FINITE VOLUME METHOD

Enrico Nobile

Dipartimento di Ingegneria e Architettura
Università degli Studi di Trieste, 34127 TRIESTE

April 7, 2025

OUTLINE

Part I

Generalities

Finite Volume Method for Incompressible fluids

- Cartesian grids for two-dimensional (2D) domains:
  - Generic conservation equation
  - Complete fluid flow-thermal problem
  - Modifications required for Cartesian grids in three-dimensional (3D) domains
  - One-dimensional (1D) case: elementary derivation left to the reader
- Unstructured grids:
  - Generic conservation equation
  - Geometric quantities and variables arrangement on the grid
  - Gradient reconstruction
  - Hybrid grids

Finite Volume Method (FVM)

- Algebraic equations obtained by enforcing conservation for each Control Volume (CV) or Cell;
- Popular method in Computational Fluid Dynamics and Numerical Heat Transfer (CFD);
- Used in several commercial and open source CFD packages;
- Simplicity and physical correspondence between FVM and conservation principles;
- Frequently adopted, also at undergraduate level, for the introduction of numerical techniques in Heat Transfer and fluid dynamics.

Comparison of FV and FD

Finite Difference method	Finite Volume method
Non inherently conservative	Inherently Conservative
Close to the equations	Physically meaningful
Simple geometries (in standard form)	Suitable for complex geometries
Derivation and interpolation	Integration and deconvolution
Higher order derivatives	Lower order derivatives
High order methods	High order difficult

All conservation (transport) equations, with the exception of the mass conservation equation, have a similar structure:

- Energy equation in conservative form:

$$\frac{\partial}{\partial \vartheta}(\rho c_p t) + \nabla \cdot (\rho \mathbf{w} c_p t) = \nabla \cdot (\lambda \nabla t) + \dot{q}$$

- Momentum equation in conservative form:

$$\frac{\partial}{\partial \vartheta}(\rho \mathbf{w}) + \nabla \cdot (\rho \mathbf{w}\mathbf{w}) = -\nabla p + \nabla \cdot (\mu \nabla \mathbf{w}) - \rho\,\beta\,(t - t_0)\,\mathbf{g}$$

General form of transport equation:

$$\frac{\partial}{\partial \vartheta}(\rho \phi) + \nabla \cdot (\rho \mathbf{w} \phi) = \nabla \cdot (\Gamma \nabla \phi) + s$$

ϕ is a generic scalar variable and Γ is the molecular transport property.

In other words:

(Accumulation) + (Convection) = (Diffusion) + (Source/Sink)

- Energy equation: for incompressible fluids with constant c, ϕ is the specific enthalpy ($c\,t$), Γ is λ/c, and the source term, s, includes the possible internal heat generation.
- Momentum equation: ϕ is the velocity component, Γ is the molecular dynamic viscosity μ and the source term includes, besides the component of the pressure gradient, also the buoyancy force, is present, and all other possible force fields.
- It will be sufficient to consider first the generic transport equation, and then the Navier-Stokes and continuity equations:
  - The velocity field $\mathbf{w}$, the thermophysical properties ρ and Γ, and the source term s will be assumed known.
  - The procedure described maintains its validity even when the velocity field is unknown, like in the case of the Navier-Stokes equations.

Use of the integral, or *finite*, formulation of the conservation equation, written for a generic *control volume* V:

$$\int_V \left[\frac{\partial}{\partial \vartheta} (\rho\phi) + \nabla \cdot (\rho \mathbf{w} \phi) - \nabla \cdot (\Gamma \nabla \phi) - s \right] \mathrm{d}V = 0$$

Applying Gauss's theorem (also known as Ostrogradsky's theorem,), with A *surface boundary* of the control volume V:

$$\int_V \frac{\partial}{\partial \vartheta} (\rho\phi)\, \mathrm{d}V + \int_A \rho\phi \mathbf{w} \cdot \mathbf{n}\, \mathrm{d}A = \int_A \Gamma \nabla\phi \cdot \mathbf{n}\, \mathrm{d}A + \int_V s\, \mathrm{d}V$$

ed in a more compact form:

$$\int_V \frac{\partial}{\partial \vartheta} (\rho\phi)\, \mathrm{d}V + \int_A \mathbf{J}'' \cdot \mathbf{n}\, \mathrm{d}A = \int_V s\, \mathrm{d}V$$

With $\mathbf{J}''$ we indicate the *flux density* vector of the variable ϕ. It includes both the convective flux density, $\mathbf{J}''_c$, and the diffusive flux density, $\mathbf{J}''_d$. Therefore, $J'' = \mathbf{J}'' \cdot \mathbf{n}$ represents the flux density component normal to the surface:

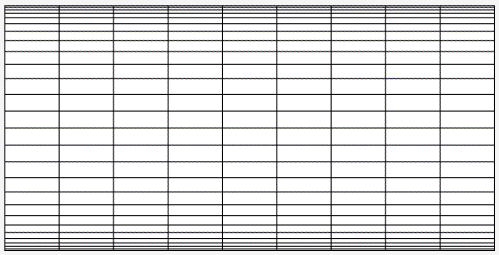
$$\mathbf{J}'' = \mathbf{J}''_c - \mathbf{J}''_d = \rho\phi\mathbf{w} - \Gamma\nabla\phi$$

$$J'' = \mathbf{J}'' \cdot \mathbf{n} = \rho\phi\mathbf{w} \cdot \mathbf{n} - \Gamma\nabla\phi \cdot \mathbf{n}$$

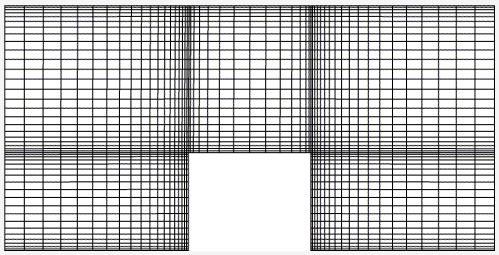
Structured Cartesian grids

- Constituted by families of, mutually orthogonal, parallel lines.
- The lines of each family, or the *cells* defined by these lines, are identified by a set of two indices (i, j) in 2D, or three indices (i, j, k) in 3D.
- Limited geometrical flexibility.
- Simplicity and efficiency of computing methods based on such grids.

Examples of structured Cartesian grids:



Single-block

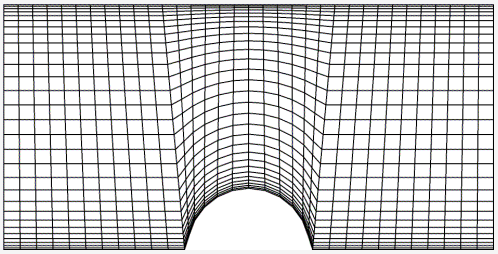


Multi-block

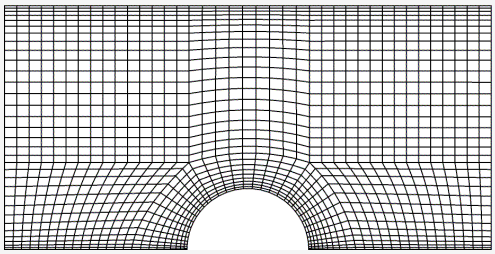
Structured curvilinear (boundary-fitted) grids

- Constituted by families of curvilinear lines, such that each line of a family will never intersect a line of the same family, and will cross only once the lines of the other families.
- They are identical to the Cartesian grids from the logical - e.g. indexing, data-structure - point of view.
- Greater geometrical flexibility compared to Cartesian grids.
- Special case: the families of the lines are mutually orthogonal.

Examples of structured curvilinear grids:



Single-block

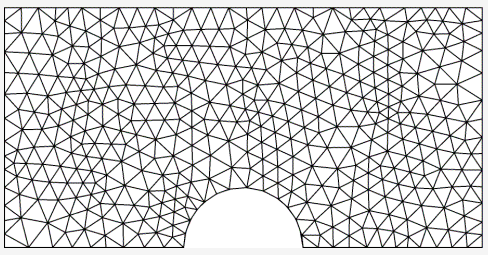


Multi-block

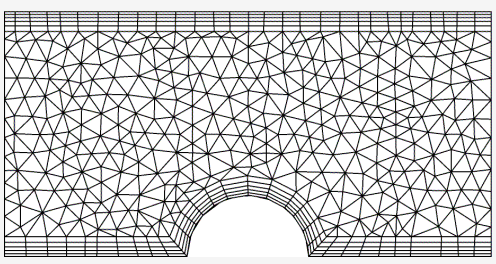
Unstructured grids

- Particularly suited to address complex geometries of industrial interest.
- The spatial domain is discretized into cells of arbitrary shape, e.g. *polyhedra*, or more typically triangles and quadrilaterals in 2D, and tetrahedra and hexaedra in 3D.
- Possibility to refine the grid, also automatically (*adaptive grids*), in specific zones of the domain.
- Greater complexity and computational costs, e.g. data-structures, memory requirements, limited choice of linear solvers.

Examples of unstructured grids:



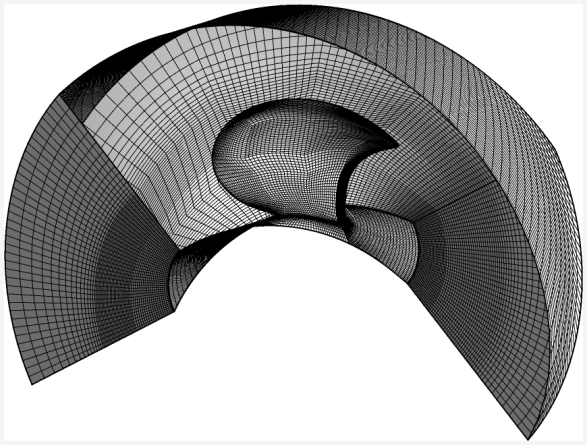
Triangle-based grid

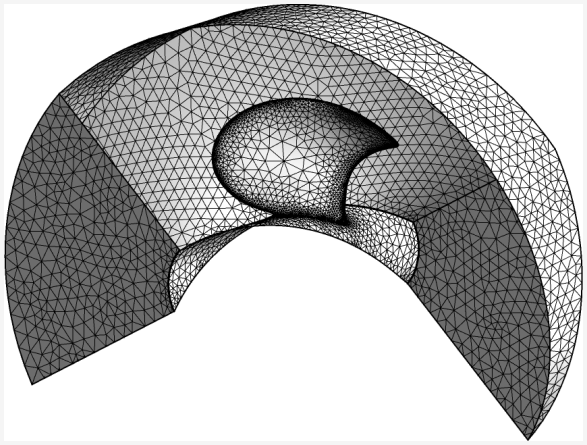


Hybrid grid

Some examples

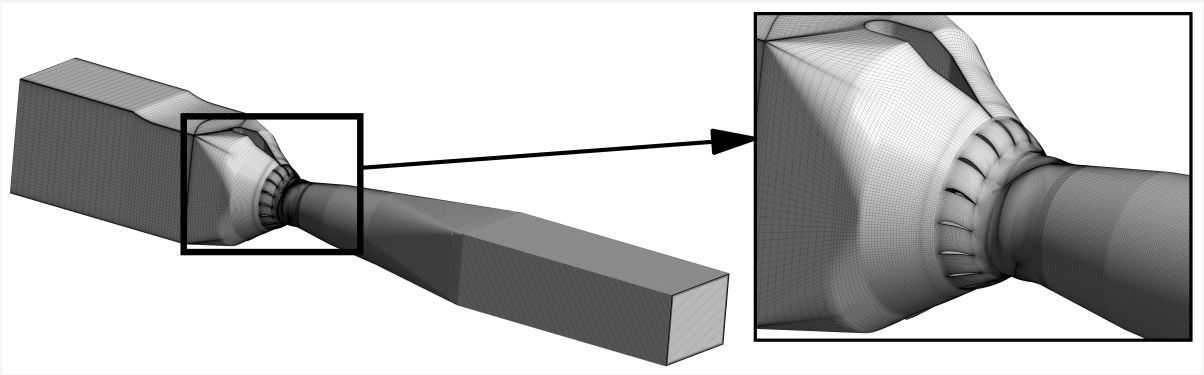
Hexa and hybrid meshing for a naval propeller (M. Morgut, 2009).





Some examples - cont.

Hexa meshing for a bulb turbine (ACCUSIM project, www.accusim.eu)

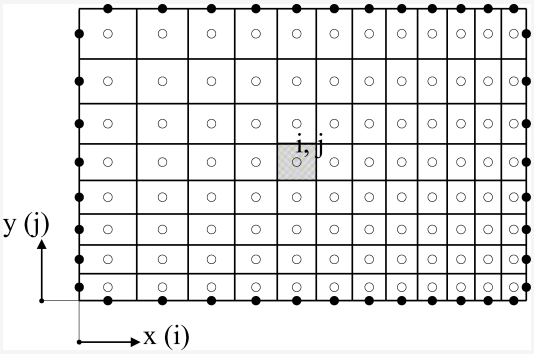


Some examples - cont.

Mixed – hybrid-hexa – meshing for a Kaplan turbine (ACCUSIM project, www.accusim.eu)

Cartesian grids for the finite volume method

- The physical domain is subdivided in a finite number of control volumes (CV), ad illustrated in the figure.
- The grid, differently from the *Finite Difference* method, defines the *faces*, and not the *nodes*, of the grid, although the variables are still collocated in the nodes.
- The most common approach, which is also adopted for cell-centered FV unstructured grids, is to place the nodes in the centroids of the CVs.
- The nodes at the centre of the CVs are indicated with '○', while the nodes at the boundary are represented by '●'.



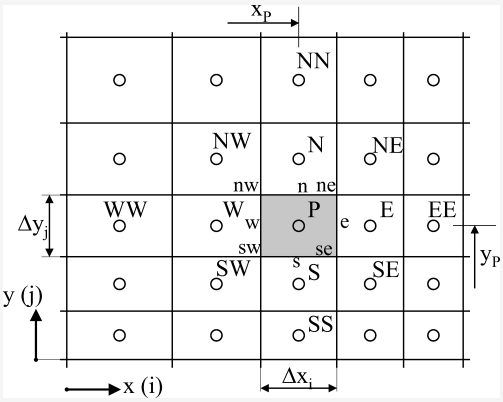
OUTLINE

Part II

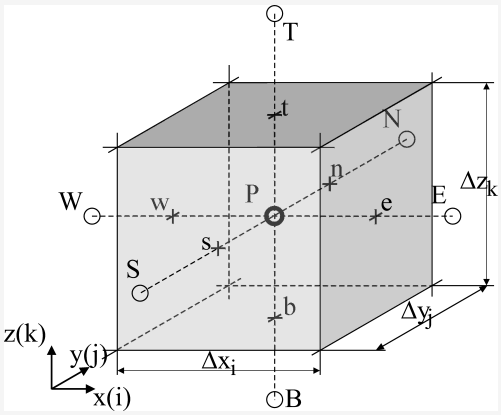
Fundamentals

Notation

Typical CVs for a Cartesian grid, together with the adopted nomenclature:



Two-dimensional



Three-dimensional

- The 2D case can be seen as a special 3D case where all variables are independent of the z coordinate.
- Attention paid to the 2D case: simple extension, for Cartesian grids, to the 3D case.

Spatial variation approximation

- As will be seen later, fluxes at the faces and sources within the cell are evaluated using the *mean value approach*, i.e. using the value of ϕ at the centroid of the surface (midpoint rule) and cell, respectively.
- This approach, in addition to the *assumed* variation of ϕ in space around point P, determine the accuracy of the entire discretization procedure.
- In the *standard* Finite Volume method, ϕ is assumed to vary linearly in space, i.e.

$$\boxed{\phi(\mathbf{x}) = \phi_P + (\mathbf{x} - \mathbf{x}_P) \cdot (\nabla\phi)_P \quad \textit{where} \quad \phi_P = \phi(\mathbf{x}_P)}$$

- ϕ_P represents the mean value of ϕ within the cell

$$\boxed{\phi_P = \overline{\phi} = \frac{1}{V}\int_V \phi(\mathbf{x})\,\mathrm{d}V}$$

Spatial variation approximation - *cont.*

- However, the spatial variation of ϕ within an element can be described via a Taylor series expansion around $\mathbf{x}_P$ as

$$\boxed{\begin{aligned}\phi(\mathbf{x}) = \phi_P &+ (\mathbf{x} - \mathbf{x}_P) \cdot (\nabla\phi)_P + \frac{1}{2}(\mathbf{x} - \mathbf{x}_P)^2 : (\nabla\nabla\phi)_P \\ &+ \frac{1}{3!}(\mathbf{x} - \mathbf{x}_P)^3 :: (\nabla\nabla\nabla\phi)_P + \ldots\ldots \\ &+ \frac{1}{n!}(\mathbf{x} - \mathbf{x}_P)^n \underbrace{:: \ldots ::}_{\text{(n-1) times}} \left(\underbrace{\nabla\nabla\ldots\nabla\phi}_{\text{n times}}\right)_P + \ldots\ldots\end{aligned}}$$

- The term $(\mathbf{x} - \mathbf{x}_P)^n$ in the equation represents the nth tensorial product of the vector $(\mathbf{x} - \mathbf{x}_P)$ with itself, producing an nth tensor.
- The operator $(:)$ is the inner product of two *2*nd rank tensors, $(::)$ is the inner product of two *3*rd rank tensors, and $(\underbrace{:::: \ldots ::}_{\text{(n-1) times}})$ is the inner product of two nth rank tensors, all yielding a *scalar*.
- The comparison between this Taylor series expansion, and the previous assumed variation of ϕ within the cell, indicates an error proportional to $\left|(\mathbf{x} - \mathbf{x}_P)^2\right|$, implying a *second order* accuracy.

Global conservation

The conservation equation, written in integral form, can be applied to any control volume, and in particular to every FV (cell) or to the entire domain:

- Summing up the equations obtained for all FVs, we are left with the global conservation equation, since fluxes over internal faces will cancel out: they will be evaluated with different sign for the two adjacent FVs.
- This property, however, holds only if the procedure for the evaluation of the face flux is unique, e.g. it is the same for both adjacent FVs.
- Although this property (*telescopic property*) seems obvious, in case of arbitrary unstructured *collocated* grids it is necessary to adopt some specific measures, in order to satisfy this rule, which constitutes the base, and the principal advantage, of the Finite Volume method.

The steady case

- Numerical approximation of surface and volume integrals that appear in the integral form of the conservation equation.
- Adoption of interpolation techniques, in order to evaluate some quantities in locations different from those where they are defined.

Conservation equation for the steady case

$$\int_A \rho\phi\mathbf{w}\cdot\mathbf{n}\,\mathrm{d}A = \int_A \Gamma\nabla\phi\cdot\mathbf{n}\,\mathrm{d}A + \int_V s\,\mathrm{d}V$$

Net flux

Net flux through the boundary of the CV

$$\int_A \mathbf{J}'' \cdot \mathbf{n}\, \mathrm{d}A = \sum_k \int_{A_k} \mathbf{J}'' \cdot \mathbf{n}\, \mathrm{d}A$$
$$= \sum_k \int_{A_k} \mathbf{J}_c'' \cdot \mathbf{n}\, \mathrm{d}A - \sum_k \int_{A_k} \mathbf{J}_d'' \cdot \mathbf{n}\, \mathrm{d}A$$

with

$$\mathbf{J}'' = \mathbf{J}_c'' - \mathbf{J}_d'' = \rho\phi\mathbf{w} - \Gamma\nabla\phi$$

where $\mathbf{J}''$ is the overall specific flux, and $\mathbf{J}_c''$ and $\mathbf{J}_d''$ are the specific convective and diffusive fluxes, respectively.

- In order to guarantee *local* conservativity at the CV level, and *global* conservativity for the entire domain, CVs should not overlap:
  - Every face belongs to one CV, if it lies on the boundary, or belongs to two CVs if it is internal to the domain.
  - This property should be guaranteed also for more complex cases, like e.g. *sliding grids*, *overset (chimera) grids* et.
- In the following we will consider just one face, the one indicated with *"e"* in the figure; similar expressions can be derived for the other faces by proper index substitution.

Approximations

- To calculate the surface integral on face *e* exactly, it would be necessary to know the integrand ($\mathbf{J}'' \cdot \mathbf{n}$) everywhere on the face.
- This is not possible, since the variable ϕ, and therefore its associated fluxes, are known only on the nodes (centers) of the CV.
- Therefore, two distinct approximations are required:
  1. The integral is approximated in terms of one, or more, variable value on the face;
  2. The variable values on the cell face are approximated in terms of nodal (center) values.
- Referring later to the approximate derivation of face values in terms of node (center) variable values, we look first at the approximate evaluation of the integrals.

Approximations of surface integrals

Midpoint rule

$$F_e = \int_{A_e} \mathbf{J}'' \cdot \mathbf{n} \, \mathrm{d}A = \overline{J''_e} \, A_e = \overline{J''_e} \, \Delta y_j \approx J''_e \, \Delta y_j$$

Trapezoidal rule

$$F_e = \int_{A_e} \mathbf{J}'' \cdot \mathbf{n} \, \mathrm{d}A \approx \frac{\Delta y_j}{2} \left(J''_{ne} + J''_{se} \right)$$

Higher order approximations: Simpson's rule

$$F_e = \int_{A_e} \mathbf{J}'' \cdot \mathbf{n} \, \mathrm{d}A \approx \frac{\Delta y_j}{6} \left(J''_{ne} + 4 J''_e + J''_{se} \right)$$

Approximations of volume integrals

Midpoint rule (2^{nd} order)

$$S_P = \int_V s \, \mathrm{d}V = \overline{s} \, \Delta V \approx s_P \, \Delta V$$

which in 2d can be expressed as:

$$S_P \approx s_P \, \Delta x_i \, \Delta y_j$$

Higher order approximation (4^{th} *order*)

$$S_P = \int_V s \, \mathrm{d}V \approx \Delta x_i \, \Delta y_j \left[a_0 + \frac{a_3}{12} (\Delta x_i)^2 + \frac{a_4}{12} (\Delta y_j)^2 + \frac{a_8}{144} (\Delta x_i)^2 (\Delta y_j)^2 \right]$$

that, for uniform Cartesian grids, where $\Delta x_i \equiv \Delta x$ e $\Delta y_j \equiv \Delta y$, became:

$$S_P \approx \frac{\Delta x \, \Delta y}{36} \left[16 \, s_P + 4 \, s_s + 4 \, s_n + 4 \, s_w + 4 \, s_e + s_{ne} + s_{se} + s_{nw} + s_{sw} \right]$$

Source term integration

- Meaning of the source term s:
  - In the source term one should insert all other contributions that can not be part of transport (advection and diffusion) and unsteady terms.
- If s_P is known and it *not* depends on ϕ, there are no difficulties.
- If, as usual, it depends on the variable ϕ, it must be linearized:
  - It is appropriate - and convenient - to take into account such dependence in the formulation of the discretized equation;
  - Such dependence can be, at most, *linear*, since the final result will be incorporated in a linear system of equations.

Source term linearization

$$s_P = s_P^{rhs} + s_P^{lhs}\,\phi_P$$

with

$$s_P^{lhs} < 0 \text{ e } s_P^{rhs} > 0.$$

Source term linearization

- If s_P is a non-linear function of ϕ, it is necessary to linearize it, i.e. specify the values of s_P^{lhs} e s_P^{rhs}, which may themselves depend on ϕ:
  - At every iteration, or time-step, s_P^{lhs} e s_P^{rhs} would then be recomputed from the new values (*in store*) of ϕ_P; in the following we indicate these, for brevity, as ϕ_P^* (ϕ_P^k or ϕ_P^n);
  - The chosen linearization for s_P should be a good representation of the $s_P = s_P(\phi_P)$ relationship.
- The basic rule $s_P^{lhs} < 0$ should be always be observed:
  - This rule would not be required if the final system of linear equations, at every iterative cycle or time-step, would be solved by a *direct* method.
  - Vice versa, this rule is of paramount importance if, as it is (almost) always the case in CFD, the solution of the linear system of equations is sought by an *iterative* method.

Let see some examples in the following.

Examples (1)

Example 1

Given:

$$s = 6 - 3\phi$$

let's see some possible linearizations:

1. $s_P^{rhs} = 6$; $s_P^{lhs} = -3$. Most obvious and recommended.
2. $s_P^{rhs} = 6 - 3\phi_P^*$; $s_P^{lhs} = 0$. Not efficient, but sometimes it represents the only choice for complex expressions of the source term (e.g. turbulence models).
3. $s_P^{rhs} = 6 + 5\phi_P^*$; $s_P^{lhs} = -8$. This proposed relationship is steeper than that given (remember that ϕ_P^* should be considered, for all purposes, a *constant*). It corresponds to an *under-relaxation*, and therefore it will lead to a slow down of the convergence of the (external) iterative cycle (see later). It is suitable if there are other, stiff non-linearities in the problem.

Examples (2)

Example 2

Given:

$$s = 4 + 11\phi$$

some possible linearizations are:

1. $s_P^{rhs} = 4$; $s_P^{lhs} = 11$. In general not acceptable, since it will make $s_P^{lhs} > 0$.
2. $s_P^{rhs} = 4 + 11\phi_P^*$; $s_P^{lhs} = 0$. This is the correct choice, since it is impossible to find a *natural* formulation which gives $s_P^{lhs} < 0$.
3. $s_P^{rhs} = 4 + 15\phi_P^*$; $s_P^{lhs} = -4$. This corresponds to an artificial creation of $s_P^{lhs} < 0$. It will slow down the convergence, but it can also be useful (robustness, reduced risks of divergence of the simulation).

