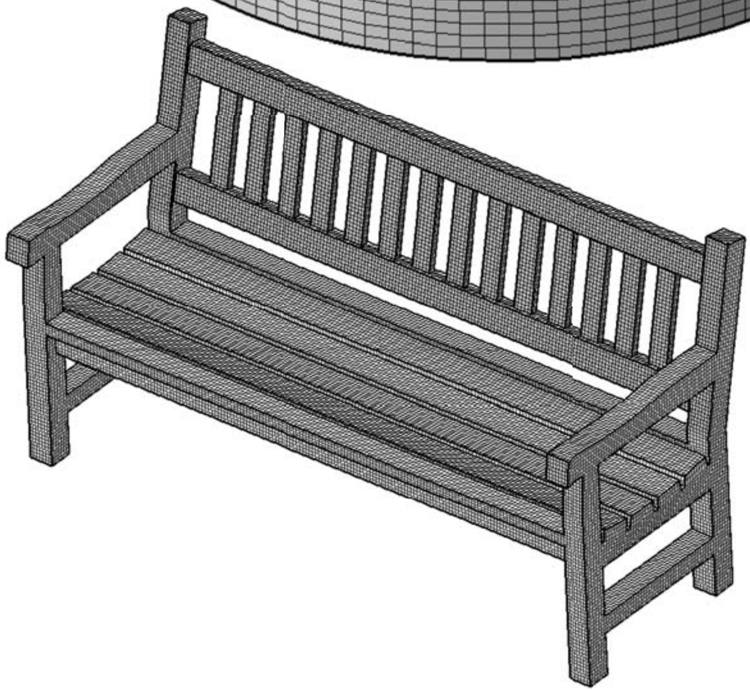


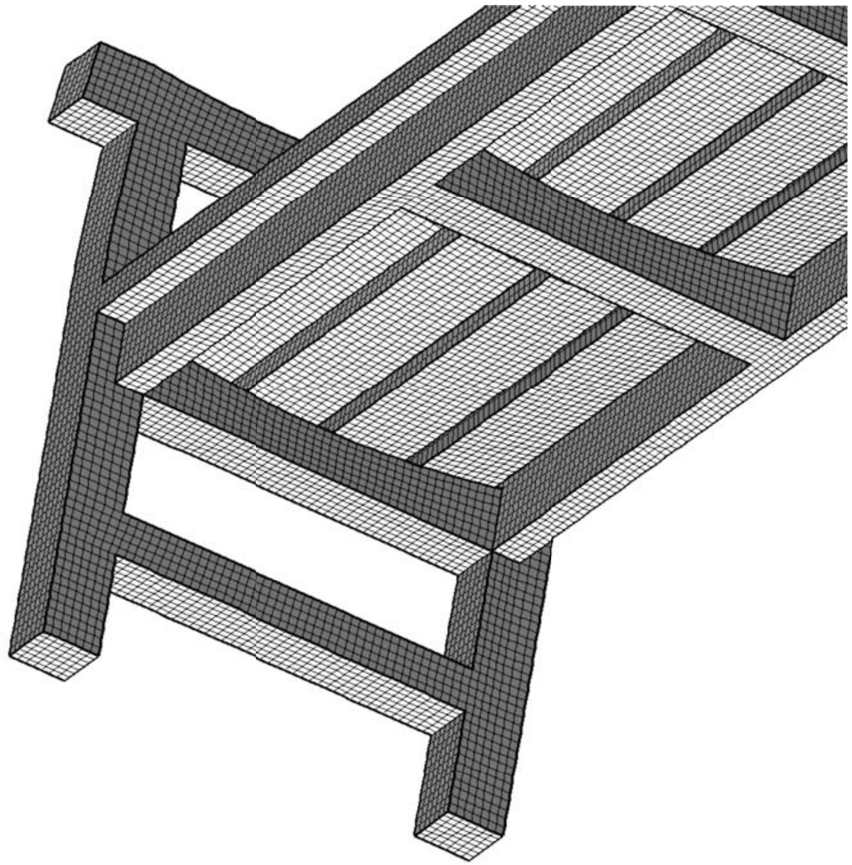
Is it a correct classification?

- **Some examples**



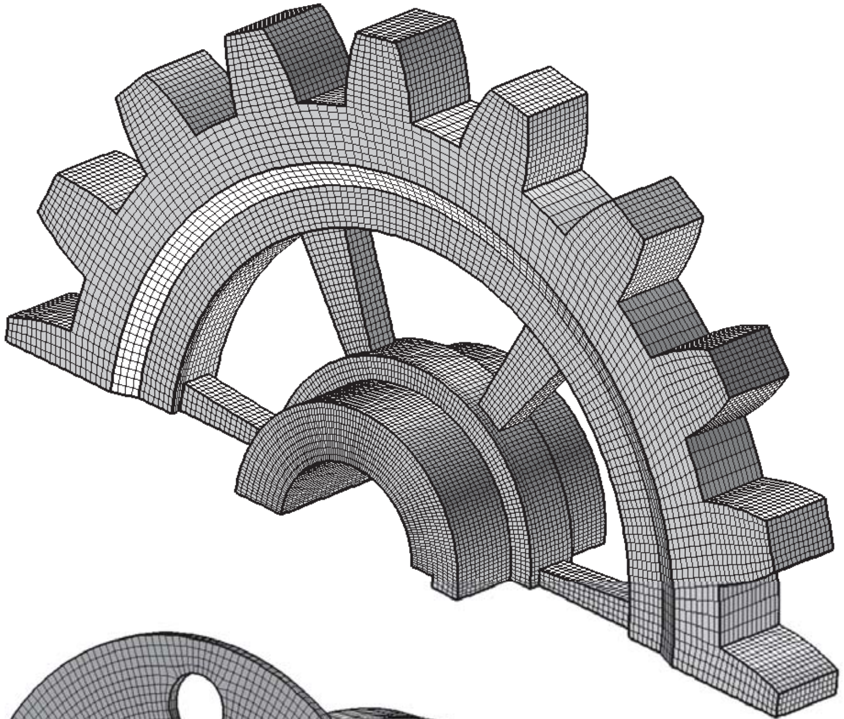


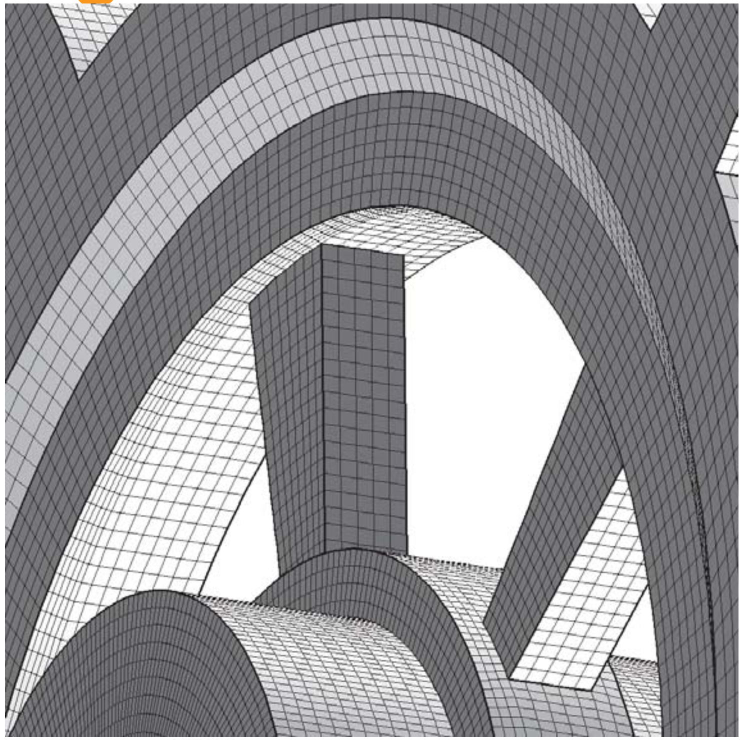


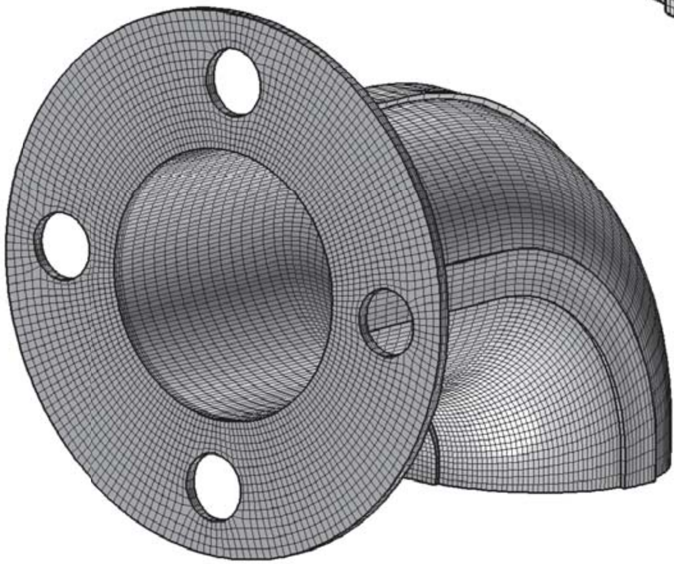


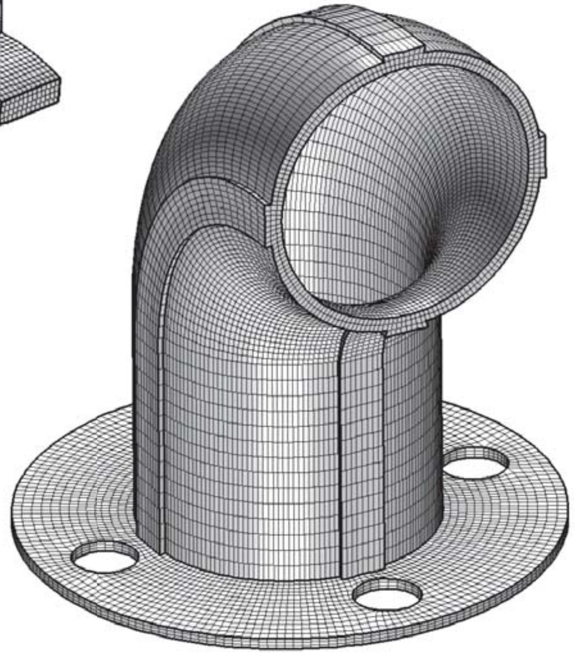
4. Structured mesh generation methods

- **Some examples**





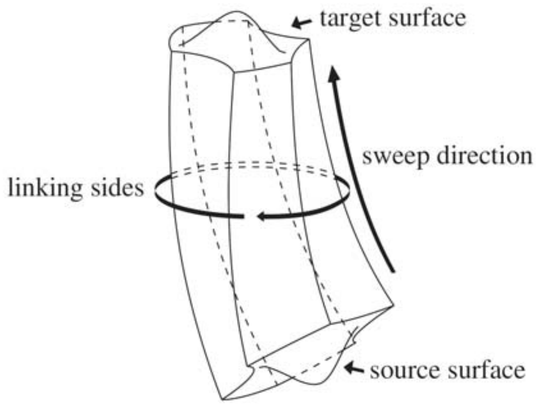




4. Structured mesh generation methods

▪ Sweeping

A method to decompose and mesh extrusion domains



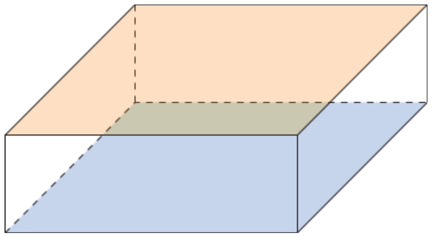
Extrusion Volume: a surface is swept along a path.
This volume is delimited by:

- Source surface
- Target surface (#logical faces$_S$ = #logical faces$_T$)

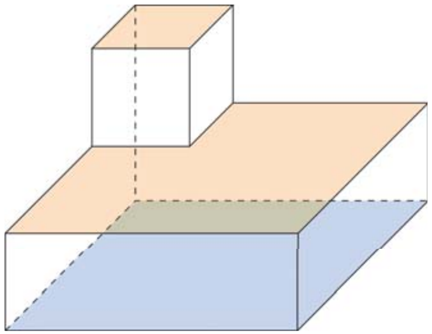
  Linking sides (4 logical faces, #lateral faces = #logical faces$_S$)

Classification

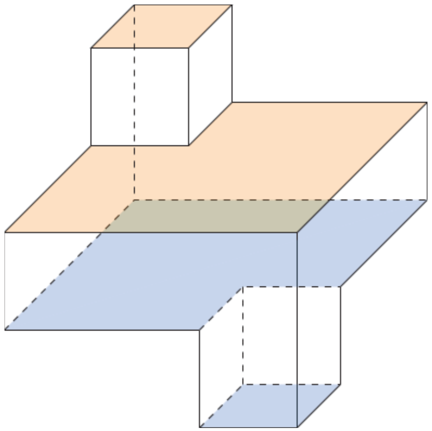
One-to-one sweeping



Many-to-one sweeping



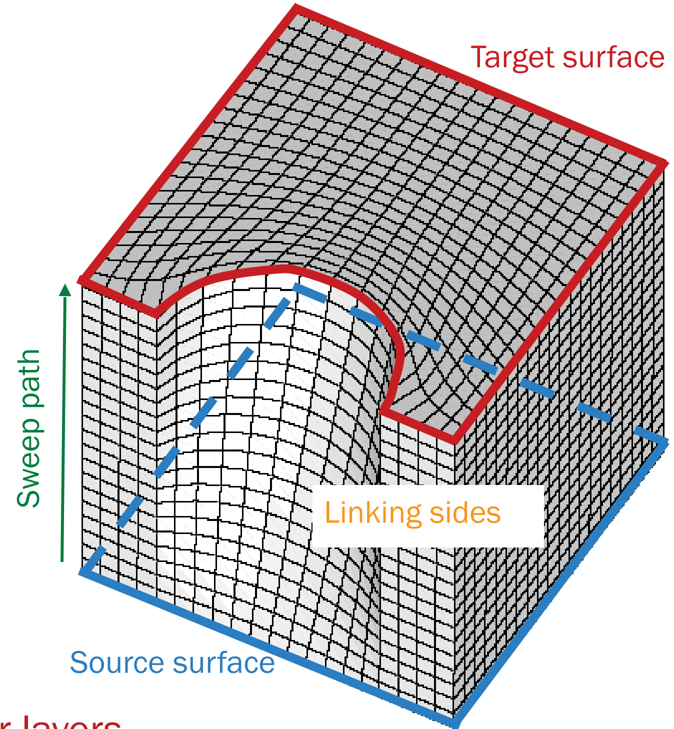
Many-to-many sweeping



Properties:

- Full automatic decomposition
- The decomposition leads to a compatible meshing of blocks
- Fast

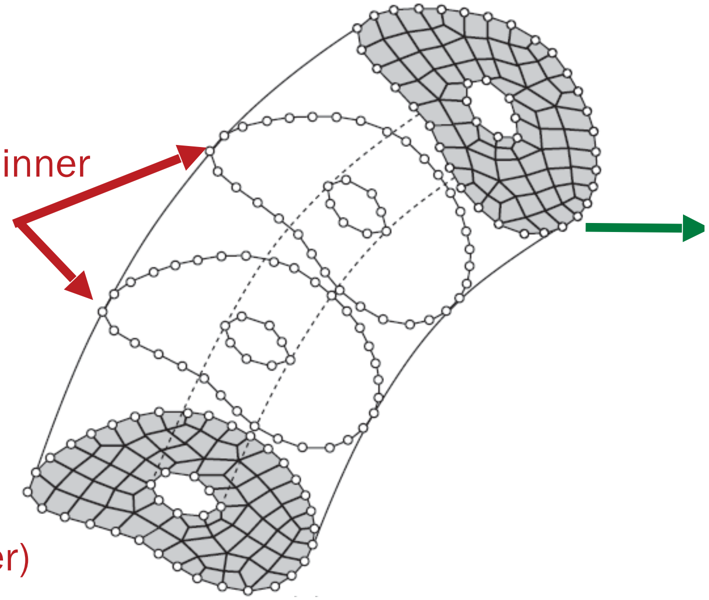
Basic tasks of a one-to-one sweeping



1. To mesh the source surface (structured or unstructured)
2. To project the source surface mesh onto the target surface
3. To create structured meshes over the linking sides (TFI, ...)

   Important: The structured meshes of the linking sides define the boundary of the inner layers
4. To project the cap surface meshes along the sweep path (creating the inner nodes)
5. Create 3D elements

Inner layers
There is not a underlying surface defining the inner layers
The inner layers are defined by
1. one outer loop
2. as many inner loops as inner holes
The inner layers are created using a weighted interpolation
1- From the cap meshes
2. Layer by layer (in an advancing front manner)

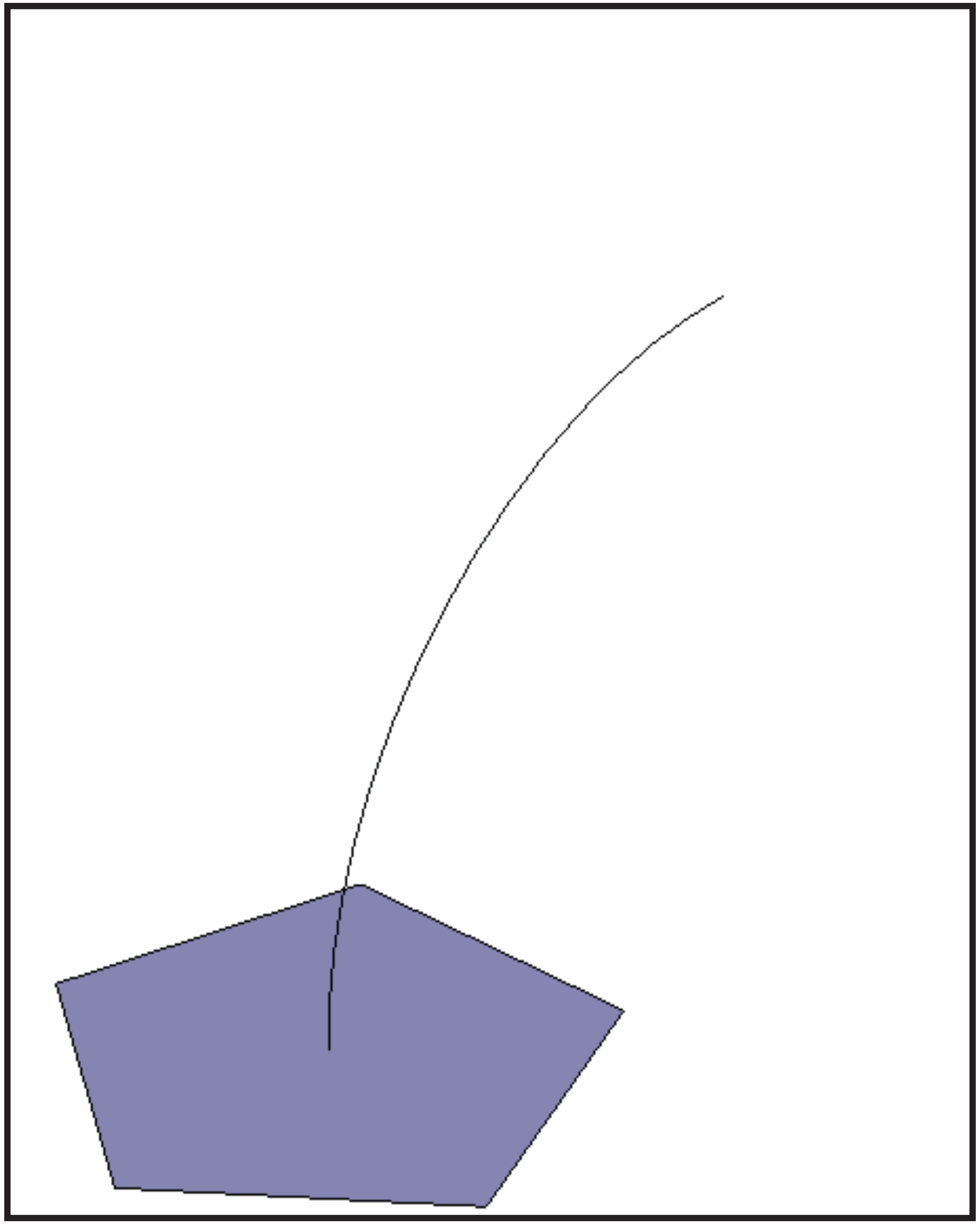


Target surface mesh
Mapped from the source surface mesh
There is an underlying geometry describing the surface (usually from the CAD model)

4. Structured mesh generation methods

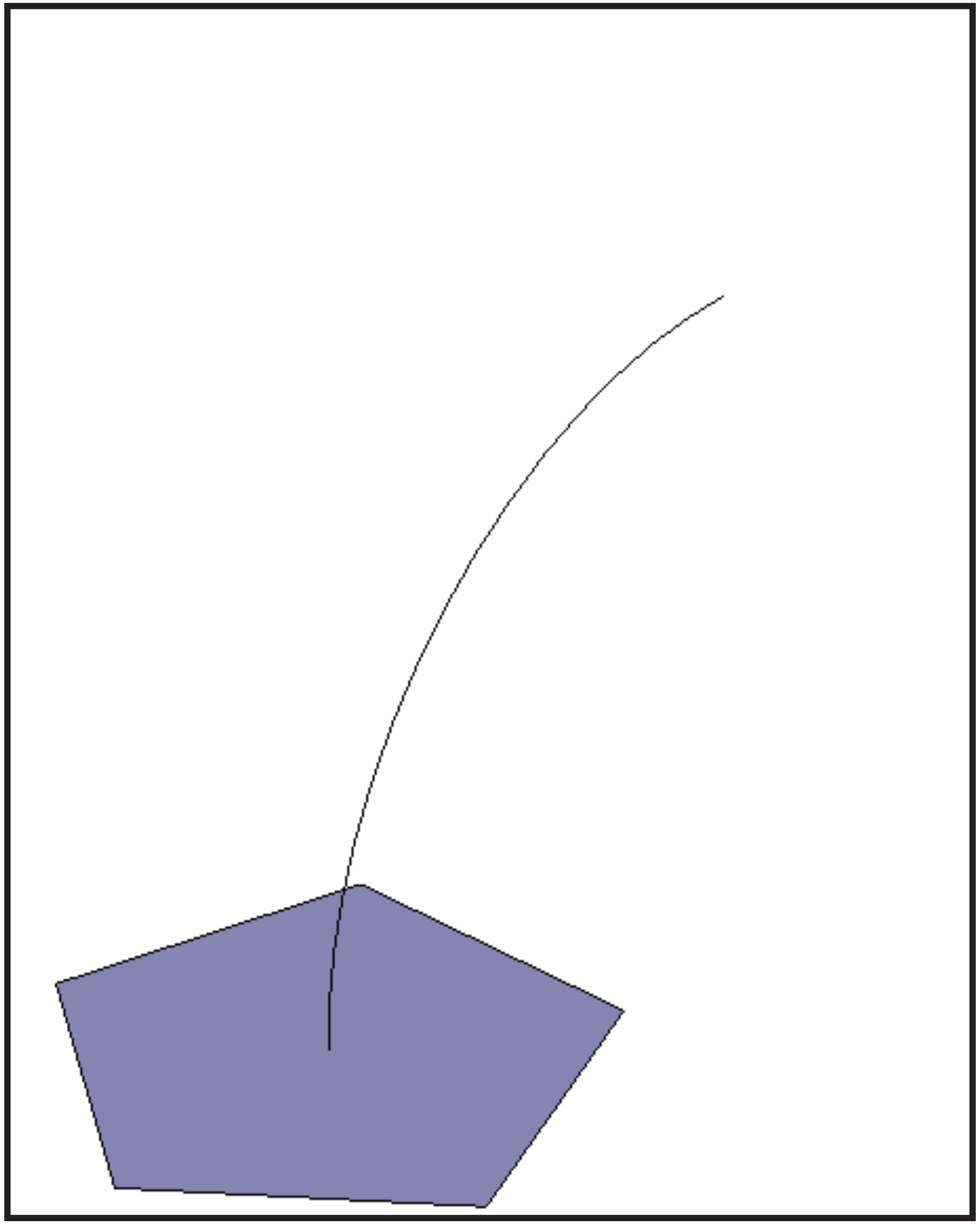
4. Structured mesh generation methods

· Initial surface and sweep path



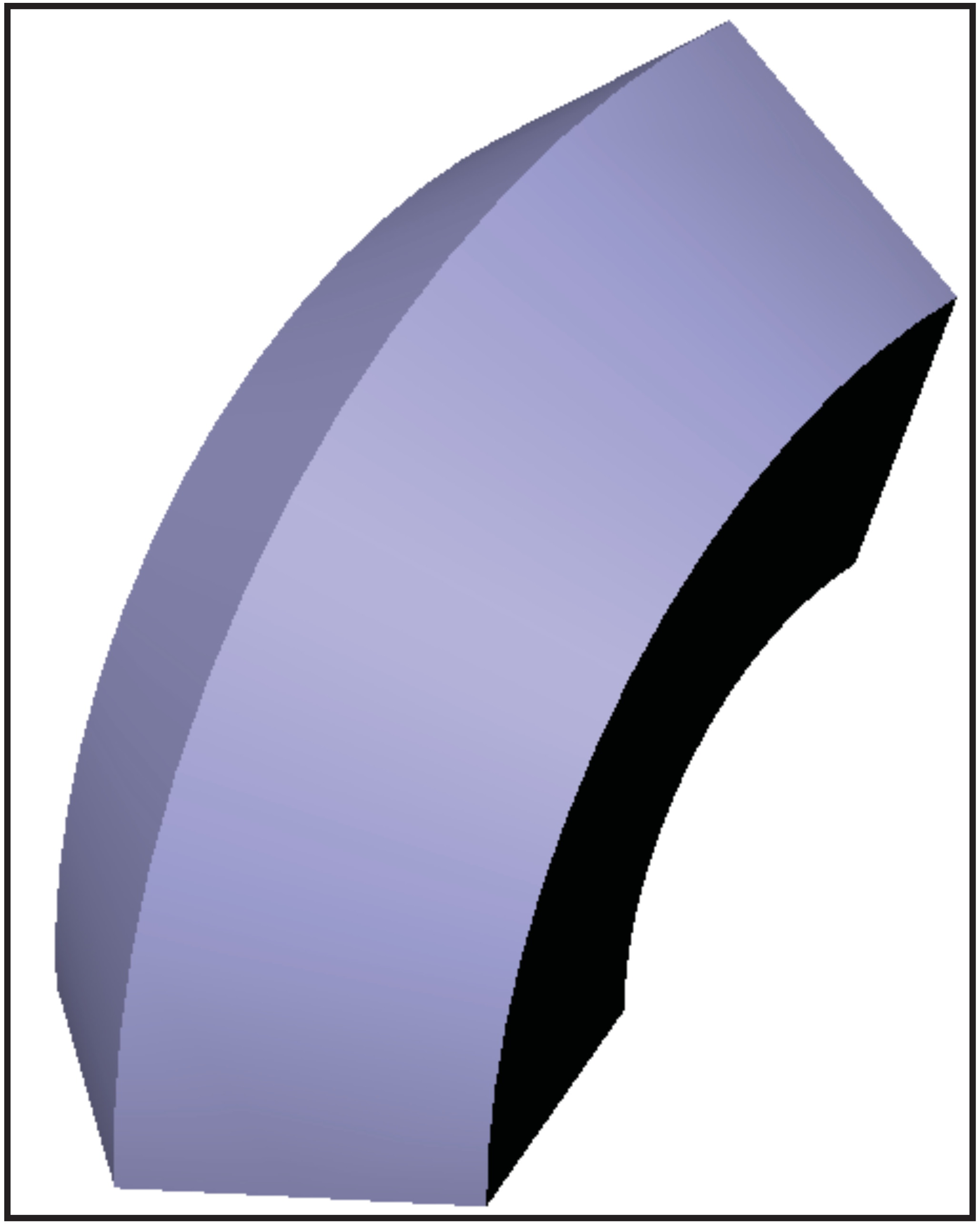
· **Initial surface and sweep path**

4. Structured mesh generation methods



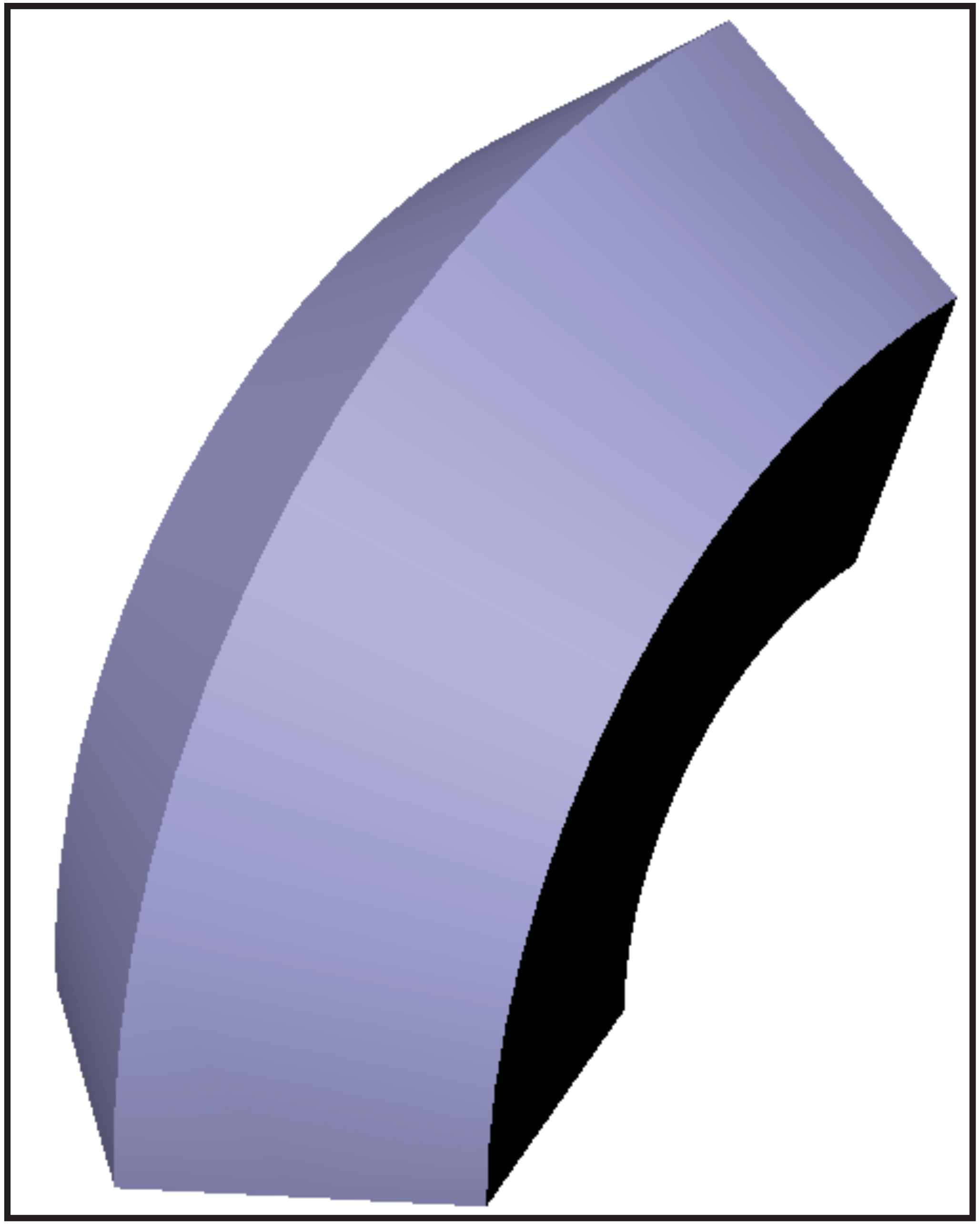
- Initial surface and sweep path
- Sweep volume

4. Structured mesh generation methods



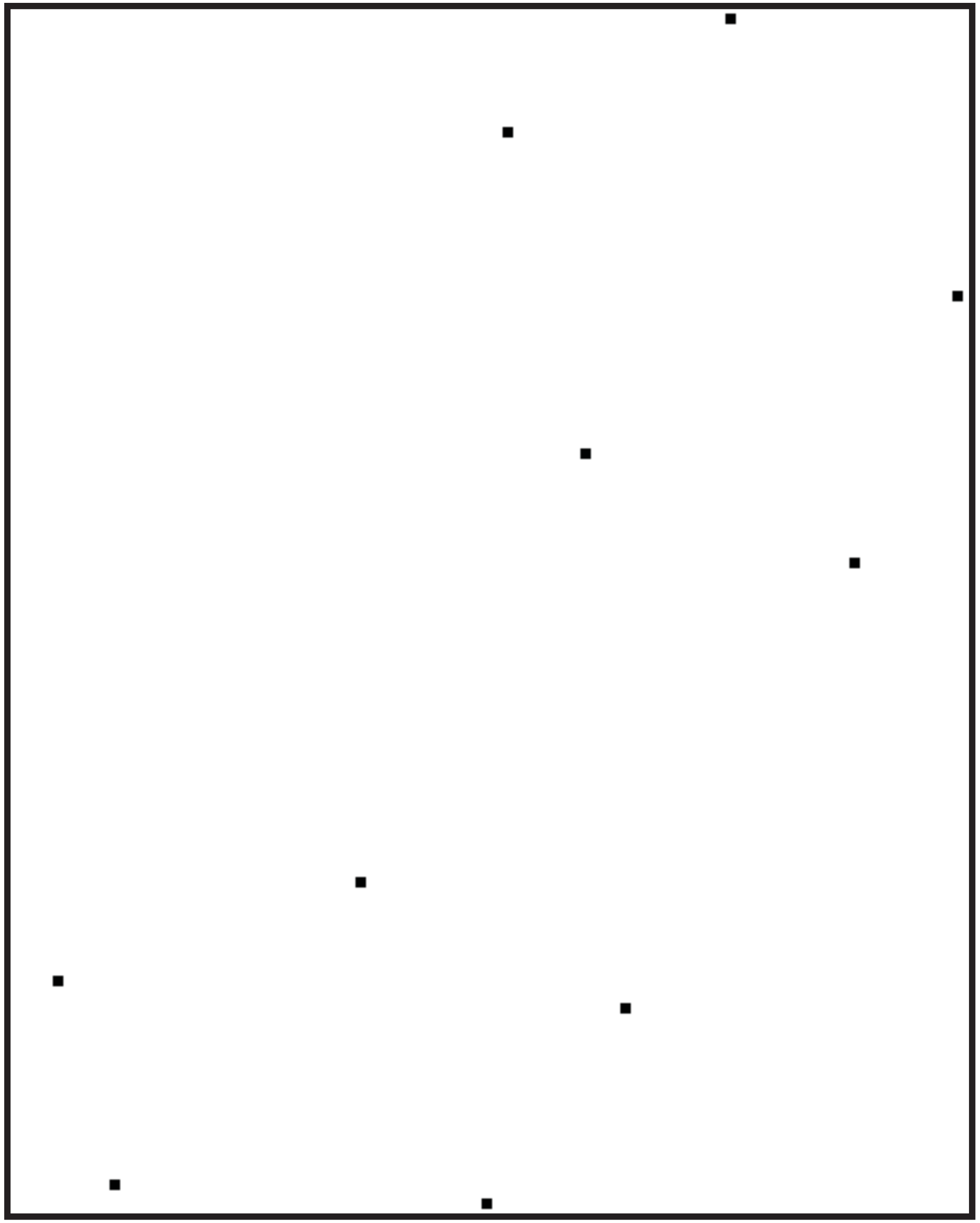
- Initial surface and sweep path
- Sweep volume

4. Structured mesh generation methods



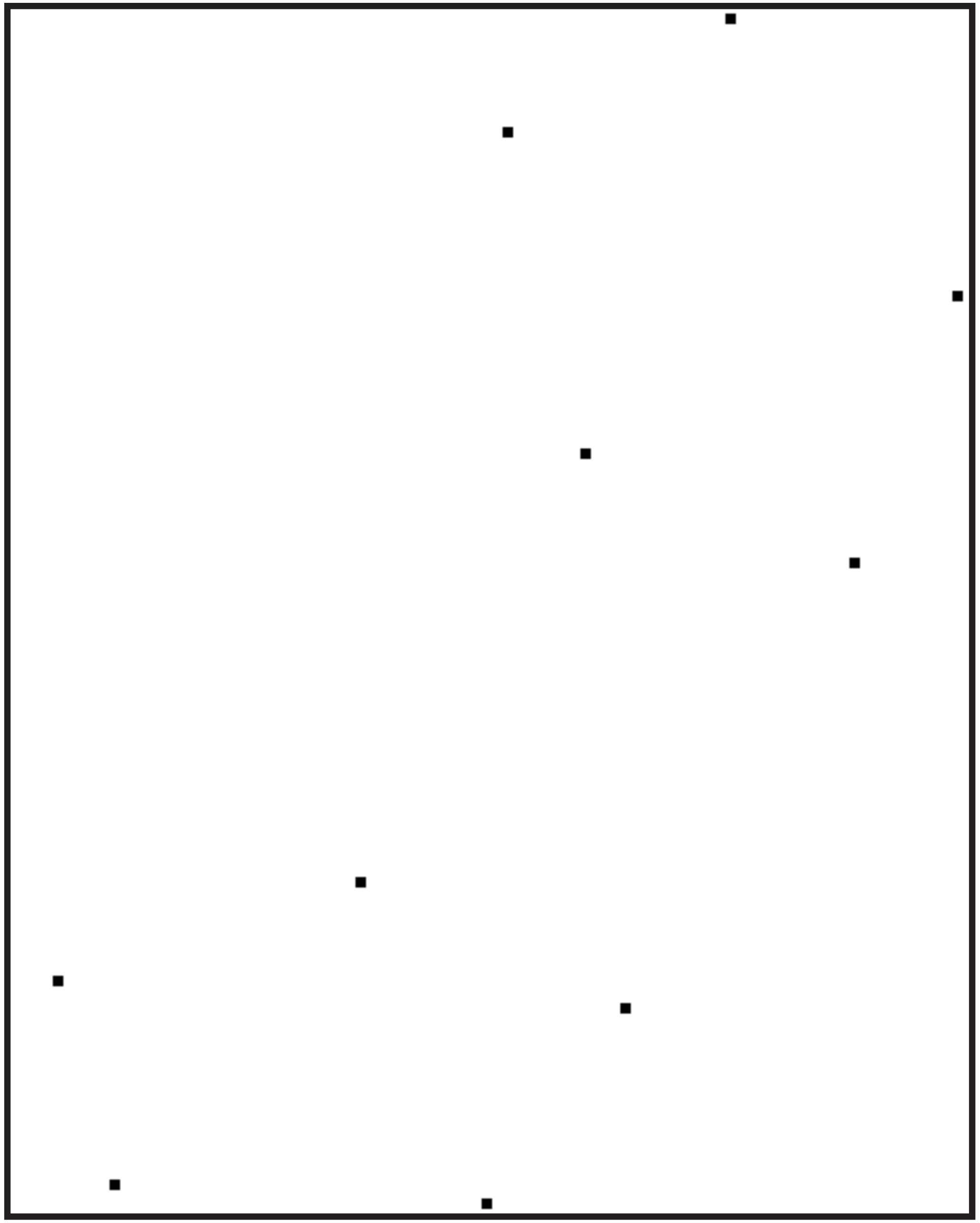
- Initial surface and sweep path
- Sweep volume
- 0D mesh

4. Structured mesh generation methods



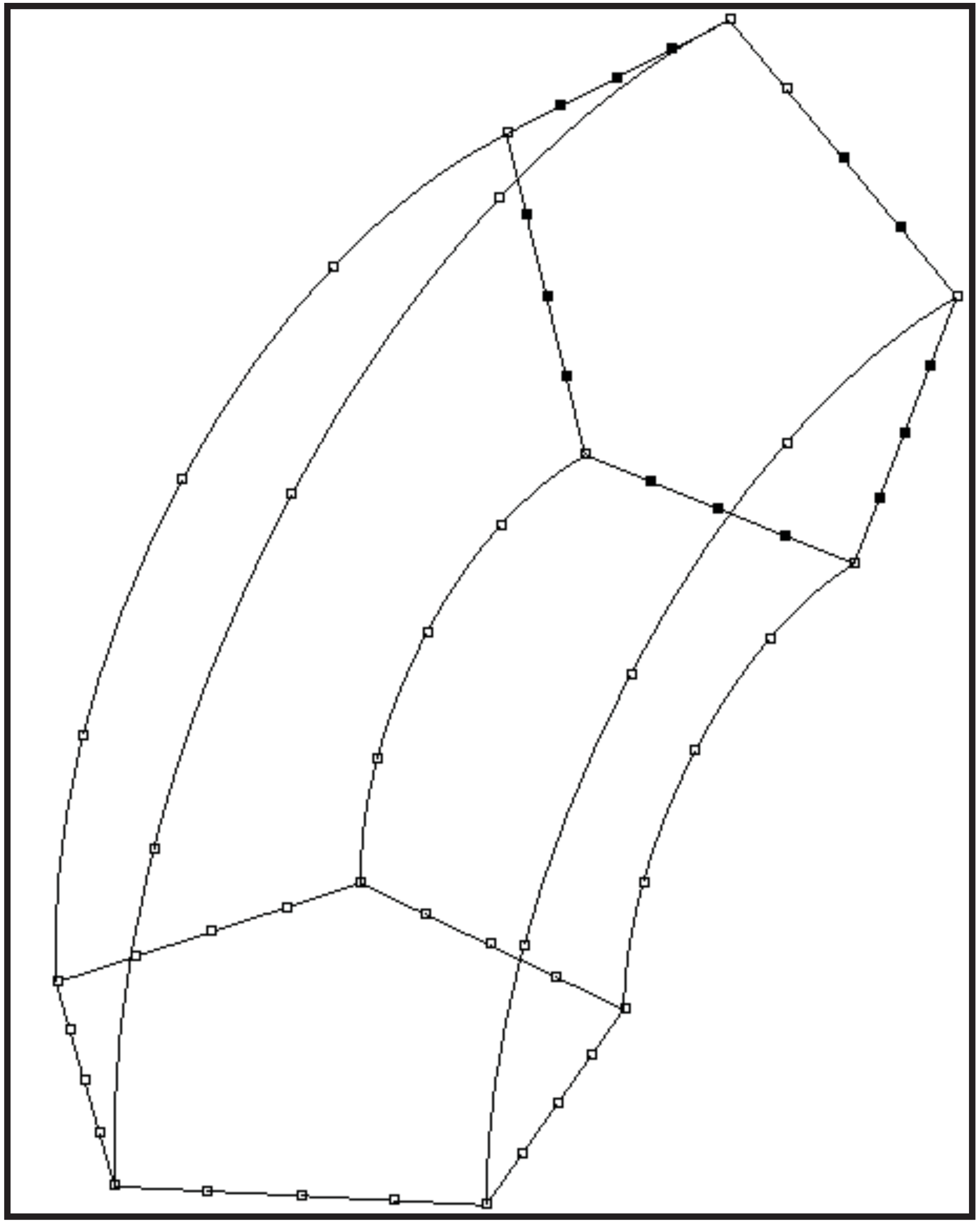
- Initial surface and sweep path
- Sweep volume
- 0D mesh

4. Structured mesh generation methods



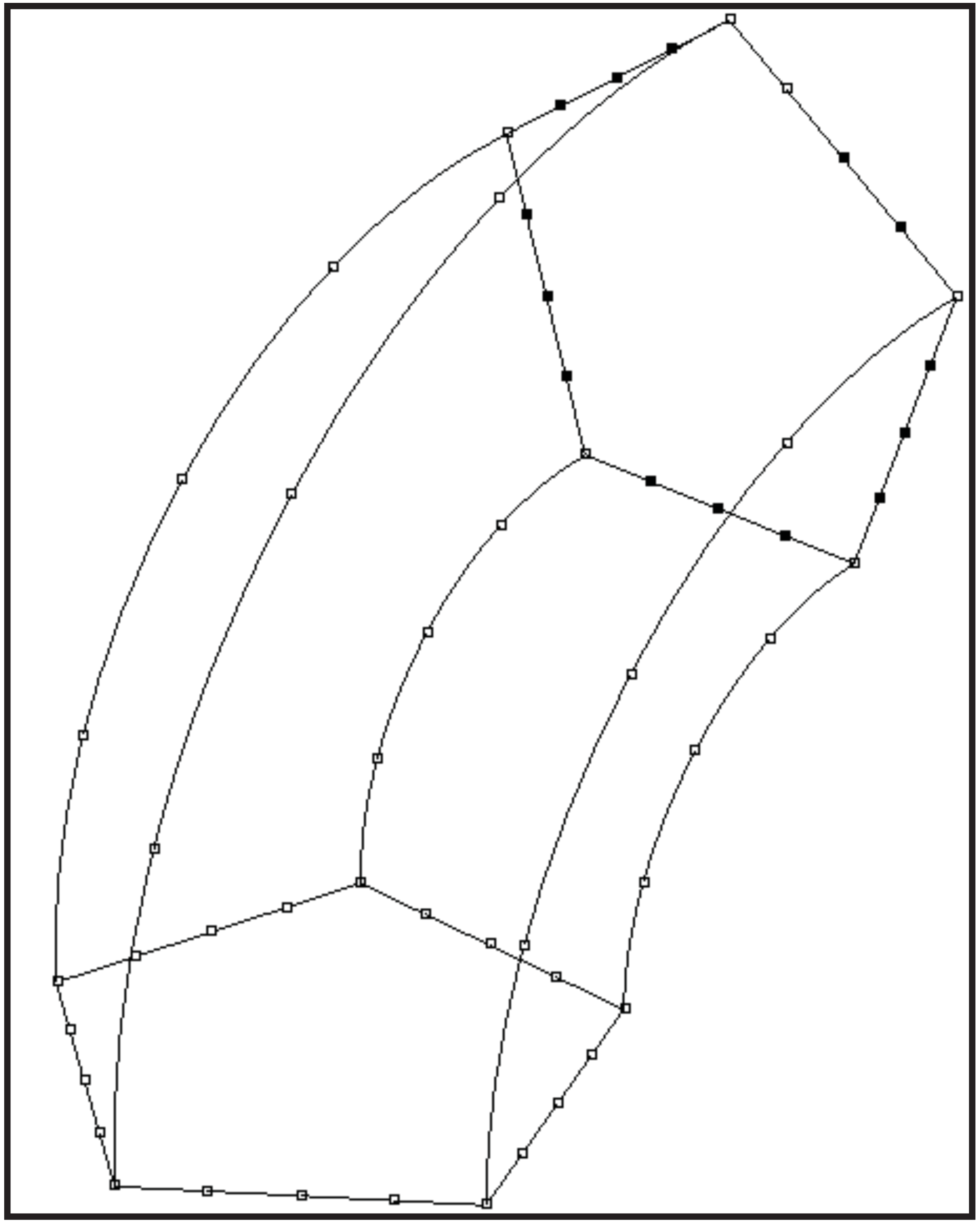
- Initial surface and sweep path
- Sweep volume
- 0D mesh
- 1D mesh

4. Structured mesh generation methods



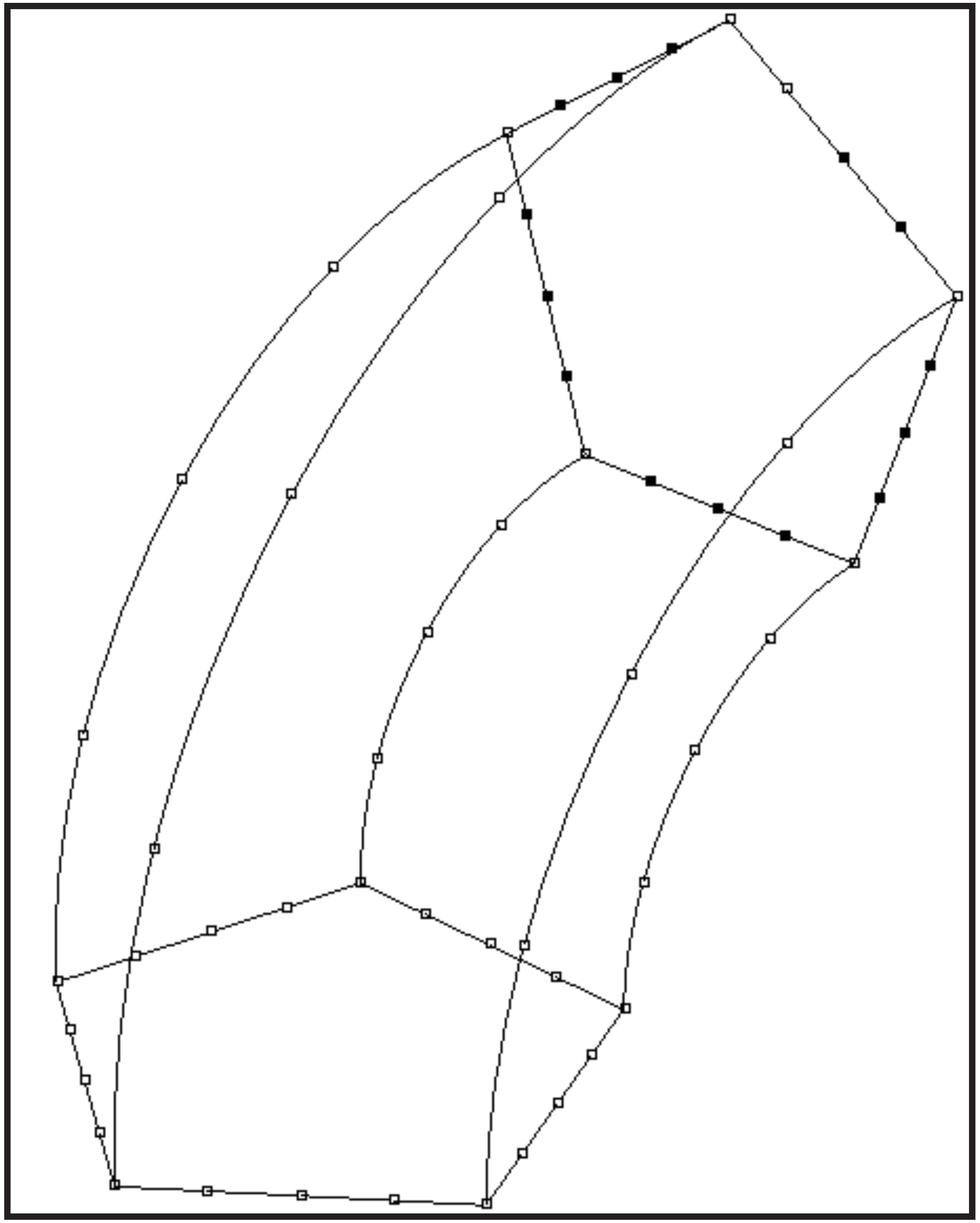
- Initial surface and sweep path
- Sweep volume
- 0D mesh
- 1D mesh

