051176 - Computational Techniques for Thermochemical Propulsion

Master of Science in Aeronautical Engineering

The Finite Volume Method (FVM)

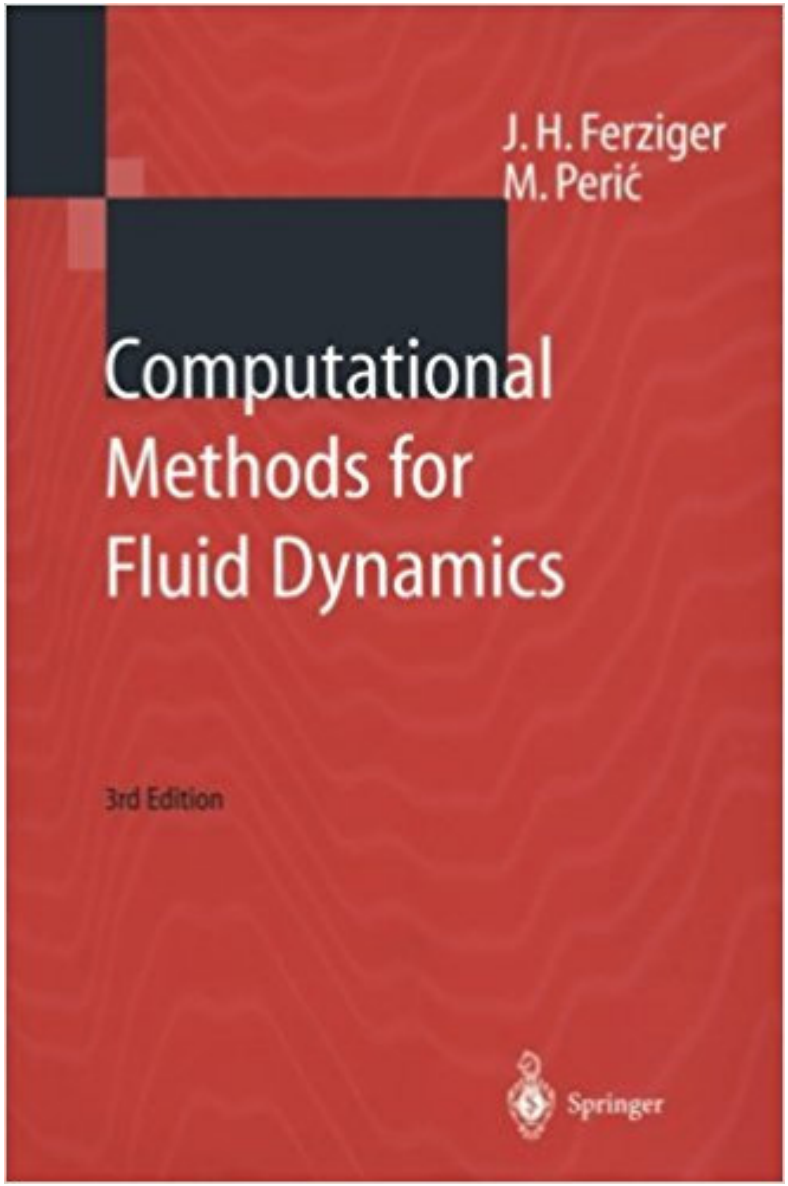
Prof. **Federico Piscaglia**

Dept. of Aerospace Science and Technology (DAER)

POLITECNICO DI MILANO, Italy

federico.piscaglia@polimi.it

Bibliography



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

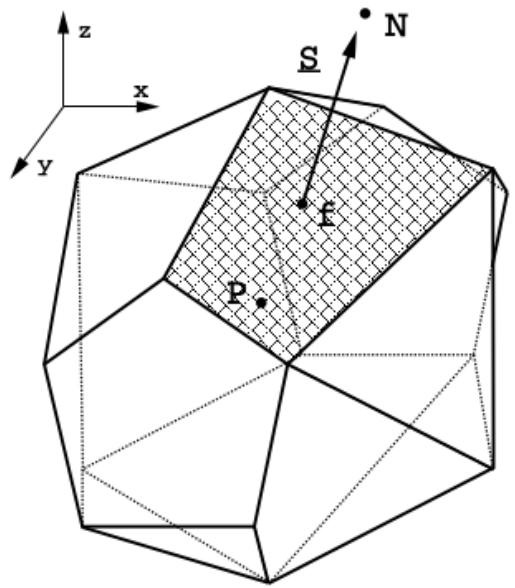
The Finite Volume Method

REMINDER: the integral conservation law is enforced for the small control volumes defined by the computational mesh:

$$\bar{V} = \bigcup_{i=1}^{N} \bar{V}_i$$

with:

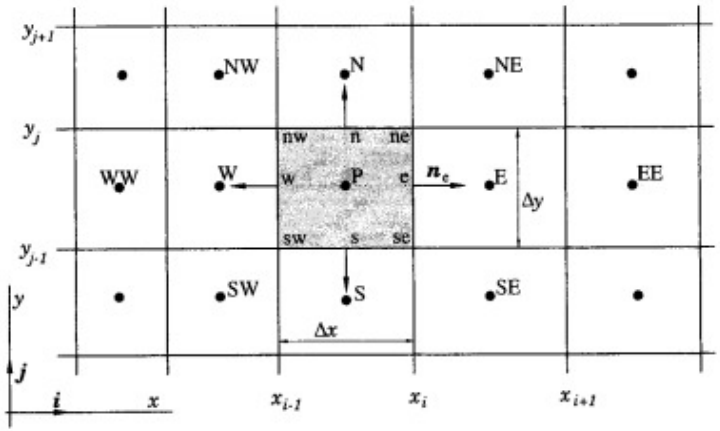
$$V_i \cap V_j = 0 \quad \forall i \neq j$$

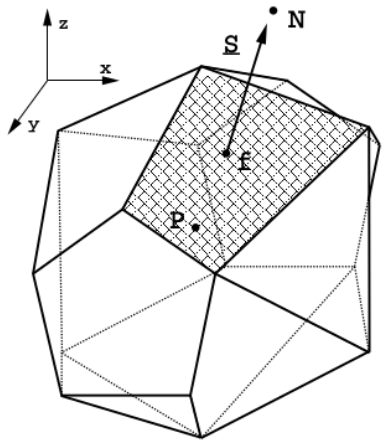


To be specified:

- concrete choice of control volumes
- type of approximation inside them
- numerical methods for evaluation of integrals and fluxes

The Finite Volume Method





The **discretization procedure** used in the finite volume method is distinctive and **involves two basic steps**:

1. **Discretization of the solution domain**: it produces a numerical description of the computational domain, including the positions of points in which the solution is sought and the description of the boundary. The space is divided into a finite number of discrete regions, called control volumes or cells. For transient simulations, the time interval is also split into a finite number of time-steps.
2. **Equation discretization**: it gives an appropriate transformation of terms of governing equations into algebraic expressions.

Discretization of the solution domain

1) **Discretization of the solution domain** produces a computational mesh on which the governing equations are subsequently solved. It also determines the positions of points in space and time where the solution is sought. **The procedure can be split into two parts: discretization of time and space**.

- **Space discretization** for the FV Method (FVM) requires a sub-division of the domain into control volumes (CV) or computational cells. Control volumes do not overlap and completely fill the computational domain. All variables share the same CV-s.
- **Temporal discretization:** since time is a parabolic coordinate, the solution is obtained by marching in time from the prescribed initial condition. For the discretization of time, it is therefore sufficient to prescribe the size of the time-step that will be used during the calculation.

The equation discretization method

2) **The equation discretization method** is based on discretizing the integral form of governing equations over each control volume. The basic quantities, such as mass and momentum will therefore be considered at the discrete level. Equations are solved in a fixed Cartesian coordinate system on the mesh that does not change in time. The method is applicable to both steady-state and transient calculations.

a) The control volumes can be of a general polyhedral shape, with a variable number of neighbours, thus creating an arbitrarily unstructured mesh. All dependent variables share the same control volume, which is usually called the colocated or non-staggered variable arrangement.

b) Non-linear differential equations are linearized before the discretization and the non-linear terms are lagged (see slides “Linear Equation Systems”).

c) Systems of partial differential equations are treated in the segregated fashion, meaning that they are solved one at a time, with the inter-equation couling treated in the explicit manner (see slides on “P-U Coupling”).

The Semi-Discretized Equations

The **conservation equation for a general scalar variable** ϕ can be expressed as:

$$\underbrace{\underbrace{\frac{\partial(\rho\phi)}{\partial t}}_{\texttt{temporal derivative}} + \underbrace{\nabla\cdot\rho(\boldsymbol{u}-\boldsymbol{u_b})\phi}_{\texttt{convection term}}}_{\frac{D\phi}{Dt}} = \underbrace{\nabla\cdot\left(\Gamma^{\phi}\nabla\phi\right)}_{\texttt{diffusion term}} + \underbrace{Q^{\phi}}_{\texttt{source term}}$$

where Q^{ϕ} represents the transport of ϕ by mechanisms other than convection and any sources or sinks of the scalar.

NOTE: the steady-state form of the above equation is obtained by dropping the transient term:

$$\nabla\cdot\rho(\boldsymbol{u}-\boldsymbol{u_b})\phi = \nabla\cdot\left(\Gamma^{\phi}\nabla\phi\right) + Q^{\phi}$$

The Semi-Discretized Equations

The **INTEGRAL FORM** of the conservation of a scalar quantity ϕ is:

$$\underbrace{\underbrace{\frac{\partial}{\partial t}\int_V \rho\phi \, dV}_{\texttt{temporal derivative}} + \underbrace{\int_V \nabla \cdot \rho(\boldsymbol{u} - \boldsymbol{u_b})\phi \, dV}_{\texttt{convection term}}}_{\frac{D\phi}{Dt}} = \underbrace{\int_V \Gamma^\phi \nabla\phi \, dV}_{\texttt{diffusion term}} + \underbrace{\int_V Q^\phi \, dV}_{\texttt{source/sink terms}}$$

where Q^ϕ represents the transport of ϕ by mechanisms other than convection and any sources or sinks of the scalar.

Properties of the Discretized Equations

What are the properties that the solution of my discretized equations must have?

Discretization yields a large system of non-linear algebraic equations. The method of solution depends on the problem.

- For **UNSTEADY FLOWS**, methods based on those used for initial value problems for ordinary differential equations (marching in time) are used. At each time step an elliptic problem has to be solved.
- **STEADY FLOW PROBLEMS** are usually solved by pseudo-time marching (LTS) or an equivalent iteration scheme. Since the equations are non-linear, an iteration scheme is used to solve them. These methods use successive linearization of the equations and the resulting linear systems are almost always solved by iterative techniques.

Properties of Numerical Solution Method

In the FV Method, it is crucial for the discretized equations to possess some properties in order to ensure a meaningful solution field.

1) **Conservation.** From a physical point of view it is very important for the transported variables, which are generally conservative quantities to be conserved in the discretized solution domain too, otherwise results may be unrealistic. This property is inherent to the FVM because the fluxes integrated at an element face are based on the values of the elements sharing the face. Any method that possesses this property is said to be conservative.

2) **Consistency.** Discretization should become exact as the grid spacing tends to zero. The difference between the discretized equation and the exact solution is called *truncation error*. For a method to be *consistent*, the truncation error must become zero when the mesh spacing $\Delta t \to 0$ and/or $\Delta x \to 0$.

3) **Convergence.** Property of a numerical method to produce a solution which approaches the exact one as the *grid spacing* goes to zero.
   - convergence is checked using numerical experiments
   - if the method is stable and if all approximations used in the discretization process are consistent, we find that the solution does converge to a *grid independent solution*.

Properties of the Discretized Equations

4) **Stability.** Even if approximations are consistent, it does not necessarily mean that that the solution of the discretized equation system will become the exact solution of the differential equation in the limit of the small step size. For this to happen, the solution method has to be *stable*.

   A numerical solution method is said to be stable if it does not magnify the errors that appear in the numerical solution process.

   - in *iterative methods*, a stable method is one that does not diverge;
   - stability is difficult to investigate, especially when boundary conditions and non-linearities are present.

   For transient problems, a stable numerical scheme keeps the error in the solution bounded as time marching proceeds. The use of explicit or implicit transient schemes has direct impact on the stability of the numerical method.

   - The stability of explicit methods is ensured by limiting the size of the time step;
   - on the other hand, the stability of implicit methods can be enhanced by under-relaxing the discretized set of algebraic equations either through the use of under relaxation factors or by applying the false-transient approach (Local Time Stepping, LTS).

Properties of the Discretized Equations

5) **Boundedness.** Numerical solutions should lie within proper bounds. Physically non-negative quantities (like density, kinetic energy of turbulence) must always be positive; other quantities, such as concentration, must lie between 0% and 100%.

   - In the absence of sources, some equations (e.g. the heat equation for the temperature when no heat sources are present) require that the minimum and maximum values of the variable be found on the boundaries of the domain. These conditions should be inherited by the numerical approximation.

   - Boundedness is difficult to guarantee. We shall show later on that only some first order schemes guarantee this property. All higher-order schemes can produce unbounded solutions; fortunately, this usually happens only on grids that are too coarse, so a solution with undershoots and overshoots is usually an indication that the errors in the solution are large and the grid needs some refinement (at least locally).

   - The problem is that (high-order) schemes prone to producing unbounded solutions may have stability and convergence problems. These methods should be avoided, if possible.