Examples (3)

Example 3

Given:

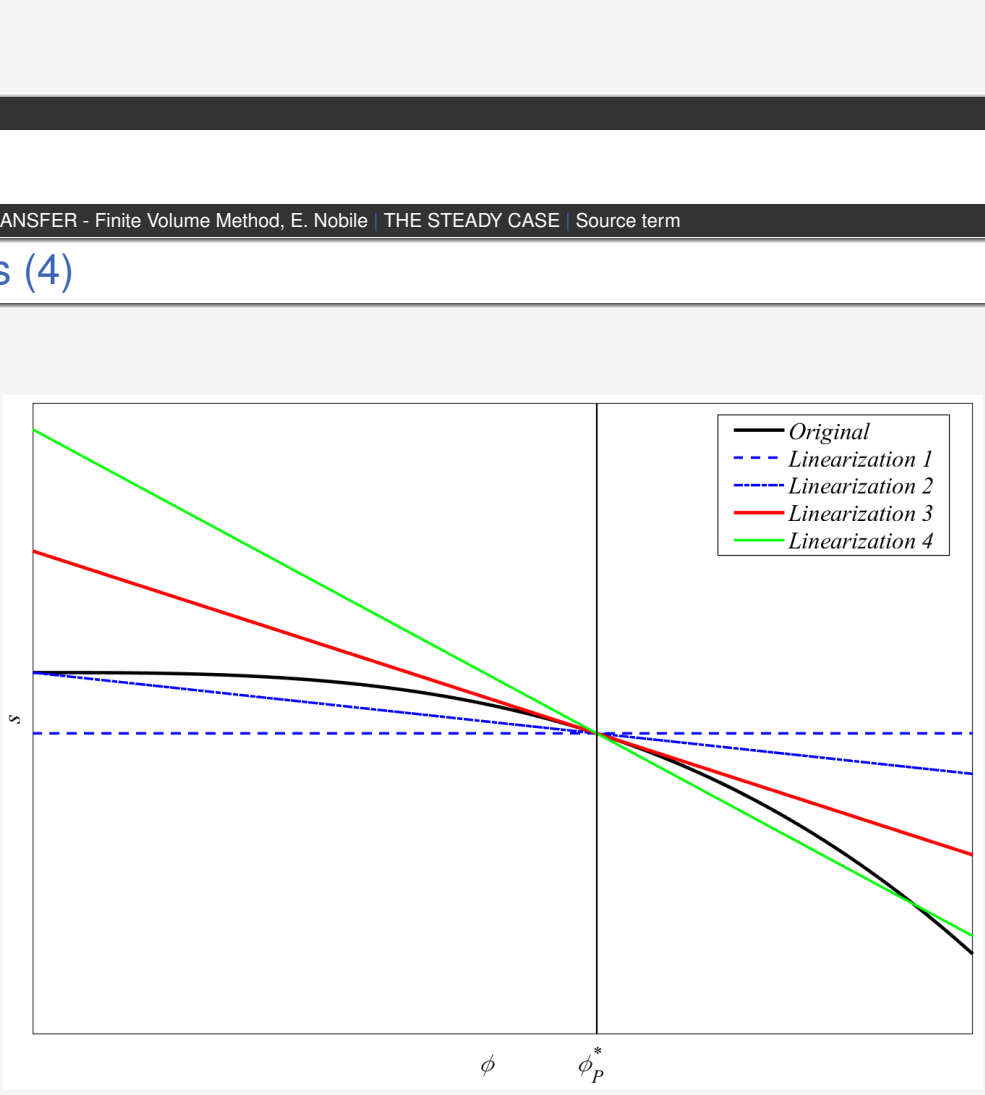
$$s = 3 - 5\phi^3$$

1. $s_P^{rhs} = 3 - 5\phi_P^{*3}$; $s_P^{lhs} = 0$. Inefficient, does not take advantage of the dependence of S on ϕ.
2. $s_P^{rhs} = 3$; $s_P^{lhs} = -5\phi_P^{*2}$. This seems correct, but the given relationship $s_P = s_P(\phi_P)$ is steeper than this linearization.
3. Recommended method:

$$s = s^* + (\mathrm{d}s/\mathrm{d}\phi)_* (\phi_P - \phi_P^*) = 3 - 5\phi_P^{*3} - 15\phi_P^{*2} (\phi_P - \phi_P^*)$$

   thus: $s_P^{rhs} = 3 + 10\phi_P^{*3}$; $s_P^{lhs} = -15\phi_P^{*2}$: tangent, in ϕ_P^*, to the curve curva $s_P = s_P(\phi_P)$.
4. $s_P^{rhs} = 3 + 20\phi_P^{*3}$; $s_P^{lhs} = -25\phi_P^{*2}$. This linearization is steeper than that given and it would slow down the convergence.

The four linearizations are illustrated in the next slide, together with the actual $s_P = s_P(\phi_P)$ curve.



Examples (4)

Interpolation

- The evaluation of the integrals requires the knowledge of the variables in locations different than those where they are defined, i.e. CV centers.
- The integrand is the product of several variables and/or their gradients: $J_c'' = \rho\phi\mathbf{w}\cdot\mathbf{n}$ for the convective - sometimes called *advective* - flux, and $J_d'' = \Gamma\nabla\phi\cdot\mathbf{n}$ for the diffusive flux.
- In order to compute these fluxes, it is first necessary to define the value of ϕ, and the face-normal component of its gradient, in one or more points of the face.
- Distinction between the face-normal component of the gradient - *diffusive flux* - and value of the variable - *convective flux*.

Schemes for the diffusive flux

CDS (Central Difference Scheme) - 2^{nd} order

$$\phi_e = \phi_E\,\lambda_{e,PE} + \phi_P\left(1-\lambda_{e,PE}\right)$$

$$\lambda_{e,PE} = \frac{x_e - x_P}{x_E - x_P}$$

Face-normal component of the gradient:

$$\left(\frac{\partial\phi}{\partial x}\right)_e \approx \frac{\phi_E - \phi_P}{x_E - x_P}$$

CDS - 4^{th} order

$$\phi(x) = a + b\,x + c\,x^2 + d\,x^3$$

For uniform grids:

$$\left(\frac{\partial\phi}{\partial x}\right)_e \approx \frac{27\,\phi_E - 27\,\phi_P + \phi_W - \phi_{EE}}{24\,\Delta x}$$

Some considerations about the schemes for the diffusive flux

- Use of 2^{nd} order schemes for the diffusive fluxes;
- Higher order schemes increase the computational cost: larger *computational molecule*;
- Higher order schemes are tendentially *less robust*;
- Use of the *deferred correction*;
- High order approximations do not necessarily guarantee higher accuracy:
  - The grid should be *sufficiently fine*;
  - Resort to *grid refinement*.

Diffusive flux

Face diffusivity

Value of face diffusivity Γ_e when the diffusivity exhibits spatial variations:

$$\Gamma_e = \left[\frac{\lambda_{e,PE}}{\Gamma_P} + \frac{(1-\lambda_{e,PE})}{\Gamma_E} \right]^{-1}$$

For a uniform grid: *harmonic mean* of Γ_P e Γ_E:

$$\Gamma_e = \frac{2\Gamma_E \Gamma_P}{\Gamma_E + \Gamma_P}$$

Diffusive flux density

$$J''_{d,e} = (\Gamma \nabla\phi \cdot \mathbf{n})_e = \Gamma_e \left(\frac{\partial \phi}{\partial x} \right)_e$$

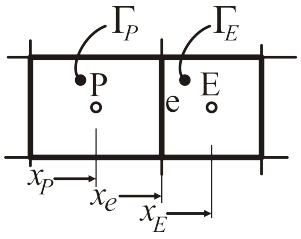
Equivalent diffusivity

Derivation of the face diffusivity

The previous expression for Γ_e can be easily found by extending the *electric analogy*, used in heat transfer, to a generic diffusion problem.
We can in fact express the specific *diffusive resistance* of a homogeneous layer of unitary area, having diffusivity Γ and thickness L, as

$$R''_\phi = \frac{L}{\Gamma} \quad \Rightarrow q'' = \frac{\phi_P - \phi_E}{R''_\phi}$$

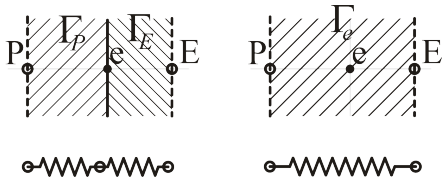
with q'' diffusive flux density.
For materials with variable diffusivity, Γ_P and Γ_E represent the (average) diffusivity of cell P and cell E, respectively.



Equivalent diffusivity - cont.

Derivation of the face diffusivity - cont.

In our case, the diffusive resistance between P and E is given by the *series* of two diffusive resistances: therefore, the equivalent diffusivity must be such as to give rise to the same specific resistance and therefore of the flux density:



$$R''_\phi = \frac{x_e - x_P}{\Gamma_P} + \frac{x_E - x_e}{\Gamma_E} = \frac{x_E - x_P}{\Gamma_e}$$

and solving for Γ_e:

$$\Gamma_e = \frac{1}{\frac{\lambda_{e,PE}}{\Gamma_P} + \frac{(1-\lambda_{e,PE})}{\Gamma_E}}$$

Basic schemes for the convective flux

Convective flux density:

$$J''_{c,e} = (\rho\,\phi\,\mathbf{w}\cdot\mathbf{n})_e = \rho\phi_e\,u_e$$

u_e is the velocity component normal to the face e, and it is assumed to be known.

CDS - Central Difference Scheme

$$\phi_e = \phi_E\,\lambda_{e,PE} + \phi_P\,(1-\lambda_{e,PE})$$

UDS - Upwind Difference Scheme

$$\phi_e = \begin{cases} \phi_P & \text{se} \quad (\mathbf{w}\cdot\mathbf{n})_e > 0 \\ \phi_E & \text{se} \quad (\mathbf{w}\cdot\mathbf{n})_e < 0 \end{cases}$$

Numerical diffusion

- The UDS scheme guarantees absence of unphysical oscillations (dispersion), but it introduces a relevant *numerical diffusion*.
- Assuming $(\mathbf{w}\cdot\mathbf{n})_e > 0$:

$$\phi_e = \phi_P + (x_e - x_P)\left(\frac{\partial\phi}{\partial x}\right)_P + \frac{(x_e-x_P)^2}{2}\left(\frac{\partial^2\phi}{\partial x^2}\right)_P + \dots$$

  The UDS scheme contains only the first term on the right and therefore it is a first order scheme. The truncation error (second term on the right) reminds the expression for the diffusive flux.
- The UDS scheme, therefore, introduces a *false diffusion flux* given by:

$$J''_{Num} = \Gamma_{Num}\left(\frac{\partial\phi}{\partial x}\right)_P \qquad \text{con} \qquad \Gamma_{Num} = \rho\,u_e\,\frac{\Delta x_P}{2}$$

- Moreover, the artificial diffusion increases when the streamlines are not aligned with the grid lines.

Use of the UDS scheme

Why the UDS scheme (or similar schemes like i.e. HY - Hybrid, which uses the CDS scheme in cases where the convective flow is modest, switching to UDS in other cases) is still available in most commercial and open source CFD products?

- Simplicity and absence of oscillations: more guarantees to obtain a preliminary (although very approximate, and frequently of no physical value) solution of the problem;
- Start of the simulation in difficult (stiff) cases, then continuing with more accurate schemes;
- Use of higher order schemes in *deferred correction* mode.

High order schemes for the convective flux

SOUDS - Second Order Upwind Difference Scheme

$$\phi_e = \begin{cases} \phi_P + \left[\lambda_{e,PW}\left(\phi_W - \phi_P\right)\right] & \text{if} \quad (\mathbf{w}\cdot\mathbf{n})_e > 0 \\ \phi_E + \left[\lambda_{e,E\,EE}\left(\phi_{EE} - \phi_E\right)\right] & \text{if} \quad (\mathbf{w}\cdot\mathbf{n})_e < 0 \end{cases}$$

$$\lambda_{e,PW} = \frac{x_e - x_P}{x_W - x_P}; \qquad \lambda_{e,E\,EE} = \frac{x_e - x_E}{x_{EE} - x_E}$$

In case $(\mathbf{w}\cdot\mathbf{n})_e > 0$:

$$\phi_e = \phi_P + \frac{\phi_W - \phi_P}{x_W - x_P}\left(x_e - x_P\right)$$

QUICK - Quadratic Upstream Interpolation for Convective Kinematics

$$\phi_e = \begin{cases} \phi_P + \left[\gamma_{1e}\phi_W - \left(\gamma_{1e} + \gamma_{2e}\right)\phi_P + \gamma_{2e}\phi_E\right] & \text{if} \quad (\mathbf{w}\cdot\mathbf{n})_e > 0 \\ \phi_E + \left[\gamma_{3e}\phi_P - \left(\gamma_{3e} + \gamma_{4e}\right)\phi_E + \gamma_{4e}\phi_{EE}\right] & \text{if} \quad (\mathbf{w}\cdot\mathbf{n})_e < 0 \end{cases}$$

$$\gamma_{1e} = \lambda_{e,EW}\,\lambda_{e,PW} \qquad \gamma_{2e} = \lambda_{e,PE}\,\lambda_{e,WE}$$
$$\gamma_{3e} = \lambda_{e,EP}\,\lambda_{e,EE\,P} \qquad \gamma_{4e} = \lambda_{e,P\,EE}\,\lambda_{e,E\,EE}$$

Some considerations about high order schemes for the convective flux

- Although more *robust* than the CDS scheme, and in the mean time more *accurate* of the UDS scheme, high-order schemes may however give rise to non-physical oscillations of the solution (wiggles).
- To make high-order schemes more stable and oscillation-free, the most common approaches can be grouped into two classes, i.e. *flux blending methods* and *composite flux limiter methods*.
- *Flux blending methods*:
  - An *anti-diffusive* flux is added to a 1^{st} order upwind scheme (eg. FCT - Flux Corrected Transport).
  - A certain amount of artificial diffusivity is added - in a selective way - to a high-order scheme to dampen oscillations. (eg. FRAM - Filtering Remedy and Methodology).
- *Composite flux limiter methods*.

  The flux on the cell-face is modified via a criterion that imposes a limit value (boundedness):
  - Total Variational Diminishing (TVD) flux limiters.
  - Normalized Variable Formulation (NVF), or Normalized Variable and Space Formulation (NVSF).

Other high order schemes

- On this basis some other high-order schemes have been developed:
  - MSOU - Monotonic Second Order Upwind scheme;
  - HLPA - Hybrid Linear/Parabolic Approximation;
  - SHARP - Simple High-Accuracy Resolution Program);
  - SMART - Sharp and Monotonic Algorithm for Realistic Transport;
  - NIRVANA - Nonoscillatory, Integrally Reconstructed Volume-Averaged Numerical Advection.
- Drawbacks of such schemes:
  - Large size of the calculation molecule, unless one uses the *deferred correction*;
  - Difficult extension to unstructured grids.

Example - description

Performance of some convective schemes for the test of the purely convective, two-dimensional transport for a step profile.

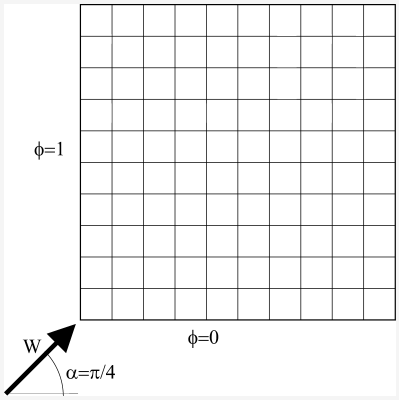
- Imposed and constant flow field, inclined with an angle $\alpha = \pi/4$ respect to the grid.
- Boundary conditions are indicated in the figure, and the transport equation considered is $\nabla \cdot (\rho \mathbf{w} \phi) = 0$ which becames

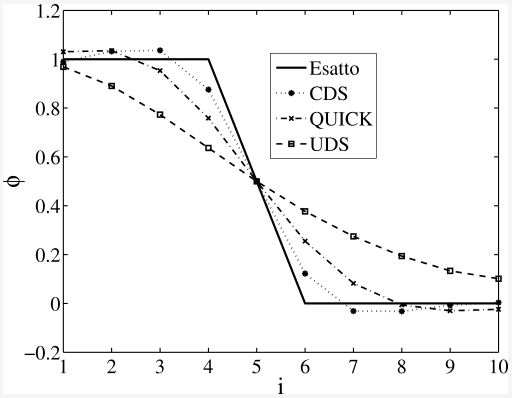
$$u\frac{\partial \phi}{\partial x} + v\frac{\partial \phi}{\partial y} = 0$$

- Due to lack of diffusion, the *step* of the ϕ profile at the inlet, should be ideally transported, without diffusion or wiggles, indefinitely.
- The presence of artificial diffusion, or/and wiggles, may alter the result.

Example - results

The profiles, obtained along the horizontal section at mid-height, are compared with the *exact* profile, e.g. that most accurate with the given grid.





- The UDS scheme is free of oscillations, but it completely alters the profile, due to its high artificial diffusivity.
- The CDS scheme almost maintains the profile (note that the grid is very coarse), but it introduces non-physical oscillations (wiggles).
- The QUICK scheme has an intermediate behavior between UDS and CDS.

Deferred correction

- Difficulties in the use of high order schemes.
- Insufficient accuracy of simple (low order) schemes.

Deferred Correction

Indicating with k the $k-$th iteration we have

$$\phi_e^k = \underbrace{\phi_e^{lo,\,k}}_{lhs} + \underbrace{\left[\phi_e^{ho,\,k-1} - \phi_e^{lo,\,k-1}\right]}_{rhs}$$

lo indicates a *low order* scheme (e.g. UDS) and *ho* indicates a *high order* one (e.g. QUICK).

- Formulation of the expressions for SOUDS and QUICK.
- Low computational cost, better convergence, reduced possibility of non-physical oscillations of the solution.
- An iterative procedure already present for the solution of (non-linear) thermo-fluid problems o, equivalently, time-steps for temporal integration.

Final algebraic equation

Two-dimensional problems

$$A_P\phi_P + A_E\phi_E + A_W\phi_W + A_N\phi_N + A_S\phi_S = S_P$$

Three-dimensional problems

$$A_P\phi_P + A_E\phi_E + A_W\phi_W + A_N\phi_N + A_S\phi_S + A_T\phi_T + A_B\phi_B = S_P$$

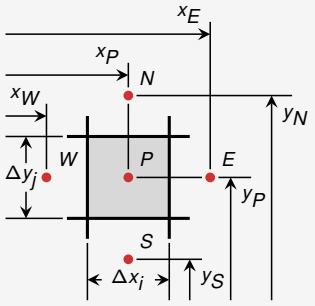
Compact form

$$A_P\phi_P + \sum_{nb} A_{nb}\phi_{nb} = S_P$$

where *nb* (from *neighbor*) means that the summation includes all adjacent VCs.

The expressions to use for A_P, A_E, A_W *etc.* depend on the schemes adopted for the diffusive and convective terms, and on the relations used to evaluate the integrals.

Steady 2D diffusion



The conservation equation for 2D steady diffusion is

$$-\int_A \mathbf{J}_d'' \cdot \mathbf{n}\, \mathrm{d}A = \int_V s\, \mathrm{d}V$$

or

$$-\sum_k \int_{A_k} \mathbf{J}_d'' \cdot \mathbf{n}\, \mathrm{d}A = \int_V s\, \mathrm{d}V$$

and, since $\mathbf{J}_d'' = \Gamma \nabla \phi$, it can be written also as

$$\sum_k \int_{A_k} (-\Gamma \nabla \phi) \cdot \mathbf{n}\, \mathrm{d}A = \int_V s\, \mathrm{d}V$$

This can be approximated as

$$\sum_k (-\Gamma \nabla \phi)_k \cdot \mathbf{S}_k = \left(s_P^{rhs} + s_P^{lhs} \phi_P\right) \Delta x_i \Delta y_j$$

which can be expanded as

$$(-\Gamma \nabla \phi)_e \cdot \mathbf{S}_e + (-\Gamma \nabla \phi)_w \cdot \mathbf{S}_w + (-\Gamma \nabla \phi)_n \cdot \mathbf{S}_n + (-\Gamma \nabla \phi)_s \cdot \mathbf{S}_s = \left(s_P^{rhs} + s_P^{lhs} \phi_P\right) \Delta x_i \Delta y_j$$

And since

$$\mathbf{S}_e = \Delta y_j \mathbf{i};\quad \mathbf{S}_w = -\Delta y_j \mathbf{i};\quad \mathbf{S}_n = \Delta x_i \mathbf{j};\quad \mathbf{S}_s = -\Delta x_i \mathbf{j}$$

we have, i.e., for the east face

$$(-\Gamma \nabla \phi)_e \cdot \mathbf{S}_e = -\Gamma_e \Delta y_j \left(\frac{\partial \phi}{\partial x}\mathbf{i} + \frac{\partial \phi}{\partial y}\mathbf{j}\right)_e \cdot \mathbf{i} = -\Gamma_e \Delta y_j \left(\frac{\partial \phi}{\partial x}\right)_e$$

Therefore, the previous balance equation can be written as

$$-\Gamma_e \Delta y_j \left(\frac{\partial \phi}{\partial x}\right)_e + \Gamma_w \Delta y_j \left(\frac{\partial \phi}{\partial x}\right)_w - \Gamma_n \Delta x_i \left(\frac{\partial \phi}{\partial y}\right)_n + \Gamma_s \Delta x_i \left(\frac{\partial \phi}{\partial y}\right)_s = \left(s_P^{rhs} + s_P^{lhs} \phi_P\right) \Delta x_i \Delta y_j$$

In particular, using the second order CDS

$$\left(\frac{\partial \phi}{\partial x}\right)_e \approx \frac{\phi_E - \phi_P}{x_E - x_P};\quad \left(\frac{\partial \phi}{\partial x}\right)_w \approx \frac{\phi_P - \phi_W}{x_P - x_W}$$

$$\left(\frac{\partial \phi}{\partial y}\right)_n \approx \frac{\phi_N - \phi_P}{y_N - y_P};\quad \left(\frac{\partial \phi}{\partial y}\right)_s \approx \frac{\phi_P - \phi_S}{y_P - y_S}$$

Steady 2D diffusion - cont.

the balance equation can be approximated as

$$+\Gamma_e \frac{\Delta y_j}{x_E - x_P}(\phi_P - \phi_E) + \Gamma_w \frac{\Delta y_j}{x_P - x_W}(\phi_P - \phi_W) + \Gamma_n \frac{\Delta x_i}{y_N - y_P}(\phi_P - \phi_N) + \Gamma_s \frac{\Delta x_i}{y_P - y_S}(\phi_P - \phi_S) = \left(s_P^{rhs} + s_P^{lhs} \phi_P\right) \Delta x_i \Delta y_j$$

or

$$\left(\Gamma_e \frac{\Delta y_j}{x_E - x_P} + \Gamma_w \frac{\Delta y_j}{x_P - x_W} + \Gamma_n \frac{\Delta x_i}{y_N - y_P} + \Gamma_s \frac{\Delta x_i}{y_P - y_S} - s_P^{lhs} \Delta x_i \Delta y_j\right) \phi_P - \frac{\Gamma_e \Delta y_j}{x_E - x_P}\phi_E - \frac{\Gamma_w \Delta y_j}{x_P - x_W}\phi_W - \frac{\Gamma_n \Delta x_i}{y_N - y_P}\phi_N - \frac{\Gamma_s \Delta x_i}{y_P - y_S}\phi_S = s_P^{rhs} \Delta x_i \Delta y_j$$

In another form:

$$\boxed{A_P \phi_P + A_E \phi_E + A_W \phi_W + A_N \phi_N + A_S \phi_S = S_P}$$

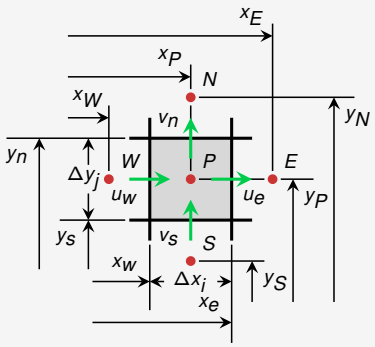
where

$$A_E = -\frac{\Gamma_e \Delta y_j}{x_E - x_P};\quad A_W = -\frac{\Gamma_w \Delta y_j}{x_P - x_W};\quad A_N = -\frac{\Gamma_n \Delta x_i}{y_N - y_P};\quad A_S = -\frac{\Gamma_s \Delta x_i}{y_P - y_S}$$

$$A_P = -(A_E + A_W + A_N + A_S) - s_P^{lhs} \Delta x_i \Delta y_j$$

$$S_P = s_P^{rhs} \Delta x_i \Delta y_j$$

Steady 2D diffusion and convection



The conservation equation for 2D steady diffusion and convection is

$$\int_A \mathbf{J}_c'' \cdot \mathbf{n}\, \mathrm{d}A - \int_A \mathbf{J}_d'' \cdot \mathbf{n}\, \mathrm{d}A = \int_V s\, \mathrm{d}V$$

or

$$\sum_k \int_{A_k} \mathbf{J}_c'' \cdot \mathbf{n}\, \mathrm{d}A - \sum_k \int_{A_k} \mathbf{J}_d'' \cdot \mathbf{n}\, \mathrm{d}A = \int_V s\, \mathrm{d}V$$

and, since $\mathbf{J}_c'' = \rho\phi\mathbf{w}$ and $\mathbf{J}_d'' = \Gamma\nabla\phi$, it can be written also as

$$\sum_k \int_A \rho\phi\mathbf{w} \cdot \mathbf{n}\, \mathrm{d}A - \sum_k \int_A \Gamma\nabla\phi \cdot \mathbf{n}\, \mathrm{d}A = \int_V s\, \mathrm{d}V$$

which can be approximated as

$$\sum_k (\rho\phi\mathbf{w})\cdot\mathbf{S}_k + \sum_k (-\Gamma\nabla\phi)\cdot\mathbf{S}_k = \left(s_P^{rhs} + s_P^{lhs}\phi_P\right)\Delta x_i \Delta y_j$$

which can be expanded as

$$(\rho\phi\mathbf{w})_e \cdot \mathbf{S}_e + (\rho\phi\mathbf{w})_w \cdot \mathbf{S}_w + (\rho\phi\mathbf{w})_n \cdot \mathbf{S}_n + (\rho\phi\mathbf{w})_s \cdot \mathbf{S}_s$$
$$+ (-\Gamma\nabla\phi)_e \cdot \mathbf{S}_e + (-\Gamma\nabla\phi)_w \cdot \mathbf{S}_w$$
$$+ (-\Gamma\nabla\phi)_n \cdot \mathbf{S}_n + (-\Gamma\nabla\phi)_s \cdot \mathbf{S}_s = \left(s_P^{rhs} + s_P^{lhs}\phi_P\right)\Delta x_i \Delta y_j$$

Proceeding as for the 2D steady diffusion case for the diffusive fluxes, and using the second order CDS also for the convective fluxes, it results i.e. for the east face

$$\frac{\phi_E - \phi_P}{x_E - x_P} = \frac{\phi_e - \phi_P}{x_e - x_P}$$

It follows

$$\phi_e = \phi_P \underbrace{\left[1 - \frac{x_e - x_P}{x_E - x_P}\right]}_{1-\lambda_{e,PE}} + \phi_E \underbrace{\left[\frac{x_e - x_P}{x_E - x_P}\right]}_{\lambda_{e,PE}}$$

and similarly

$$\phi_w = \phi_W \lambda_{w,PW} + \phi_P(1 - \lambda_{w,PW})$$
$$\phi_n = \phi_N \lambda_{n,PN} + \phi_P(1 - \lambda_{n,PN})$$
$$\phi_s = \phi_S \lambda_{s,PS} + \phi_P(1 - \lambda_{s,PS})$$

with

$$\lambda_{e,PE} = \frac{x_e - x_P}{x_E - x_P}; \quad \lambda_{w,PW} = \frac{x_w - x_P}{x_W - x_P}$$
$$\lambda_{n,PN} = \frac{y_n - y_P}{y_N - y_P}; \quad \lambda_{s,PS} = \frac{y_s - y_P}{y_S - y_P}$$

Steady 2D diffusion and convection - cont.