4. Structured mesh generation methods



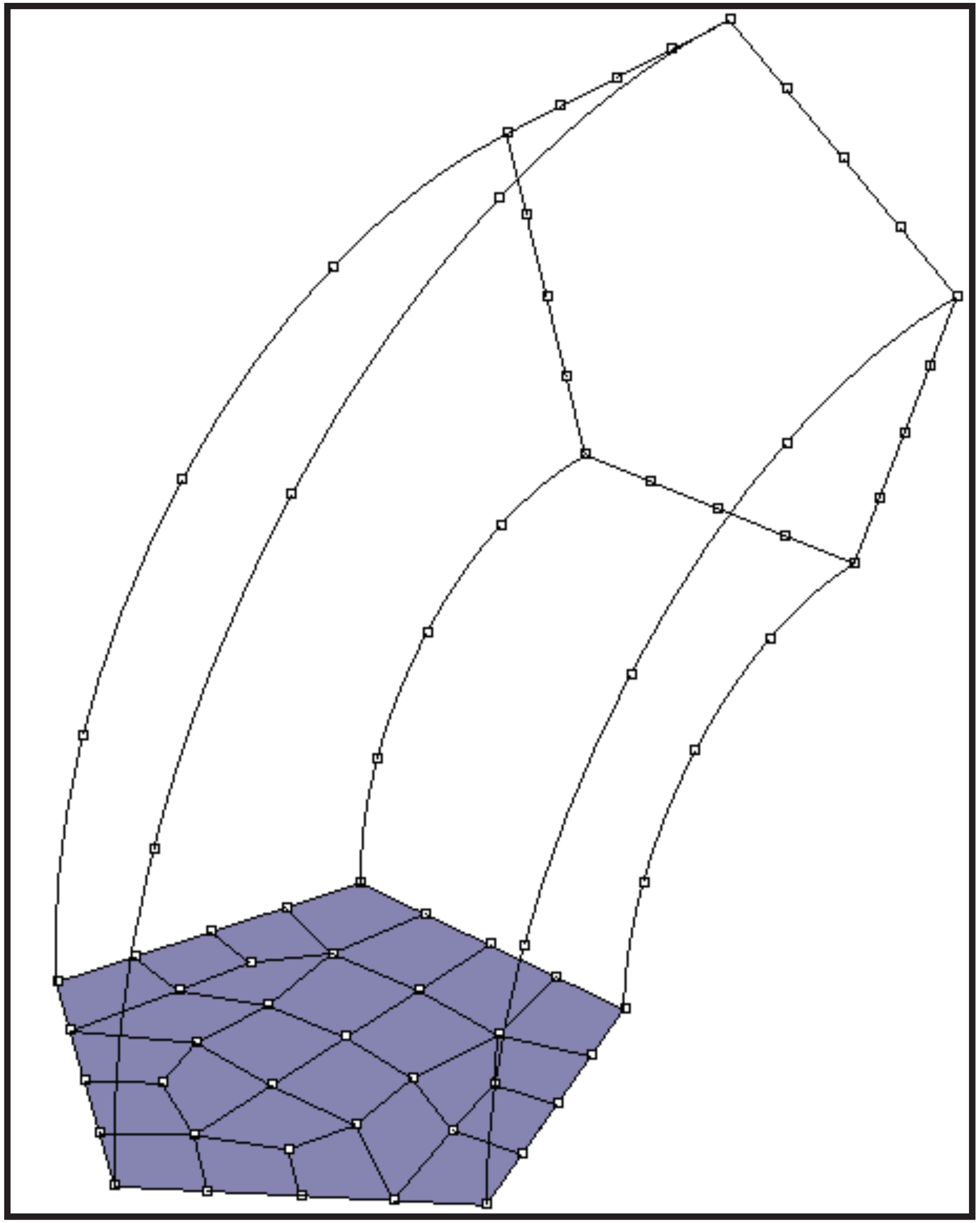
- Initial surface and sweep path
- Sweep volume
- 0D mesh
- 1D mesh
- 2D mesh

4. Structured mesh generation methods



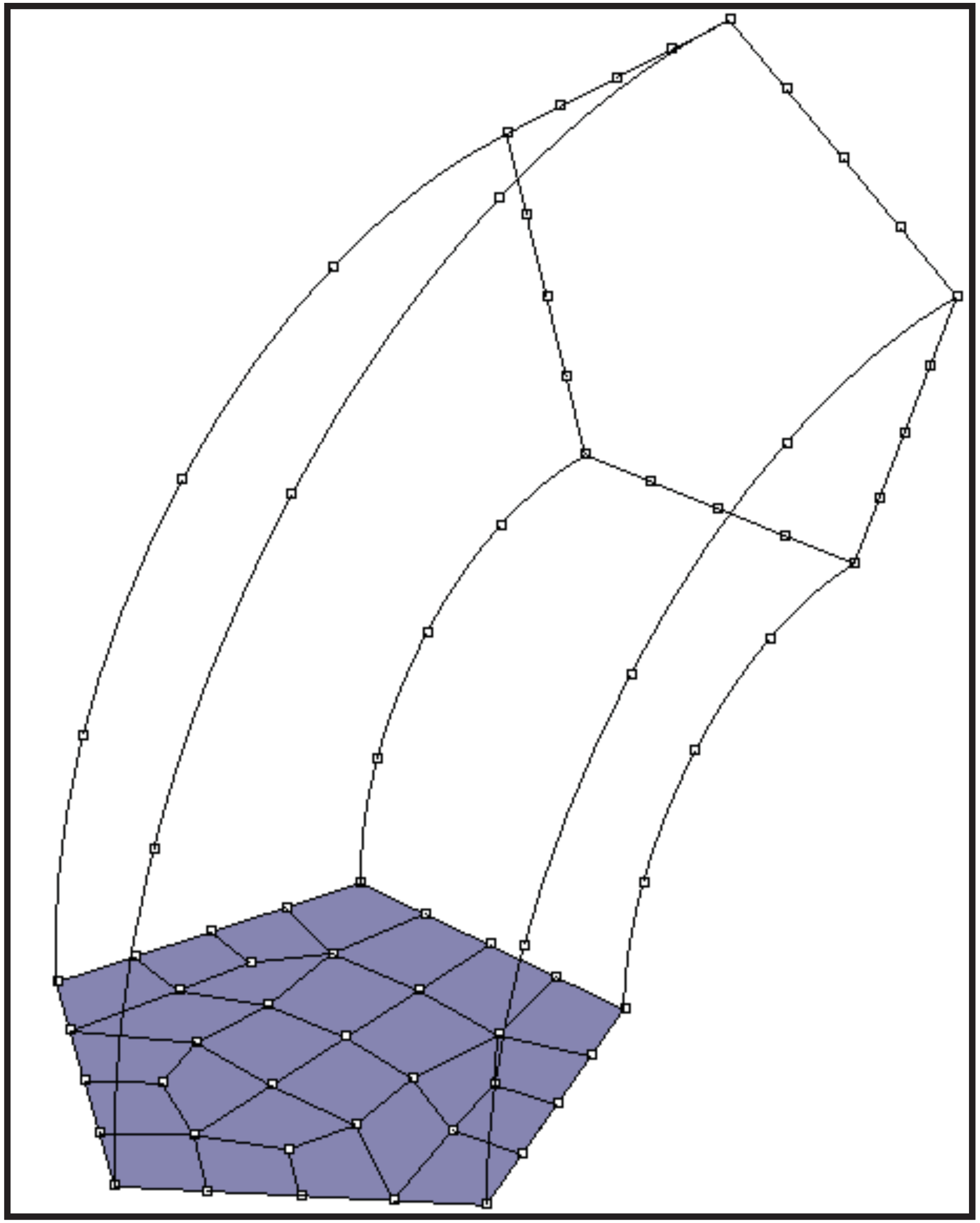
- Initial surface and sweep path
- Sweep volume
- 0D mesh
- 1D mesh
- 2D mesh
  - Source surface mesh (unstructured)

4. Structured mesh generation methods



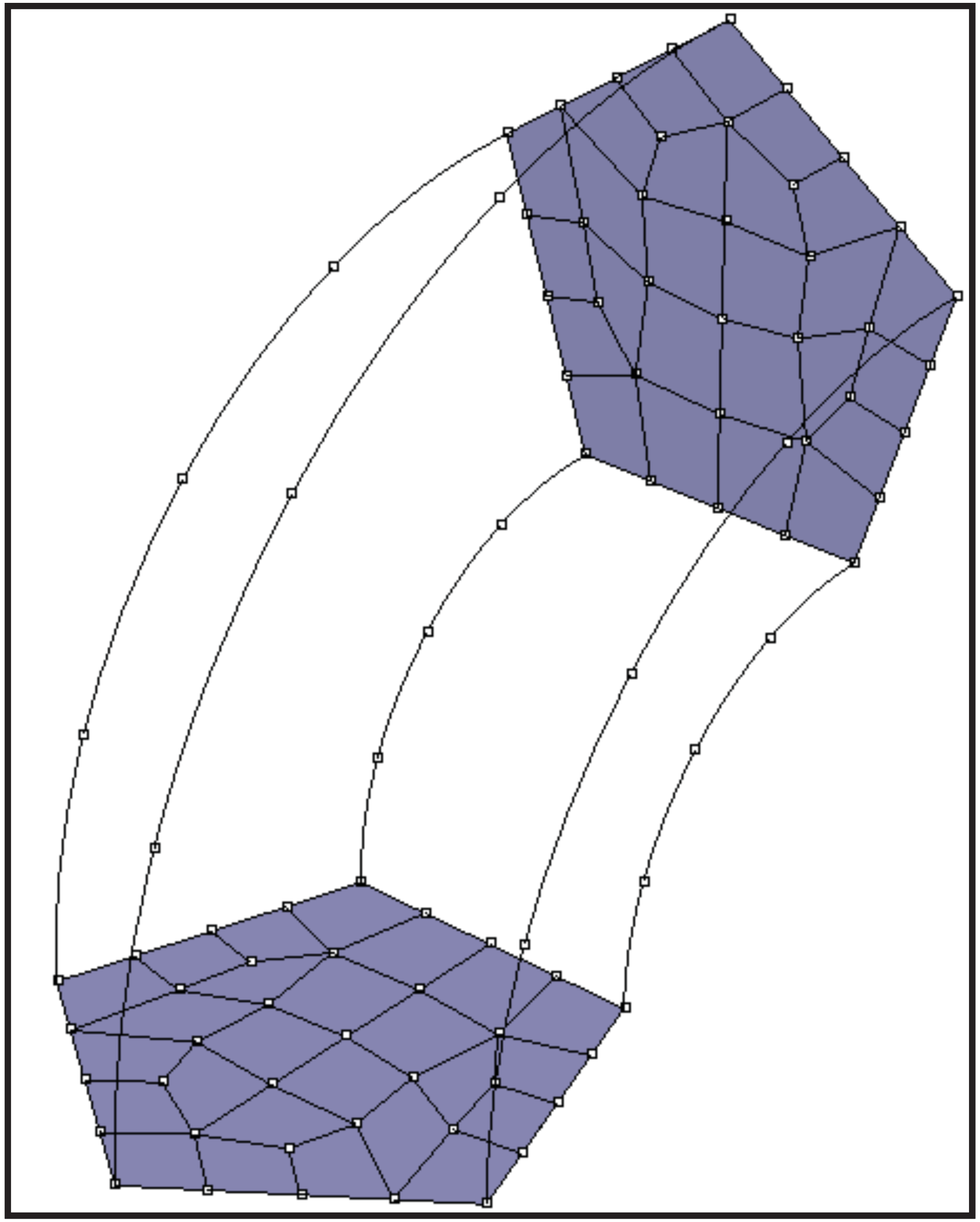
- **Initial surface and sweep path**
- **Sweep volume**
- **0D mesh**
- **1D mesh**
- **2D mesh**
  - **Source surface mesh** (unstructured)

4. Structured mesh generation methods



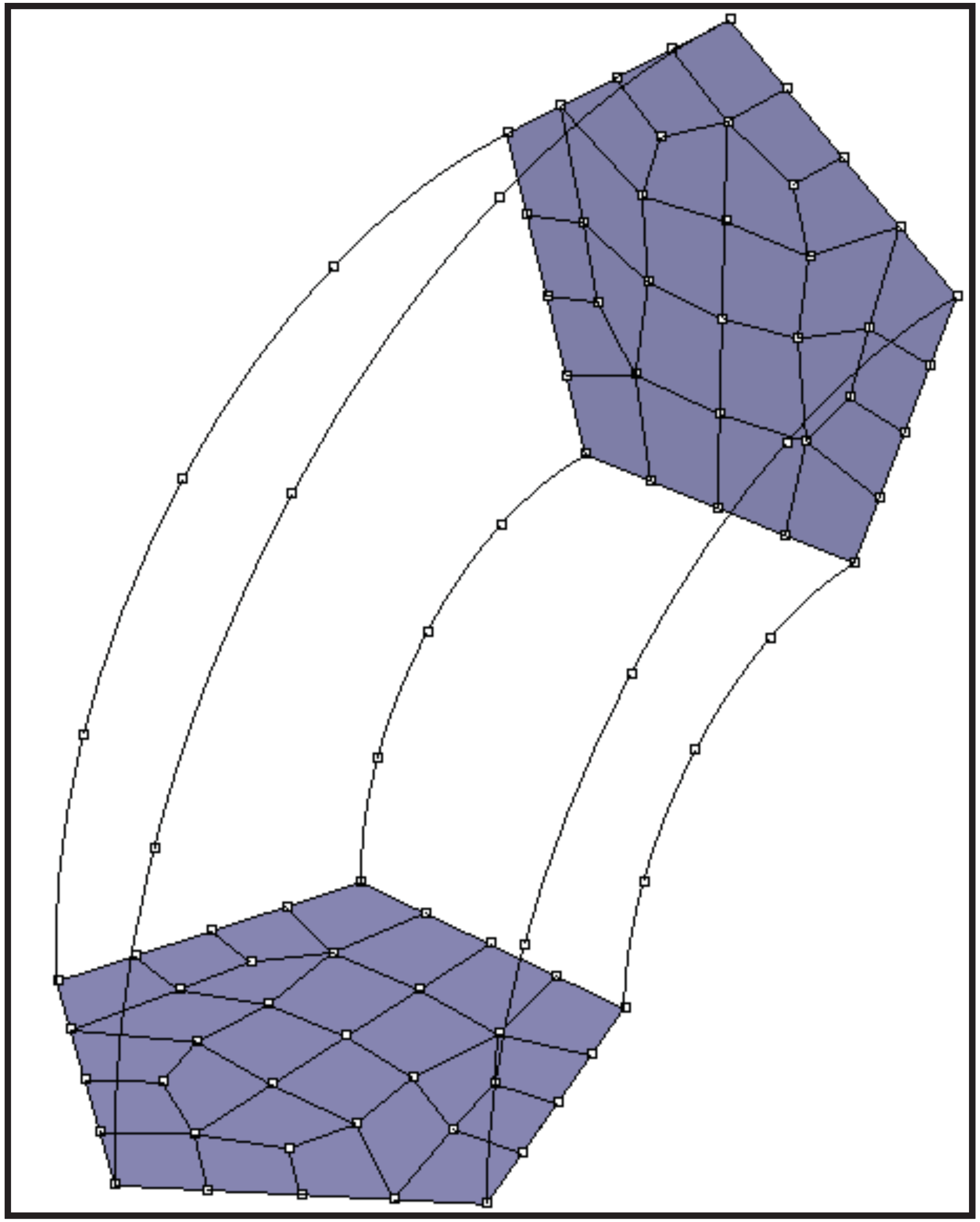
- Initial surface and sweep path
- Sweep volume
- 0D mesh
- 1D mesh
- 2D mesh
    - **Source surface mesh** (unstructured)
    - **Target surface mesh** (projected using least-squares or other methods)

4. Structured mesh generation methods



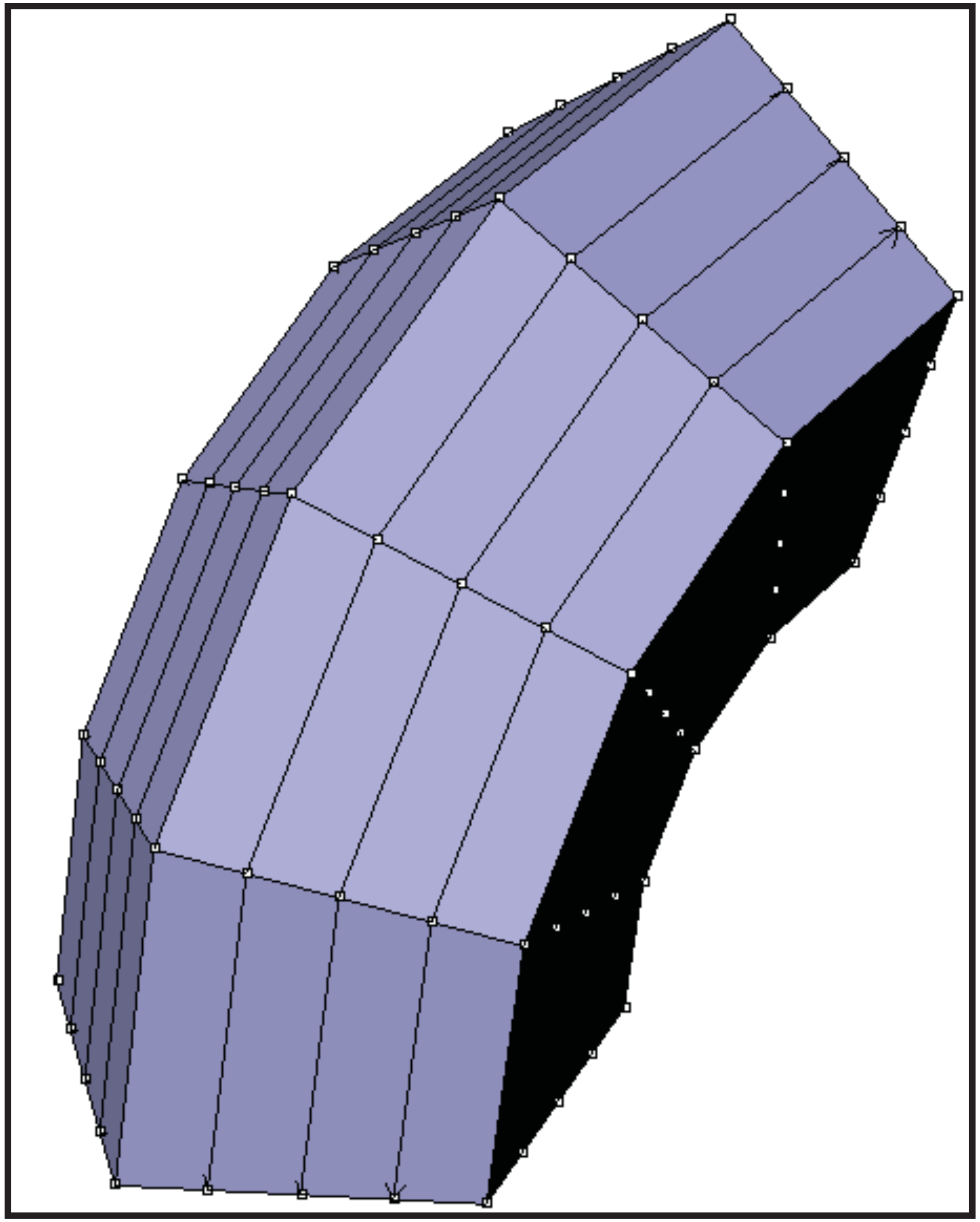
- **Initial surface and sweep path**
- **Sweep volume**
- **0D mesh**
- **1D mesh**
- **2D mesh**
  - **Source surface mesh** (unstructured)
  - **Target surface mesh** (projected using least-squares or other methods)

4. Structured mesh generation methods



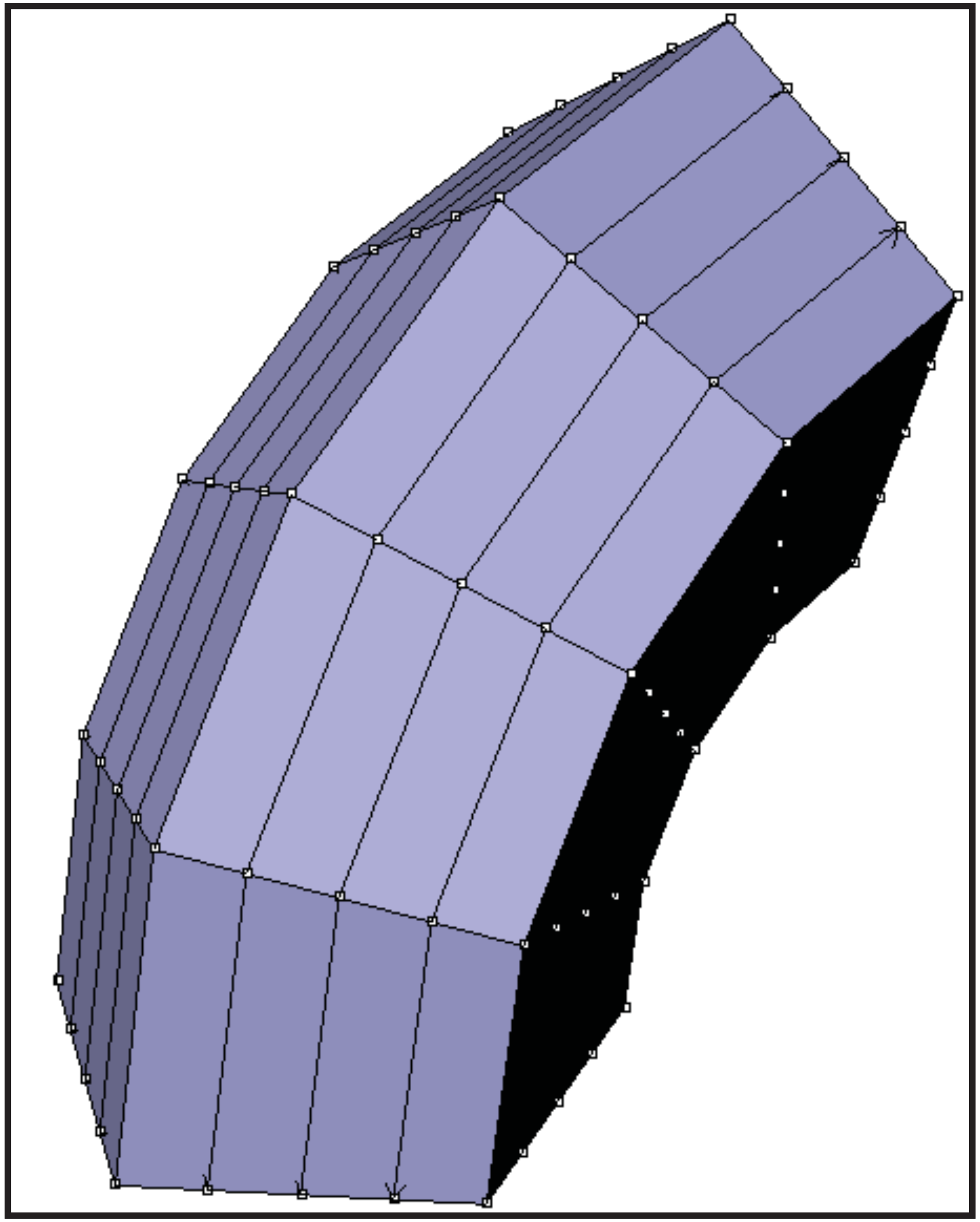
- **Initial surface and sweep path**
- **Sweep volume**
- **0D mesh**
- **1D mesh**
- **2D mesh**
  - **Source surface mesh** (unstructured)
  - **Target surface mesh** (projected using least-squares or other methods)
  - **Linking sides meshes** (structured, TFI)

4. Structured mesh generation methods



- Initial surface and sweep path
- Sweep volume
- 0D mesh
- 1D mesh
- 2D mesh
  - **Source surface mesh** (unstructured)
  - **Target surface mesh** (projected using least-squares or other methods)
  - **Linking sides meshes** (structured, TFI)

4. Structured mesh generation methods



· **Initial surface and sweep path**

· **Sweep volume**

· **0D mesh**

· **1D mesh**

· **2D mesh**

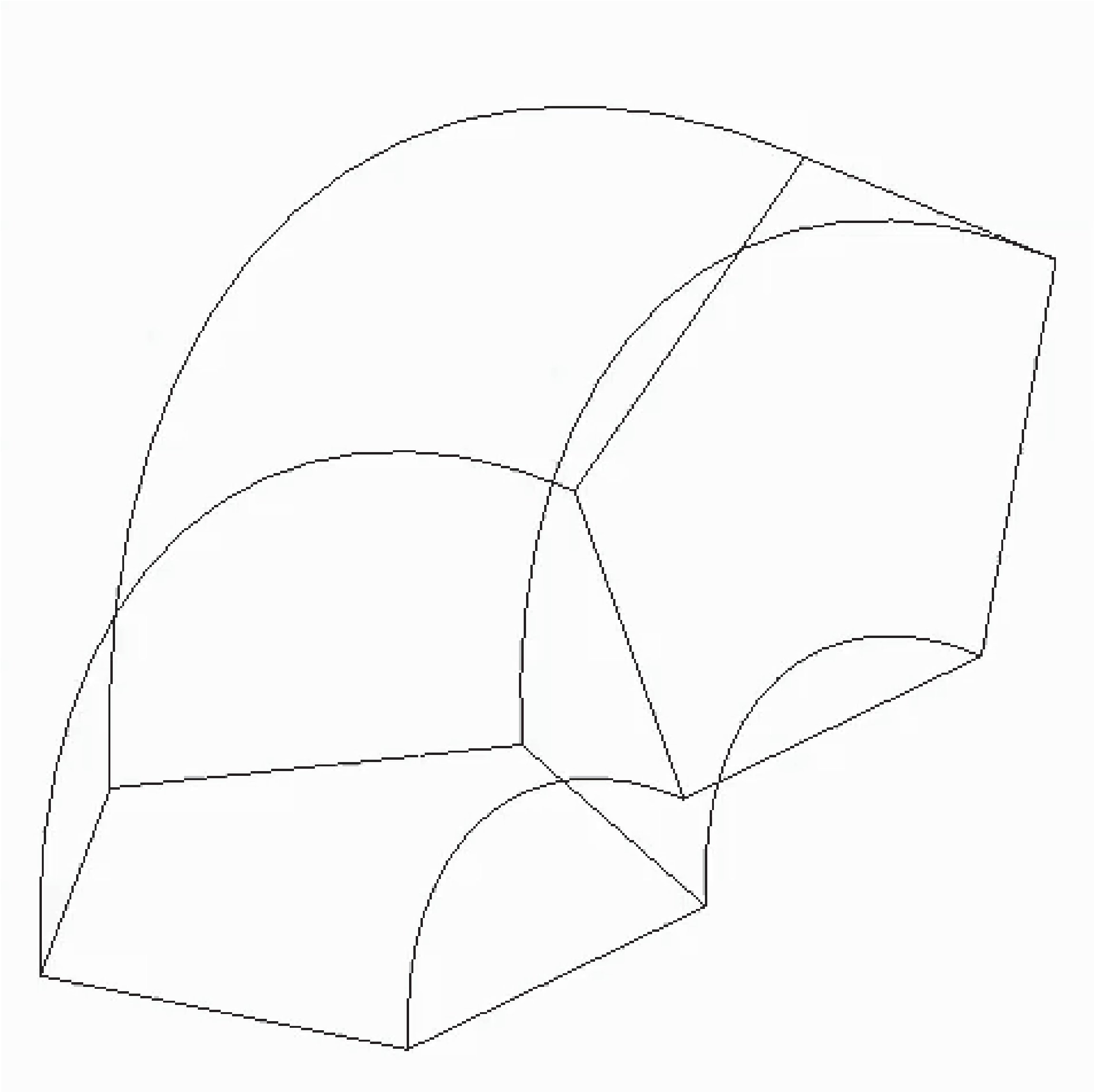
  · **Source surface mesh** (unstructured)

  · **Target surface mesh** (projected using least-squares or other methods)

  · **Linking sides meshes** (structured, TFI)

· **3D mesh**

4. Structured mesh generation methods

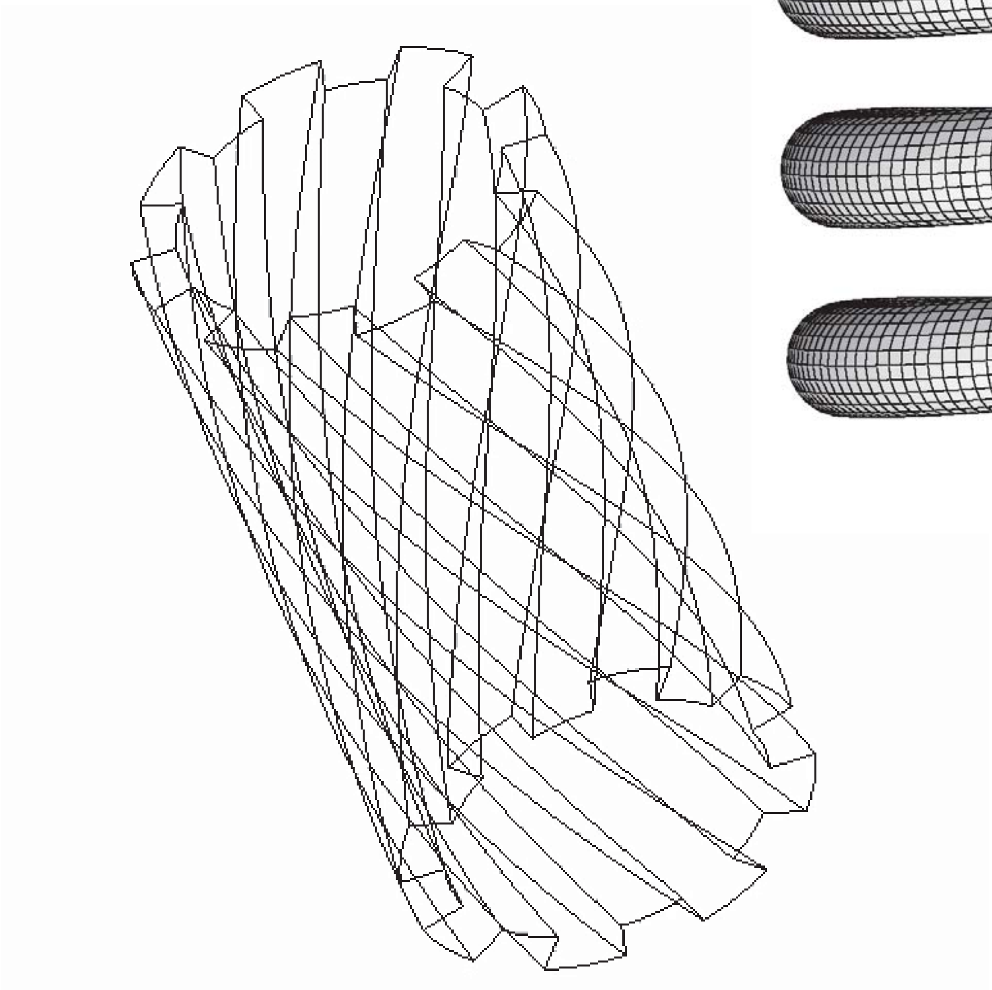


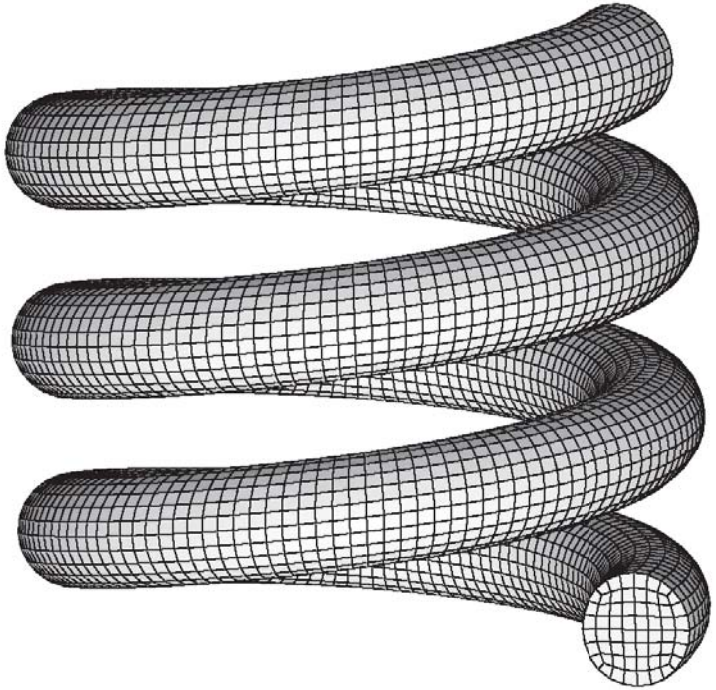
- Initial surface and sweep path
- Sweep volume
- 0D mesh
- 1D mesh
- 2D mesh
  - **Source surface mesh** (unstructured)
  - **Target surface mesh** (projected using least-squares or other methods)
  - **Linking sides meshes** (structured, TFI)
- 3D mesh
  - **inner nodes** (projected using least-squares or other methods)

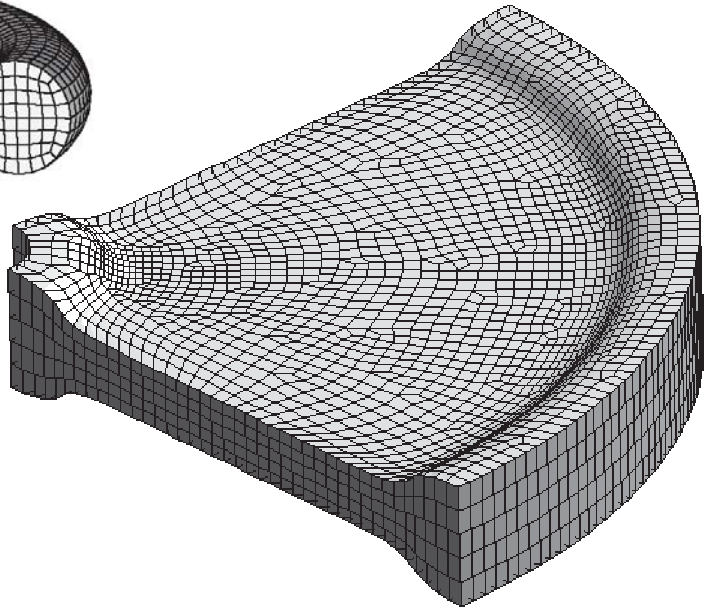
  and 3D elements

4. Structured mesh generation methods

- **Some examples**: one-to-one sweep meshes

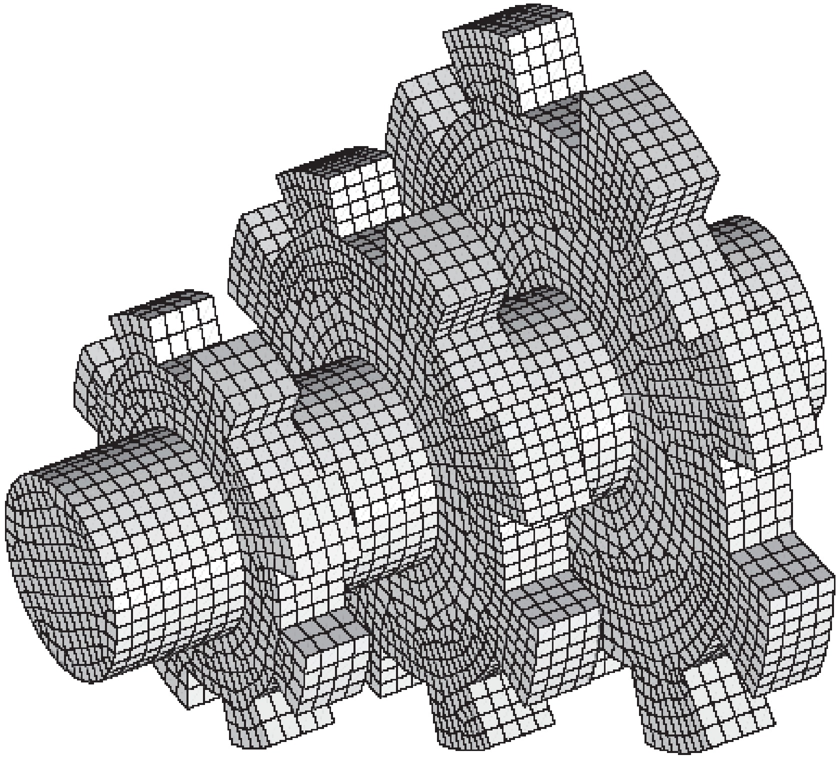
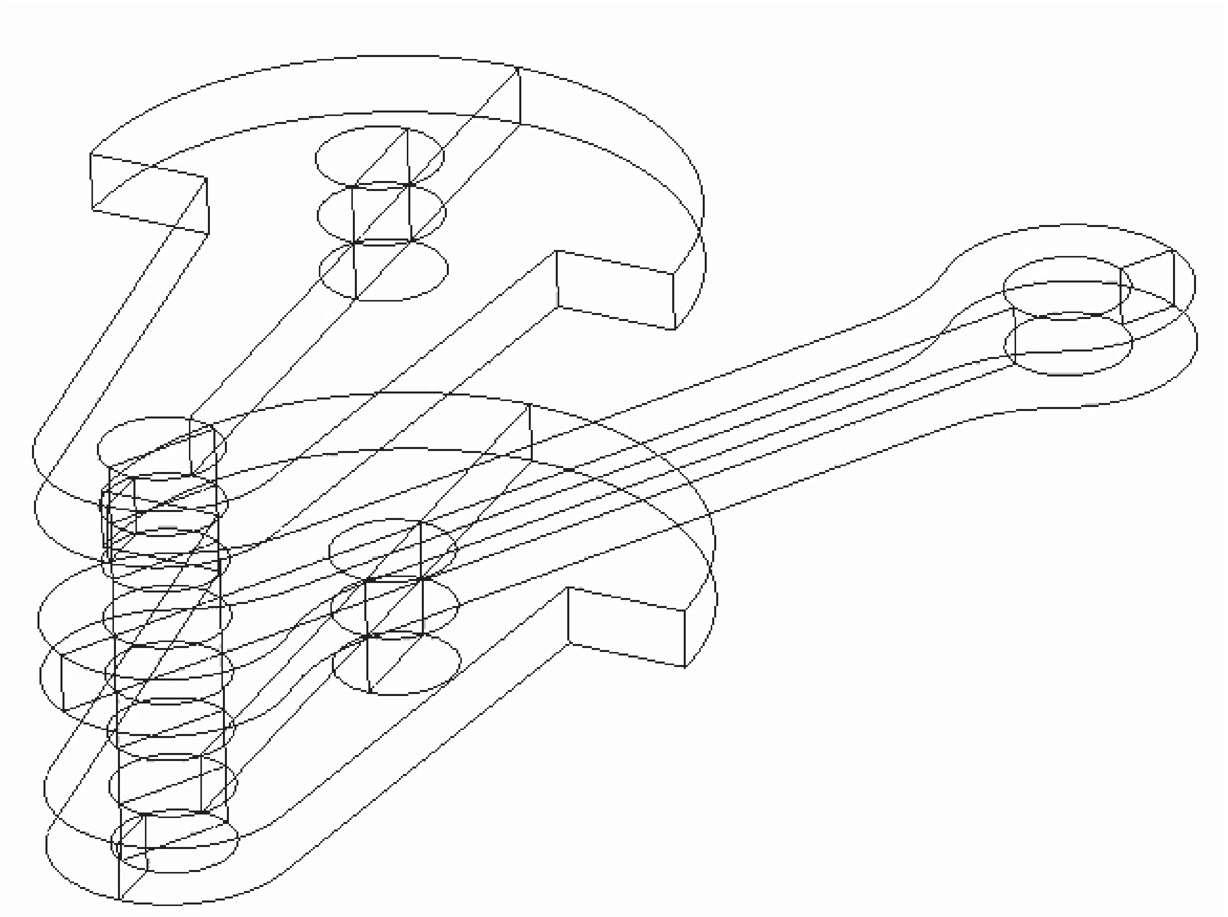






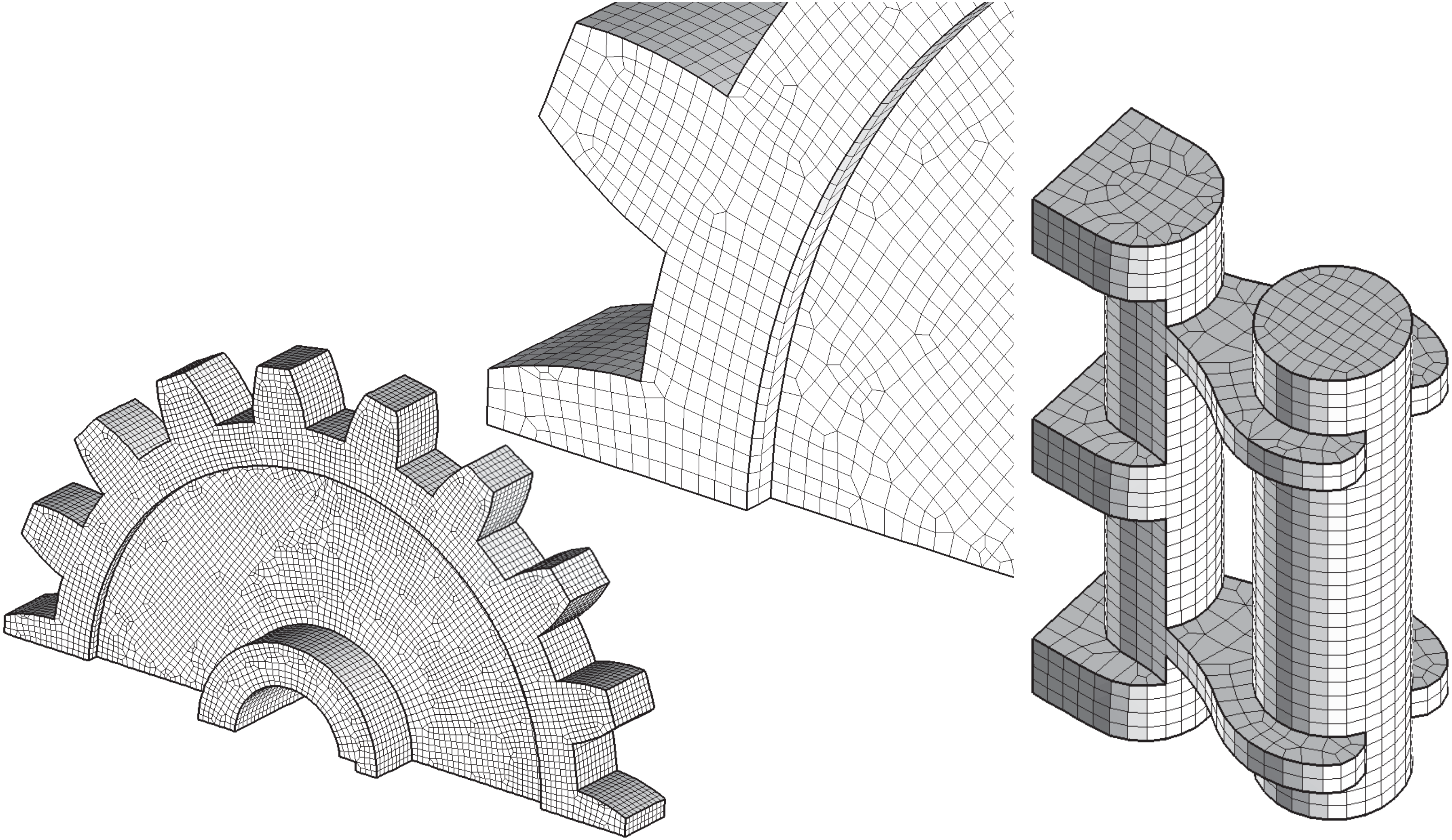
4. Structured mesh generation methods

- **Some examples**: many-to-many meshes

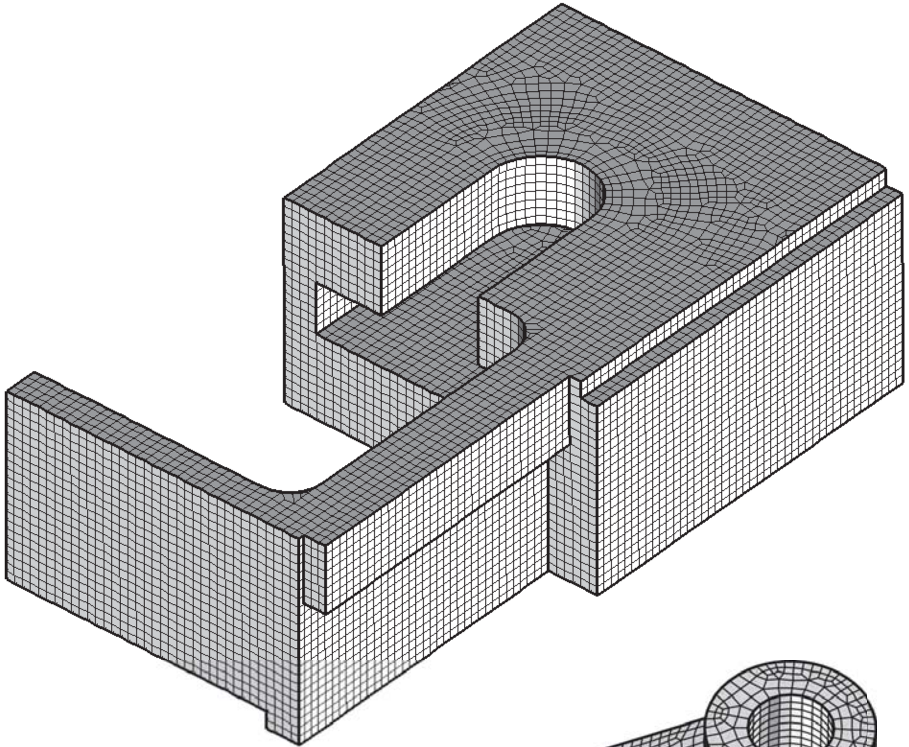
4. Structured mesh generation methods

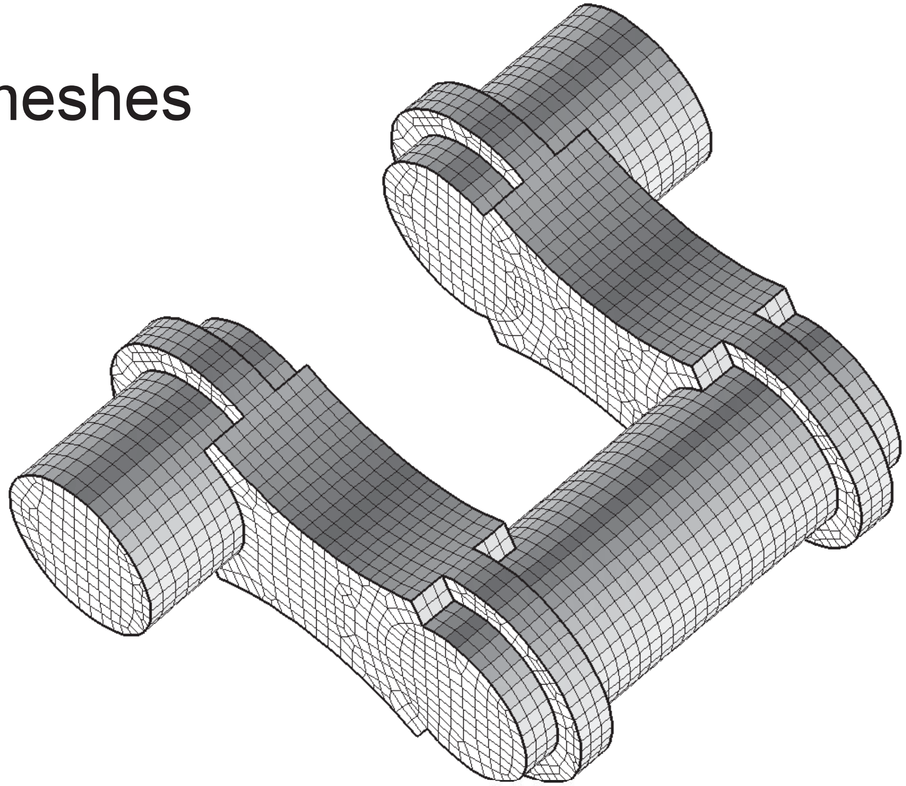
- **Some examples**: many-to-many meshes

