051176 - Computational Techniques for Thermochemical Propulsion

Master of Science in Aeronautical Engineering

The Finite Volume Mesh

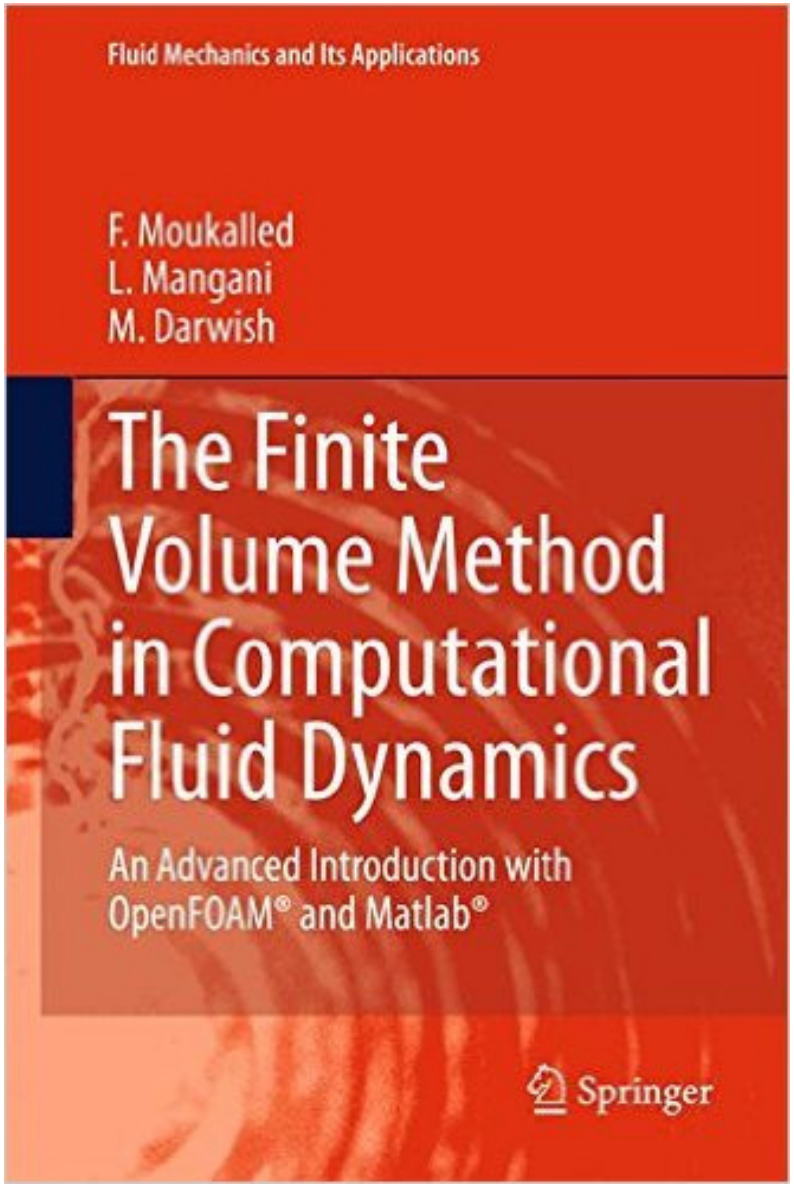
Prof. **Federico Piscaglia**

Dept. of Aerospace Science and Technology (DAER)

POLITECNICO DI MILANO, Italy

federico.piscaglia@polimi.it

Bibliography

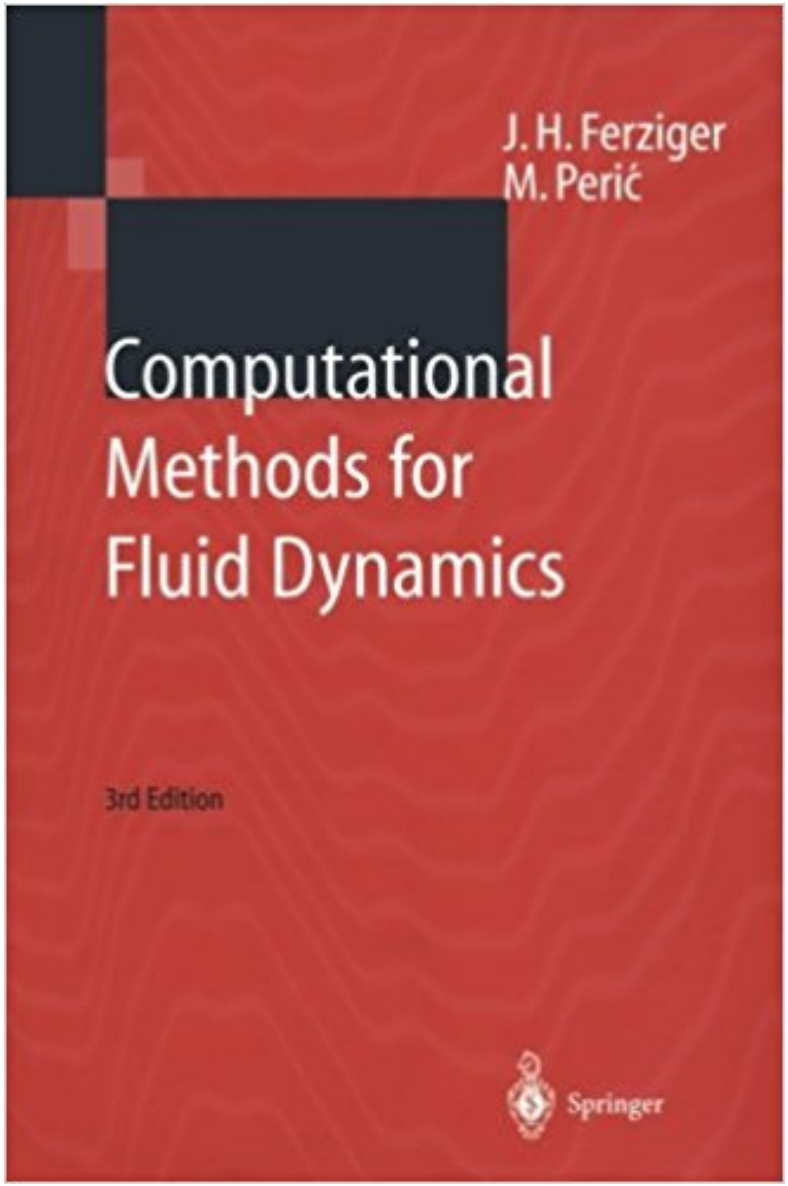


Some figures in this slides are taken from the reference text book:

F. Moukalled, L. Mangani, M. Darwish. "**The Finite Volume Method in Computational Fluid Dynamics**", Springer International Publishing Switzerland 2016.

Prof. Marwan Darwish is greatly acknowledged for sharing the images from his book and for allowing to include them in this course's material.

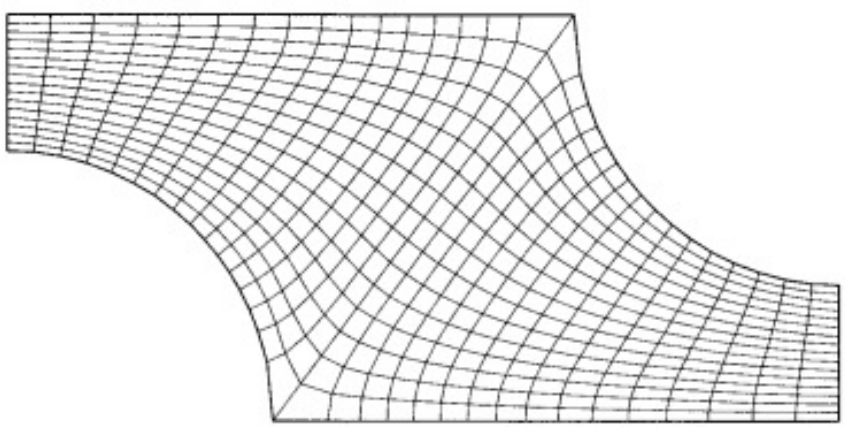
Bibliography

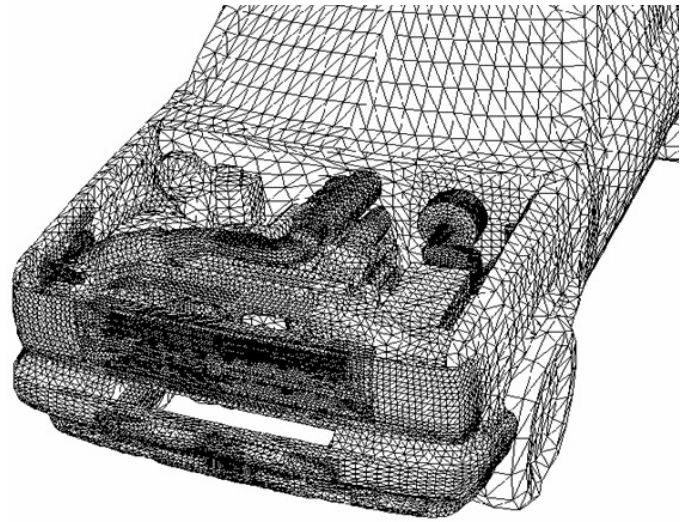


Ferziger, Joel H., Peric, Milovan. "**Computational Methods for Fluid Dynamics**", Third Edition, Springer 2002.

The choice of the grid

CFD codes are structured around the numerical algorithms that can tackle fluid flow problems.





The discrete locations at which the variables are calculated are defined by the numerical grid which is essentially a discrete representation of the geometric domain on which the problem is to be solved.

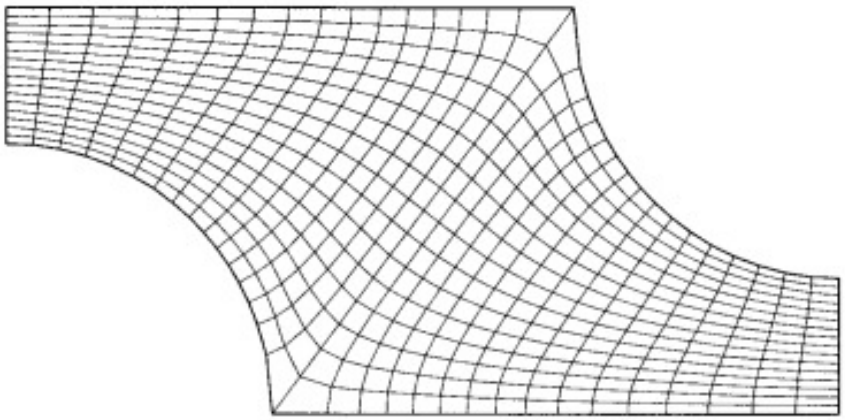
- The computational grid (or mesh) is subject to constraints imposed by the discretization method: if the algorithm is designed for hexahedra, grids consisting of triangles and tetrahedra cannot be used, etc.
- When the geometry is complex and the constraints cannot be fulfilled, compromises have to be made.