And since

$$\mathbf{S}_e = \Delta y_j \mathbf{i}; \quad \mathbf{S}_w = -\Delta y_j \mathbf{i}; \quad \mathbf{S}_n = \Delta x_i \mathbf{j}; \quad \mathbf{S}_s = -\Delta x_i \mathbf{j}$$

we have, i.e., for the east face

$$(\rho\phi\mathbf{w})_e \cdot \mathbf{S}_e = \rho \Delta y_j u_e \phi_e = \dot{m}_e \phi_e$$
$$= \dot{m}_e \phi_E \lambda_{e,PE} + \dot{m}_e \phi_P (1 - \lambda_{e,PE})$$

and similarly for the other faces

$$(\rho\phi\mathbf{w})_w \cdot \mathbf{S}_w = \dot{m}_w \phi_W \lambda_{w,PW} + \dot{m}_w \phi_P (1 - \lambda_{w,PW})$$
$$(\rho\phi\mathbf{w})_n \cdot \mathbf{S}_n = \dot{m}_n \phi_N \lambda_{n,PN} + \dot{m}_n \phi_P (1 - \lambda_{n,PN})$$
$$(\rho\phi\mathbf{w})_s \cdot \mathbf{S}_s = \dot{m}_s \phi_S \lambda_{s,PS} + \dot{m}_s \phi_P (1 - \lambda_{s,PS})$$

with

$$\dot{m}_e = \rho\, u_e\, \Delta y_j; \quad \dot{m}_w = \rho\, u_w \Delta y_j$$
$$\dot{m}_n = \rho\, v_n\, \Delta x_i; \quad \dot{m}_s = \rho\, v_s \Delta x_i$$

Defining the cofficients for the convective fluxes

$$A_E^c = \dot{m}_e\, \lambda_{e,PE}; \quad A_W^c = -\dot{m}_w\, \lambda_{w,PW}$$
$$A_N^c = \dot{m}_n\, \lambda_{n,PN}; \quad A_S^c = -\dot{m}_s\, \lambda_{s,PS}$$

and, as already seen, the coefficents for the diffusive fluxes

$$A_E^d = -\frac{\Gamma_e\, \Delta y_j}{x_E - x_P}; \quad A_W^d = -\frac{\Gamma_w\, \Delta y_j}{x_P - x_W}$$
$$A_N^d = -\frac{\Gamma_n\, \Delta x_i}{y_N - y_P}; \quad A_S^d = -\frac{\Gamma_s\, \Delta x_i}{y_P - y_S}$$

we have

$$\left[A_E^c + A_E^d\right]\phi_E + \left[A_W^c + A_W^d\right]\phi_W$$
$$+ \left[A_N^c + A_N^d\right]\phi_N + \left[A_S^c + A_S^d\right]\phi_S$$
$$+\Big\{-\left[A_E^c + A_E^d\right] - \left[A_W^c + A_W^d\right] - \left[A_N^c + A_N^d\right] - \left[A_S^c + A_S^d\right]$$
$$+\dot{m}_e - \dot{m}_w + \dot{m}_n - \dot{m}_s\Big\}\phi_P = \left(s_P^{rhs} + s_P^{lhs}\phi_P\right)\Delta x_i\, \Delta y_j$$

Observing that

$$\sum_k \dot{m}_k = \dot{m}_e - \dot{m}_w + \dot{m}_n - \dot{m}_s \equiv 0$$

and defining

$$A_E = A_E^c + A_E^d; \quad A_W = A_W^c + A_W^d$$
$$A_N = A_N^c + A_N^d; \quad A_S = A_S^c + A_S^d$$

$$A_P = -\left(A_E + A_W + A_N + A_S\right) - s_P^{lhs}\, \Delta x_i\, \Delta y_j$$

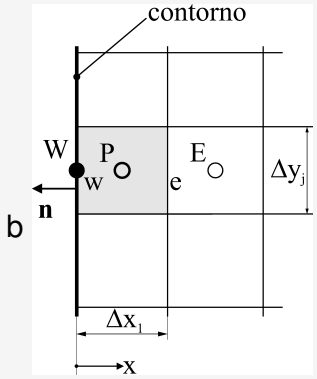
$$S_P = s_P^{rhs}\, \Delta x_i\, \Delta y_j$$

We are left with the final form of the equation

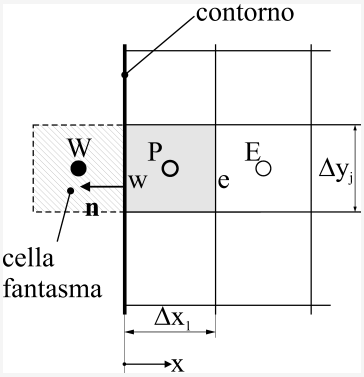
$$\boxed{A_P\phi_P + A_E\phi_E + A_W\phi_W + A_N\phi_N + A_S\phi_S = S_P}$$

Boundary conditions

- The boundary conditions have to be applied *before* assembling, and finally solving, the system of equations.
- For Cartesian grids, or more generally *structured grids*, two different approaches are possible:



Method 1: node (cell with zero volume) on the boundary.



Method 2: use of a *ghost cell*.

Convective boundary condition - Method 1

$$\mathbf{n} \cdot \Gamma \nabla \phi + \alpha \left(\phi_{BC} - \phi_\infty \right) = 0$$

In another form:

$$-\Gamma \frac{\partial \phi}{\partial x} + \alpha \, \phi_{BC} = \alpha \, \phi_\infty$$

and in discrete terms:

$$-\Gamma \frac{\phi_P - \phi_W}{\Delta x_1 / 2} + \alpha \, \phi_W = \alpha \, \phi_\infty$$

$$\phi_W = \phi_P \underbrace{\left[\frac{2\,\Gamma}{2\,\Gamma + \alpha\,\Delta x_1} \right]}_{C_{lhs}} + \underbrace{\left[\frac{\alpha\,\Delta x_1}{2\,\Gamma + \alpha\,\Delta x_1} \, \phi_\infty \right]}_{C_{rhs}}$$

Algebraic equation for the cell P:

$$\underbrace{\left(A_P + A_W \, C_{lhs} \right)}_{A_P} \phi_P + A_E \phi_E + A_N \phi_N + A_S \phi_S = \underbrace{\left(S_P - A_W \, C_{rhs} \right)}_{S_P}$$

Dirichlet boundary condition - Method 2

As already seen for the FD method, it is the most simple boundary condition, and it corresponds, for the case of the energy equation, to an imposed value of temperature.
Indicating with ϕ_{BC} the assigned value of the temperature, we have:

$$\phi_{BC} = \frac{\phi_P + \phi_W}{2}$$

it follows:

$$\phi_W = 2\,\phi_{BC} - \phi_P$$

Algebraic equation for the cell P:

$$\underbrace{(A_P - A_W)}_{A_P}\,\phi_P + A_E\phi_E + A_N\phi_N + A_S\phi_S = \underbrace{(S_P - 2\,A_W\,\phi_{BC})}_{S_P}$$

Under-relaxation of the iterative process

We have already seen that the general form of the discretization equation is

$$A_P\phi_P + A_E\phi_E + A_W\phi_W + A_N\phi_N + A_S\phi_S = S_P$$
$$A_P\phi_P + A_E\phi_E + A_W\phi_W + A_N\phi_N + A_S\phi_S + A_T\phi_T + A_B\phi_B = S_P$$

for 2D and 3D problems, respectively. In a general compact form

$$A_P\phi_P + \sum_{nb} A_{nb}\phi_{nb} = S_P$$

For a general convection-diffusion problem, both Γ and s may be function of ϕ, which may lead to large variations in the coefficients and in the source term.
This, in turn, may cause divergence, or slow oscillatory convergence, of the iterative solution procedure.
To favor convergence, it may be useful to *slow down* the variations of ϕ between iterations by *under-relaxation*. This can be done in several ways, one of the most common is illustrated next.

Under-relaxation of the iterative process - *cont.*

The previous equation can be written as

$$\phi_P = \frac{-\sum_{nb} A_{nb}\phi_{nb} + S_P}{A_P}$$

If k is the iteration index, ϕ_P^{k-1} represents the variable at the previous iteration, which can be added and subtracted from the right hand side of the previous equation

$$\phi_P^k = \phi_P^{k-1} + \left(\frac{-\sum_{nb} A_{nb}\phi_{nb}^k + S_P}{A_P} - \phi_P^{k-1} \right)$$

where the term between parentheses is the change in ϕ_P obtained from the current iteration. This change can be modified by the introduction of a *relaxation factor* α_ϕ such that

$$\phi_P^k = \phi_P^{k-1} + \alpha_\phi \left(\frac{-\sum_{nb} A_{nb}\phi_{nb}^k + S_P}{A_P} - \phi_P^{k-1} \right)$$

Under-relaxation of the iterative process - *cont.*

The previous equation can be rewritten as

$$\underbrace{\frac{A_P}{\alpha_\phi}}_{A_P}\phi_P^k + \sum_{nb} A_{nb}\phi_{nb}^k = \underbrace{S_P + \frac{(1-\alpha_\phi)\,A_P}{\alpha_\phi}\phi_P^{k-1}}_{S_P}$$

- At convergence $\phi_P^k \equiv \phi_P^{k-1} = \phi_P$ and its value is independent of α_ϕ and also satisfies the original equation.
- Depending on the value of α_ϕ, the equation can be
  1. Under-relaxed, i.e. $0 < \alpha_\phi < 1$
  2. Over-relaxed, i.e. $\alpha_\phi > 1$
- In CFD and CHT applications, under-relaxation is usually used.
- The *optimum* value of α_ϕ is problem dependent and there are not general rules. It depends, among other, by the type of problem, grid size and quality, iterative solver et.
- The value of α_ϕ can vary within the domain, and can change between iterations.

OUTLINE

Part III

Temporal integration

Temporal Integration

Generic transport equation in unsteady form:

$$\frac{\partial}{\partial \vartheta}(\rho\phi) + \nabla \cdot (\rho \mathbf{w}\phi) = \nabla \cdot (\Gamma \nabla \phi) + s$$

- Integration in time by *finite differences*: the overall time span ϑ is divided in a certain number of time intervals $\Delta\vartheta$.
- Temporal integration by *marching* in time, knowing the *initial condition* at time $\vartheta = \vartheta_0$, and the boundary conditions at all successive time instants $\vartheta = l\,\Delta\vartheta$, with $l = 1, \ldots, n, n+1, \ldots$

The equation can be written in a compact form:

$$\frac{\partial}{\partial \vartheta}(\rho\phi) = \mathcal{F}(\vartheta, \phi(\vartheta))$$

where

$$\mathcal{F} = -\nabla \cdot (\rho \mathbf{w}\phi) + \nabla \cdot (\Gamma \nabla \phi) + s$$

This formulation resembles the classical initial value problems, i.e. ordinary differential equations (ODE) with initial values.

ODE - Ordinary Differential Equations

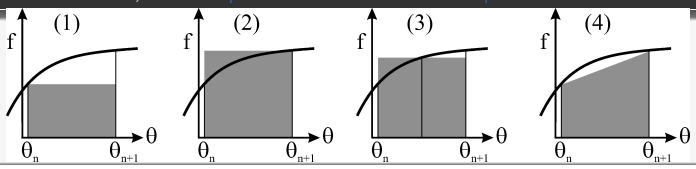
First order Ordinary Differential Equation (ODE), with initial condition at time ϑ_0:

$$\frac{\mathrm{d}\phi(\vartheta)}{\mathrm{d}\vartheta} = f(\vartheta, \phi(\vartheta)) \qquad \text{con} \quad \phi(\vartheta_0) = \phi^0$$

We want to calculate ϕ after an interval $\Delta\vartheta$ from the initial instant, in order to obtain ϕ^1 at the instant $\vartheta_1 = \vartheta_0 + \Delta\vartheta$. This will follow with the determination of ϕ^2 at instant $\vartheta_2 = \vartheta_1 + \Delta\vartheta$ and so on.
The most simple methods are obtained by integrating the previous equation between ϑ_n and $\vartheta_{n+1} = \vartheta_n + \Delta\vartheta$:

$$\int_{\vartheta_n}^{\vartheta_{n+1}} \frac{\mathrm{d}\phi(\vartheta)}{\mathrm{d}\vartheta}\,\mathrm{d}\vartheta = \phi^{n+1} - \phi^n = \int_{\vartheta_n}^{\vartheta_{n+1}} f(\vartheta, \phi(\vartheta))\,\mathrm{d}\vartheta$$

where $\phi^{n+1} = \phi(\vartheta_{n+1})$. The expression is still correct, but the term on the right hand-side cannot be evaluated without knowing the solution at the new time instant: this fact, therefore, requires some approximation. In the following we will see the most common methods.



(1) - *Explicit* or *forward Euler* method

$$\phi^{n+1} = \phi^n + f(\vartheta_n, \phi^n)\,\Delta\vartheta$$

(2) - *Implicit* or *backward* Euler method

$$\phi^{n+1} = \phi^n + f\left(\vartheta_{n+1}, \phi^{n+1}\right)\Delta\vartheta$$

(3) - *Midpoint* method

$$\phi^{n+1} = \phi^n + f\left(\vartheta_{n+1/2}, \phi^{n+1/2}\right)\Delta\vartheta$$

(4) - *Crank-Nicolson* method

$$\phi^{n+1} = \phi^n + \frac{1}{2}\left[f\left(\vartheta_{n+1}, \phi^{n+1}\right) + f(\vartheta_n, \phi^n)\right]\Delta\vartheta$$

Stability and accuracy of basic methods

- The previous methods, a part from (3) (Midpoint method) are so-called *two-level* methods because they involve the values of the unknown at only two time instants.
- These methods but the (1) require the value of $\phi(\vartheta)$ at some point other than $\vartheta = \vartheta_n$, which is the initial point at which the solution is known. Therefore, for these methods, the right hand side cannot be calculated without further approximation or iteration.
- For this reason, (1) is an *explicit* method while all the others are *implicit*.
- Method (1), as for all explicit methods, is simpler and more economical, but it can be used only for values of $\Delta\vartheta$ lower than its stability limit:

$$\left| \Delta\vartheta \frac{\partial f(\vartheta, \phi)}{\partial \phi} \right| < 2 \tag{1}$$

- Implicit methods are computationally more expensive, but are *unconditionally stable*:
  - implicit methods – time-scale greater than the stability limit of explicit methods;
  - explicit methods – time-scale lower or of the same order of magnitude of the stability limit.
- Methods (1) and (2) are first-order accurate, while (3) and (4) are second-order accurate, $((\Delta\vartheta)^2)$.
- Although rarely used for time integration, the *midpoint* method can be regarded as the basis of an important method for solving partial differential equations - the *leapfrog* method.
- It can be proved that, with two-level methods, the accuracy can be, at best, of second-order.

Multilevel and predictor-corrector methods

Higher order methods:

- *Multistep* methods.
  - *Explicit Adams-Bashforth* methods: these are selected from the time instants already considered for which the solution is available.

    Second order

$$\phi^{n+1} = \phi^n + \frac{1}{2}\left[3 f\left(\vartheta_n, \phi^n\right) - f\left(\vartheta_{n-1}, \phi^{n-1}\right)\right] \Delta\vartheta$$

    Third order

$$\phi^{n+1} = \phi^n + \frac{1}{12}\left[23 f\left(\vartheta_n, \phi^n\right) - 16 f\left(\vartheta_{n-1}, \phi^{n-1}\right) + 5 f\left(\vartheta_{n-2}, \phi^{n-2}\right)\right] \Delta\vartheta$$

  - *Implicit Adams-Moulton* methods: these are selected from the actual (unknown) time instant and those already considered for which the solution is available.

    Second order (Crank-Nicolson)

$$\phi^{n+1} = \phi^n + \frac{1}{2}\left[f\left(\vartheta_{n+1}, \phi^{n+1}\right) + f\left(\vartheta_n, \phi^n\right)\right] \Delta\vartheta$$

    Third order

$$\phi^{n+1} = \phi^n + \frac{1}{12}\left[5 f\left(\vartheta_{n+1}, \phi^{n+1}\right) + 8 f\left(\vartheta_n, \phi^n\right) - f\left(\vartheta_{n-1}, \phi^{n-1}\right)\right] \Delta\vartheta$$

Multilevel and predictor-corrector methods - cont.

- *Multistage Schemes: Runge-Kutta*.
  The most famous multistage schemes are the Runge-Kutta. Unlike multistep schemes, multistage schemes only require the solution at the previous time step ϕ^n to approximate the new solution ϕ^{n+1} at time ϑ_{n+1}.

  A popular four-stage Runge-Kutta method (RK4), and perhaps the most popular of all Runge-Kutta methods is

$$\begin{aligned}
\phi^*_{n+1/2} &= \phi^n + \frac{\Delta\vartheta}{2} f\left(\vartheta_n, \phi^n\right) \\
\phi^{**}_{n+1/2} &= \phi^n + \frac{\Delta\vartheta}{2} f\left(\vartheta_{n+1/2}, \phi^*_{n+1/2}\right) \\
\phi^*_{n+1} &= \phi^n + \Delta\vartheta\, f\left(\vartheta_{n+1/2}, \phi^{**}_{n+1/2}\right) \\
\phi^{n+1} &= \phi^n + \frac{\Delta\vartheta}{6}\left[f\left(\vartheta_n, \phi^n\right) + 2\, f\left(\vartheta_{n+1/2}, \phi^*_{n+1/2}\right) + \right. \\
&\qquad \left. 2\, f\left(\vartheta_{n+1/2}, \phi^{**}_{n+1/2}\right) + f\left(\vartheta_{n+1}, \phi^*_{n+1}\right)\right]
\end{aligned}$$

Second order backward Euler (Gear) scheme - 1

The integrals that result from integration of our first order differential equation (ODE)

$$\int_{\vartheta_n}^{\vartheta_{n+1}} \frac{\mathrm{d}\phi\left(\vartheta\right)}{\mathrm{d}\vartheta}\,\mathrm{d}\vartheta = \phi^{n+1} - \phi^n = \int_{\vartheta_n}^{\vartheta_{n+1}} f\left(\vartheta, \phi\left(\vartheta\right)\right)\,\mathrm{d}\vartheta$$

reported here again for convenience, might be evaluated if one knows the *mean values* within the integration interval.

In particular, the *second order backward Euler*, or *Gear*, scheme, can be derived by integrating in a time interval $\Delta\vartheta$ centered around ϑ_{n+1}, i.e. between $\vartheta_{n+1} - \Delta\vartheta/2$ and $\vartheta_{n+1} + \Delta\vartheta/2$), and applying the mean value rule for both terms of the equation.
The time derivative can be approximated by differentiating a parabola passing through the solution at the three instants ϑ_{n-1}, ϑ_n and ϑ_{n+1}, obtaining

$$\left(\frac{\mathrm{d}\phi}{\mathrm{d}\vartheta}\right)^{n+1} \approx \frac{3\,\phi^{n+1} - 4\,\phi^n + \phi^{n-1}}{2\,\Delta\vartheta}$$

Second order backward Euler (Gear) scheme - 2

The resulting scheme is the following

$$\phi^{n+1} = \frac{4}{3}\phi^{n} - \frac{1}{3}\phi^{n-1} + \frac{2}{3} f\left(\vartheta_{n+1}, \phi^{n+1}\right) \Delta\vartheta$$

Its major properties are:

- Implicit - *f* is evaluated at the *new* time instant.
- Second order accurate.
- Simpler and computationally more economical in comparison to the *Crank-Nicolson* scheme, but its truncation error is four times larger.
- The major truncation error is due to the failure to take into account the temporal variation of the fluxes, which the Crank-Nicolson scheme does.
- Generally more stable - *A-stability* - than the Crank-Nicolson scheme.
- Negligible induced numerical diffusion for small time steps, as is the case i.e. for LES simulations.

Second order backward Euler (Gear) scheme - 3

Derivation based on Taylor series expansion

Developing a Taylor series expansion of ϕ^{n-1} and ϕ^{n} around ϕ^{n+1} we have

$$\phi^{n-1} = \phi^{n+1} - 2\left(\frac{\mathrm{d}\phi}{\mathrm{d}\vartheta}\right)^{n+1}\Delta\vartheta + 2\left(\frac{\mathrm{d}^2\phi}{\mathrm{d}\vartheta^2}\right)^{n+1}\Delta\vartheta^2 + O(\Delta\vartheta^3)$$

$$\phi^{n} = \phi^{n+1} - \left(\frac{\mathrm{d}\phi}{\mathrm{d}\vartheta}\right)^{n+1}\Delta\vartheta + \frac{1}{2}\left(\frac{\mathrm{d}^2\phi}{\mathrm{d}\vartheta^2}\right)^{n+1}\Delta\vartheta^2 + O(\Delta\vartheta^3)$$

Multiplying the second expression by 4 and subtracting it from the first

$$\phi^{n-1} - 4\phi^{n} = \phi^{n+1} - 4\phi^{n+1} - 2\left(\frac{\mathrm{d}\phi}{\mathrm{d}\vartheta}\right)^{n+1}\Delta\vartheta + 4\left(\frac{\mathrm{d}\phi}{\mathrm{d}\vartheta}\right)^{n+1}\Delta\vartheta + 2\left(\frac{\mathrm{d}^2\phi}{\mathrm{d}\vartheta^2}\right)^{n+1}\Delta\vartheta^2 - 2\left(\frac{\mathrm{d}^2\phi}{\mathrm{d}\vartheta^2}\right)^{n+1}\Delta\vartheta^2 + O(\Delta\vartheta^3)$$

and then

$$\left(\frac{\mathrm{d}\phi}{\mathrm{d}\vartheta}\right)^{n+1} \approx \frac{3\,\phi^{n+1} - 4\,\phi^{n} + \phi^{n-1}}{2\,\Delta\vartheta}$$

Special methods

The schemes seen so far are among those most frequently used in general-puropose CFD codes.

Temporal integration schemes aimed at particular applications or characterized by particular advantages:

- For time-dependent, highly dynamic simulations, like i.e. *LES, DES* and *DNS*, which require small values of the time-step, it might be convenient to integrate the (non linear) advective term with an *explicit* method, and adopt an *implicit* method for the (linear) viscous term, thus eliminating the necessity of sub-iterations within a time-step;
- For *LES, DES* and *DNS* simulations, it is more common to adopt high order temporal integration schemes, in particular when the spatial integration schemes are also of high order. This is more common for simple computational domains discretized with Cartesian or in general structured grids, where high order spatial discretization schemes are easy to implement.

Mixed explicit-implicit methods

Mixed *explicit-implicit* methods, for the convective and diffusion fluxes respectively, are commonly used for DNS and LES simulations in simple geometries.

- Second order Adams-Bashfort scheme for the convective term, and Gear (second order Euler backward) for the diffusive term

$$\frac{3\phi^{n+1} - 4\phi^n + \phi^{n-1}}{2\,\Delta\vartheta} = -\frac{1}{2}\left(3\,\nabla\cdot\left(\rho\mathbf{w}\phi^n\right) - \nabla\cdot\left(\rho\mathbf{w}\phi^{n-1}\right)\right) + \left(\nabla\cdot\left(\Gamma\nabla\phi\right)\right)^{n+1} + s^{n+1}$$

- Alternatively, Runge-Kutta scheme for the convective term and Crank-Nicolson or Gear scheme for the diffusive contribution.

Application to the generic transport equation

Two-dimensional case, first order Euler implicit scheme for time integration, and CDS spatial scheme for both convective and diffusive fluxes

$$A_P\phi_P^{n+1} + A_E\phi_E^{n+1} + A_W\phi_W^{n+1} + A_N\phi_N^{n+1} + A_S\phi_S^{n+1} = S_P$$

$$\begin{aligned}
\dot{m}_e &= \rho\, u_e\, \Delta y_j \\
A_E^c &= \dot{m}_e\, \lambda_{e,PE} \\
A_E^d &= -\frac{\Gamma_e\, \Delta y_j}{x_E - x_P} \\
A_E &= A_E^c + A_E^d \\
A_P &= \frac{\rho}{\Delta\vartheta}\Delta x_i\, \Delta y_j - (A_E + A_W + A_N + A_S) - s_P^{lhs}\, \Delta x_i\, \Delta y_j \\
S_P &= \left(s_P^{rhs} + \frac{\rho\,\phi^n}{\Delta\vartheta}\right) \Delta x_i\, \Delta y_j
\end{aligned}$$

- The unsteady term $\frac{\rho}{\Delta\vartheta}\Delta x_i\, \Delta y_j$ increases the weight of the A_P coefficient.
- Use of the unsteady formulation of the equations for the solution of steady problems.
- Explicit schemes do not require the solution of system(s) of equations, but the time-step is limited for stability reasons.

OUTLINE

Part IV

Solution of Linear Equation Systems

Solution of linear equation systems - 1

Generalities

- The result of the discretization process is a system of algebraic equations, which are linear or non-linear according to the nature of the partial differential equation (PDE) from which they are derived:
  - If the PDE is linear, a linear system of equations arises.
  - If the PDE is non-linear, the discretized equation must be solved by some iterative technique that involves guessing a starting solution, linearizing the equation about that solution, and improving the solution till convergence.
- In both cases, efficient methods for solving linear systems of algebraic equations are needed.
- The matrices derived from PDEs are always *sparse*, i.e. most of their elements are zero.
- Some methods for solving the equations that arise when structured grids are used will be described next. In this case all of the non-zero elements of the matrices lie on a small number of well-defined diagonals.
- Some of the methods are applicable to matrices arising from *unstructured grids* as well.

Solution of linear equation systems - 2

System of equations:

$$\mathbf{A}\Phi = \mathbf{S}$$

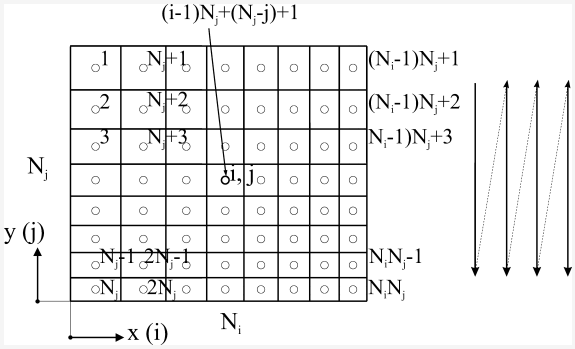
where **A** is the coefficient matrix, Φ is the unknowns vector, and **S** is the right hand-side (RHS) vector:

$$\mathbf{A} = \begin{bmatrix} A_{11} & A_{12} & \dots & A_{1N} \\ A_{21} & A_{22} & \dots & A_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ A_{N1} & A_{N2} & \dots & A_{NN} \end{bmatrix} \quad \Phi = \begin{Bmatrix} \phi_1 \\ \phi_2 \\ \vdots \\ \phi_N \end{Bmatrix} \quad \mathbf{S} = \begin{Bmatrix} S_1 \\ S_2 \\ \vdots \\ S_N \end{Bmatrix}$$

- The matrix **A** is *sparse*.
- The arrangement of non-zero elements within **A**, depends on how the unknowns are numbered.
- The most common approach, for structured grids, is to use the *lexicographic ordering*.

Solution of linear equation systems - 3

Lexicographic ordering:



For a 2D problem with $N_j = 3$ ed $N_i = 4$:

$$\mathbf{A} = \left[\begin{array}{ccc|ccc|ccc|ccc} A_{P1} & A_{S1} & & A_{E1} & & & & & & & & \\ A_{N2} & A_{P2} & A_{S2} & & A_{E2} & & & & & & & \\ & A_{N3} & A_{P3} & & & A_{E3} & & & & & & \\ \hline A_{W4} & & & A_{P4} & A_{S4} & & A_{E4} & & & & & \\ & A_{W5} & & A_{N5} & A_{P5} & A_{S5} & & A_{E5} & & & & \\ & & A_{W6} & & A_{N6} & A_{P6} & & & A_{E6} & & & \\ \hline & & & A_{W7} & & & A_{P7} & A_{S7} & & A_{E7} & & \\ & & & & A_{W8} & & A_{N8} & A_{P8} & A_{S8} & & A_{E8} & \\ & & & & & A_{W9} & & A_{N9} & A_{P9} & & & A_{E9} \\ \hline & & & & & & A_{W10} & & & A_{P10} & A_{S10} & \\ & & & & & & & A_{W11} & & A_{N11} & A_{P11} & A_{S11} \\ & & & & & & & & A_{W12} & & A_{N12} & A_{P12} \end{array}\right]$$

Solution of linear equation systems - 4

The matrix **A** is characterized by a *polydiagonal* structure, also called *block-diagonal* or *block-tridiagonal*.