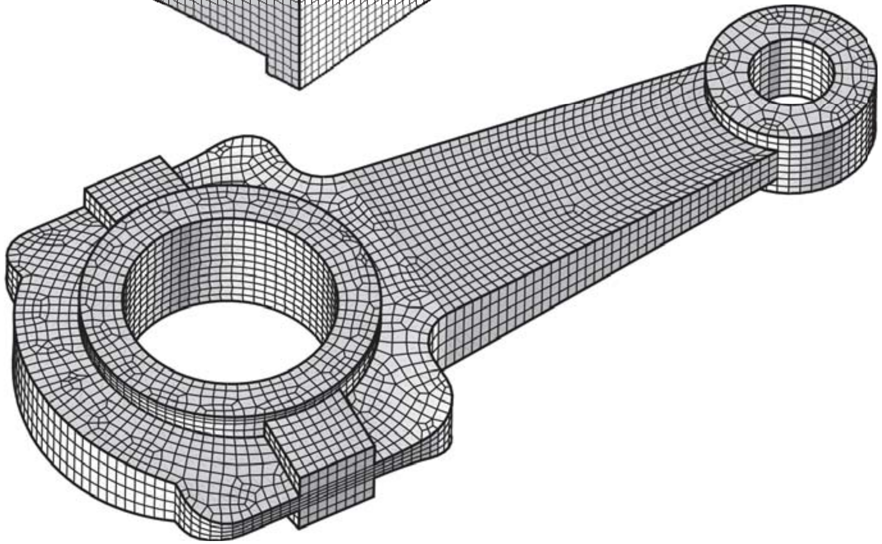


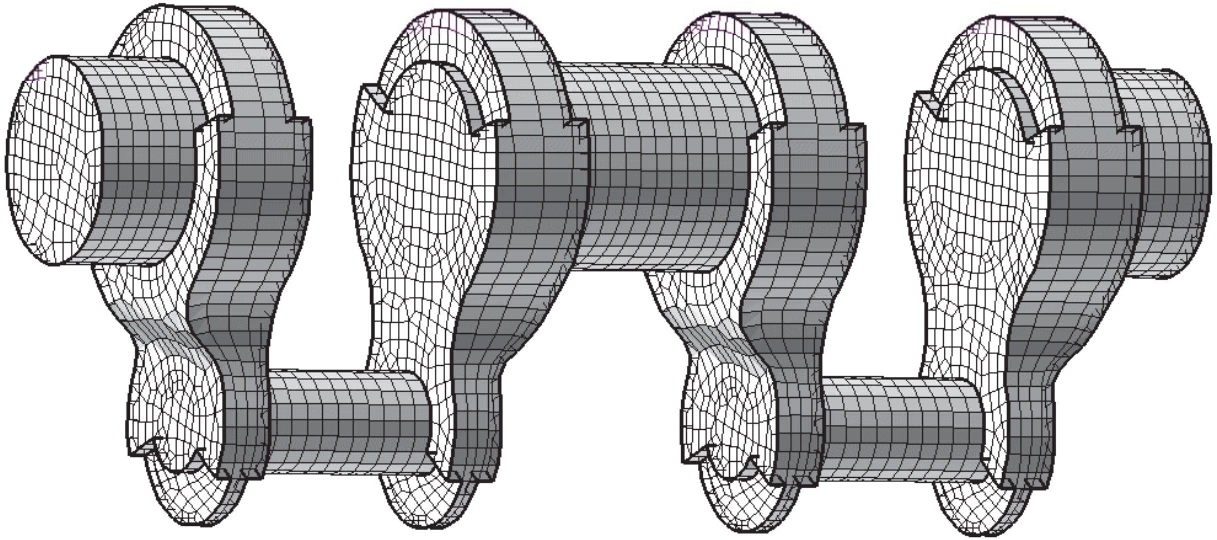
4. Structured mesh generation methods

- **Some examples**: many-to-many meshes









Layout of the course

1. Why do we need meshes?
2. Geometry description
3. Classification of mesh generation methods
4. Structured mesh generation methods
5. Unstructured mesh generation methods
6. Mesh optimization and mesh adaption
7. Concluding remarks

5. Unstructured mesh generation methods

- **Classification**
  - Methods for triangular and tetrahedral meshes
    - Tree-based methods
    - Advancing front
    - Delaunay
    - Combined approaches (Advancing-front Delaunay approach)
  - Methods for quadrilateral and hexahedral meshes
    - Indirect methods
      - Qmorh / Hmorph
      - Blossom-quad
    - Direct methods
      - Grid based
      - Medial axis
      - Paving / Plastering
      - Cross field

5.a. triangular and tetrahedral meshing

- **Triangular and tetrahedral mesh generation**

  Triangular and tetrahedral meshes are preferred by several authors because:
  - Easier to adapt to geometric features of the model
  - Easier to adapt to large gradients in the element size field
  - Accept local refinement easier to use in an adaptive strategy

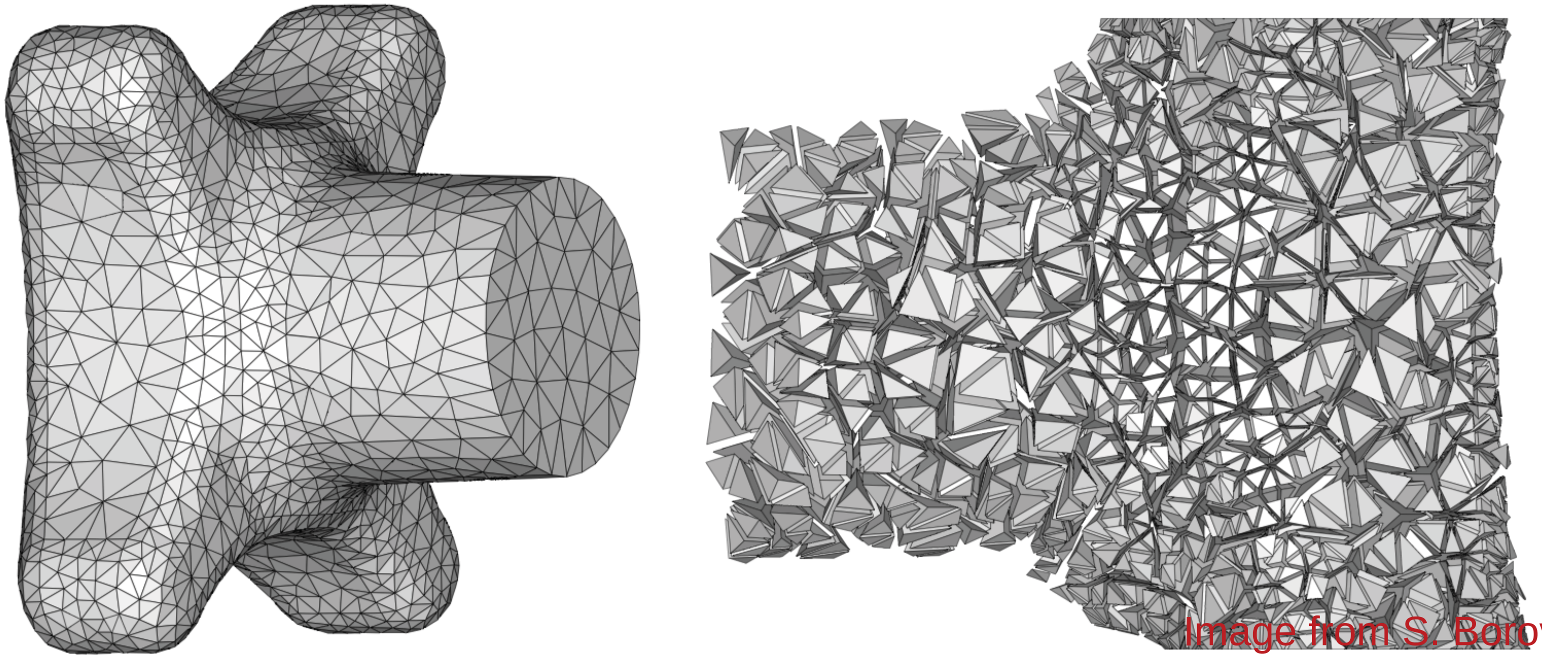


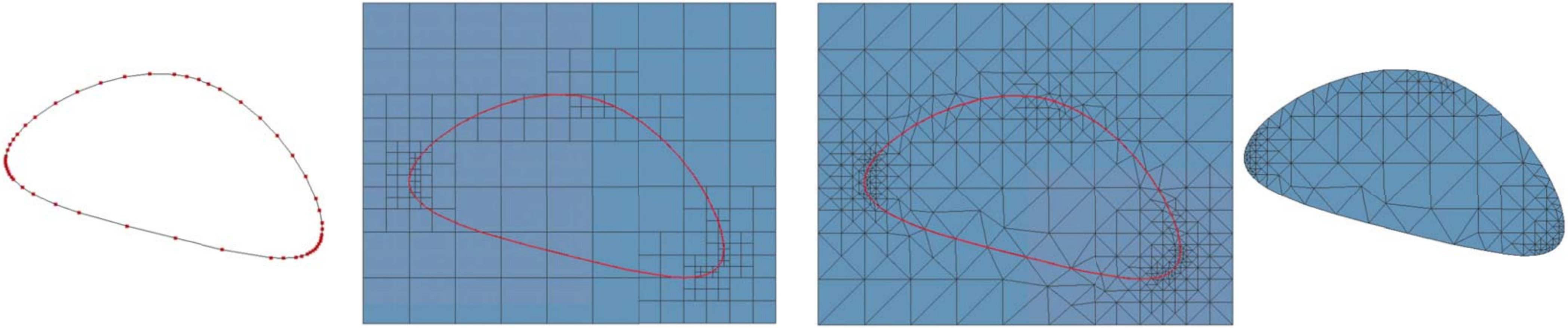
Image from S. Borovikov et al.

5.a. triangular and tetrahedral meshing

- **Classification**
  - Tree-based methods
    - Use trees to describe the geometry
    - Geometry details and size field are caught by tree refinement
    - Use templates to catch the geometry
  - Advancing-front methods
    - Starting at the boundaries, new layers of elements are added (outside-to-inside-approach)
    - High quality meshes (specially for boundary layers)
    - Efficient and robust
  - Delaunay methods:
    - Given an initial triangulation, new nodes are added according to Delaunay criteria (therefore, the connectivity is updated)
    - Theoretical results about the minimum angle of the triangulation
    - Efficient and robust
  - Combined approaches (Advancing-front Delaunay approach)
    - Increase the efficiency (intersection checking routine, …) and robustness (merging fronts, …) of the advancing-front method

5.a. triangular and tetrahedral meshing

- **Octree-based methods**
  - How it works?



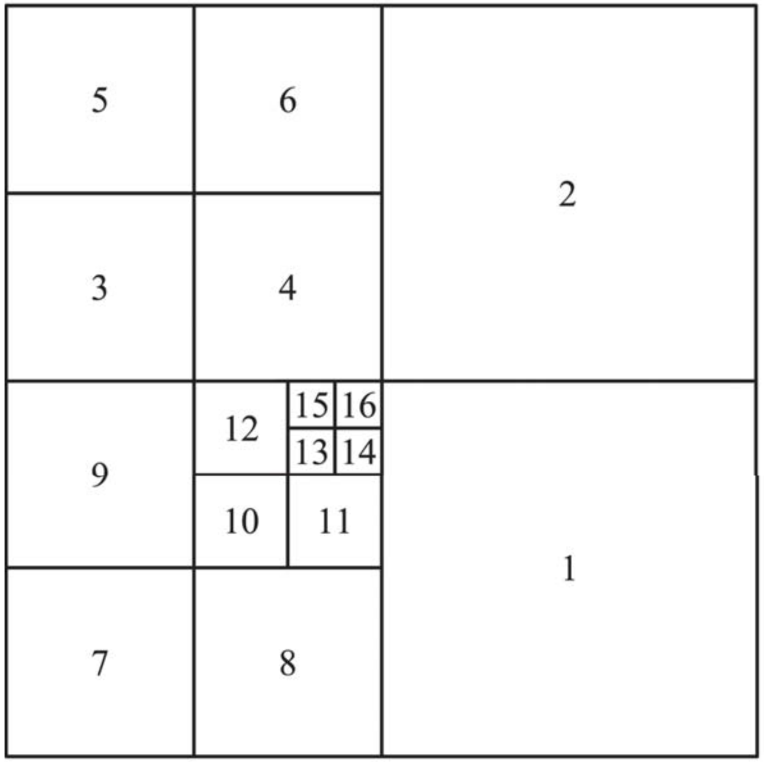
Images from X. Liang & Y. Zhang

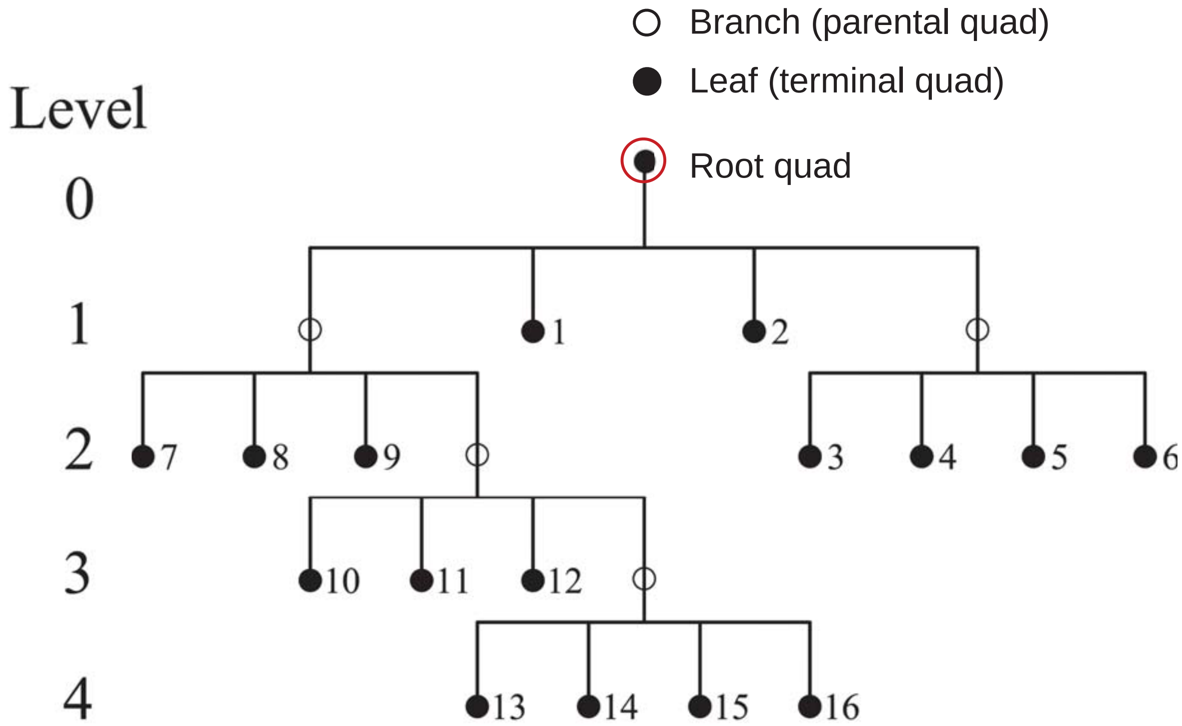
  - Basic steps
    - Create the tree
    - Generate the mesh
      - element size compatibility
      - boundary compatibility
      - cell subdivision (templates)
    - Optimize the mesh

5.a. triangular and tetrahedral meshing

- **What's a tree?**

  Hierarchic data structure (used here to localize points in space)



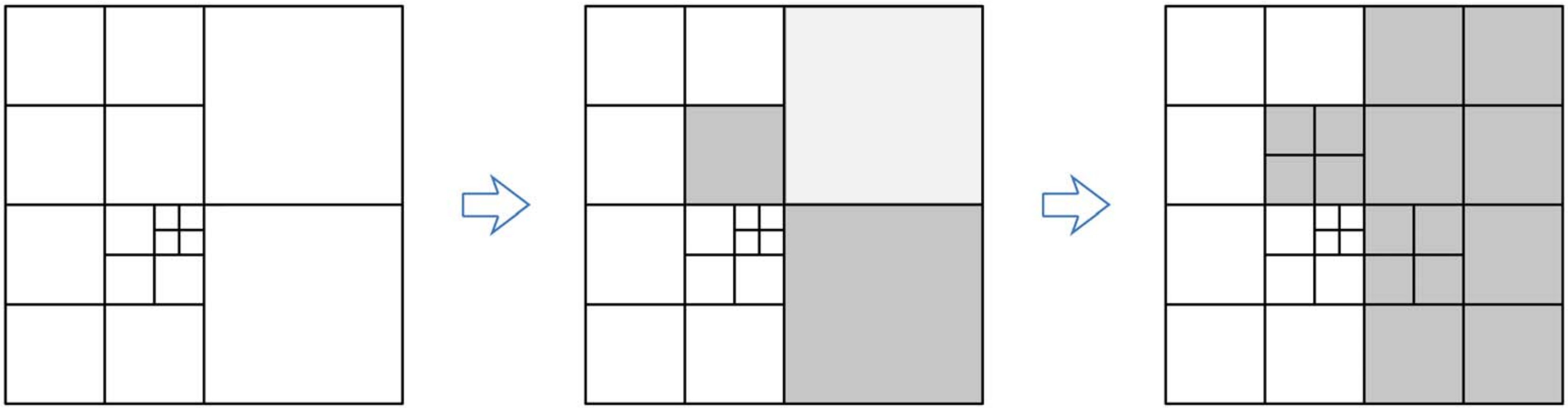


Efficient algorithms to move through the tree

- vertical (top-down, bottom-up) traversal: *O(N log(N))*
- horizontal (neighbor at the same level): traversal: *O(1)*

5.a. triangular and tetrahedral meshing

- One-level difference rule enforcement (tree balancing) each two cells sharing at least an edge are at the **same** or **subsequent** levels of hierarchic tree data structure
  -



Smoother transition in the element size !

5.a. triangular and tetrahedral meshing

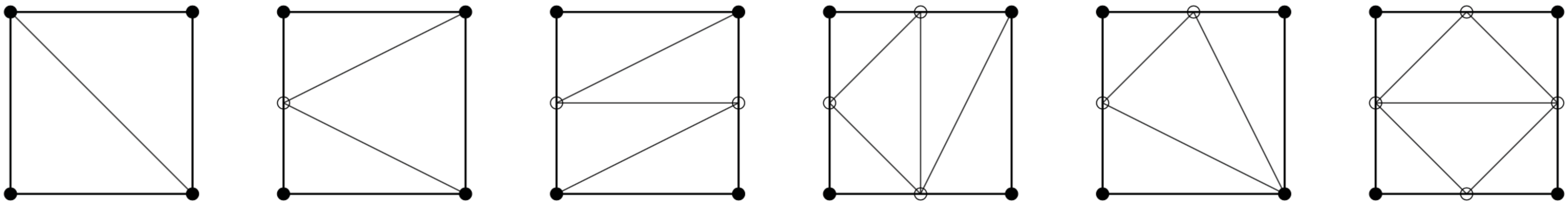
- Boundary refinement using a recursive subdivision until prescribed tolerance
  - desired cell size, desired cell level
  - geometry features (curvature, geometric details, …)

- Global refinement by recursive subdivision according to the prescribed element size (background mesh, grid, sources)

- Terminal cell classification
  - interior / boundary / exterior
  - in / out test (round-off errors!!!)

5.a. triangular and tetrahedral meshing

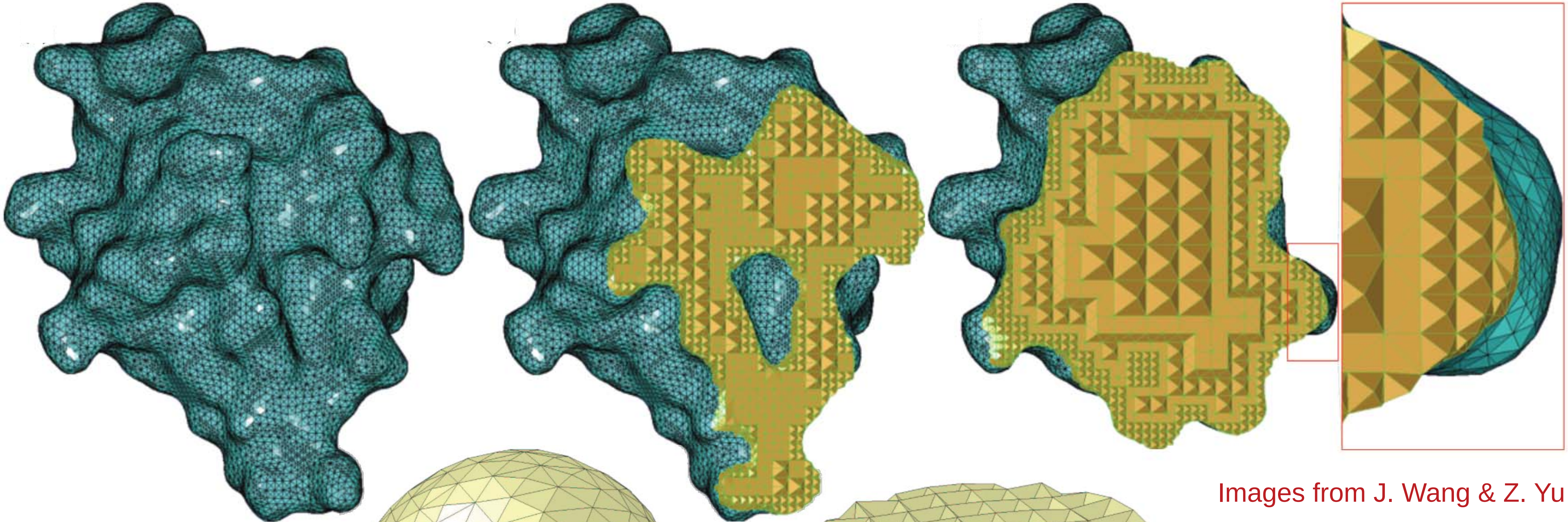
- Once we have the tree we generate the mesh:
  - Exterior cells are deleted
  - Inner cells are partitioned using templates
    - one-level difference rule ⇒ one midside maximum
    - Finite number of templates



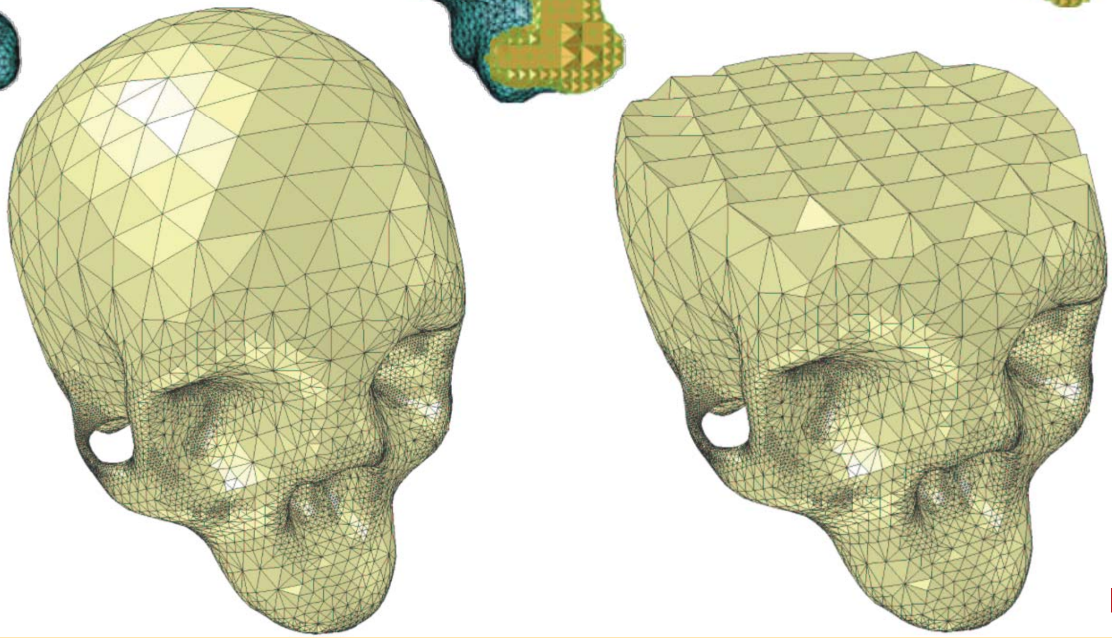
  - Boundary cells
    - We only mesh the inner part of the mesh
    - Compatibility between the boundary and cell
  - Global smoothing (no new nodes or elements !!!)

5.a. triangular and tetrahedral meshing

- **Examples**



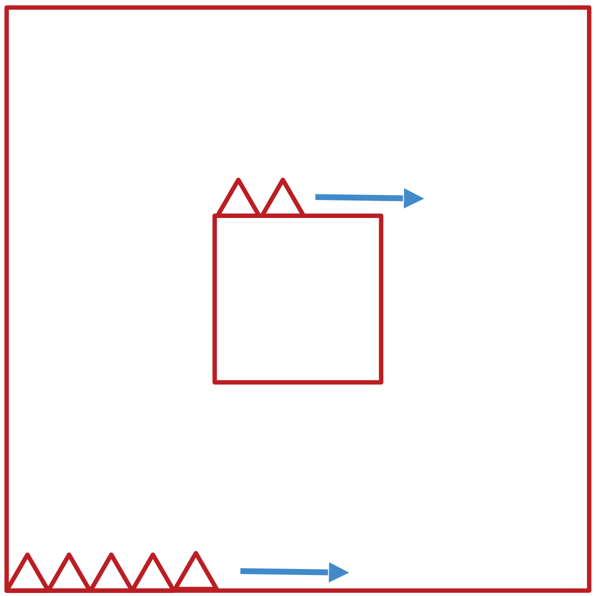
Images from J. Wang & Z. Yu



Images from N.Molino et.al.

5.a. triangular and tetrahedral meshing

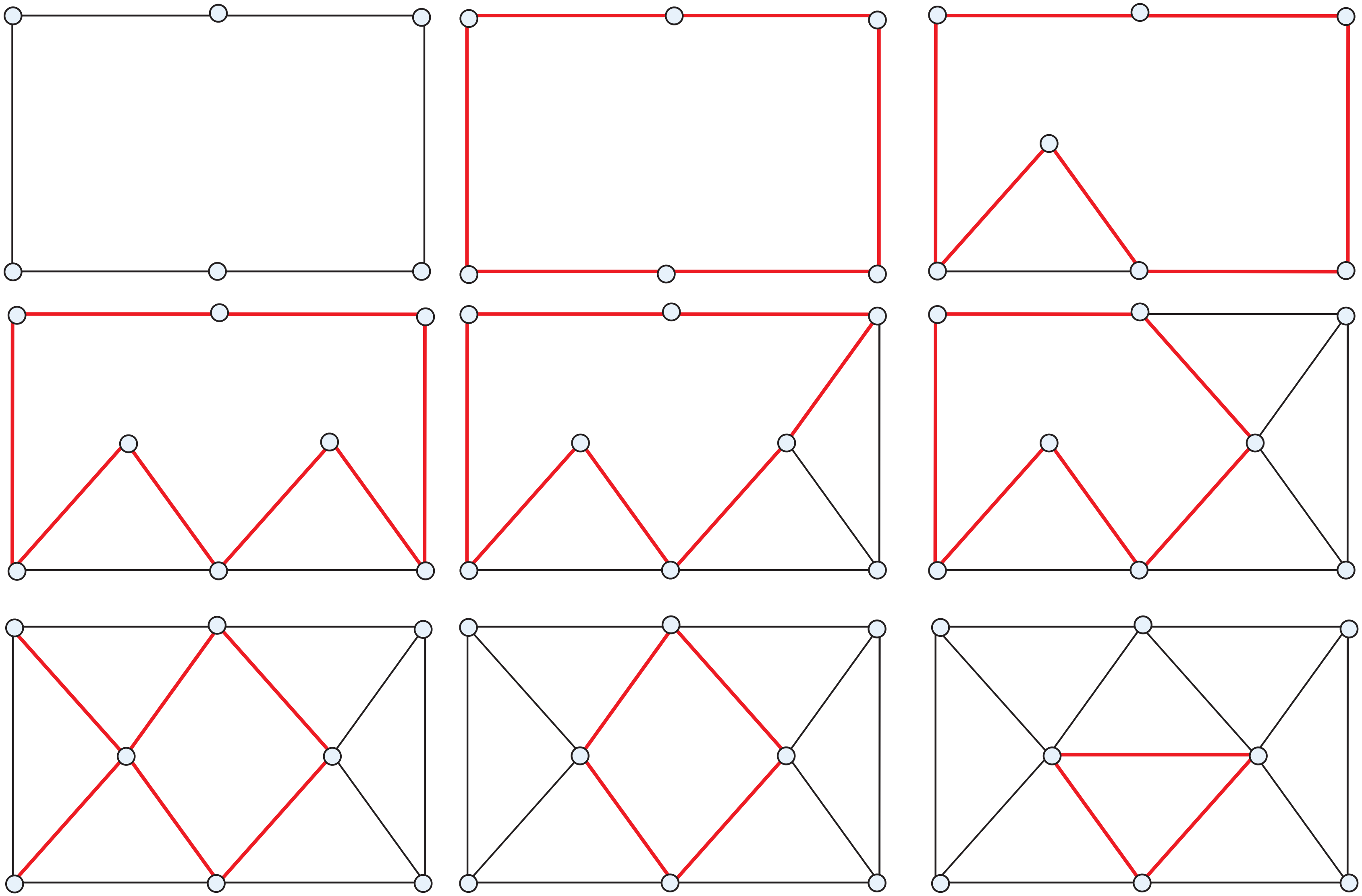
- **Advancing front method**
  - Pre-meshed boundary
  - Layers advance inward from the boundaries ⇨ nice alignment with the boundaries (boundary layers)
  - The front is a dynamic data structure (grows / reduces / appears / disappears)
  - Several fronts may exist



5.a. triangular and tetrahedral meshing

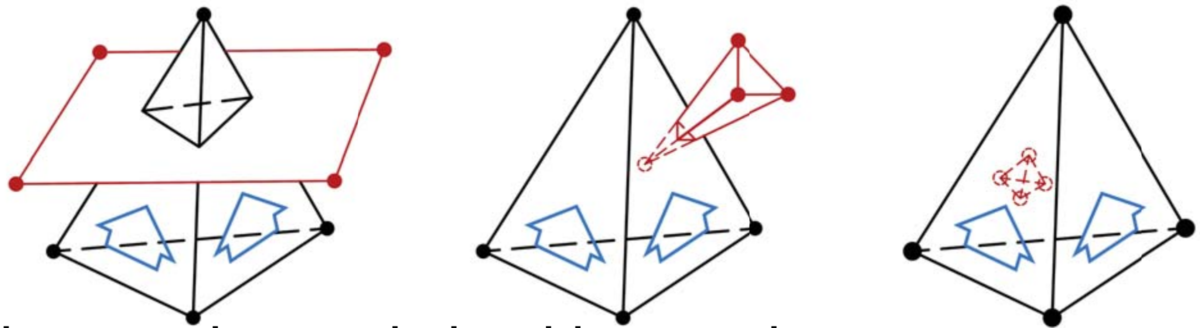
- **Main algorithm**
  1. Data input (boundary mesh, *T*, and element-size field)
  2. Initialization of the front, *F*, with *T*
  3. Analysis of the front *F* as long as *F* is not empty
     - Select a front entry, *f* (based on a criterion)
     - Determine the best point position P_{opt}
     - Determine if a point P exists in the current mesh that should be used instead of P_{opt}
     - Generate element *K* using *f* and P_{opt}
     - Check if element *K* intersects any mesh entity.
  4. Update the front and the current mesh
     - Remove f from front F and any entity of F used to form K
     - Add those entities of the new element K that belongs to the new front
     - Update the current mesh T
  5. If the front is not empty, return to step 3
  6. Mesh optimization

- **Algorithm illustration**



5.a. triangular and tetrahedral meshing

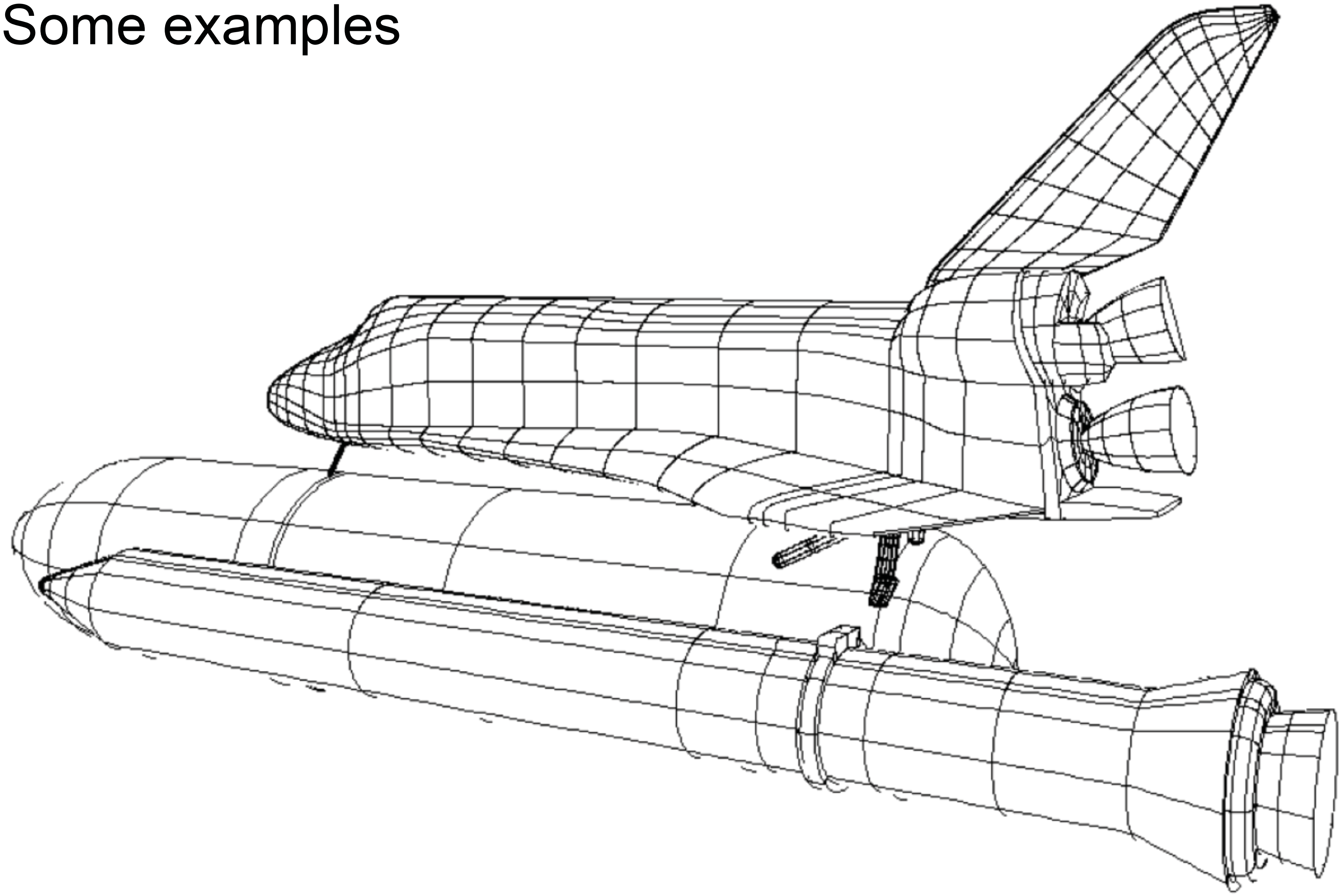
- Critical aspects of the method:
  - Robustness.
    - The identification of the local situation at some neighborhood of a point
    - Numerical precision (round off errors): checking for intersection of edges, faces, …



  - Processing time.
    - Extensive searching, sorting and checking routines
    - Efficient data structure: efficient access to the “neighborhood” of given entity (Alternating Digital Tree, ADT)
  - Quality.
    - No theoretical results on the quality of the final mesh
    - Isotropic and anisotropic meshes

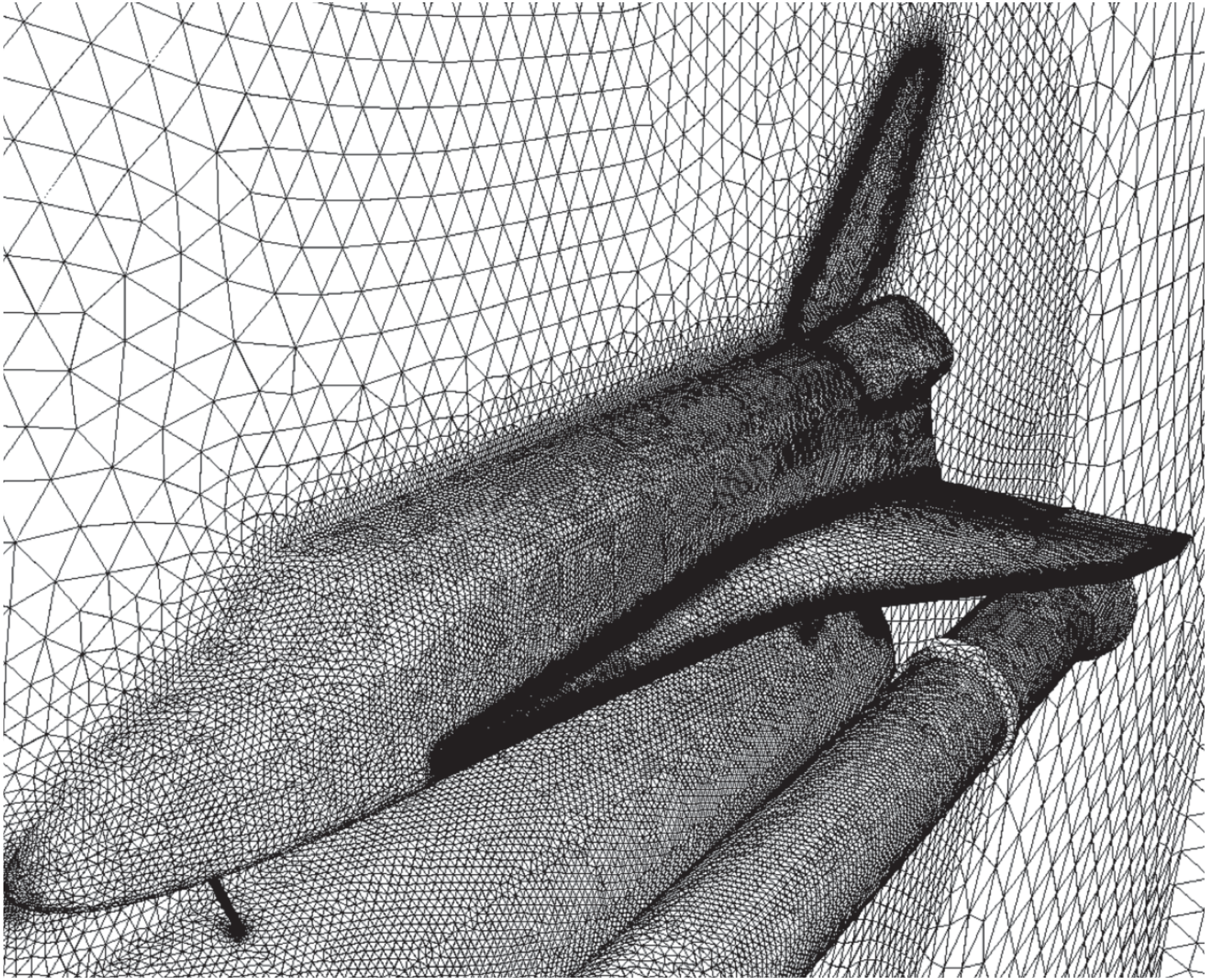
5.a. triangular and tetrahedral meshing

- Some examples



Images from R. Lohner

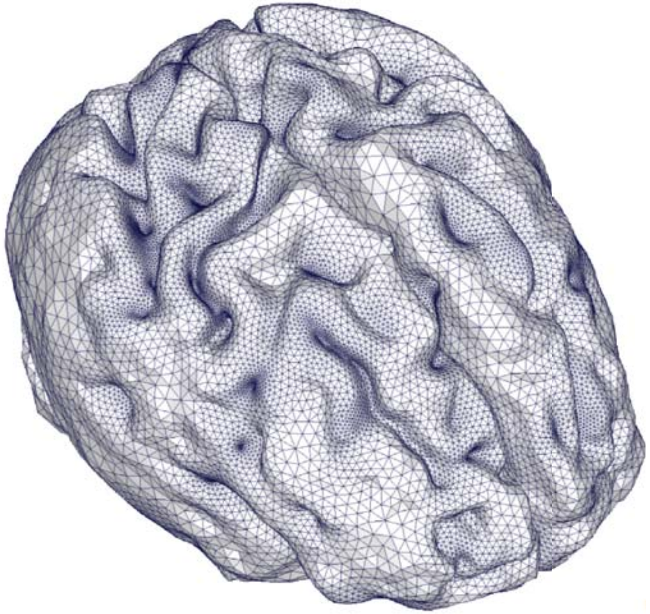
5.a. triangular and tetrahedral meshing

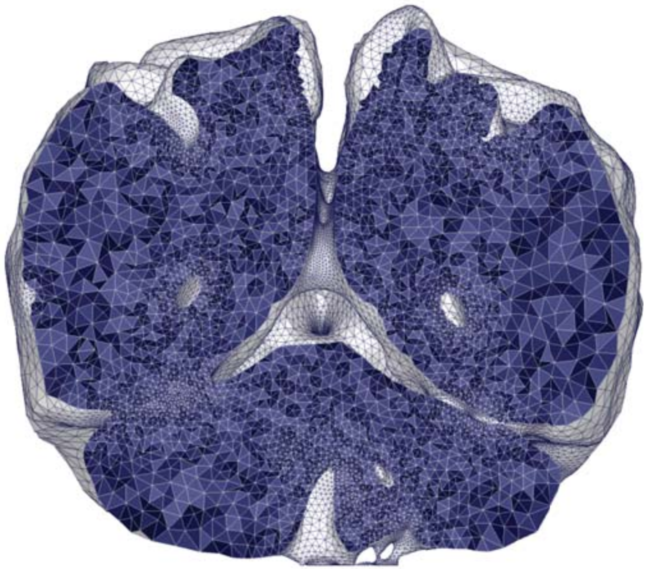
- 

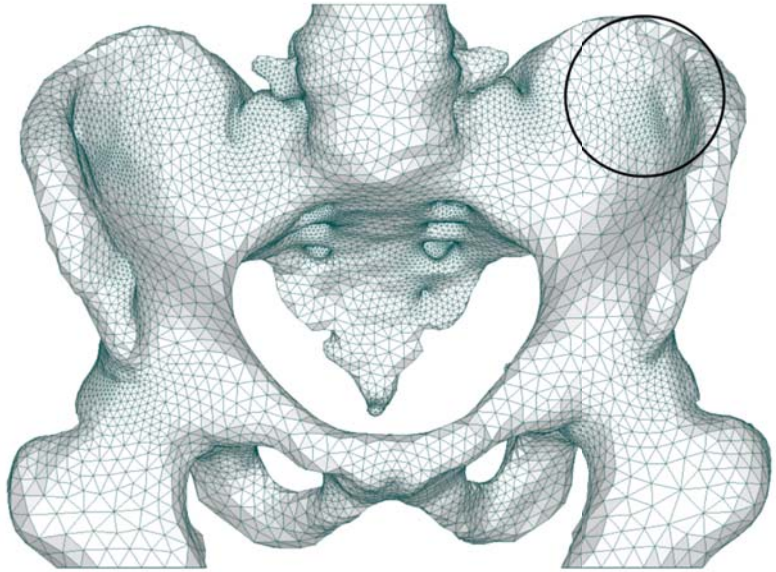
Images from R. Lohner

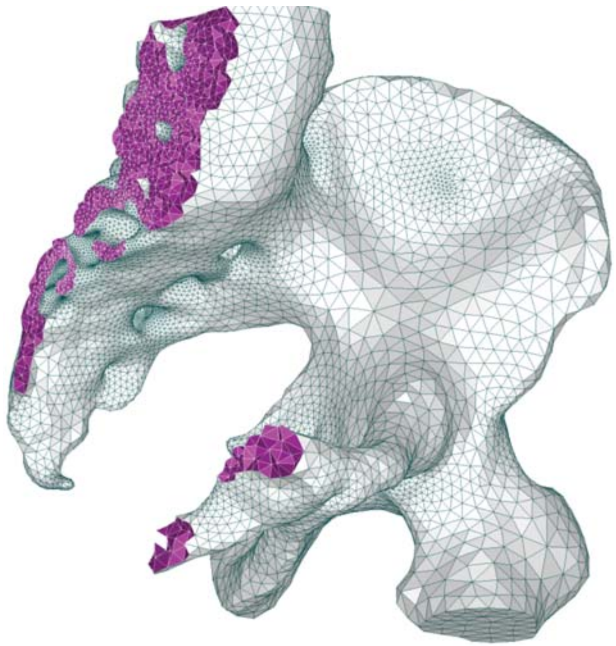
5.a. triangular and tetrahedral meshing

- Some examples









Images from
Y. Ito et al

Delaunay based methods

Given a set of *s* points $S = \{\boldsymbol{P}_i\}_{i=1,\ldots,s} \in \mathbb{R}^n, \quad \text{with } n \geq 2$
we define the **Voronoi diagram** associated to *S* as the set of cells

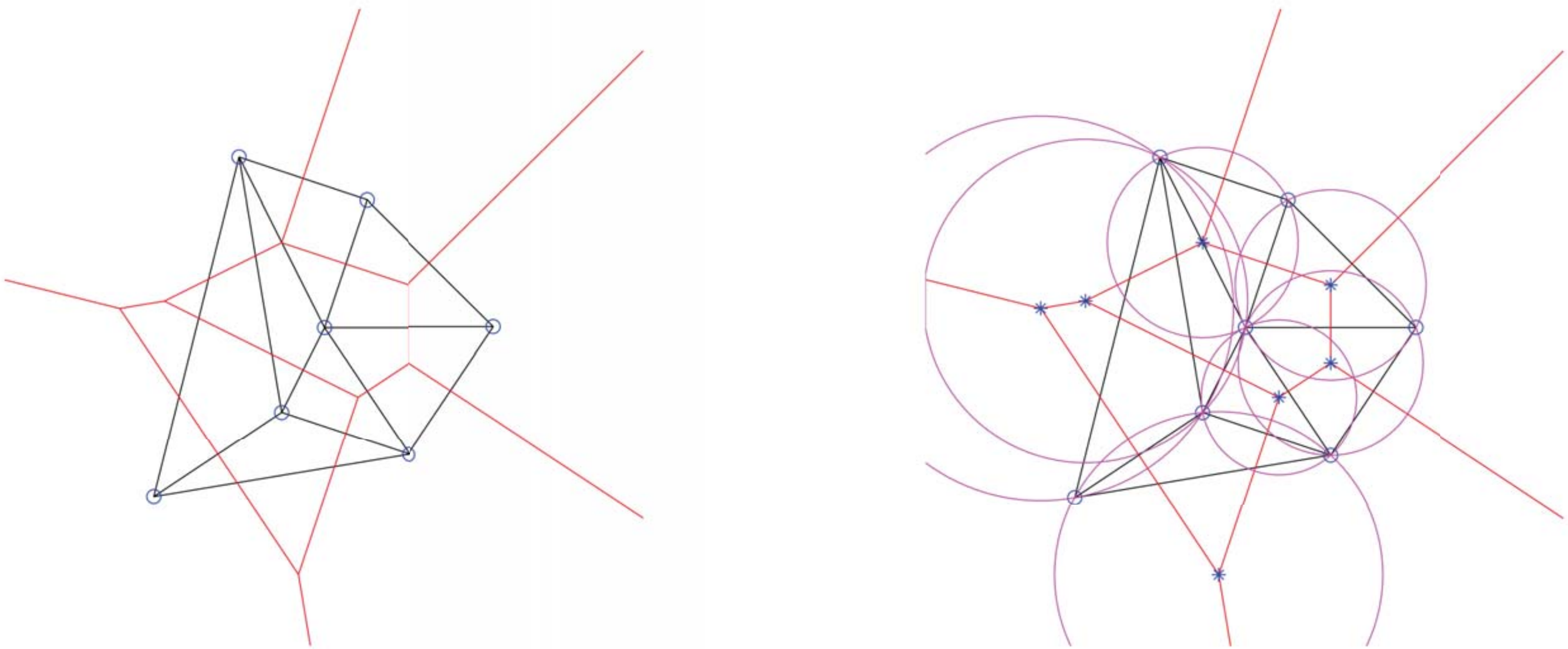
$$\boxed{V_i = \left\{\boldsymbol{P} \in \mathbb{R}^n \text{ such that } d(\boldsymbol{P}, \boldsymbol{P}_i) \leq d(\boldsymbol{P}, \boldsymbol{P}_j), \quad \forall i \neq j\right\}}$$

- V_i are the set of points closer to P_i than any other point in S
- V_i are closed (bounded or not) polygons
- They tile the space

Joining all the pairs of points P_i and P_j across polyhedral boundaries result in a triangulation of the convex hull of *S*.
This triangulation is called **Delaunay Triangulation (DT)**

Theoretical properties

Empty Circle criterion: The circumcircle around every triangle of the DT contains no vertices of the triangulation other than the three vertices that define the triangle



Min-max criterion: Of all possible triangulations of a group of vertices, the DT is the one that maximizes the minimum angle in the triangulation

Min circumcircle criterion: The DT minimizes the largest circumcircle that can be constructed around any triangle

5.a. triangular and tetrahedral meshing

Algorithms

•**Topological flipping Algorithms.** They generate a Delaunay triangulation without using the Voronoi diagram

Lawson C.L. (1977)

•**Non-incremental algorithms.** They require all vertex positions to be known in advance.

Divide and Conquer algorithms

Shanos M.I. & Hoey D. (1975)

Lee D.T. & Schachter B.J. (1980)

Guibas L. & Stolfi J. (1985)

Sweep line algorithms

Fortune S.J. (1987)

Zalik B. (2005)

5.a. triangular and tetrahedral meshing

•**Incremental algorithms (point insertion algorithms).** Vertices are added to the triangulation one at the time.