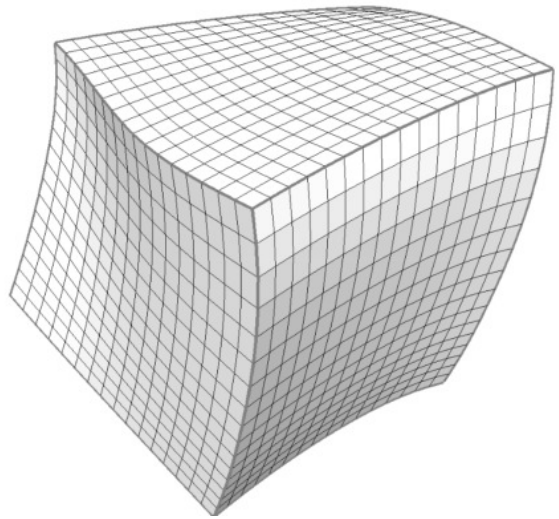
The Numerical Grid

The computational grid (or mesh) divides the solution domain into a finite number of subdomains (elements or control volumes). A first classification for computational grids is:

- **Structured (regular)**
- **Unstructured**
- **Block-structured**
- **Overlapping/Chimera**

Structured (regular)



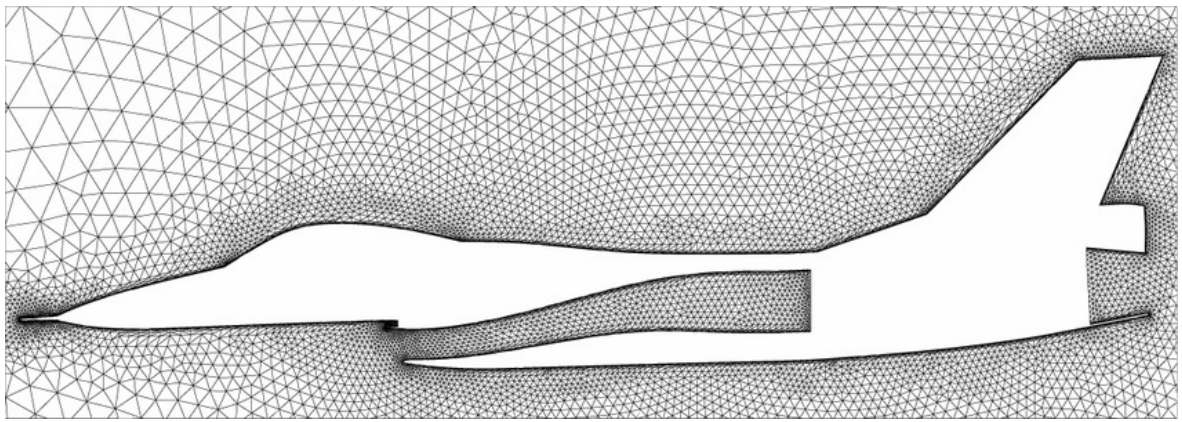


Structured (regular) grid: consist of families of grid lines with the property that members of a single family do not cross each other and cross each member of the other families only once:

- the lines of a given set to be numbered consecutively;
- the position of any grid point (or control volume) within the domain is uniquely identified by a set of two (in 2D) or three (in 3D) indices, e.g. *(i, j, k)*. This neighbor connectivity simplifies programming and the matrix of the algebraic equation system has a regular structure
- The disadvantage of structured grids is that they can be used only for geometrically simple solution domains. Another disadvantage is that it may be difficult to control the distribution of the

Unstructured grid

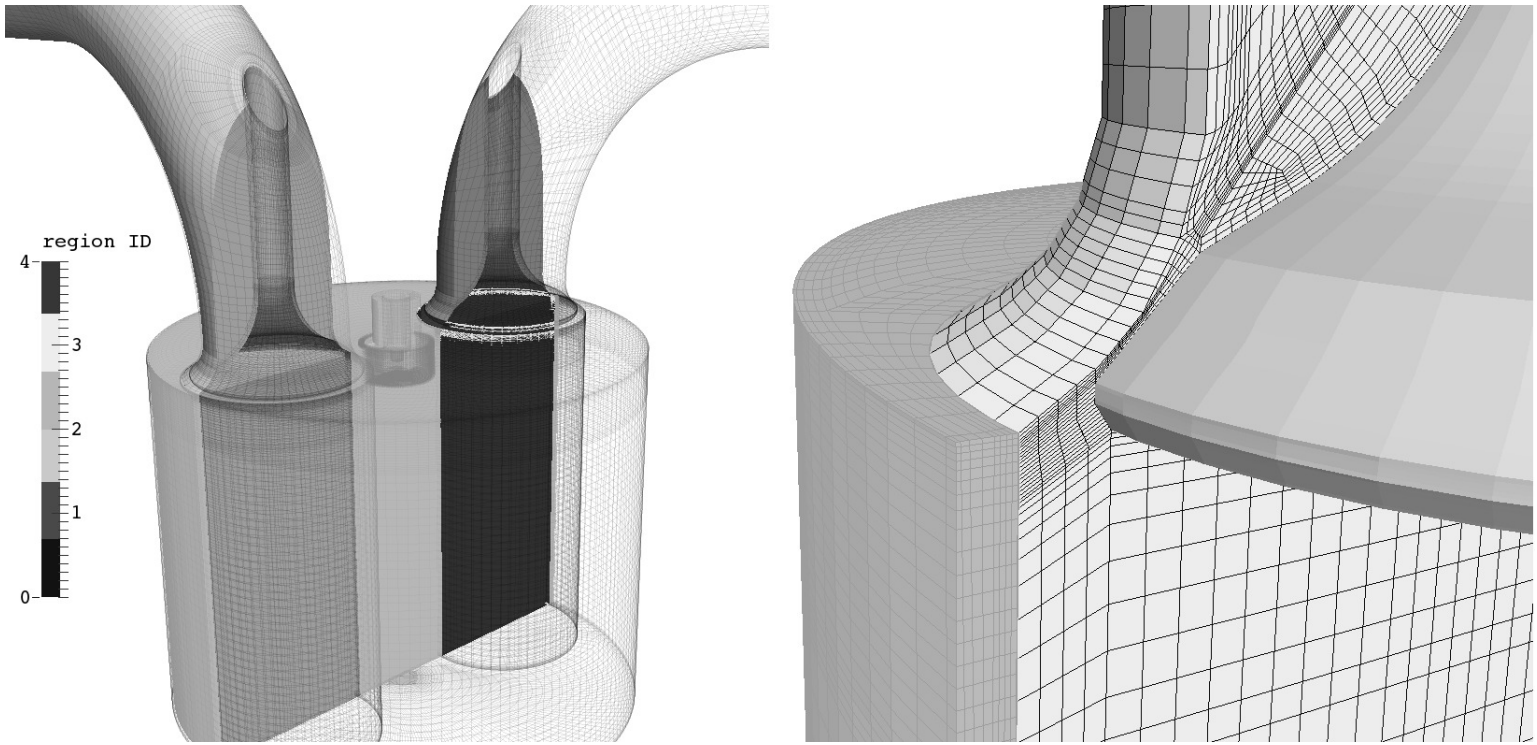
Unstructured grid: For very complex geometries, the most flexible type of grid is one which can fit an arbitrary solution domain boundary. In principle, such grids could be used with any discretization scheme, but they are best adapted to the finite volume and finite element approaches.



- control volumens (CVs) may have any shape; there is not a restriction on the number of neighbor elements or nodes. In practice, grids made of triangles or quadrilaterals in 2D, and tetrahedra or hexahedra in 3D are most often used;
- The advantage of flexibility is offset by the disadvantage of the irregularity of the data structure. Node locations and neighbor connections need be specified explicitly.
- The matrix of the algebraic equation system no longer has regular, diagonal structure; the band width needs to be reduced by reordering of the points. The solvers for the algebraic equation systems are usually slower than those for regular grids.

Block-structured grid

Block-structured grid: In a block structured grid, there is a two (or more) level subdivision of solution domain.

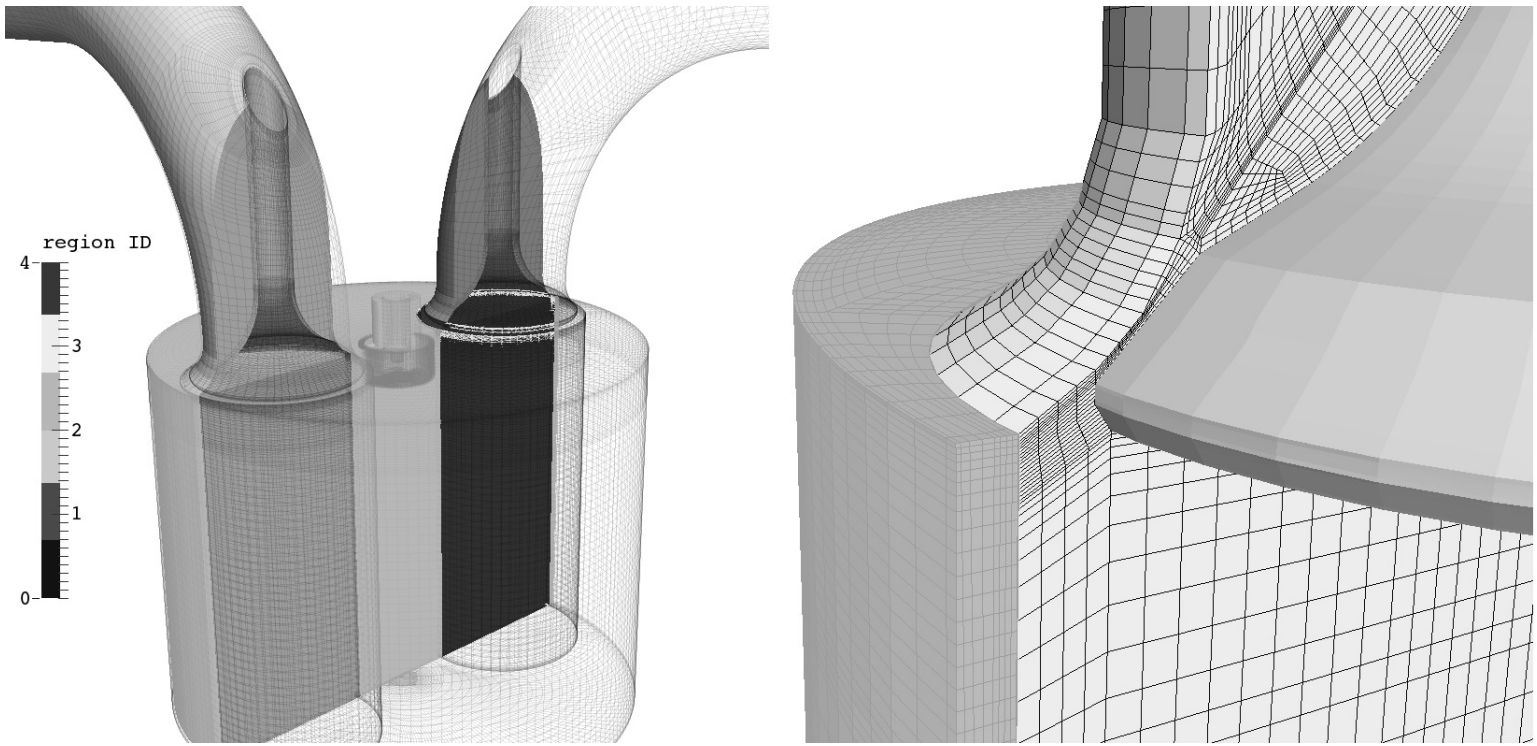


from SAE 2015-01-0284

Global indices are generally used for building the full system of equations over the computational domain, while local indices are employed to define the local stencil for an element, information that is useful during the discretization process. In structured grid systems local indices are used interchangeably as global indices