- It is not necessary to store the entire matrix.
- For these type of matrices there are efficient iterative solution algorithms and, for particular cases, also direct solution algorithms.
- The structure of the matrix does not change – some terms are treated using a *deferred correction approach* – for curvilinear grids.
- The structure of the matrix is maintained, within each block, using multi-block structured grids.
- For 1D problems, the matrix is *tridiagonal*, i.e. only the main diagonal, the above diagonal and the below diagonal are different from zero.
- The structure of the matrix, again sparse, changes when using general unstructured grids.
- Distinction between *direct* and *iterative* methods:
  - For the former, the solution is obtained by a pre-determined number of operations, which is function of the number of unknowns.
  - For iterative methods, the solution is obtained using successive iterations, starting from a *guessed* (tentative) solution and the overall number of operations is unknown, since it depends on several factors.

Gauss Elimination - 1

Among the direct methods, the simplest one is the Gauss elimination. N is the total number of unknowns and is equal to $N_i \times N_j$ for 2D problems, and $N_i \times N_j \times N_k$ in 3D.

- We start by eliminating A_{21} from matrix **A**:
  - This is accomplished by multiplying the first equation (first row of the matrix) by A_{21}/A_{11} and subtracting it from the second row or equation;
  - In the process, all the other elements of the second row are modified, as well as the second element S_2 of the RHS.
- Then we multiply the first row by A_{31}/A_{11} and subtracting it from the third row.
- We proceed in the same way for all the other elements of the first column of matrix **A**

$$\begin{aligned} A'_{ij} &= A_{ij} - \frac{A_{i1}}{A_{11}} A_{1j} \\ S'_i &= S_i - \frac{A_{i1}}{A_{11}} S_1 \end{aligned} \qquad i = 2, \ldots N,\ j = 1, \ldots N$$

Gauss Elimination - 2

- After this first passage, the matrix **A** is modified as

$$\mathbf{A}' = \begin{bmatrix} A_{11} & A_{12} & \ldots & A_{1N} \\ 0 & A'_{22} & \ldots & A'_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & A'_{N2} & \ldots & A'_{NN} \end{bmatrix}$$

- It follows that none of the modified equations $2, 3, \ldots, N$ contain the variable ϕ_1.
- Then we proceed, in the same way, to eliminate the variable ϕ_2 in this reduced ($(N-1)$ unknowns) system of equations.
- We continue in the same way for the columns $3, 4 \ldots, N-1$.
- In general terms:

$$\begin{aligned} A'_{ij} &= A'_{ij} - \frac{A'_{ik}}{A'_{kk}} A_{kj} \\ S'_i &= S'_i - \frac{A'_{ik}}{A'_{kk}} S'_k \end{aligned} \qquad k = 1, \ldots N-1, i = k+1, \ldots N,\ j = k \ldots, N$$

Gauss Elimination - 3

- All of the elements, except those in the first row, differ from those in the original matrix **A**.
  - As the elements of the original matrix will never be needed again, it is efficient to store the modified elements in place of the original ones.
  - In the uncommon case requiring that the original matrix be saved, a copy can be created prior to starting the elimination procedure.
- The elements on the RHS of the equation, S_i, are also modified in this procedure.
- This portion of the algorithm just described is called *forward elimination*.
- The elements A'_{ik}/A'_{kk} are called *multiplicative factors*, and the resulting matrix **U** is an *upper triangular matrix*

$$\mathbf{U} = \begin{bmatrix} A_{11} & A_{12} & \dots & A_{1N} \\ 0 & A'_{22} & \dots & A'_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & A'_{NN} \end{bmatrix} \qquad \mathbf{S}' = \begin{Bmatrix} S_1 \\ S'_2 \\ \vdots \\ S'_N \end{Bmatrix}$$

Gauss Elimination - 4

- The upper triangular system of equations resulting from *forward elimination* is easily solved: the last equation contains only one variable, ϕ_N, and is immediately solved:

$$\phi_N = \frac{S'_N}{A'_{NN}}$$

- The next to last equation contains only and ϕ_N and ϕ_{N-1} and, once ϕ_N is known, it can be solved for ϕ_{N-1}.
- Proceeding upward in this manner, each equation is solved in turn; the ith equation yields ϕ_i

$$\phi_i = \frac{S'_i - \sum_{j=i+1}^{N} A'_{ij}\,\phi_j}{A'_{ii}} \qquad i = N-1, \dots 1$$

- The right hand side of the previous equation is easily computed because all of the ϕ_j appearing in the sum have already been evaluated. In this way, all of the variables may be computed.

Gauss Elimination - 5

- The part of the Gauss elimination algorithm which starts with the triangular matrix and computes the unknowns is called *back substitution*.
- It can be proved that, for large values of N, the number of operations required to solve a linear system of N equations by Gauss elimination is proportional to $N^3/3$. However, the back substitution requires only $N^2/2$ arithmetic operations and is much less costly than the forward elimination.
- The high cost of Gauss elimination justifies the search for more efficient special solvers for matrices such as the *sparse* ones arising from the discretization of differential equations.
- The Gauss elimination, therefore, is rarely - if any - applied in CFD problems.

LU Decomposition - 1

One variant of the Gauss elimination of interest in CFD is the LU decomposition.

- The main idea is to factorize the original matrix **A** into the product of lower (**L**) and upper (**U**) triangular matrices

$$\mathbf{A} = \mathbf{L}\,\mathbf{U}$$

- To make the factorization unique, we require that the diagonal elements of the matrix **L**, L_{ii}, all be unity (or equivalently, we could require the diagonal elements of **U**, U_{ii}, to be unity).
- The upper triangular matrix **U** is the one produced by the *forward* phase of Gauss elimination. while the elements of **L** are the multiplicative factors, e.g. A_{ji}/A_{ii}, used in the elimination process. This allows to construct the factorization with only a minor modification of the Gauss elimination.
- With this factorization, the solution of the system of equations can be obtained in two stages. Defining

$$\mathbf{U}\,\Phi = \mathbf{Y} \tag{2}$$

the original system of equations becomes

$$\mathbf{L}\,\mathbf{Y} = \mathbf{S} \tag{3}$$

Once the system (**??**) has been solved for **Y**, eq. (**??**), which is practically identical to the triangular system solved in the back-substitution phase of Gauss elimination, can be solved for Φ.

LU Decomposition - 2

The importance of the LU factorization derives from the following observations:

1. The factorization can be performed without knowledge of the RHS vector **S**. It then follows that, if many systems involving the same matrix have to be solved, like e.g. some unsteady problems, a noteworthy saving can be obtained by performing the factorization first once for all.
2. Variations of LU factorization are the basis of some of the better iterative methods of solving systems of linear equations.

Particular methods

There are numerous special cases of linear systems, and some of them are of relevance in some CFD problems:

- They are limited to simple geometries, e.g. *parallelepiped channels, cylinders, spheres et.*
- They are particularly efficient for repeated solution of the same system matrix with different RHS.

The algorithms of major interest in CFD and Numerical Heat transfer are

- Algorithms for *tridiagonal*and *cyclic tridiagonal* linear systems: Thomas, Temperton et. The number of operations scales linearly with N, rather than N^3 for the Gauss algorithm.
- Algorithms for *uniform* Cartesian, cylindrical and spherical 2D and 3D grids: they are based on FFT (Fast Fourier Transform) and cyclic reduction. In this case the number of operations, excluding the factorization phase, is of the order of $N \times (\ln_2 N)$.
- Algorithms for *non-uniform* Cartesian, cylindrical and spherical 2D and 3D grids: special matrix decomposition, for which the number of operations, again for just the solution phase, is of the order $N^{4/3}$.

Iterative methods - 1

The use of iterative methods for the solution of linear system of equations in CFD is justified by the following observations:

- With the LU method, although the matrix **A** is *sparse*, the matrices **L** and **U** are *full*.
- Discretization errors are usually far larger than rounding errors, so there is no reason to solve the system with extreme accuracy.
- In most practical and engineering problems the system of equations are very large and direct methods require more memory than iterative methods.

The idea behind iterative methods is to start from a guessed (tentative) solution, and then improve it until the desired accuracy: after n iterations we will have an approximate solution $\mathbf{\Phi}^n$, which does not satisfy the original equation exactly. Instead, there will be a non-zero residual $\mathbf{r}^n$

$$\mathbf{A}\mathbf{\Phi}^n = \mathbf{S} - \mathbf{r}^n$$

By subtracting this equation from the original system of equations, we obtain a relation between the *convergence* (or *iteration*) *error* ϵ^n defined by

$$\epsilon^n = \mathbf{\Phi} - \mathbf{\Phi}^n$$

and the residual $\mathbf{r}^n$:

$$\mathbf{A}\epsilon^n = \mathbf{r}^n$$

The purpose of the iteration method is to drive the residual to zero; in the process, also ϵ becomes zero.

Iterative methods - 2

An iterative scheme for a linear system can be written as

Iterative scheme:

$$\mathbf{M}\mathbf{\Phi}^{n+1} = \mathbf{N}\mathbf{\Phi}^n + \mathbf{B}$$

An obvious requirement for an iterative scheme is that the converged solution must satisfy the original equation. Since, by definition, at convergence $\mathbf{\Phi}^{n+1} = \mathbf{\Phi}^n = \mathbf{\Phi}$, we must have

$$\mathbf{A} = \mathbf{M} - \mathbf{N} \qquad \text{e} \qquad \mathbf{B} = \mathbf{S}$$

or, more generally

$$\mathbf{P}\,\mathbf{A} = \mathbf{M} - \mathbf{N} \qquad \text{e} \qquad \mathbf{B} = \mathbf{P}\,\mathbf{S}$$

where **P** is a non-singular *pre-conditioning matrix*.

Alternative version of the iterative scheme:

$$\mathbf{M}\left(\mathbf{\Phi}^{n+1} - \mathbf{\Phi}^n\right) = \mathbf{B} - (\mathbf{M} - \mathbf{N})\,\mathbf{\Phi}^n \qquad \text{or} \qquad \mathbf{M}\delta^n = \mathbf{r}^n$$

where $\delta^n = \mathbf{\Phi}^{n+1} - \mathbf{\Phi}^n$ is the *correction* or *update*, and is an approximation of the convergence error.

Iterative methods - 3

- For an iterative method to be effective, solving the modified system must be cheap and the method must converge rapidly, i.e. with a low number of iterations.
- The speed of convergence is a functions of the *spectral radius* λ_1, defined as the magnitude of the largest eigenvalue of the matrix $\mathbf{M}^{-1}\mathbf{N}$.
- An approximate expression of the required number of iterations is given by

$$n \approx \frac{\ln\left(\frac{\delta}{a_1}\right)}{\ln\ \lambda_1}$$

  where a_1 is a constant and δ is the tolerance required for the iteration error, e.g. convergence is defined as the reduction of the iteration error below δ.
- We see that, if the spectral radius λ_1 is very close to unity, the iterative procedure will converge very slowly.

Basic methods - 1

Jacobi method

M is a diagonal matrix whose elements are the diagonal elements of A.

$$\phi_P^{n+1} = \frac{S_P - A_E\phi_E^n - A_W\phi_W^n - A_N\phi_N^n - A_S\phi_S^n}{A_P}$$

- It may be proved that this method requires a number of iterations proportional to the *square* of the number of VCs in one direction.
- This means that this method is rather slow and therefore it is rarely used.
- This method, however, can be easily *parallelized*, and therefore it is used for specific applications, i.e. *video games, special effects* et.

Basic methods - 2

Jacobi method - matrix-based formulation

The matrix **A** can be decomposed into a diagonal component **D**, a lower triangular part **L** and an upper triangular part **U**

$$\mathbf{D} = \begin{bmatrix} A_{11} & 0 & \dots & 0 \\ 0 & A_{22} & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & A_{NN} \end{bmatrix}, \quad \mathbf{L} = \begin{bmatrix} 0 & 0 & \dots & 0 \\ A_{21} & 0 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ A_{N1} & A_{N2} & \dots & 0 \end{bmatrix}, \quad \mathbf{U} = \begin{bmatrix} 0 & A_{12} & \dots & A_{1N} \\ 0 & 0 & \dots & A_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & 0 \end{bmatrix}$$

The solution is then obtained iteratively via

$$\mathbf{\Phi}^{n+1} = \mathbf{D}^{-1} \left(\mathbf{S} - (\mathbf{L} + \mathbf{U}) \, \mathbf{\Phi}^{n} \right)$$

Basic methods - 3

SOR - Successive Over-Relaxation method

The matrix **M** is the lower triangular portion of **A**.
Following the lexicographic ordering

$$\phi_P^{n+1} = (1-\omega)\, \phi_P^n + \omega \, \frac{S_P - A_E \phi_E^n - A_W \phi_W^{n+1} - A_N \phi_N^{n+1} - A_S \phi_S^n}{A_P}$$

where ω is the over-relaxation factor, necessary to accelerate the convergence.

- The SOR method is more efficient than the Jacobi method, since it uses the newly updated variables as soon as they are available.
- When the optimum over-relaxation factor is used (optimal values of ω are in the range $1.5 \leq \omega \leq 1.85$), the number of iterations is proportional to the *number* of VCs in one direction.

Basic methods - 4

Gauss-Seidel method

It corresponds to the SOR method with $\omega = 1$:

$$\phi_P^{n+1} = \frac{S_P - A_E\phi_E^n - A_W\phi_W^{n+1} - A_N\phi_N^{n+1} - A_S\phi_S^n}{A_P}$$

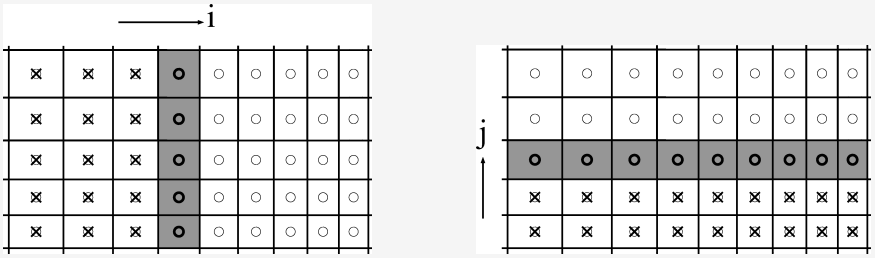
It converges twice as fast as the Jacobi method, but it is significantly slower than the optimal SOR method.

Basic methods - 5

SLOR - Successive Line Over-Relaxation method

Iterative update of the variable of an entire line - row or column - of VCs.
The matrix **M** is constituted by the subset of the Matrix **A** relative to the variables on the line.
E.g. proceeding along columns for increasing values of i

$$A_P\phi_{i,j}^{n+1} + A_N\phi_{i,j+1}^{n+1} + A_S\phi_{i,j-1}^{n+1} = \left[S_P - A_E\phi_{i+1,j}^n - A_W\phi_{i-1,j}^{n+1}\right]$$



- It results in a tridiagonal system for each line.
- Possible use of a over-relaxation factor ω.
- Limited to *structured* grids.

Some notes on conjugate gradient methods

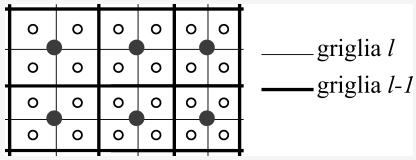
- The conjugate gradient method is based on the fact that it is possible to minimize a function, with respect to several directions simultaneously, while searching in one direction at a time. This is made possible by a smart choice of directions.
- The conjugate gradient methods can be classified according to the type of linear systems they are aimed to solve:
  1. *Symmetric systems*: CG (Conjugate Gradient) and ICCG (Incomplete Cholesky Conjugate Gradient).
  2. *Unsymmetric systems*: BCG (BiConjugate Gradient), CGS (Conjugate Gradient Squared), CGSTAB (CGS Stabilized) and GMRES (Generalized Minimal Residual).
- It can be shown that the rate of convergence of these method depends on the *condition number* κ of the matrix

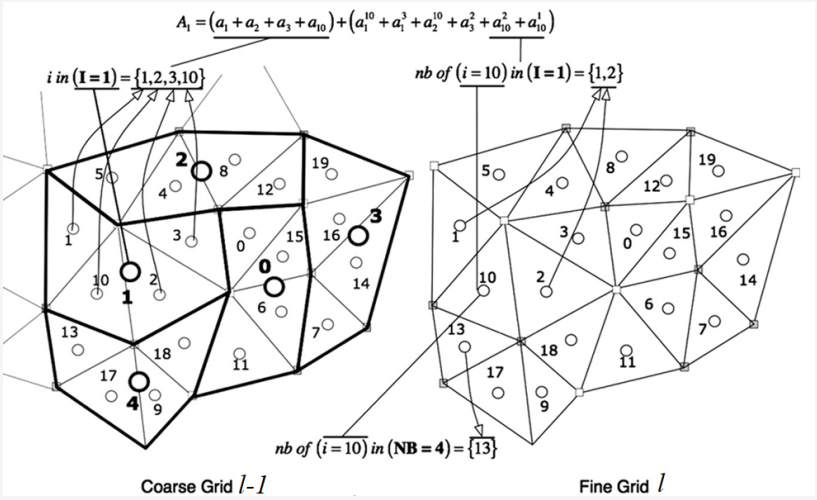
$$\kappa = \frac{\lambda_{max}}{\lambda_{min}}$$

  Where λ_{max} and λ_{min} are the largest and smallest eigenvalues of the original matrix, respectively.
- The condition numbers of matrices of CFD problems are usually approximately the square of the maximum number of FVs in any direction. With 100 FVs in each direction, the condition number should be about 10^4 and the standard conjugate gradient method would converge very slowly.
- These methods can be improved by replacing the problem by another one with the same solution but a smaller condition number. This is called *preconditioning*.

Some notes on multigrid methods - 1

They are based on the use of several grids: the *original computational grid* (level l), and a sequence of progressively *coarser grids* ($l-1, l-2, \ldots, 1$), where typically, for FV discretization, each cell of the coarser grids is obtained by *summation* (*agglomeration*) of more cells of the closest finer grid.





Some notes on multigrid methods - 2

The idea behind multigrid methods, limiting for simplicity the discussion for the case of two grids l and $l-1$, stems from the following observations:

- All iterative methods are effective in the reduction of errors (residuals) of small *spatial* wavelength, or high frequency errors. These are errors whose wavelength is comparable to the cell size.
- The reduction of small wavelength errors occurs in the first iterations.
- Conversely, the reduction of errors at increasingly higher wavelengths (lower frequencies) occurs more slowly, and in fact, after the first few iterations, the convergence speed slows down.

Some notes on multigrid methods - 3

- *Low frequency (large wavelength)* errors are reduced, more economically, on a *coarser* grid (*interpolation* or summation of the residuals).
- Once the solution is (economically) obtained on the coarse grid, this is *injected* or summed on the fine grid.
- The iterations on the coarse grid are computationally less expensive.
- Faster convergence on the coarse grid, since the error has a (relative) higher frequency.
- The advantage increases using more grids, or levels.
- Ideally, the number of *equivalent* iterations, and therefore the computing time, varies linearly with the number N of unknowns.

Multigrid methods of major interest in CFD, in particular for the FV method:

1. *Additive* multigrid - ACM (Additive Correction Multigrid).
2. *Algebraic* multigrid - AGM (AlGebraic Multigrid).

The iterative solver to be used on the various grids, in this case called *smoother*, can be one of the iterative methods already described.

OUTLINE

Part V

Thermal-fluid problems - FV approach

THERMAL-FLUID PROBLEMS - Overview

- FV: common approach for the numerical solution of thermal-fluid problems.
- Large diffusion of commercial and open source packages based on FV method.
- The algebraic equations obtained from the discretization of Navier-Stokes equations - and other transport/conservation equations - could be solved *simultaneously* or *sequentially*. For the latter, the overall solution proceeds, for each variable, by *freezing* the other variables at the previous time-step or iteration.
- The first FV codes used the sequential, or so-called *segregated* approach, and only Cartesian structured grids.
- Extension of the FV codes to structured curvilinear grids, single-block and later multi-block.
- Nowadays FV for general unstructured grids:
  - FVs constituted by arbitrary *polyhedra*.
  - Automated grid generation for (very) complex geometries.
  - Import of CAD geometries.
  - Adaptive grids.
  - Availability of *segregated* and *coupled* solution methods.

THERMAL-FLUID PROBLEMS - FV packages

Non-exhaustive list of FV packages (in alphabetical order)

Commercial

AVL FIRE™ M	AVL List GmbH.
CFX	ANSYS, Inc.
CONVERGE	Convergent Science.
Fidelity CFD	Cadence Design Systems, Inc.
FLOW-3D	FLOW SCIENCE, INC.
Fluent	ANSYS, Inc.
ICFD++	Metacomp Technologies Inc.
PHOENICS	Concentration, Heat and Momentum (CHAM).
SC/Tetra	Software Cradle Co., Ltd.
STAR-CCM+	Siemens Digital Industries Software.

Open source

Basilisk	`http://basilisk.fr/`
code_saturne	EDF
OpenFOAM	ESI Group | OpenFOAM Foundation.
SU2	SU2 Foundation.

Cloud-based CFD
(SaaS - Software as a Service)

SIMSCALE	`https://www.simscale.com/`
luminary	`https://www.luminarycloud.com/`
AirShaper	`https://airshaper.com/`
SYMULA	`https://symulacfd.com/`

THERMAL-FLUID PROBLEMS - Overview of solution methods

- With the FVM it is common to use iterative methodologies for steady problems, and implicit time-integration methods for unsteady problems.
- *SIMPLE* (Semi-Implicit Method for Pressure-Linked Equations), and similar SIMPLE-like methods like e.g. SIMPLER, SIMPLEC, SIMPLEST *et.*:
  - They do not differ substantially from *Projection* methods.
  - For simplicity we will refer, in the following, to *projection* methods.
- *Projection methods:* derivation of a differential equation for the pressure from the mass conservation constraint.
- *Projection:* the tentative (approximated) velocity field obtained in the first step of the procedure, is then *projected*, taking into account the pressure correction, in a *divergence-free*, or *solenoidal*, vector field.
- Differences between the Navier-Stokes equations and the generic transport equation: consequences of the possible locations of velocity and pressure variables on the grid.

THERMAL-FLUID PROBLEMS - solution of the Navier-Stokes equations

The Navier-Stokes and continuity equations, in conservative form and in two-dimensional Cartesian coordinates are

$$\frac{\partial}{\partial \vartheta}(\rho u) + \frac{\partial}{\partial x}(\rho u u) + \frac{\partial}{\partial y}(\rho v u) = -\frac{\partial p}{\partial x} + \frac{\partial}{\partial x}\left(\mu \frac{\partial u}{\partial x}\right) + \frac{\partial}{\partial y}\left(\mu \frac{\partial u}{\partial y}\right) - \rho_0 \beta (t - t_0) g_x$$

$$\frac{\partial}{\partial \vartheta}(\rho v) + \frac{\partial}{\partial x}(\rho u v) + \frac{\partial}{\partial y}(\rho v v) = -\frac{\partial p}{\partial y} + \frac{\partial}{\partial x}\left(\mu \frac{\partial v}{\partial x}\right) + \frac{\partial}{\partial y}\left(\mu \frac{\partial v}{\partial y}\right) - \rho_0 \beta (t - t_0) g_y$$

$$\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} = 0$$

- The source term, in particular the pressure gradient, requires special attention.
- The buoyancy term does not pose particular problems.
- The convective term is linearized using velocity values, in the definition of the flow rate through the FV face, at the previous time-step or iteration.

Discretization of the pressure term

As already seen, e.g. in the x-component of the momentum equation, we have the pressure gradient term

$$-\int_V \frac{\partial p}{\partial x} \, \mathrm{d}V$$

In FV methods, this pressure term is usually treated as a surface force (conservative approach)

$$-\int_V \frac{\partial p}{\partial x} \, \mathrm{d}V = -\int_V \nabla p \cdot \mathbf{i} = -\int_A p \mathbf{i} \cdot \, \mathrm{d}A$$

The methods described previously for the approximation of surface integrals can then be used. As we will see next, the treatment of this term and the arrangement of variables on the grid play an important role in assuring the computational efficiency and accuracy of the numerical solution method.
Alternatively, the pressure term could be treated *non-conservatively*, by retaining the above integral in its volumetric form

$$-\int_V \frac{\partial p}{\partial x} \, \mathrm{d}V = -\int_V \nabla p \cdot \mathbf{i} \, \mathrm{d}V$$

In this case, the derivative needs to be approximated at one or more locations within the CV. This *non-conservative* approach introduces a *global non-conservative error*. The difference between the two approaches is significant only in the FV methods.

Variable arrangement on the grid - 1

Let's consider the first momentum equation for 2D problems.

- We have already seen that with the FV method the pressure gradient term is usually treated as a surface force (conservative approach)

$$-\int_V \frac{\partial p}{\partial x}\,\mathrm{d}V = -\int_A p\,\mathbf{i}\cdot\,\mathrm{d}A = -\left[\int_{A_e} p\,\mathrm{d}A - \int_{A_w} p\,\mathrm{d}A\right] \approx \Delta y_j\,(p_w - p_e)$$

- Using a linear interpolation (CD - Central Difference), we have

$$\Delta y_j\,(p_w - p_e) \approx \Delta y_j\left(\frac{p_W + p_P}{2} - \frac{p_P + p_E}{2}\right) = \Delta y_j\left(\frac{p_W - p_E}{2}\right)$$

- It therefore results that the pressure gradient term is computed using a grid that is two times coarser than the original grid.
- This leads to a weak coupling between the velocity and pressure fields: *checkerboard pressure field*.
- This nonphysical situation may lead to highly inaccurate results or even to a (catastrophic) divergence of the calculation.
- It is therefore mandatory to avoid this problem.

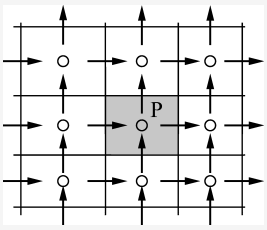
Variable arrangement on the grid - 2

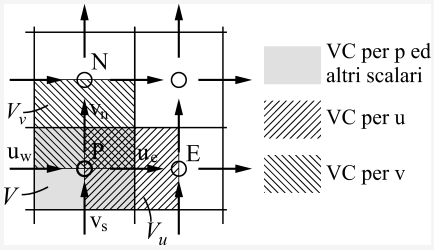
A *possible* non-physical checkerboard pressure field

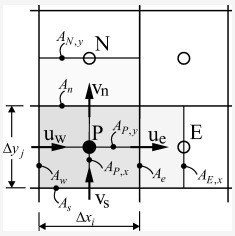
1000	-400	1000	-400	1000	-400
-400	1000	-400	1000	-400	1000
1000	-400	1000	-400	1000	-400
-400	1000	-400	1000	-400	1000
1000	-400	1000	-400	1000	-400
-400	1000	-400	1000	-400	1000

Variable arrangement on the grid - 3

- Use of grids, for the velocity components, different from the principal one used for pressure and any other scalar quantity.
- These different grids are obtained by translating (*staggering*) the VCs of the original (principal) grid, for every velocity component and along the corresponding direction, of a quantity equal to half grid size.