Green P.J. & Sibson R. (1978)

Bowyer & Watson algorithm

Bowyer A. (1981)

Watson D.F. (1981)

Issues:

- Insertion point criterion

  Chew P.L. (1993)

  Weatherill N.P. (1993)

  Rupert (1995)

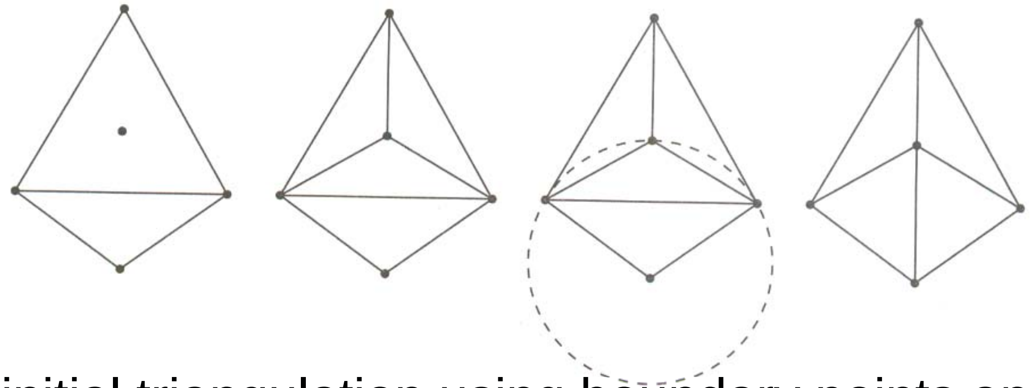
  Borouchaki H.& George PL. (1997)

- Reconnection of the inserted point to the triangulation while maintaining the Delaunay properties of the mesh.

5.a. triangular and tetrahedral meshing

- **Incremental algorithms (point insertion algorithms).**

Edge-flipping algorithm (Lawson, 1977)



Algorithm:

1. Form initial triangulation using boundary points and outer box
2. Replace an undesired element (bad or large) by inserting its circumcenter, and split it into three triangles
3. If any of the circumcircle these triangles contain the opposite corner node of a neighbouring triangle flip the diagonals
4. For every new triangle created by flipping the circumcircle test also has to be carried out
5. Repeat until mesh is good

Properties:

- Will converge with high element qualities in 2-D
- Does not extend to 3D

5.a. triangular and tetrahedral meshing

- **Incremental algorithms (point insertion algorithms).**

  Insertion polygon method (Bowyer – Watson, 1981)

Algorithm:

1. Form initial triangulation using boundary points and outer box
2. Replace an undesired element (bad or large) by inserting its circumcenter
3. Identify all triangles such that the new point falls inside their circumcenter (this enclosed polygon is called the insertion polygon)
4. Retriangulate the insertion polygon
5. Repeat until mesh is good

5.a. triangular and tetrahedral meshing

• **Incremental algorithms (point insertion algorithms).**

Insertion polygon method (Bowyer – Watson, 1981)

Algorithm:

$$\mathcal{T}_{i+1} = \mathcal{T}_i - \mathcal{C}_{\boldsymbol{P}} + \mathcal{B}_{\boldsymbol{P}}$$

$\boldsymbol{P}$	$(i+1)$th inserted point from a convex hull (the new one)
$\mathcal{T}_i$	Delaunay triangulation of first i points from a convex hull
$\mathcal{C}_{\mathbf{P}}$	cavity of $\boldsymbol{P}$, set of elements from $\mathcal{T}_i$ whose circumcicle contains $\boldsymbol{P}$
$\mathcal{B}_{\boldsymbol{P}}$	ball of $\mathbf{P}$, set of new elements generated from boundary edges of $\mathcal{C}_{\boldsymbol{P}}$ and $\boldsymbol{P}$

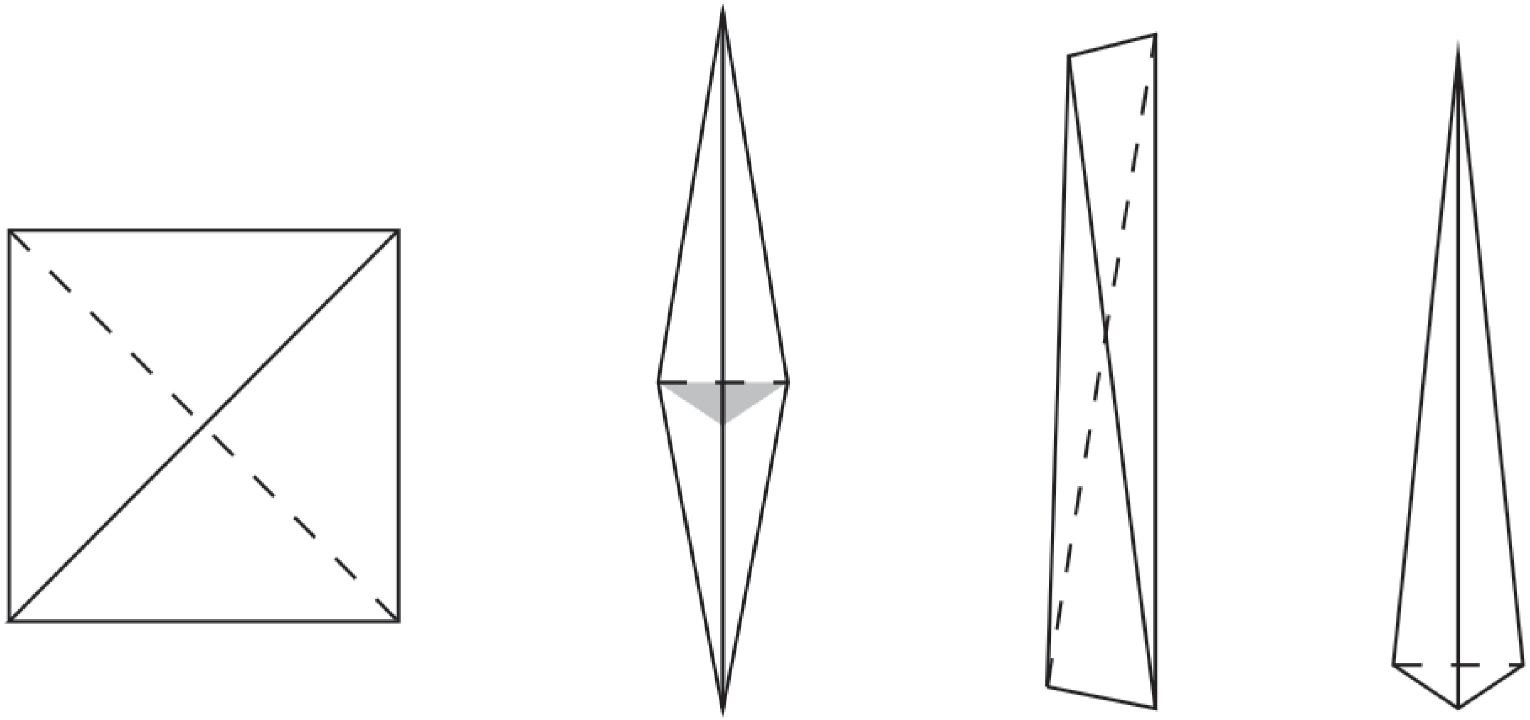
Properties:

- Will converge with high element qualities in 2-D
- Extends to 3D
- Very fast – time almost linear in number of nodes

5.a. triangular and tetrahedral meshing

Quality of the elements:

Bad shaped elements can be generated



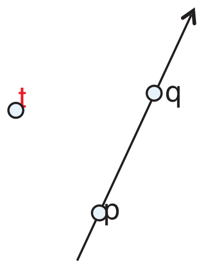
Sliver tet Spear tet Splinter tet Needle tet

Geometric predicates:

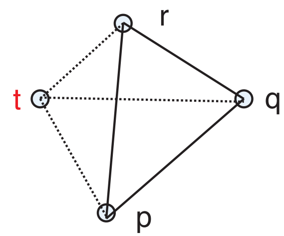
The two well-known predicates needed for Dealunay triangulations are:

- **The orientation test**

Decides on which side of the line defined by two points lies a third point



Decides on which side of the plane oriented by three non-aligned points lies a third point



$$\texttt{orientation2D(p,q,t)} = \begin{vmatrix} p_x & p_y & 1 \\ q_x & q_y & 1 \\ t_x & t_y & 1 \end{vmatrix} = \begin{vmatrix} p_x - t_x & p_y - t_y \\ q_x - t_x & q_y - t_y \end{vmatrix}$$

$$\texttt{orientation3D(p,q,r,t)} = \begin{vmatrix} p_x & p_y & p_z & 1 \\ q_x & q_y & q_z & 1 \\ r_x & r_y & r_z & 1 \\ t_x & t_y & t_z & 1 \end{vmatrix} = \begin{vmatrix} p_x - t_x & p_y - t_y & p_z - t_z \\ q_x - t_x & q_y - t_y & q_z - t_z \\ r_x - t_x & r_y - t_y & r_z - t_z \end{vmatrix}$$

$$\texttt{orientation2D(p,q,t)} \begin{cases} > 0 & \text{on the left (counter-clockwise)} \\ = 0 & \text{on the line} \\ < 0 & \text{on the righ (clockwise)} \end{cases}$$

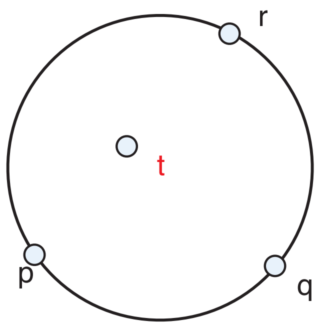
$$\texttt{orientation3D(p,q,r,t)} \begin{cases} > 0 & \text{on the left} \\ = 0 & \text{on the line} \\ < 0 & \text{on the right} \end{cases}$$

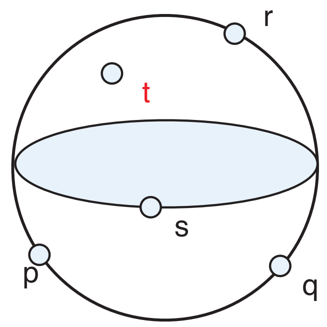
5.a. triangular and tetrahedral meshing

- **The in-sphere test**

Given three positive oriented points, decides when a fourth point lies inside the circumscribing circle of the three points

Given four positive oriented points, decides when a fifth point lies inside the circumscribing sphere of the four points





$$\texttt{inCircle(p,q,r,t)} = \begin{vmatrix} p_x & p_y & p_x^2+p_y^2 & 1 \\ q_x & q_y & q_x^2+q_y^2 & 1 \\ r_x & r_y & r_x^2+r_y^2 & 1 \\ t_x & t_y & t_x^2+t_y^2 & 1 \end{vmatrix} = \begin{vmatrix} p_x-t_x & p_y-t_y & (p_x-t_x)^2+(p_y-t_y)^2 \\ q_x-t_x & q_y-t_y & (q_x-t_x)^2+(q_y-t_y)^2 \\ r_x-t_x & r_y-t_y & (r_x-t_x)^2+(r_y-t_y)^2 \end{vmatrix}$$

$$\texttt{inSphere(p,q,r,s,t)} = \begin{vmatrix} p_x & p_y & p_z & p_x^2+p_y^2+p_z^2 & 1 \\ q_x & q_y & q_z & q_x^2+q_y^2+q_z^2 & 1 \\ r_x & r_y & r_z & r_x^2+r_y^2+r_z^2 & 1 \\ s_x & s_y & s_z & s_x^2+s_y^2+s_z^2 & 1 \\ t_x & t_y & t_z & t_x^2+t_y^2+t_z^2 & 1 \end{vmatrix}$$

$$= \begin{vmatrix} p_x-t_x & p_y-t_y & p_z-t_z & (p_x-t_x)^2+(p_y-t_y)^2+(p_z-t_z)^2 \\ q_x-t_x & q_y-t_y & q_z-t_z & (q_x-t_x)^2+(q_y-t_y)^2+(q_z-t_z)^2 \\ r_x-t_x & r_y-t_y & r_z-t_z & (r_x-t_x)^2+(r_y-t_y)^2+(r_z-t_z)^2 \\ s_x-t_x & s_y-t_y & s_z-t_z & (s_x-t_x)^2+(s_y-t_y)^2+(s_z-t_z)^2 \end{vmatrix}$$

$$\texttt{inCircle(p,q,r,t)} \begin{cases} >0 & \text{inside the circle} \\ =0 & \text{on the circumference} \\ <0 & \text{outside the circle} \end{cases}$$

$$\texttt{inSphere(p,q,r,s,t)} \begin{cases} >0 & \text{inside the sphere} \\ =0 & \text{on the sphere} \\ <0 & \text{outside the sphere} \end{cases}$$

To make the incremental algorithms more robust it is needed to incorporate:

- More checking and correction procedures

  [Borouchaki H, George PL, Lo SH, IJNME 39 3407-3437 (1996)]

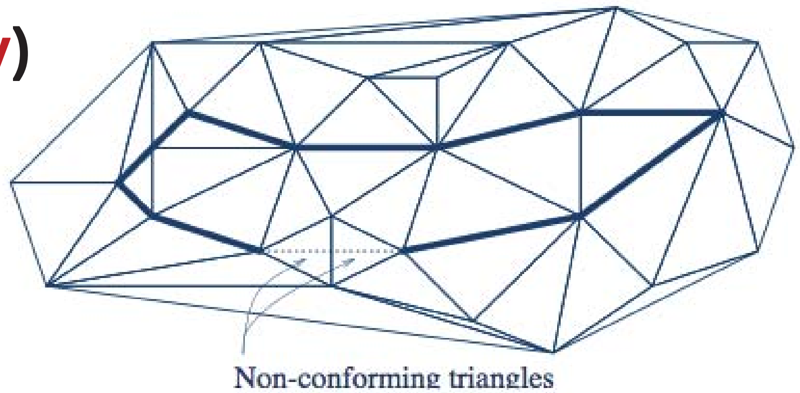
- Adaptive precision floating point arithmetic and fast robust geometric predicates

  [Shewchuck JR. “Delaunay Refinement mesh generation” PhD thesis, Carnegie Mellon University, Pittsburgh, USA (1997)]
  [http://www.eecs.berkeley.edu/~jrs/]

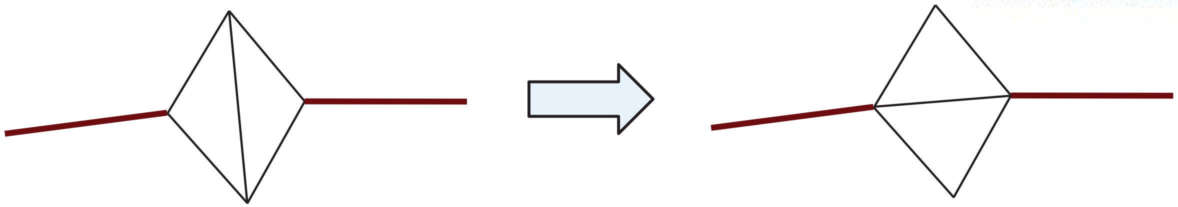
5.a. triangular and tetrahedral meshing

Boundary recovery (Constrained Delaunay)

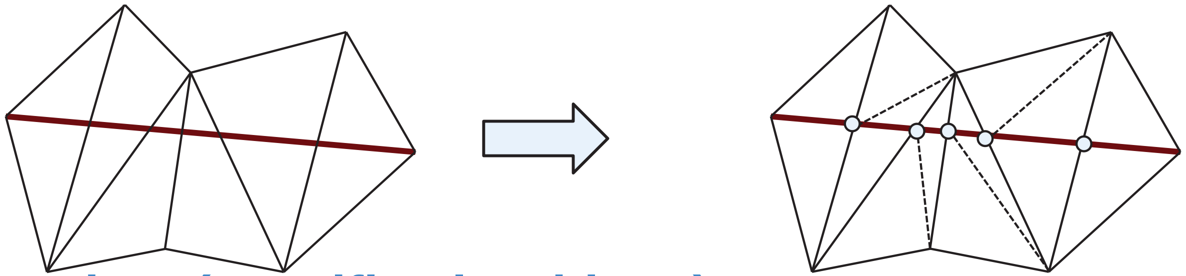
For non-convex domains Dalaunay triangulation may not conform to the boundary or might not respect a given set of edges



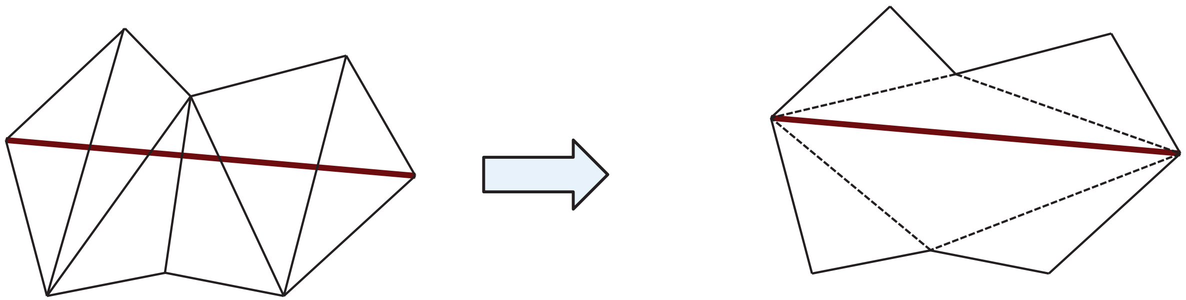
Local edge flip



Point insertion (constraint partitioning)



Enforcing the constraints (specific algorithms)



5.a. triangular and tetrahedral meshing

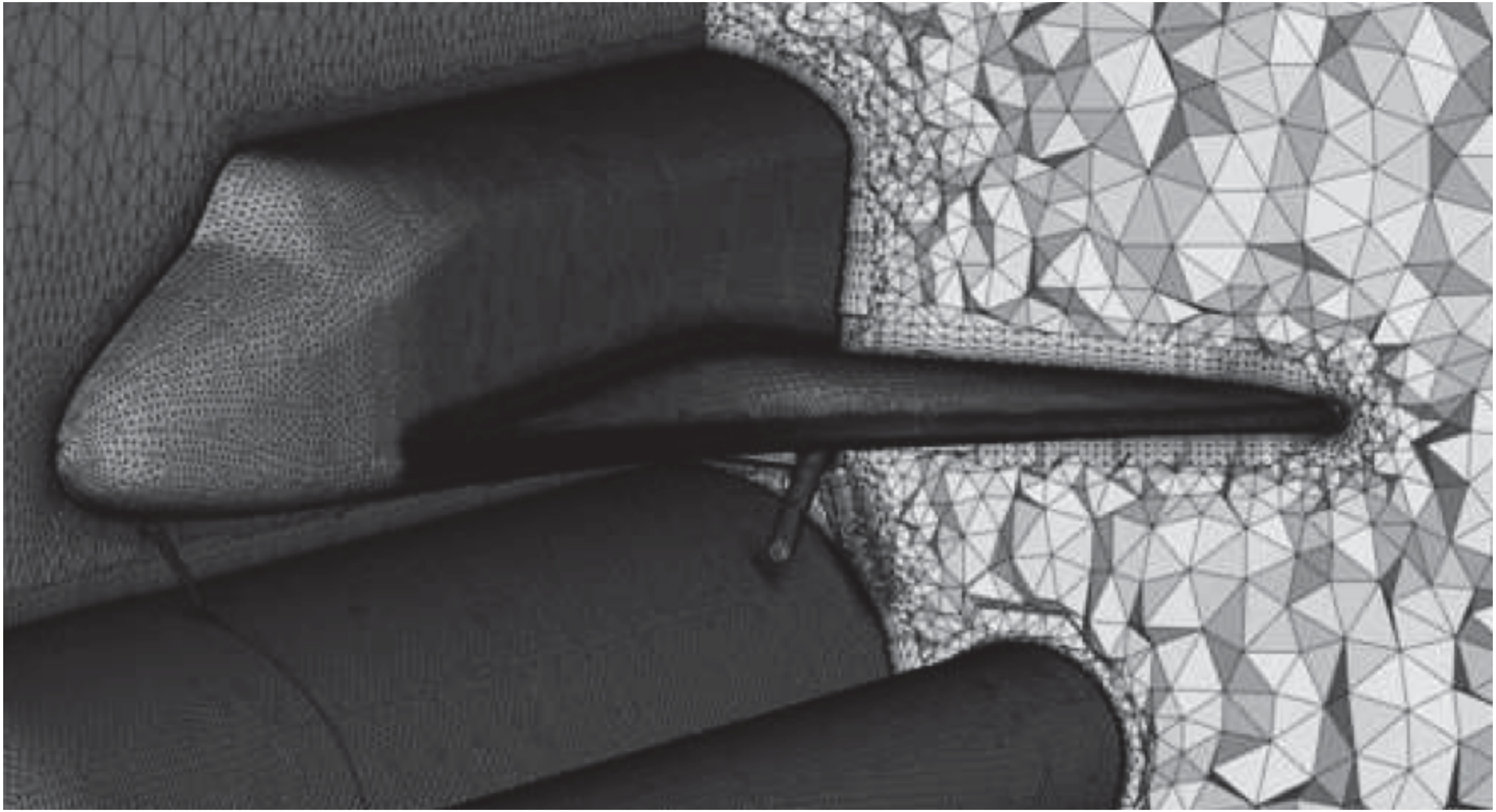


Image from A. Loseille and R. Löhner

5.a. triangular and tetrahedral meshing

- **Some examples**

Image from F. Alauzet and D. Marcum

5.a. triangular and tetrahedral meshing

- **Some examples**

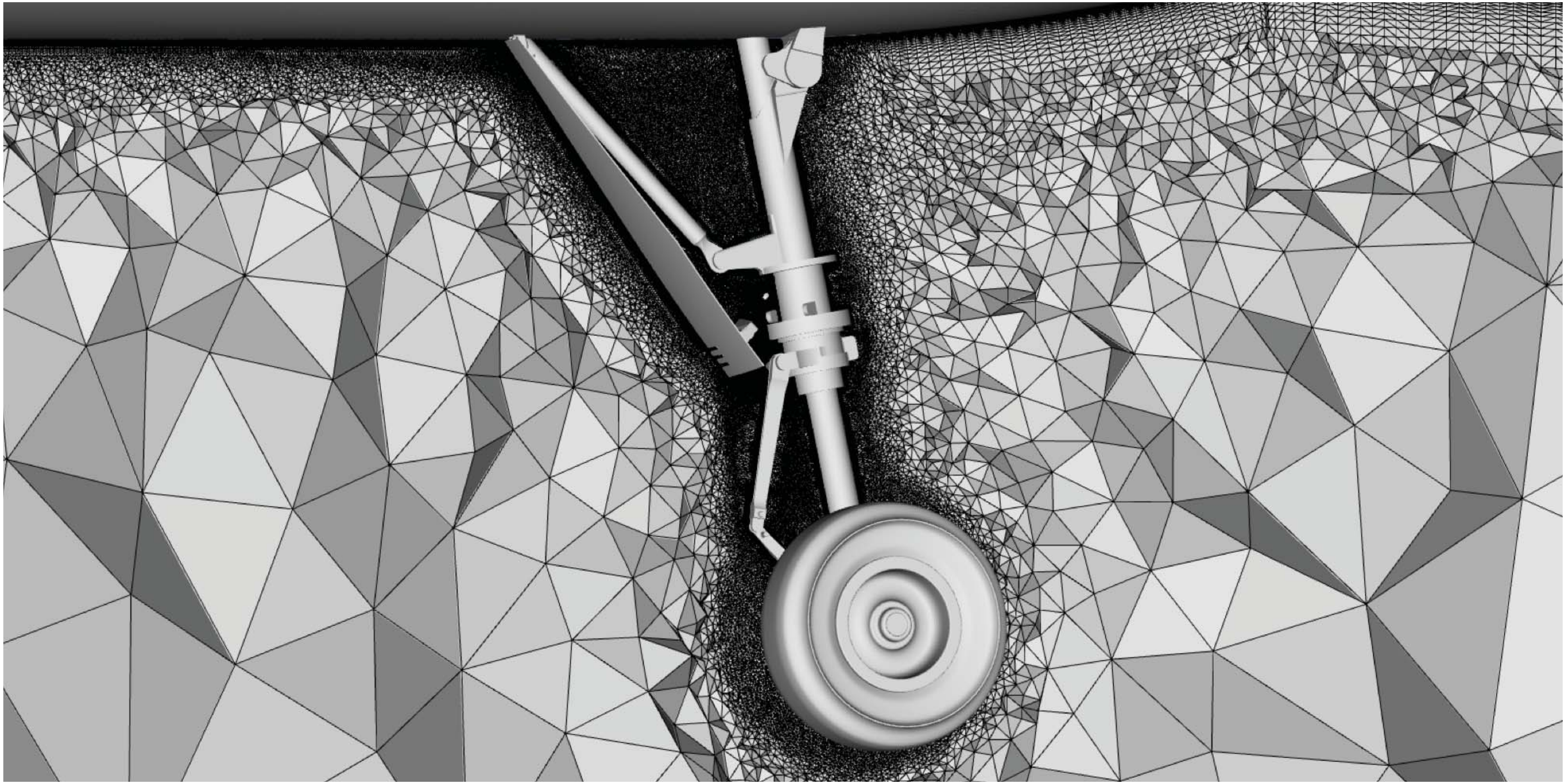


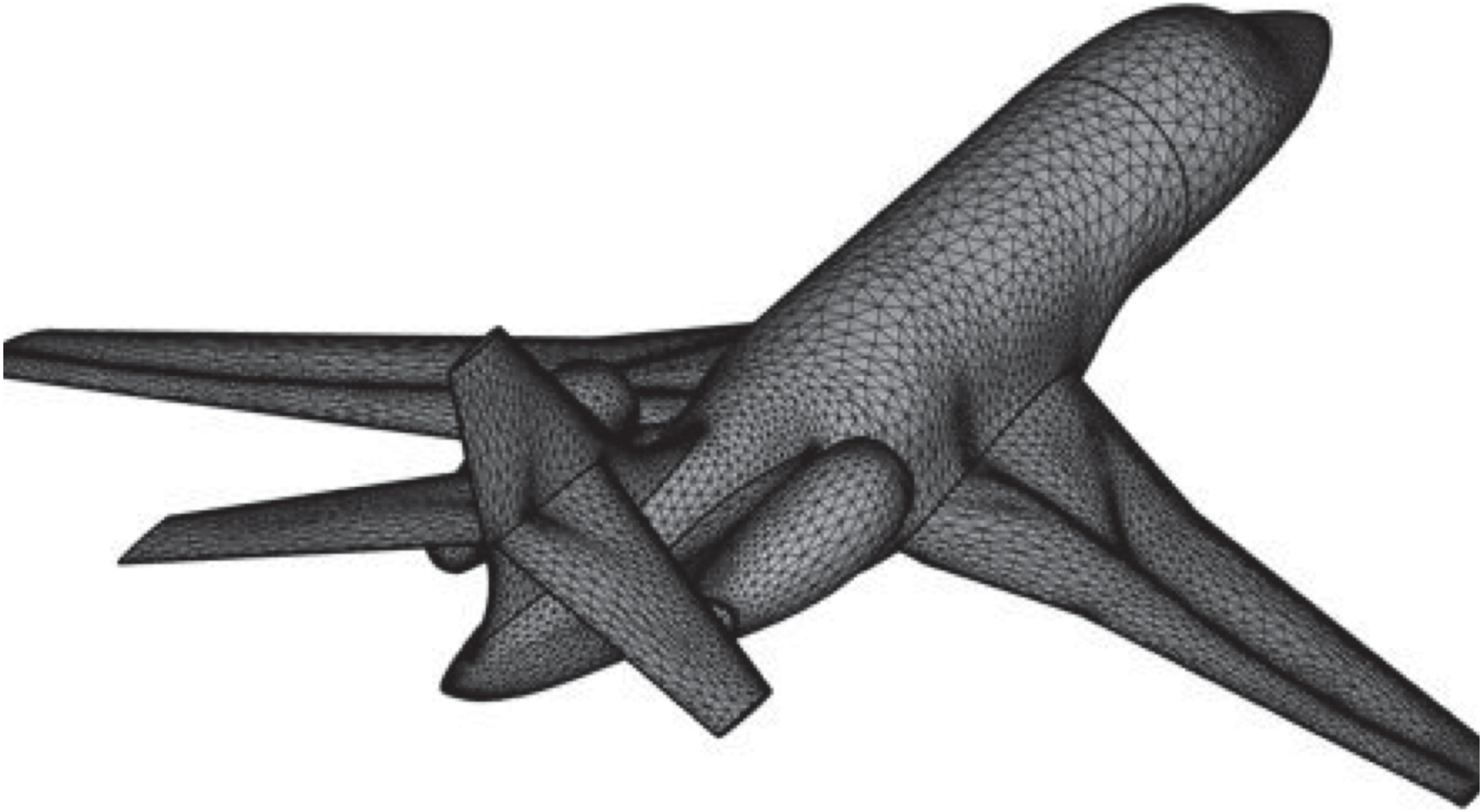
Image from F. Alauzet and D. Marcum

5.a. triangular and tetrahedral meshing

- **Some examples**

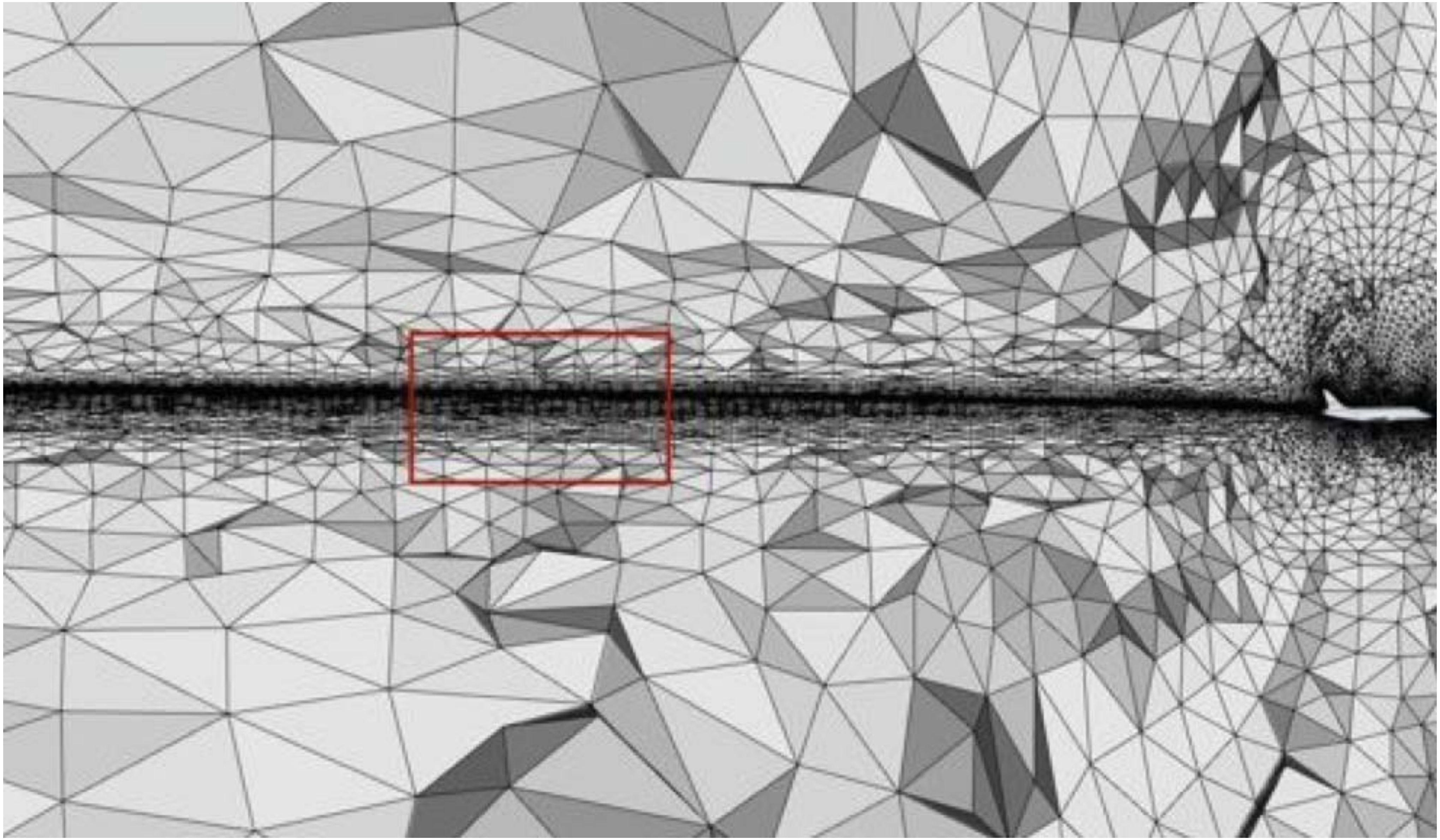
Image from F. Alauzet and D. Marcum

5.a. triangular and tetrahedral meshing



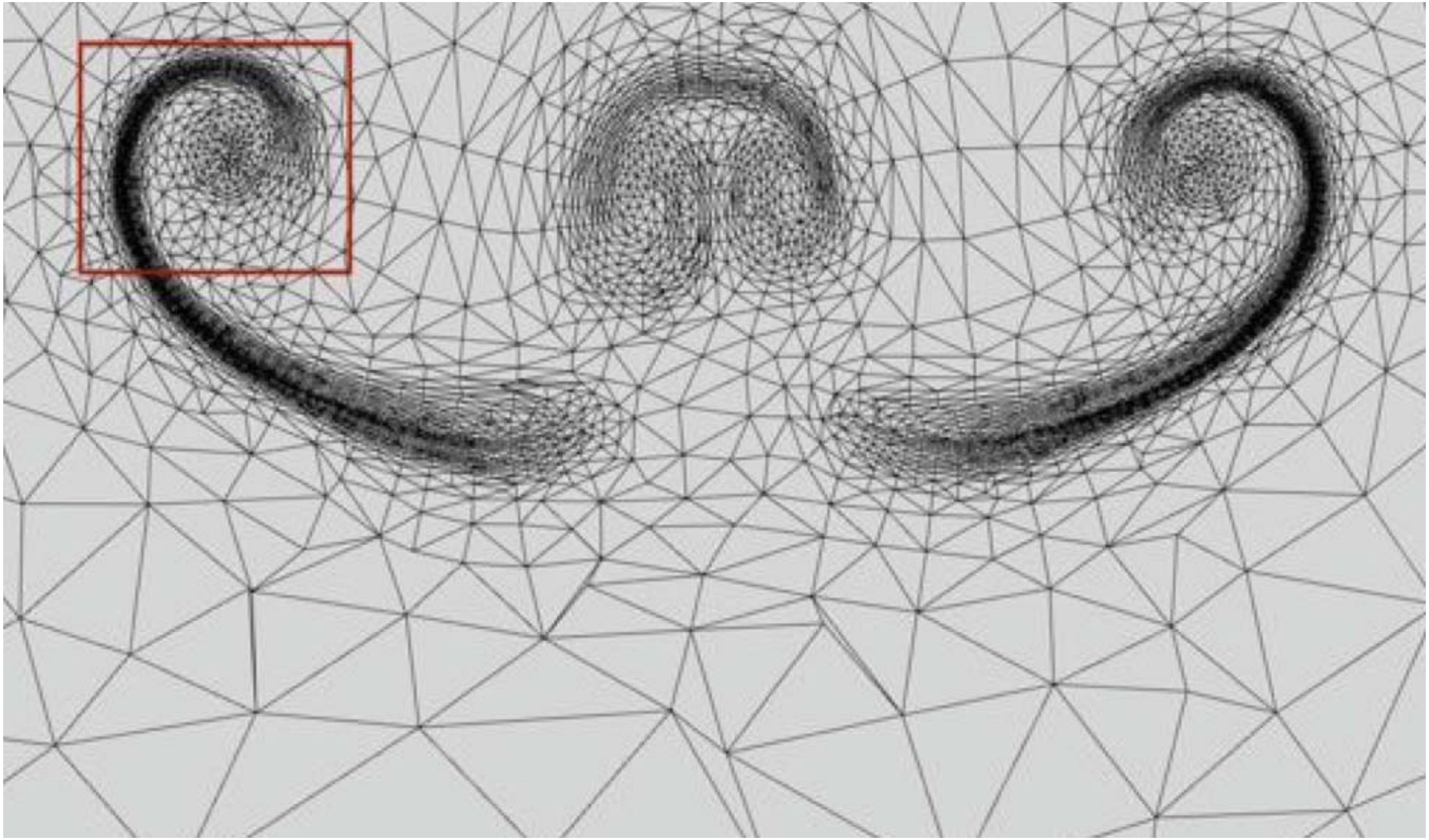
Images from A. Loseille

5.a. triangular and tetrahedral meshing



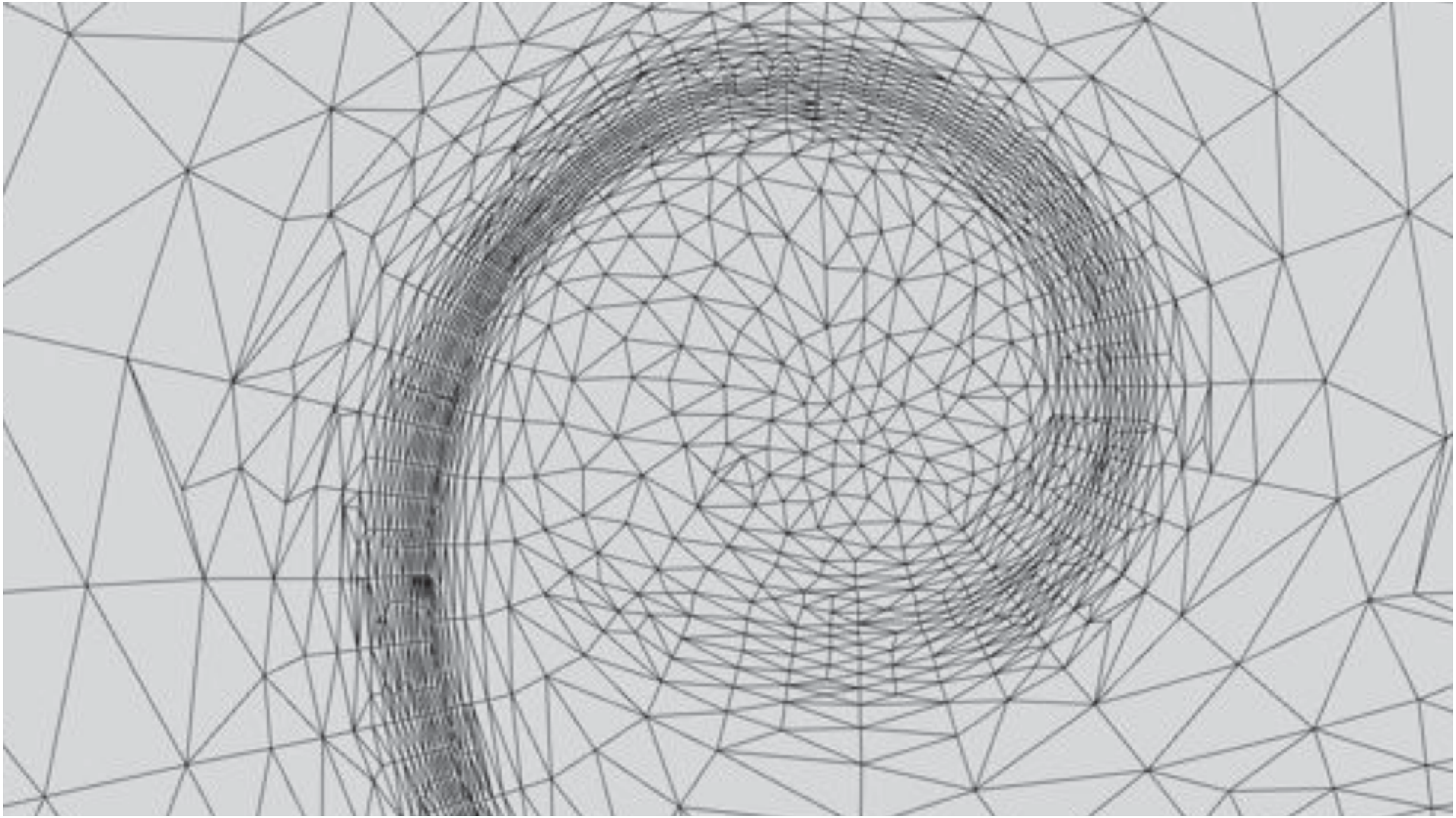
Images from A. Loseille

5.a. triangular and tetrahedral meshing



Images from A. Loseille

5.a. triangular and tetrahedral meshing



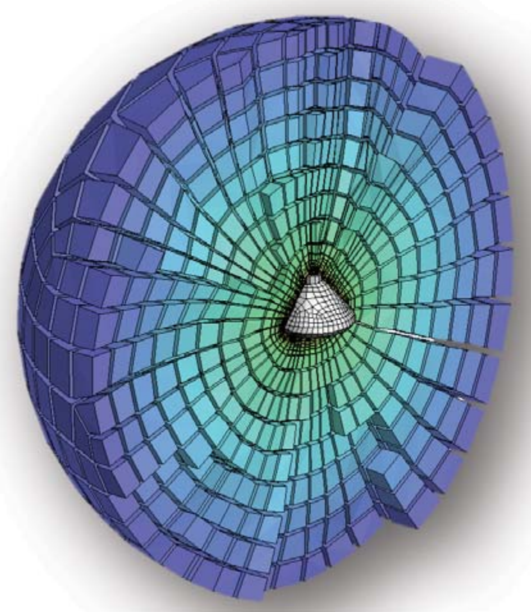
Images from A. Loseille

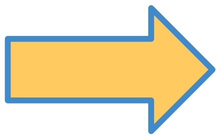
5.b. Quadrilateral and hexahedral meshing

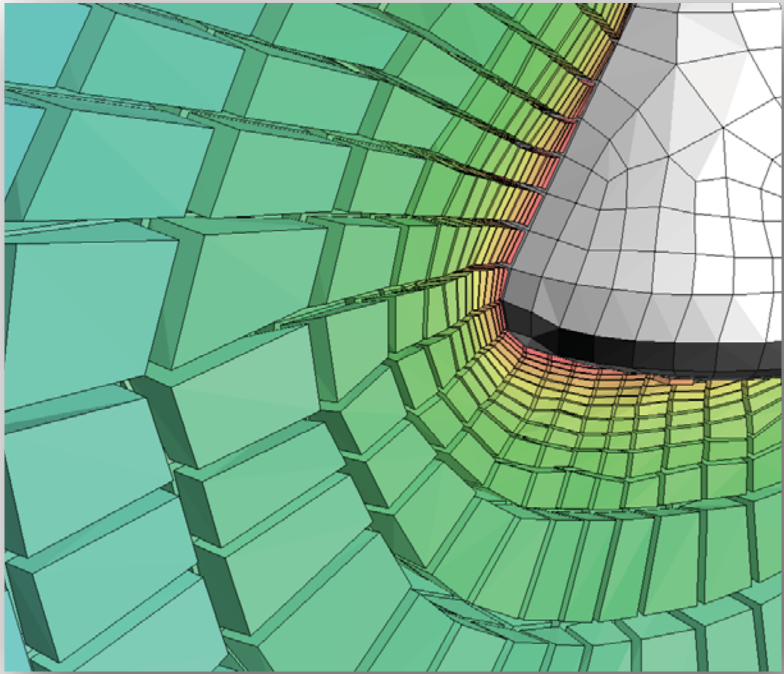
- **Quadrilateral and hexahedral mesh generation**

  Quadrilateral and hexahedral meshes are more constrained, and therefore much more difficult to generate. However:

  - Preferred by several authors (mixed formulations)
  - Perform better in some applications where a strict alignment of elements can be required by the analysis:
    - boundary layers in computational fluid dynamics
    - composites in solid mechanics

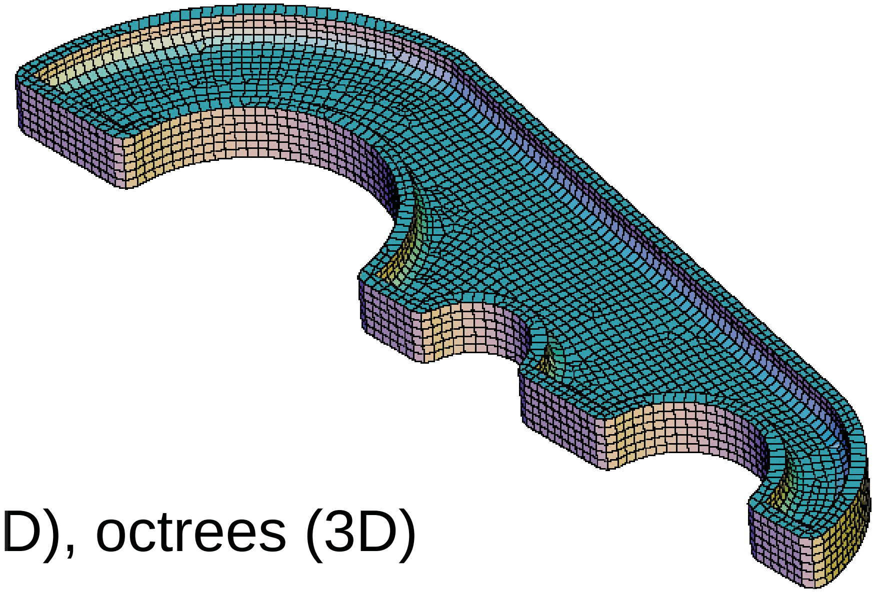


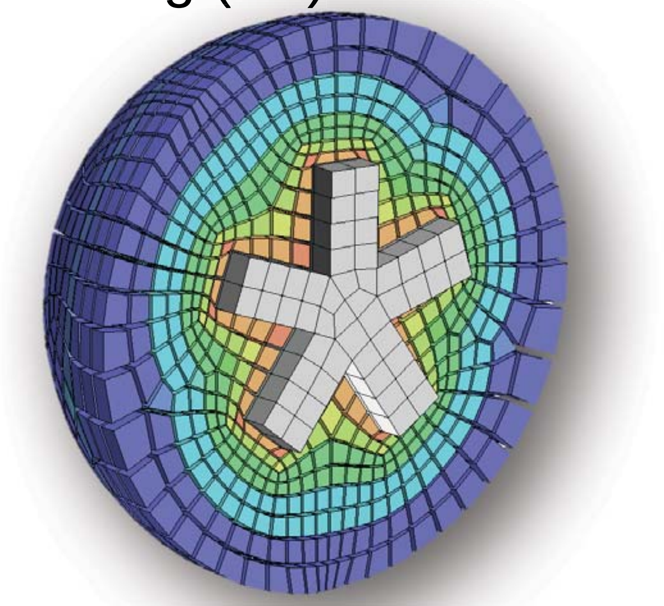




5.b. Quadrilateral and hexahedral meshing

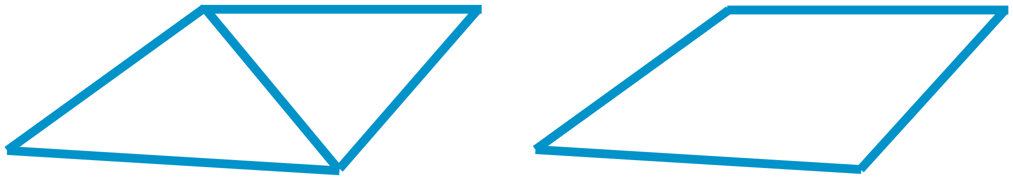
- **Classification**
  - **Indirect methods**
    - Tri/tet Combine
    - Qmorph, Hmorph
    - Blossom
  - **Direct methods**
    - Grid based methods: quadtrees (2D), octrees (3D)
    - Medial axis
    - Advancing front methods: Paving (2D), Plastering (3D)
    - Partition methods: Gen4U
    - Cross field methods
  - **Dual methods**
    - Whisker Weaving
    - Sheet manipulation





5.b. Quadrilateral and hexahedral meshing

- **Indirect methods**
  - All methods work well for 2D problems
  - Do not guarantee a full unstructured hex mesh in 3D
- **Tri/Tet combine**
  - Two triangles can be combined to generate a quad

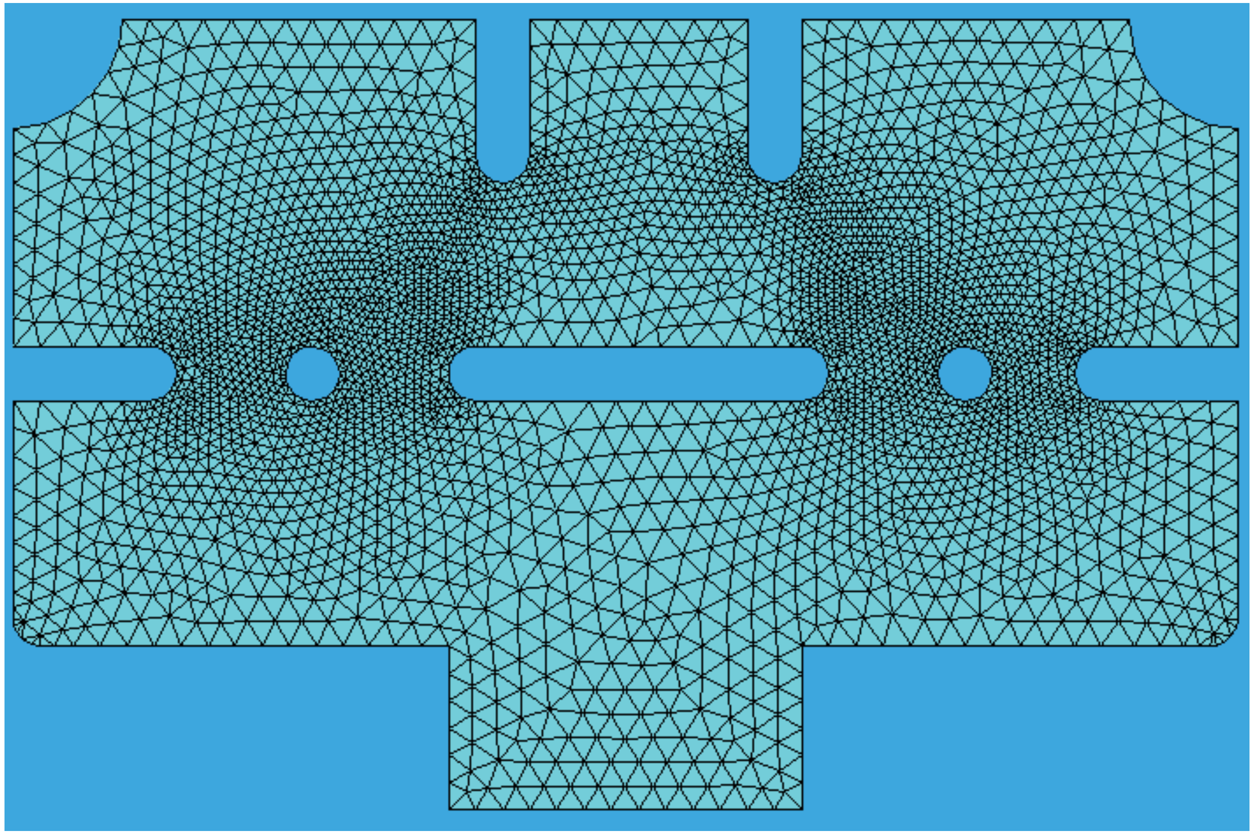


  - A triangle can be subdivided in three quads

  - The algorithm starts at a given boundary.
  - The principal operation is merge adjacent triangle. The algorithm select the “best choice”.
  - However, triangle splitting can also occur.
  - The algorithm delivers an all-quad mesh