Block-structured grid

Block-structured grid: In a block structured grid, there is a two (or more) level subdivision of solution domain.



from SAE 2015-01-0284

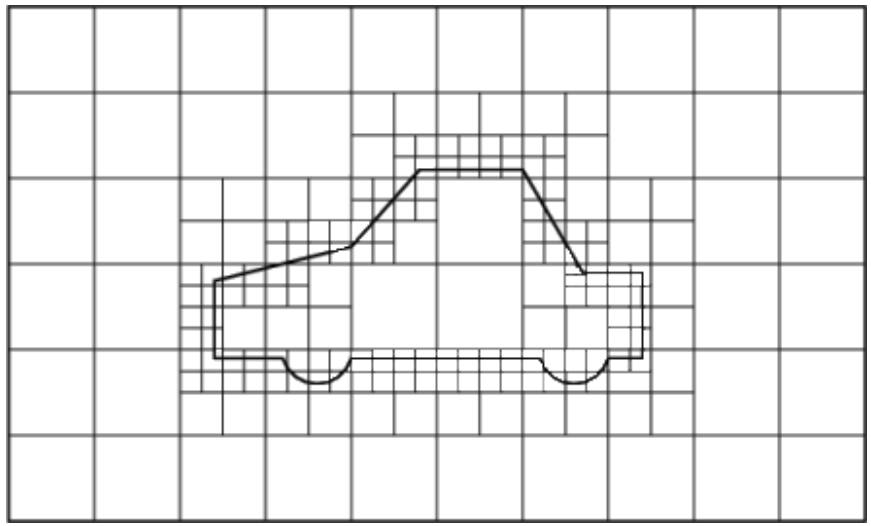
- On the coarse level, there are blocks which are relatively large segments of the domain; their structure may be irregular and they may or may not overlap;
- on the fine level (within each block) a structured grid is defined.

The choice of the grid

When the geometry is regular (e.g. rectangular or circular), choosing the grid is simple: the grid lines usually follow the coordinate directions. In complicated geometries, the choice is not at all trivial. The grid is subject to constraints imposed by the discretization method. If the algorithm is designed for curvilinear orthogonal grids, non-orthogonal grids cannot be used; if the CVs are required to be quadrilaterals or hexahedra, grids consisting of triangles and tetrahedra cannot be used, etc. When the geometry is complex and the constraints cannot be fulfilled, compromises have to be made.

Stepwise Approximation Using Regular Grids

The simplest approach uses orthogonal grids (Cartesian or polar-cylindrical). In order to apply such a grid to solution domains with inclined or curved boundaries, the boundaries have to be approximated by staircase-like steps.



The castellated mesh in `snappyHexMesh` is an example of application of stepwise approximation of inclined boundaries.

- The number of grid points (or CVs) per grid line is not constant, as it is in a fully regular grid. This requires either indirect addressing.
- The steps at the boundary introduce errors into the solution, especially when the grid is coarse. The treatment of the boundary conditions at stepwise walls also requires special attention.

Body-Fitted Non-Orthogonal Grids

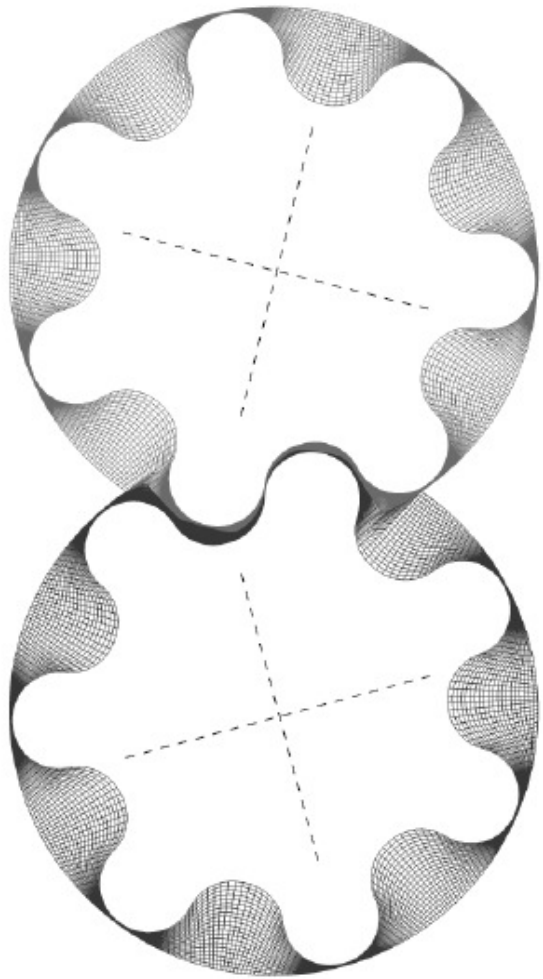
Boundary-fitted non-orthogonal grids are most often used to calculate flows in complex geometries (most commercial codes use such grids). They can be structured, block-structured, or unstructured.

Advantages:

- they can be adapted to any geometry, and that optimum properties are easier to achieve than with orthogonal curvilinear grids;
- Since the grid lines follow the boundaries, the boundary conditions are more easily implemented than with stepwise approximation of curved boundaries;
- the grid can also be adapted to the flow

Disadvantages:

- transformed equations contain more terms thereby increasing both the difficulty of programming and the cost of solving the equations,
- the grid non-orthogonality may cause unphysical solutions and the arrangement of variables on the grid affects the accuracy and efficiency of the algorithm.



Overlapping (Chimera) Grids

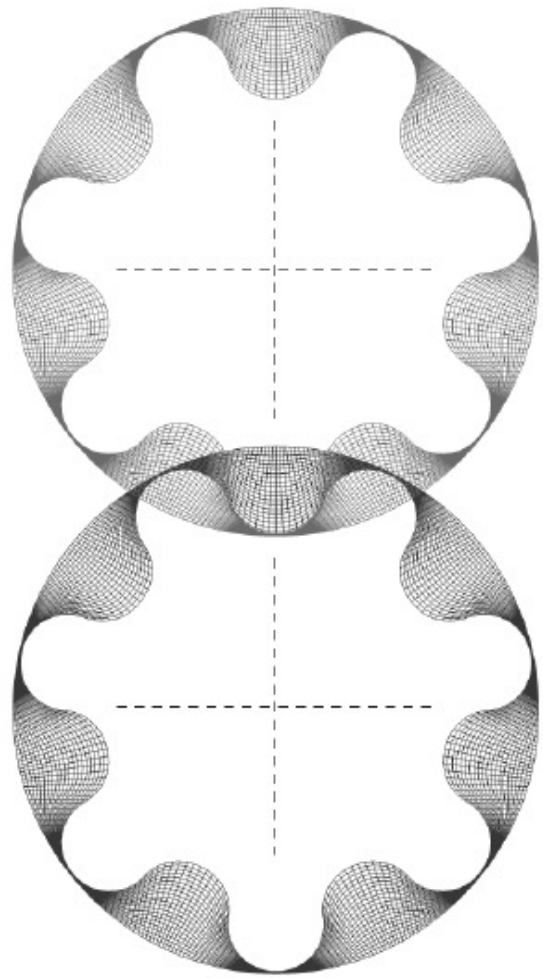
Overlapping (Chimera) grids use of a set of regular grids to cover irregular solution domains: rectangular, cylindrical, spherical or nonorthogonal grids can be combined near bodies with Cartesian grids in the rest of the solution domain.

Advantages:

- tt allows - without additional difficulty - calculation of flows around bodies which move relative to the environment or each other.

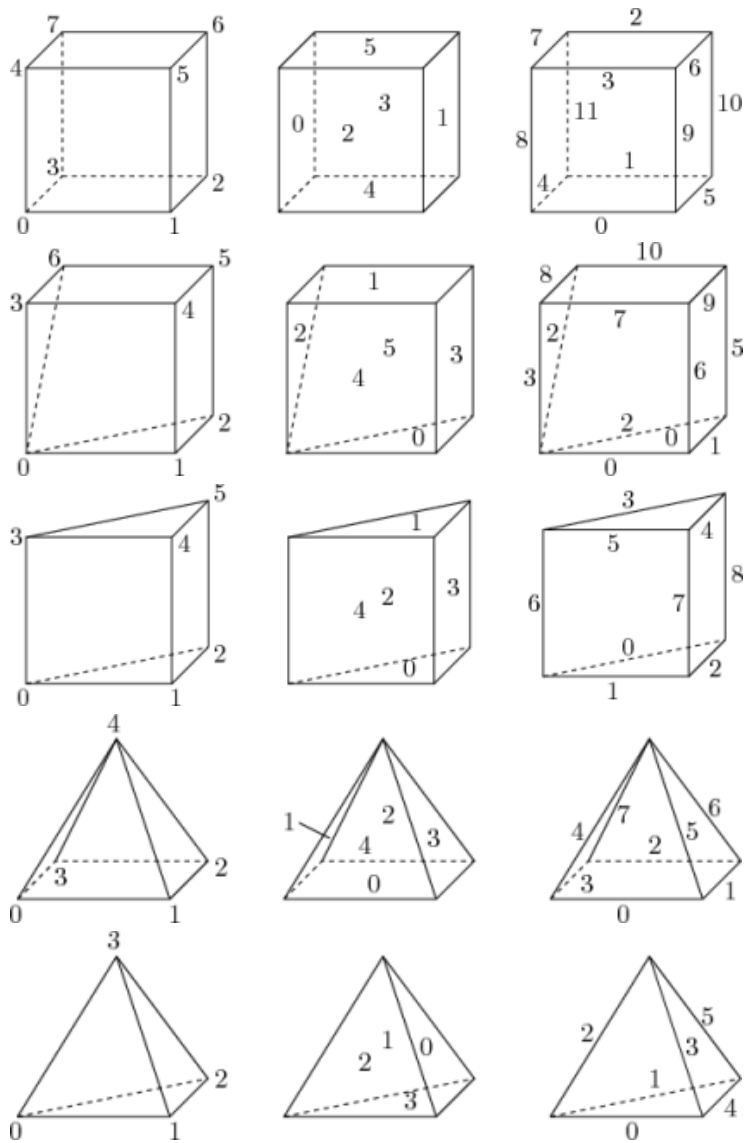
Disadvantages:

- the programming and coupling of the grids can be complicated. The computation is usually sequential; the solution method is applied on one grid after another, the interpolated solution from one grid providing the boundary conditions for the next iteration on adjacent grids;
- tt is difficult to maintain conservation at the interfaces, and the interpolation process may introduce errors or convergence problems if the solution exhibits strong variation near the interface.