Properties of the Discretized Equations

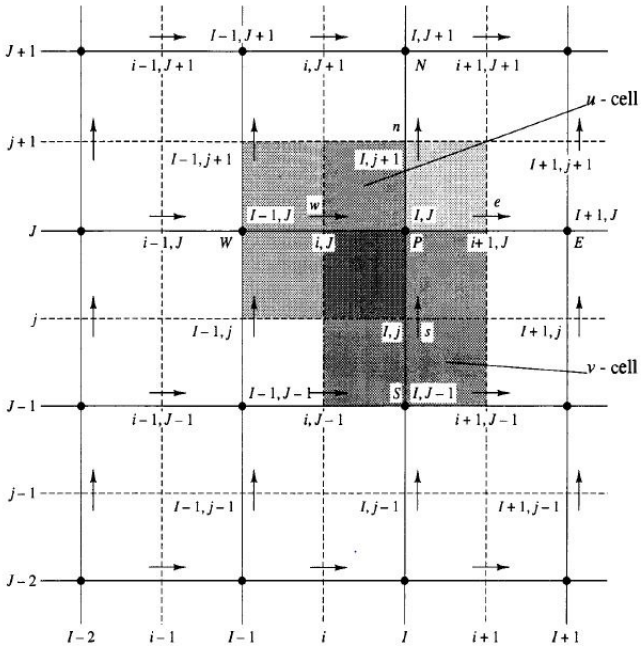
6) **Accuracy.** Numerical solutions of fluid flow and heat transfer problems are only approximate solutions. In addition to the errors that might be introduced in the course of the development of the solution algorithm, in programming or setting up the boundary conditions, numerical solutions always include three kinds of systematic errors:

   - Modeling errors, which are defined as the difference between the actual flow and the exact solution of the mathematical model;

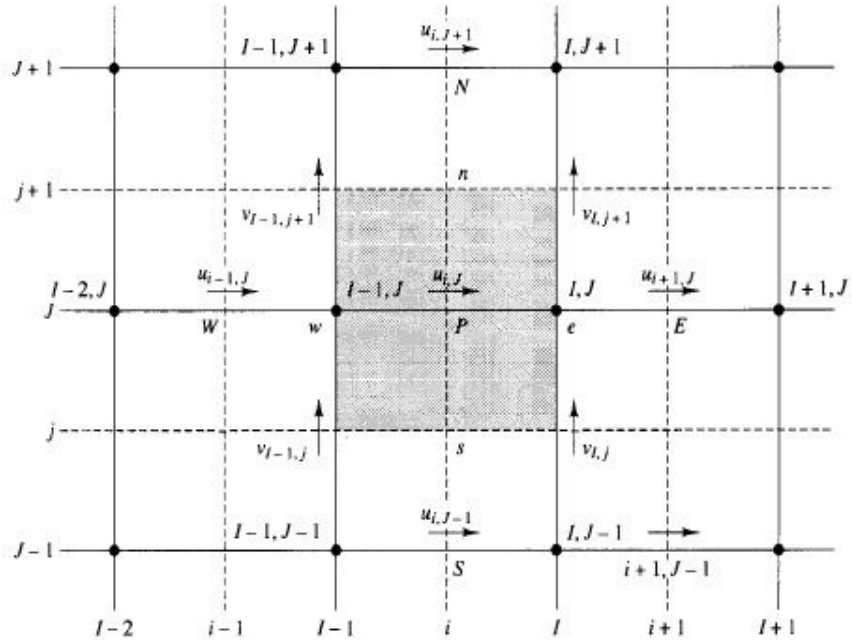
   - Discretization errors, defined as the difference between the exact solution of the conservation equations and the exact solution of the algebraic system of equations obtained by discretizing these equations,

   - Iteration errors, defined as the difference between the iterative and exact solutions of the algebraic equations systems.

Choice of Variable Arrangement on the Grid



Staggered grid arrangement



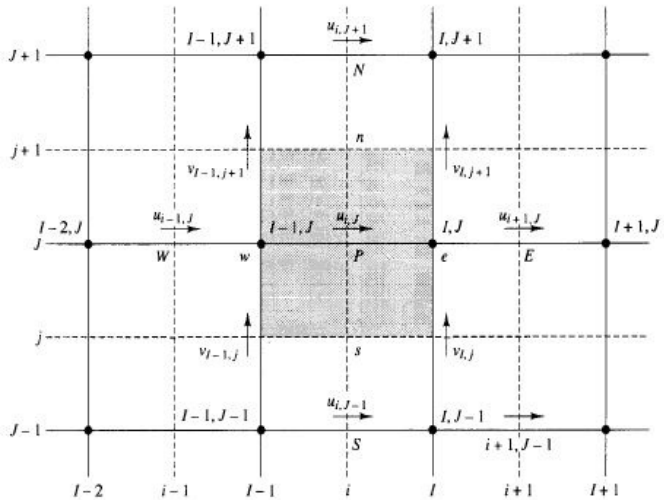
Colocated grid arrangement

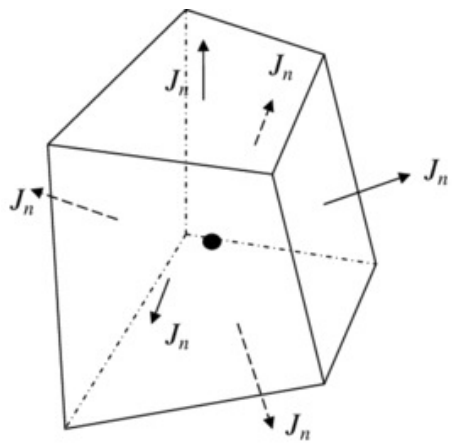
- One of the first issues in FV dicretization is to select the points in the domain at which the values of the unknown dependent variables are to be computed. There is more to this than one might think.
- There are variants of the distribution of computational points within the solution domain. These arrangements may become more complicated when coupled equations for vector fields (like the Navier-Stokes equations) are being solved.

Choice of Variable Arrangement on the Grid

Colocated Arrangement

The obvious choice is to store all the variables at the same set of grid points and to use the same control volumes for all variables; such a grid is called colocated.

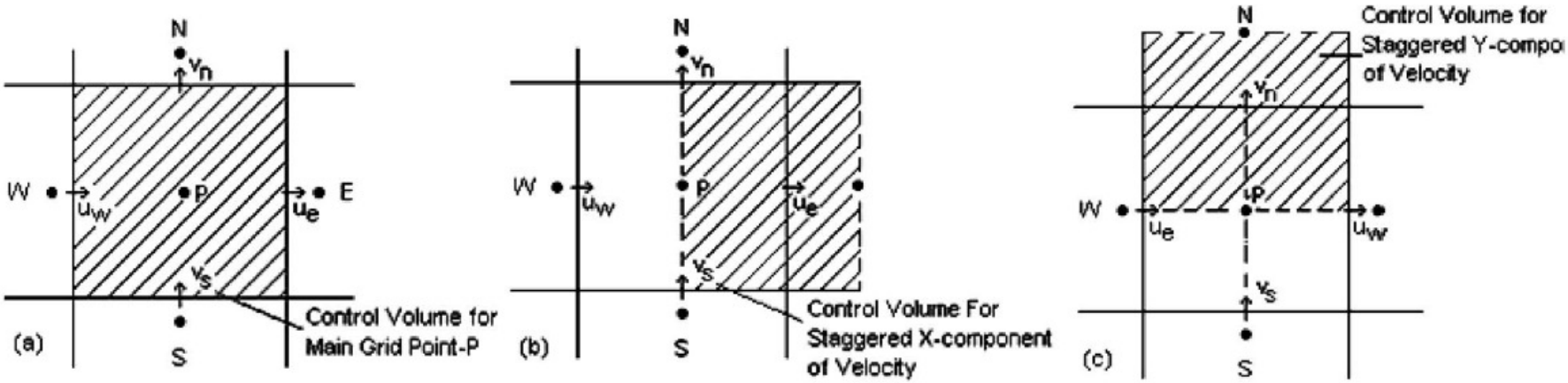




- Since many of the terms in each of the equations are essentially identical, the number of coefficients that must be computed and stored is minimized and the programming is simplified by this choice;
- the colocated arrangement also has significant advantages in **complicated solution domains**, especially when the boundaries have slope discontinuities or the boundary conditions are discontinuous.

Choice of Variable Arrangement on the Grid

Staggered Arrangement

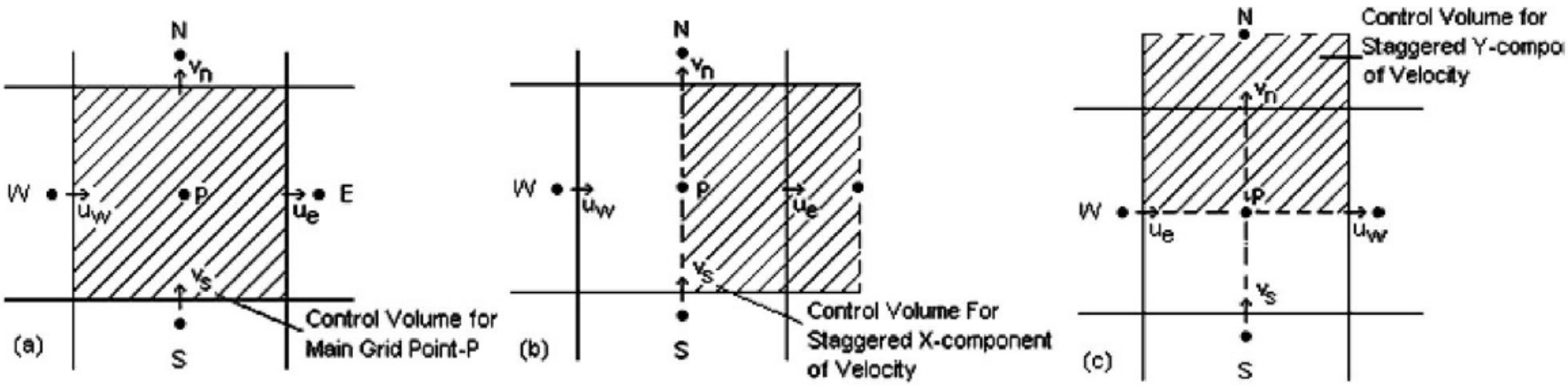


In **Cartesian coordinates**, the staggered arrangement introduced by Harlow and Welsh (1965) offers several advantages over the colocated arrangement:

- several terms that require interpolation with the colocated arrangement, can be calculated (to a second-order approximation) without interpolation;
- Both pressure and diffusion terms are very naturally approximated by CD approximations without interpolation, since the pressure nodes lie at CV face centers and the velocity derivatives needed for the diffusive terms are readily computed at the CV faces
- straightforward evaluation of mass fluxes in the continuity equation on the pressure-CV faces

Choice of Variable Arrangement on the Grid

Staggered Arrangement

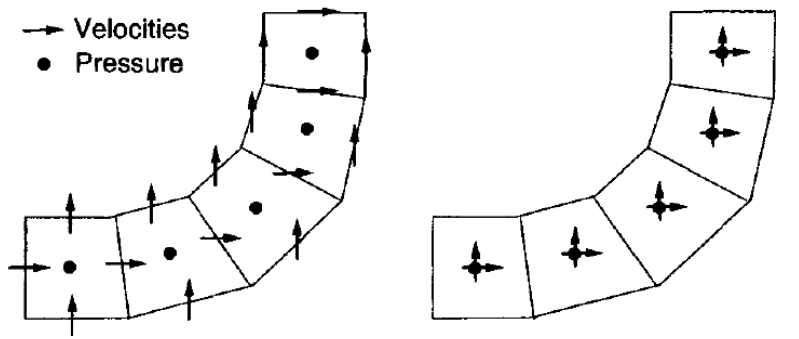


The biggest advantage of the staggered arrangement is the strong coupling between the velocities and the pressure. This helps to *avoid some types of convergence problems and oscillations in pressure and velocity fields*. The numerical approximation on a staggered grid is also conservative of kinetic energy.

Staggered vs Colocated

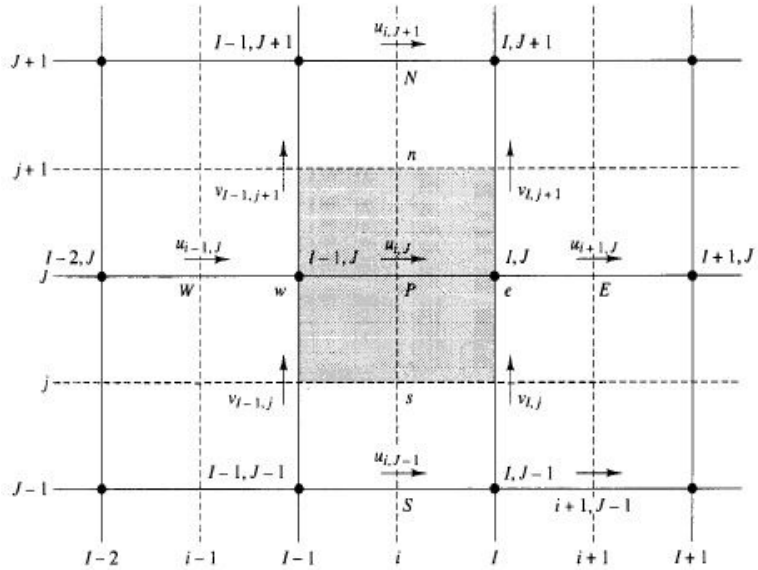
Colocated arrangement:

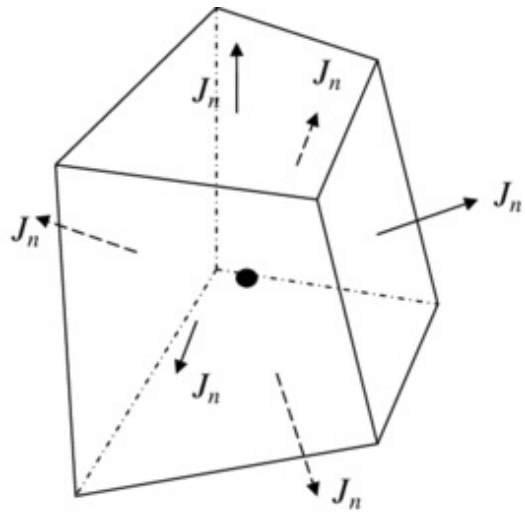
- store all the variables at the same location (CV center) for all variables
- Advantages: easy to code
- Drawbacks: pressure-velocity decoupling, surface-term approximations



- **Staggered arrangement**:
  - pressure and velocity defined at different locations
  - Advantages: strong coupling between pressure and velocity; 2) pressure and diffusion terms are very naturally approximated by CD approximations without interpolation
  - Drawbacks: high-order numerical schemes hard to implement

Variable arrangement in OpenFOAM





- **Discretization** in OpenFOAM is based on the **COLOCATED ARRANGEMENT**, that allows significant advantages in complicated solution domains, especially when the boundaries have slope discontinuities or the boundary conditions are discontinuous.

Differentiation Practices

Interpolation and Differentiation Practices

The approximations to the integrals require the values of variables at locations other than computational nodes (CV centers). The integrand, denoted in the previous sections by f, involves the product of several variables and/or variable gradients at those locations:

convective flux: $f^c = \rho\phi\boldsymbol{v}\cdot\boldsymbol{n}$

diffusive flux: $f^d = \Gamma\nabla\phi$

We assume that the velocity field and the fluid properties ρ and Γ are known at all locations.

- To calculate the convective and diffusive fluxes, the value of and its gradient normal to the cell face at one or more locations on the CV surface are needed.
- Volume integrals of the source terms may also require these values. They have to be expressed in terms of the nodal values by interpolation.
- Numerous possibilities are available; we shall mention a few that are most commonly used. In particular we shall show how the value of and its normal derivative at cell face 'e' can be approximated

Flux Integration Over Element Faces

The **Gauss theorem** is then applied to integrate the **CONVECTION** and **DIFFUSION** term of the governing equations:

Convection term:

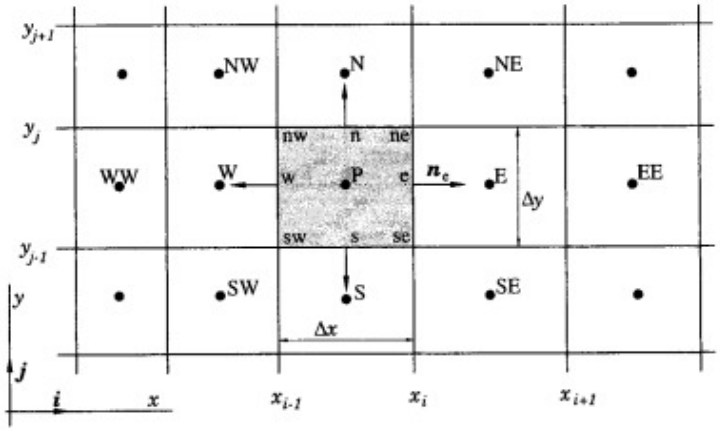
$$\underbrace{\int_V \nabla \cdot \rho(\boldsymbol{u} - \boldsymbol{u_b})\phi \, dV}_{\text{convection term}} = \int_S \rho(\boldsymbol{u} - \boldsymbol{u_b})\phi \cdot \boldsymbol{n} dS$$

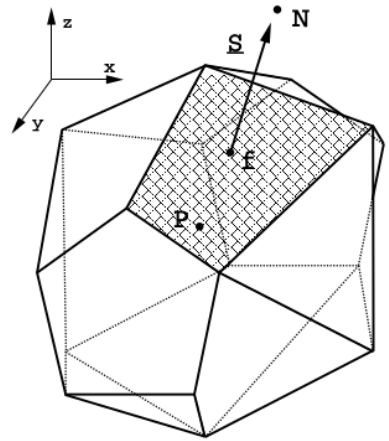
Diffusion term:

$$\underbrace{\int_V \nabla \cdot (\Gamma^\phi \nabla \phi) \, dV}_{\text{convection term}} = \int_S \Gamma^\phi \nabla \phi \cdot \boldsymbol{n} dS$$

The surface fluxes are evaluated at the faces of the element rather than integrated within it: **this transformation has important consequences on the properties of the FVM, one of which is that it renders the method conservative**.