5.b. Quadrilateral and hexahedral meshing

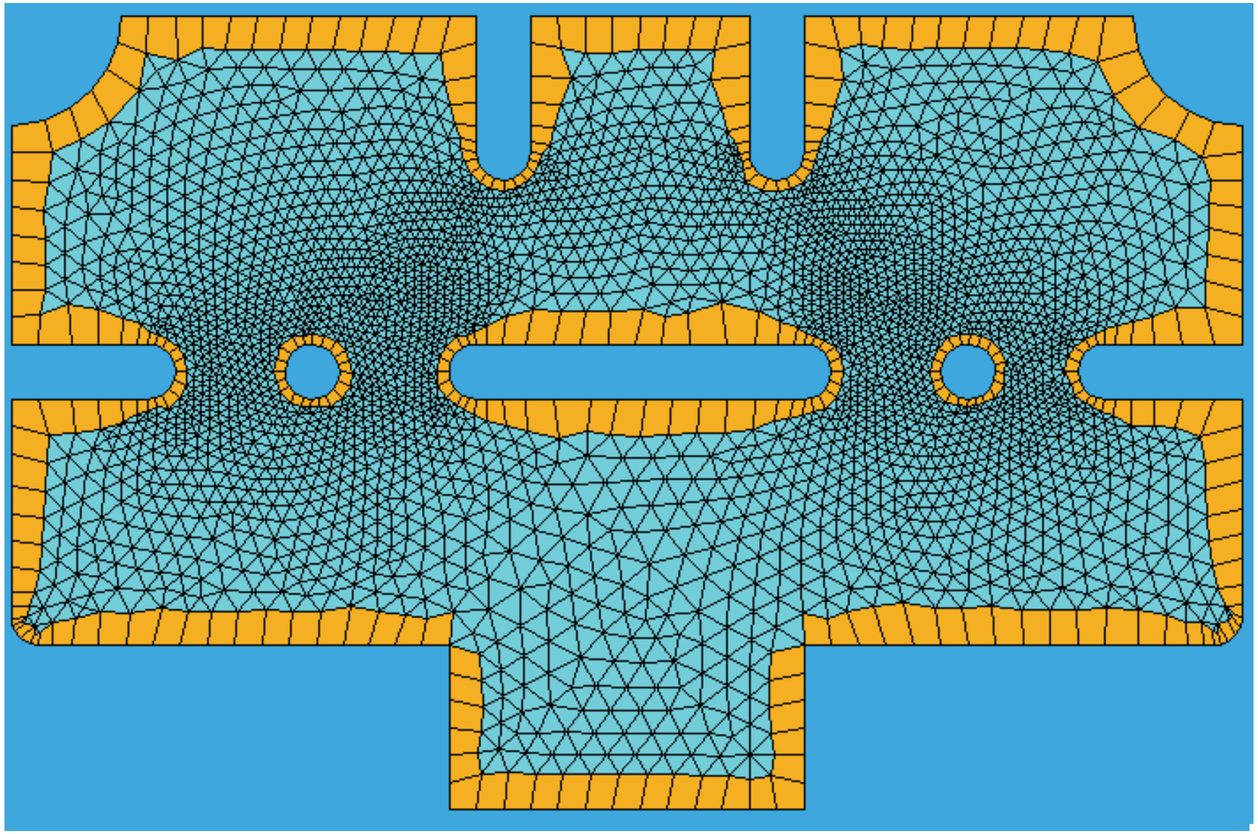
- **Qmorph / Hmorph**
  - Uses an advancing front approach
  - Local swaps applied to improve resulting quad
  - Any number of triangles merged to create a quad
  - Hex dominant meshes in 3D



Images by Steve Owen

5.b. Quadrilateral and hexahedral meshing

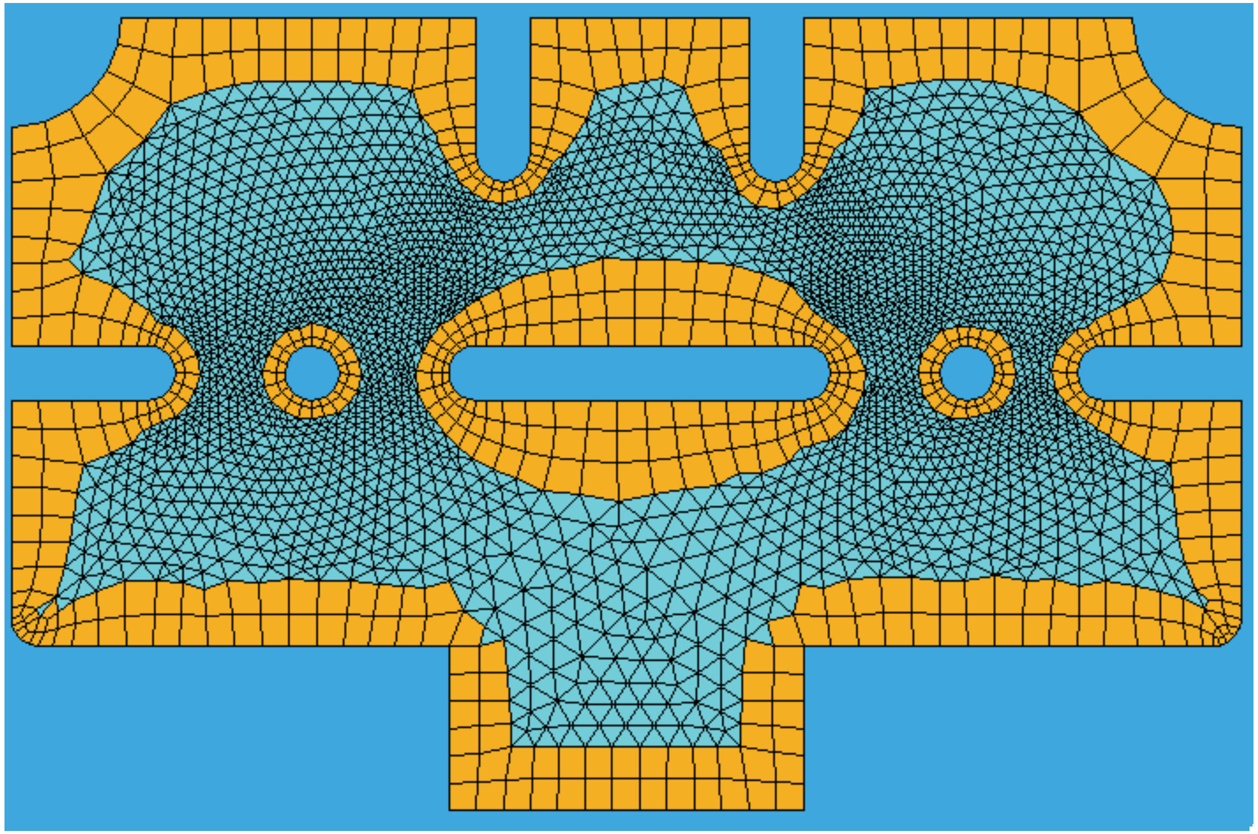
- **Qmorph / Hmorph**
  - Uses an advancing front approach
  - Local swaps applied to improve resulting quad
  - Any number of triangles merged to create a quad
  - Hex dominant meshes in 3D



Images by Steve Owen

5.b. Quadrilateral and hexahedral meshing

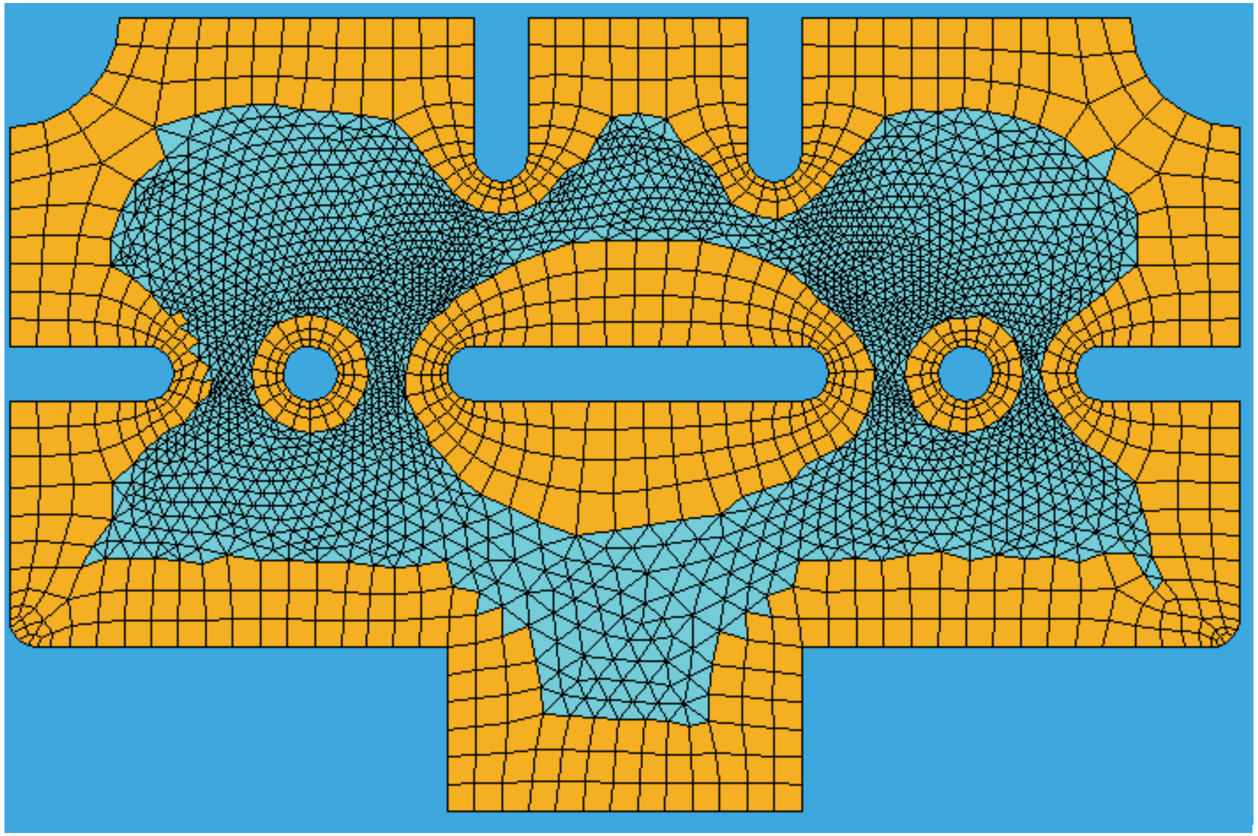
- **Qmorph / Hmorph**
  - Uses an advancing front approach
  - Local swaps applied to improve resulting quad
  - Any number of triangles merged to create a quad
  - Hex dominant meshes in 3D



Images by Steve Owen

5.b. Quadrilateral and hexahedral meshing

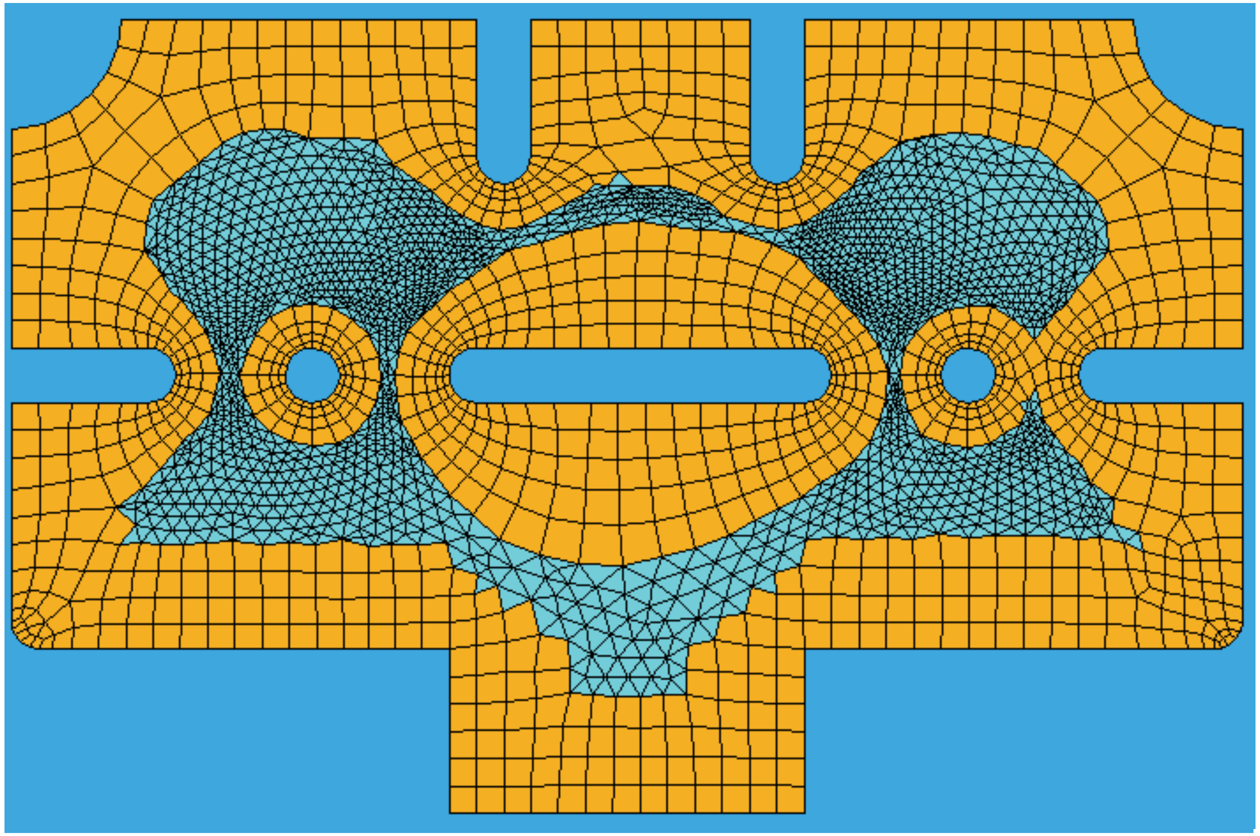
- **Qmorph / Hmorph**
  - Uses an advancing front approach
  - Local swaps applied to improve resulting quad
  - Any number of triangles merged to create a quad
  - Hex dominant meshes in 3D



Images by Steve Owen

5.b. Quadrilateral and hexahedral meshing

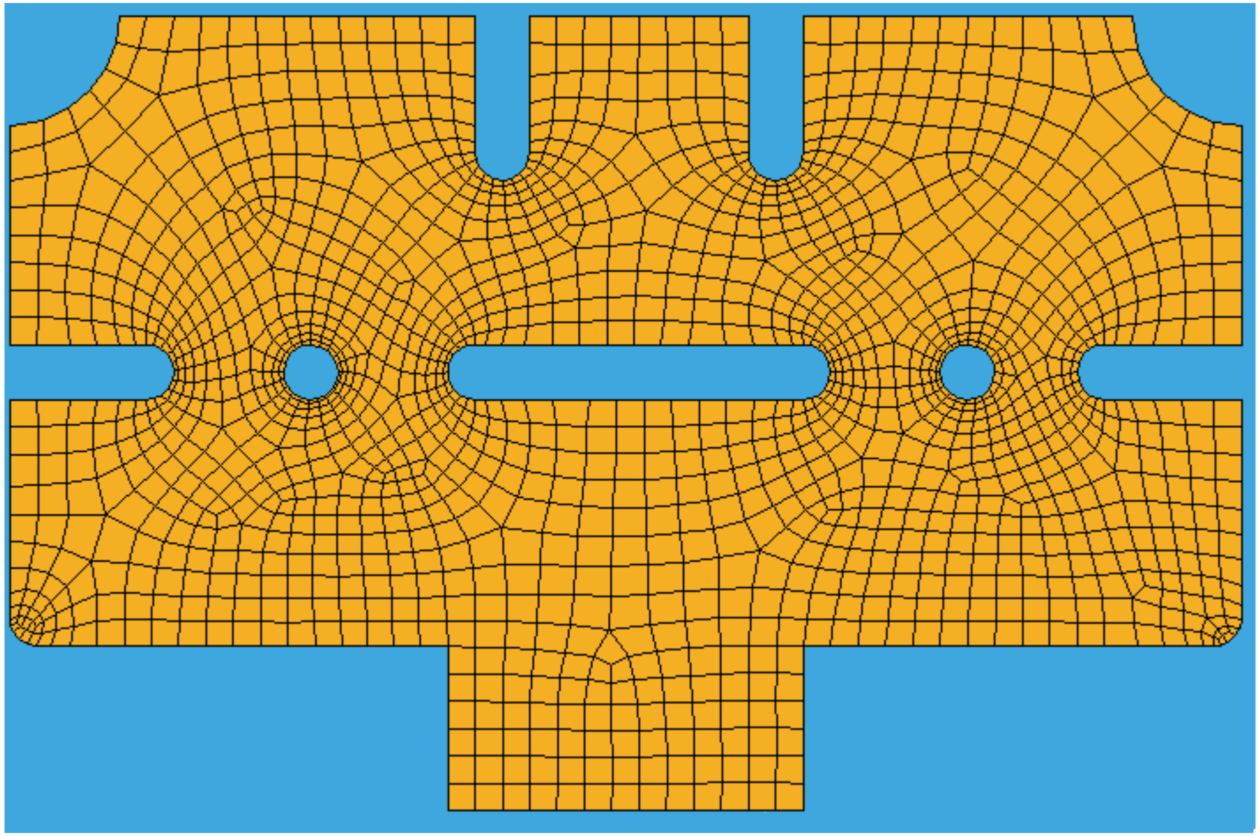
- **Qmorph / Hmorph**
  - Uses an advancing front approach
  - Local swaps applied to improve resulting quad
  - Any number of triangles merged to create a quad
  - Hex dominant meshes in 3D



Images by Steve Owen

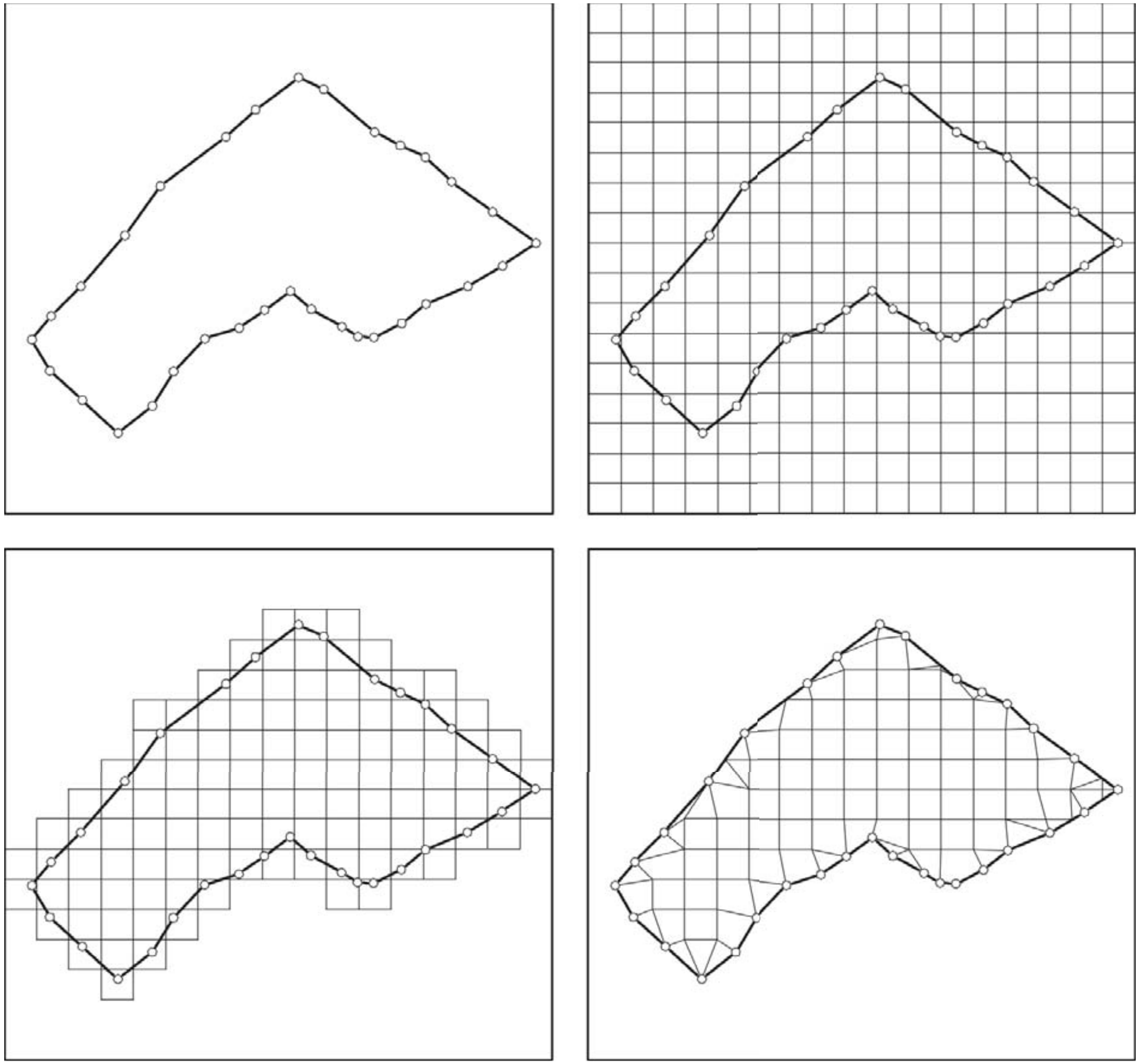
5.b. Quadrilateral and hexahedral meshing

- **Qmorph / Hmorph**
  - Uses an advancing front approach
  - Local swaps applied to improve resulting quad
  - Any number of triangles merged to create a quad
  - Hex dominant meshes in 3D



5.b. Quadrilateral and hexahedral meshing

- **Grid based methods**



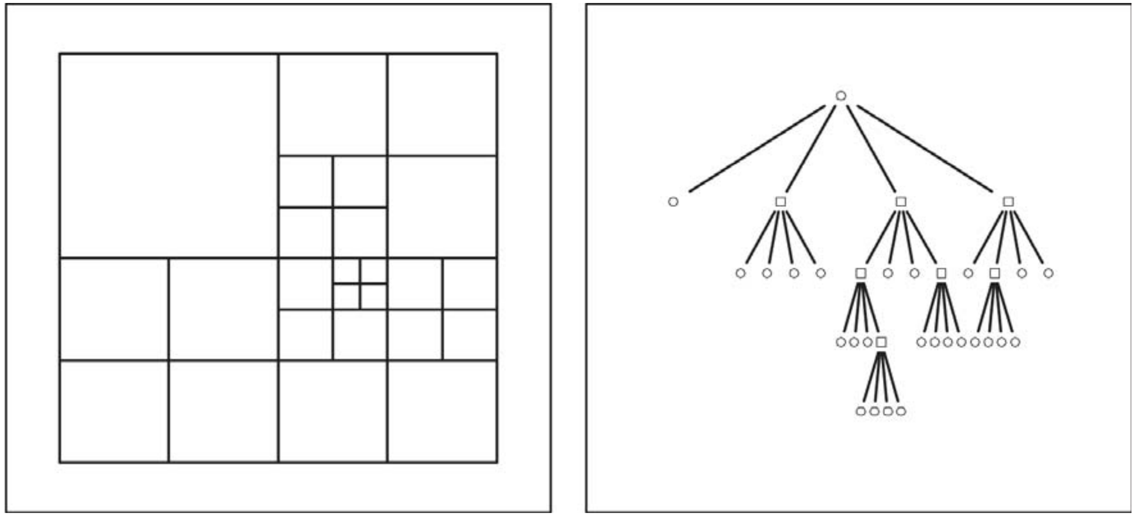
1. Generate regular grid of quads/hexes on the interior of model
2. Mark inner elements that do not touch the boundary.
3. Remove elements outside the domain
4. Fit elements to the boundary by projecting interior faces towards the surfaces

Note that:

- Lower quality elements near boundary
- Non-boundary conforming
- **Extended to 3D.** However low quality elements may appear at the boundary (on going research)

5.b. Quadrilateral and hexahedral meshing

- **Grid based methods**
  - Graded meshes (mesh refinement) is obtained using quadtree (2D) or octrees (3D)



  - Also used to generate tetrahedral meshes
  - Specific topological operators and smoothing techniques are typically used to conform curved boundaries

5.b. Quadrilateral and hexahedral meshing

- **Medial axis**
  - The Medial Axis (or skeleton) of a 2D region is defined as the locus of the center of all the maximal inscribed circle of the object.

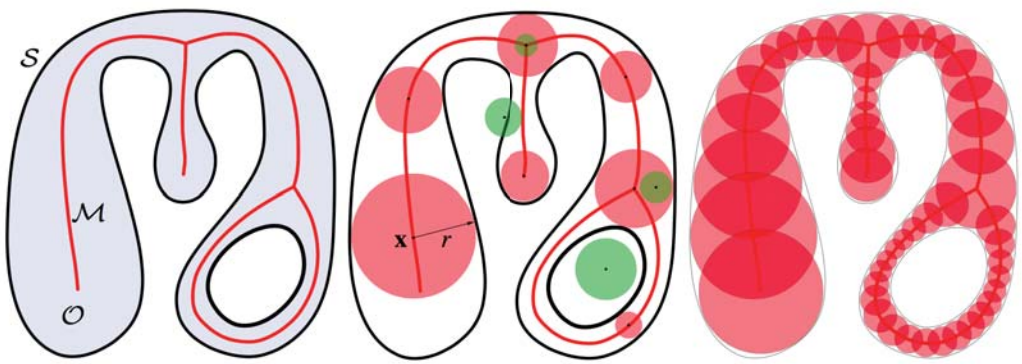


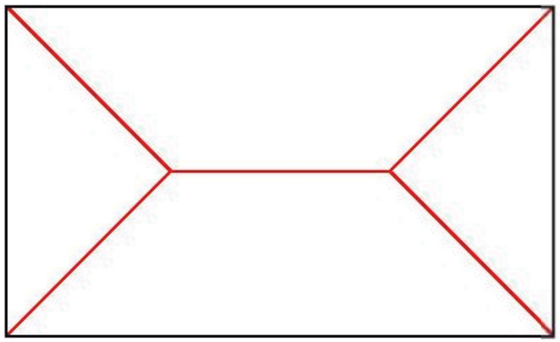
Image from A. Tagliasacchi et al.

  - Extension to 3D: medial surface or surface skeleton of a 3D object is the locus of centers of maximally inscribed balls
  - Can be understood as a n-1 representation of an n-dimensional object
  - Extensively used in many disciplines:

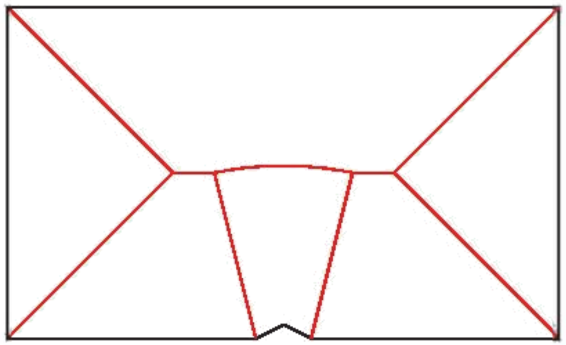
    computer graphics, medical imaging, computer aided design, visualization, digital inspection, metrology, robotics, ...

5.b. Quadrilateral and hexahedral meshing

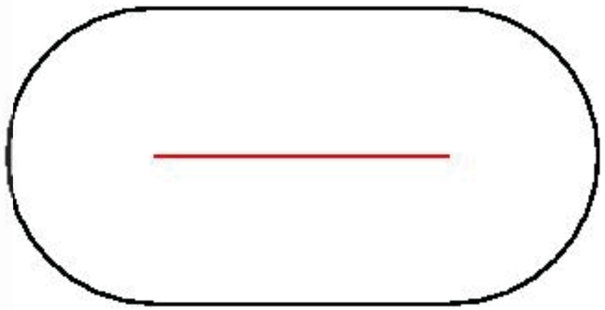
- **Mesh generation:** used to obtain a bloc decomposition of the domain into simple subregions suitable for meshing with hexahedral elements



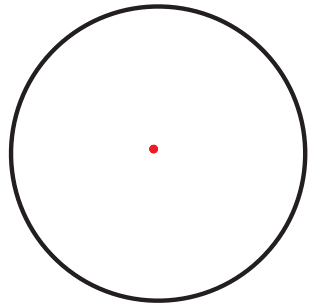
Each part can be meshed with a compatible quad mesh



Medial axis is sensitive to small boundary perturbations



Degeneration to a line

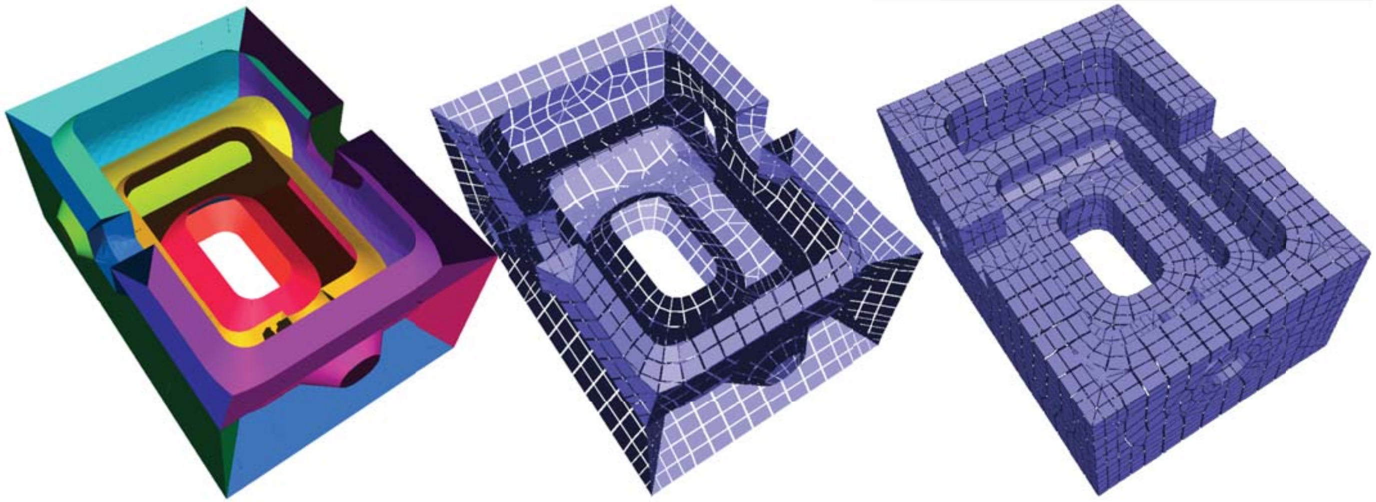


Degeneration to a point

- Advanced algorithms to compute an approximation of the the medial surface

5.b. Quadrilateral and hexahedral meshing

- **Some examples**



Images from P. Sampl

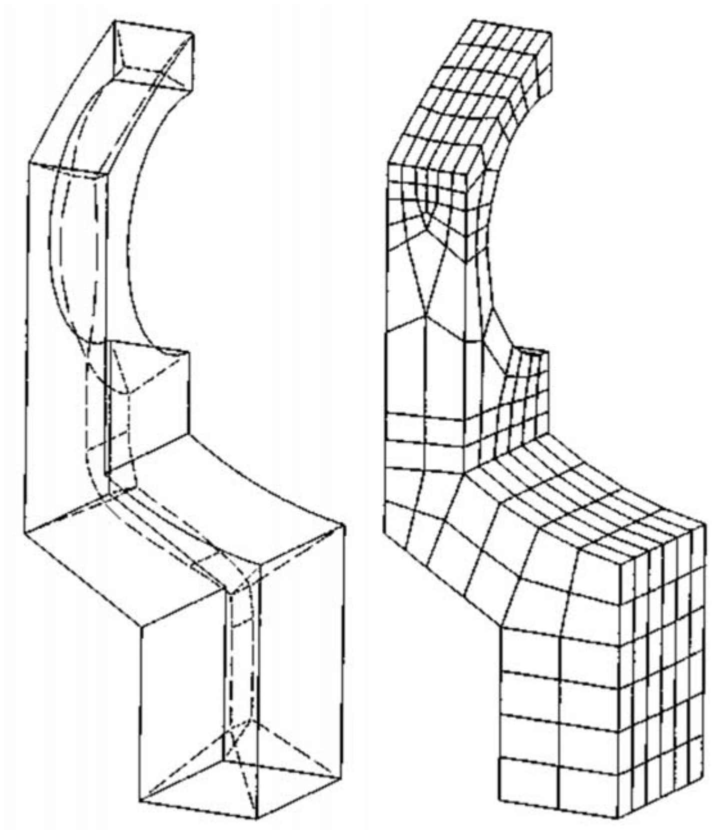
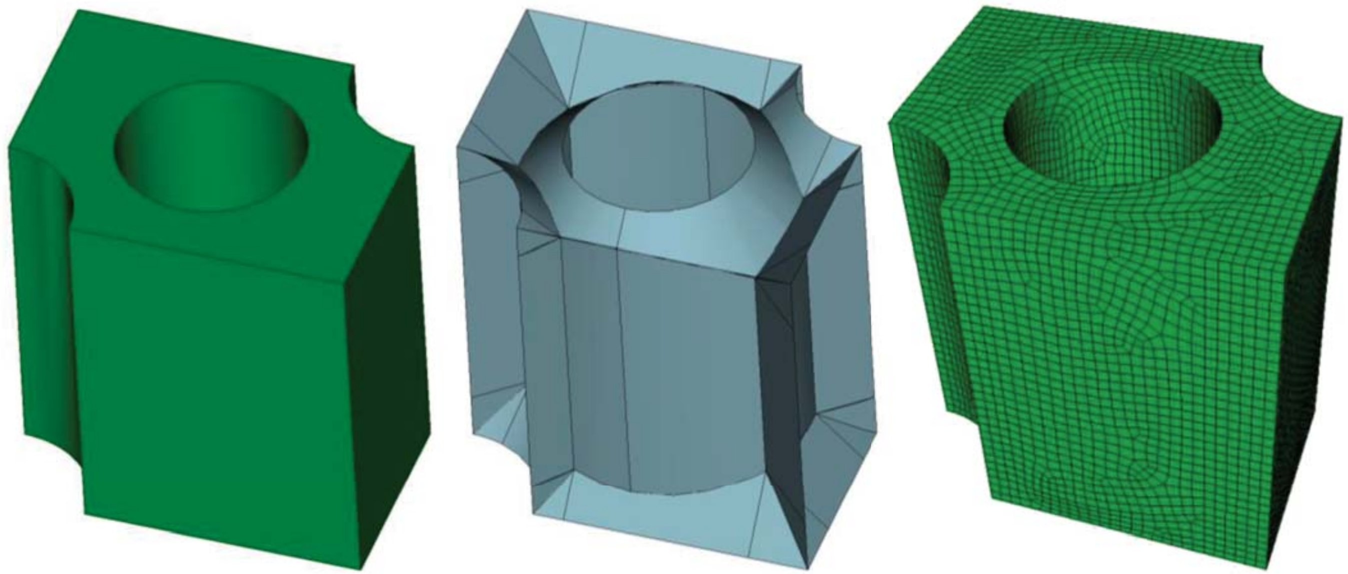


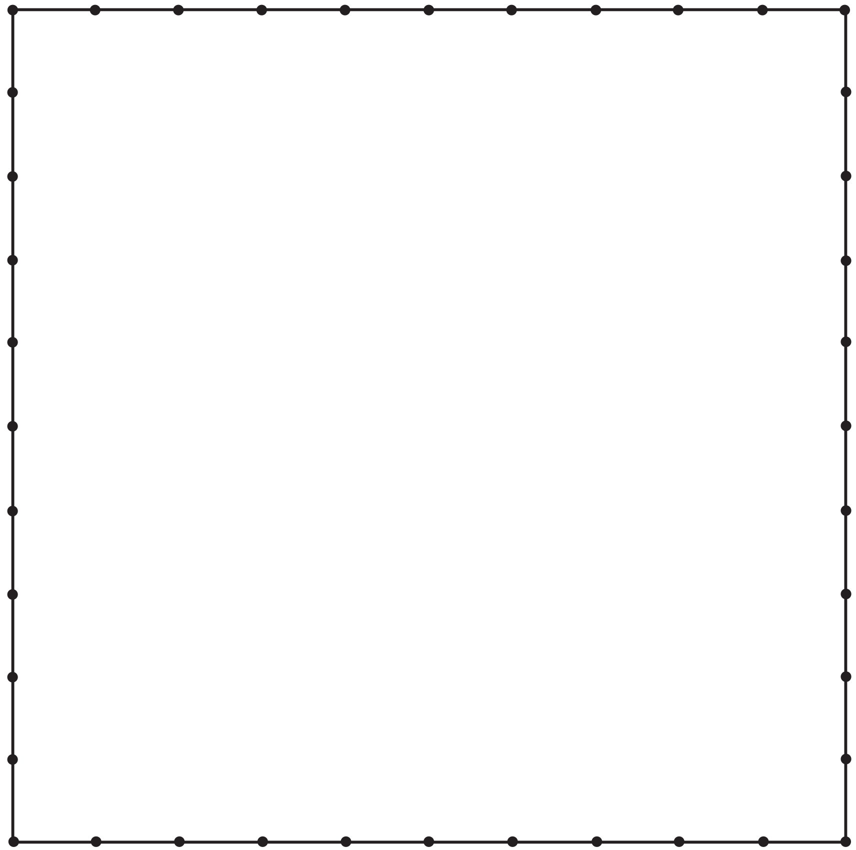
Image from M.A. Price & C.G. Armstrong



Images from W.R.Quadros

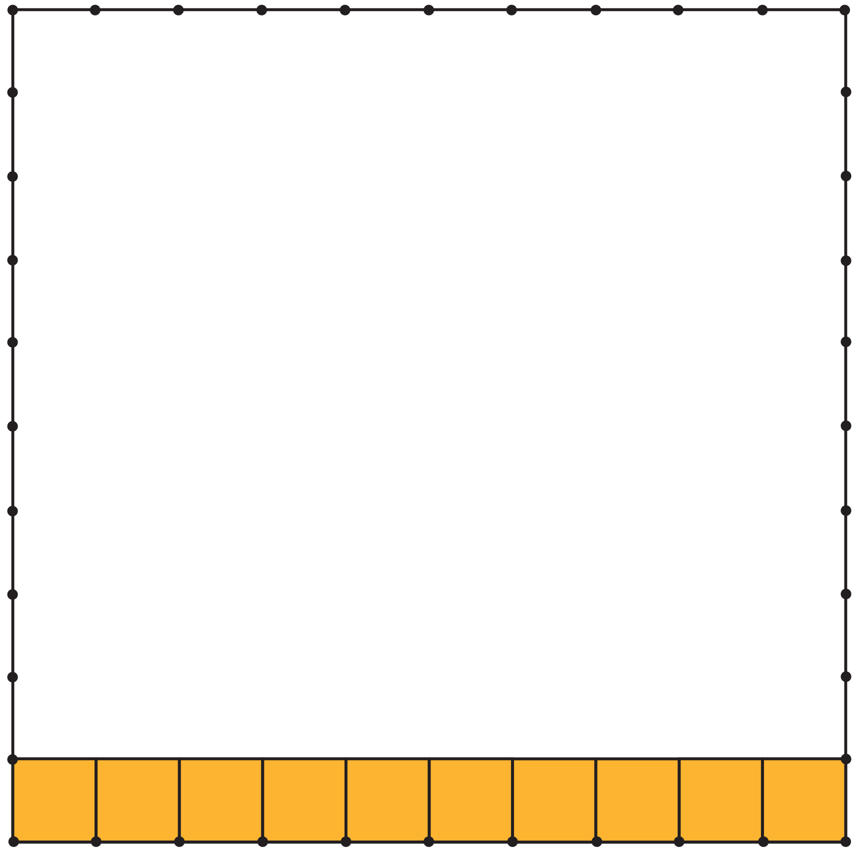
5.b. Quadrilateral and hexahedral meshing

- **Paving method (2D) / PLASTERING (3D)**
  - Pre-meshed boundary
  - Layers advance inward from the boundaries

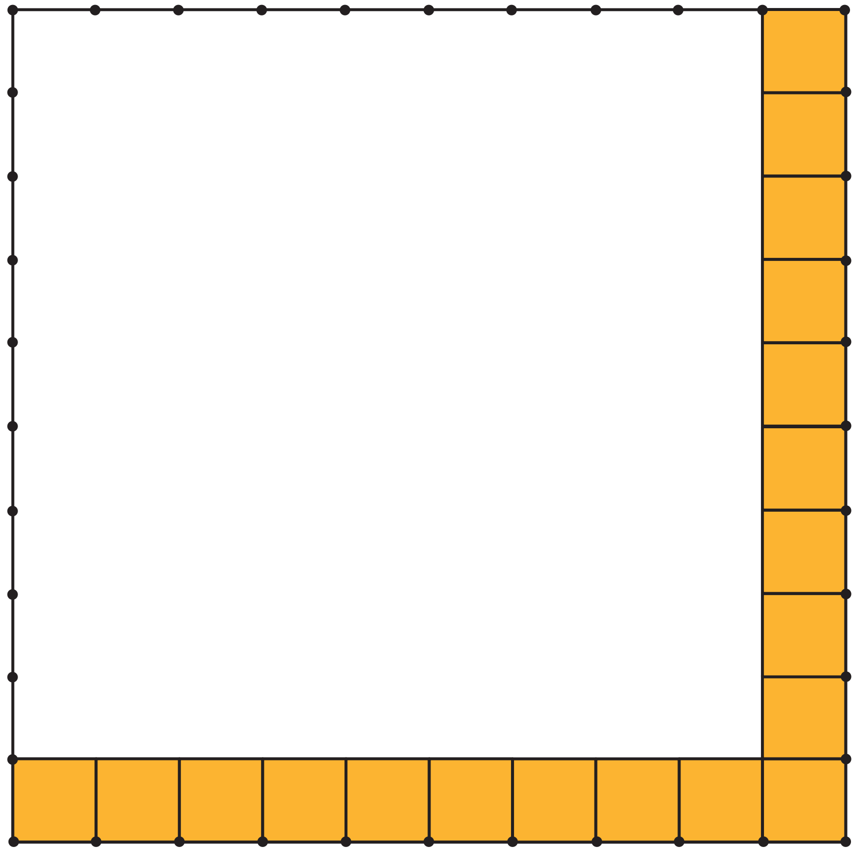


5.b. Quadrilateral and hexahedral meshing

- **Paving method (2D) / PLASTERING (3D)**
  - Pre-meshed boundary
  - Layers advance inward from the boundaries

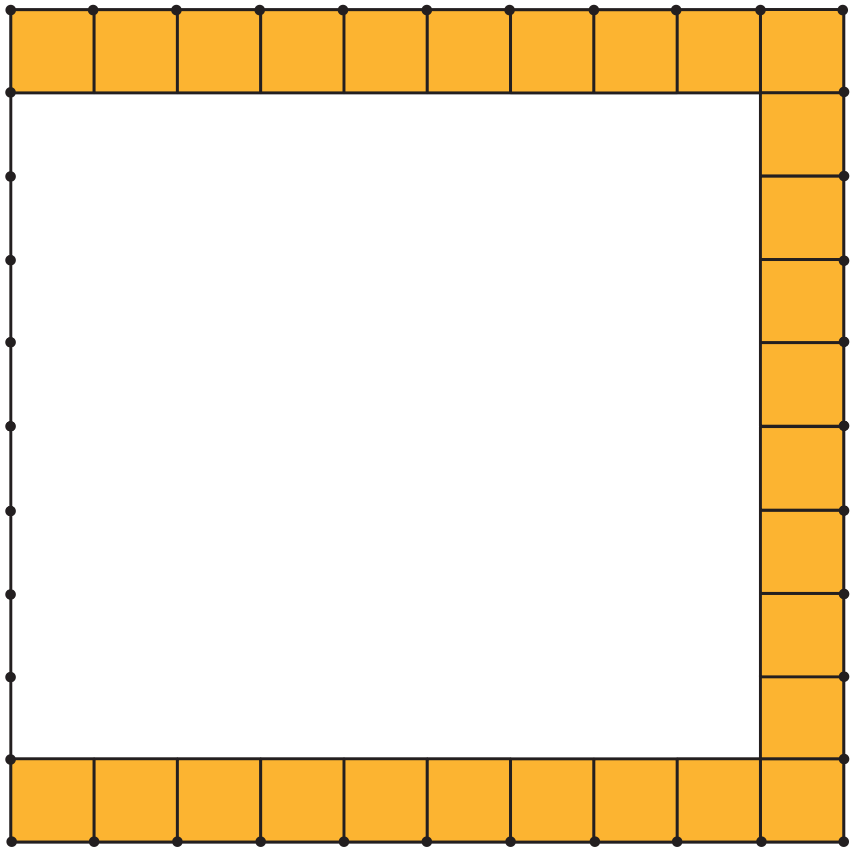


- **Paving method (2D) / PLASTERING (3D)**
  - Pre-meshed boundary
  - Layers advance inward from the boundaries



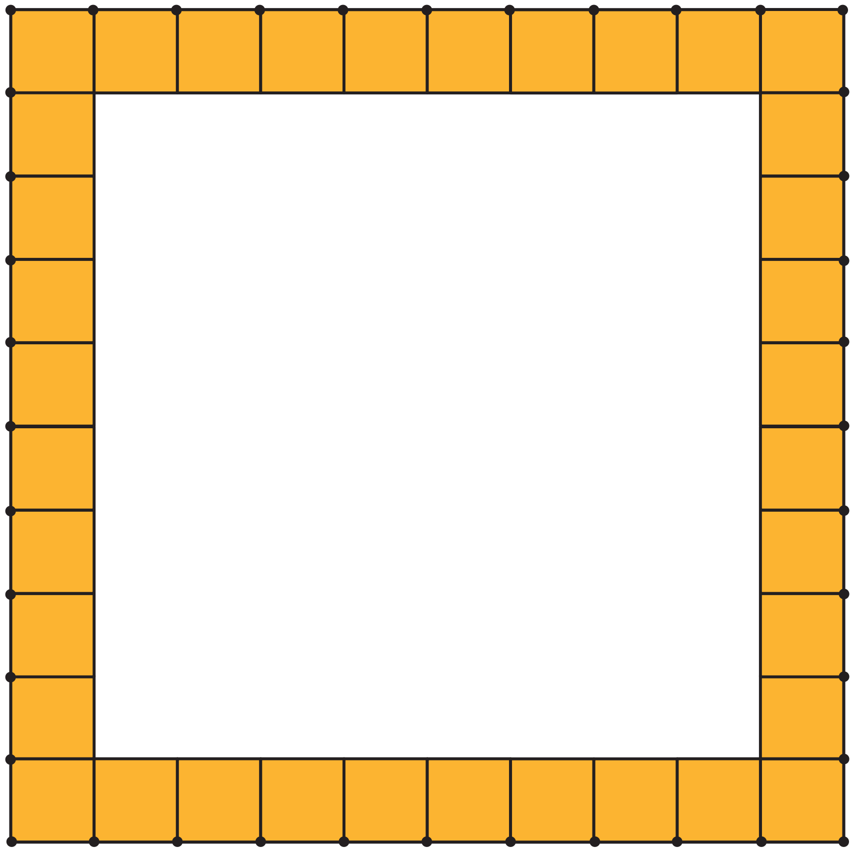
5.b. Quadrilateral and hexahedral meshing

- **Paving method (2D) / PLASTERING (3D)**
  - Pre-meshed boundary
  - Layers advance inward from the boundaries



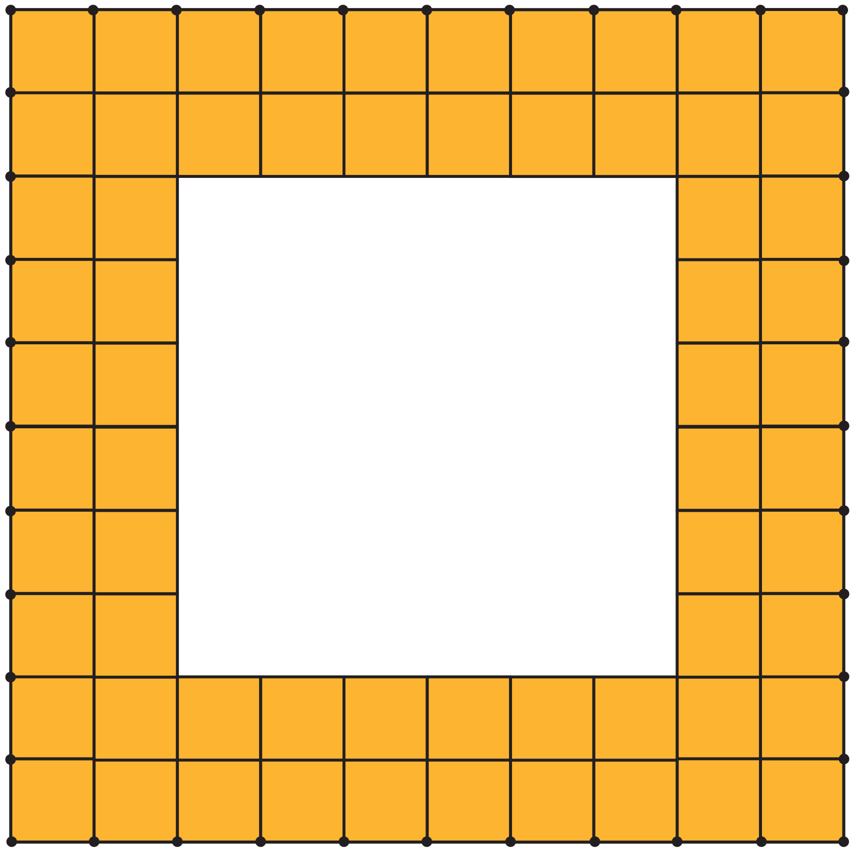
5.b. Quadrilateral and hexahedral meshing