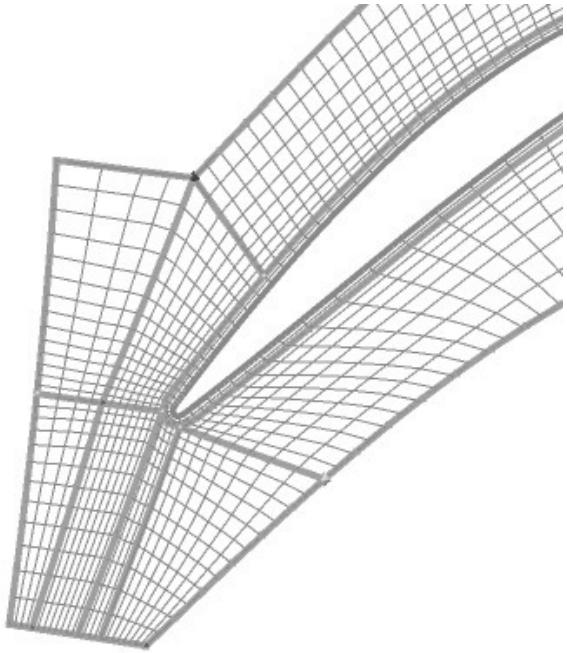


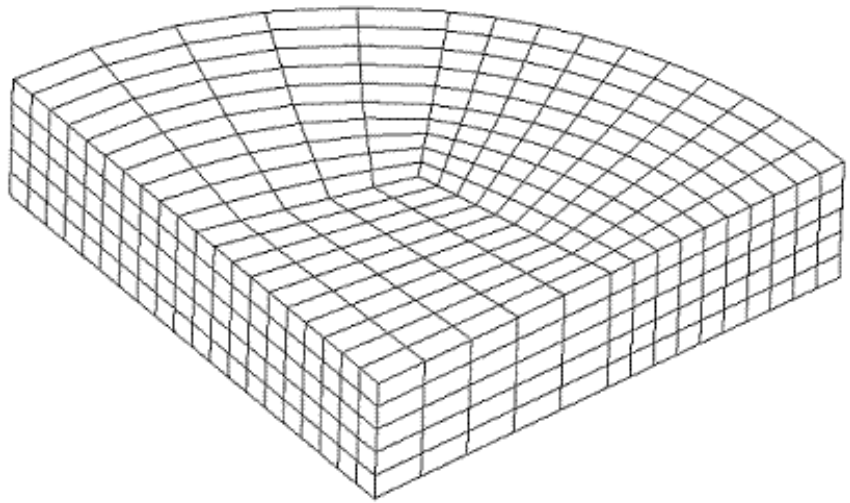
Grid generation

- **Cell topology is also important**: accuracy is usually higher if the CVs are hexaedra: the reason is that parts of the errors made at opposite cell faces when discretizing diffusion terms cancel partially (if cell faces are parallel and of equal area, they cancel completely) on quadrilateral (and hexahedral) CVs. To obtain the same accuracy on triangles and tetrahedra, more sophisticated interpolation and difference approximations must be used.
- Especially **near solid boundaries** it is desirable to have quadrilaterals or hexahedra, since all quantities vary substantially there and accuracy is especially important in this region
- the **ratio of the sizes of adjacent cells** in non-uniform grids should be kept under control, as accuracy is adversely affected if it is large, to limit the *truncation error*.



Grid generation



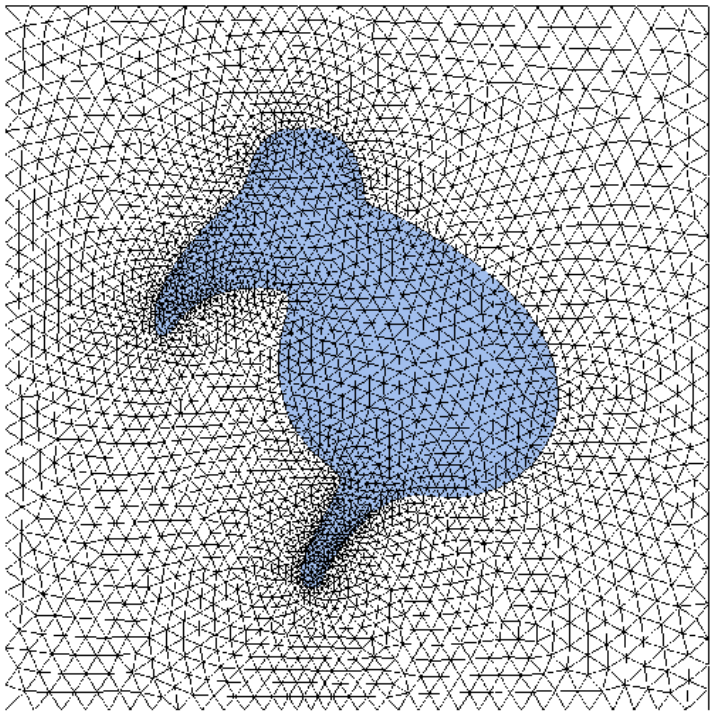


When the geometry is complex, grid generation usually consumes the largest amount of user time by far.

Since the accuracy of the solution depends as much (if not more) on the grid quality as on the approximations used for discretization of the equations, **grid optimization is a worthwhile investment of time**.

Grid generation

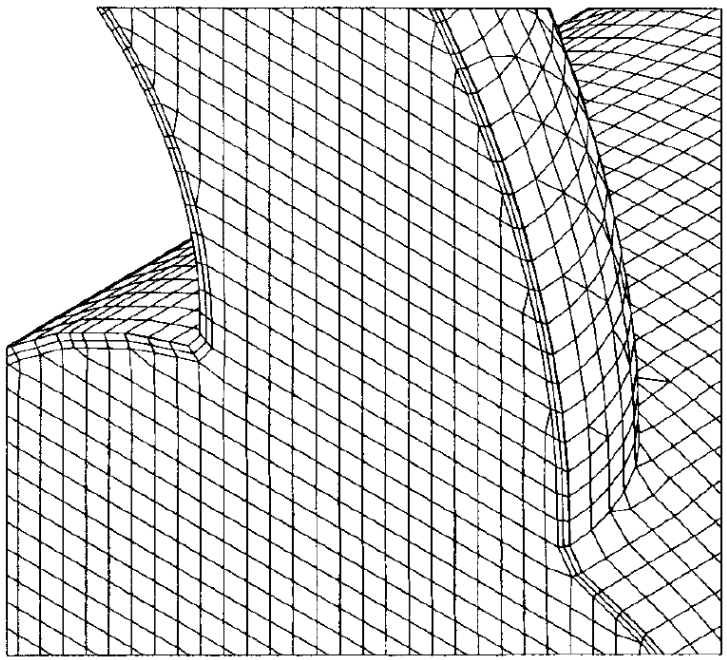
Tetrahedral cells are not desirable near walls if the boundary layer needs to be resolved because the first grid point must be very close to the wall while relatively large grid sizes can be used in the directions parallel to the wall.



- These requirements lead to long thin tetrahedra, creating problems in the approximation of diffusive fluxes.
- For this reason, some grid generation methods generate first a layer of prisms or hexahedra near solid boundaries, starting with a triangular or quadrilateral discretization of the surface; on top of this layer, a tetrahedral mesh is generated automatically in the remaining part of the domain.

Grid generation

Another approach to automatic grid generation is to **cover the solution domain with a (coarse) Cartesian grid, and adjust the cells cut by domain boundaries to fit the boundary**.

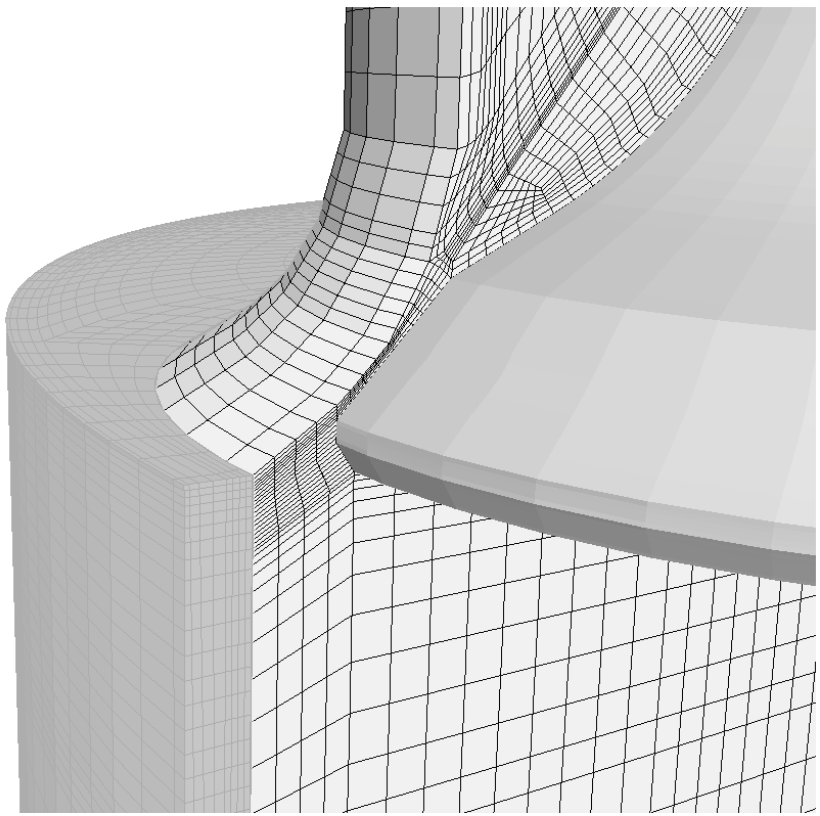


- The problem with this approach is that the cells near boundary are irregular and may require special treatment.
- However, if this is done on a very coarse level and the grid is then refined several times, the irregularity is limited to a few locations and will not affect the accuracy much but the degree of boundary irregularity is limited.
- In order to move the irregular cells further away from walls, one can first create a layer of regular prisms or hexahedra near walls; the outer regular grid is then cut by the surface of the near-wall cell layer.

This approach allows fast grid generation but requires a solver that can deal with the polyhedral cells created by cutting regular cells with an arbitrary surface. Again, all types of methods can be adapted to this type of grid.