Approximation of surface integrals





To calculate the surface integral exactly, one would need to know the integrand *f* everywhere on the surface S_p.

$$\int_S f d\boldsymbol{S} = \int_S (f \cdot \boldsymbol{n})\, dS = \sum_k (f_k \cdot \boldsymbol{n}_k)\, \omega_k\, S_k$$

This information is not available, as only **node center** values are calculated, so an **APPROXIMATION MUST BE INTRODUCED**. The best way to do it is:

- the integral is approximated in terms of the variable values at the face center
- cell face values are approximated in terms of the CV-centered values

Flux Integration Over Element Faces

Face values are assumend to be **uniform** over the cell surface and **equal** to the value calculated **at a (face-centered) integration point**.

$$\int_S f d\boldsymbol{S} = \int_S (f \cdot \boldsymbol{n})\, dS = \sum_k (f_k \cdot \boldsymbol{n}_k)\, \omega_k\, S_k$$

This treatment, in addition to the assumed variation of ϕ in space around point C, i.e. $\phi = \phi(\boldsymbol{x})$, determines the accuracy of the discretization procedure. In the adopted method, ϕ is assumed to vary linearly in space, i.e.,

$$\phi(\boldsymbol{x}) = \phi_C + (\boldsymbol{x} - \boldsymbol{x}_C) \cdot (\nabla \phi_C)$$

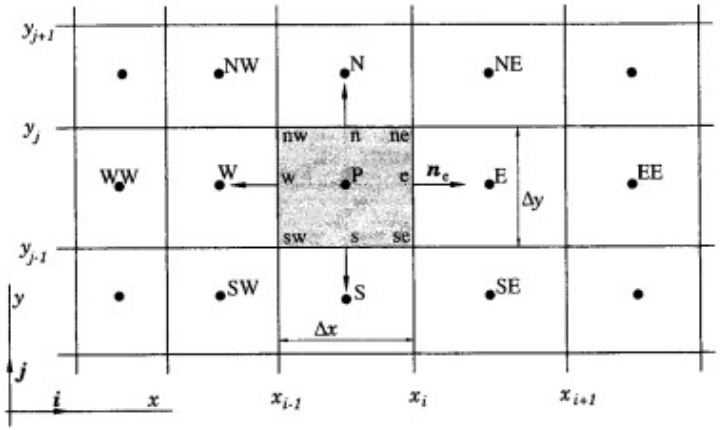
being

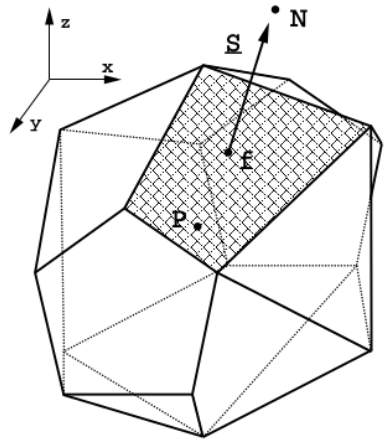
$$\phi_C = \phi(\boldsymbol{x}_C)$$

IMPORTANT: what is the term ω_k in Eq. (23)?

- ω_k represents the weight used for the calculation of the flux: its formulation is dependent on the way to discretize the operators.
- Please have a look at the `fvSchemes` file in OpenFOAM.

Flux Integration Over Element Faces





To proceed further with the discretization, the surface integral at each face of the element in addition to the volume integral of the source term have to be evaluated:

$$\int_S f d\boldsymbol{S} = \int_S (f \cdot \boldsymbol{n})\, dS = \sum_k (f_k \cdot \boldsymbol{n}_k)\, \omega_k\, S_k$$

Fluxes at the faces and sources over the element are evaluated following the mean value approach, i.e., using the value at the centroid of the surface (midpoint rule) and cell, respectively.

→ Generally speaking, the number of integration points along surface *f* may be higher than 1; in OpenFOAM (as in most of the CFD codes!), only one point for each cell face is taken.

Flux Integration Over Element Faces

The simplest approximation to the integral is the **MIDPOINT RULE**: the integral is approximated as a product of the integrand at the cell-face center (which is itself an approximation to the mean value over the surface) and the cell-face area:

$$F_e = \int_S f \cdot \boldsymbol{n} dS = \bar{f}_e S_e \simeq f_e S_e$$

Since the value of *f* is not available at the cell face center '*e*', it has to be obtained by interpolation. In order to preserve the second-order accuracy of the midpoint rule approximation of the surface integral, the value of fe has to be computed with at least second-order accuracy.

Widely used approximations are:

- upwind interpolation;
- linear interpolation;
- higher order interpolations (QUICK, higher order schemes, etc)

Discretization of local sub-problems

Integral equation for a single finite volume

$$\frac{\partial \vec{u}_i}{\partial t} + \frac{1}{V_i} \sum_k \vec{f} \cdot \vec{n}_k dS = q_i$$

where

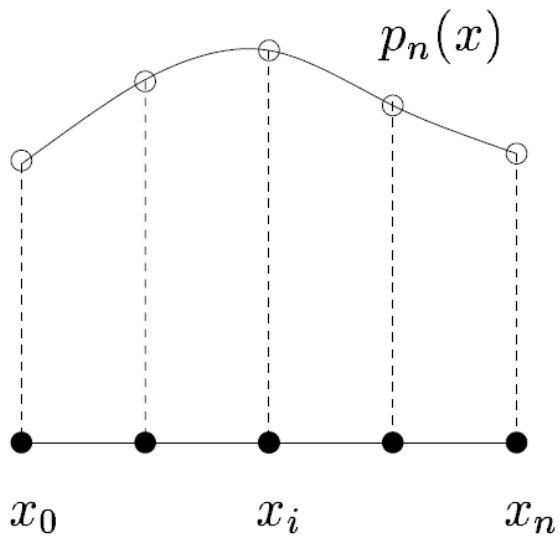
$$u_i = \frac{1}{|V_i|} \int_{V_i} u dV, \qquad q_i = \frac{1}{|V_i|} \int_V q dV$$

- the integral conservation law is satisfied for each CV and for the entire domain
- to obtain a linear system, integrals must be expressed in terms of mean values

NUMERICAL INTEGRATION:

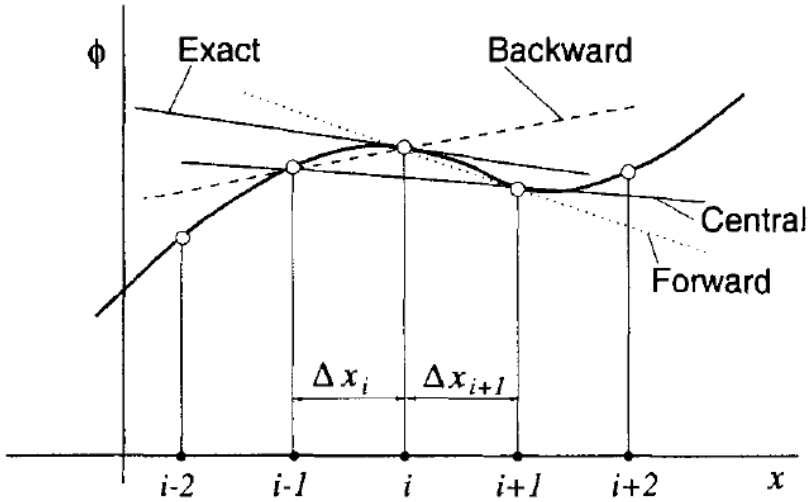
$$\int_V f(\vec{x}) dV \simeq \sum_{i=0}^{n} \omega_i f(\vec{x}_i)$$

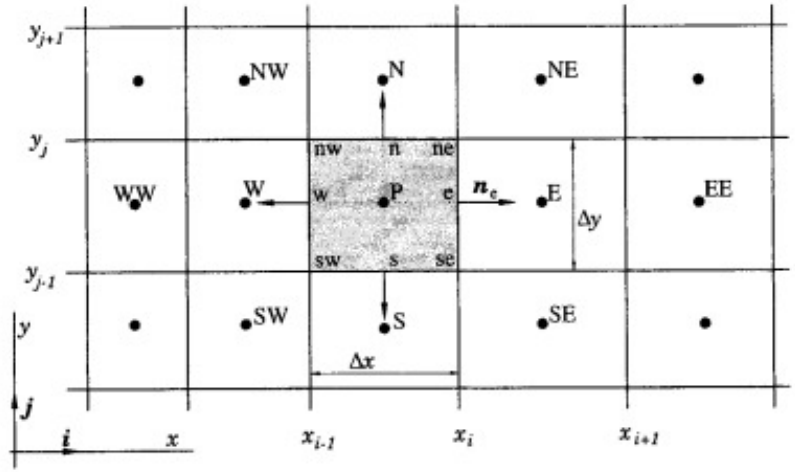
where ω_i are the weights and $\vec{x}_i$ are the nodes of the quadrature rule.



Such formulae can be derived by exact integration of an interpolation polynomial.

Finite Differences: Basic Concepts





The idea behind finite difference approximations is borrowed directly from the definition of a derivative:

$$\left(\frac{\partial \phi}{\partial x}\right)_{x_i} = \lim_{\Delta x \to 0} \frac{\phi(x_i + \Delta x) - \phi(x_i)}{\Delta x}$$

The first derivative $\frac{\partial \phi}{\partial x}$ at a point is the slope of the tangent to the curve $\phi(x)$ at that point, the line marked 'exact' in the figure. Its slope can be approximated by the slope of a line passing through two nearby points on the curve. The dotted line shows approximation by a forward difference; the derivative at x_i is approximated by the slope of a line passing through the point x_i and another point at $x_i \pm \Delta x$. The line labeled 'central' uses the slope of a line passing through two points lying on opposite sides of the point at which the derivative is approximated.

Temporal derivatives

Temporal derivatives: first order

First-order - `Euler`

The discrete form of the transient term by first-order implicit Euler method for a generic conservation equation reads::

$$\frac{\partial}{\partial t}\int_{\tilde{\Omega}(t)} \phi \, \mathrm{d}V \approx \frac{V^{n+1}\phi^{n+1} - V^{n}\phi^{n}}{\Delta t}$$

n is the time index ($t^n = t_0 + n\Delta t$) and V^n the cell volume at time t^n.
$\rightarrow$ The formal order of accuracy of the method is 1 and the method is bounded and unconditionally stable.

Temporal derivatives: second order

Second Order Upwind Euler - `backward`

The Second-order Upwind Euler (SOUE) is a two-step Adams-Moulton method, where a linear combination of the current-time and the old-time derivatives that yields the following expression for the transient term:

$$\frac{\partial}{\partial t}\int_{\tilde{\Omega}(t)} \phi \, \mathrm{d}V \approx \frac{1}{\Delta t}\left(\frac{3}{2}V^{n+1}\phi^{n+1} - 2V^{n}\phi^{n} + \frac{1}{2}V^{n-1}\phi^{n-1}\right) \tag{1}$$

$\rightarrow$ The SOUE method is formally second-order and unbounded.

Time derivatives: Crank-Nicolson

Crank-Nicolson - `crankNicolson`

The Crank-Nicolson method uses a linear interpolation in time according to the scheme:

$$\frac{\partial}{\partial t}\int_{\tilde{\Omega}(t)} \phi \, \mathrm{d}V \approx (1+\theta)\frac{V^{n+1}\phi^{n+1} - V^{n}\phi^{n}}{\Delta t} - \theta V^{n}\mathcal{F}(\phi^{n})$$

being:

$$\mathcal{F}(\phi^{n}) = \left[(1+\theta)\frac{V^{n}\phi^{n} - V^{n-1}\phi^{n-1}}{\Delta t} - \theta V^{n-1}\mathcal{F}(\phi^{n-1})\right]\frac{1}{V^{n-1}}$$

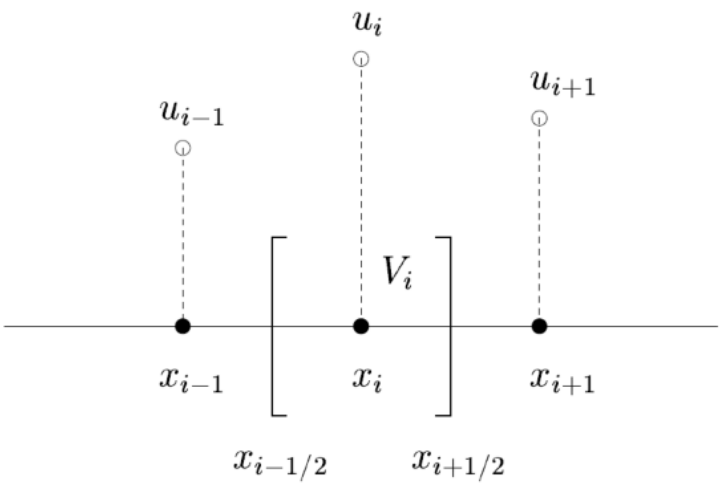
where θ is the off-center coefficient. In a practical implementation, $\mathcal{F}(\phi^{n-1})$ the stored value of the time derivative computed at t^{n-1}.

$\rightarrow$ For $\theta = 0$ the scheme is equivalent to the first order implicit Euler;

$\rightarrow$ for $\theta = 1$ the centered Crank-Nicolson method, with a formal second-order accuracy, is recovered.

Spatial discretization

Interpolation Techniques



In a FVM based on a **colocated grid arrangement**, the solution is available only at computational nodes (CV centers). Interpolation is needed to obtain the function values at quadrature points.