- **Paving method (2D) / PLASTERING (3D)**
  - Pre-meshed boundary
  - Layers advance inward from the boundaries



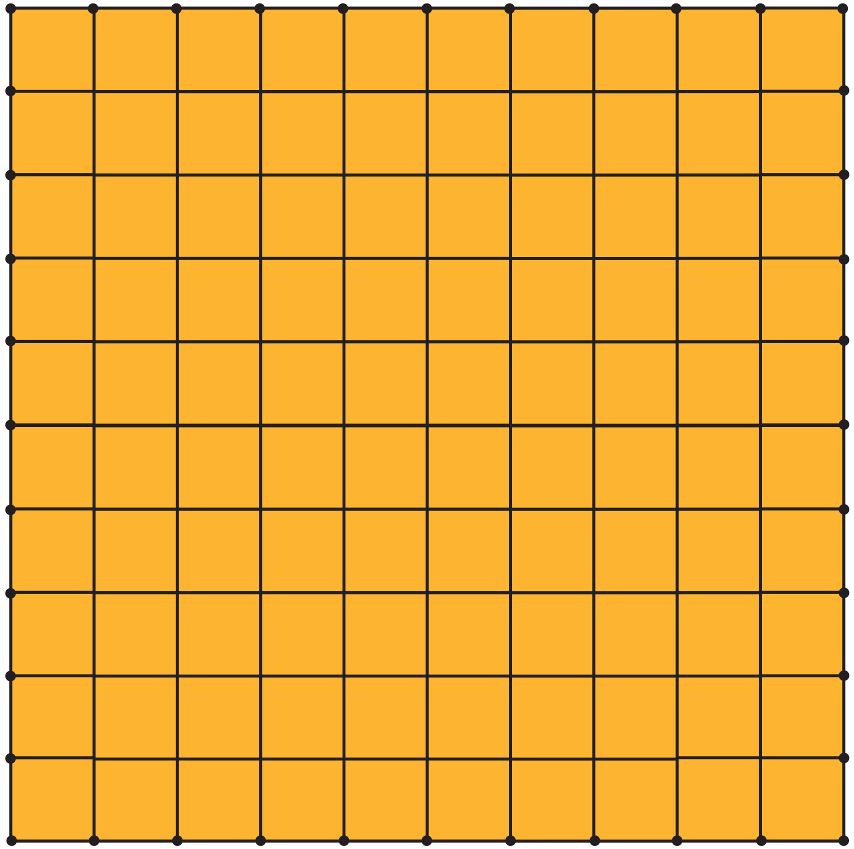
5.b. Quadrilateral and hexahedral meshing

- **Paving method (2D) / PLASTERING (3D)**
  - Pre-meshed boundary
  - Layers advance inward from the boundaries



5.b. Quadrilateral and hexahedral meshing

- **Paving method (2D) / PLASTERING (3D)**
  - Pre-meshed boundary
  - Layers advance inward from the boundaries



5.b. Quadrilateral and hexahedral meshing

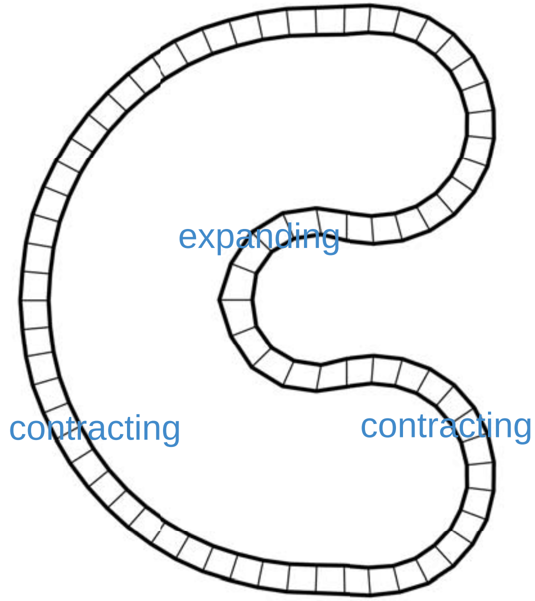
- **Paving method (2D) / PLASTERING (3D)**
  - Pre-meshed boundary
  - Layers advance inward from the boundaries
- Advantages:
  - Fully automatic
  - High quality meshes near the boundary
  - Boundary meshes are respected (compatibility)
- Drawbacks
  - May lead to mixed meshes in 3D
  - Time consuming

5.b. Quadrilateral and hexahedral meshing

- Main operation

  - Node classification

  - Node location
    - Wedge insertion (expanding areas)
    - Tuck formation (contracting areas)
    - Seaming

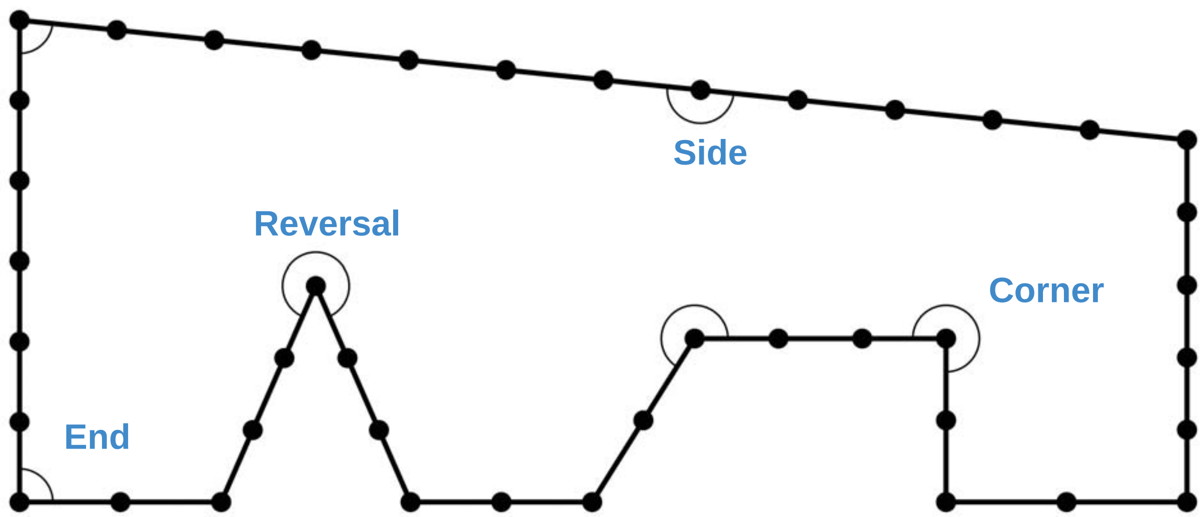


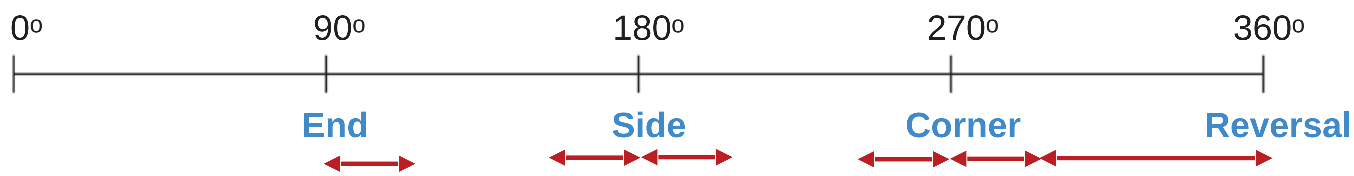
  - Small domains are closed using templates (loop closure)

  - Front collision

5.b. Quadrilateral and hexahedral meshing

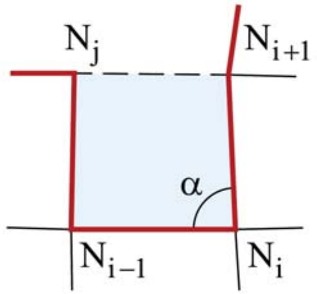
- Node classification
  - Definition of node types from the internal angles



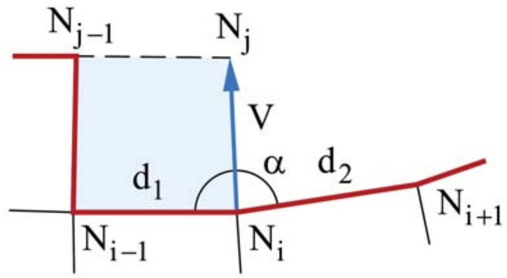


5.b. Quadrilateral and hexahedral meshing

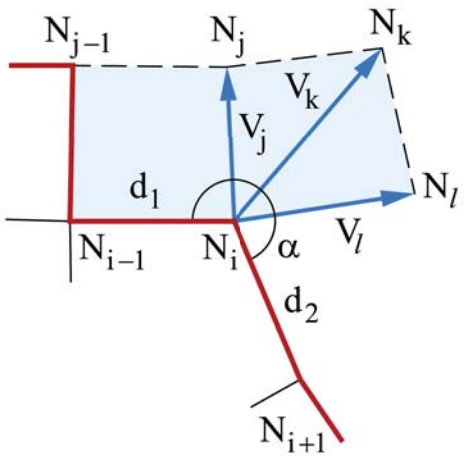
- Node location



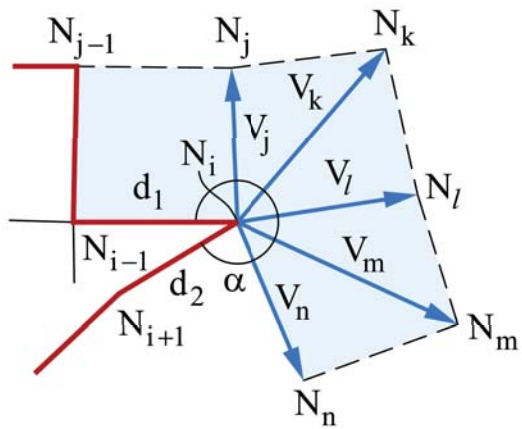
Element generation
at a row **end** node



Node and element generation
at a row **side** node

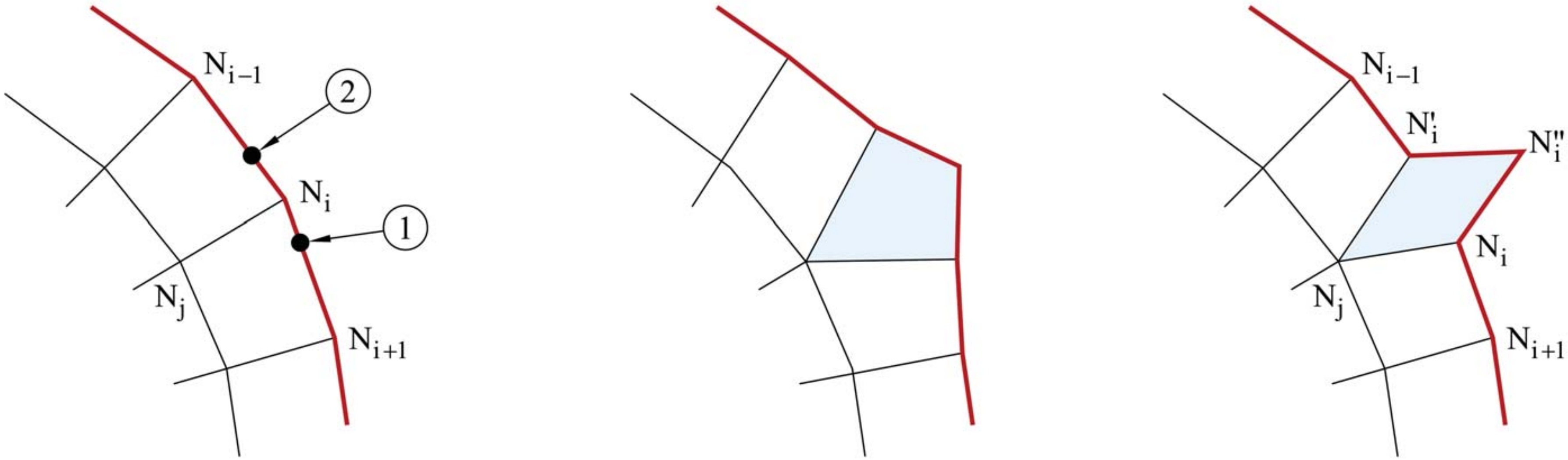


Node and element generation
at a row **corner** node



Node and element generation
at a row **reversal** node

- Node location: wedge insertion

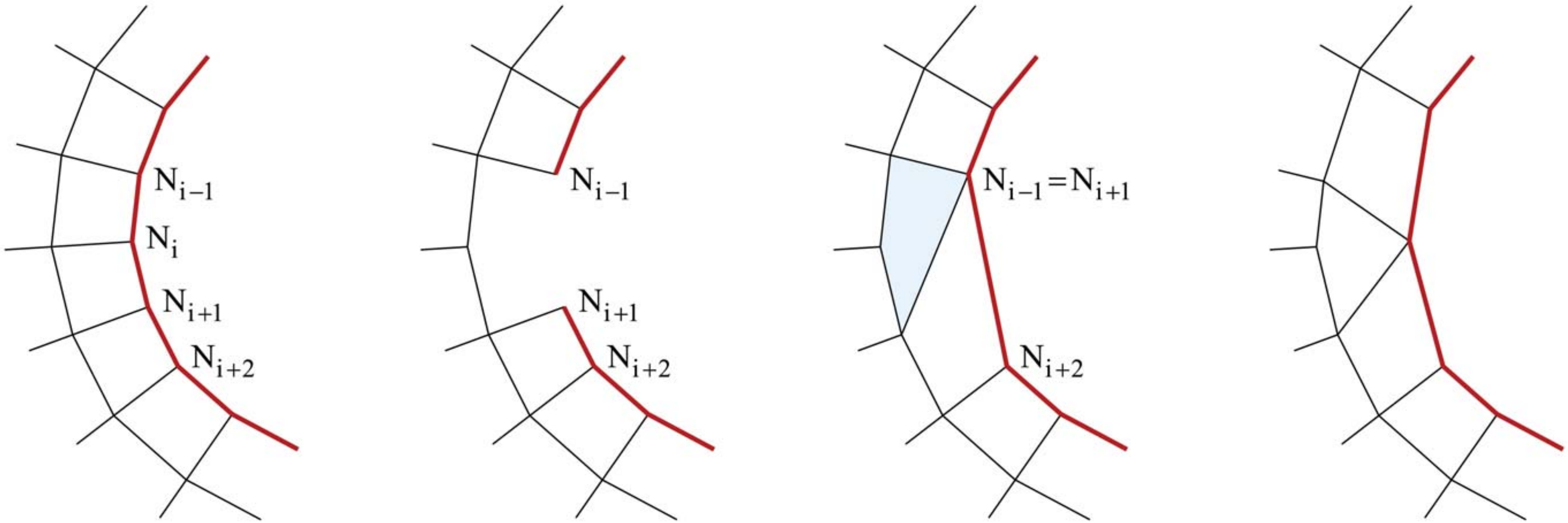


Node N_i is moved to position 1 and another node is created at position 2

A new quadrilateral element is inserted

Afterwards the front is smoothed

- Node location: tuck formation



Contraction of size of the element sides

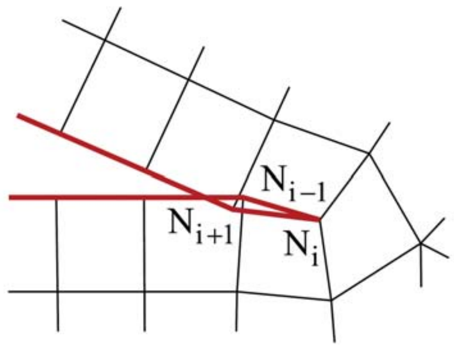
Two quadrilateral elements are removed

Node N_{i+1} is merged with node N_{i-1} and a new quadrilateral element is formed

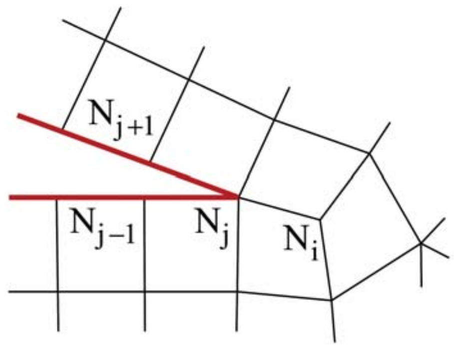
The front is smoothed after tuck formation

5.b. Quadrilateral and hexahedral meshing

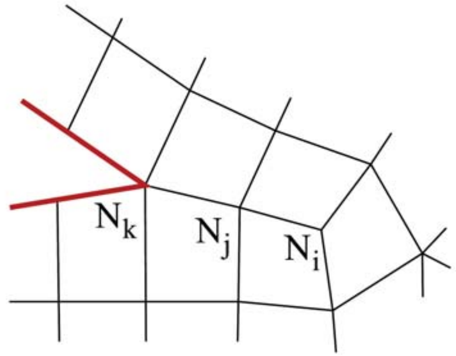
- Seaming



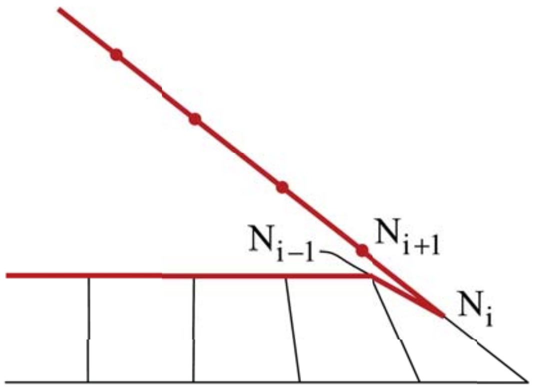
The front overlaps with itself



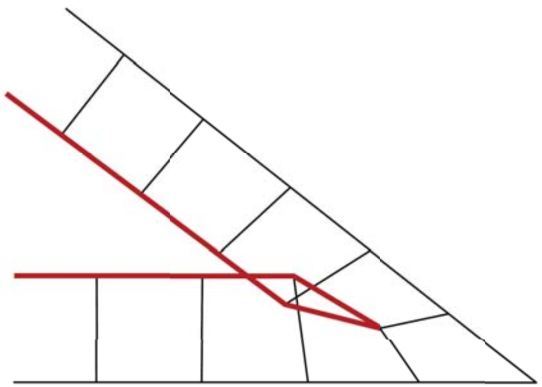
Nodes N_{i-1} and N_{i+1} are merged into N_j



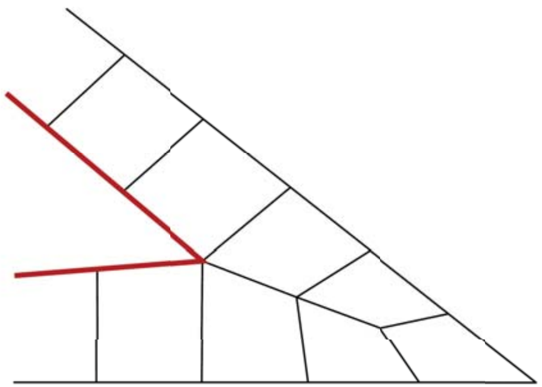
The small angle at node N_j is eliminated by merging nodes N_{j-1} and N_{j+1}



Seaming is delayed to avoid the formation of a badly shaped quadrilateral element



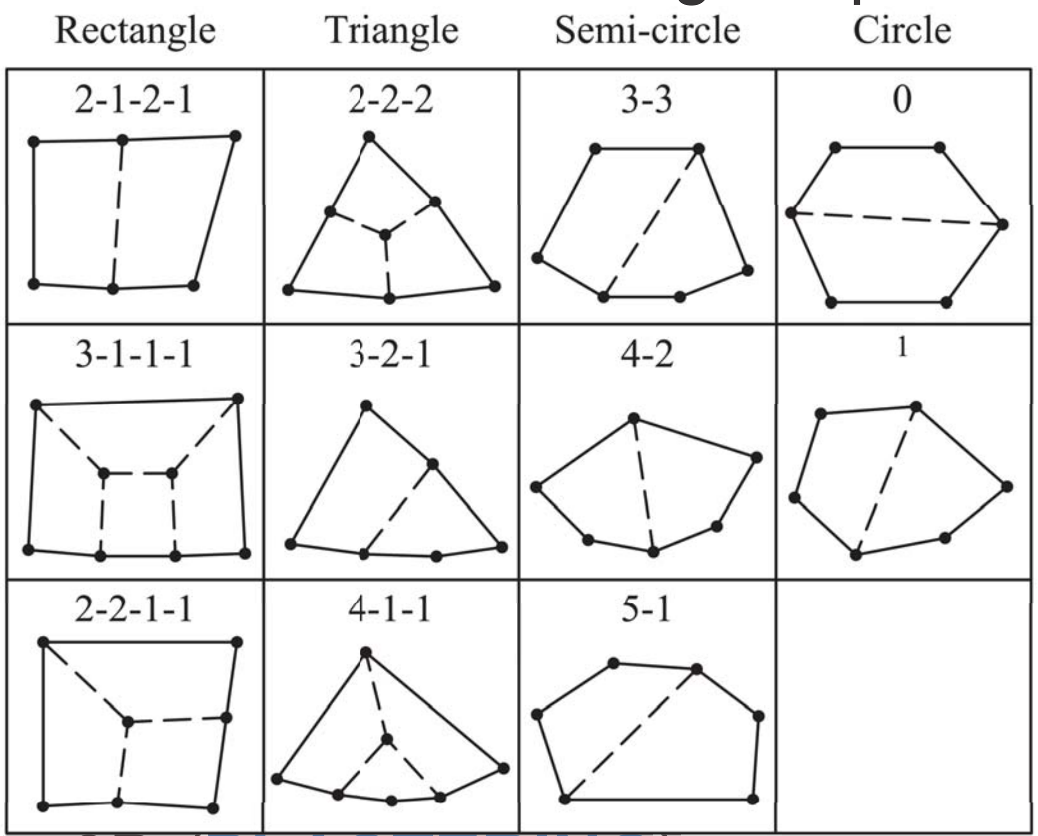
Paving front overlaps itself



Seaming is performed

5.b. Quadrilateral and hexahedral meshing

- Small domains are closed using templates (loop closure)



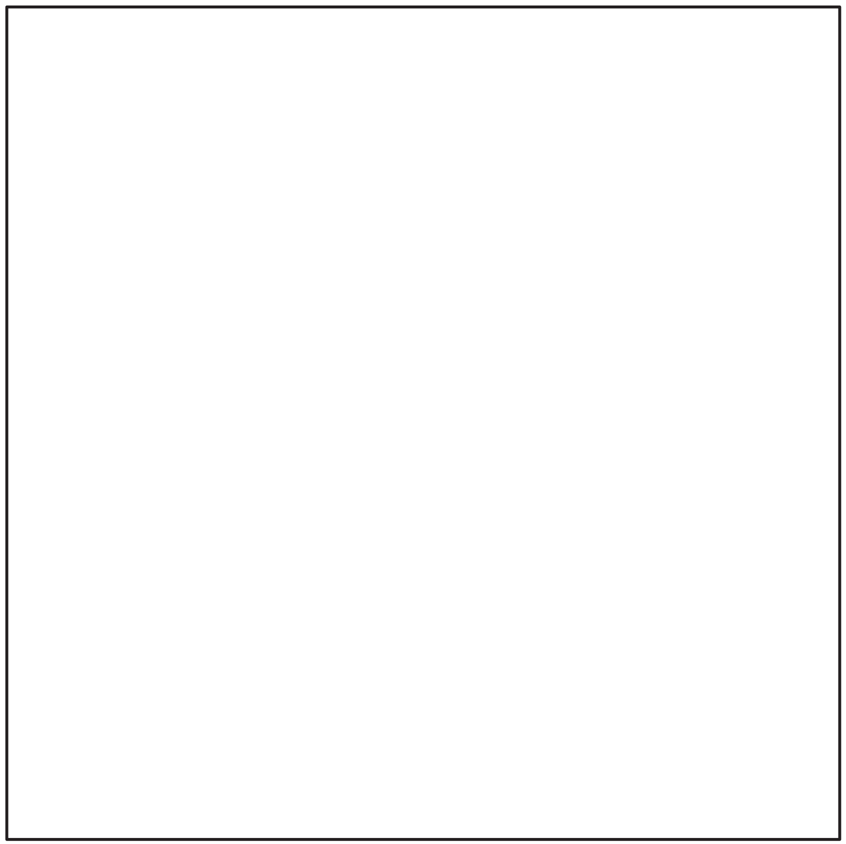
- Extends to 3D (**PLASTERING**)
  - Fails to generate a fully unstructured hex mesh for some geometries
  - Hex meshes are too much constrained !

5.b. Quadrilateral and hexahedral meshing

- **Unconstrained Plastering**
  - Unmeshed boundary
  - Layers advance inward from the boundaries

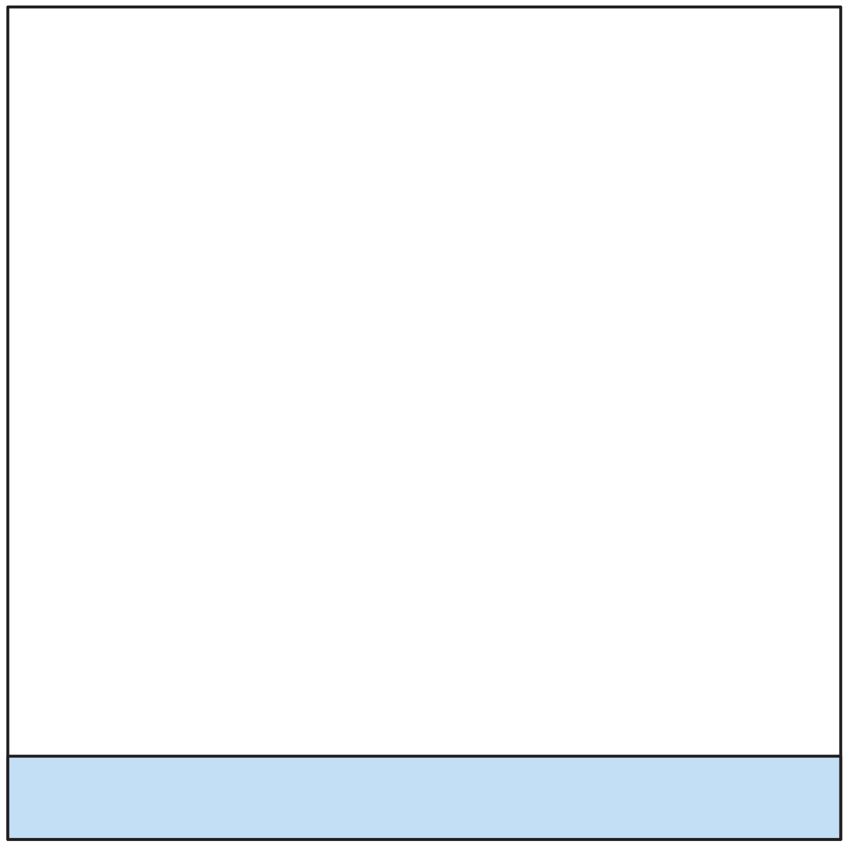
5.b. Quadrilateral and hexahedral meshing

- **Unconstrained Plastering**
  - Unmeshed boundary
  - Layers advance inward from the boundaries



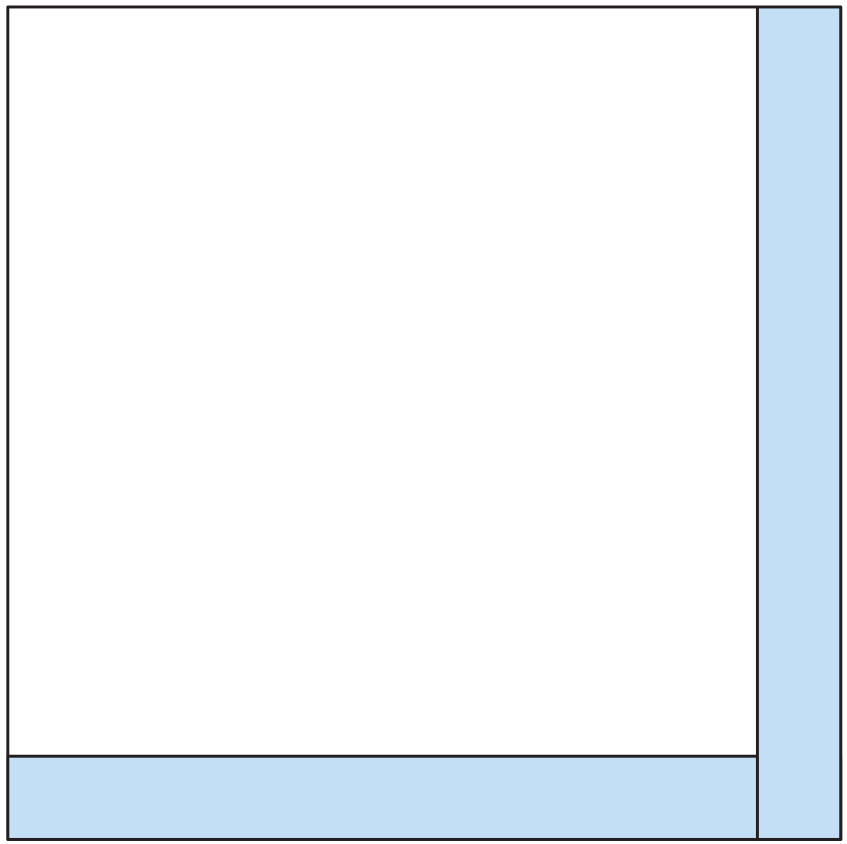
5.b. Quadrilateral and hexahedral meshing

- **Unconstrained Plastering**
  - Unmeshed boundary
  - Layers advance inward from the boundaries

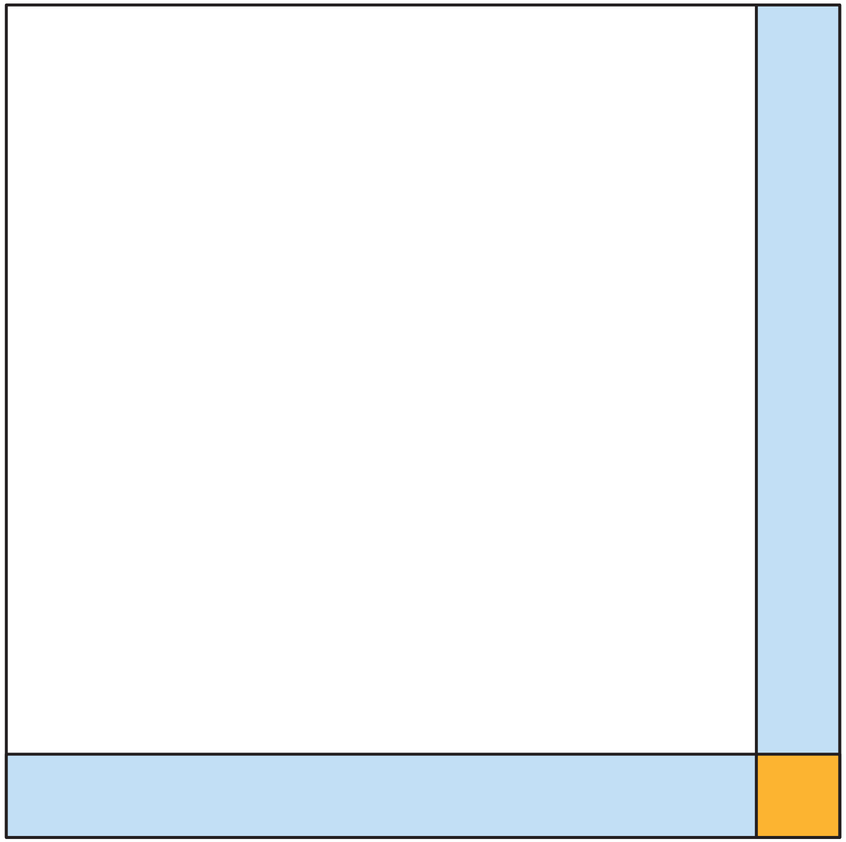


5.b. Quadrilateral and hexahedral meshing

- **Unconstrained Plastering**
  - Unmeshed boundary
  - Layers advance inward from the boundaries

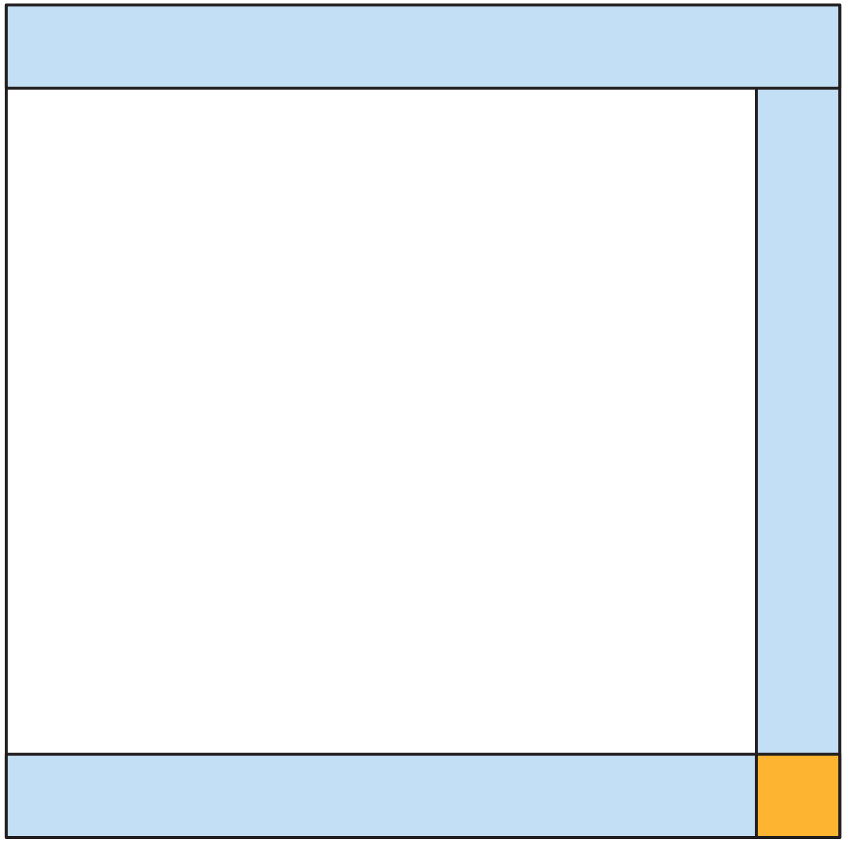


- **Unconstrained Plastering**
  - Unmeshed boundary
  - Layers advance inward from the boundaries



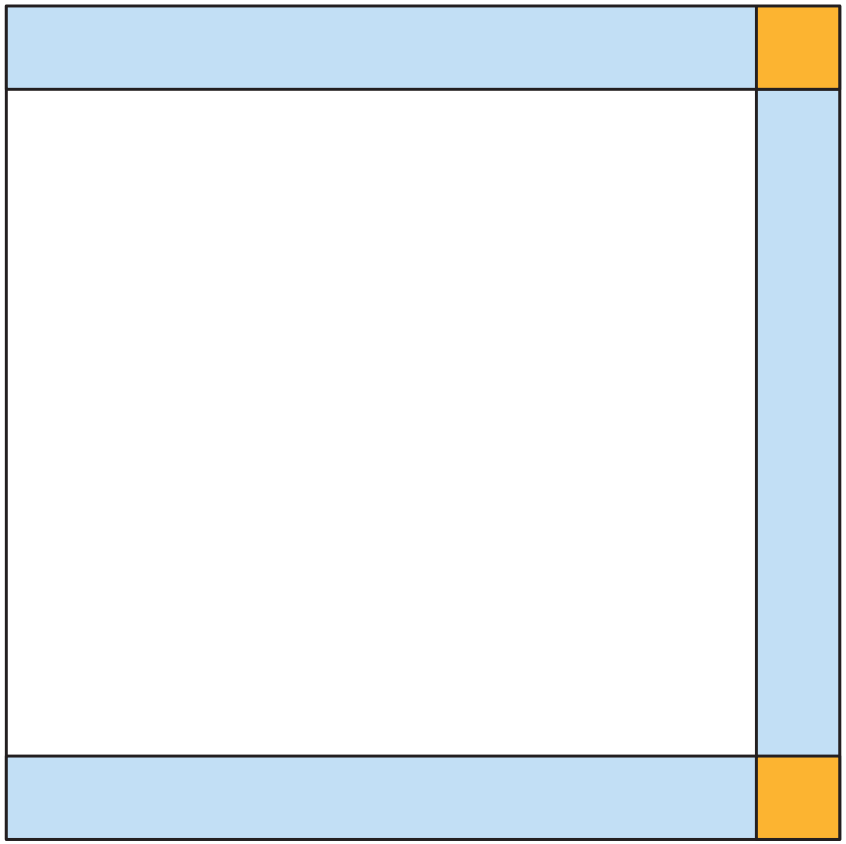
5.b. Quadrilateral and hexahedral meshing

- **Unconstrained Plastering**
  - Unmeshed boundary
  - Layers advance inward from the boundaries



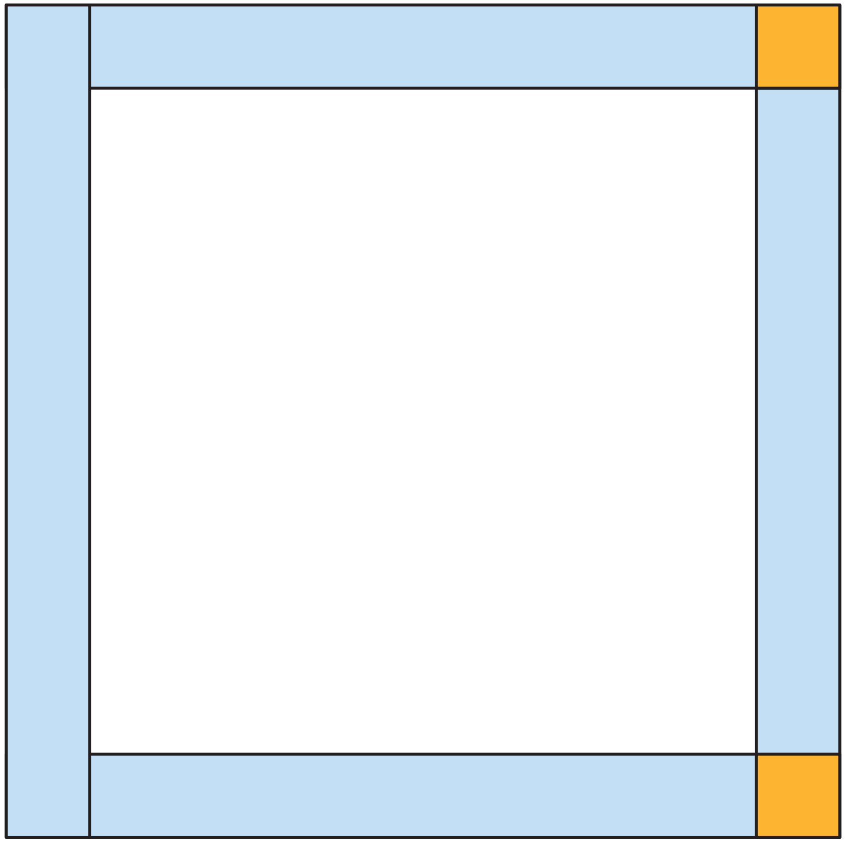
5.b. Quadrilateral and hexahedral meshing

- **Unconstrained Plastering**
  - Unmeshed boundary
  - Layers advance inward from the boundaries



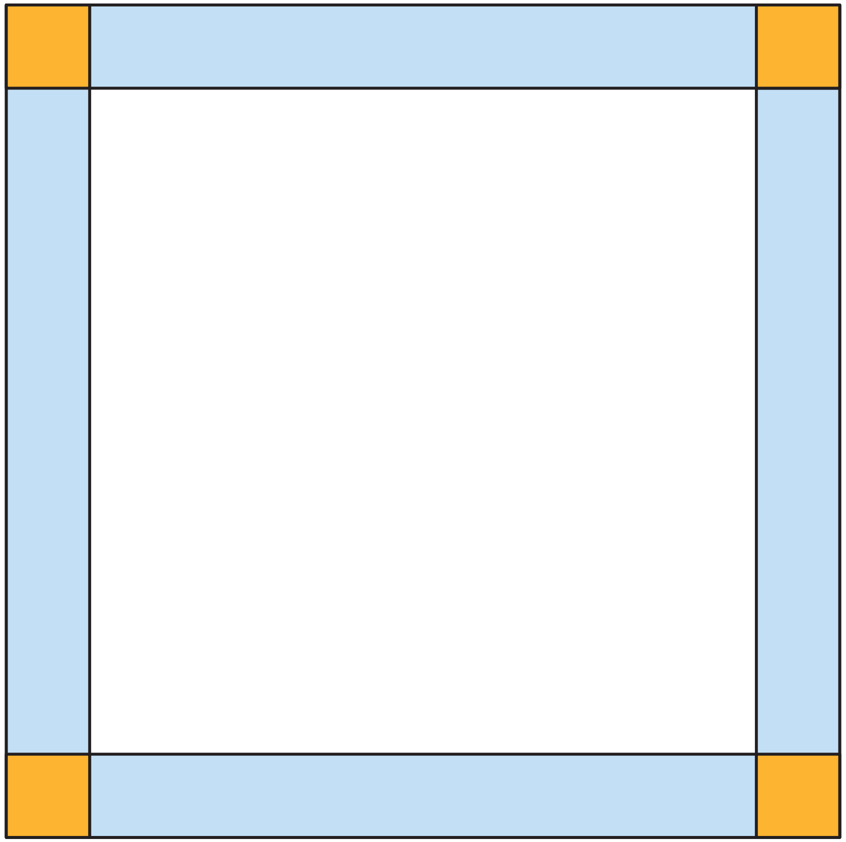
5.b. Quadrilateral and hexahedral meshing

- **Unconstrained Plastering**
  - Unmeshed boundary
  - Layers advance inward from the boundaries



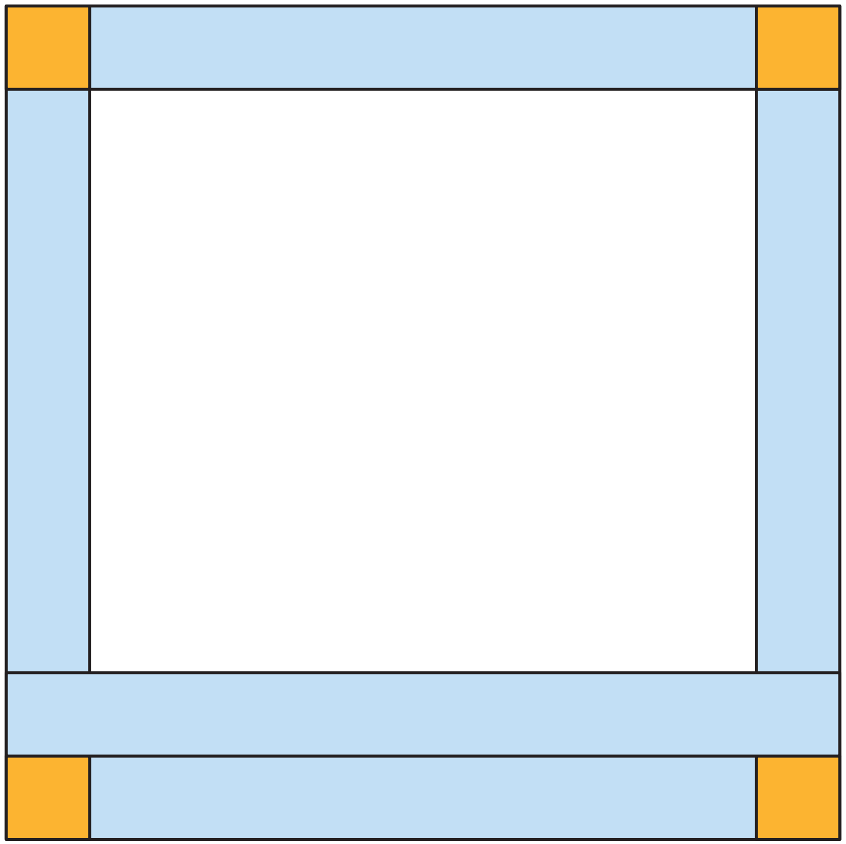
5.b. Quadrilateral and hexahedral meshing

- **Unconstrained Plastering**
  - Unmeshed boundary
  - Layers advance inward from the boundaries

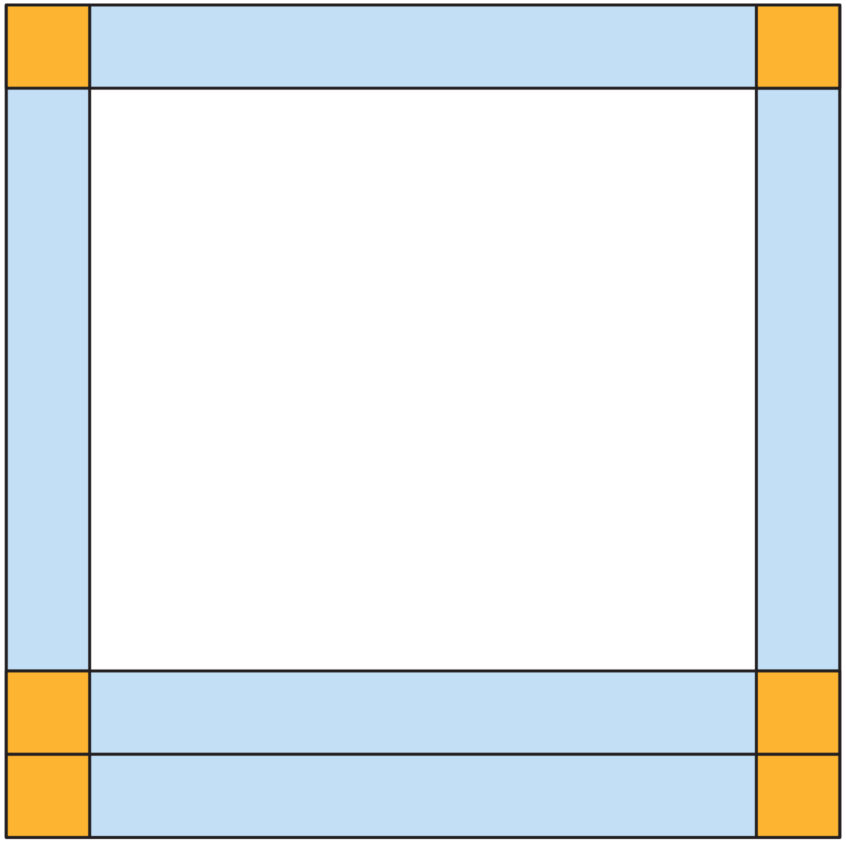


5.b. Quadrilateral and hexahedral meshing

- **Unconstrained Plastering**
  - Unmeshed boundary
  - Layers advance inward from the boundaries

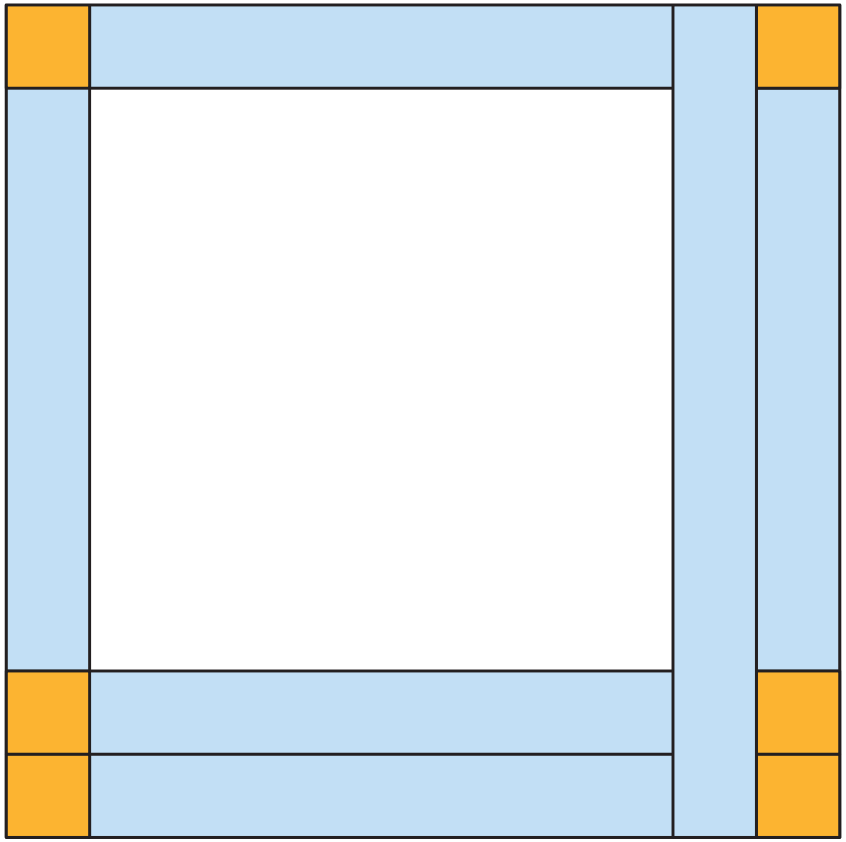


- **Unconstrained Plastering**
  - Unmeshed boundary
  - Layers advance inward from the boundaries