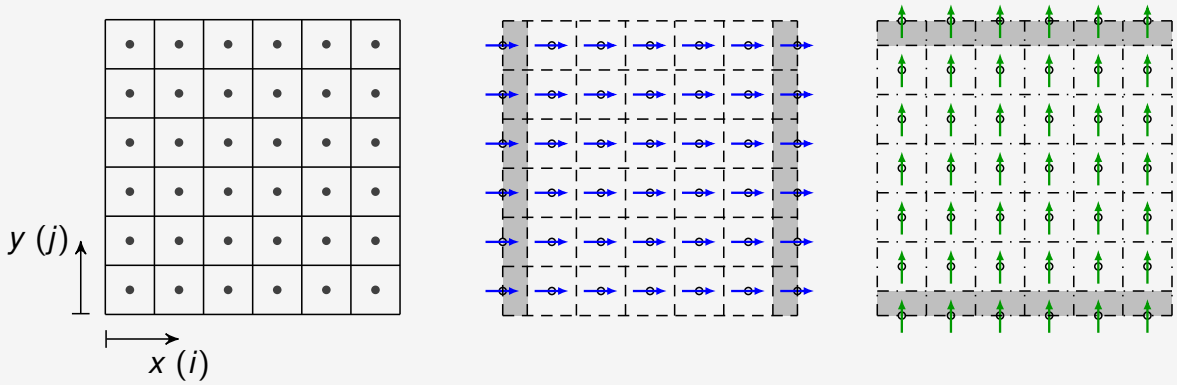
- With these variables arrangement - *staggered grid* - the integration of the pressure gradient term for u_e gives

$$-\int_{V_e} \frac{\partial p}{\partial x}\, dV = -\left[\int_{A_E} p\, dA - \int_{A_P} p\, dA\right] \approx \Delta y_j\,(p_P - p_E)$$

- Absence of checkerboarded pressure field: the pressure gradient is computed using the pressure values in two adjacent cells.

Variable arrangement on the grid - 4

Staggered grids

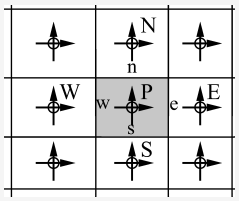


p (and all other scalars) *u* *v*

- Note the *half-cells* (shaded) for both *u* (x-velocity component) and *v* (y-velocity component).
- A similar cell pattern holds for *w* (z-velocity component).

Variable arrangement on the grid - 5

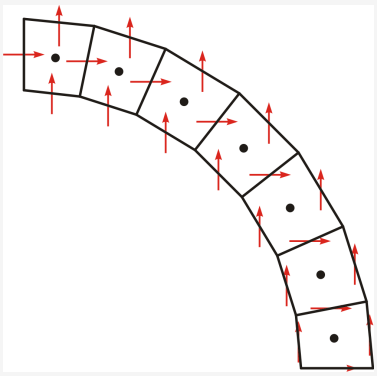
- *Staggered grids* in the FV method correspond to *unequal order interpolation*, i.e. shape function of lower order for the pressure, in the Finite Element (FE) method.
- Further advantages of the staggered grids
  - Additional terms of the equations, which would otherwise require interpolation, can be evaluated, with second order of accuracy, in a simpler way.
  - Conservation of kinetic energy is guaranteed.
- Difficulties in using staggered *curvilinear grids*, and even more for general *unstructured grids*.
- Use of *co-located arrangement* of the variables:



- The conservation property for the kinetic energy, makes staggered grids particularly suited for DNS (Direct Numerical Simulation) and LES (Large Eddy Simulation) of turbulent flows.

Difficulties in using staggered grids - 1

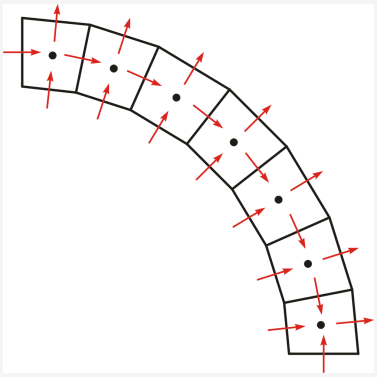
- The use of *staggered grids*, as already seen, solves the problem of pressure *checkerboarding* for structured Cartesian grids.
- However, *staggering* is not easy for curvilinear and generally unstructured grids, as shown in the figure:



In this case, in fact, the Cartesian components of the velocity, u and v, placed again on the centroids of the faces, do not represent anymore the *mass fluxes* normal to these faces.

Difficulties in using staggered grids - 2

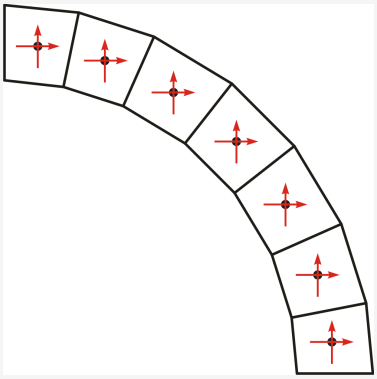
- This difficulty could be circumvented using *contravariant* velocity components, locally oriented along the grid:



- Although this approach was used - and is still used - for structured curvilinear grids, it has at least three drawbacks:
  - ➢ The conservation equations, and their corresponding discrete formulation, become rather complex, in particular in 3D;
  - ➢ It is necessary to store the geometric information (metric coefficients) for 3 (in 2D) or 4 (in 3D) grids;
  - ➢ Its extension to arbitrary unstructured grids, if possible, presents quite a few difficulties.

Difficulties in using staggered grids - 3

- The simplest solution is to adopt the co-located arrangement of *all* variables:



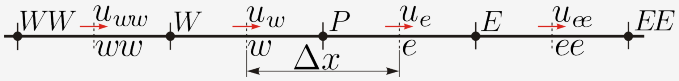
- However, the problem of weak pressure-velocity coupling arises, and the resulting possible *checker-boarded)* pressure field.

What to do ?

- In what follows we will see a possible strategy to avoid this inconvenience, limiting the discussion to the Cartesian case only, leaving the extension to the case of arbitrary unstructured grids to the literature.

Co-located grids - 1

- Before moving on to how to deal with velocity-pressure coupling on co-located grids, let us briefly recall how the staggered arrangement of variables works, which for simplicity we illustrate with reference to the momentum equation of magnitude along x, on a 1D staggered grid:



The discretized momentum equation along x for the velocity u_e can be written, in compact form and using the geographical notation:

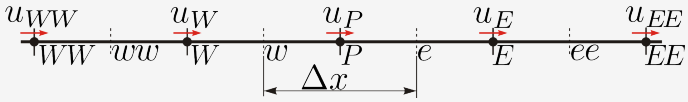
$$A^u_{p_e} u_e + \sum_{nb} A^u_{nb} u_{nb} = \Delta y_j \, (p_P - p_E) + S^u_{p_e}$$

where u_e represents the velocity component along x placed on the face e.

- The pressure difference that contributes to the momentum equation is $(p_P - p_E)$, which straddles the node e:
  - ➢ the velocity u_e, we recall, lies right on the face e of the main control volume;
  - ➢ the continuity equation is applied to the control volume of the main grid to derive the pressure correction equation, in the projection method or in the SIMPLE-based methods.

Co-located grids - 2

- Let us now consider the co-located arrangement of the variables also illustrated here, for simplicity, for a 1D grid:



The discretized x-momentum equation for the velocity u_P is:

$$A^u_{P_P} u_P + \sum_{nb} A^u_{nb} u_{nb} = \Delta y_j \, (p_w - p_e) + S^u_P$$

- We had already seen that, using a linear interpolation, we obtain:

$$\Delta y_j \, (p_w - p_e) \approx \left(\frac{p_W + p_P}{2} - \frac{p_P + p_E}{2} \right) = \Delta y_j \left(\frac{p_W - p_E}{2} \right)$$

which when replaced in the previous one gives:

$$A^u_{P_P} u_P + \sum_{nb} A^u_{nb} u_{nb} = \boxed{\frac{\Delta y_j}{2} \, (p_W - p_E)} + S^u_P$$

Co-located grids - 3

- Explicitly expressing u_P from the previous equation we have:

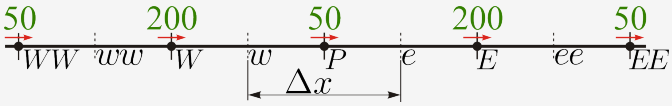
$$u_P = \frac{S_P^u - \sum_{nb} A_{nb}^u u_{nb}}{A_{P_P}^u} + \boxed{\frac{d_P^u}{2}(p_W - p_E)}$$

  with $d_P = \Delta y_j / A_{p_P}^u$. Similarly for the velocity u_E:

$$u_E = \frac{S_E^u - \sum_{nb} A_{nb}^u u_{nb}}{A_{P_E}^u} + \boxed{\frac{d_E^u}{2}(p_P - p_{EE})}$$

  with $d_E = \Delta y_j / A_{P_E}^u$.

- Now consider a *rapidly varying* pressure field as in figure:



  - ➢ One would expect that pressure differences between nodes P and E, and between nodes W and P, would give rise to *flows entering* the control volume at P through faces e and w.
  - ➢ However, as seen, the pressure difference $(p_W - p_E)$ is zero, and similar conclusions are reached by considering the equations for the cells W and E.
  - ➢ This is the *checkerboarding* problem, responsible for unphysical solutions as a consequence of linear interpolation for pressure differences in the co-located arrangement of variables.

Co-located grids - 4

- Recall that the velocities on the faces e and w are used in the Poisson equation for calculating the pressure correction:
  - ➢ As can be easily understood, if the pressure difference is not adequately *represented* in the velocities on the faces, it is reasonable to expect that the pressure correction equation *will not* be able to provide a correct solution.
  - ➢ For example, the velocity u_e in the co-located arrangement is obtained using the velocities u_E and u_P given by the relations just seen.
  - ➢ It follows that the expression for u_e will *not* contain the pressure gradient across the faces of the control volume - $(p_W - p_P)$ and $(p_P - p_E)$ - but rather the gradient evaluated on the *coarse* (or double) grid - $(p_W - p_E)$ and $(p_P - p_{EE})$, with the consequence that the resulting solution may have an oscillatory character.
- To overcome this problem, several authors in 1981 (Hsu; Prakash; Rhie) reported the first successes in the implementation of schemes based on the pressure correction equation on co-located grids.
- The derivation reported here follows the original work of Rhie and Chow (1983).

Rhie-Chow interpolation - 1

- The Rhie-Chow interpolation consists of including a higher order term to express u_e in the pressure correction equation:

Rhie-Chow interpolation

$$u_e = \left[\frac{u_P + u_E}{2}\right] + \frac{1}{2}\left(d_P + d_E\right)\left(p_P - p_E\right) - \frac{1}{4}d_P\left(p_W - p_E\right) - \frac{1}{4}d_E\left(p_P - p_{EE}\right)$$

  - ➢ The velocity u_e is now related to the pressure difference $(p_P - p_E)$ across the face.
  - ➢ The first term on the right is the average value of the velocity in e.
  - ➢ By comparing this relation with the sum of the expressions seen before for u_E and u_P, it is easy to verify that the last two terms on the right remove the unwanted effects due to the pressure differences between non-adjacent cells: in fact, they are replaced, in the second term on the right, by a contribution that takes into account the pressure difference $(p_P - p_E)$ across the face.

- It remains to be proved that the three additional terms in the expression of u_e just seen represent a *high-order term*, which therefore does not affect the accuracy of the solution.

Rhie-Chow interpolation - 2

- Assuming for simplicity, but without detracting from the generality of what follows, that the "'d"' terms are constant, we rewrite the expression of u_e as follows

$$\begin{aligned} u_e &= \frac{u_P + u_E}{2} + d\left(p_P - p_E\right) - \frac{1}{4}d\left(p_W - p_E\right) - \frac{1}{4}d\left(p_P - p_{EE}\right) \\ &= \frac{u_P + u_E}{2} + \boxed{\frac{d}{4}\left[3p_P - 3p_E + p_{EE} - p_W\right]} \end{aligned}$$

- To appreciate the significance of the additional pressure term, consider the third derivative of the pressure across the face e:

$$\begin{aligned} \left.\frac{\partial^3 p}{\partial x^3}\right|_e &= \frac{\partial}{\partial x}\left(\frac{\partial^2 p}{\partial x^2}\right) \approx \frac{1}{\Delta x}\left[\left(\frac{\partial^2 p}{\partial x^2}\right)_E - \left(\frac{\partial^2 p}{\partial x^2}\right)_P\right] \\ &\approx \frac{1}{\Delta x}\left[\frac{p_{EE} - 2p_E + p_P}{\Delta x^2} - \frac{p_E - 2p_P + p_W}{\Delta x^2}\right] \\ &\approx \frac{1}{\Delta x^3}\left[p_{EE} - 3p_E + 3p_P - p_W\right] \end{aligned}$$

Rhie-Chow interpolation - 3

- From what we have seen

$$(p_{EE} - 3p_E + 3p_P - p_W) \approx \frac{\partial^3 p}{\partial x^3}\Delta x^3$$

- Comparing this expression with that of the additional pressure term of the Rhie-Chow interpolation, we find that:

Order of the additional pressure term

$$u_e = \frac{u_P + u_E}{2} + \frac{d}{4}\left.\frac{\partial^3 p}{\partial x^3}\right|_e \Delta x^3$$

- This shows that the Rhie-Chow interpolation consists of adding a third-order term to the pressure gradient:
  - ➢ Since the method is *overall* at most accurate to second order, this addition does not compromise the accuracy of the solution.
  - ➢ Its beneficial effects consist in damping of spurious oscillations caused by the co-located arrangement of the variables, and for this reason it is usually called, in the literature, *pressure smoothing term* or *added dissipation term* or even *pressure redistribution term*.
  - ➢ The damping is due to the recovered connection between the pressure differences across the cell faces, and the velocity on the faces that appears in the pressure correction equation (continuity).

Rhie-Chow interpolation - 4

- The Rhie-Chow interpolation procedure just seen can be extended, without particular difficulty, to curvilinear structured grids and to arbitrary unstructured grids.
- It has proven to be particularly effective for structured and unstructured co-located grids; for this reason it is adopted, albeit with modifications and variations, in several commercial and Open Source FV programs.
- We recall again that interpolation should be used only in evaluating the velocity components on the faces to determine the continuity error in the pressure correction equation, and not in calculating the fluxes in the momentum equation or in the relations for calculating the velocity corrections.
- Although the addition of a third order term does not affect the accuracy of the solution, the energy conserving property of the scheme is destroyed in this process, introducing the possibility of instability.

Segregated methods - 1

The following assumptions apply:

- Two-dimensional flow with constant thermophysical properties.
- Oberbeck-Boussinesq approximation for the buoyancy term.
- Computation of the tentative (estimate) velocities u^* and v^*, assuming the tentative pressure p^* equal to p^n:

$$\rho\frac{u^*-u^n}{\Delta\vartheta}+\nabla\cdot\left(\rho\mathbf{w}^n\gamma u^*\right)+\nabla\cdot\left(\rho\mathbf{w}^n\left(1-\gamma\right)u^n\right)$$
$$=\nabla\cdot\left(\mu\nabla\gamma u^*\right)+\nabla\cdot\left(\mu\nabla\left(1-\gamma\right)u^n\right)-\frac{\partial p^*}{\partial x}-\rho\beta\left(t^n-t_0\right)g_x$$

$$\rho\frac{v^*-v^n}{\Delta\vartheta}+\nabla\cdot\left(\rho\mathbf{w}^n\gamma v^*\right)+\nabla\cdot\left(\rho\mathbf{w}^n\left(1-\gamma\right)v^n\right)$$
$$=\nabla\cdot\left(\mu\nabla\gamma v^*\right)+\nabla\cdot\left(\mu\nabla\left(1-\gamma\right)v^n\right)-\frac{\partial p^*}{\partial y}-\rho\beta\left(t^n-t_0\right)g_y$$

where the convective terms are linearized according to the Picard's method:

$$\rho\mathbf{w}\gamma\mathbf{w}\approx\left(\rho\mathbf{w}^n\right)\gamma\mathbf{w}$$

Segregated methods - 2

- Bringing the terms that do not contribute to the coefficient matrix to the right-hand side, and integrating over the respective control volumes, we obtain:

$$\frac{\rho}{\Delta\vartheta}\int_{V_u}u^*\,\mathrm{d}V+\rho\gamma\int_{A_u}u^*\mathbf{w}^n\cdot\mathbf{n}\,\mathrm{d}A-\mu\gamma\int_{A_u}\nabla u^*\cdot\mathbf{n}\,\mathrm{d}A$$
$$=-\rho\left(1-\gamma\right)\int_{A_u}u^n\mathbf{w}^n\cdot\mathbf{n}\,\mathrm{d}A+\mu\left(1-\gamma\right)\int_{A_u}\nabla u^n\cdot\mathbf{n}\,\mathrm{d}A$$
$$-\left[\int_{A_{E,x}}p^*\,\mathrm{d}A-\int_{A_{P,x}}p^*\,\mathrm{d}A\right]-\rho\beta g_x\int_{V_u}\left(t^n-t_0\right)\,\mathrm{d}V+\frac{\rho}{\Delta\vartheta}\int_{V_u}u^n\,\mathrm{d}V$$

$$\frac{\rho}{\Delta\vartheta}\int_{V_v}v^*\,\mathrm{d}V+\rho\gamma\int_{A_v}v^*\mathbf{w}^n\cdot\mathbf{n}\,\mathrm{d}A-\mu\gamma\int_{A_v}\nabla v^*\cdot\mathbf{n}\,\mathrm{d}A$$
$$=-\rho\left(1-\gamma\right)\int_{A_v}v^n\mathbf{w}^n\cdot\mathbf{n}\,\mathrm{d}A+\mu\left(1-\gamma\right)\int_{A_v}\nabla v^n\cdot\mathbf{n}\,\mathrm{d}A$$
$$-\left[\int_{A_{N,y}}p^*\,\mathrm{d}A-\int_{A_{P,y}}p^*\,\mathrm{d}A\right]-\rho\beta g_y\int_{V_v}\left(t^n-t_0\right)\,\mathrm{d}V+\frac{\rho}{\Delta\vartheta}\int_{V_v}v^n\,\mathrm{d}V$$

Segregated methods - 3

- The pressure correction is calculated by solving the corresponding Poisson equation which, integrated over the pressure control volume, gives:

$$\int_A \nabla p' \cdot \mathbf{n}\, \mathrm{d}A = \frac{\rho}{\Delta\vartheta} \int_A \mathbf{w}^* \cdot \mathbf{n}\, \mathrm{d}A$$

  which in discrete form, using the midpoint rule, becomes:

$$\int_{A_e} \left.\frac{\partial p'}{\partial x}\right|_e \mathrm{d}y - \int_{A_w} \left.\frac{\partial p'}{\partial x}\right|_w \mathrm{d}y + \int_{A_n} \left.\frac{\partial p'}{\partial y}\right|_n \mathrm{d}x - \int_{A_s} \left.\frac{\partial p'}{\partial y}\right|_s \mathrm{d}x = \left\{ \left[u_e^* - u_w^*\right] \Delta y_j + \left[v_n^* - v_s^*\right] \Delta x_i \right\} \frac{\rho}{\Delta\vartheta}$$

- The gradients of the pressure correction on the faces are evaluated according to the central difference scheme, obtaining:

$$\left[\frac{p'_E - p'_P}{x_E - x_P} - \frac{p'_P - p'_W}{x_P - x_W}\right] \Delta y_j + \left[\frac{p'_N - p'_P}{y_N - y_P} - \frac{p'_P - p'_S}{y_P - y_S}\right] \Delta x_i = \left\{ \left[u^*_{i,j} - u^*_{i-1,j}\right] \Delta y_j + \left[v^*_{i,j} - v^*_{i,j-1}\right] \Delta x_i \right\} \frac{\rho}{\Delta\vartheta}$$

Segregated methods - 4

- The u' and v' components of the velocity correction are calculated as a function of the gradient of the pressure correction:

$$\frac{\rho}{\Delta\vartheta} \int_{V_u} u'\, \mathrm{d}V = -\left[\int_{A_{E,x}} p'\, \mathrm{d}A - \int_{A_{P,x}} p'\, \mathrm{d}A\right]$$

$$\frac{\rho}{\Delta\vartheta} \int_{V_v} v'\, \mathrm{d}V = -\left[\int_{A_{N,y}} p'\, \mathrm{d}A - \int_{A_{P,y}} p'\, \mathrm{d}A\right]$$

- In discrete form, and using the simplest second-order expressions for integration and interpolation, it gives:

$$u'_e = -\frac{\Delta\vartheta}{\rho} \frac{(p'_P - p'_E)}{(x_P - x_E)}$$

$$v'_n = -\frac{\Delta\vartheta}{\rho} \frac{(p'_P - p'_N)}{(y_P - y_N)}$$

Segregated methods - 5

- The values of the pressure and velocity components at the end of the time step are given by:

$$\begin{aligned} p_P^{n+1} &= p_P^* + p'_P \\ u_e^{n+1} &= u_e^* + u'_e \\ v_n^{n+1} &= v_n^* + v'_n \end{aligned}$$

- Once the new *divergence-free* velocity field is obtained, the temperature is obtained by integrating the energy equation over the VC, using the same time integration algorithm used for the velocity components:

$$\begin{aligned} &\frac{\rho\, c_p}{\Delta\vartheta} \int_V t^{n+1}\, \mathrm{d}V + \rho\, c_p\, \gamma \int_A t^{n+1} \mathbf{w}^n \cdot \mathbf{n}\, \mathrm{d}A - \lambda\gamma \int_A \nabla t^{n+1} \cdot \mathbf{n}\, \mathrm{d}A \\ &\quad = -\rho\, c_p\, (1-\gamma) \int_A t^n \mathbf{w}^n \cdot \mathbf{n}\, \mathrm{d}A + \lambda\, (1-\gamma) \int_A \nabla t^n \cdot \mathbf{n}\, \mathrm{d}A \\ &\quad + \frac{\rho\, c_p}{\Delta\vartheta} \int_V t^n\, \mathrm{d}V \end{aligned}$$

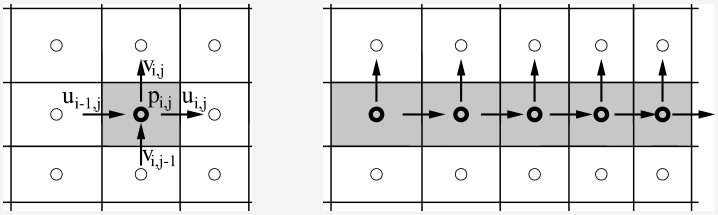
Segregated methods - 6

Remarks

- We have assumed constant thermophysical properties, but the algorithm requires little modifications in order to consider variable properties.
- The *projection* time integration algorithm just seen presents many possible variants, such as that given by the combination of an *explicit Adams-Bashfort scheme* for the convective term, and an *implicit Gear (second order Euler) scheme* for the diffusive term.
- The methodology just illustrated can serve, given the noteworthy similarities, as a reference for other solution algorithms, such as *SIMPLE* and derivatives.

Coupled methods - 1

- Segregated methods can give rise to convergence problems when the coupling between variables is very pronounced.
- This leads to the necessity to adopt very small time-steps or small values ??of the under-relaxation coefficients.
- For these situations it may be convenient to use a solution strategy that preserves, in an *implicit* way, the original coupling between variables.
- Simultaneously solved variables typically belong to appropriate *subdomains*, which are *visited* in sequence. Subdomains usually consist of:
  1. A single VC. SCGS - *Symmetrically Coupled Gauss-Seidel* - method. It can also be used for unstructured grids.
  2. A line, row or column, of VCs. For example, CELS - *Coupled Equation Line Solver*. Restricted to structured grids.



(SCGS) (CELS)

Coupled methods - 2

SCGS method, only pressure and velocity variables in the 2D steady case

$$A^u_{P_{i,j}} u_{i,j} + A^u_{E_{i,j}} u_{i+1,j} + A^u_{W_{i,j}} u_{i-1,j} + A^u_{N_{i,j}} u_{i,j+1} + A^u_{S_{i,j}} u_{i,j-1} + (p_{i+1,j} - p_{i,j})\,\Delta y_j = S^u_{i,j}$$

$$A^u_{P_{i-1,j}} u_{i-1,j} + A^u_{E_{i-1,j}} u_{i,j} + A^u_{W_{i-1,j}} u_{i-2,j} + A^u_{N_{i-1,j}} u_{i-1,j+1} + A^u_{S_{i-1,j}} u_{i-1,j-1} + (p_{i,j} - p_{i-1,j})\,\Delta y_j = S^u_{i-1,j}$$

$$A^v_{P_{i,j}} v_{i,j} + A^v_{E_{i,j}} v_{i+1,j} + A^v_{W_{i,j}} v_{i-1,j} + A^v_{N_{i,j}} v_{i,j+1} + A^v_{S_{i,j}} v_{i,j-1} + (p_{i,j+1} - p_{i,j})\,\Delta x_i = S^v_{i,j}$$

$$A^v_{P_{i,j-1}} v_{i,j-1} + A^v_{E_{i,j-1}} v_{i+1,j-1} + A^v_{W_{i,j-1}} v_{i-1,j-1} + A^v_{N_{i,j-1}} v_{i,j} + A^v_{S_{i,j-1}} v_{i,j-2} + (p_{i,j} - p_{i,j-1})\,\Delta x_i = S^v_{i,j-1}$$

$$(u_{i,j} - u_{i-1,j})\,\Delta y_j + (v_{i,j} - v_{i,j-1})\,\Delta x_i = 0$$

- The pressure gradient term has been written explicitly.
- Linear system of five equations in the unknowns $u_{i-1,j}$, $u_{i,j}$, $v_{i,j-1}$, $v_{i,j}$ and $p_{i,j}$.

Coupled methods - 3

To simplify the system, in order to economically obtain the solution analytically, the following strategy is adopted:

1. For the momentum equations, all contributions are brought to the right-hand side, with the exception of the diagonal term A_P and the contribution of the pressure $p_{i,j}$.
2. The continuity equation is maintained in its complete, original form.

By doing this the following system is obtained:

$$\begin{bmatrix} A^u_{P_{i-1,j}} & 0 & 0 & 0 & \Delta y_j \\ 0 & A^u_{P_{i,j}} & 0 & 0 & -\Delta y_j \\ 0 & 0 & A^v_{P_{i,j-1}} & 0 & \Delta x_i \\ 0 & 0 & 0 & A^v_{P_{i,j}} & -\Delta x_i \\ -\Delta y_j & \Delta y_j & -\Delta x_i & \Delta x_i & 0 \end{bmatrix} \begin{Bmatrix} u_{i-1,j} \\ u_{i,j} \\ v_{i,j-1} \\ v_{i,j} \\ p_{i,j} \end{Bmatrix} = \begin{Bmatrix} b^u_{i-1,j} \\ b^u_{i,j} \\ b^v_{i,j-1} \\ b^v_{i,j} \\ 0 \end{Bmatrix}$$

Coupled methods - 4

The terms on the right-hand side contain contributions that do not explicitly appear in the system matrix, e.g.:

$$b^u_{i,j} = -A^u_{E_{i,j}} u^*_{i+1,j} - A^u_{W_{i,j}} u^*_{i-1,j} - A^u_{N_{i,j}} u^*_{i,j+1} - A^u_{S_{i,j}} u^*_{i,j-1} - p^*_{i+1,j}\Delta y_j + S^u_{i,j}$$

The solution of this simpler system of equations consists of the following sequential steps:

$$r_1 = -\Delta y_j / A^u_{P_{i-1,j}}; \quad r_2 = \Delta y_j / A^u_{P_{i,j}}; \quad r_3 = -\Delta x_i / A^v_{P_{i,j-1}}; \quad r_4 = \Delta x_i / A^v_{P_{i,j}}$$

$$\text{DEN} = r_1 \Delta y_j - r_2 \Delta y_j + r_3 \Delta x_i - r_4 \Delta x_i$$

$$p_{i,j} = \left(r_1 b^u_{i-1,j} + r_2 b^u_{i,j} + r_3 b^v_{i,j-1} + r_4 b^v_{i,j} \right) / \text{DEN}$$

$$u_{i-1,j} = \left(b^u_{i-1,j} - \Delta y_j p_{i,j} \right) / A^u_{P_{i-1,j}}; \quad u_{i,j} = \left(b^u_{i,j} + \Delta y_j p_{i,j} \right) / A^u_{P_{i,j}}$$

$$v_{i,j-1} = \left(b^v_{i,j-1} - \Delta x_i p_{,j} \right) / A^v_{P_{i,j-1}}; \quad v_{i,j} = \left(b^v_{i,j} + \Delta x_i p_{i,j} \right) / A^v_{P_{i,j}}$$

- The SCGS method can be applied also to unstructured grids, and it can be extended to include other coupled variables, e.g. *temperature* for natural convection problems.
- The convenience of coupled methods comes, in particular, from their use in context with a *multigrid* method.