1) Volume Integrals: $u_i = \frac{1}{|V_i|}\int_{V_i} u dV \simeq u(\boldsymbol{x_i})$ $\rightarrow$ (midpoint rule)

2) Surface integrals $\boldsymbol{f} = \boldsymbol{u}\phi - \Gamma\nabla\phi \quad \rightarrow \quad \frac{1}{|V_i|}\sum_k \int_{S_k} \boldsymbol{f}\cdot\boldsymbol{n}_k dS = I_c + I_d$

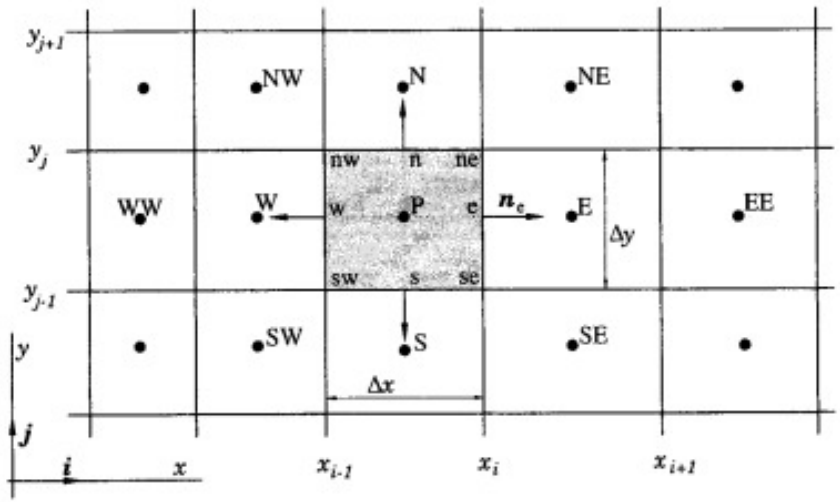
being:

$$I_c = \frac{1}{|V_i|} = \sum_k \int_{S_k} (\phi\cdot\boldsymbol{n}_k)\boldsymbol{u} dS, \qquad I_d = \frac{1}{|V_i|}\sum_k \int_{S_k} \Gamma\,(\vec{n_k}\cdot\nabla\phi) dS$$

Problem: how to define the values $u_{i\pm 1/2}$ at the cell faces?

Upwind Interpolation (UDS)

Approximating ϕ_e, by its value at the node upstream of 'e' is equivalent to using a backward- or forward-difference approximation for the first derivative (depending on the flow direction), hence the name *upwind differencing scheme* (UDS) for this approximation.



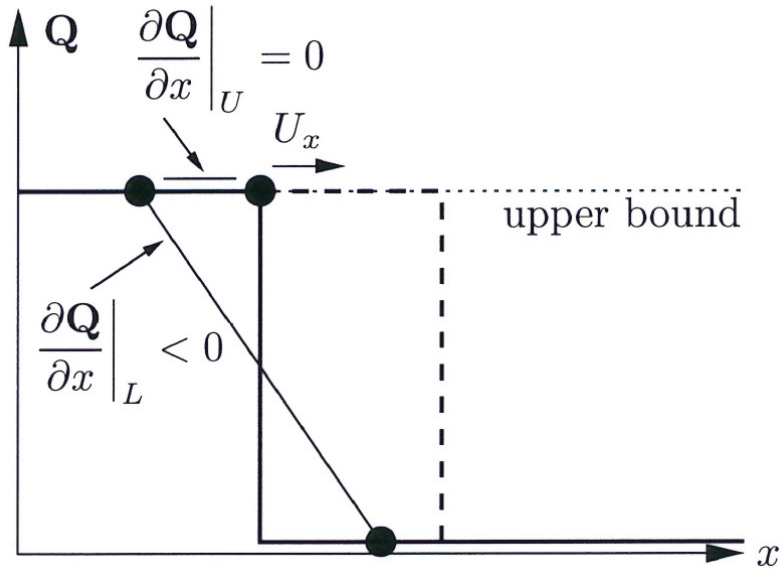
In UDS, ϕ_e, is approximated as:

$$\phi_e = \begin{cases} \phi_P & if \quad (\boldsymbol{v} \cdot \boldsymbol{n})_e > 0 \\ \phi_E & if \quad (\boldsymbol{v} \cdot \boldsymbol{n})_e < 0 \end{cases}$$

This is the only approximation that unconditionally satisfies the boundedness criterion i.e. it will never yield oscillatory solutions. However, it achieves this by being numerically diffusive.

Diffusion in Upwind Interpolation (UDS)

Upwind Interpolation (UDS) unconditionally satisfies the boundedness criterion i.e. it will never yield oscillatory solutions. However, it achieves this by being numerically diffusive.



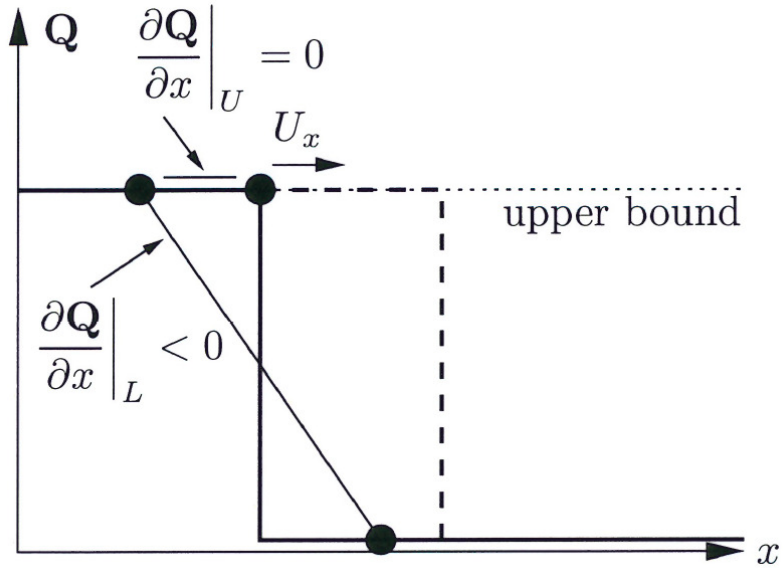
For Cartesian grids, Taylor series expansion about P gives:

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

where H denotes higher-order terms.

Upwind Interpolation (UDS)

The UDS approximation retains only the first term on the right-hand side, so it is a first-order scheme.

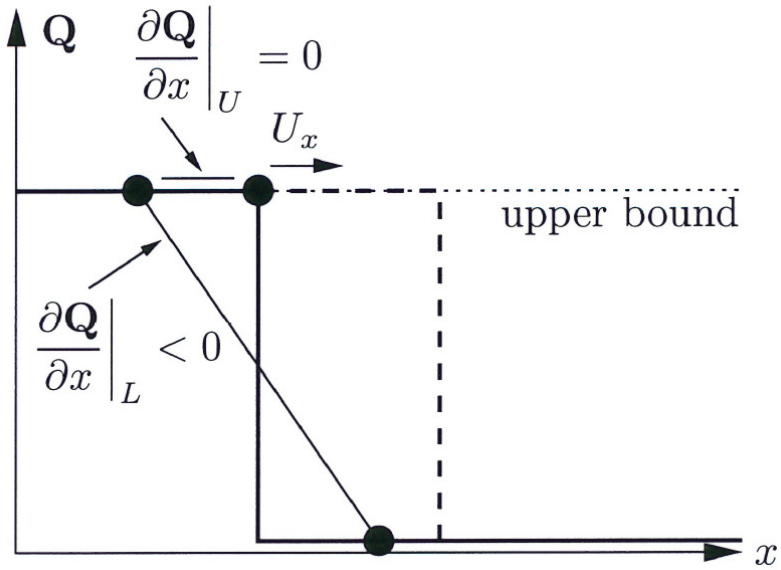


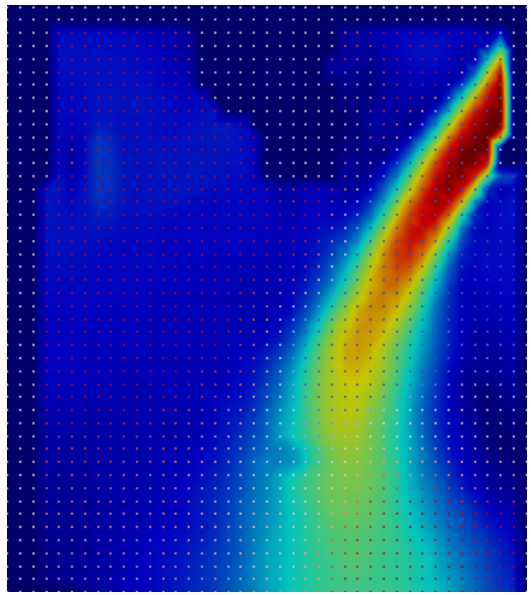
Its leading truncation error term is diffusive i.e. it resembles a diffusive flux:

$$f_e^d = \Gamma_e \left(\frac{\partial \phi}{\partial x}\right)_e$$

The coefficient of numerical, artificial, or false diffusion is $\Gamma_e^{num} = (\rho u)\Delta x/2$.

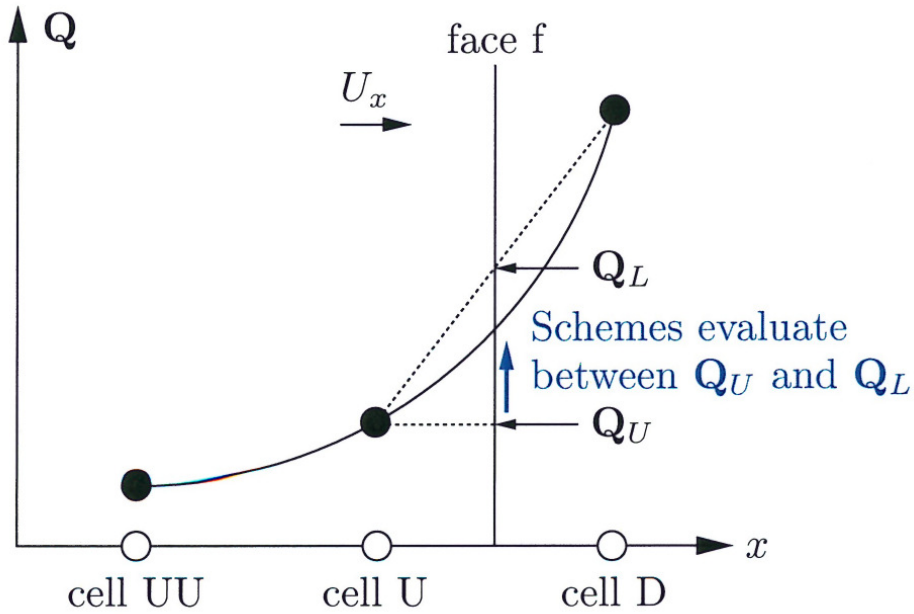
Upwind Interpolation (UDS)





- This numerical diffusion is magnified in multidimensional problems if the flow is oblique to the grid; the truncation error then produces diffusion in the direction normal to the flow as well as in the streamwise direction, a particularly serious type of error.
- Peaks or rapid variations in the variables will be smeared out and, since the rate of error reduction is only first order, very fine grids are required to obtain accurate solutions.

Linear Interpolation (CDS)



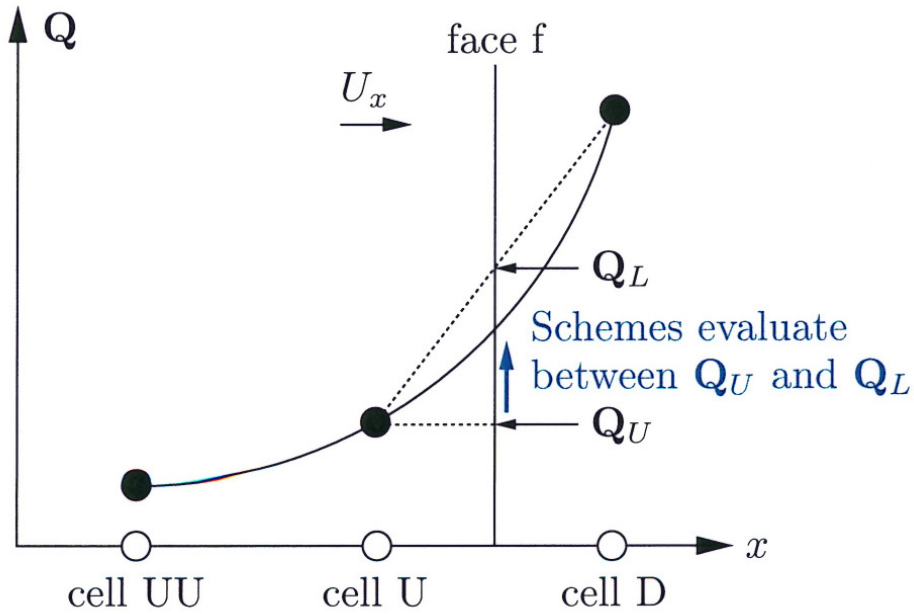
Another straightforward approximation for the value at CV-face center is linear interpolation between the two nearest nodes. At location 'e' on a Cartesian grid:

$$\phi_e = \phi_E \lambda_e + \phi_P (1 - \lambda_e)$$

being the linear interpolation factor λ_e defined as:

$$\lambda_e = \frac{x_e - x_P}{x_E - x_P}$$

Linear Interpolation (CDS)



The linear interpolation factor λ_e

$$\lambda_e = \frac{x_e - x_P}{x_E - x_P}$$

is second-order accurate as can be shown by using the Taylor series expansion of ϕ_E about the point x_p to eliminate the first derivative in the Taylor expansion about P. The result is:

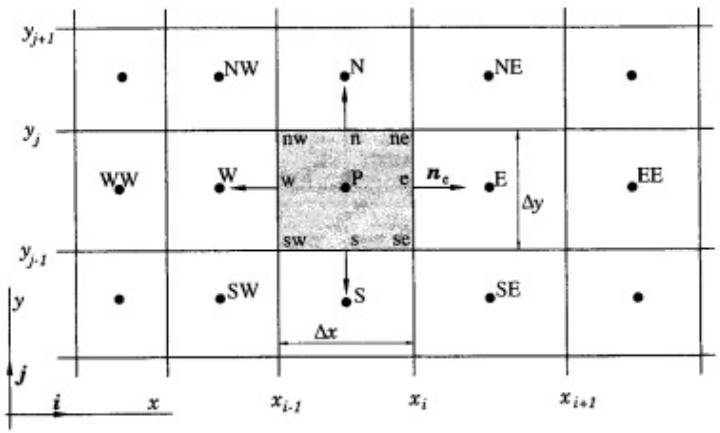
$$\phi_v = \phi_E \lambda_e + \phi_P (1 - \lambda_e) - \frac{(x_e - x_P)(x_E - x_e)}{2} \left(\frac{\partial^2 \phi}{\partial x^2} \right)_P + H$$

Linear Interpolation (CDS)

By using Taylor series expansion around ϕ_e, one can show that truncation error of the above approximation is:

$$\epsilon_\tau = \frac{(x_e - x_P)^2 - (x_E - x_e)^2}{2\,(x_E - x_P)} \left(\frac{\partial^2 \phi}{\partial x^2}\right)_e - \frac{(x_e - x_P)^3 - (x_E - x_e)^3}{6\,(x_E - x_P)} \left(\frac{\partial^3 \phi}{\partial x^3}\right)_e + H$$

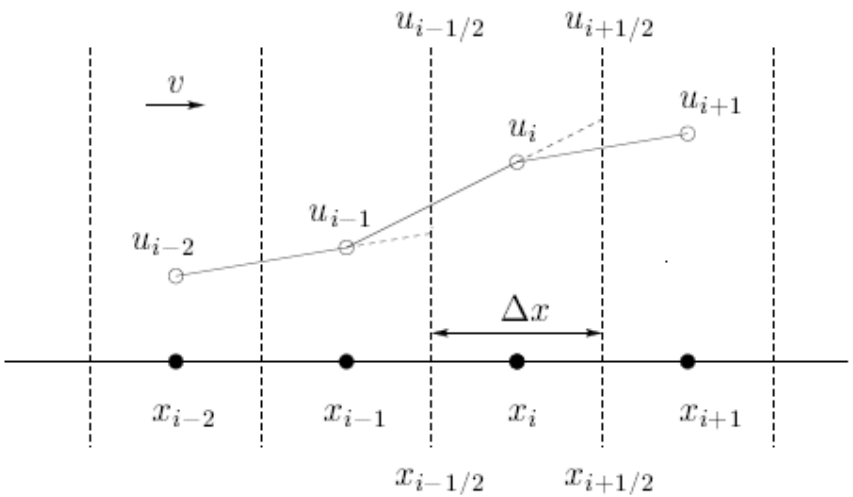
When the location 'e' is midway between P and E (for example on a uniform grid), the approximation is of second-order accuracy, since the first term on the right-hand side vanishes and the leading error term is then proportional to Δx^2.