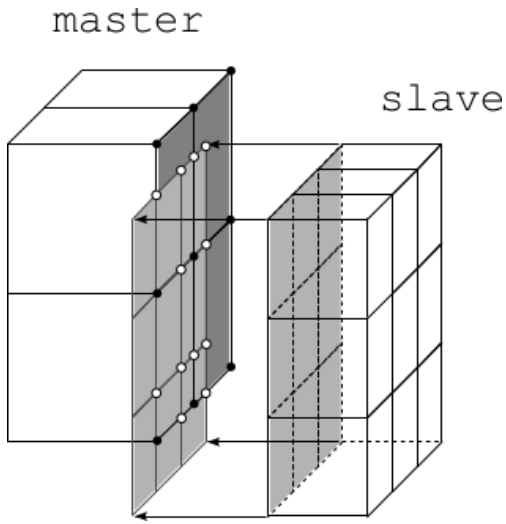
Hexa-block Meshing

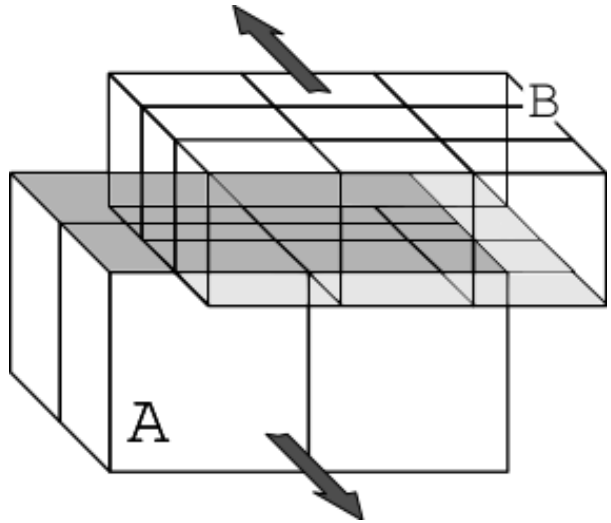
Non-conformal interfaces: if the solver can be applied on an unstructured grid with cells of varying topology/polyhedral cells, the grid generation program is subject to few constraints.

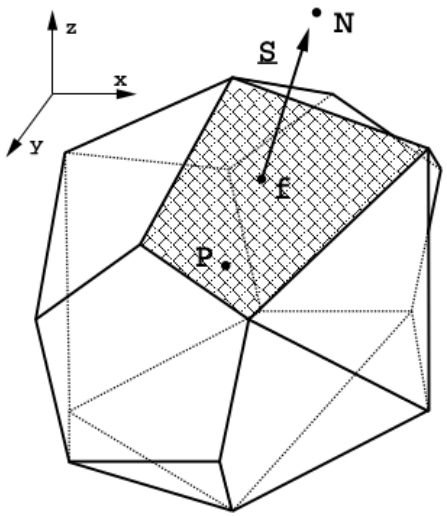


- The solution domain can first be divided into blocks which can be subdivided into grids with good properties; one has the freedom to choose the best grid topology (structured H-, 0-, or C-grid, or unstructured tetrahedral or hexahedral grid) for each block.
- The cells on the block interfaces then have faces of irregular shape and have to be treated as polyhedra.
- Local refined cells by subdivision of cells into smaller ones, although it retains their original shape (e.g. a hexahedron), **become a logical polyhedron**, since a face is replaced by a set of sub-faces.

Non-conformal grids

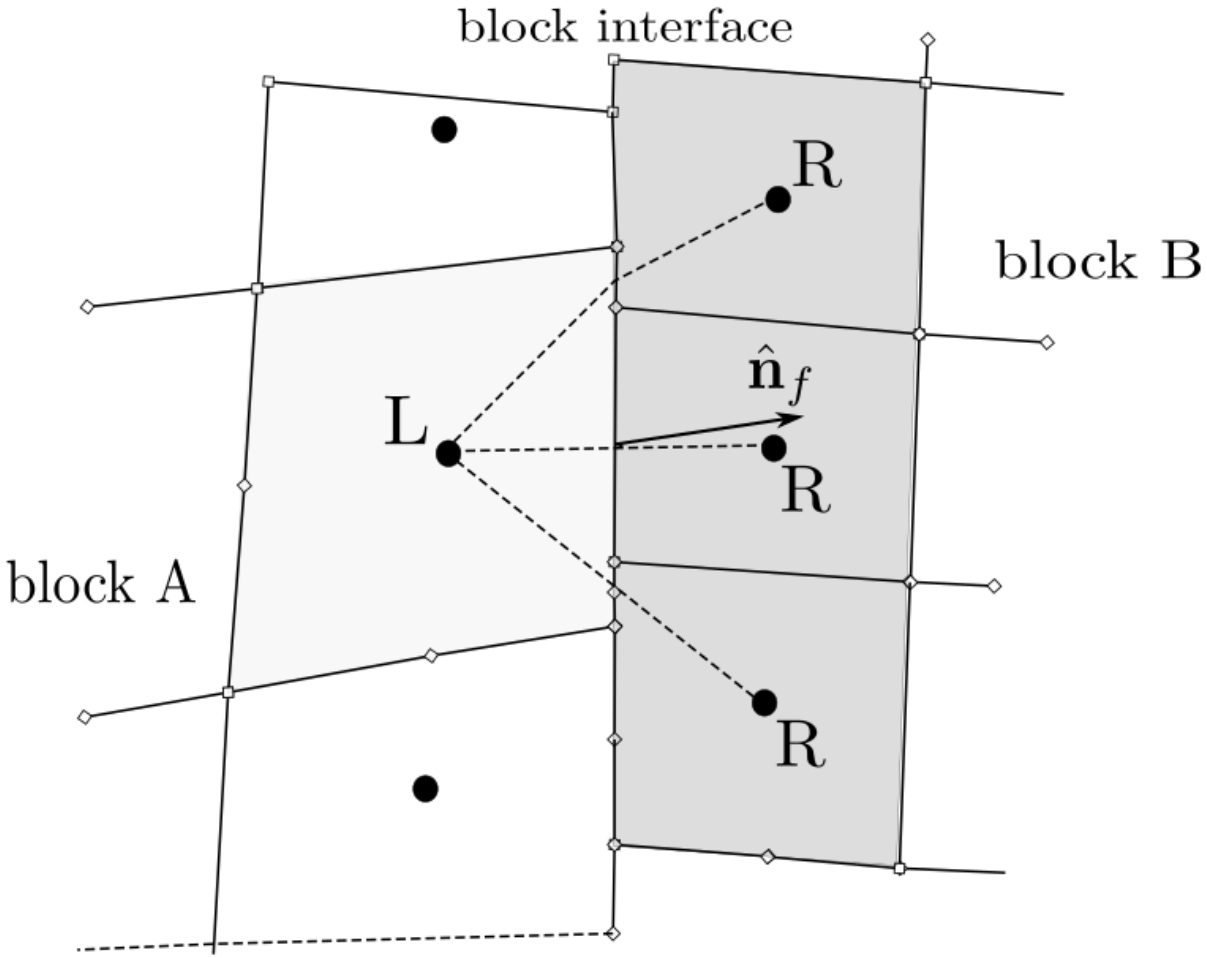






Local refined cells by subdivision of cells into smaller ones, although it retains their original shape (e.g. a hexahedron), **become a logical polyhedron**, since a face is replaced by a set of sub-faces.

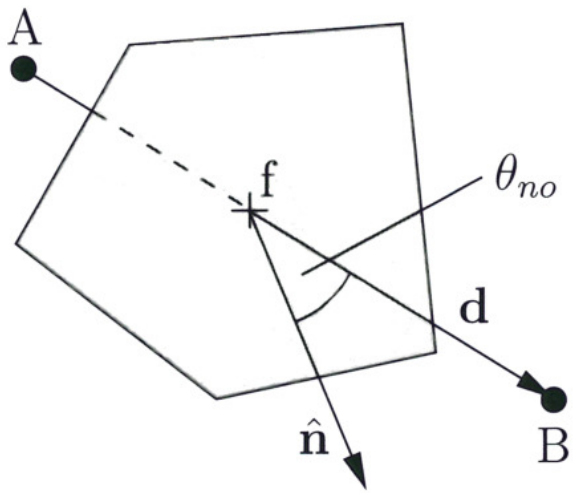
Grid generation: split-hexa grids



Mesh Quality and Numerics

Why is the mesh so important?

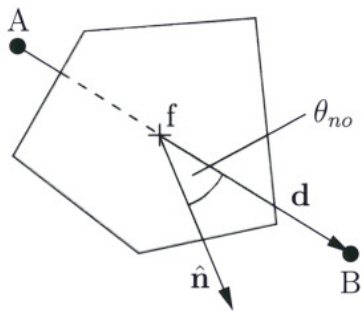
The generation of grids for complex geometries is a big issue; we shall present only some basic ideas and the properties that a grid should have.

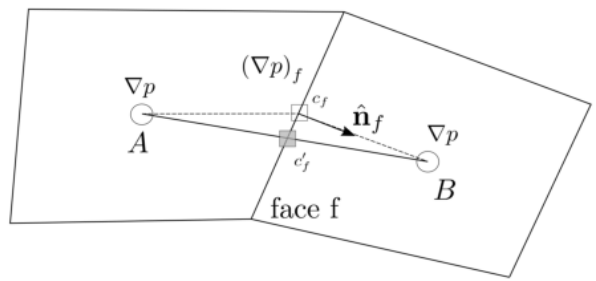


- Even though necessity demands that the grid be non-orthogonal, it is important to make it as nearly orthogonal as possible.
- **In FV methods** orthogonality of grid lines at CV vertices is unimportant - **it is the angle between the cell face surface normal vector and the line connecting the CV centers on either side of it that matters**.

Measures of Mesh Quality (Metrics)

The quality of the mesh plays a significant role in the accuracy and stability of the numerical computation. The attributes associated with mesh quality are node point distribution, orthogonality, and skewness.



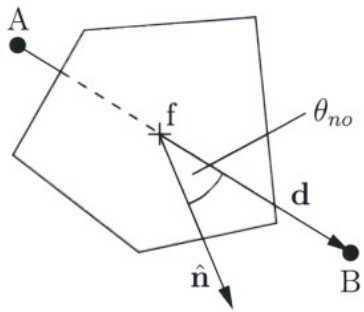


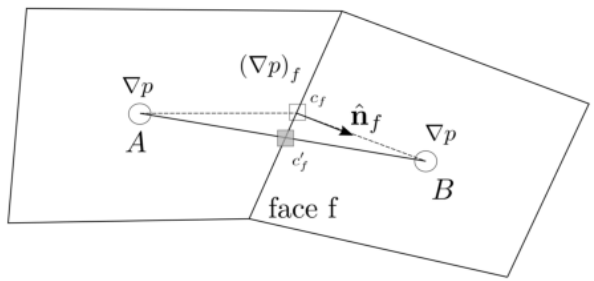
Depending on the cell types in the mesh (tetrahedral, hexahedral, polyhedral, etc.), different quality criteria are evaluated:

1) **Grid non-orthogonality** is measured by the angle θ_{no} between the vector $\vec{d}$ connecting the nodes adjacent to a face and the face area vector $\vec{S}$. The angle should be as small as possible.

Measures of Mesh Quality (Metrics)

The quality of the mesh plays a significant role in the accuracy and stability of the numerical computation. The attributes associated with mesh quality are node point distribution, orthogonality, and skewness.