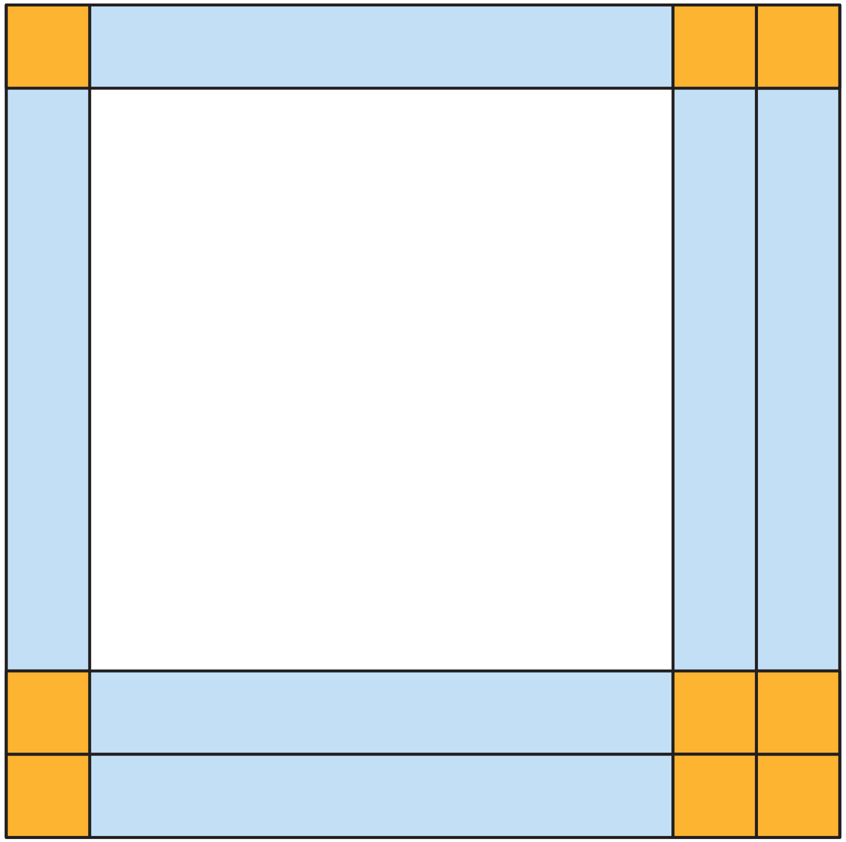
5.b. Quadrilateral and hexahedral meshing

- **Unconstrained Plastering**
  - Unmeshed boundary
  - Layers advance inward from the boundaries



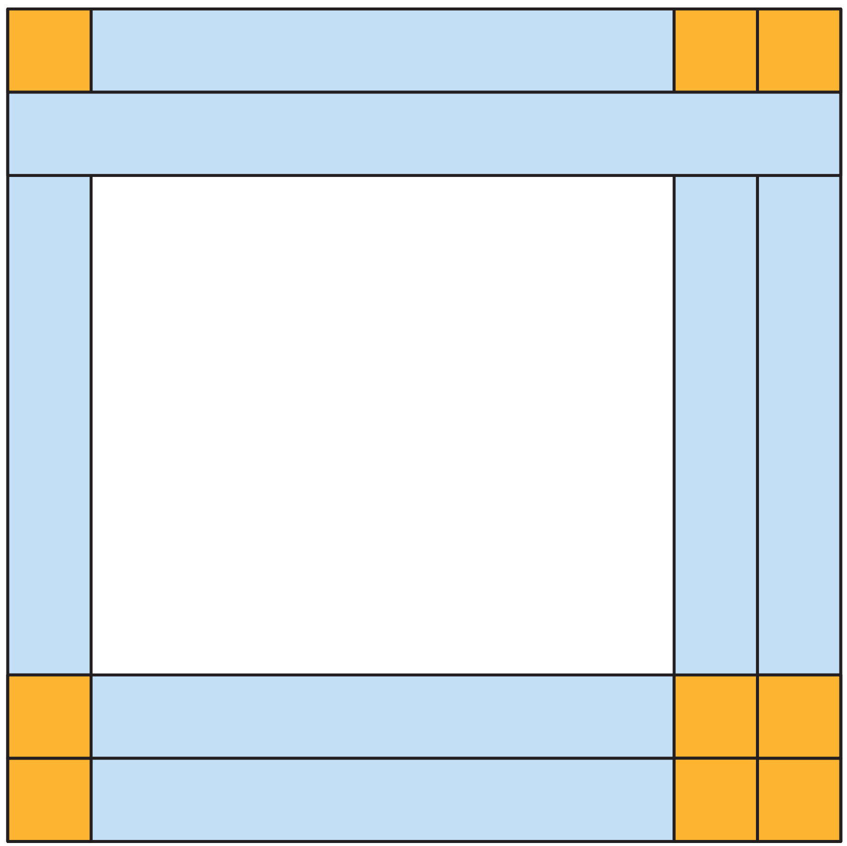
5.b. Quadrilateral and hexahedral meshing

- **Unconstrained Plastering**
  - Unmeshed boundary
  - Layers advance inward from the boundaries



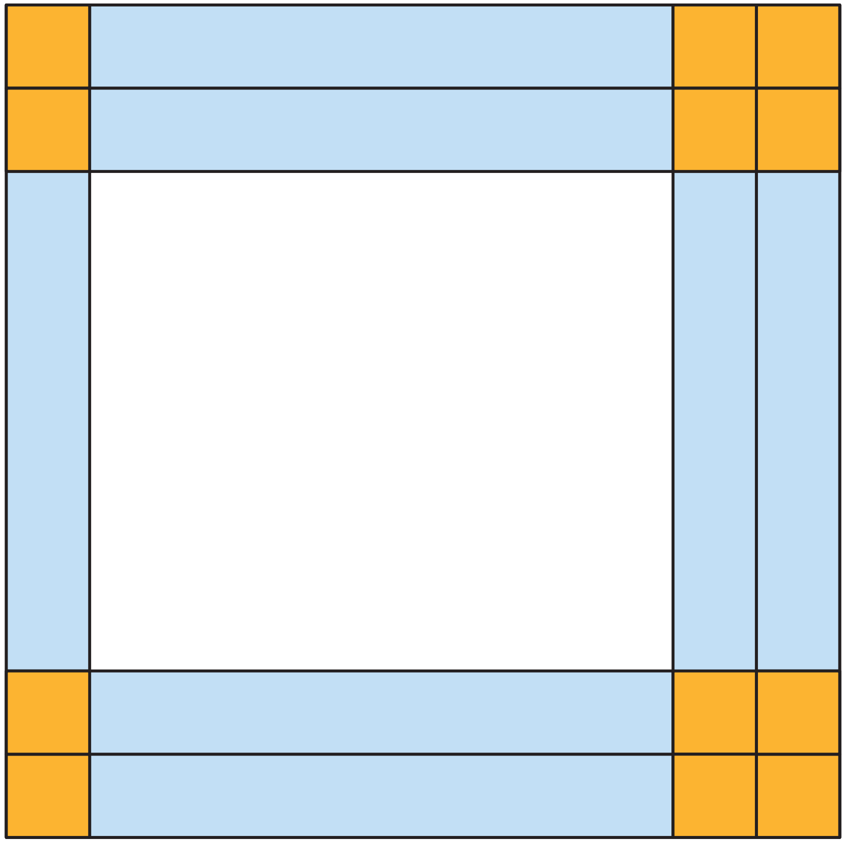
5.b. Quadrilateral and hexahedral meshing

- **Unconstrained Plastering**
  - Unmeshed boundary
  - Layers advance inward from the boundaries

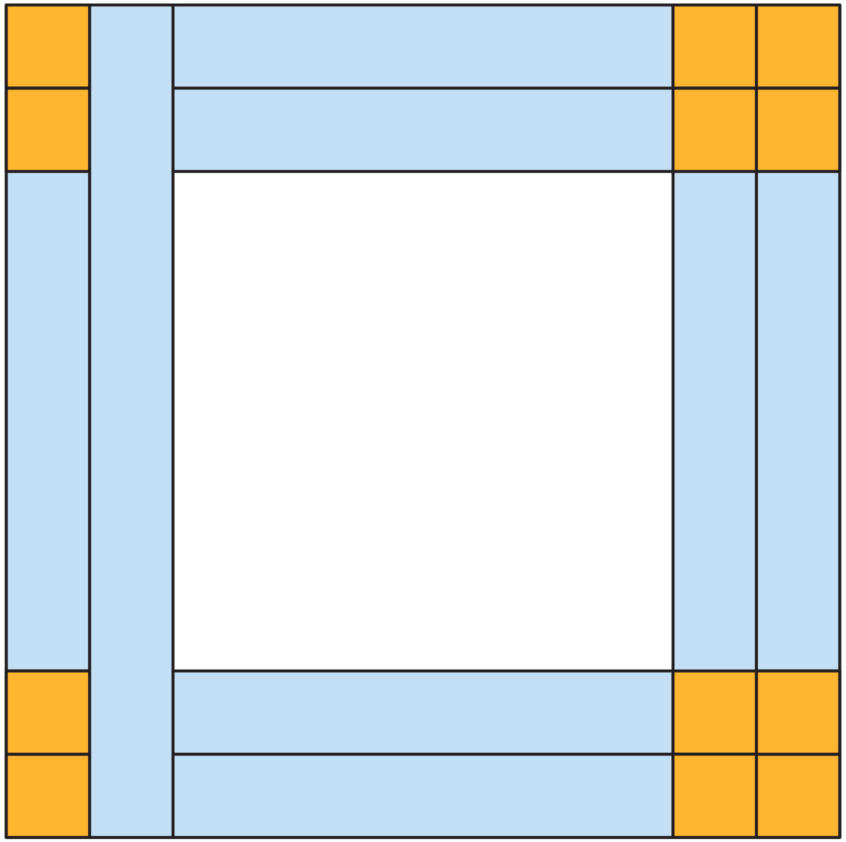


5.b. Quadrilateral and hexahedral meshing

- **Unconstrained Plastering**
  - Unmeshed boundary
  - Layers advance inward from the boundaries

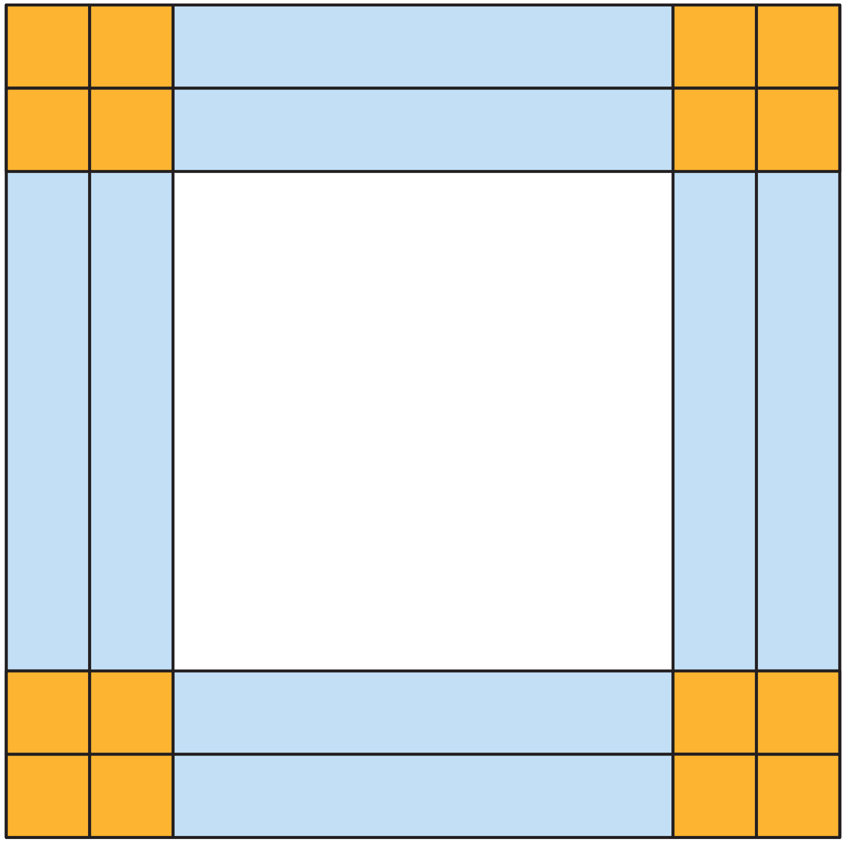


- **Unconstrained Plastering**
  - Unmeshed boundary
  - Layers advance inward from the boundaries



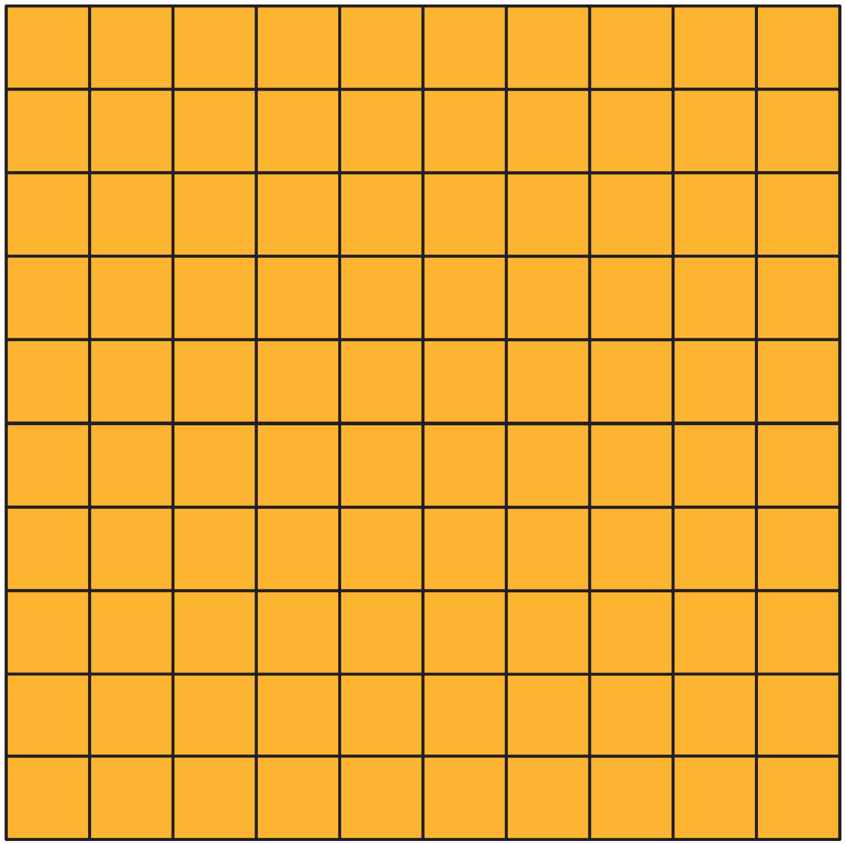
5.b. Quadrilateral and hexahedral meshing

- **Unconstrained Plastering**
  - Unmeshed boundary
  - Layers advance inward from the boundaries



5.b. Quadrilateral and hexahedral meshing

- **Unconstrained Plastering**
  - Unmeshed boundary
  - Layers advance inward from the boundaries

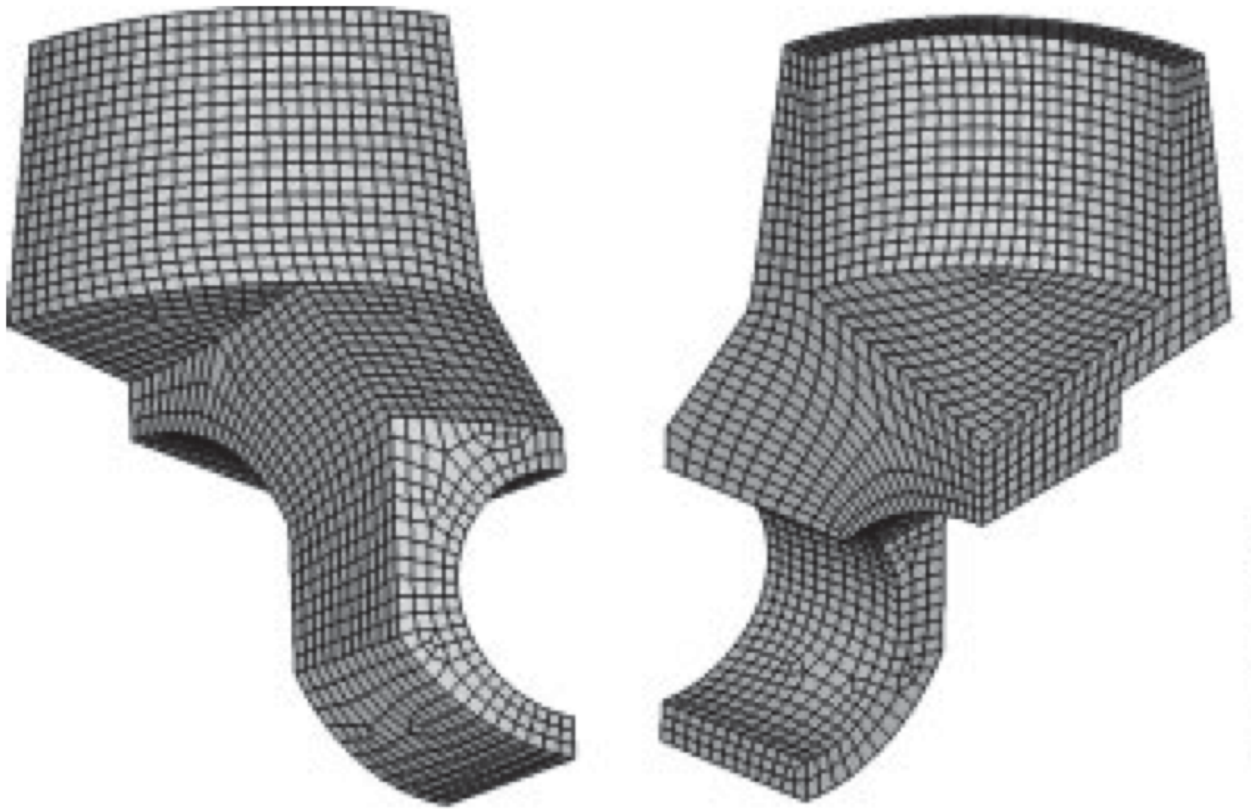


5.b. Quadrilateral and hexahedral meshing

- **Unconstrained Plastering**
  - Unmeshed boundary
  - Layers advance inward from the boundaries

- Advantages:
  - Fully automatic
  - High quality meshes near the boundary
- Drawbacks
  - May lead to mixed meshes
  - Time consuming

5.b. Quadrilateral and hexahedral meshing

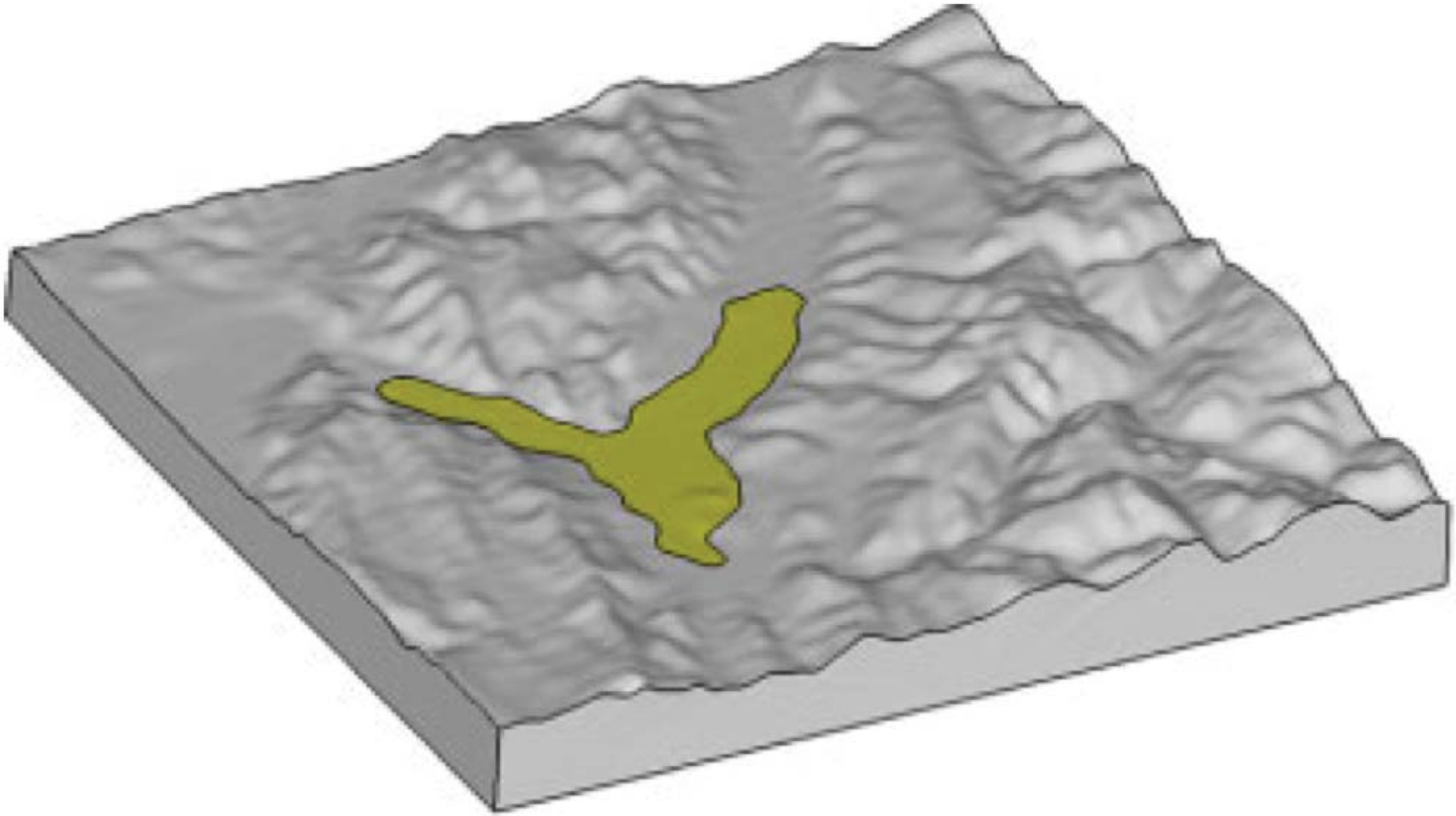


Images from M. Staten et al.

5.b. Quadrilateral and hexahedral meshing

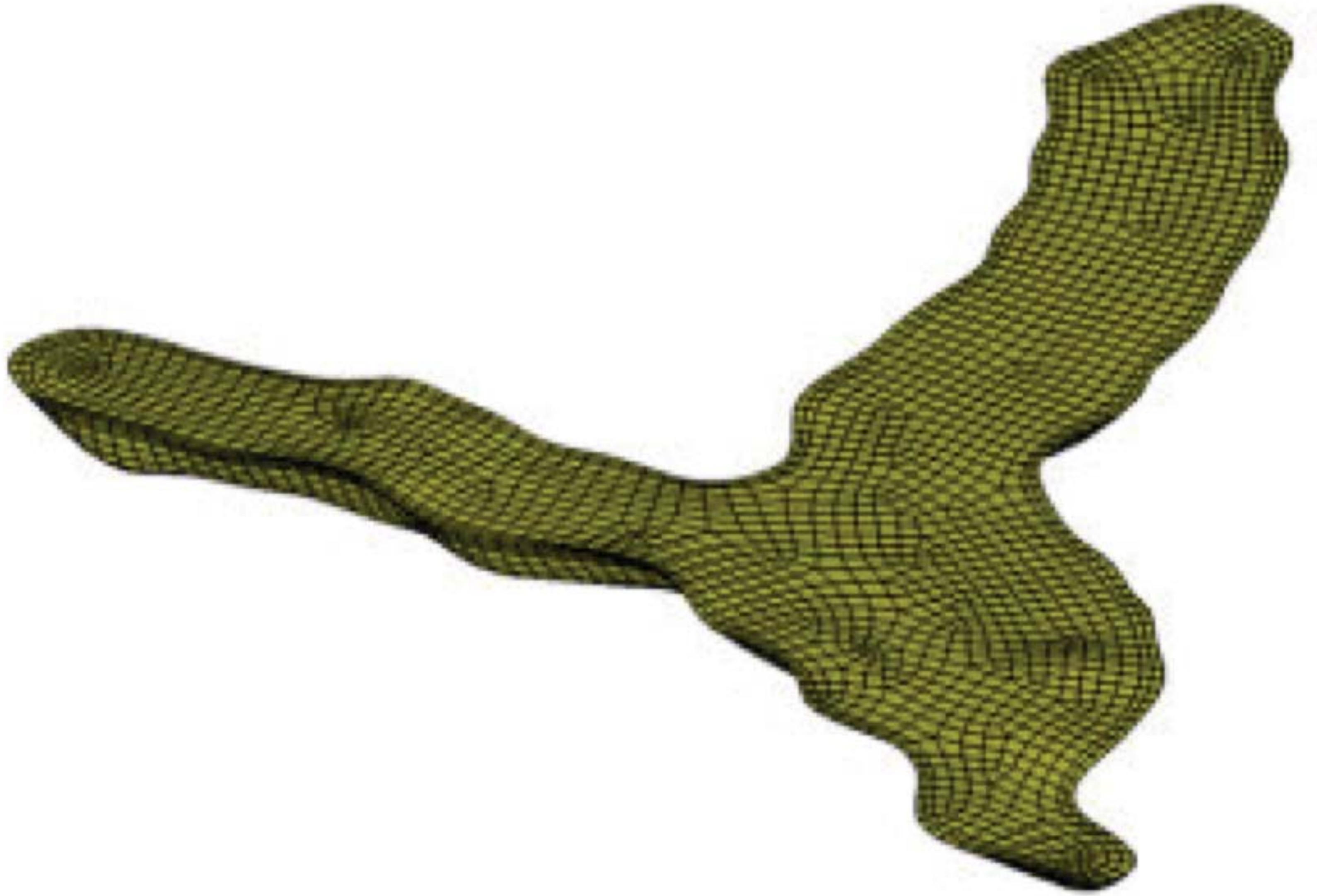
Images from M. Staten et al.

5.b. Quadrilateral and hexahedral meshing



Images from M. Staten et al.

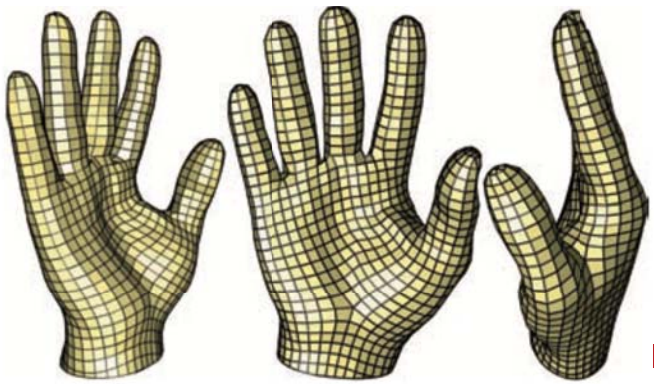
5.b. Quadrilateral and hexahedral meshing



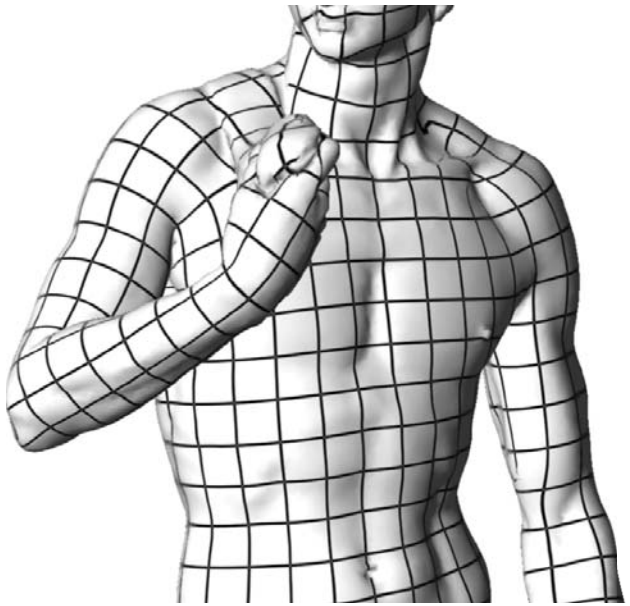
Images from M. Staten et al.

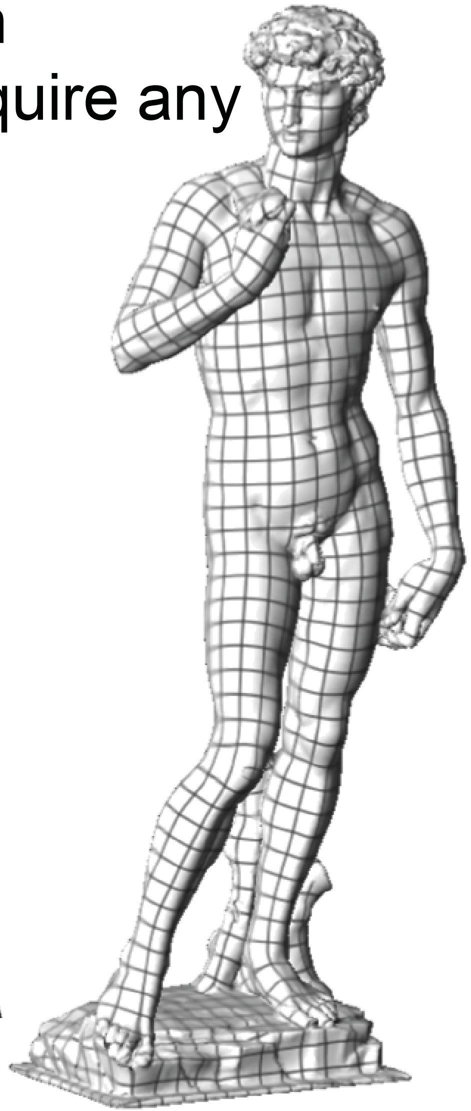
5.b. Quadrilateral and hexahedral meshing

- **Cross field based methods**
  - These methods initially sought a globally smooth parameterization of the surface that does not require any previous partition of the geometry.
  - These parameterizations are derived from a directional field
    - Curvature
    - Based on a PDE
  - They provide well shaped quadrilateral regions that are almost structured



Images from F. Kälberer et al.

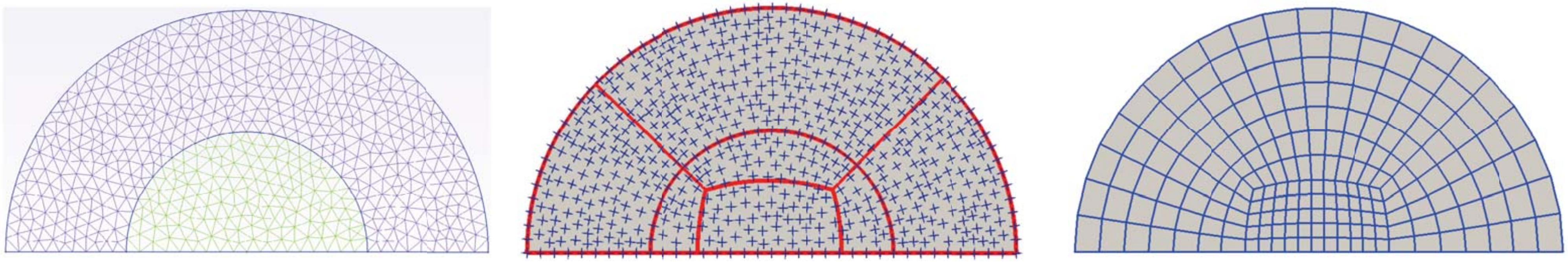




Images from N. Ray et al.

5.b. Quadrilateral and hexahedral meshing

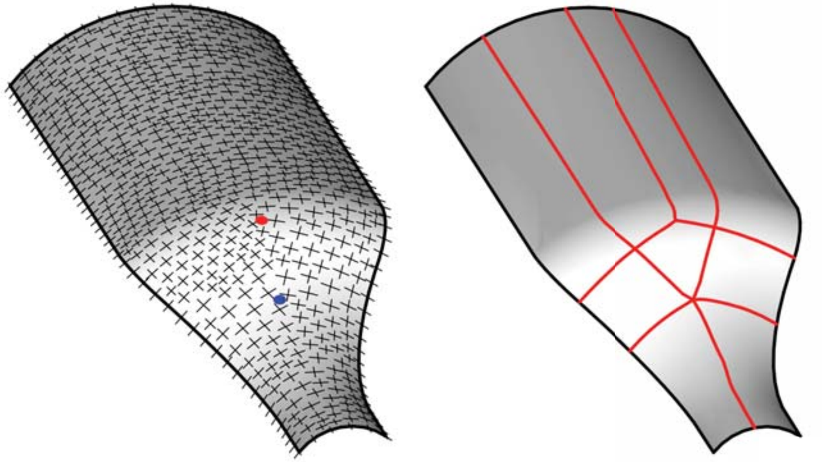
- These cross fields are used to compute an automatic partitioning of an arbitrary geometry
- The cross field is prescribed on the boundary and propagated towards the inner part using
  - PDE
  - fast marching algorithm
- Similar to sub-mapping but
  - for arbitrary geometries !!!
  - solve a PDE instead of linear integer problem
- A compatible quad mesh is generated from the partitioning



Images from N. Kowalski et al.

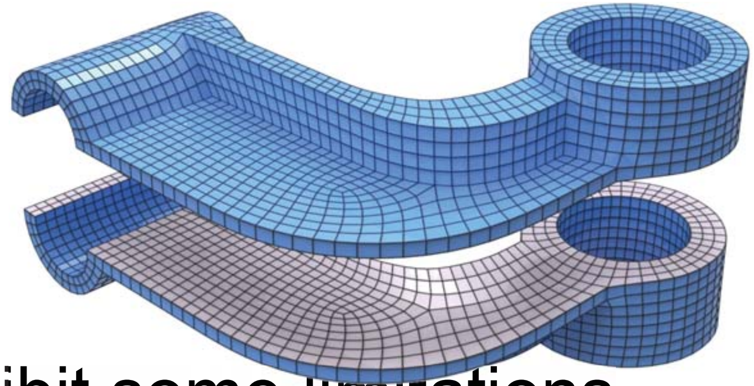
5.b. Quadrilateral and hexahedral meshing

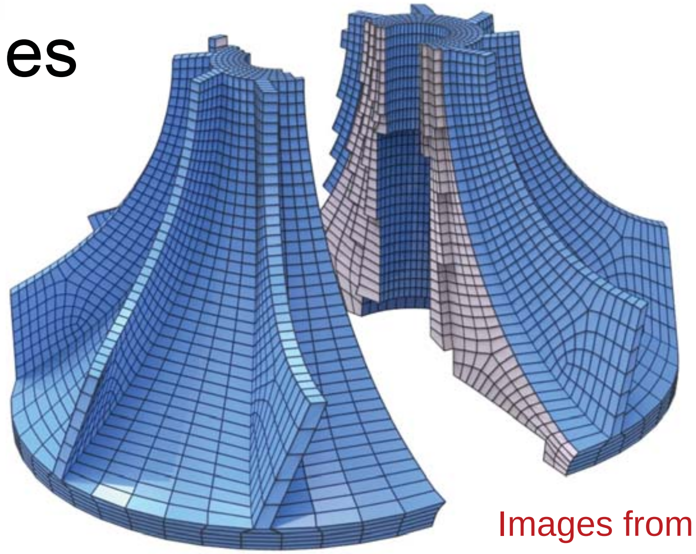
- The method can be extended to surfaces



Images from H.J. Fogg et al.

- Also extended to hexahedral meshes





Images from Y. Li et al.

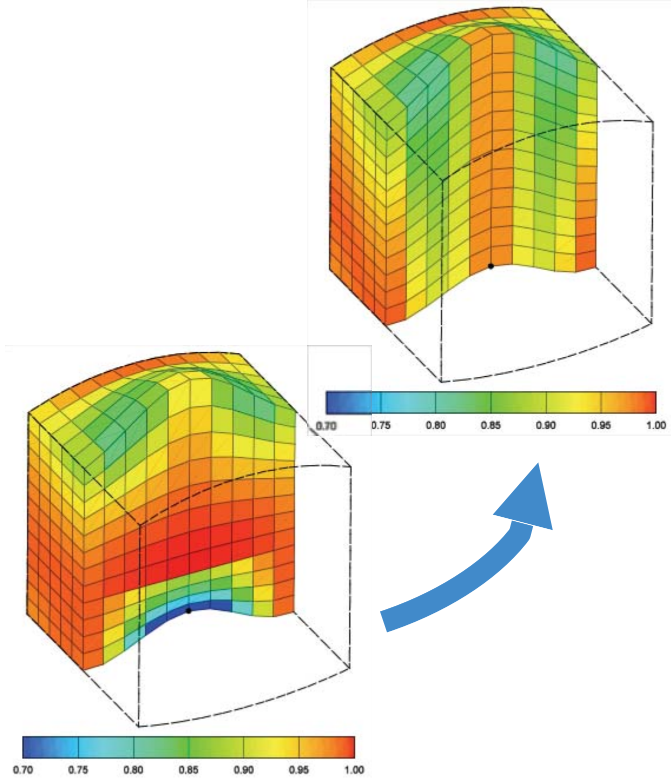
- Still exhibit some limitations
  - No guarantee of all-hex mesh for any
  - Further research is needed to deal with non-constant element size

Layout of the course

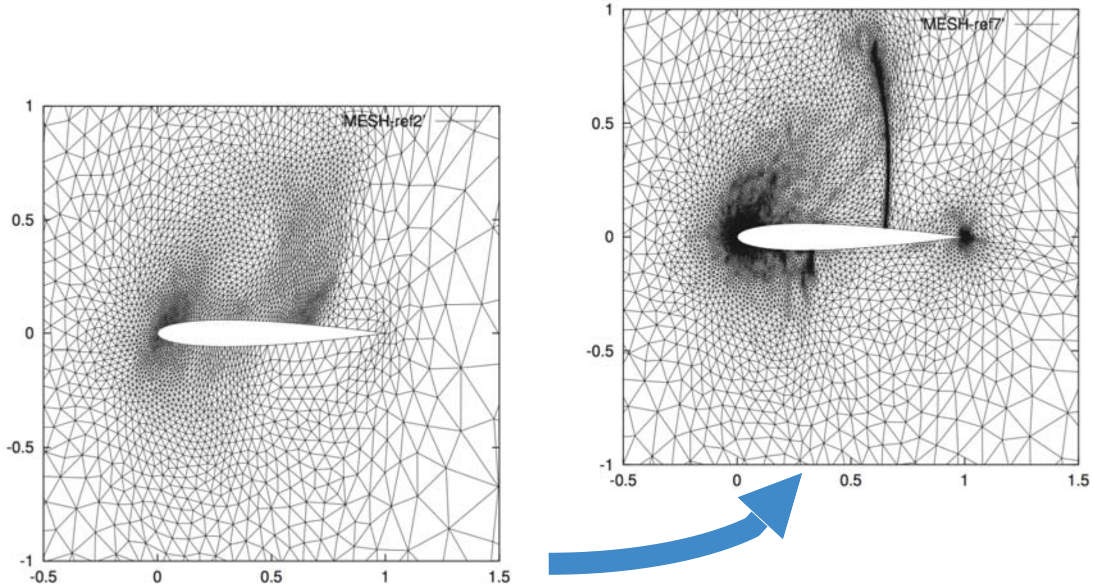
1. Why do we need meshes?
2. Geometry description
3. Classification of mesh generation methods
4. Structured mesh generation methods
5. Unstructured mesh generation methods
6. Mesh optimization and mesh adaption
7. Concluding remarks

6. Mesh optimization and mesh adaption

- Summary
  - Mesh optimization
    - Quality measures
    - Topological mesh optimization techniques
      - Tri / Tets
      - Quad / Hexes
    - Mesh smoothing techniques
      - Geometry based methods
      - Optimization based methods



  - Mesh adaption
    - Basic concepts
    - Embedded adaption
    - New mesh generation



2D Images from V. Dolejsi

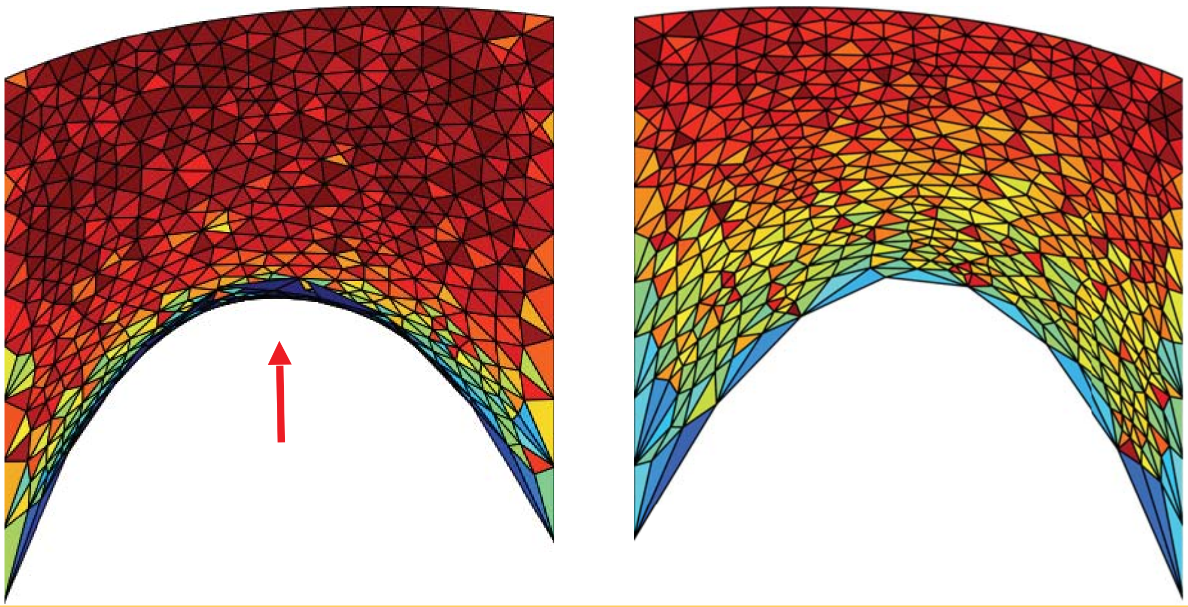
6. Mesh optimization and mesh adaption

- **Quality measures**
  - **Element quality**. A continuous strictly monotonic function
    - maximum for an ideal element and minimum for degenerated elements

$$\boxed{q \in [0, 1]}$$

    - invariant under translation, rotation, reflection and uniform scaling
  - **Mesh quality.** Based on the quality of the elements in a mesh:
    - Minimum / Maximum
    - Arithmetic average / Geometric average

Measure	Mesh 1	Mesh 2
Min. Q	0.00	0.22
Max Q.	1.00	0.99
Mean Q.	0.76	0.79
Std. Dev.	0.29	0.15
N. Tangl.	61	0



Examples

- Ratio between the inradius and the longest edge
- Ratio between the inradius and the circumradius

2D	3D
$q_{rR} = \dfrac{2r}{R}$	$q_{rR} = \dfrac{3r}{R}$
$q_{rh_{\max}} = 2\sqrt{3}\dfrac{r}{h_{\max}}$	$q_{rh_{\max}} = 2\sqrt{6}\dfrac{r}{h_{\max}}$
$h_{\max} = \max\limits_{i=1,\ldots,3} h_i$	$h_{\max} = \max\limits_{i=1,\ldots,6} h_i$

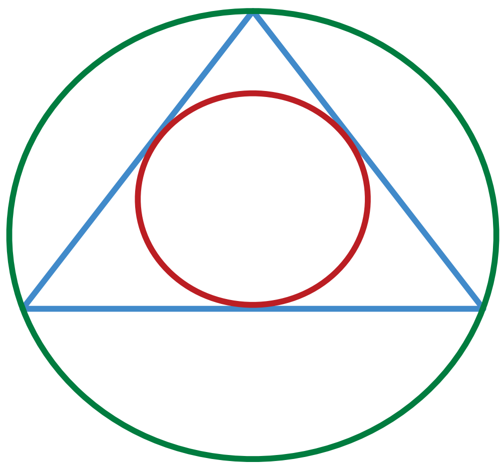
A Area

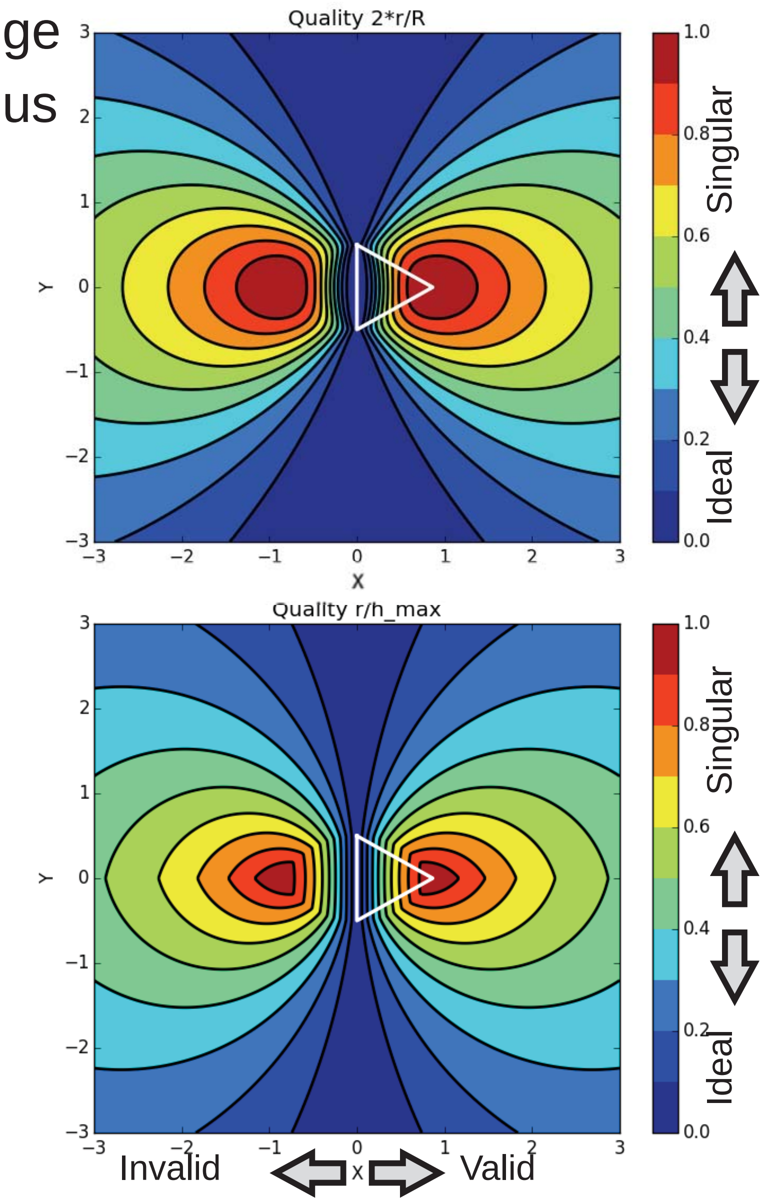
V Volume

r inradius

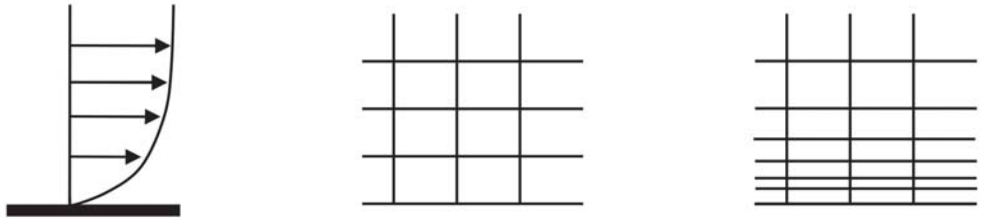
R circumradius

l_i length of the i-th edge

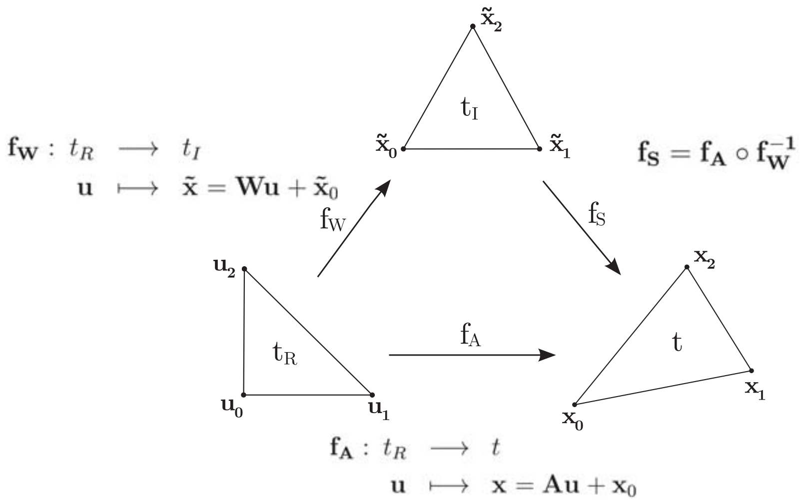




- Which is the ideal for a given problem?