- when the grid is non-uniform, the leading error term is proportional to the product of Δx and the grid expansion factor minus unity;
- in spite of the formal first-order accuracy, the error reduction when the grid is refined is SIMILAR to that of a second-order approximation even on non-uniform grids.

Linear upwind difference scheme (LUDS)

Piecewise-constant solution



Please see slide 33 for the nomenclature.

Upwind-biased extrapolation

$u > 0$

$$\phi_{i-1/2} \simeq \frac{3\phi_{i-1} - \phi_{i-2}}{2}$$

$$\phi_{i+1/2} \simeq \frac{3\phi_i - \phi_{i-1}}{2}$$

$$I_c \simeq \frac{3\phi_i - 4\phi_{i-1} + \phi_{i-2}}{2\Delta x}$$

$u < 0$

$$\phi_{i-1/2} \simeq \frac{3\phi_i - \phi_{i+1}}{2}$$

$$\phi_{i+1/2} \simeq \frac{3\phi_{i+1} - \phi_{i+2}}{2}$$

$$I_c \simeq -\frac{3\phi_i - 4\phi_{i+1} + \phi_{i+2}}{2\Delta x}$$

- LUDS is second-order accurate, equivalent to the one-sided 3-point finite difference;
- the matrix is no longer tridiagonal (shifted, upper/lower triangular if I_d=0);
- defect correction: $I^{m+1}_{LUDS} = I^{m+1}_{UDS} + \left[I^m_{LUDS} - I^m_{UDS}\right]$

Source Volume Integration

INTEGRAL OF THE VOLUME TERMS

Volume integration is used for the source term. Adopting a Gaussian quadrature integration, the volume integral of the source term is computed as:

$$\int_V Q^\phi \, dV = \sum_{i \in pts(V)} (Q_i^\phi \, \omega_i \, V)$$

As with surface flux integration, different options for volume integration are available with an accuracy that is depending on the number of integration points *i* used and the weighing function ω_i. The accuracy increases with increasing the number of integration points but so does the computational cost.

In OpenFOAM, one point Gauss integration is used and $\omega_i = 1$ with the integration point located at the centroid of the Control Volume. This approximation is second order accurate and is applicable in two- and three- dimensions.

Source Volume Integration

While the above terms can be discretized with any specified number of integration points, it is customary for the FV Method to use one integration point, yielding second order accuracy. This was found to be a good compromise between accuracy and flexibility while keeping the method simple and relatively of low computational cost.

$$\sum_{i \in pts(V)} (Q_i^{\phi}\, \omega_i\, V) = \sum_{f} (Q_f^{\phi}\, \omega_f\, V)$$

Source Volume Integration

- The simplest second-order accurate approximation is to replace the volume integral by the product of the mean value of the integrand and the CV volume and approximate the former as the value at the CV center.

- **Using the mid-point integration approximation**, the semi-discrete steady state finite volume equation can be finally simplified to:

$$Q_P = \int_V q \, dV = \bar{q} \Delta V \simeq q_P \Delta V$$

  where q_p stands for the value of q at the CV center. This quantity is easily calculated; since all variables are available at node P, no interpolation is necessary.

- The above approximation becomes exact if q is either constant or varies linearly within the CV; otherwise, it contains a second-order error, as is easily shown.

Discretization of the Navier Stokes Equations

FV Solution of the Navier Stokes Equations

The unsteady and advection terms in the momentum equations have the same form as in the generic conservation equation.

- The **diffusive (viscous) terms** are similar to their counterparts in the generic equation but, because the momentum equations are vector equations, these contributions become a bit more complex and their treatment needs to be considered in more detail.
- The **momentum equations** also contain a contribution from the pressure, which has no analog in the generic equation. It may be regarded either as:
  - a **source term** (treating the pressure gradient as body force, nonconservatively) as a surface force (conservative treatment) but, due to the close connection of the pressure and the continuity equation, it requires special attention.
- Finally, the fact that the principal variable is a vector allows more freedom in the choice of a grid.

Convective Terms: the Picard Approach

The convective term in the momentum equation is non-linear.

$$\frac{\partial(\rho u_i u_j)}{\partial x_j} \quad \text{and} \quad \int_S \rho u_i \boldsymbol{v} \cdot \boldsymbol{n} \cdot \boldsymbol{n} dS$$

Nonlinear terms (convective fluxes, source terms) in the coupled non-linear equations are solved by a sequential decoupled method and **are usually linearized using Picard iteration approach**. A typical example is given by the convective terms in the momentum equation that appears in OpenFOAM as:

```
fvm::div(phi,U)
```

begin `phi` = $(\rho u_j)^*$ and `U` = u_i. The mass flux is treated as known, so the non-linear convective term in the equation for the u_i momentum component is approximated by:

$$\rho u_j u_i \simeq (\rho u_j)^* u_i$$

where index * denotes that the values are taken from the result of the previous outer iteration.

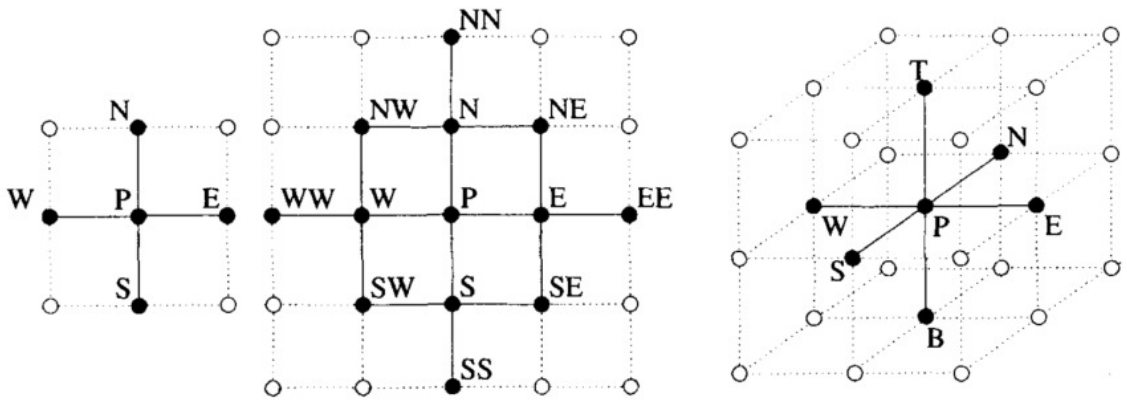
This kind of linearization, together with the discretization methods commonly employed in CFD codes, requires to iterate and under-relax variables to achieve the solution of the equations.

Deferred-Correction Methods

If all terms containing the nodal values of the unknown variable are kept on the left-hand side of the equation:

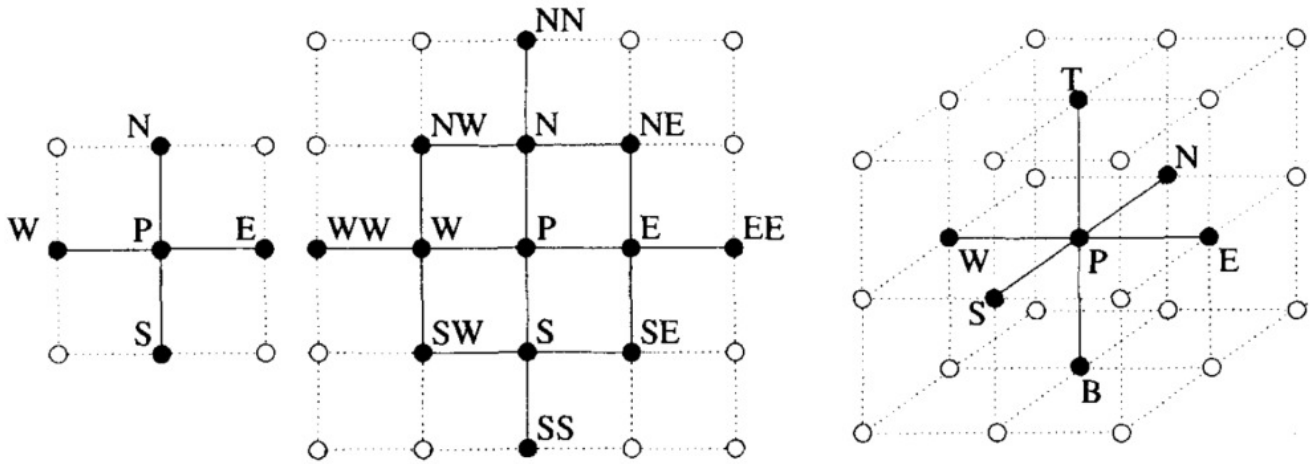
$$A_P \phi_P + \sum_l A_l \phi_l = Q_P$$

so the computational molecule may become very large.



- Since the size of the computational molecule affects both the storage requirements and the effort needed to solve the linear equation system, we would like to keep it as small as possible;
- often, only the nearest neighbors of node P are kept on the left hand sides of the equations; approximations which produce such simple computational molecules are usually not accurate enough, so we are forced to use approximations that refer to more nodes than just the nearest neighbors.

Deferred-Correction Methods



- One way around this problem would be to leave only the terms containing the nearest neighbors on the left-hand side of $\boldsymbol{AX} = \boldsymbol{B}$ and bring all other terms to the right-hand side; this requires that these terms be evaluated using values from the previous iteration.
- To prevent divergence, strong under-relaxation of the changes from one iteration to the next would be required under-relaxation, resulting in slow convergence.

Deferred-Correction Methods

The usual approach to the solution of coupled non-linear equations

$$\int_\Omega \rho \boldsymbol{u}\, d\Omega + \int_S \rho \boldsymbol{v}\,(\boldsymbol{v}-\boldsymbol{v_b})\cdot \boldsymbol{n} dS = \int_S T\cdot \boldsymbol{n} dS + \int_\Omega \rho \boldsymbol{b}\, d\Omega$$

is the sequential decoupled method described in the previous section. **Non-linear terms (convective flux, source term) are usually linearized using Picard iteration approach.**

a) For the **convective terms**, this means that the mass flux is treated as known, so the non-linear convective term in the equation for the ui momentum component is approximated by:

$$\rho u_j u_i \simeq (\rho u_j)^0 u_i$$

where index 0 denotes that the values are taken from the result of the previous outer iteration.

b) Similarly, the **source term** is decomposed into two parts:

$$q_\phi = b_0 + b_1 \phi$$

The portion b_0 is absorbed into the right hand side of the algebraic equation, while b_l contributes to the coefficient matrix $\boldsymbol{A}$.

This kind of linearization requires to iterate until the final solution.

Deferred-Correction Methods

Newton's method is sometimes used to linearize the non-linear terms; for example, the convective term along the $i-th$ direction in the momentum equation can be expressed as:

$$\rho u_j u_i \simeq \rho u_j^0 u_i - \rho u_j^0 u_i^0$$

Non-linear source terms can be treated in the same way; this leads to a coupled linear system of equations which is difficult to solve, and the convergence is not quadratic unless the full Newton technique is used.

In OpenFOAM, linearization of non-linear terms is very apparent in the convective terms is included in the term:

```
div(phi,U)
```

The first argument of the function call (`phi`) is treated explicitly, i.e. is the flux at the previous time step; the second argument, `U`, is updated at each iteration.

Deferred-Correction Methods in OpenFOAM

IMPORTANT NOTE: the deferred correction methods and the iterative solution of the governing equations are implemented in OpenFOAM at the **solver level** (see `$FOAM_SOLVERS`). Iterative methods for the solution of the linear systems are included in the classes (`FOAM_SRC`) for the linear algebra.

Be careful: iterative methods for the solution of the coupled equations are a different thing with respect to iterative methods for the solution of linear systems!

Viscous Terms

The **viscous terms** in the momentum equations correspond to the diffusive term in the generic equation; their differential and integral forms are

$$\frac{\partial \tau_{ij}}{\partial x_j} \qquad \text{and} \qquad \int_S (\tau_{ij} \cdot \boldsymbol{j}_i) \cdot \boldsymbol{n} dS$$

where, for a Newtonian fluid:

$$\tau_{ij} = \nu \left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i} \right)$$

- Because the momentum equations are vector equations, the viscous term is more complicated than the generic diffusive term.
- The part of the viscous term in the momentum equations which corresponds to the diffusive term in the generic conservation equation is

$$\frac{\partial}{\partial x_j} \left(\mu \frac{\partial u_i}{\partial x_j} \right) \qquad \text{and} \qquad \int_S \mu \frac{\partial u_j}{\partial x_i} \boldsymbol{i}_j \cdot \boldsymbol{n} dS$$

Pressure Terms and Body Forces

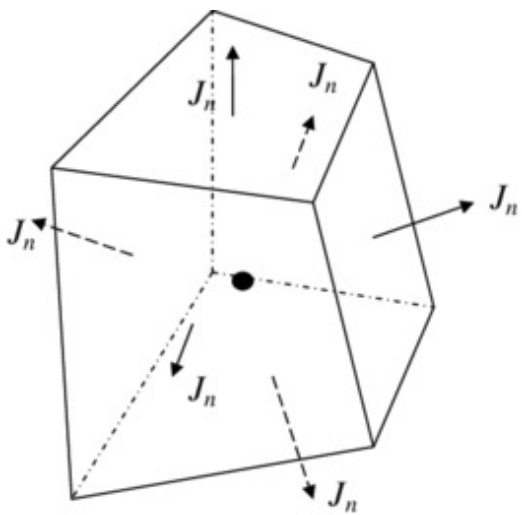
BODY FORCES

Other body forces, like the non-conservative ones arising when covariant or contravariant velocities are used in non-Cartesian coordinate systems are are integrated over the CV volume.

Usually, the mean value approach is used, so that the value at CV center is multiplied by cell volume.