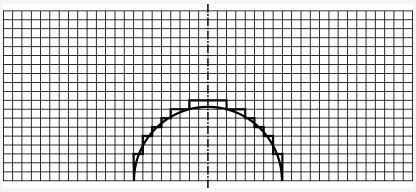
OUTLINE

Part VI

Complex geometries: unstructured grids

Complex geometries

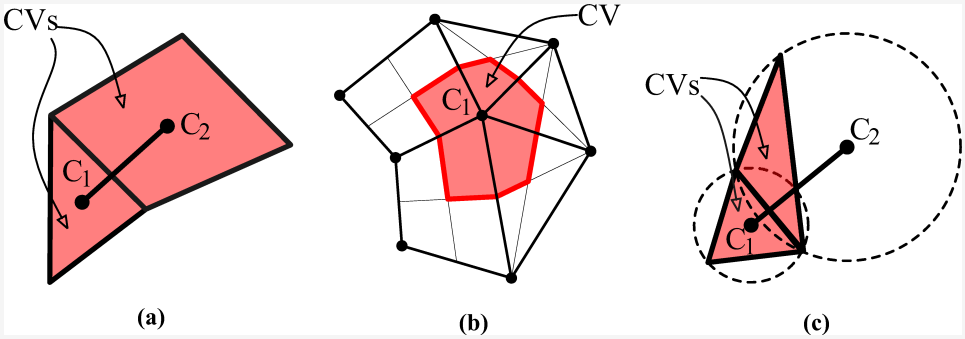
- The procedure illustrated for structured Cartesian grids is simple and accurate, and easy to implement, and in fact it has found application in research activities and in early commercial CFD codes.
- However, it is difficult to use for complex geometries:



- Need to modify the Finite Volume method to address these issues as well.
- This was done initially using curvilinear structured grids (an approach still used in some commercial/proprietary programs), then moving on to *unstructured* grids.

Finite Volume classification

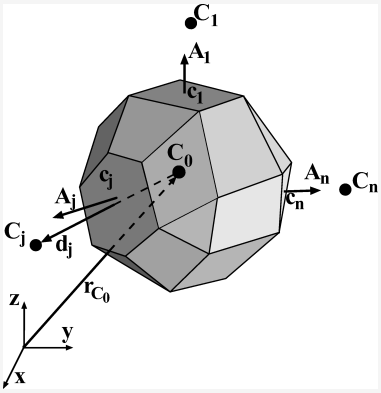
- Various methods of constructing Finite Volumes (cells) starting from unstructured grids.
- The most common are:
  - **(a)** Cell-centered
  - **(b)** Vertex-centered (or node-centered)
  - **(c)** Circum-centered



- In the following, given their greater diffusion, we will refer only to the *cell-centered* case (a).

Generalities and notation

- Domain discretized with an unstructured grid, consisting of VCs of arbitrary polyhedral shape with a number of faces $n \geq 4$:



- The only requirements are that the cells must be *convex* and the constituent faces must be *planar*.
- Greater accuracy with regular *hexahedron* shaped cells.
- Grid generators that produce polyhedral meshes:
  - The use of polyhedra, in general, can guarantee greater accuracy, due to the higher degree of orthogonality of the grid (see in the following) and the better reconstruction of the gradient.
  - It is useful to note that for node-centered grids the VCs are already cell-centered polyhedra.
- Finally, note that when operating with FVM, it is not necessary to carry out coordinate transformations.

Centroids

Centroid C_0 (center of mass) of the VC

$$\int_V (\mathbf{r} - \mathbf{r}_{C_0})\, \mathrm{d}V = 0$$

Centroid c_j of the face A_j included between C_0 and C_j

$$\int_{A_j} (\mathbf{r} - \mathbf{r}_j)\, \mathrm{d}A = 0$$

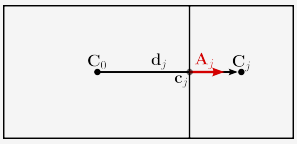
$\mathbf{A}_j$ is the *area vector* of the face:

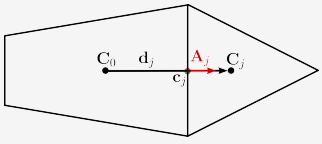
$$\mathbf{A}_j = A_j \mathbf{n}$$

- In general $\mathbf{d}_j = \mathbf{r}_{C_j} - \mathbf{r}_{C_0}$ and $\mathbf{A}_j$ are not parallel, and the angle formed by them increases for very *distorted*, i.e. *non-orthogonal* grids.
- The application point of $\mathbf{A}_j$, i.e. c_j, is in general different from the point obtained by the intersection of $\mathbf{d}_j$ with the face itself. This difference constitutes the *skewness* (asymmetry, obliquity), and gives the measure of the deviation of the current grid from an *ideal* grid.

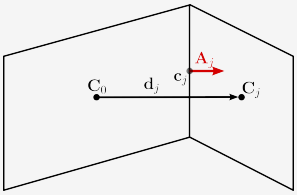
Mesh quality

- Orthogonal and not skewed mesh

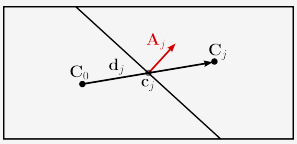




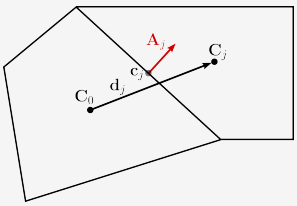
- Orthogonal and skewed mesh



- Non-orthogonal and not skewed mesh



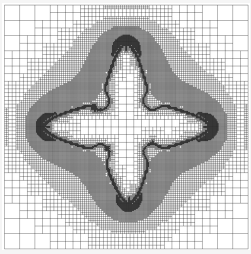
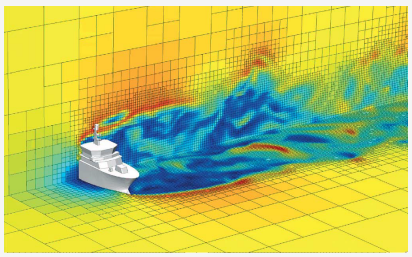
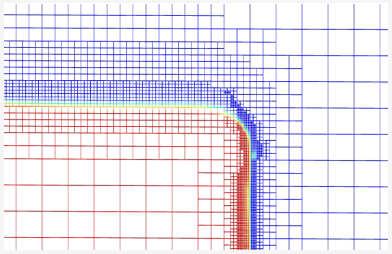
- Non-orthogonal and skewed mesh



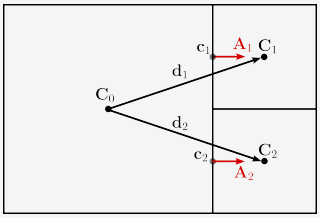
- The *non-orthogonality* add numerical diffusion to the solution, hence reducing the overall accuracy. It can also cause *wiggles*, which in turn may lead to nonphysical values and divergence.
- *Skewness* also adds numerical diffusion and reduces the accuracy. It may also leads to unboundedness, which in turn can lead to the divergence of the solution.

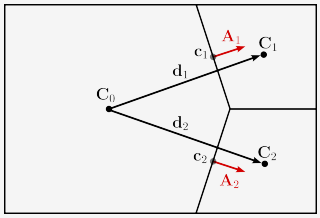
Improving the mesh

- There are several techniques available, in most mesh generators and/or CFD solvers, to improve the quality of the mesh, among these:
  - Smoothing nodes (e.g. Laplacian smoothing)
  - Swapping
  - Removing *slivers*, i.e. degenerate cells which are characterized by nodes that are nearly coplanar
- *Quadtree* and *octree* meshes:
  - They are frequently used for *adaptive meshing*

  - It is possible to improve the orthogonality





Mesh metrics

- There are several ways - *metrics* - which can be used to assess the quality of the grid.
- These are usually available both in the mesher and in the solver.
  - Some of these are (from e.g. ANSYS Mesh): *Skewness*, *Orthogonal Quality*, *Element Quality*, *Aspect Ratio* for triangles or quadrilaterals, *Jacobian Ratio*, *Warping Factor*, *Parallel Deviation*, *Maximum Corner Angle*, *et.*
- For example, for *face orthogonality* a simple metric is

$$F_{ortho} = \frac{\mathbf{A_j} \cdot \mathbf{d_j}}{|\mathbf{A_j}|\,|\mathbf{d_j}|}$$

- The *skewness* could be based on the deviation from a normalized equilateral angle. This method can be applied to all cell and face shapes. It is defined as

$$F_{skew} = \max\left[\frac{\theta_{max} - \theta_e}{180 - \theta_e}, \frac{\theta_e - \theta_{min}}{\theta_e}\right]$$

where:
θ_{max} is the largest angle for the face or cell [deg]
θ_{min} is the smallest angle for the face or cell [deg]
θ_e is the angle for an equiangular face/cell (60° for a triangle, 90° for a square et.) [deg]

Transport equation

- It is important to operate with grids in which the two vectors $\mathbf{d}_j$ and $\mathbf{A}_j$ are as parallel as possible and arranged along the same line, that is, with grids that are as *orthogonal* as possible.
- Note that in the FVM the orthogonality is measured between the line joining the centroids $\mathbf{d}_j$, and the area vector $\mathbf{A}_j$ of the intervening face.

Generic transport equation integrated on the VC:

$$\int_V \frac{\partial}{\partial \vartheta} (\rho\phi)\, \mathrm{d}V + \int_A \rho\phi \mathbf{w} \cdot \mathbf{n}\, \mathrm{d}A = \int_A \Gamma \nabla\phi \cdot \mathbf{n}\, \mathrm{d}A + \int_V s\, \mathrm{d}V$$

Similarly to what we saw for Cartesian grids, we will consider the various terms of the transport equation individually, after having first defined some aspects of the calculation procedure.

Choice of velocity components

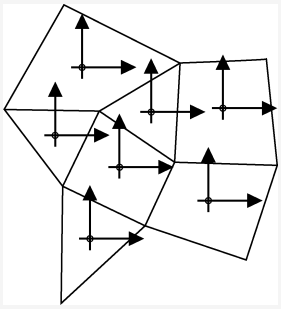
- In the FV method, the momentum equations are expressed in *conservative* form:
  - All terms of the equations can be represented as a divergence of a vector or a tensor.
  - The use of the conservative form of the equations with the FVM guarantees the global conservation of momentum even in the *discrete* representation.
  - However, the conservative form of the equations requires the components of the momentum to be expressed in a *fixed* reference frame, under the penalty of introducing non-conservative force fields, according to the definition seen.
- For this reason, it is appropriate to use components of this type, and in particular Cartesian ones, which are the simplest and most intuitive.
- In the following, we will always refer to quantities (vectors and tensors) expressed in terms of Cartesian components.

Arrangement of the variables on the grid - 1

- The *staggered* arrangement of the variables on the grid ensures a good coupling between velocity components and pressure.
- This is only true for Cartesian structured grids, for which the velocity components are *normal* to the VC faces.
- For unstructured grids, on the other hand, it would be necessary to store all the velocity components on each face.
- Using velocity components aligned locally to the grid (*contravariant components*) solves this problem, but global conservation of the momentum would not be guaranteed.

Arrangement of the variables on the grid - 2

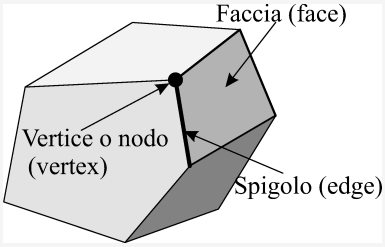
- It is now common practice to co-locate all variables in the centroid of the VCs:



- Easy management of the data structure and programming (geometric information of a single set of control volumes).
- Use of appropriate interpolation procedures - e.g. Rhie-Chow - to overcome the weak coupling between the velocity field and the pressure field.
- In FVM on unstructured grids, the use of Cartesian components of the velocity, and the co-located arrangement of the variables, represents a good compromise between simplicity of implementation, accuracy and guarantee of conservation: application in numerous codes, both *commercial* and *open source*.

Geometric quantities

- The computational domain is divided into a finite number of contiguous VCs via a grid.
- The grid, in analogy with the Finite Element method, defines the *vertices* or *nodes*, which, connected two by two, in turn constitute the *edges* of the VCs.



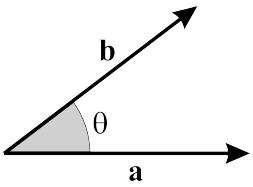
- The edges define the *faces* of the VCs, which are not necessarily flat; however, their projections onto the Cartesian planes are independent of the actual shape of the face.
- These projections are the Cartesian components of the area vector **A**, uniquely defined on the basis of the coordinates of the vertices of the face.

Vector calculus reminders - 1

Scalar (dot) product

The scalar product of two vectors **a** and **b** (*dot product*) is usually written $\mathbf{a} \cdot \mathbf{b}$, and is defined by the scalar quantity:

$$\mathbf{a} \cdot \mathbf{b} = |\mathbf{a}|\,|\mathbf{b}| \cos\theta$$



Some properties of the scalar product:

$$\mathbf{a} \cdot (\mathbf{b} + \mathbf{c}) = \mathbf{a} \cdot \mathbf{b} + \mathbf{a} \cdot \mathbf{c}$$

$$\mathbf{a} \cdot \mathbf{a} = |\mathbf{a}|^2$$

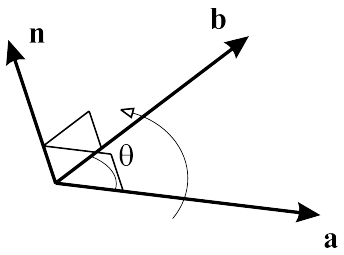
$$\begin{aligned} \mathbf{a} \cdot \mathbf{b} &= (a_x \mathbf{i} + a_y \mathbf{j} + a_z \mathbf{k}) \cdot (b_x \mathbf{i} + b_y \mathbf{j} + b_z \mathbf{k}) \\ &= a_x b_x + a_y b_y + a_z b_z \end{aligned}$$

Vector calculus reminders - 2

Vector product

The cross product of two vectors **a** and **b** (*cross product*) is usually written $\mathbf{a} \times \mathbf{b}$ (or $\mathbf{a} \wedge \mathbf{b}$), and is defined by the vector quantity:

$$\mathbf{a} \times \mathbf{b} = |\mathbf{a}|\,|\mathbf{b}| \sin\theta\, \mathbf{n}$$



Some properties of the vector product:

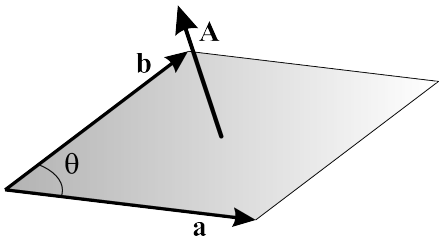
$$\begin{aligned}
\mathbf{b} \times \mathbf{a} &= -\mathbf{a} \times \mathbf{b} \\
\mathbf{a} \times (\mathbf{b} + \mathbf{c}) &= \mathbf{a} \times \mathbf{b} + \mathbf{a} \times \mathbf{c} \\
\mathbf{a} \times \mathbf{b} &= (a_x\mathbf{i} + a_y\mathbf{j} + a_z\mathbf{k}) \times (b_x\mathbf{i} + b_y\mathbf{j} + b_z\mathbf{k}) \\
&= (a_y b_z - a_z b_y)\,\mathbf{i} + (a_z b_x - a_x b_z)\,\mathbf{j} + (a_x b_y - a_y b_x)\,\mathbf{k}
\end{aligned}$$

$$\mathbf{a}\times\mathbf{b} = \begin{vmatrix} \mathbf{i} & \mathbf{j} & \mathbf{k} \\ a_x & a_y & a_z \\ b_x & b_y & b_z \end{vmatrix}$$

Vector calculus reminders - 3

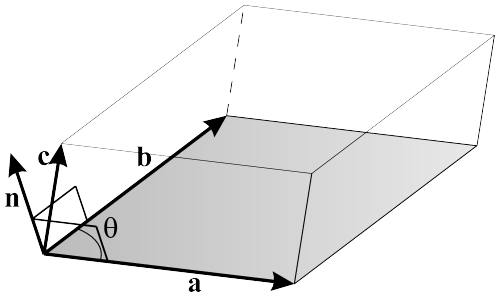
Area vector **A**

$$\mathbf{A} = \mathbf{a} \times \mathbf{b} = |\mathbf{a}|\,|\mathbf{b}| \sin\theta\, \mathbf{n}$$



Volume of a parallelepiped

$$V = \mathbf{c}\cdot(\mathbf{a} \times \mathbf{b}) = \mathbf{a}\cdot(\mathbf{b} \times \mathbf{c}) = \mathbf{b}\cdot(\mathbf{c} \times \mathbf{a})$$



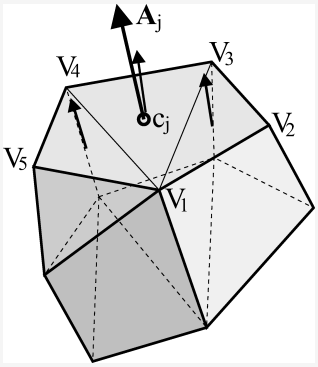
Area of a face

- Any face joining N_V vertices V_l, with $l = 1 \dots N_V$ can be decomposed into $N_V - 2$ triangles with a common vertex.

Area vector $\mathbf{A}_j$ of the face

Obtained from the sum of the area vectors of the triangles that compose it:

$$\mathbf{A}_j = \frac{1}{2} \sum_{l=3}^{N_V} \left[\left(\mathbf{r}_{V_{l-1}} - \mathbf{r}_{V_1} \right) \times \left(\mathbf{r}_{V_l} - \mathbf{r}_{V_1} \right) \right]$$



Centroid of a face

- For the triangle defined by the vertices V_1, V_2 and V_3, the position of the centroid is given by:

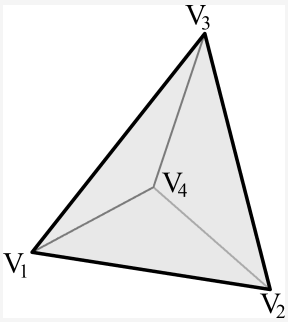
$$\mathbf{r}_{123} = \frac{1}{3} \left(\mathbf{r}_{V_1} + \mathbf{r}_{V_2} + \mathbf{r}_{V_3} \right)$$

Centroid $\mathbf{r}_j$ of the face

Obtained from the average, weighted by the respective area, of the centroids of all the triangles used to divide it:

$$\mathbf{r}_j = \frac{\frac{1}{3} \sum_{l=3}^{N_V} \left(\mathbf{r}_{V_1} + \mathbf{r}_{V_{l-1}} + \mathbf{r}_{V_l} \right) \left| \left(\mathbf{r}_{V_{l-1}} - \mathbf{r}_{V_1} \right) \times \left(\mathbf{r}_{V_l} - \mathbf{r}_{V_1} \right) \right|}{\sum_{l=3}^{N_V} \left| \left(\mathbf{r}_{V_{l-1}} - \mathbf{r}_{V_1} \right) \times \left(\mathbf{r}_{V_l} - \mathbf{r}_{V_1} \right) \right|}$$

Volume and centroid of a tetrahedron



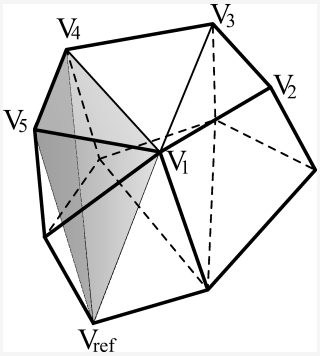
Volume of a tetrahedron

$$V_{1234} = \frac{1}{6}\left(\mathbf{r}_{V_3} - \mathbf{r}_{V_1}\right) \cdot \left[\left(\mathbf{r}_{V_2} - \mathbf{r}_{V_1}\right) \times \left(\mathbf{r}_{V_4} - \mathbf{r}_{V_1}\right)\right]$$

Centroid of a tetrahedron

$$\mathbf{r}_{1234} = \frac{1}{4}\left(\mathbf{r}_{V_1} + \mathbf{r}_{V_2} + \mathbf{r}_{V_3} + \mathbf{r}_{V_4}\right)$$

Volume of a polyhedron



Volume of a polyhedron

Decomposition into tetrahedra and adding the contribution of each of these:

$$V = \frac{1}{6}\sum_{i=1}^{N_f}\sum_{l=3}^{N_{V,i}}\left(\mathbf{r}_{V_{i,1}} - \mathbf{r}_{V_{ref}}\right) \cdot \left[\left(\mathbf{r}_{V_{i,l-1}} - \mathbf{r}_{V_{ref}}\right) \times \left(\mathbf{r}_{V_{i,l}} - \mathbf{r}_{V_{ref}}\right)\right]$$

with N_f number of faces of the polyhedron, $N_{V,i}$ number of vertices of the i-th face, and V_{ref} an arbitrary reference vertex (or any other point).

Centroid of a polyhedron

Centroid of a polyhedron

From the weighted average, with respective volume, of the position vectors of the centroids of the tetrahedra making up the polyhedron:

$$\mathbf{r}_{C_0} = \frac{1}{24}\frac{1}{V}\sum_{i=1}^{N_f}\sum_{l=3}^{N_{V,i}}\left(\mathbf{r}_{V_{ref}} + \mathbf{r}_{V_{i,1}} + \mathbf{r}_{V_{i,l-1}} + \mathbf{r}_{V_{i,l}}\right) \left\{\left(\mathbf{r}_{V_{i,1}} - \mathbf{r}_{V_{ref}}\right)\cdot\left[\left(\mathbf{r}_{V_{i,l-1}} - \mathbf{r}_{V_{ref}}\right)\times\left(\mathbf{r}_{V_{i,l}} - \mathbf{r}_{V_{ref}}\right)\right]\right\}$$

Simplified relationships

- Taking advantage of the identity:

$$\nabla\mathbf{r} = \nabla\left(x\mathbf{i} + y\mathbf{j} + z\mathbf{k}\right) \equiv 3$$

the volume of the cell can be calculated using Gauss' theorem:

$$V = \frac{1}{3}\int_V \nabla\mathbf{r}\,\mathrm{d}V = \frac{1}{3}\int_A \mathbf{r}\cdot\mathbf{n}\mathrm{d}A = \frac{1}{3}\sum_{i=1}^{N_f}\mathbf{r}_i\cdot\mathbf{n}_i\,A_i$$

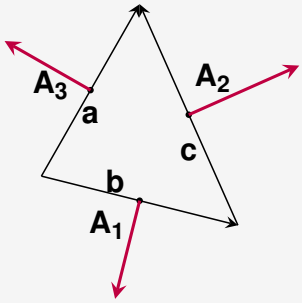
- With similar considerations, through a slightly more complex derivation, the centroid can be evaluated with the relation:

$$\mathbf{r}_{C_0} = \frac{3\sum_{i=1}^{N_f}\left(\mathbf{r}_i\cdot\mathbf{n}_i\right)\mathbf{r}_i\,A_i}{4\sum_{i=1}^{N_f}\mathbf{r}_i\cdot\mathbf{n}_i\,A_i}$$

- These expressions are generally approximate, but are exact for cells having triangular or flat quadrilateral faces.

Sum of the vector areas of a triangle

We can prove that the three vectors (with outward-pointing normals) of the faces of a triangle sum to zero.