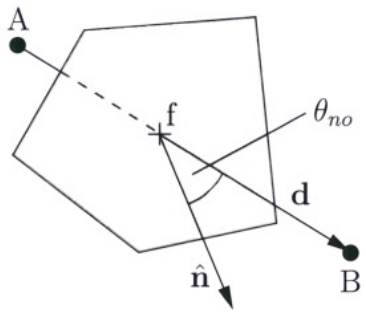
Depending on the cell types in the mesh (tetrahedral, hexahedral, polyhedral, etc.), different quality criteria are evaluated:

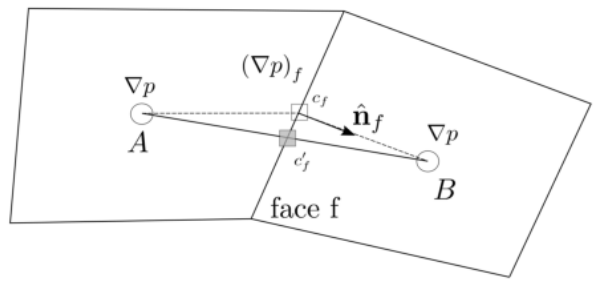
2) **Mesh skewness.** When the vector $\vec{d}$ does not intersect a face in its centre the mesh is defined as skewed. The degree of skewness can be measured by:

$$\psi = \frac{|\boldsymbol{d}_{AB}|}{|\boldsymbol{c_f} - \boldsymbol{c_{f'}}|}$$

Measures of Mesh Quality (Metrics)

The quality of the mesh plays a significant role in the accuracy and stability of the numerical computation. The attributes associated with mesh quality are node point distribution, orthogonality, and skewness.





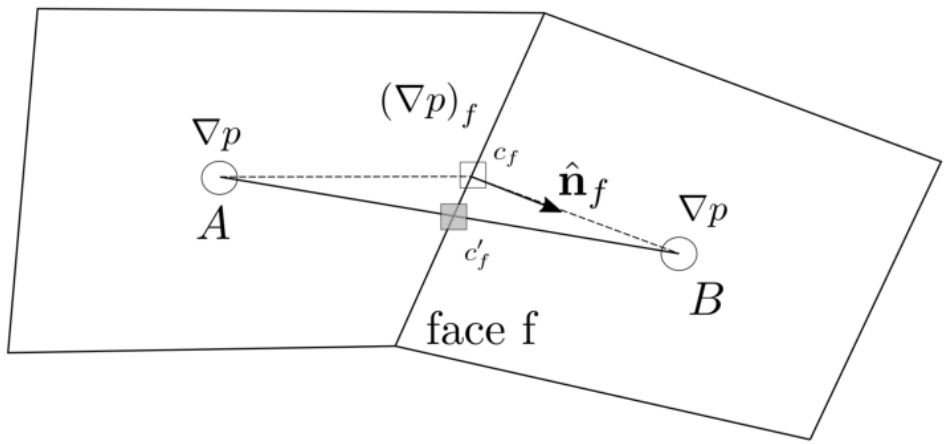
Depending on the cell types in the mesh (tetrahedral, hexahedral, polyhedral, etc.), different quality criteria are evaluated:

3) **Uniformity.** A mesh is uniform when $\vec{d}$ intersects the face midway between the nodes P and N. Uniformity can be measured by:

$$f_x = \frac{|\boldsymbol{x}_{\boldsymbol{f_i}} - \vec{\boldsymbol{x}_N}|}{|\boldsymbol{d}|}$$

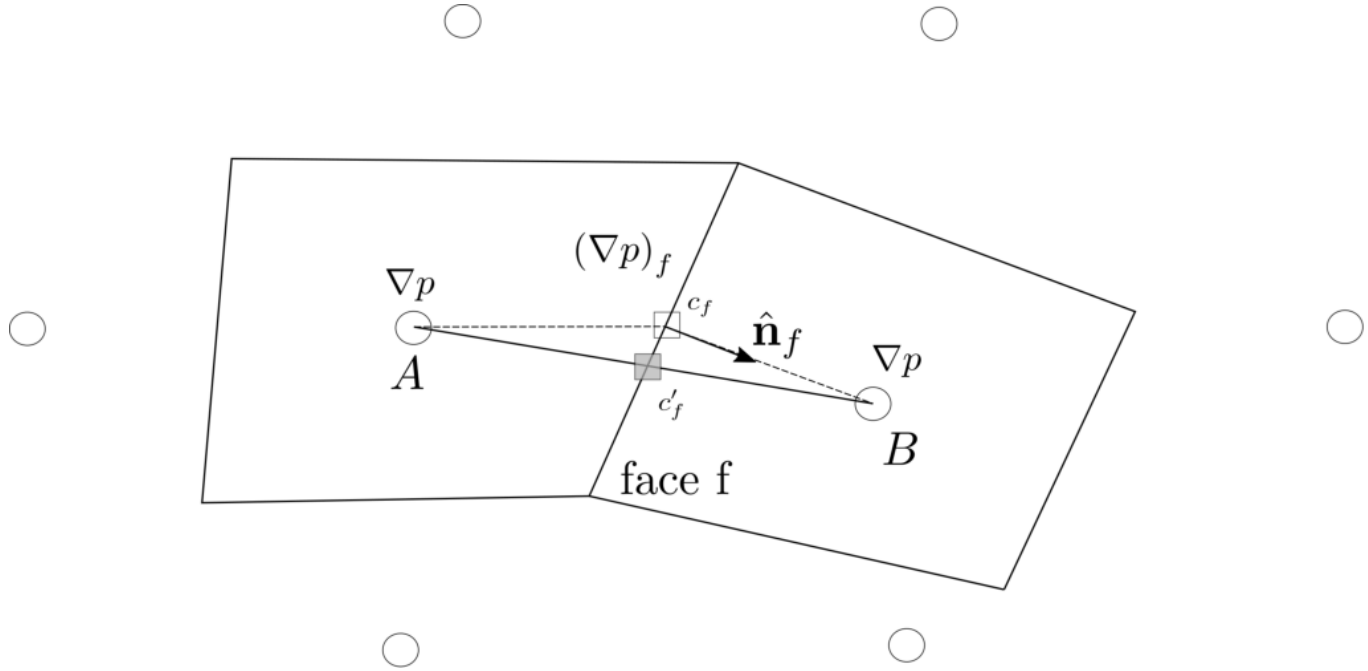
thus f_x=0.5 on uniform grids.

Skewness



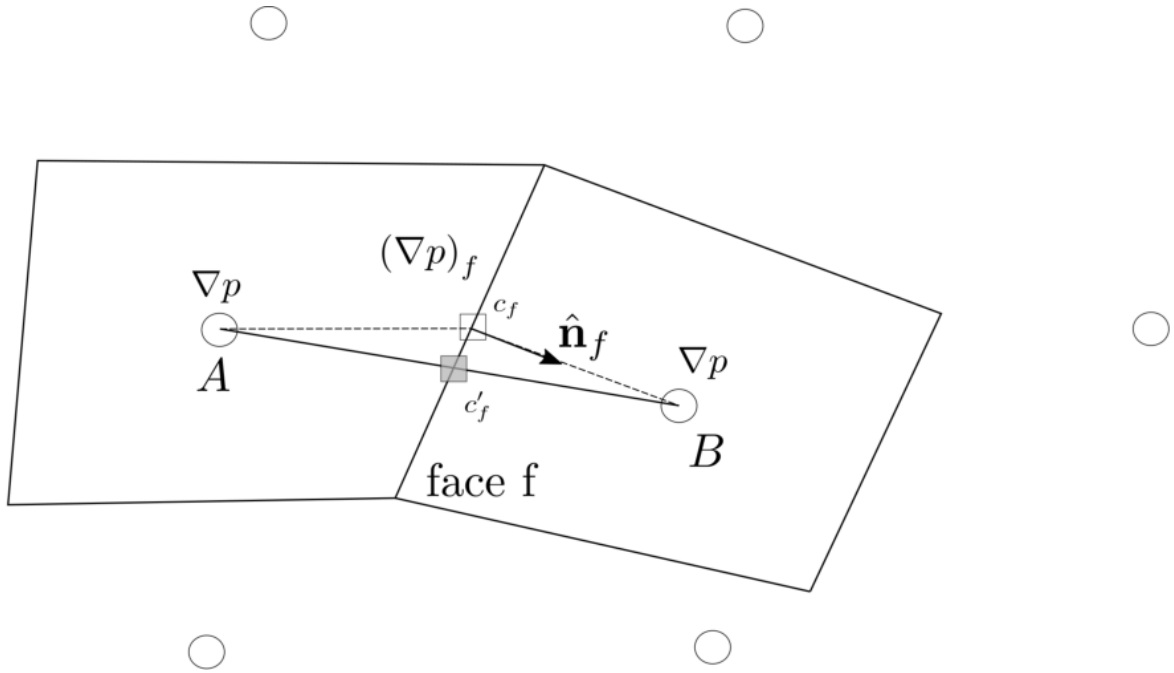
- To evaluate many of the terms constituting the general discretized equation of a quantity ϕ, it is necessary to estimate its value at element faces.
- An estimated value at the face should be the average value of the entire face (midpoint rule).
- At different steps of the discretization process, linear variation of variables between nodes is assumed. If this is extended to variation of variables along the face, then the average value of any variable ϕ should be found at the face centroid.

Skewness



- The common practice is to use a linear interpolation profile and to estimate the value at the face at the intersection between the face and the line connecting the two nodes straddling the face.
- **When the grid is skewed the line does not necessarily pass through the centroid of the face**: the intersection point of segment $\bar{AB}$ with the face f does not coincide with the face centroid.