If these sink terms involve the unknowns, they can be treated

- `implicitly`, if the contribution of the sink term to the central coefficient A_p (coefficient matrix) in the discretized equation is positive, in order to avoid destabilization of the iterative solution scheme by reducing the diagonal dominance of the matrix;
- `explicitly`, if contribution to extra-diagonal terms is not negligible.



In some cases the non-conservative terms, considered as body forces, dominate the transport equation (like in swirling flows). The treatment of the non-linear source terms and the variable coupling may then become very important.

Pressure Terms and Body Forces

PRESSURE TERMS

CFD solvers (e.g. OpenFOAM) usually deal with the "pressure" in terms of the combination:

$$p - \rho_0 \boldsymbol{g}\boldsymbol{h} + \mu \frac{2}{3} \nabla \boldsymbol{v}$$

being, *for incompressible flows*, $\mu \frac{2}{3} \nabla \boldsymbol{v} = 0$.

In FV methods, the pressure term is usually treated as a surface force (conservative approach), i.e. in the equation for u_i the integral:

$$- \int_S p \boldsymbol{i} \cdot \cdot \boldsymbol{n} dS$$

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.

Pressure Terms and Body Forces

PRESSURE TERMS

CFD solvers (e.g. OpenFOAM) usually deal with the "pressure" in terms of the combination:

$$p - \rho_0 \boldsymbol{g}\boldsymbol{h} + \mu \frac{2}{3} \nabla \boldsymbol{v}$$

being, *for incompressible flows*, $\mu \frac{2}{3} \nabla \boldsymbol{v} = 0$.

Alternatively, the pressure can be treated non-conservatively, by retaining the above integral in its volumetric form:

$$- \int_\Omega \nabla p \cdot \boldsymbol{i} \, d\Omega$$

In this case, the derivative (or, for non-orthogonal grids, all three derivatives) needs to be approximated at one or more locations within the CV. The nonconservative approach introduces a global non-conservative error; although this error tends to zero as the grid size goes to zero, it may be significant for finite grid size.

Building the Linear System

Building the Linear System

$$\underbrace{\underbrace{\frac{\partial}{\partial t}\int_V \rho\phi \, dV}_{\texttt{temporal derivative}} + \underbrace{\int_V \nabla\cdot\rho(\boldsymbol{u}-\boldsymbol{u_b})\phi \, dV}_{\texttt{convection term}}}_{\frac{D\phi}{Dt}} = \underbrace{\int_V \Gamma^\phi \nabla\phi \, dV}_{\texttt{diffusion term}} + \underbrace{\int_V Q^\phi \, dV}_{\texttt{source/sink terms}}$$

After **EACH** integral form of the conservation of a scalar quantity ϕ is discretized, it is written in the form of a linear equation.

- N equations (N is the number of CVs) must be solved at the same time; each (linear) equation must be **solved for each CV**;
- the N equations must be converted into **a system of linear equations**

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} & \dots & a_{1n} \\ a_{21} & a_{22} & a_{23} & \dots & a_{2n} \\ \vdots & \vdots & \mathbf{A} & \vdots & \vdots \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & a_{n3} & \dots & a_{nn} \end{bmatrix} \begin{bmatrix} \phi_1 \\ \phi_2 \\ \mathbf{X} \\ \vdots \\ \phi_n \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ \mathbf{B} \\ \vdots \\ b_n \end{bmatrix}$$

- the coefficient of the matrix A are constant and their stencil depends on the differencing scheme adopted for the discretization of the operators.

Building the Linear System

Why is the structure of the mesh so important? **It is strongly connected with the computation of the operators**, like the gradient.

$$AX=B$$

A profile for the variation of the the mesh topology/structure has a relevant influence on:

- the building of the coefficient matrix of the linear system.
- array storage/allocation in computer memory

*Mesh Topology is how a mesh is constructed: structured, unstructured, polyhedral, multiblock, etc...

Example: computation of the gradient

Example:

Many techniques can be used to compute the gradient of an element field. For example, we consider the calculation of the gradient based on the Green-Gauss theorem:

$$\overline{\nabla\phi}_C = \frac{1}{V_C}\int_V \nabla\phi \, dV = \frac{1}{V_C}\int_S \phi \cdot \boldsymbol{n} dS \approx \frac{1}{V_C}\sum_{f_i} \overline{\phi}_f S_f$$

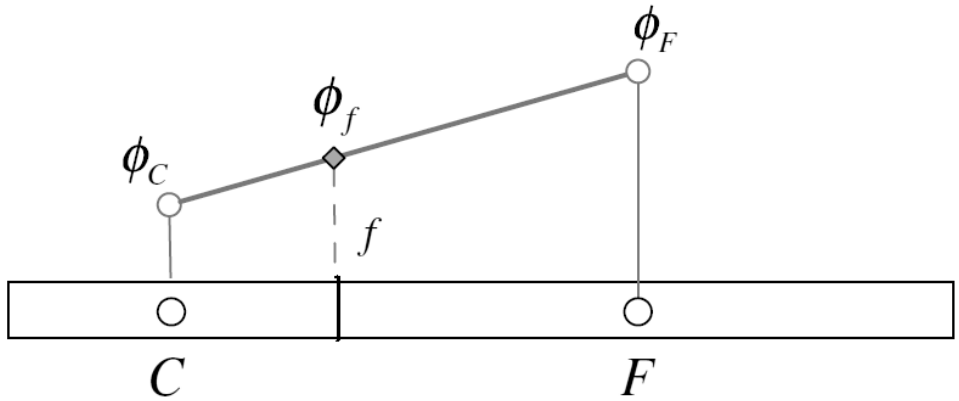
- Gradient calculation based on the **Green-Gauss theorem** is relatively straightforward and can be used for a variety of topologies and grids (structured/unstructured, orthogonal/nonorthogonal).
- It is clear that to compute the average of the gradient over the control element C, information about the face area and direction $\boldsymbol{S}$ is required, as well as information about the neighboring elements and the fluxes at the centroids of the faces.
- The value of ϕ will have to be computed at each face centroid.
- Assuming a linear profile for the variation of ϕ between two centroids, an approximate value for ϕ_f is:

$$\overline{\phi}_f = g_A\phi_A + g_B\phi_B$$

Weighting factors may be calculated as:

$$g_A = \frac{V_A}{V_A + V_B} \quad \text{and} \quad g_B = \frac{V_B}{V_A + V_B} = 1 - g_A$$

Face Weighting Factors



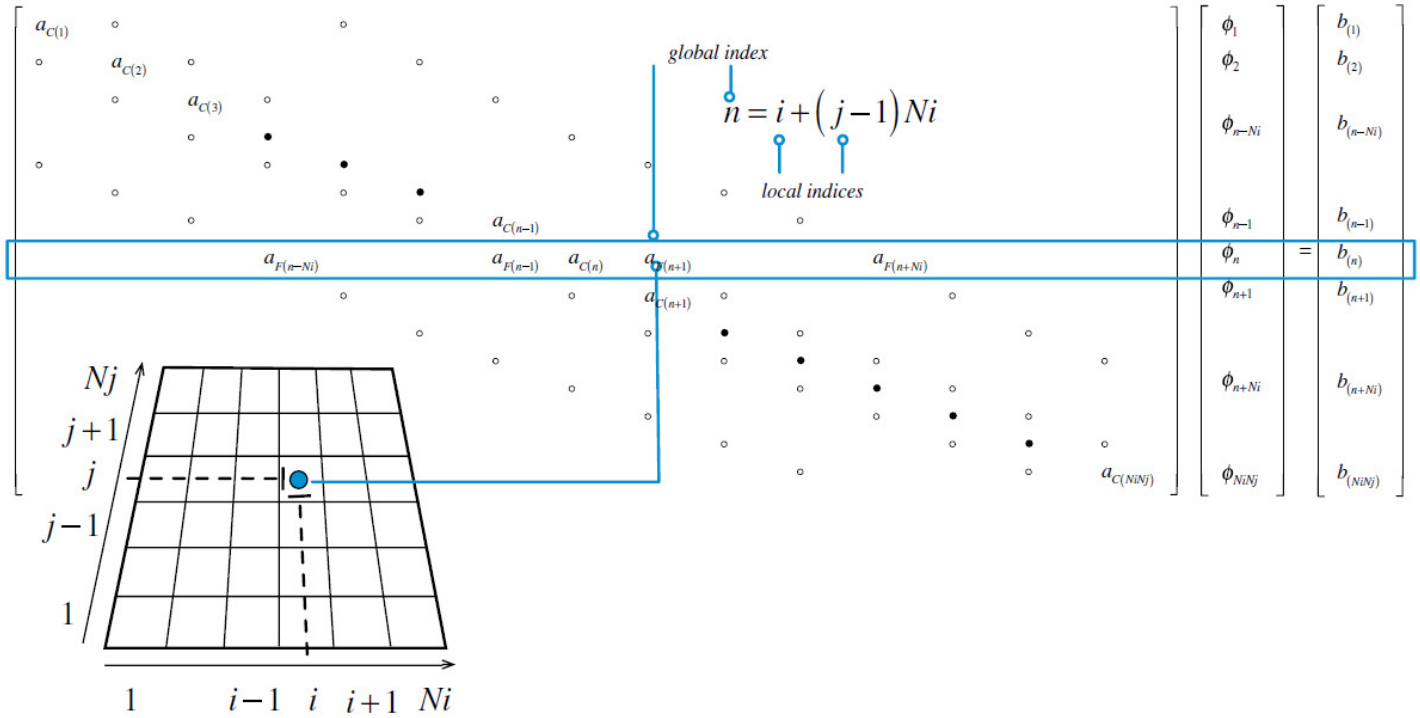
- In a FV mesh, values of a general quantity ϕ are known at the control volume centroids C and F, and are to be used to compute the value of ϕ_f at the cell interface.
- A simple linear interpolation will result in the following formula:

$$\phi_f = g_f \phi_F + (1 - g_f)\phi_C$$

where g_f is the face weighting factor.

Face weighting factors depend on the discretization method used ($\rightarrow$ see file `fvSchemes` in your `FOAM_TUTORIALS/<yourTutorial>`)

Structured (regular)

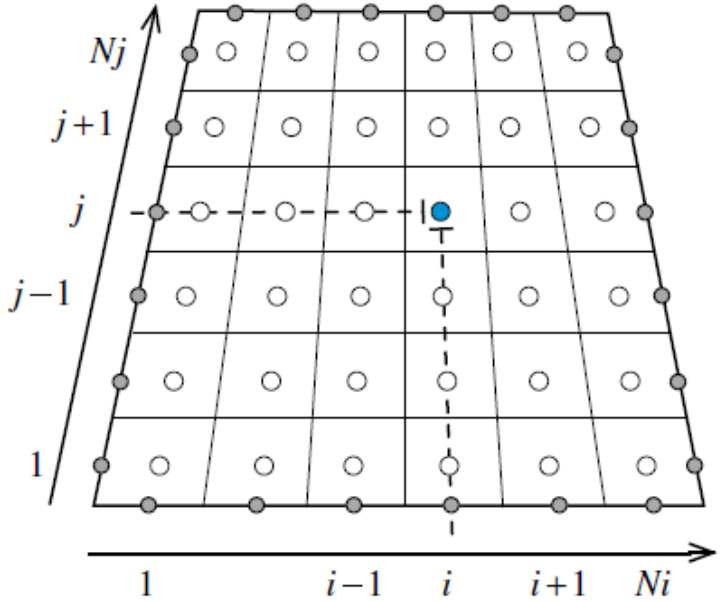


Local to global coordinates.

- In **structured (regular) grid** there is a direct connection between local and glovbal coordinates.
- In a three-dimensional space:

$$n = i + (j-1)N_i + (k-1)N_i * N_j \qquad 1 \leq i \leq N_i \quad 1 \leq j \leq N_j \quad 1 \leq k \leq N_k$$

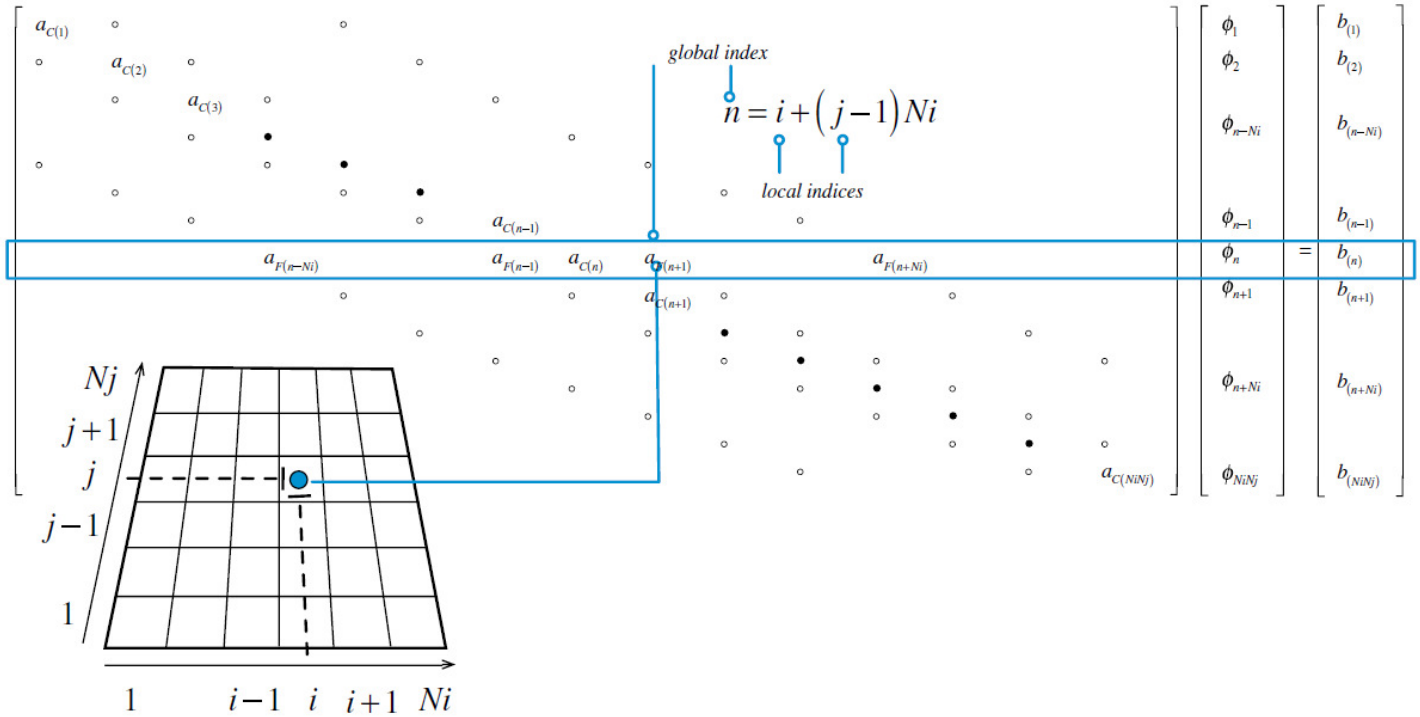
Structured (regular)



In a **structured grid**, one can associate with each computational cell an ordered set of indices (i, j, k), where:

- each index varies over a fixed range, independently of the values of the other indices
- neighboring cells have associated indices that differ by plus or minus one.