Since

$$\mathbf{A_1} = \mathbf{b} \times \mathbf{k}$$
$$\mathbf{A_2} = \mathbf{c} \times \mathbf{k}$$
$$\mathbf{A_3} = \mathbf{k} \times \mathbf{a}$$

It follows that

$$\mathbf{A_1} + \mathbf{A_2} + \mathbf{A_3} = \mathbf{b} \times \mathbf{k} + \mathbf{c} \times \mathbf{k} + \mathbf{k} \times \mathbf{a}$$

but:

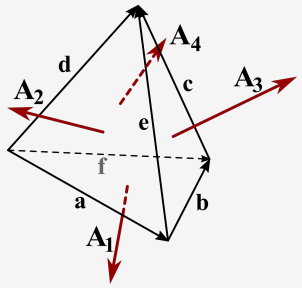
$$\mathbf{a} = \mathbf{b} + \mathbf{c}$$

and therefore, remembering that $\mathbf{a} \times \mathbf{b} = -\mathbf{b} \times \mathbf{a}$

$$\begin{aligned}\mathbf{A_1} + \mathbf{A_2} + \mathbf{A_3} &= \mathbf{b} \times \mathbf{k} + \mathbf{c} \times \mathbf{k} + \mathbf{k} \times (\mathbf{b} + \mathbf{c}) \\ &= \cancel{\mathbf{b} \times \mathbf{k}} + \cancel{\mathbf{c} \times \mathbf{k}} + \cancel{\mathbf{k} \times \mathbf{b}} + \cancel{\mathbf{k} \times \mathbf{c}} \\ &= 0\end{aligned}$$

Sum of the vector areas of a tetrahedron

We can prove that the four vector areas (with outward-pointing normals) of the faces of a tetrahedron sum to zero.



$$\mathbf{A_1} = \frac{1}{2}\mathbf{b} \times \mathbf{a}$$
$$\mathbf{A_2} = \frac{1}{2}\mathbf{a} \times \mathbf{d}$$
$$\mathbf{A_3} = \frac{1}{2}\mathbf{b} \times \mathbf{e}$$
$$\mathbf{A_4} = \frac{1}{2}\mathbf{d} \times \mathbf{f}$$

Therefore

$$\begin{aligned}&\mathbf{A_1} + \mathbf{A_2} + \mathbf{A_3} + \mathbf{A_4} \\ &\quad = \frac{1}{2}(\mathbf{b} \times \mathbf{a} + \mathbf{a} \times \mathbf{d} + \mathbf{b} \times \mathbf{e} + \mathbf{d} \times \mathbf{f})\end{aligned}$$

but

$$\mathbf{d} = \mathbf{a} + \mathbf{e} \quad \mathbf{f} = \mathbf{a} + \mathbf{b}$$

it follows that

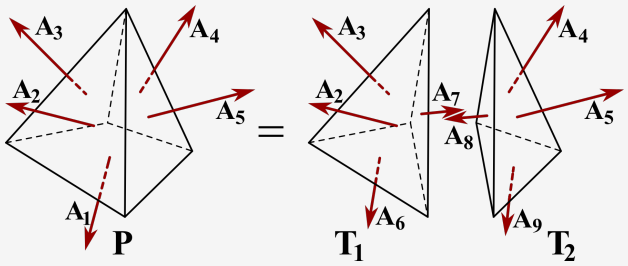
$$\begin{aligned}\mathbf{a} \times \mathbf{d} = \mathbf{a} \times (\mathbf{a} + \mathbf{e}) &= \cancel{\mathbf{a} \times \mathbf{a}} + \mathbf{a} \times \mathbf{e} \\ \mathbf{d} \times \mathbf{f} = (\mathbf{a} + \mathbf{e}) \times (\mathbf{a} + \mathbf{b}) &= \cancel{\mathbf{a} \times \mathbf{a}} + \mathbf{a} \times \mathbf{b} + \mathbf{e} \times \mathbf{a} + \mathbf{e} \times \mathbf{b}\end{aligned}$$

and substituting

$$\begin{aligned}&\mathbf{A_1} + \mathbf{A_2} + \mathbf{A_3} + \mathbf{A_4} \\ &\quad = \frac{1}{2}(\cancel{\mathbf{b} \times \mathbf{a}} + \cancel{\mathbf{a} \times \mathbf{e}} + \cancel{\mathbf{b} \times \mathbf{e}} + \cancel{\mathbf{a} \times \mathbf{b}} + \cancel{\mathbf{e} \times \mathbf{a}} + \cancel{\mathbf{e} \times \mathbf{b}}) \equiv 0\end{aligned}$$

Sum of the vector areas of a pyramid

It is easy to see that the five vector areas (with outward-pointing normals) of the faces of a pyramid sum to zero by observing that it can be splitted in two tetrahedrons



P = pyramid; $\mathbf{T_1}$ = tetrahedron; $\mathbf{T_2}$ = tetrahedron.

$$\textit{For}\ \ \mathbf{T_1} : \mathbf{A_2} + \mathbf{A_3} + \mathbf{A_6} + \mathbf{A_7} = 0$$
$$\textit{For}\ \ \mathbf{T_2} : \mathbf{A_4} + \mathbf{A_5} + \mathbf{A_8} + \mathbf{A_9} = 0$$

Summing up

$$\mathbf{A_2} + \mathbf{A_3} + \mathbf{A_6} + \mathbf{A_7} + \mathbf{A_4} + \mathbf{A_5} + \mathbf{A_8} + \mathbf{A_9} = 0$$

and grouping

$$\mathbf{A_2} + \mathbf{A_3} + (\mathbf{A_6} + \mathbf{A_9}) + (\mathbf{A_7} + \mathbf{A_8}) + \mathbf{A_4} + \mathbf{A_5} = 0$$

Observing that

$$(\mathbf{A_6} + \mathbf{A_9}) = \mathbf{A_1}$$
$$(\mathbf{A_7} + \mathbf{A_8}) = 0$$

it results that

$$\mathbf{A_1} + \mathbf{A_2} + \mathbf{A_3} + \mathbf{A_4} + \mathbf{A_5} = 0$$

In other words, the rule

$$\sum_i \mathbf{A_i} = 0$$

holds also for a square pyramid.

Sum of the vector areas of a polyhedron

The results obtained for a *tetrahedron* and a *pyramid*, can be extended to a generic *polyhedron* by observing that it can also be decomposed in *tetrahedrons* and therefore:

- The surface area of a face of the polyhedron is equal to the sum of the triangular surface areas of the corresponding tetrahedrons.
- The contribution of surface areas of the tetrahedrons which are *inside* the polyhedron will cancel since, as shown for the pyramid, they are counted twice with different sign.

The same result can be obtained also by the *Gauss* theorem.
Let's consider a polyhedron V with a surface area A, characterized by $\mathbf{A}_i$, $i = 1, \ldots, N$ surface area vectors ($\mathbf{A}_i = \mathbf{n}_i A_i$), and assume that $\mathbf{u}$ is a constant vector. From Gauss's theorem:

$$\mathbf{u} \sum_i \mathbf{A_i} = \int_A \mathbf{u} \cdot \mathbf{n} \mathrm{d}A = \int_V \nabla \mathbf{u} \mathrm{d}V$$

But, since $\mathbf{u}$ is constant, its divergence vanishes and consequently:

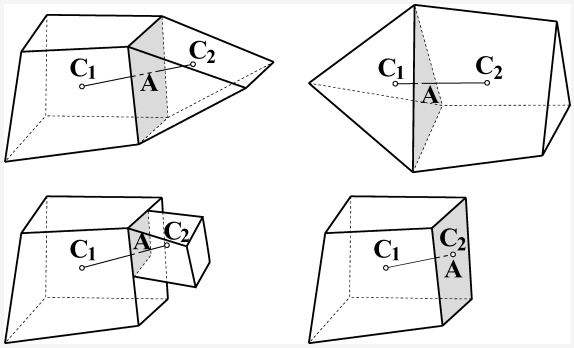
$$\mathbf{u} \sum_i \mathbf{A_i} = 0$$

But $\mathbf{u}$ is arbitrary, so this can happen only if:

$$\sum_i \mathbf{A_i} = 0$$

Cell type

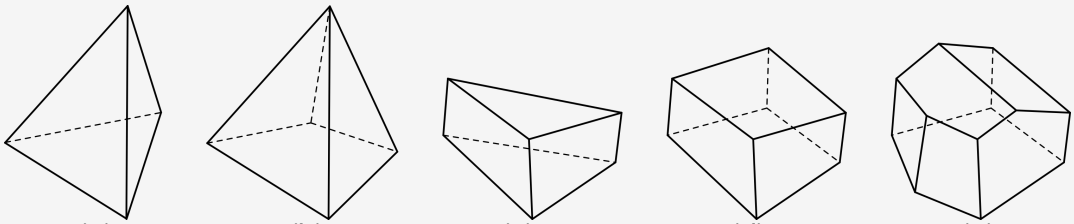
- Given the possibility of discretizing the domain into VCs characterized by an arbitrary number of faces, it is common to adopt, in the implementation, a *face-based* data structure, rather than a *VC- or node-based* data structure.
- In this way, it is easy to consider cells and combinations of cells such as those shown in the figure:



- For each face it is sufficient to know the adjacent cells C_1 and C_2, and the area vector $\mathbf{A}$, in order to evaluate the convective and diffusive fluxes.

Common cell shapes

- Most common cell types - (a) tetrahedron; (b) pyramid; (c) prism; (d) hexahedron; (e) polyhedron.



- The numerical evaluation of surface and volume integrals requires knowledge of the coordinates of the VC centroids, C_j, and the faces, c_j.
- It is also necessary to know the area vector of each face, $\mathbf{A}_j$, and the volume V of the VC.
- This data is typically supplied by grid generators, otherwise we can proceed with their evaluation, through the relations seen, knowing the coordinates of the V_j vertices of the VC.

Spatial distribution of variables - 1

- The dependent variables, and the values of the thermophysical properties, are defined in the centroids of the VCs.
- However, to calculate surface integrals, we need their value at the centroids of the faces.
- We assume a linear variation of the variable ϕ within the cell:

$$\phi = \phi_{C_0} + (\nabla\phi)_{C_0} \cdot (\mathbf{r} - \mathbf{r_{C_0}})$$

  where $(\nabla\phi)_{C_0}$ indicates the gradient evaluated in C_0, constant over the entire control volume.
- This expression provides a different value of ϕ on the face, depending on which of the two VCs it is considered to belong to.
- For this reason, a more complex formulation is adopted which, analogously to the Cartesian case, guarantees perfect symmetry.

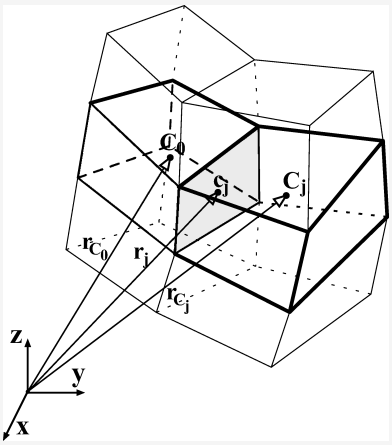
Spatial distribution of variables - 2

Symmetric formulation - CDS

$$\phi_j = \frac{1}{2}\left(\phi_{C_0} + \phi_{C_j}\right) + \frac{1}{2}\left[(\nabla\phi)_{C_0} \cdot (\mathbf{r}_j - \mathbf{r}_{C_0}) + (\nabla\phi)_{C_j} \cdot \left(\mathbf{r}_j - \mathbf{r}_{C_j}\right)\right]$$

where the subscript j indicates the face between the VCs C_0 and C_j, adjacent to that face.

It corresponds to the *Central Difference Scheme - CDS* already seen for Cartesian grids.



Surface and volume integrals

- The transport equation must be integrated over each VC.
- Adoption of second-order schemes, suitable for industrial CFD applications.
- Use of the midpoint formula.

Integration

$$\int_V \phi \, \mathrm{d}V \approx \phi_{C_0} \, \Delta V$$

$$\int_{A_j} \phi \, \mathrm{d}\mathbf{A} \approx \phi_j \mathbf{A}_j$$

$$\int_{A_j} \nabla\phi \cdot \mathrm{d}\mathbf{A} \approx (\nabla\phi)_j \cdot \mathbf{A}_j$$

Gradient calculation

- In order to calculate the values of the variable at the center of the face, and to evaluate the diffusive term, it is necessary to know the gradient of ϕ in the centroid of the cell.
- The calculation of the convective term also requires knowledge of the gradient.
- There are several strategies, adopted in commercial and open source programs, to obtain (*reconstruct*) this quantity based on only the values of the variable on the centroids of the cells.
- The following are two of the most popular alternatives.

Gauss-Green method

- Let us call ϕ_{x_0}, ϕ_{y_0} and ϕ_{z_0} the components of the gradient of ϕ in C_0, and recall that ϕ_{x_0} can be considered the divergence of the vector $\phi\mathbf{i}$.
- Applying the Gauss-Green theorem:

$$\phi_{x_0}\Delta V \approx \int_V \phi_x \mathrm{d}V = \int_V \nabla\cdot(\phi\,\mathbf{i})\,\mathrm{d}V = \int_A \phi\,\mathbf{i}\cdot\mathrm{d}\mathbf{A} \approx \sum_j \phi_j A_{j,x}$$

- Proceeding in a similar way for the other components of the gradient we obtain:

$$\phi_{x_0}\,\mathbf{i} + \phi_{y_0}\,\mathbf{j} + \phi_{z_0}\,\mathbf{k} \approx \frac{\sum_j \phi_j\, A_{j,x}}{\Delta V}\,\mathbf{i} + \frac{\sum_j \phi_j\, A_{j,y}}{\Delta V}\,\mathbf{j} + \frac{\sum_j \phi_j\, A_{j,z}}{\Delta V}\,\mathbf{k}$$

Gradient reconstruction with the Gauss-Green method

$$(\nabla\phi)_{C_0} \approx \frac{\sum_j \phi_j\,\mathbf{A}_j}{\Delta V}$$

It remains to be determined how to evaluate the value of the variable on the face, ϕ_j.

Gauss-Green *cell based* method

- Arithmetic mean:

$$\phi_j = \frac{1}{2}\left(\phi_{C_0} + \phi_{C_j}\right)$$

- Alternatively, weighted average (weighted interpolation):

$$\phi_j = \lambda\phi_{C_0} + (1-\lambda)\phi_{C_j}$$

where

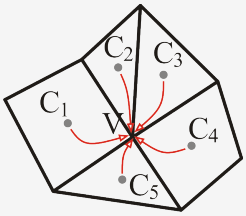
$$\lambda = \frac{\left|\mathbf{r}_{C_j} - \mathbf{r}_j\right|}{\left|\mathbf{r}_{C_j} - \mathbf{r}_{C_0}\right|}$$

- This approach is simple and economical to implement, but it gives rise to a second-order error only if the vectors $(\mathbf{r}_{C_j} - \mathbf{r}_{C_0})$ and $\mathbf{A}_j$ are aligned.
- Otherwise, an iterative correction procedure can be used.

Gauss-Green *node based* method

1. Values of the variable on the nodes (vertices) by *weighted average* of the value of ϕ in the centroids of the cells that define the node.
   In the simplest formulation the weight is assumed equal to the inverse of the distance between the node (vertex) and the centroid of the cell (2D case representation for convenience):

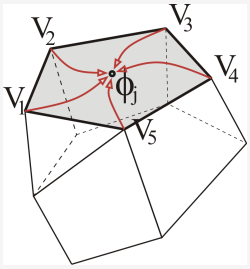
$$\phi_V = \frac{\sum_{j=1}^{N_C} \dfrac{\phi_{C_j}}{\left|\mathbf{r}_V - \mathbf{r}_{C_j}\right|}}{\sum_{j=1}^{N_C} \dfrac{1}{\left|\mathbf{r}_V - \mathbf{r}_{C_j}\right|}}$$



   where N_C is the number of cells defining the node.

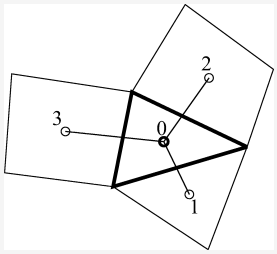
2. Average over the $N_{V,j}$ nodes (vertices) defining the j-th face:

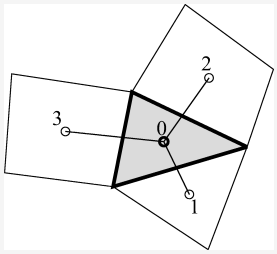
$$\phi_j = \frac{1}{N_{V,j}} \sum_{l=1}^{N_{V,j}} \phi_l$$



Gauss-Green *cell based* method examples - 1

The relation found gives a value of the gradient, in C_0, which *does not depend* on ϕ_{C_0}.





Example 2D

It results

$$\phi_{x_0} \approx \frac{1}{\Delta V}\left[\frac{(\phi_{C_0}+\phi_{C_1})}{2}A_{1,x} + \frac{(\phi_{C_0}+\phi_{C_2})}{2}A_{2,x} + \frac{(\phi_{C_0}+\phi_{C_3})}{2}A_{3,x}\right]$$

$$\phi_{y_0} \approx \frac{1}{\Delta V}\left[\frac{(\phi_{C_0}+\phi_{C_1})}{2}A_{1,y} + \frac{(\phi_{C_0}+\phi_{C_2})}{2}A_{2,y} + \frac{(\phi_{C_0}+\phi_{C_3})}{2}A_{3,y}\right]$$

Gauss-Green *cell based* method examples - 2

Example 2D (*continues*)

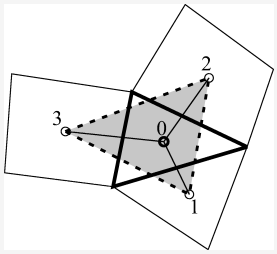
Simplifying the expressions found:

$$\phi_{x_0} \approx \frac{1}{2\,\Delta V}\left[\phi_{C_1} A_{1,x} + \phi_{C_2} A_{2,x} + \phi_{C_3} A_{3,x}\right]$$

$$\phi_{y_0} \approx \frac{1}{2\,\Delta V}\left[\phi_{C_1} A_{1,y} + \phi_{C_2} A_{2,y} + \phi_{C_3} A_{3,y}\right]$$

Gauss-Green *cell based* method examples - 3

Adoption of an *extended* integration domain: it reduces the errors associated with the evaluation of variable values on face-centroids through interpolation.



Example 2D - Extended integration domain

$$\phi_{x_0} \approx \frac{1}{2\,\Delta V}\left[\phi_{C_1}\left(y_{C_2} - y_{C_3}\right) + \phi_{C_2}\left(y_{C_3} - y_{C_1}\right) + \phi_{C_3}\left(y_{C_1} - y_{C_2}\right)\right]$$

$$\phi_{y_0} \approx \frac{1}{2\,\Delta V}\left[\phi_{C_1}\left(x_{C_3} - x_{C_2}\right) + \phi_{C_2}\left(x_{C_1} - x_{C_3}\right) + \phi_{C_3}\left(x_{C_2} - x_{C_1}\right)\right]$$

Even in this case, however, the components of the gradient in C_0 do not depend on ϕ_{C_0}.

Gauss-Green method - comments

- The Gauss-Green *cell based* method, for the evaluation of gradients, is accurate only for Cartesian grids, or in any case for structured *orthogonal* grids. It is rather imprecise for non-uniform grids and unstructured grids consisting of cells of different types, as is the case of *hybrid grids*.
- The Gauss-Green *node based* method - in particular adopting the weighted average proposed by D. G. Holmes and S. D. Connell, 1989 - is more accurate than the *cell based* method on irregular grids, however it requires a greater computational effort.
- The *least squares* method, described in the following, is also able to guarantee a better accuracy than the *cell based* Gauss-Green method, in particular for distorted and irregular grids, but it also requires a slightly higher computational effort, comparable or lower than that of the *node based* method.

Least Squares method - 1

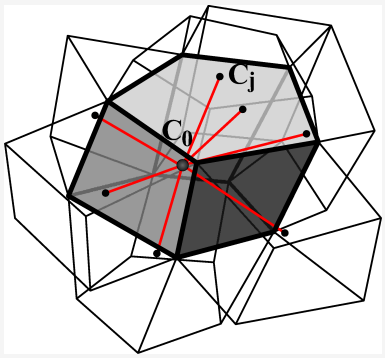
The (non-symmetric) relationship seen which provides the linear variation of the generic variable ϕ inside the cell, can be written in explicit form:

$$\phi = \phi_{C_0} + \phi_{x_0}\left(x - x_{C_0}\right) + \phi_{y_0}\left(y - y_{C_0}\right) + \phi_{z_0}\left(z - z_{C_0}\right)$$

By imposing that this relationship is valid not only on the cell in question, but describes the spatial variation of the variable up to the centroids of the adjacent cells, we have:

$$\phi_{C_j} = \phi_{C_0} + \phi_{x_0}\left(x_{C_j} - x_{C_0}\right) + \phi_{y_0}\left(y_{C_j} - y_{C_0}\right) + \phi_{z_0}\left(z_{C_j} - z_{C_0}\right) - r_j$$

where r_j is the residue: we are faced with a *overdetermined* linear system.



Least Squares method- 2

In compact form:

$$r_j = \phi_{x_0}\, d_{xj} + \phi_{y_0}\, d_{yj} + \phi_{z_0}\, d_{zj} - \Delta_j$$

where:

$$\begin{aligned} \Delta_j &= \phi_{C_j} - \phi_{C_0} \\ d_{xj} &= x_{C_j} - x_{C_0} \\ d_{yj} &= y_{C_j} - y_{C_0} \\ d_{zj} &= z_{C_j} - z_{C_0} \end{aligned}$$

The square of the residue then results into:

$$\begin{aligned} r_j^2 = \Delta_j^2 &+ \left(\phi_{x_0}\, d_{xj}\right)^2 + \left(\phi_{y_0}\, d_{yj}\right)^2 + \left(\phi_{z_0}\, d_{zj}\right)^2 \\ &-2\left(\Delta_j\, \phi_{x_0}\, d_{xj} + \Delta_j\, \phi_{y_0}\, d_{yj} + \Delta_j\, \phi_{z_0}\, d_{zj}\right) \\ &+2\left(\phi_{x_0}\, \phi_{y_0}\, d_{xj}\, d_{yj} + \phi_{x_0}\, \phi_{z_0}\, d_{xj}\, d_{zj} + \phi_{y_0}\, \phi_{z_0}\, d_{yj}\, d_{zj}\right) \end{aligned}$$

which summed over all *nb (neighbors)* adjacent cells gives:

$$R = \sum_{j=1}^{nb} r_j^2$$

Least Squares method - 3

Minimizing by least squares:

$$\begin{aligned} \frac{\partial R}{\partial \phi_{x_0}} &= 0 \\ \frac{\partial R}{\partial \phi_{y_0}} &= 0 \\ \frac{\partial R}{\partial \phi_{z_0}} &= 0 \end{aligned}$$

Developing, collecting and expressing in matrix form, it results:

$$\begin{bmatrix} \sum_{j=1}^{nb} (d_{xj})^2 & \sum_{j=1}^{nb} d_{xj} d_{yj} & \sum_{j=1}^{nb} d_{xj} d_{zj} \\ \sum_{j=1}^{nb} d_{xj} d_{yj} & \sum_{j=1}^{nb} (d_{yj})^2 & \sum_{j=1}^{nb} d_{yj} d_{zj} \\ \sum_{j=1}^{nb} d_{xj} d_{zj} & \sum_{j=1}^{nb} d_{yj} d_{zj} & \sum_{j=1}^{nb} (d_{zj})^2 \end{bmatrix} \begin{Bmatrix} \phi_{x_0} \\ \phi_{y_0} \\ \phi_{z_0} \end{Bmatrix} = \begin{Bmatrix} \sum_{j=1}^{nb} \Delta_j\, d_{xj} \\ \sum_{j=1}^{nb} \Delta_j\, d_{yj} \\ \sum_{j=1}^{nb} \Delta_j\, d_{zj} \end{Bmatrix}$$

Least Squares method - 4

With compact notation

$$\mathbf{D}_0 \nabla\phi_{C_0} = \mathbf{f}_0$$

Gradient reconstruction with the least squares method (LSQ)

$$\nabla\phi_{C_0} = \mathbf{D}_0^{-1} \mathbf{f}_0$$

- $\mathbf{D}_0$ is a 3×3 symmetric matrix, whose elements depend only on the grid. Therefore, it is necessary to compute the inverse matrix $\mathbf{D}_0^{-1}$, for the case of an undeformable grid, only at the beginning of the calculation.
- The reconstruction of the gradient with the least squares method therefore requires storing, and inverting, the matrix $\mathbf{D}_0$ for each cell, for a total of $6 \times N$ real numbers, with N total number of cells.

Least Squares method - comments

- For very distorted grids the matrix $\mathbf{D}_0$ may be ill-conditioned:
  - Adoption of a solution methodology based on *QR factorization* (factorization of the original matrix into the product of an orthogonal matrix Q and an upper triangular matrix R via the Gram-Schmidt process).
  - Introduction of *weights* inversely proportional to the distance of neighboring cells.
- The method is characterized by *first-order accuracy* for grids made of cells of arbitrary shape, and is also *consistent*, i.e. the gradient of a linear function is computed *exactly* for any type of cell. It is therefore particularly suitable for dealing with *hybrid grids*.

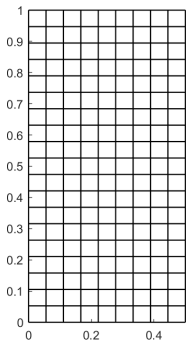
Gradient reconstruction - numerical tests

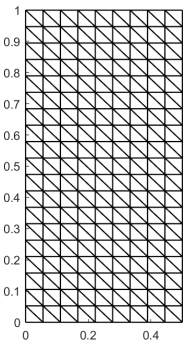
In the following, we will see how well the previous schemes can compute (*reconstruct*) the gradient on several types of grids. In particular:

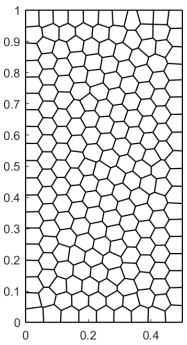
- The domain selected is a single 2D rectangle, in order to provide the possibility to use also Cartesian and regular grids.
- The grids are characterized by the same number of cells, e.g. 200. This low number has been selected since it can more easily favor a graphical comparison between results.
- The comparison is performed in case of linear, quadratic, cubic and sinusoidal (trigonometric) functions.
- Since the value of the scalar variable ϕ is obtained by means of an analytical function, the latter is used to set its value at the *cell centroids*, at the *face centroids* and (when required) at the *nodes* (vertices).

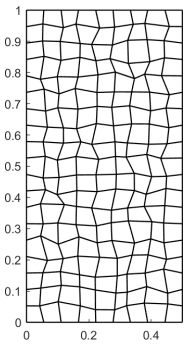
Gradient reconstruction - numerical tests - cont. 1

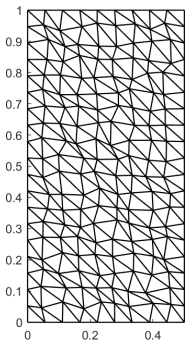
Grids and functions considered


Cartesian


Triangular


Polygonal


Distorted Cartesian


Distorted Triangular

1. Linear: $\phi = x + y$
2. Quadratic: $\phi = x^2 + y^2 + 2x + y + 1$
3. Cubic: $\phi = x^3 + 2y^3 + xy + 2$
4. Composite trigonometric function: $\phi = \sin(9x) + \cos(15y) + 5xy$

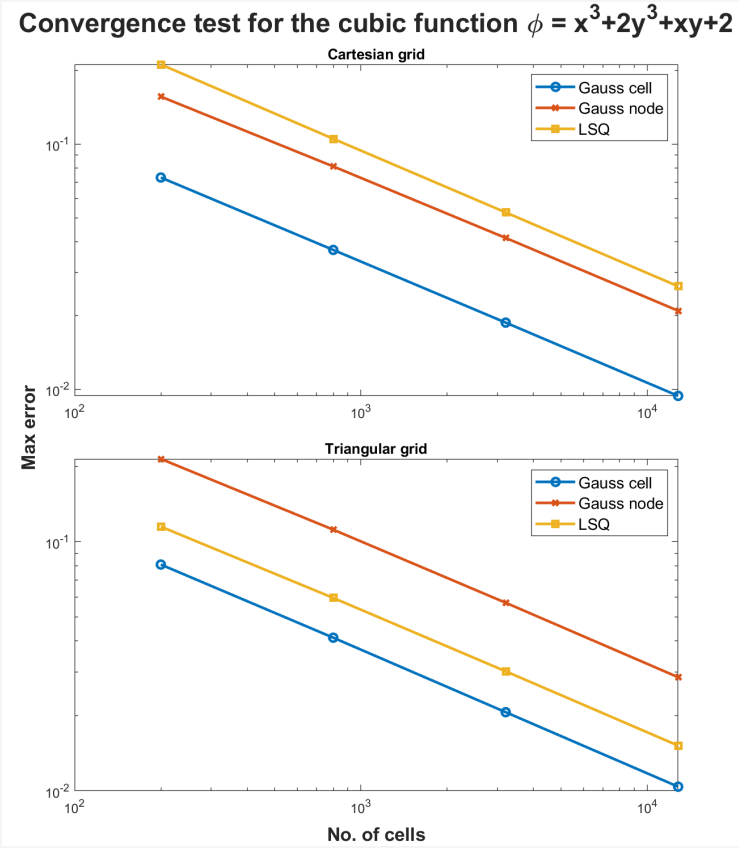
Gradient reconstruction - numerical tests - cont. 2

Max error of gradient reconstruction schemes for the various grids

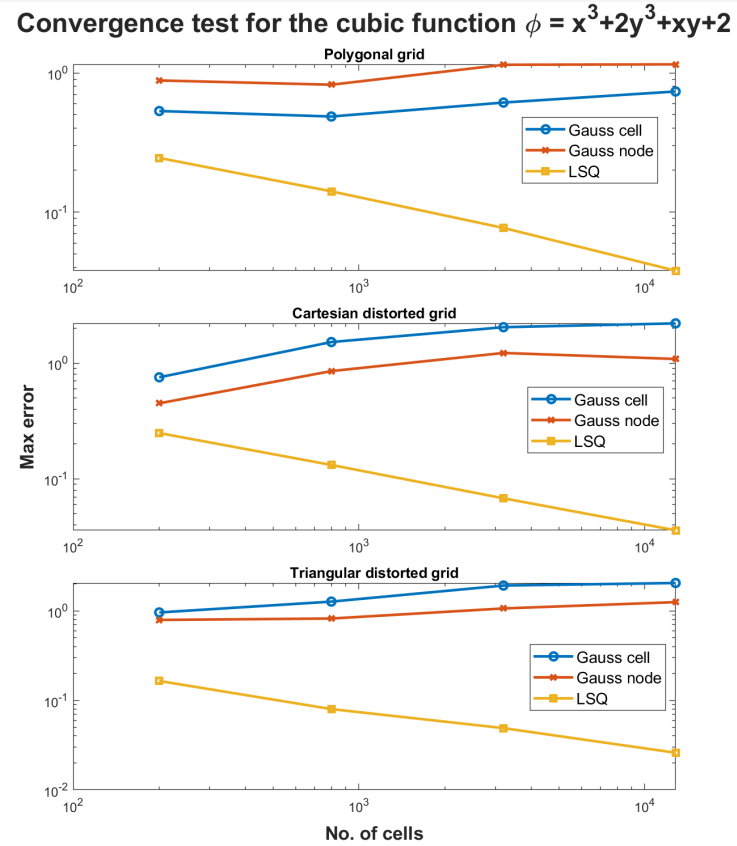
		Cartesian	Triangular	Polygonal	Distorted Cartesian	Distorted Triangular
Linear	*GCell*	5.44×10^{-15}	3.99×10^{-6}	0.093	0.270	0.375
	GNode	5.01×10^{-6}	4.44×10^{-6}	0.198	0.194	0.213
	LSq	2.44×10^{-15}	5.00×10^{-15}	1.67×10^{-15}	2.22×10^{-15}	4.22×10^{-15}
Quadratic	*GCell*	0.014	0.019	0.284	0.766	0.846
	GNode	0.043	0.070	0.443	0.423	0.525
	LSq	0.039	0.027	0.057	0.061	0.060
Cubic	*GCell*	0.073	0.081	0.532	0.755	0.962
	GNode	0.156	0.215	0.882	0.450	0.794
	LSq	0.211	0.115	0.245	0.249	0.165
Trigonometric	*GCell*	1.425	2.076	1.735	2.759	7.082
	GNode	2.886	4.617	3.208	3.533	5.450
	LSq	3.116	2.222	4.171	3.982	3.460

GCell = Gauss-Green Cell-based; *GNode = Gauss-Green Node-based;* *LSq = Least Square*

Gradient reconstruction - numerical tests - cont. 3



Gradient reconstruction - numerical tests - cont. 4



Unsteady term

- The procedure, and the result, depends on the temporal integration scheme adopted.
- For stationary or slow time-varying problems, it is sufficient to adopt the first order implicit Euler scheme.
- For flows with rapid temporal variations, it is convenient to use the Crank-Nicolson or the implicit second order Euler (Gear) scheme.