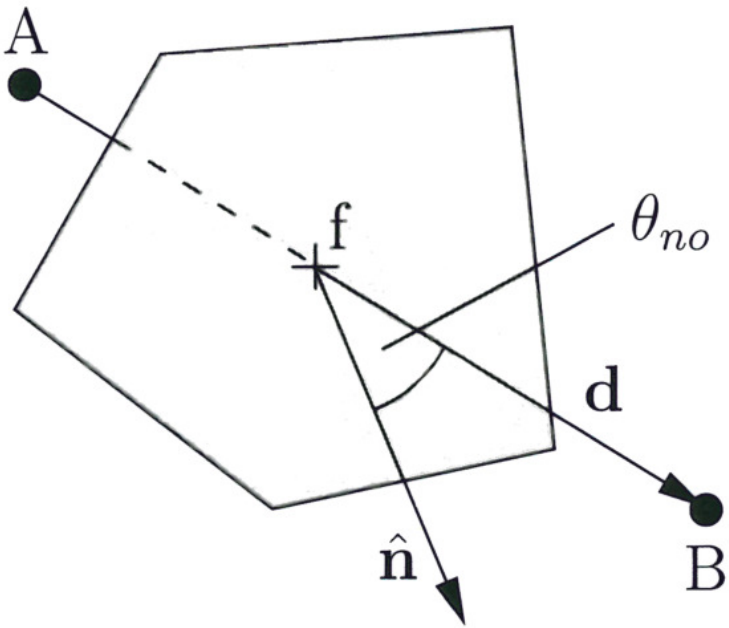
Skewness



- To keep the overall accuracy of the discretization method second order, all face integrations need to take place at point c_f. Thus a correction for the interpolated value at c'_f is needed in order to get the value at c_f.
- The skewness correction is derived by expressing the value of ϕ at the centroid of the face c_f in terms of its value and the value of its derivative at c'_f via a Taylor expansion:

$$\phi_f = \phi_{f'} + (\nabla\phi)_{f'} \cdot \boldsymbol{d}_{f'f}$$

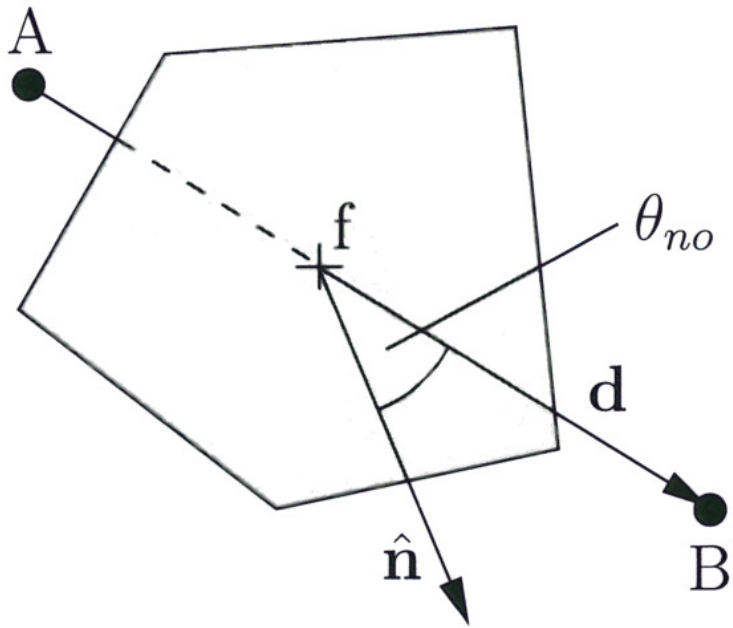
Non-orthogonality



- Angle between $\hat{\boldsymbol{d}}(\bar{AB})$ and $\hat{\boldsymbol{n}}_f$
- $\hat{\boldsymbol{n}}_f$: unit vector normal to face
- d: vector between cell centers A and B
- θ_{no}: angle between $\vec{d}$ and $\vec{n}$

- In general, structured curvilinear grids and unstructured grids are non-orthogonal. Therefore the surface vector $\boldsymbol{S}_f$ and the vector $\boldsymbol{d}(\bar{AB})$ joining the centroids of the elements straddling the interface are not collinear;
- in this case, the gradient normal to the surface cannot be written as a function of ϕ_A and ϕ_B, as it has a component in the direction perpendicular to AB.

Non-orthogonality



- Angle between $\boldsymbol{d}(\bar{AB})$ and $\hat{\boldsymbol{n}}_f$
- $\hat{\boldsymbol{n}}_f$: unit vector normal to face
- d: vector between cell centers A and B
- θ_{no}: angle between $\vec{d}$ and $\vec{n}$

Why non-orthogonality is important?

- Non-orthogonality affects Laplacian (diffusion) terms $\nabla \cdot (\Gamma \nabla p)$.
- error is not reduced by mesh refinement!
- **solution stability affected by worst face**

Error Analysis in FV CFD

Error Analysis in FV CFD

For a conservation equations of a given **scalar quantity**, whose integral form (**TRANSPORT EQUATION**) can be expressed as:

$$\underbrace{\underbrace{\frac{\partial}{\partial t}\int_\Omega \rho\phi d\Omega}_{\texttt{temporal derivative}} + \underbrace{\int_\Omega \rho\phi(\vec{u}-\vec{u_b})\cdot\vec{n}dS}_{\texttt{convection term}}}_{\frac{D\Phi}{Dt}} = \underbrace{\int_\Omega \Gamma grad\Phi\cdot\vec{n}dS}_{\texttt{diffusion term}} + \underbrace{\sum f_\Phi}_{\texttt{source/sink terms}}$$

The metrics of the computational mesh has a direct impact on the interpolation error of the spatial terms (skewness, mesh non-uniformity and non-orthogonality) of the mesh. In particular:

- mesh skewness and non-uniformity have a direct impact on the truncation error in the interpolation practice of the face-centered quantities;
- non-orthogonality affects the accuracy of the calculation of the laplacian operators.

Truncation Error in Interpolation Practice

A second-order interpolation practice from the nodes onto internal faces can be written as follows:

$$\underline{\phi_f} = \underline{\phi_{f_i}} + \boldsymbol{m} \cdot \underline{(\nabla\phi)_{f_i}}$$

IMPORTANT NOTES:

1) $\overline{\phi}_f$ and $\overline{(\nabla\phi)}_{f_i}$ WOULD BE the interpolated values of the transported scalar over the cell faces at the point where the vector $\boldsymbol{d}$ intersects the face.

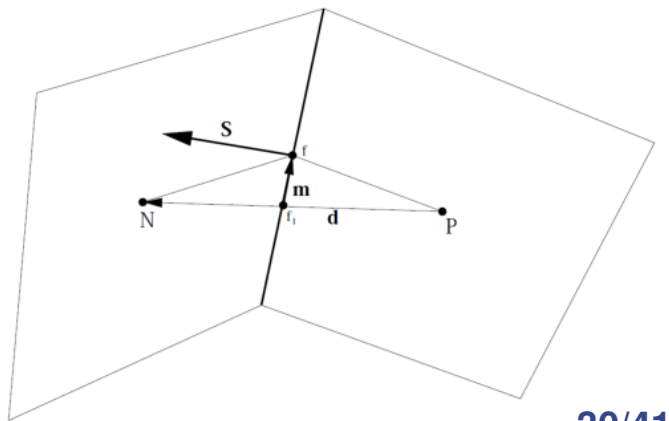
2) $\overline{\phi_f}$ and $\overline{(\nabla\phi)}_{f_i}$ are **EVALUATED** using an interpolation scheme (e.g. linear):

$$\bar{\phi}_f = f_x\phi_P + (1 - f_x)\phi_N$$

$$(\nabla\bar{\phi})_{f_i} = f_x(\nabla\bar{\phi})_P + (1 - f_x)(\nabla\bar{\phi})_N$$

The *linear* interpolation factor f_x is defined as:

$$f_x = \frac{|\boldsymbol{x}_{f_i} - \boldsymbol{x}_N|}{|\boldsymbol{d}|}$$



Truncation Error in Interpolation Practice

The **truncation error** for the interpolation practice defined for

$$\bar{\phi}_f = f_x \phi_P + (1 - f_x)\phi_N \tag{1}$$

can be estimated by using the following Taylor expansions:

$$\phi_P = \phi_{f_i} + (\boldsymbol{x}_P - \boldsymbol{x}_{f_i}) \cdot (\nabla\phi)_{f_i} + \frac{1}{2}(\boldsymbol{x}_P - \boldsymbol{x}_{f_i})^2 : (\nabla\nabla\phi)_{f_i}$$

$$\phi_N = \phi_{f_i} + (\boldsymbol{x}_N - \boldsymbol{x}_{f_i}) \cdot (\nabla\phi)_{f_i} + \frac{1}{2}(\boldsymbol{x}_N - \boldsymbol{x}_{f_i})^2 : (\nabla\nabla\phi)_{f_i}$$

$$\phi_f = \phi_{f_i} + \boldsymbol{m} (\nabla\phi)_{f_i} + \frac{1}{2}\boldsymbol{m}^2 : (\nabla\nabla\phi)_{f_i}$$

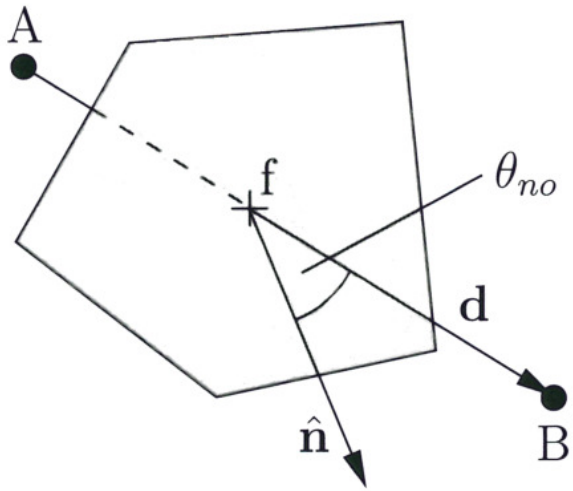
By substituting ϕ_P and ϕ_N in Eqn. (1) respectively, the **truncation error for linear interpolation from Eqn.** (1) is:

$$\begin{aligned} e_l &= \phi_{f_i} - \underline{\phi_{f_i}} \\ &= -\frac{1}{2}|\boldsymbol{x}_P - \boldsymbol{x}_{f_i}||\boldsymbol{x}_N - \boldsymbol{x}_{f_i}| \left(\hat{\boldsymbol{d}} : (\nabla\nabla\phi)_{f_i}\right) \\ &= \frac{1}{2} f_x (1 - f_x)|\boldsymbol{d}|^2 (\hat{\boldsymbol{d}}^2 : (\nabla\nabla\phi)_{f_i}) \end{aligned}$$

Truncation Error in Interpolation Practice

The **truncation error in interpolation practice** is present any time an interpolation is performed ($\rightarrow$ see the calculation of gradient and divergence operators!) and it **occurs with ANY spatial discretization scheme adopted** (linear, Gamma, upwind ...).

Non-Orthogonality



Non-orthogonality affects Laplacian (diffusion) terms (e.g. $\nabla \cdot \nabla p$):

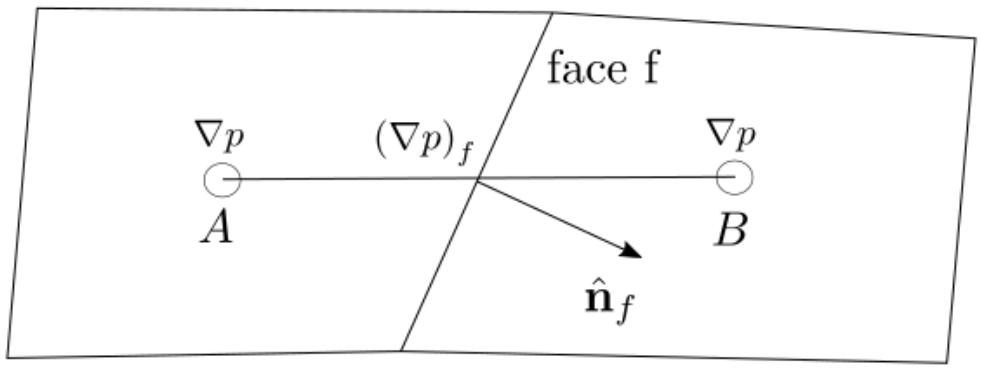
$$\int_V \nabla \cdot (\Gamma \nabla p)\, dV = \int_S (\Gamma \nabla p) \cdot \boldsymbol{n} dS \approx \sum_f \Gamma_f \boldsymbol{S}_f \cdot (\nabla p)_f = \sum_f \Gamma_f |\boldsymbol{S}_f| \left(\frac{\partial p}{\partial n}\right)_f$$

Gauss integration — Sum over faces — `interpolate` — `snGrad`

Example : `Gauss` `linear` `corrected`

NOTE: Γ_f is typically calculated by a `linear interpolate` to the face.