Structured (regular)

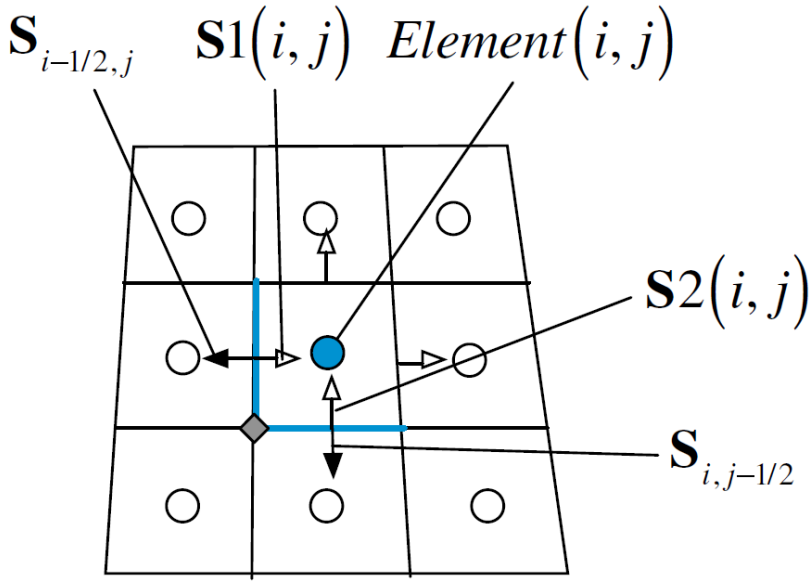


Local to global coordinates.

- In **structured (regular) grid** there is a direct connection between local and glovbal coordinates.
- In a three-dimensional space:

$$n = i + (j-1)N_i + (k-1)N_i * N_j \qquad 1 \leq i \leq N_i \quad 1 \leq j \leq N_j \quad 1 \leq k \leq N_k$$

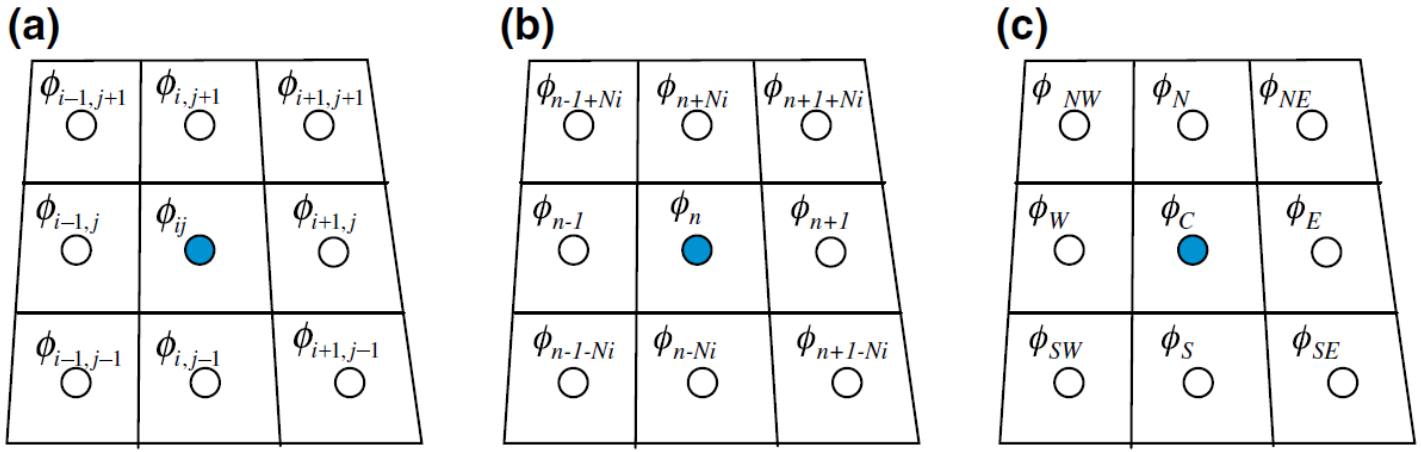
Geometric information



- The element field in a structured two or three dimensional mesh is also defined as an array of size [Nx][Ny][Nz] respectively.
- Thus accessing the element value and its neighbors is equally simple. The element field of a multi-dimensional system may also be defined using a one dimensional array, which in two dimensions will be of size $[N_i * N_j]$ and in three dimensions of size $[N_i * N_j * N_k]$.

Accessing the local geometric information around an element is quite simple.

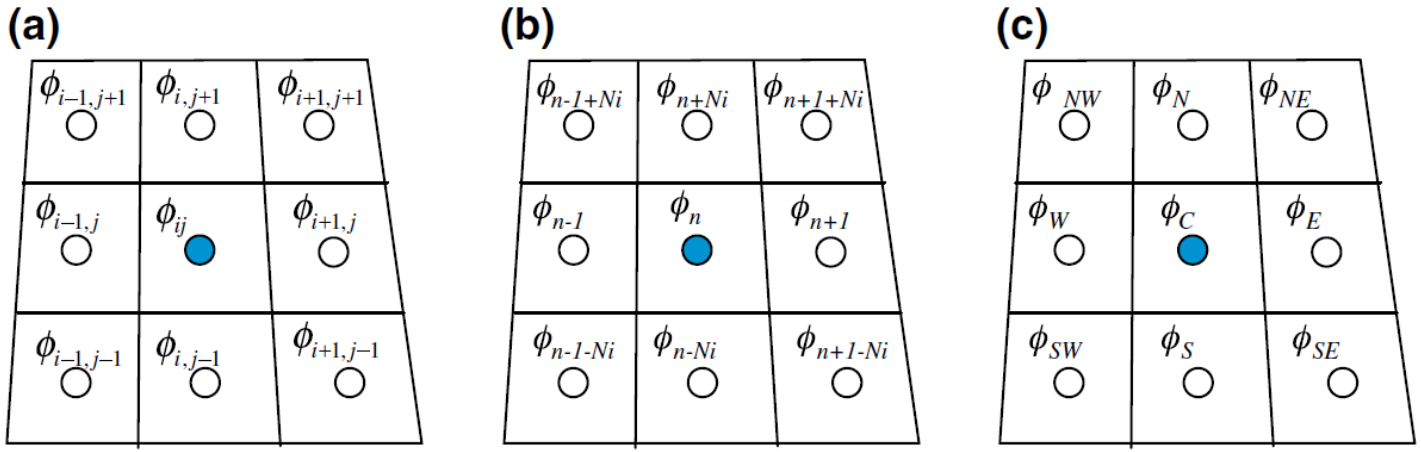
Accessing the element field



Accessing the field in a **STRUCTURED GRID** is as simple as using the indices of the element.

- $\phi(i, j)$ is the value of field ϕ at element (i, j);
- values of ϕ at the neighboring cells to (i, j) are, respectively, $\phi_{i+1,j}$, $\phi_{i-1,j}$, $\phi_{i,j-1}$, , $\phi_{i,j+1}$;

Accessing the element field



Computing the gradient requires calculating the value of ϕ at each face of the finite volume. In 2-D:

$$\overline{\nabla\phi}_n = \frac{1}{V_n}\left(\overline{\phi}_{n+1/2}\boldsymbol{S}_{n+1/2} + \overline{\phi}_{n-1/2}\boldsymbol{S}_{n-1/2} + \overline{\phi}_{n+N_i}\boldsymbol{S}_{n+N_i} + \overline{\phi}_{n-N_i}\boldsymbol{S}_{n-N_i}\right)$$

or, if the **discretization indexing** is used:

$$\overline{\nabla\phi}_C = \frac{1}{V_C}\left(\overline{\phi}_e\boldsymbol{S}_e + \overline{\phi}_w\boldsymbol{S}_w + \overline{\phi}_n\boldsymbol{S}_n + \overline{\phi}_s\boldsymbol{S}_s\right)$$

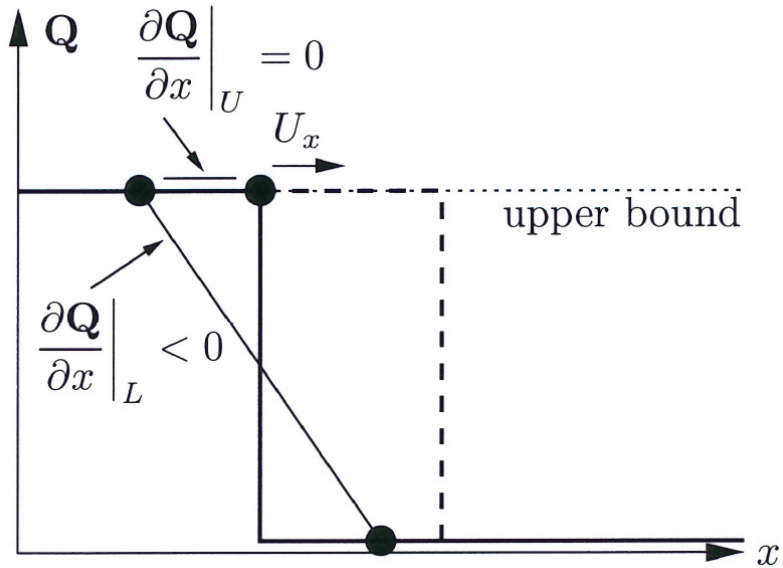
OpenFOAM: numerics best practice

Numerics best practice in OpenFOAM

First, **ALWAYS check mesh non-ortogonality!** → see OpenFOAM's utility `checkMesh`

```
Mesh stats
    points:           651861
    internal points:  596895
    faces:            1877134
    internal faces:   1822278
    cells:            612697
    faces per cell:   6.0379143
    boundary patches: 18
    point zones:      0
    face zones:       0
    cell zones:       0

    ...


Checking geometry...
    Overall domain bounding box (-0.046005353 -0.046019692 -0.0955) (0.045988153 0.045960254 0.008

    ...
    ...

    Mesh non-orthogonality Max: 73.573145 average: 15.825303
   *Number of severely non-orthogonal (> 70 degrees) faces: 83.
    Non-orthogonality check OK.
  <<Writing 83 non-orthogonal faces to set nonOrthoFaces
    Face pyramids OK.
    Max skewness = 3.4441983 OK.
    Coupled point location match (average 0) OK.
```

The need for upwind

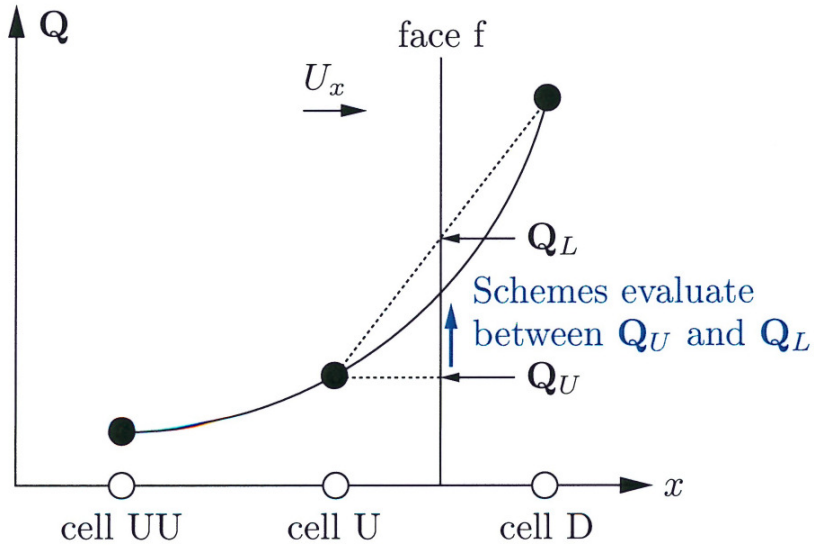


Consider advection of a discontinuity $\boldsymbol{Q}$ in 1D:

$$\frac{\partial \boldsymbol{Q}}{\partial t} = -U_x \frac{\partial \boldsymbol{Q}}{\partial x} \quad \text{equivalent to} \frac{\partial \boldsymbol{Q}}{\partial t} + \boldsymbol{U} \cdot \nabla \boldsymbol{Q} = 0$$

- Upwind (U) differencing maintains **Q** within bounds since $\frac{\partial \boldsymbol{Q}}{\partial x} = 0$
- Down-winding with linear differencing can be unbounded

Improving accuracy



- Convection schemes operates between extremes of
  - ... `upwind`: bounded, most stable
  - ... `linear`: (potentially) most accurate
  - convection schemes broadly evaluate between these extremes
  - ... and limit for boundedness

Good practice

User can check the commonly used choices by typing:

```
find $FOAM_TUTORIALS -name fvSchemes | xargs grep -h "div(phi,U" | sort -u
```

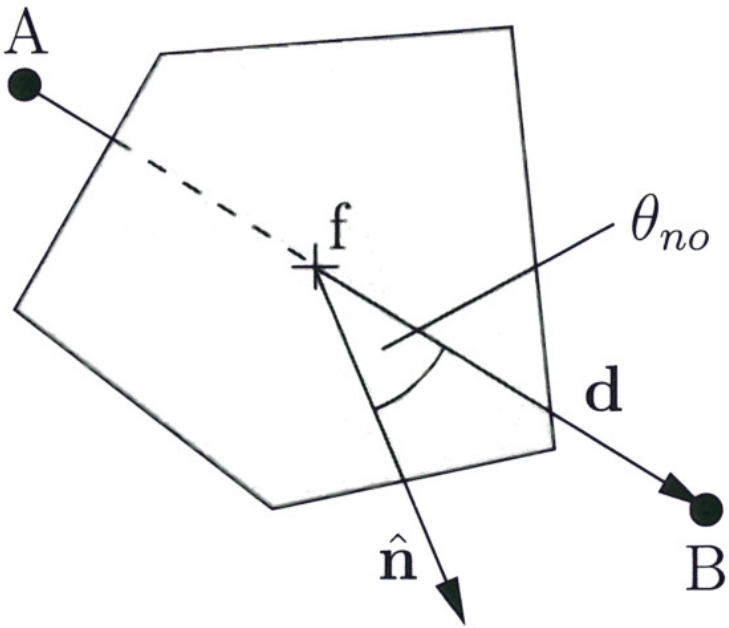
Momentum `div(phi,U)`	Bounded variables e.g. `div(phi,k)`
`linear`	`linear`
`LUST`	`vanLeer`
`linearUpwind`	`limitedLinear`
`upwind`	`upwind`

- Example syntax:

```
div(div(phi,U)) Gauss upwind;
div(div(phi,U)) Gauss linear;
div(phi,U)      Gauss limitedLinearV 1;
div(phi,U)      Gauss linearUpwind grad(U);
div(phi,U)      Gauss LUST grad(U);
```

- `LUST` and `linearUpwind` use discretised `grad(U)`
- `limitedlinear` has a coeff controlling transition from `upwind` → `linear`
- ... Coeff = 1 “pure” `limitedLinear`
- ... Coeff < 1 (typically 0.2): more rapid transition to `linear`