Integration of the unsteady term

$$\int_V \frac{\partial}{\partial \vartheta} (\rho\phi)\, \mathrm{d}V \approx \frac{\partial}{\partial \vartheta} \left(\rho_{C_0}\, \phi_{C_0}\, \Delta V\right) \approx \rho_{C_0} \frac{\Delta V}{\Delta \vartheta} \left(\phi_{C_0}^{n+1} - \phi_{C_0}^{n}\right)$$

Source term

Integration of the source term

$$\int_V s \mathrm{d}V \approx s_{C_0} \Delta V$$

If the source term depends on the variable itself, it is linearized as seen in the case of Cartesian grids.

Convective flux - 1

The convective term is calculated taking into account the contribution of all the faces that make up the VC:

$$\int_A \rho\phi\mathbf{w} \cdot \mathrm{d}\mathbf{A} = \sum_j \int_{A_j} \rho\phi\mathbf{w} \cdot \mathrm{d}\mathbf{A}$$

and, based on the midpoint formula:

Convective flux

$$\sum_j \int_{A_j} \rho\phi\mathbf{w} \cdot \mathrm{d}\mathbf{A} \approx \sum_j \dot{m}_j \phi_j$$

where $\dot{m}_j$ represents the mass flow rate through the face A_j

$$\dot{m}_j = \int_{A_j} \rho\mathbf{w} \cdot \mathrm{d}\mathbf{A} \approx \rho_j \left(\mathbf{w}_j \cdot \mathbf{A}_j\right)$$

Linearization of the convective term by the Picard method.

Convective flux - 2

Calculation of ϕ_j:

CDS (Central Difference Scheme)

$$\phi_j = \frac{1}{2}\left(\phi_{C_0} + \phi_{C_j}\right) + \frac{1}{2}\left[(\nabla\phi)_{C_0} \cdot (\mathbf{r}_j - \mathbf{r}_{C_0}) + (\nabla\phi)_{C_j} \cdot \left(\mathbf{r}_j - \mathbf{r}_{C_j}\right)\right]$$

- Possibility of instability and oscillations (wiggles), and convergence difficulties, particularly for convective-dominated flows, i.e. high Reynolds number.
- Cure: selective grid refinement.

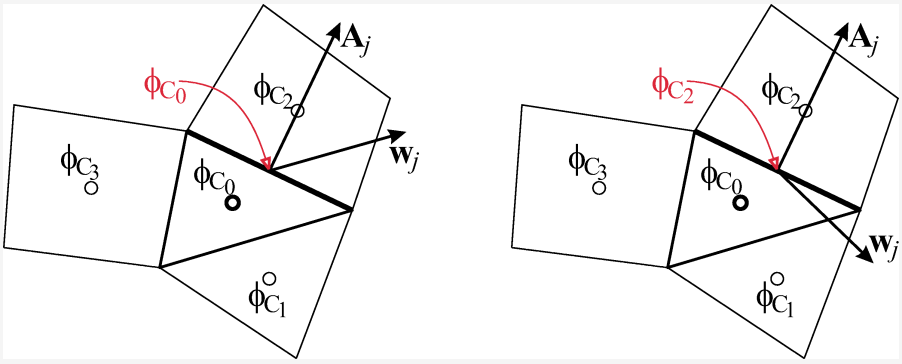
UDS (Upwind difference scheme)

$$\phi_j = \begin{cases} \phi_{C_0} & \text{se} \quad (\mathbf{w}_j \cdot \mathbf{A}_j) > 0 \\ \phi_{C_j} & \text{se} \quad (\mathbf{w}_j \cdot \mathbf{A}_j) < 0 \end{cases}$$

- Introduction of *artificial diffusion*: use, if absolutely necessary, with caution and only for specific cases.
- Acceptable for *source-dominated* transport equations, e.g. turbulence models.

Convective flux - 3

Graphical representation of the UDS scheme:



Convective flux - 4

SOUDS (Second order upwind difference scheme)

$$\phi_j = \begin{cases} \phi_{C_0} + (\nabla\phi)_{C_0} \cdot (\mathbf{r}_j - \mathbf{r}_{C_0}) & \text{se} \quad (\mathbf{w}_j \cdot \mathbf{A}_j) > 0 \\ \phi_{C_j} + (\nabla\phi)_{C_j} \cdot \left(\mathbf{r}_j - \mathbf{r}_{C_j}\right) & \text{se} \quad (\mathbf{w}_j \cdot \mathbf{A}_j) < 0 \end{cases}$$

Use of mixed schemes, through a *blending* factor γ between 0 and 1:

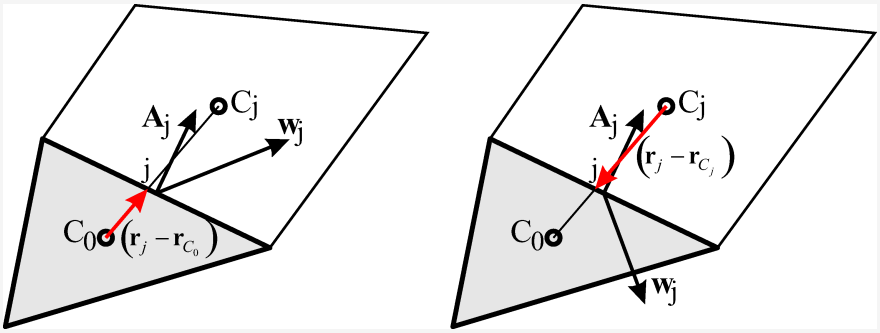
Mixed scheme (first-to-higher-order blending)

$$\phi_j = \gamma\phi_j^{ho} + (1-\gamma)\,\phi_j^{lo} = \phi_j^{lo} + \left[\gamma\left(\phi_j^{ho} - \phi_j^{lo}\right)\right]$$

- *lo* indicates a low-order scheme (UDS), and *ho* indicates a high-order scheme (CDS).
- $\gamma = 1$ corresponds to the CDS scheme, and $\gamma = 0$ corresponds to the UDS scheme.
- Used in *deferred correction* mode.

Convective flux - 5

Graphical representation of the SOUDS scheme:



Diffusive flux - 1

The diffusive term is calculated considering the contribution of all the faces constituting the VC:

$$\int_A \Gamma \nabla\phi \cdot \mathrm{d}\mathbf{A} = \sum_j \int_{A_j} \Gamma \nabla\phi \cdot \mathrm{d}\mathbf{A}$$

and, based on the midpoint formula:

$$\sum_j \int_{A_j} \Gamma \nabla\phi \cdot \mathrm{d}\mathbf{A} \approx \sum_j \Gamma_j \left(\nabla\phi\right)_j \cdot \mathbf{A}_j$$

The value of $(\nabla\phi)_j$ is approximated by:

$$\left(\nabla\phi\right)_j \approx \frac{1}{2}\left[\left(\nabla\phi\right)_{C_0} + \left(\nabla\phi\right)_{C_j}\right]$$

which, replaced in the previous one, results:

$$\sum_j \int_{A_j} \Gamma \nabla\phi \cdot \mathrm{d}\mathbf{A} \approx \frac{1}{2}\sum_j \Gamma_j \left[\left(\nabla\phi\right)_{C_0} + \left(\nabla\phi\right)_{C_j}\right] \cdot \mathbf{A}_j$$

Diffusive flux - 2

Collecting, and observing that $\sum_j \Gamma_j \mathbf{A}_j \approx 0$, and in particular $\Gamma \sum_j \mathbf{A}_j = 0$ in the case of constant thermophysical properties, we obtain:

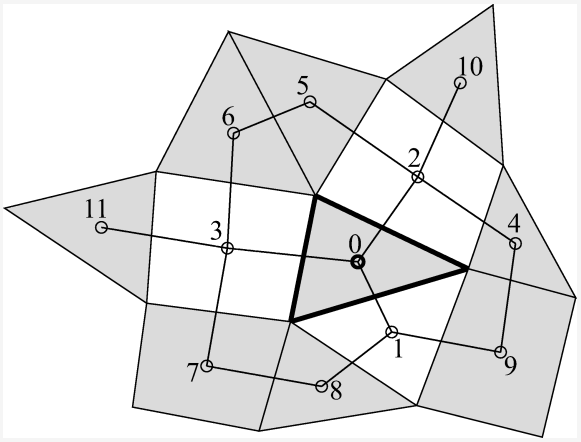
$$\int_A \Gamma \nabla\phi \cdot \mathrm{d}\mathbf{A} \approx \frac{1}{2}\sum_j \Gamma_j \left(\nabla\phi\right)_{C_j} \cdot \mathbf{A}_j$$

The expression now obtained has two drawbacks:

1. We have included *non-adjacent* VCs (second neighbors) in the discretization.
2. Although the contributions of the *adjacent* VCs (first neighbors) are formally present, they cancel each other out: solutions with unphysical oscillations (*wiggles* or *checkerboarding*).

Diffusive flux - 3

Discretization of the diffusive term for the cell C_0: only the shaded VCs contribute with non-zero coefficients.



Diffusive flux - 4

Using a *modified* expression for the gradient, which includes the effect of adjacent cells in the discretization, for example:

$$\widetilde{(\nabla\phi)}_j = (\nabla\phi)_j + \left(\frac{\phi_{C_j} - \phi_{C_0}}{|\mathbf{d}_j|} \frac{\mathbf{A}_j}{|\mathbf{A}_j|} - \frac{(\nabla\phi)_j \cdot \mathbf{d}_j}{|\mathbf{d}_j|} \frac{\mathbf{A}_j}{|\mathbf{A}_j|} \right)$$

With these modifications, the discrete expression of the diffusive flux, written using the *deferred correction* approach, becomes:

Diffusive flux

$$\int_A \Gamma \nabla\phi \cdot \mathrm{d}\mathbf{A} \approx \sum_j \Gamma_j \left\{ \frac{\phi_{C_j} - \phi_{C_0}}{|\mathbf{d}_j|} |\mathbf{A}_j| + \left[(\nabla\phi)_j - \frac{(\nabla\phi)_j \cdot \mathbf{d}_j}{|\mathbf{d}_j|} \frac{\mathbf{A}_j}{|\mathbf{A}_j|} \right] \cdot \mathbf{A}_j \right\}$$

The first term on the right is treated implicitly, while the expression in square brackets, known as *cross-diffusion term*, is treated explicitly.

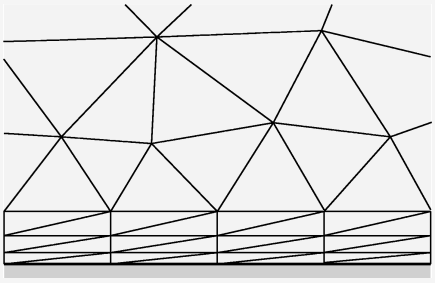
Diffusive flux - some observations

The procedure seen for the discretization of the diffusive term constitutes the basis of several commercial and open source codes, but it has some defects:

- The accuracy of the method, and its convergence properties, are reduced for grids far from orthogonality: the contribution of the *cross-diffusion* term increases.
- For very distorted grids the cross-diffusion term can even become negative, with catastrophic consequences for positive definite variables, e.g. turbulent kinetic energy.
- *Tetrahedra* grids are less accurate, with the discretization seen, than grids made of only *hexahedra*. The former, however, are often preferred for reasons related to the ease of automatic generation of the grid, and possible use of *adaptive* grids.

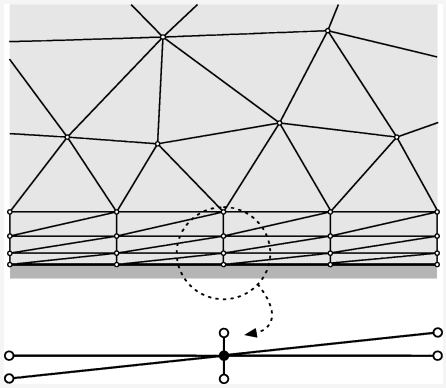
Hybrid grids - 1

- The FVM approach for unstructured grids is very popular and is adopted, albeit with some variations, in numerous CFD packages.
- However, it presents a drawback when used with tetrahedral, in 3D, or triangle grids in 2D.
- In the calculation of turbulent flows with RANS or LES models, the centroid of the cells arranged on the solid walls must be at a distance y^+ from the latter, included in an appropriate range.
- This results in the need to *squeeze*, and therefore elongate, the first cells near the walls:



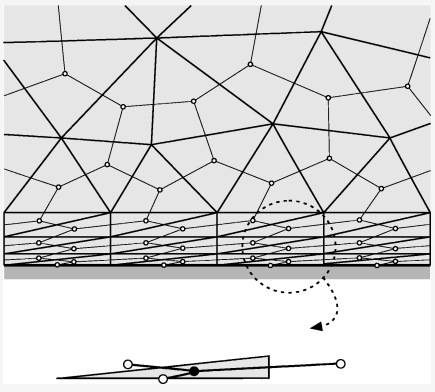
Hybrid grids - 2

- With the FE (Finite Element) method there would be no stability problems since all the nodes, indicated in the detail, would give a contribution to the row of the matrix corresponding to the node in question, indicated by the full symbol.
- There would possibly be problems related to the reduced accuracy of very *elongated* elements).



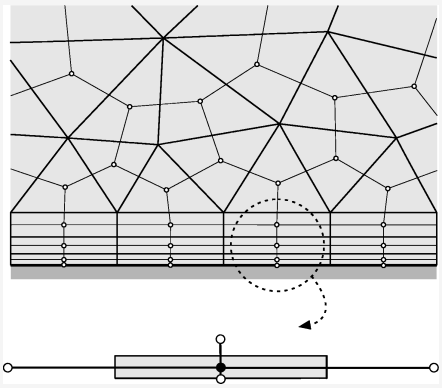
Hybrid grids - 3

- With the FV method described, the calculation molecule is very deformed: some lines joining the centroids of the cells form a significant angle with the face-normals and, in the limiting case of very elongated cells, they can be almost parallel to the faces themselves:
  - The contribution of the explicit diffusive term (cross-diffusion term) increases, and the implicit term decreases: this can lead to convergence difficulties.
  - Increases the contribution of cross-diffusion terms which can sometimes become negative: reduced accuracy.
  - For momentum equations, the accuracy of calculating the pressure gradient near solid walls decreases.



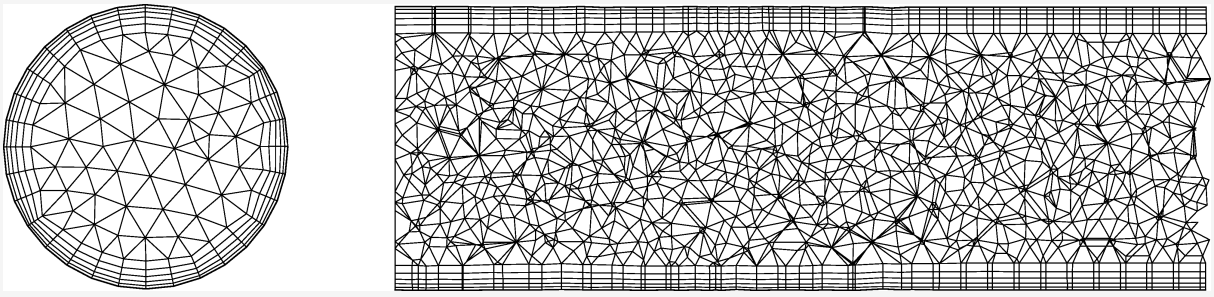
Hybrid grids - 4

- Solution: use *hybrid grids*, i.e. hexahedrons and/or prisms (2D rectangles) for the first layers of cells, and tetrahedrons (2D triangles) at the center of the domain.
- The computation molecule is much more regular, due to the greater orthogonality between the face-normals and the lines joining the centroids.



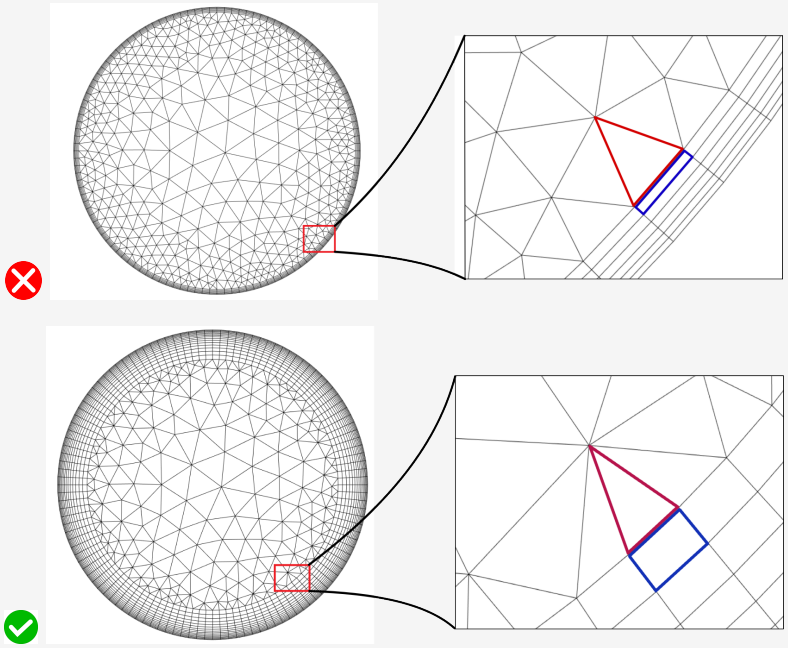
Hybrid grids - 5

- Opportunity (necessity) to use hybrid grids, instead of grids with only tetrahedra, as confirmed by most commercial and open source mesh generators, which provide its automatic generation as an option.
- Adoption, where possible, of structured grids with only *hexahedrons* or, alternatively, unstructured grids with extruded wall layers: a possibility offered by numerous CFD packages, and specific products for grid generation.



Hybrid grids - 6

- The volume of the cells of the final extruded layer should be similar, in size, to that of the first cells attached to them.
- If this does not happen, avoid to increase the growth-ratio above 1.2-1.3, but rather increase the number of layers.



Initial conditions

- For non-stationary problems, it is necessary to fix the values of the variables at the initial time.
- If only the periodic behavior is of interest (e.g. internal combustion engines, turbomachines, rotating parts of machines, etc.), these values ??can be arbitrary, but it is necessary to integrate for an adequate time, in order to eliminate the effects due to the (arbitrary) initial conditions.
- To trigger some phenomena, such as non-stationary separations of boundary layers, it is convenient, if possible, to perturb the solution.
- For stationary problems, the use of initial conditions as close as possible to the solution sought makes the calculation more economical, and reduces the risk of failure (divergence).
- Such initial conditions can be obtained by solving, for example:
  - Simplified problems (e.g. purely diffusive flows and heat transfer).
  - Interpolation/extrapolation of the solution of more economical problems (e.g. coarse grid, two-dimensional problem).
  - Most commercial CFD packages provide self-contained procedures for initializing the flow field and other scalar fields, e.g. temperature, concentration etc.

Boundary conditions

- The Cartesian components of velocity are generally not aligned with the cell faces on the boundary.
- No use is made of *ghost cells*, and in general the integrals on the faces of the VCs on the boundary must be expressed, as in the Cartesian case, as a function of the boundary conditions and the values of the variables at the centers of the same VCs:
  - With *Dirichlet* boundary conditions, the convective flow is immediately available, while the diffusive flow requires approximating the gradient on the boundary.
  - With *Neumann* boundary conditions, the value of the diffusive flux is immediately available, while the value of the variable, which is necessary in the case of *convective* boundary conditions, is obtained based on the (discrete) approximation of the gradient.
- The boundary conditions on solid walls, in turbulent flow problems, depend on the turbulence model adopted, and on the treatment modalities of the wall zone.

Final discretization equation

- After assembling, for every CV, all terms of the transport equation integrated over the CV itself (unsteady term(s), diffusive and convective fluxes, source terms, boundary conditions et.), the final equation is obtained, here written in compact form

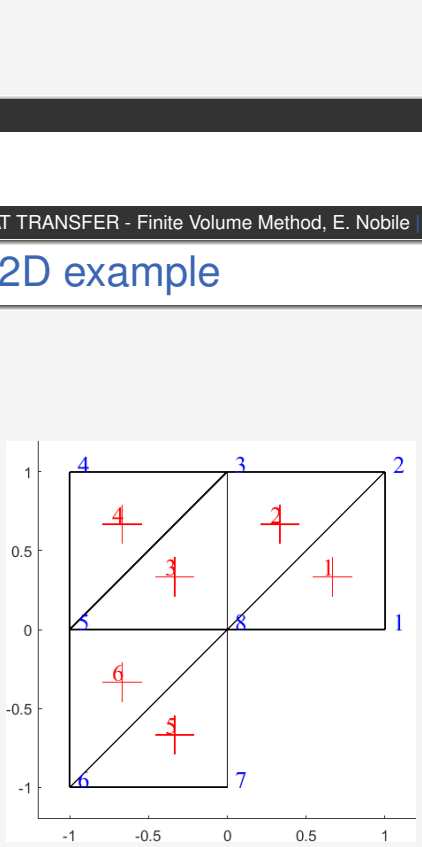
$$\boxed{A_0\phi_0 + \sum_{nb} A_{nb}\phi_{nb} = S_0}$$

  where, again, with *nb* we intend that the summation is extended to all neighboring CVs, whose number, differently from the case of structured grids, varies from one CV to the other.
- The coefficients in the equation depends on the approximations chosen for the various terms, e.g. convective and diffusive fluxes, unsteady term and source term.
- The resulting global coefficient matrix is *sparse*, and is usually solved by one of the available iterative methods.
- Introducing an under-relaxation coefficient α_ϕ within the iterative procedure, as already seen for Cartesian grids, the final discretization equation becomes

$$\boxed{\underbrace{\frac{A_0}{\alpha_\phi}}_{A_0}\phi_0^k + \sum_{nb} A_{nb}\phi_{nb}^k = \underbrace{S_0 + \frac{(1-\alpha_\phi)\,A_0}{\alpha_\phi}\phi_0^{k-1}}_{S_0}}$$

Mesh description

- The mesh is composed of FVs - cells - defined by a set of vertices and bounded by faces;
- Information about mesh topology:
  - List of vertices;
  - List of faces, defined in term of vertices;
  - List of cells, defined in term of faces (3D) or vertices (2D);
  - Element, face and vertex connectivity.
- The mesh boundary is divided into numbered patches of boundary faces: these patches are used to define the physical boundary conditions for the problem at hand.



List of nodes (vertices):

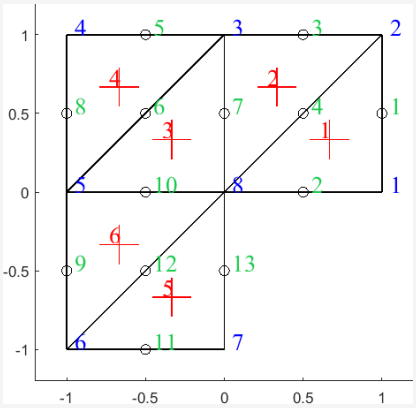
8		
1	1.0	0.0
2	1.0	1.0
3	0.0	1.0
4	-1.0	1.0
5	-1.0	0.0
6	-1.0	-1.0
7	0.0	-1.0
8	0.0	0.0

Element connectivity (list of elements):

6			
1	1	2	8
2	3	8	2
3	8	3	5
4	4	5	3
5	7	8	6
6	5	6	8

Mesh: 2D example

Mesh: 2D example - cont.



Vertex to element connectivity:

8				
1	0	0	0	0
1	2	0	0	0
2	3	4	0	0
4	0	0	0	0
3	4	6	0	0
5	6	0	0	0
5	0	0	0	0
1	2	3	5	6

Edge connectivity:

13				
1	1	2	1	-1
2	1	8	1	-1
3	2	3	2	-1
4	2	8	1	2
5	3	4	4	-1
6	3	5	3	4
7	3	8	2	3
8	4	5	4	-1
9	5	6	6	-1
10	5	8	3	6
11	6	7	5	-1
12	6	8	5	6
13	7	8	5	-1

Neighbor connectivity:

6						
1	-1	2	-1	1	4	2
2	3	1	-1	7	4	3
3	2	4	6	7	6	10
4	-1	3	-1	8	6	5
5	-1	6	-1	13	12	11
6	-1	5	3	9	12	10