Surface Gradient



- Gradient field ∇p can be calculated (cell centres)
- Full gradient at face $(\nabla p)_f$ can be interpolated from (∇p)
- Surface normal component of $(\nabla p)_f$ is:

$$\left(\frac{\partial p}{\partial n}\right)_f = \hat{\boldsymbol{n}}_f \cdot (\nabla p)_f = \frac{\phi_B - \phi_A}{||\boldsymbol{r}_B - \boldsymbol{r}_A||} = \frac{\phi_B - \phi_A}{d_{AB}}$$

Non-orthogonality

The diffusive flux of the variable ϕ through an internal cell face f is approximated as

$$D_f = \int_{S_f} \Gamma_\phi \nabla\phi \cdot d\boldsymbol{s} \simeq \Gamma_\phi (\nabla\phi)_f^* \cdot \hat{\boldsymbol{n}}_f S$$

and the gradient $(\nabla\phi)_f^*$ is constructed as follows:

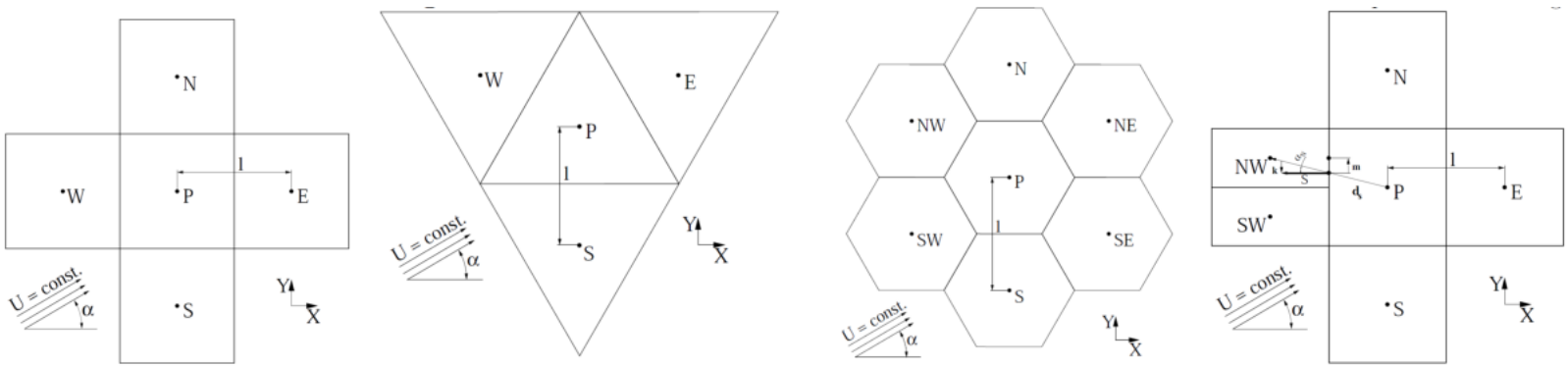
$$(\nabla\phi)_f^* = (\nabla\phi)_f + \underbrace{\left[\frac{\phi_N - \phi_P}{|\hat{\boldsymbol{d}}|} - \frac{(\nabla\phi)_f \cdot \hat{\boldsymbol{d}}}{|\hat{\boldsymbol{d}}|}\right]\frac{\hat{\boldsymbol{d}}}{|\hat{\boldsymbol{d}}|}}_{\texttt{correction}}$$

- The correction term in the equation represents the difference between the central difference approximation of the derivative in the direction of vector $\hat{\boldsymbol{d}}$ and the corresponding value obtained by interpolating the cell-center gradients;
- This correction term detects and smoothes out any unphysical oscillations that might occur in the iteration process.

Discretization Error vs Grid Topology

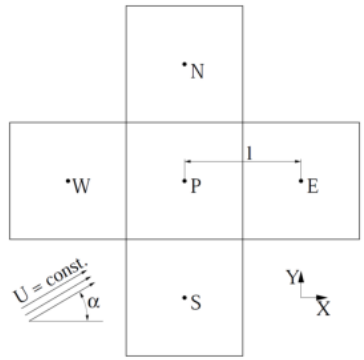
Discretization Error vs Grid Topology

Discretisation errors are dependent both on the type of computational mesh (i.e. tetrahedral, hexahedral, polyhedral) and the approximations for different terms (convection and diffusion terms mostly) in the governing equations.

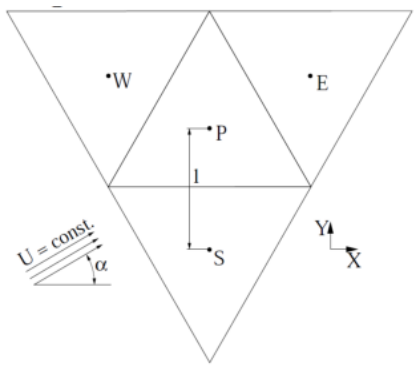


NOTE: discretisation procedures for divergence and gradient terms are very similar to the discretisation procedure for the convection term and everything that will be said about error for the convection term can be applied to them.

Discretization Error vs Grid Topology



Square mesh. This mesh is orthogonal, it is not skewed and f_x=0.5 because the centre of each internal face lies midway between the neighbouring nodes.

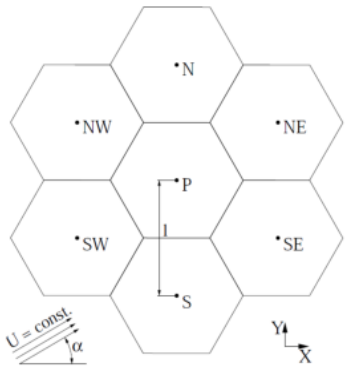


Triangular mesh. A mesh consisting of equilateral triangles is uniform, orthogonal and not skewed. This mesh is orthogonal, it is not skewed and f_x=0.5 because the centre of each internal face lies midway between the neighbouring nodes.

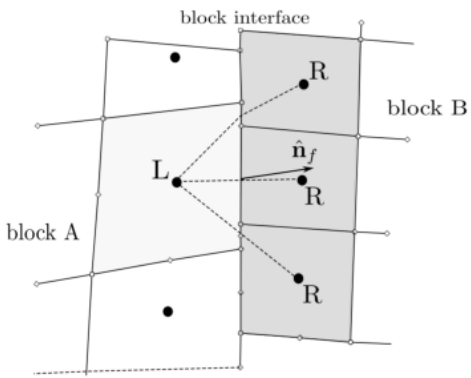
Discretization Error vs Grid Topology

Square and Triangular mesh

Regular hexagonal mesh. This type of mesh can be generated using a Delaunay algorithm. It consists of hexagons and is uniform, orthogonal and not skewed.



Split-hexahedron mesh. This type of mesh is produced when local refinement of hexahedral cells is done by splitting cells, such that a cell face gets divided into two or four faces.

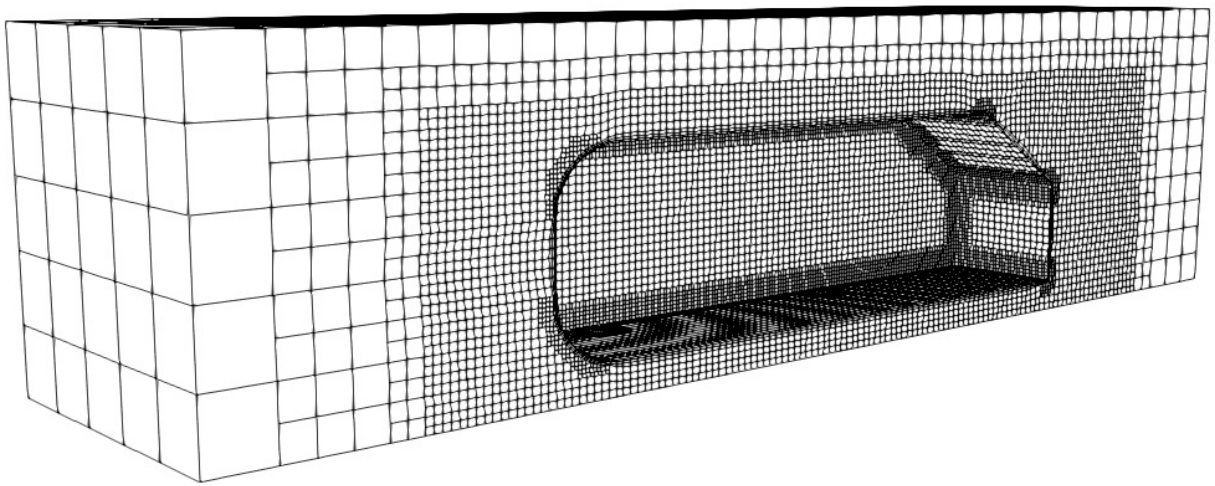


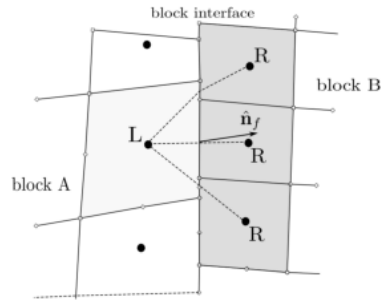
$$\boldsymbol{d}_s = l_s \hat{\boldsymbol{d}} = \frac{l}{cos\alpha_N} \hat{\boldsymbol{d}}$$

$$\tan(\theta_{no}) = \frac{0.25\, l}{l} = \frac{1}{4}$$

Unlike the previous examples, a split-hexahedron mesh is skewed and not orthogonal but $f_x = 0.5$. The distances between nodes at the split face are also larger than on other faces.

Split Hexahedron Mesh





Split-hexahedron mesh. This type of mesh is produced when local refi ment of hexahedral cells is done by splitting cells, such that a cell face gets divided into two or four faces. Unlike the previous examples, a split-hexahedron mesh is skewed and not orthogonal but $f_x = 0.5$. The distances between nodes at the split face are also larger than on other faces. Thus:

$$\boldsymbol{d}_s = l_s \hat{\boldsymbol{d}} = \frac{l}{cos\alpha_N} \hat{\boldsymbol{d}}$$

where

$$\tan(\theta_{no}) = \frac{0.25\, l}{l} = \frac{1}{4}$$

Thank you for your attention!

contact: federico.piscaglia@polimi.it