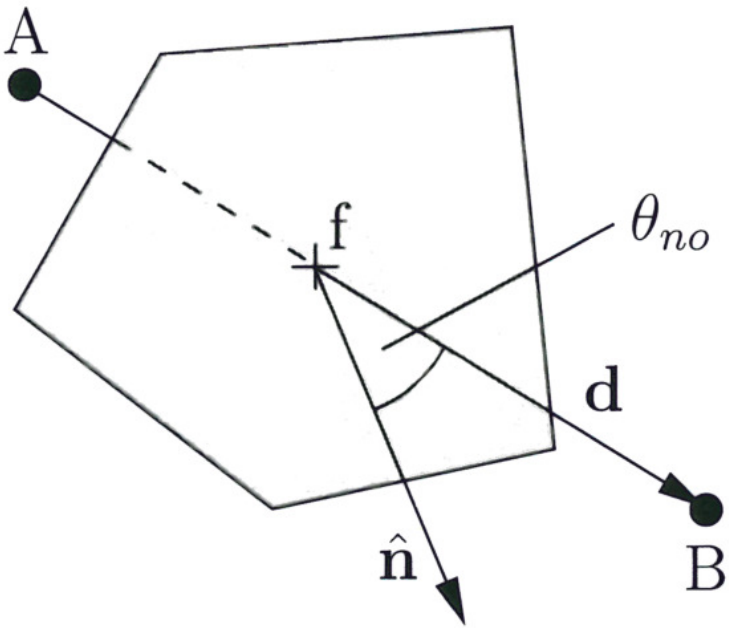
Non-orthogonality



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

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

Non-orthogonality

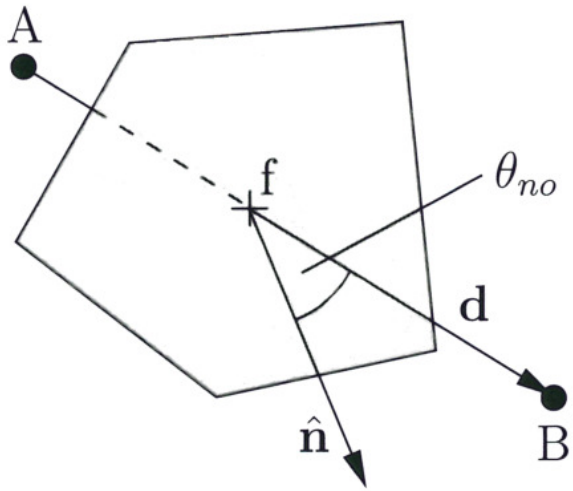


- Angle between $\boldsymbol{d}(\bar{AB})$ and $\hat{\boldsymbol{n}}_f$
- $\hat{\boldsymbol{n}}_f$: unit vector normal to face
- d: vector between cell centers A and B
- θ_{no}: angle between $\vec{d}$ and $\vec{n}$

Why non-orthogonality is important?

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

Non-orthogonality



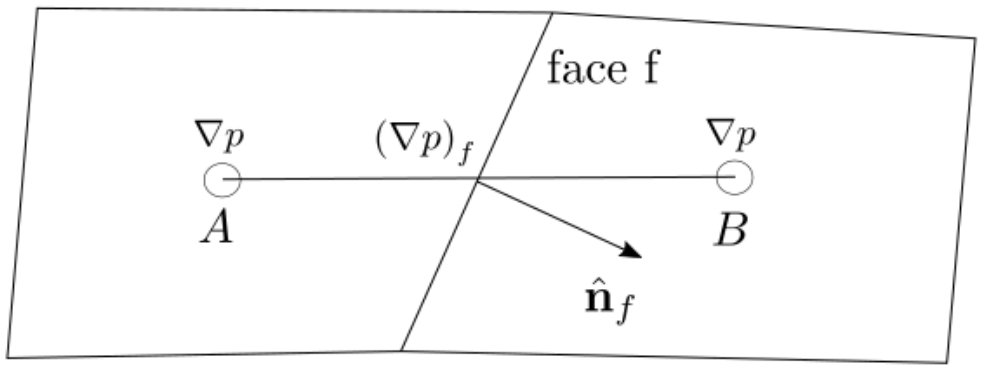
Non-orthogonality affects Laplacian (diffusion) terms (e.g. $\nabla \cdot \nabla p$):

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

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

Example : `Gauss` `linear` `corrected`

Full gradient at face



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

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

The dictionary `fvSchemes`

The file `system/fvSchemes` is the dictionary where the discretization options are set. For example, see:

```
laplacianSchemes
{
    default     Gauss linear corrected;
}


interpolationSchemes
{
    default     linear;
}


snGradSchemes
{
    default     orthogonal;
}
```

Non-orthogonality

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

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

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

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

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

Non-orthogonality

As a result, one gets:

$$D_f \simeq \underbrace{\Gamma_\phi \frac{\hat{\boldsymbol{n}}_f \cdot \hat{\boldsymbol{n}}_f}{\hat{\boldsymbol{d}} \cdot \hat{\boldsymbol{d}}} (\phi_N - \phi_P)\, S}_{\text{implicit}} + \underbrace{\Gamma_\phi \left[(\nabla\phi)_f \cdot \hat{\boldsymbol{n}}_f - \frac{\hat{\boldsymbol{d}} \cdot \hat{\boldsymbol{n}}_f}{\hat{\boldsymbol{d}} \cdot \hat{\boldsymbol{d}}} (\nabla\phi)_f \cdot \hat{\boldsymbol{d}} \right] S}_{\text{explicit}}$$

In OpenFOAM, the relation by Jasak [1] is used, where a geometric interpretation to the approximation of the diffusive term is given:

$$D_f \simeq \Gamma_\phi (\nabla\phi)^*_f \cdot \hat{\boldsymbol{n}}_f\, S = \Gamma_\phi \left[\alpha \frac{\phi_B - \phi_A}{|\hat{\boldsymbol{d}}|} + (\hat{\boldsymbol{n}}_f - \alpha \hat{\boldsymbol{d}}) \cdot (\nabla\phi)_f \right] S$$

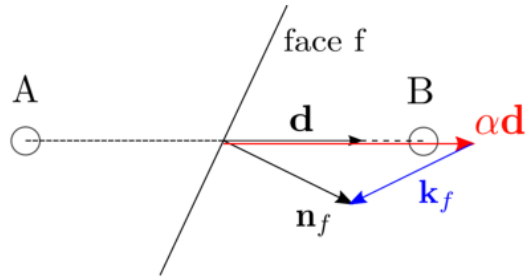
where vectors Δ_f and $\hat{\boldsymbol{k}}_f$ satisfy the condition:

$$\hat{\boldsymbol{k}}_f = -(\hat{\boldsymbol{n}}_f - \alpha \hat{\boldsymbol{d}})$$

[1] H. Jasak. "*Error Analysis and Estimation for the Finite Volume Method with Applications to Fluid Flows*". PhD thesis, Imperial College of London (UK), 1996.

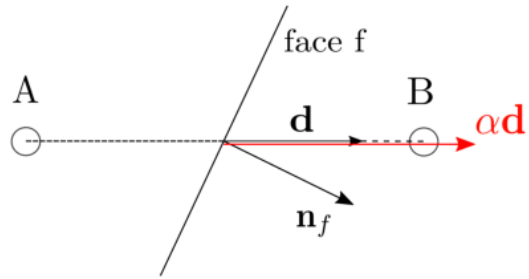
Non-orthogonality

corrected



$$\underbrace{\alpha \frac{p_B - p_A}{|\mathbf{d}|}}_{\text{implicit}} + \underbrace{(\vec{n}_f - \alpha \mathbf{d}) \cdot (\nabla p)_f}_{\text{explicit}}$$

uncorrected



$$\alpha \frac{p_B - p_A}{|\mathbf{d}|}$$

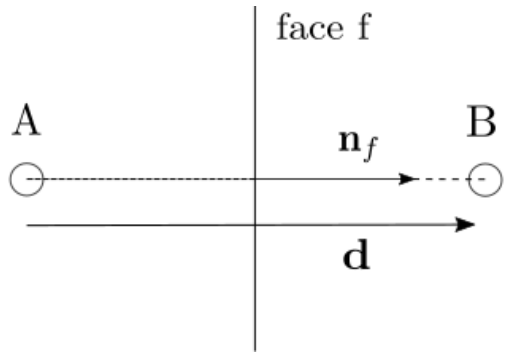
The formulation of Δ_f is depending on the level of approximation used for the diffusion term D_f, as shown in the figure.

Three different corrections are proposed for the diffusion term, depending on the grid:

- `orthogonal` correction
- `minimum` correction
- `overrelaxed approach`

Discretization of `snGrad` in OpenFOAM

orthogonal



$$\frac{p_B - p_A}{|\mathbf{d}|}$$

Example : `Gauss` `orthogonal`

On orthogonal grids the gradient in the direction normal to the interface yields:

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

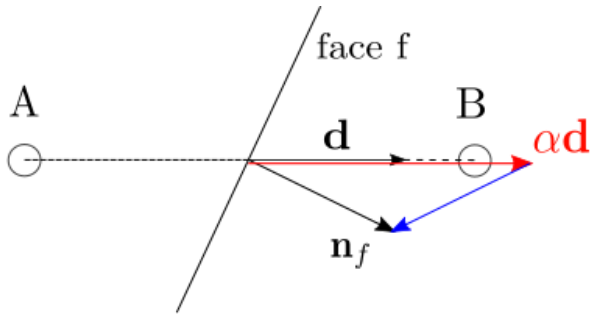
because $\hat{\boldsymbol{d}}$ and $\hat{\boldsymbol{n}}_f$ are aligned, on non-orthogonal grids, the gradient direction that yields an expression involving ϕ_A and ϕ_B will have to be along the line joining the two points A and B.

If $\hat{\boldsymbol{d}}$ represents the unit vector along the direction defined by the line connecting nodes A and B then:

$$\hat{\boldsymbol{d}} = \frac{\boldsymbol{r}_B - \boldsymbol{r}_A}{||\boldsymbol{r}_B - \boldsymbol{r}_A||} = \frac{\boldsymbol{d_{AB}}}{d_{CF}}$$

Discretization of `snGrad`

corrected



$$\underbrace{\alpha \frac{p_B - p_A}{|\mathbf{d}|}}_{\text{implicit}} + \underbrace{(\vec{n}_f - \alpha \mathbf{d}) \cdot (\nabla p)_f}_{\text{explicit}}$$

- Stability requires underrelaxation by factor

$$\alpha = 1/\cos\theta_{no}$$

  as θ_{no} increases, non-orthogonal correction increases, stability decreases;

- Correction can be limited: in the `corrected` scheme a coefficient limits the non-orthogonal correction according to

$$\phi = \begin{cases} 0.33 & \text{explicit corr.} \leq 0.5 \cdot \text{implicit} \\ 0.5 & \text{explicit corr.} \leq \text{implicit} \\ 1 & \text{explicit corr.} \leq 2 \cdot \text{implicit} \end{cases}$$

Limiting the `snGrad`

Suggested non-orthogonality settings:

- $\theta_{no} > 0°$: orthogonal
- $0° < \theta_{no} < 0°$: uncorrected (possibly)
- $5° < \theta_{no} < 60°$: `corrected`
- $\theta_{no} > 60°$: `limited corrected 0.33`
- $\theta_{no} > 80°$: stability very difficult to attain

Limiting the `snGrad`

Example:

```
laplacianSchemes
{
  default    Gauss linear limited corrected 0.33;
}
```

Limiting occurs only at highly non-orthogonal faces but decreases accuracy!

Mesh support in OpenFOAM

- At the core of OpenFOAM there is a set of libraries, implemented as object classes that allow the programmer to manipulate meshes, geometries, and discretization techniques at a high level of coding. These classes represent the basic bricks for the development of OpenFOAM based applications and utilities.

- In OpenFOAM, discretizing operators return a system of equations including a LHS matrix and a RHS source that represent the discretization of the convection operator.

- These LHS matrices and RHS vectors are generated for each of the operators and then added or subtracted as needed to yield the final system of equations that represent the discretized set of algebraic equations defined over each element of the computational domain.

Mesh support in OpenFOAM

- The namespaces `fvm::` and `fvc::` allow for the evaluation of a variety of operators *implicitly* and *explicitly*, respectively.

- The explicit operator `fvc::`, named “finite volume calculus”, returns an equivalent field based on the actual field values.

- The `fvm::` implicit operator, instead, defines the implicit finite volume discretization in terms of matrices of coefficients.

- The role of the `fvm::` and `fvc::` operators is to construct the LHS and RHS of a system of equations in the discretized form over each element in the mesh.

- The discretization process yields a system of equations that can be represented in the matrix form. Each row of the system represents the discretized equation in one element, with the off-diagonal coefficients representing the mutual influence of the neighboring elements.

Conservation Properties

The Navier-Stokes equations have the property that

- the **momentum** in any control volume (microscopic or macroscopic) is changed only by:
  - flow through the surface
  - forces acting on the surface
  - volumetric body forces
- Overall **mass conservation** follows in the same way from the continuity equation.
- **Energy conservation is a more complex issue**:
  - In incompressible isothermal flows, the only energy of significance is kinetic energy.
    - When heat transfer is important, the kinetic energy is generally small compared to the thermal energy so the equation introduced to account for energy transport is a conservation equation for thermal energy.
    - As long as the temperature dependence of the fluid properties is not significant, the thermal energy equation can be solved after solution of the momentum equations is complete: the coupling is then entirely one-way and an equation for the kinetic energy can be derived by taking the scalar product of the momentum equation with the velocity. This is equivalent to write the transport of a passive scalar.
  - In **compressible flow there is a separate conservation equation for the total energy**. In this case, the thoery becomes much more complicated and it will be discussed in the further sections.

The CFL condition

The Courant–Friedrichs–Lewy or CFL condition is a condition for the stability of unstable numerical methods that model convection or wave phenomena. As such, it plays an important role in CFD (computational fluid dynamics).
The section below confers the numerical discussion that derives the CFL condition. After this discussion, the CFL condition is presented, and in later sections, its practical consequences are examined.

Definition of the CFL Condition

Culbert B. Laney's definition of the CFL condition, from the book "Computational Gasdynamics", is as follows: the full numerical domain of dependence must contain the physical domain of dependence.
Therefore, the CFL condition expresses that the distance that any information travels during the timestep length within the mesh must be lower than the distance between mesh elements. In other words, information from a given cell or mesh element must propagate only to its immediate neighbors.
The CFL condition is commonly confused with linear stability conditions or nonlinear stability conditions. However, it's important to emphasize that it is not a sufficient condition for stability, and other stability conditions are generally more restrictive than the CFL condition.

The CFL condition

https://www.simscale.com/blog/2017/08/cfl-condition/

CFL: reference paper from 1928...

https://www.digizeitschriften.de/dms/img/?PID=GDZPPN002272636

What's next....

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(s) from which they are derived. In the non-linear case, the discretized equations must be solved by an iterative technique that involves guessing a solution, linearizing the equations about that solution, and improving the solution; the process is repeated until a converged result is obtained. So, whether the equations are linear or not, efficient methods for solving linear systems of algebraic equations are needed.

Thank you for your attention!

contact: federico.piscaglia@polimi.it