- Algebraic quality measures



We can compute $\mathrm{f_S}$

$$\boxed{\begin{aligned} \mathbf{f_S} : t_I &\longrightarrow t \\ \tilde{\mathbf{x}} &\longmapsto \mathbf{x} = \mathbf{S}\tilde{\mathbf{x}} + \mathbf{v} \\ \mathbf{S} &= \mathbf{A}\mathbf{W}^{-1} \end{aligned}}$$

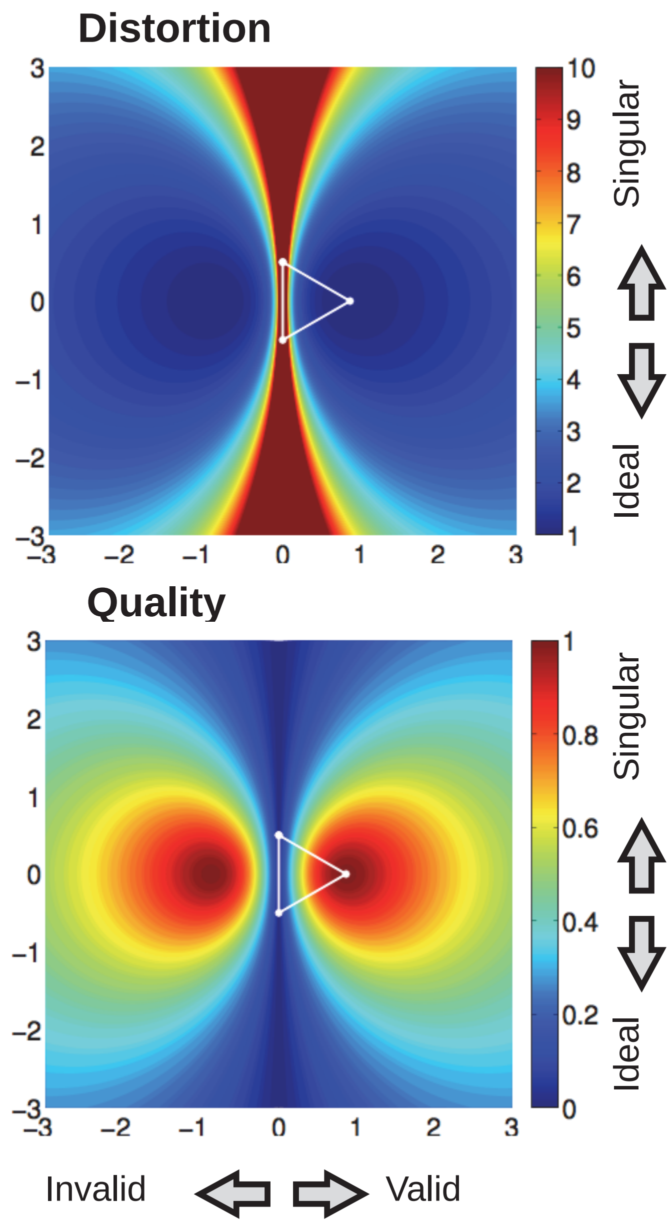
Shape Distortion

$$\boxed{\eta(\mathbf{S}) = \frac{|\mathbf{S}|^2}{n\sigma(\mathbf{S})^{2/n}}}$$

$$\sigma(\mathbf{S}) = \det(\mathbf{S})$$
$$|\mathbf{S}| = \sqrt{\mathrm{tr}\,(\mathbf{S}^T\mathbf{S})}$$

Shape Quality

$$\boxed{q = \frac{1}{\eta}} \qquad q \in [0, 1]$$



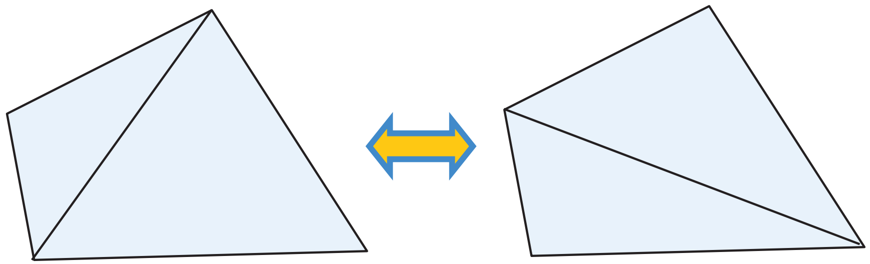
6. Mesh optimization and mesh adaption

Topological mesh optimization techniques

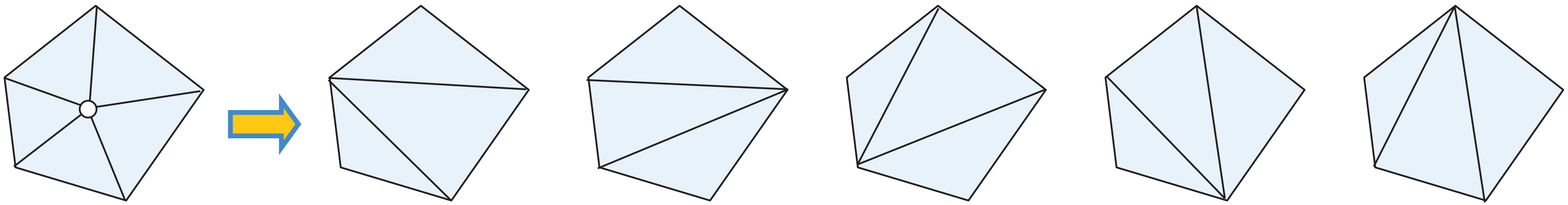
- **Objective**. Improve the quality of a given mesh by modifying the mesh topology (connectivity)
- **Classification**. According the dimension and element type
  - **2D**
    - Triangles
    - Quadrilaterals
  - **3D**
    - Tetrahedra
    - Hexahedra
- Simple in 2D
- More complicated in 3D
  - **Tetrahedra**. Optimize the number of edges around a node and the number of faces around and edge. Complex series of local topology operators !!
  - **Hexahedra**. Difficult !! Local modifications propagate far away

Triangles (1/2)

- **Edge swapping**. Swap the shared edge of two triangles forming a convex quadrilateral. It is the only local topological operator in two dimensions.



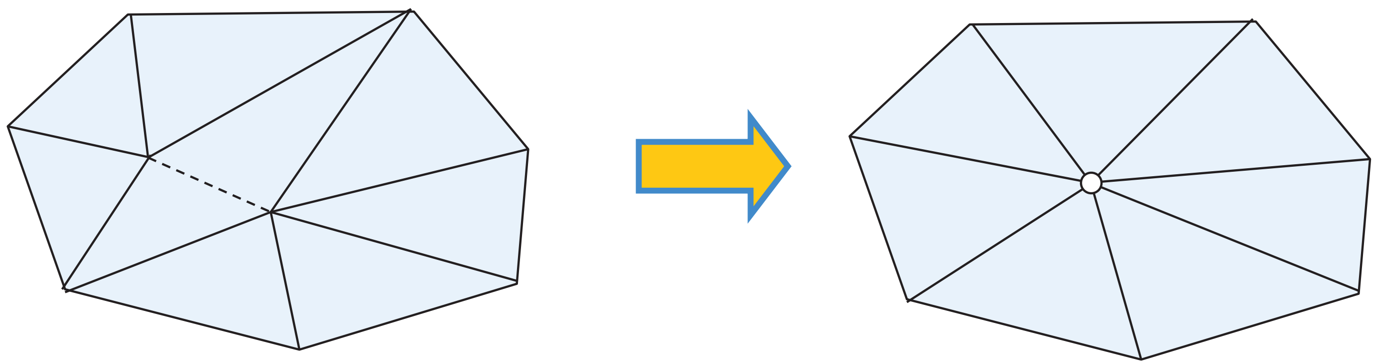
- **Node suppression**. The set of the possible remeshings of the polygon related to the triangles around a node are analyzed (in terms of quality). Optimum remeshing is chosen.



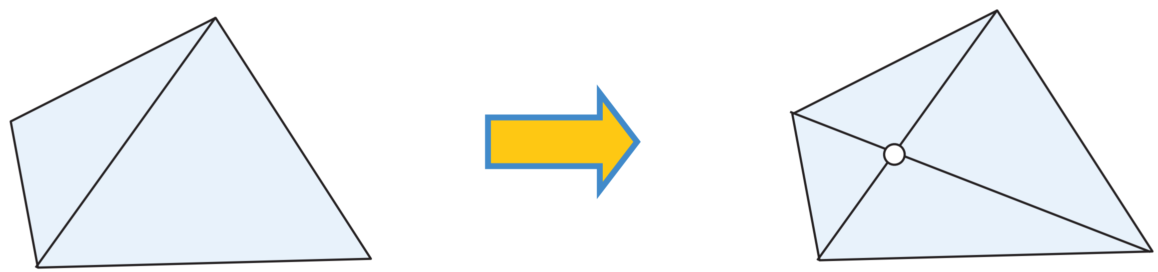
5 triangles around a node

The 5 triangulations related to a 5 points polygon

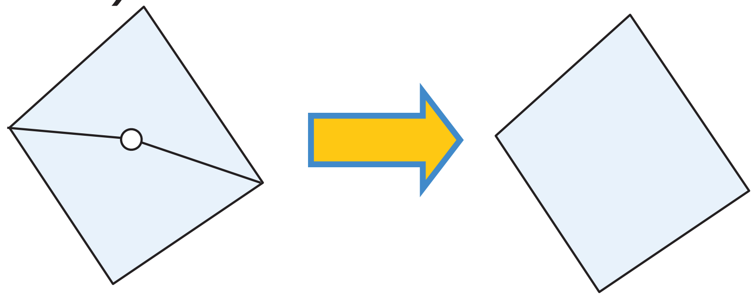
- **Triangles (2/2)**
  - **Edge suppression**. Replace an edge by a node



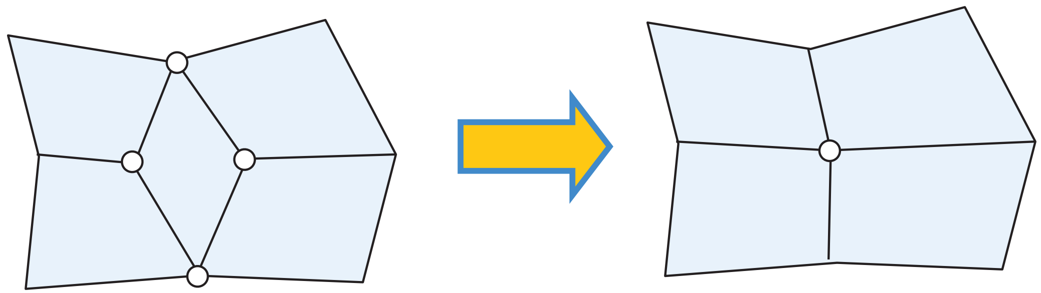
  - **Edge splitting**. Replace 2 triangles sharing and edge by 4 triangles adding a node along the shared edge.



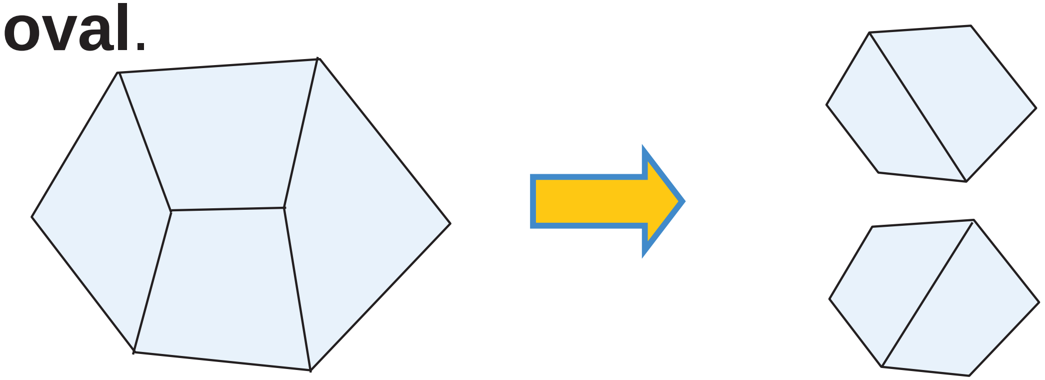
- **Quadrilaterals**
  - **Node suppression**. Suppress all nodes with only to adjacent quadrilaterals (doublet).



  - **Element removal**.

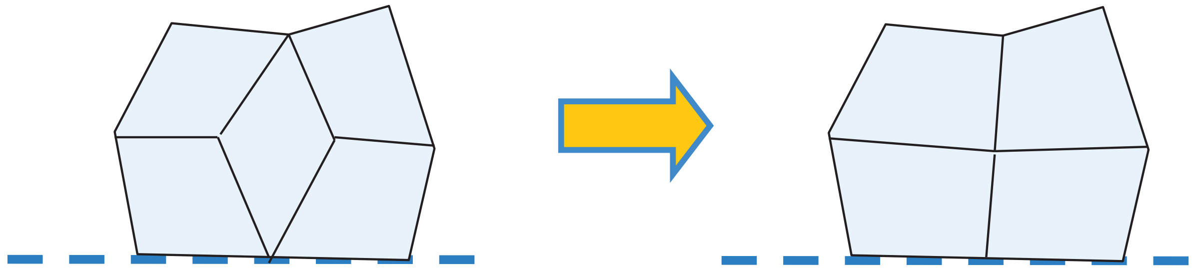


  - **Edge removal**.

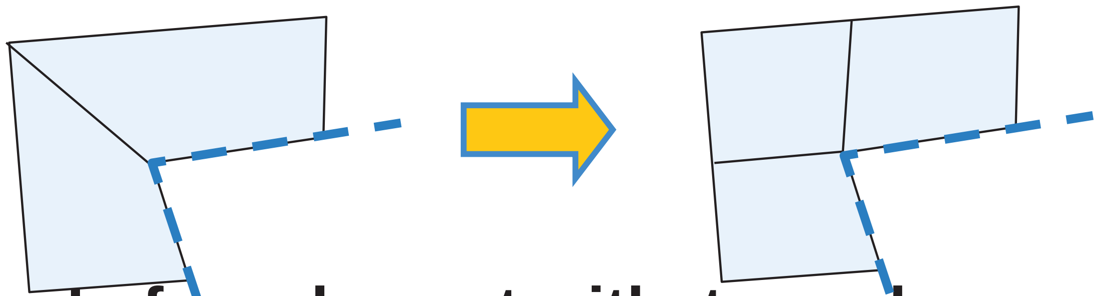


Quadrilaterals boundary clean up

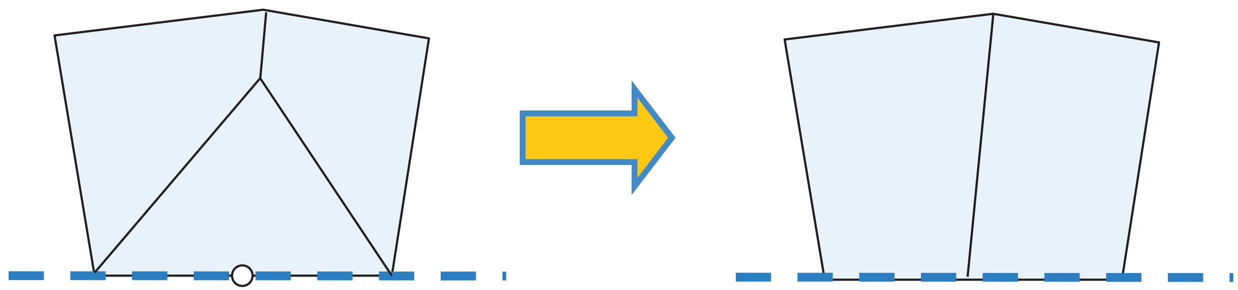
- **Element removal at boundary.**

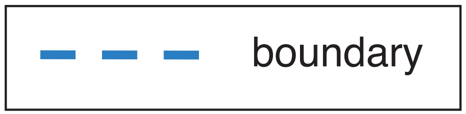


- **Element open at boundary.**

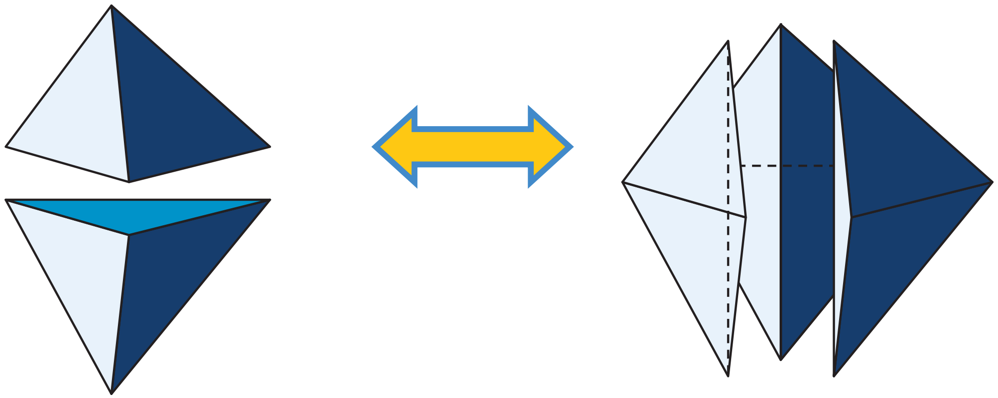


- **Removal of an element with two edges on boundary.**

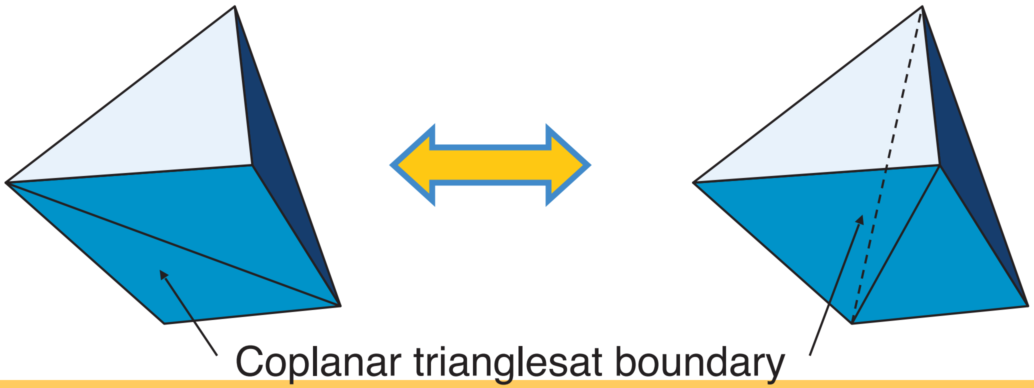




- ## Tetrahedra
  - **Face swapping**. Each inner face separates two tetrahedra, 5 nodes. There are two configurations:
    - **Inner tetrahedra**. From 2 to 3 tetrahedra adding interior edge.

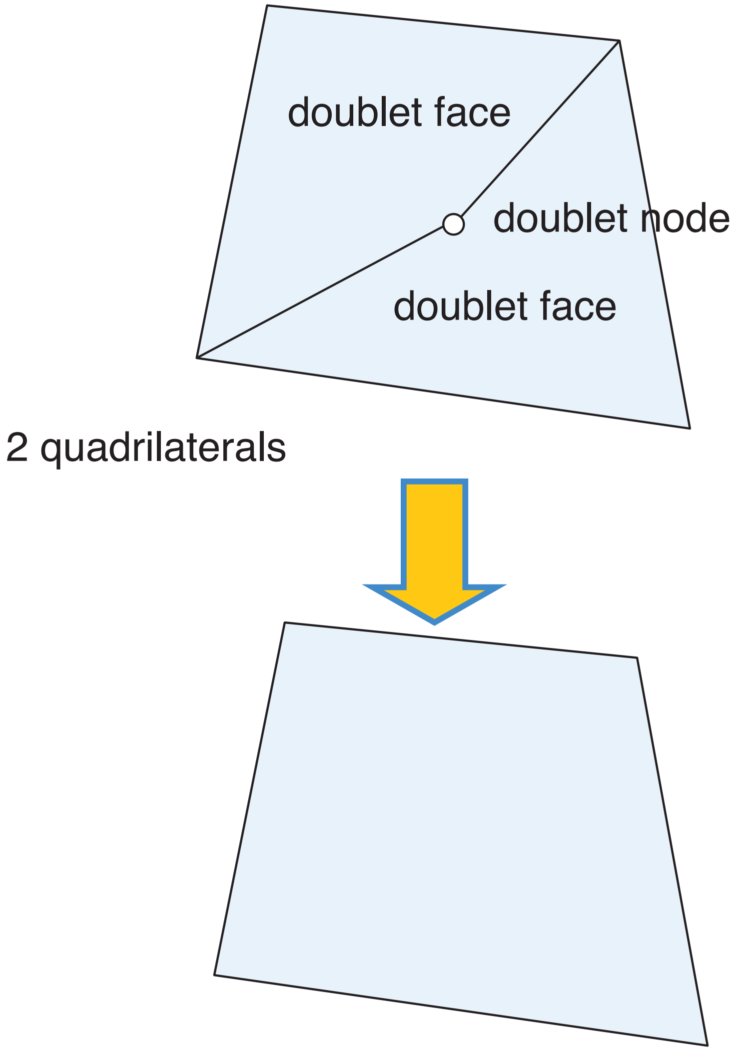


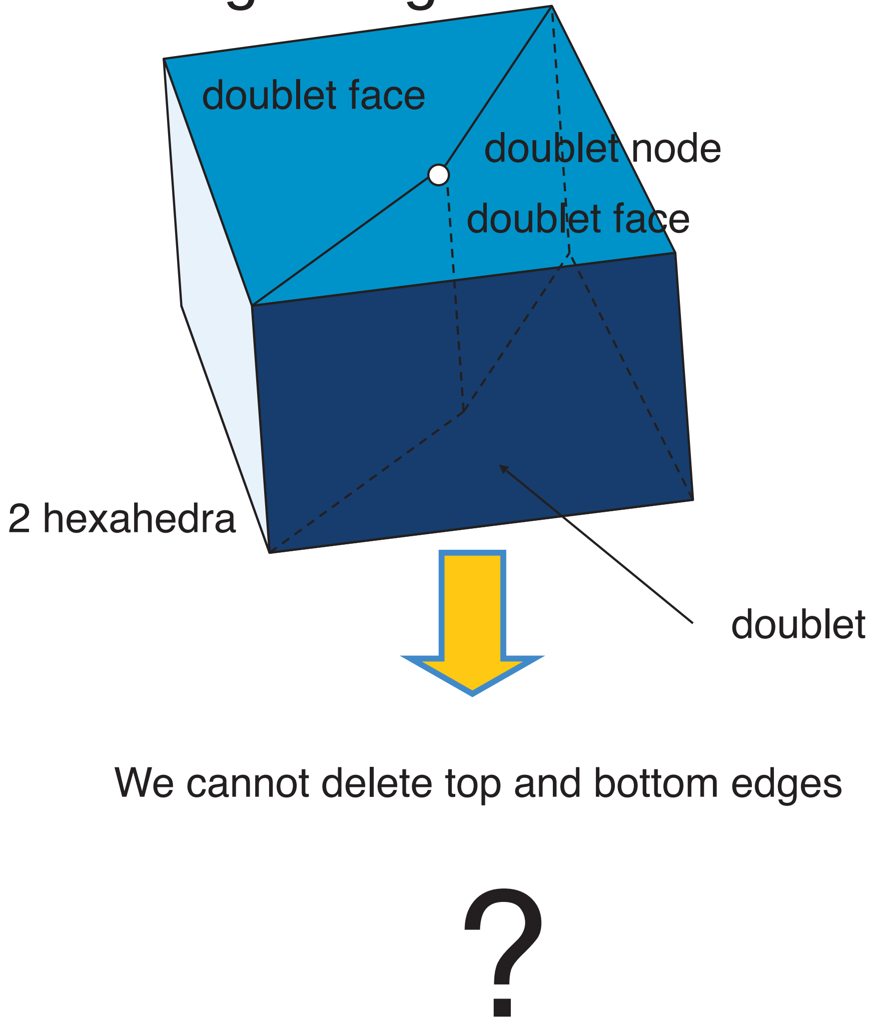
    - **Coplanar boundary**. Base are coplanar on the boundary. We can switch from 2 to 2 tetrahedra by swapping inner face.



6. Mesh optimization and mesh adaption

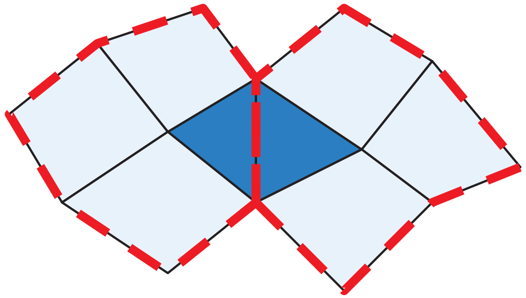
- **Hexahedra**
  - Doublet. 2 quadrilateral faces sharing 2 edges !!



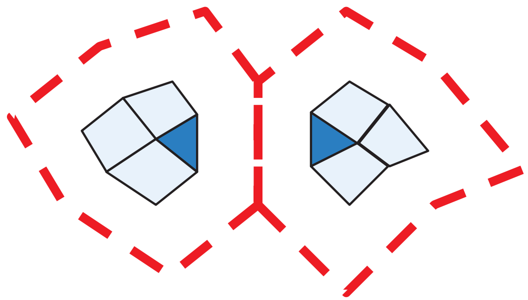


6. Mesh optimization and mesh adaption

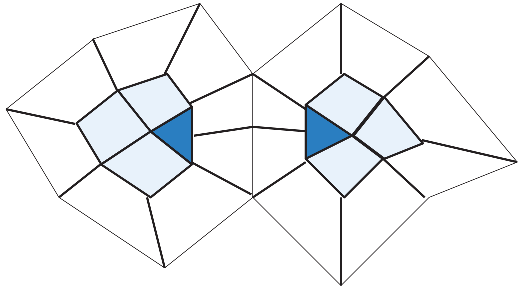
- **Hexahedra**
  - Pillowing (nonlocal in 3D):
    - For every doublet face find shrink set



    - Shrink sets



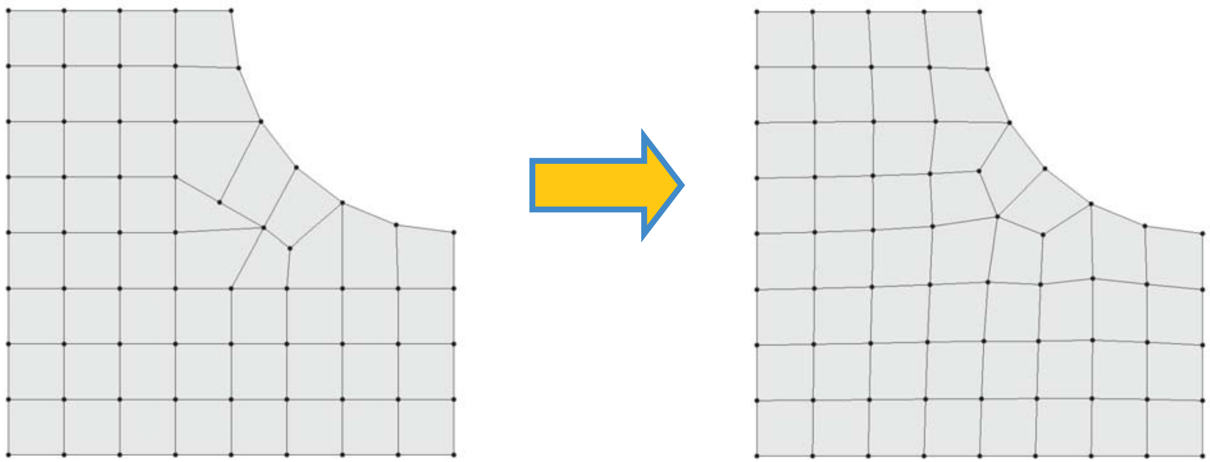
    - Connect



- **How to apply topological operators?**

1. Select a sequence of local operators
   - **Ordered** sequence of operators: some operators can destroy improvements of previous ones !!
2. For each operator
   - Calculate mesh quality
   - Select those elements to be improved
   - Use topology operators and propose solutions
   - Calculate the quality for the proposed solutions
   - Select best solution (initial or proposed solutions)
3. Use next operator, step 2

- **Mesh smoothing**
- **Objective:** Improve the quality of the mesh by changing the location of nodes (no topological modifications)



- **Classification**:
  - Laplacian like methods
  - Based on mechanical analogies
  - Optimization based methods

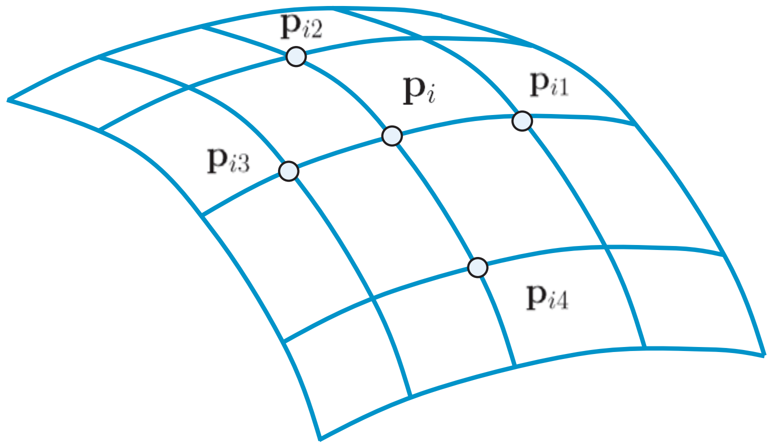
6. Mesh optimization and mesh adaption

- ## Laplacian smoothing
  - **General overview**
    - A number of smoothing techniques are lumped under this name
    - It is the most commonly used and the simplest smoothing method.
    - It can be applied to 2D (plane and surfaces) and 3D geometries.
  - **Advantages:**
    - This method is inexpensive to compute
  - **Drawbacks:**
  - It does not guarantee an improvement in the mesh quality
    - Sometimes generate poor quality meshes
    - Sometimes generate meshes with tangled elements (inner nodes move out of the domain)

6. Mesh optimization and mesh adaption

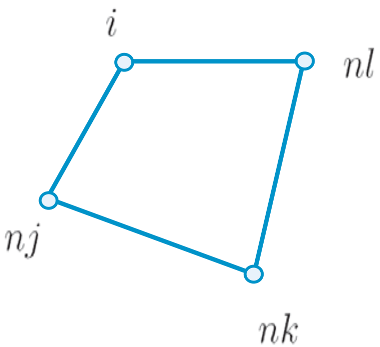
Laplacian smoothing

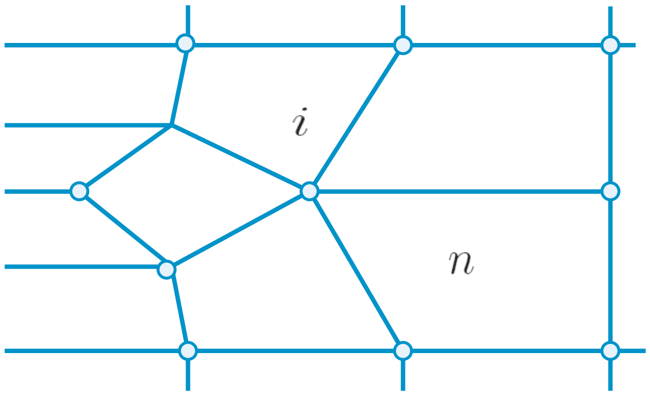
- For a structured and regular mesh:



$$\boxed{\mathbf{p}_i^{m+1} = \frac{1}{4}\left[\mathbf{p}_{i1}^m + \mathbf{p}_{i2}^m + \mathbf{p}_{i3}^m + \mathbf{p}_{i4}^m\right] \quad \text{for } i = 1, \ldots, M}$$

- Extension to unstructured meshes

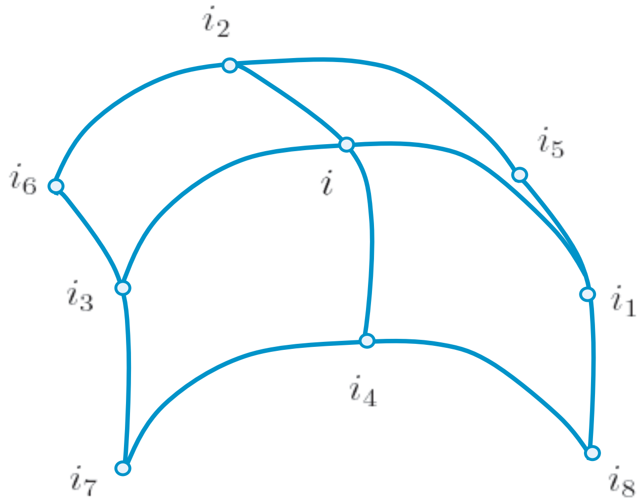




$$\boxed{\mathbf{p}_i^{m+1} = \frac{1}{2N_i}\sum_{n=1}^{N_i}\left[\mathbf{p}_{nj}^m + \mathbf{p}_{nl}^m\right] \quad \text{for } i = 1, \ldots, M}$$

A more general formulation for the Laplacian method

This figure suggests a single isoparametric element with curved sides and with the origin of the isoparametric coordinates at node "*i* "



$$\mathbf{p}_i(\xi,\eta) = \sum_{j=1}^{8} \mathbf{p}_{ij}\, N_j(\xi,\eta)$$

The new location of node "*i*" is

$$\mathbf{p}_i = \sum_{j=1}^{8} \mathbf{p}_{ij}\, N_j(0,0)$$

$$\boxed{\mathbf{p}_i = \frac{1}{4}\Big[2\,(\mathbf{p}_{i1}+\mathbf{p}_{i2}+\mathbf{p}_{i3}+\mathbf{p}_{i4}) - (\mathbf{p}_{i5}+\mathbf{p}_{i6}+\mathbf{p}_{i7}+\mathbf{p}_{i8})\Big]}$$

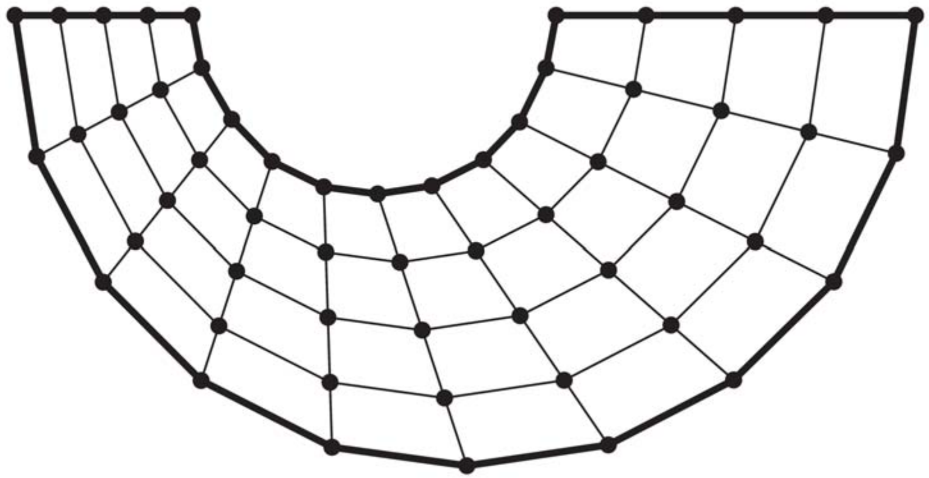
Generalization for non structured meshes:

$$\boxed{\mathbf{p}_i^{m+1} = \frac{1}{N_i(2-\omega)} \sum_{n=1}^{N_i} (\mathbf{p}_{nj}^{m} + \mathbf{p}_{nl}^{m} - \omega\mathbf{p}_{ik}^{m}) \qquad 0 \le \omega \le 1}$$

ω=0	laplacian method
ω=1	isoparametric method

6. Mesh optimization and mesh adaption

ω=0
Laplacian method



ω=1
Isoparametric method

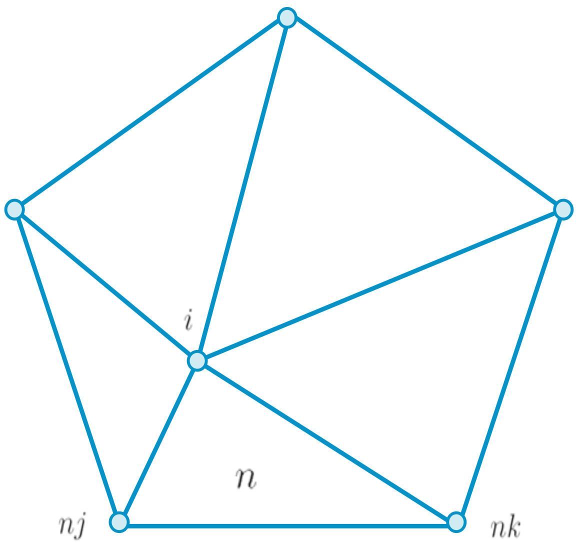
Extension to graded meshes:

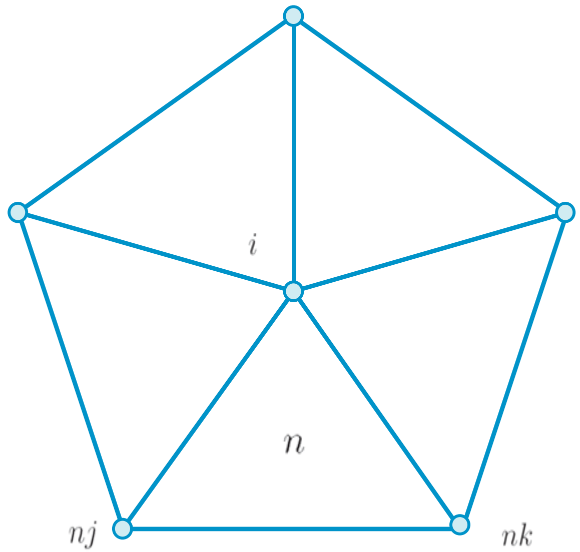
$$\mathbf{p}_i^{m+1} = \frac{\displaystyle\sum_{n=1}^{N_i} \frac{1}{l_n}(\mathbf{p}_{nj}^m + \mathbf{p}_{nl}^m) - \mathbf{p}_{ik}^m}{\displaystyle\sum_{n=1}^{N_i} \frac{1}{l_n}}$$

$$l_n = \frac{\|\mathbf{p}_{nj}^m - \mathbf{p}_i^m\| + \|\mathbf{p}_{nl}^m - \mathbf{p}_i^m\|}{2}$$

l_n is called the characteristic length

- **Extension to triangular meshes**



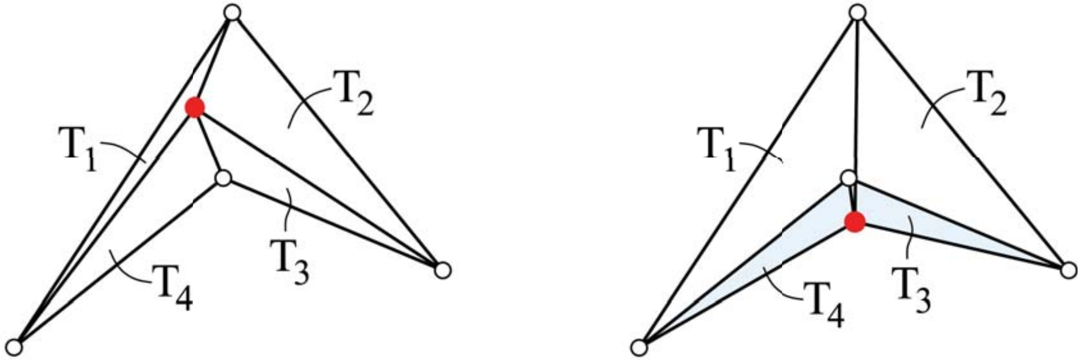


$$\boxed{\mathbf{p}_i^{m+1} = \frac{1}{N_i}\sum_{n=1}^{N_i}\left(\mathbf{p}_{nj}^m + \mathbf{p}_{nk}^m\right)}$$

All these methods has been extended to 3D (tets and hexes)

Smart Laplacian

- No effort is made to ensure that mesh quality is improved
- Poor elements (even tangled elements) can be generated



- Smart Laplacian algorithm

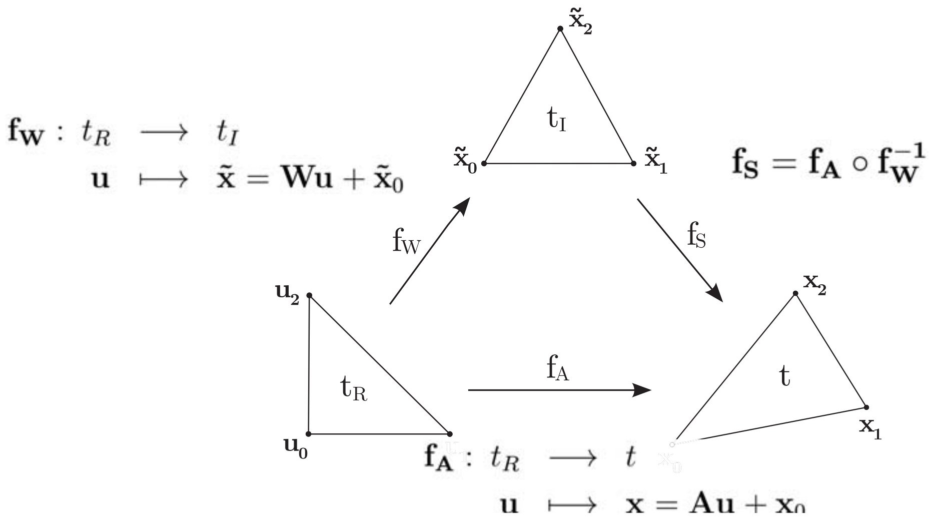
  For each node

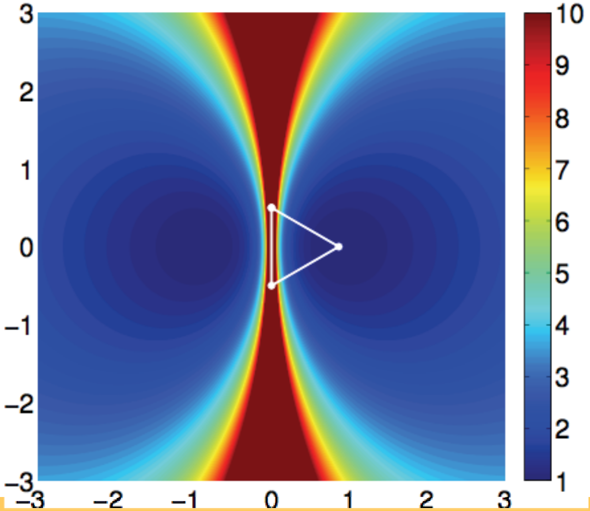
  - Compute a quality measure: $A(\mathbf{p}_i)$
  - Compute the new position: $$\mathbf{p}_i^{m+1} = \frac{1}{2N_i}\sum_{n=1}^{N_i}\left[\mathbf{p}_{nj}^m + \mathbf{p}_{nl}^m\right]$$
  - Compute the new quality: $A(\widetilde{\mathbf{p}}_i)$
  - If $A(\widetilde{\mathbf{p}}_i) > A(\mathbf{p}_i)$ $\quad\texttt{then}\quad$ $\mathbf{p}_i^{m+1} = \widetilde{\mathbf{p}}_i$

6. Mesh optimization and mesh adaption

Optimization based methods

- Minimization of an objective function:
- Several merit functions can be used: shape distortion measure





Shape Distortion $\boxed{\eta(\mathbf{S}) = \dfrac{|\mathbf{S}|^2}{n\sigma(\mathbf{S})^{2/n}}}$

$$\mathbf{S} = \mathbf{A}\mathbf{W}^{-1}$$

$$\sigma(\mathbf{S}) = \det(\mathbf{S})$$

$$|\mathbf{S}| = \sqrt{\operatorname{tr}(\mathbf{S}^T\mathbf{S})}$$

Shape Quality $\boxed{q = \dfrac{1}{\eta}}$ $\quad q \in [0, 1]$

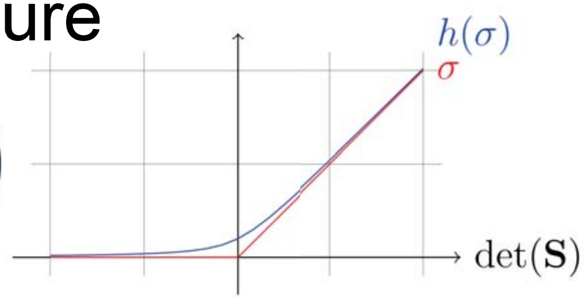
Vertical Barriers !!!

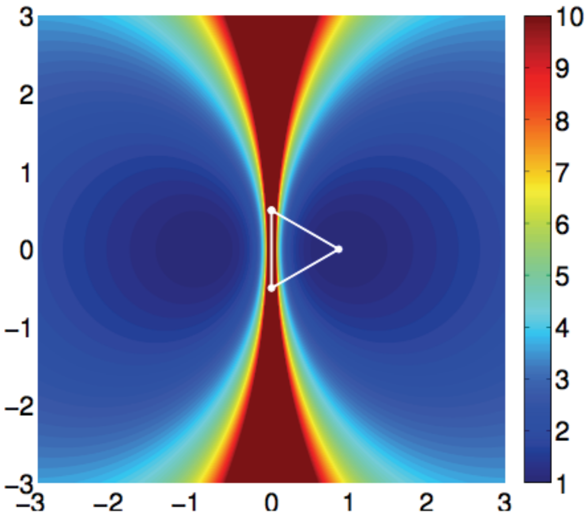
Not can be used in a continuous optimization procedure !!!

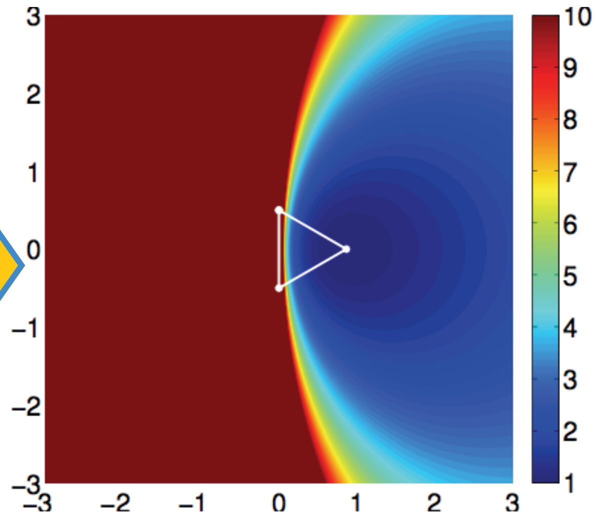
6. Mesh optimization and mesh adaption

- Regularization of the shape distortion measure

$$\boxed{\eta_{\text{mod}} = \frac{|\mathbf{S}|^2}{n h(\sigma)^{2/n}}} \qquad h(\sigma) = \frac{1}{2}\left(\sigma + \sqrt{\sigma^2 + 4\delta^2}\right)$$







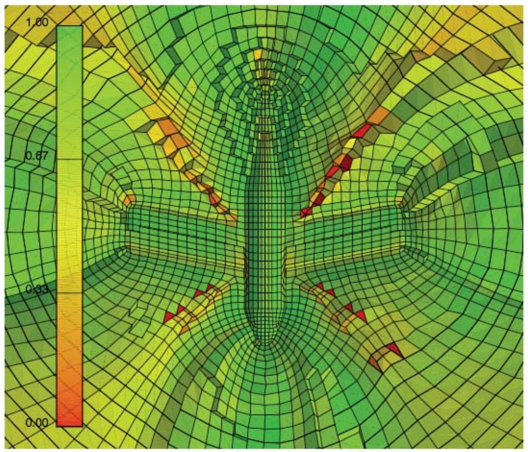
Singular

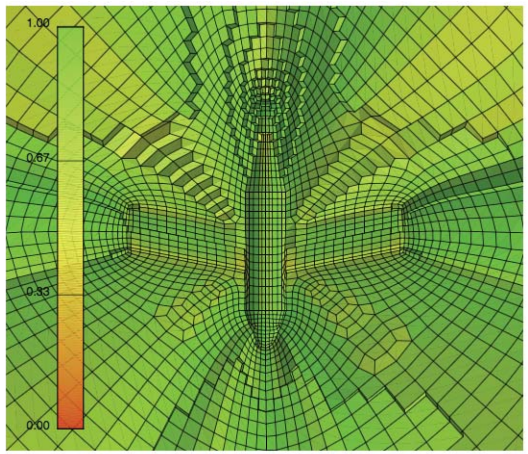
Ideal

No Barriers !!!

Can be used in a continuous optimization procedure !!!

Untangle capabilities !!!





Layout of the course

1. Why do we need meshes?
2. Geometry description
3. Classification of mesh generation methods
4. Structured mesh generation methods
5. Unstructured mesh generation methods
6. Mesh optimization and mesh adaption
7. Concluding remarks

- **Mesh generation is**
  - a required step in the numerical simulation process

    ⇒ has a major impact in industry
    - directly related to the geometry representation (CAD model, images, …)
    - directly related to the physics of the problem
    - directly related to the numerical method used in the simulation
  - a thrilling research field where **engineering** and **mathematical** skills are combined
  - a path that we are paving

Torture

Painful process

Innocuous treatment

Pleasant experience

- **Research lines (tentative list)**
  - Automatic geometry (CAD, image, …) adaption (cleaning, healing, …)
  - Automatic geometry decomposition
  - Unstructured hexahedral mesh generation (there not exist an automatic unstructured mesh generation algorithm for hexahedral meshes)
  - Anisotropic mesh adaption
  - High-order mesh generation
  - Mesh morphing
  - N-1 model representation (medial axis, skeletons, …)
  - Specific methods/applications require specific meshes:
    - always something to do …
    - hopefully we will still get paid !!!

- **Suggested readings**
  - Thompson J.F., Soni B.K., Weatherill N.P, Handbook of Grid Generation, CRC Press, 1999
  - Frey P., George P.L., Mesh Generation, John Wiley & Sons, 2000
  - George P.L., Borouchaki H., Delaunay Triangulation and Meshing, Hermes, Paris, 1998
  - Lo, S.H., Finite Element Mesh Generation. London: CRC Press, 2015
  - Topping B.H.V., Muylle J., Putanowicz R. Cheng B., Finite Element Mesh Generation, Saxe-Coburg Publications, 2004
  - Knupp P., Steinberg S., Fundamentals of Grid Generation, CRC Press, 1993
  - International Meshing Roundtable (http://imr.sandia.gov/)


That's all Folks!