Chapter 11: Unstructured Grids

Ibrahim Sezai
Department of Mechanical Engineering
Eastern Mediterranean University
Fall 2012-2013

Introduction

- When the solution domain is not rectangular Cartesian grids cannot be used.
- When the boundary does not coincide with the co-ordinate lines we can use stepwise approximation near the boundary.

Such an approximation is tedious and time consuming.
- Introduces errors in the computation of wall shear stresses and heat transfer.
- Cells in the solid cylinder do not take part in the calculations \(\rightarrow\) waste of computer storage.
Body fitted grids

- CFD methods for complex geometries:
  1. Structured curvilinear grids (body fitted grids)
  2. Unstructured grids

- Structured curvilinear grids are based on mapping of the flow domain onto a computational domain.

- It is very difficult to find viable mappings when the geometry is complex.

- In such cases the domain may be sub-divided into several sub-regions or blocks. → Block structured grids.

- For the most complex geometries unstructured grids are used.

- Most commercial packages use unstructured grids.

Body fitted grids

An example of an orthogonal curvilinear mesh for calculating flow around an aerofoil

Use of a non-orthogonal body-fitted grid arrangement for the prediction of flow over a half-cylinder
Cartesian vs. curvilinear grids

Flow over a heat exchanger tube bank (only a part shown) →

Body fitted grids give better results with no waste of grids

(a) Cartesian grid using an approximated profile to represent cylindrical surfaces; (b) predicted flow pattern using a 40 X 15 Cartesian grid

(a) Non-orthogonal body-fitted grid for the same problem; (b) predicted flow pattern using a 40 X 15 structured body-fitted grid

Curvilinear grids - Difficulties

The governing equations in curvilinear co-ordinate system are much more complex. The equations in 2-D are:

Continuity
\[
\frac{\partial}{\partial \xi} \left( \rho U \right) + \frac{\partial}{\partial \eta} \left( \rho V \right) = 0
\]  

x-momentum
\[
\frac{\partial}{\partial \xi} \left( \rho U u_{\xi} \right) + \frac{\partial}{\partial \eta} \left( \rho U u_{\eta} \right) + \frac{\partial}{\partial \xi} \left[ \mu \left( \frac{\partial u_{\xi}}{\partial \xi} + \frac{\partial u_{\eta}}{\partial \eta} \right) \right] + \frac{\partial}{\partial \eta} \left[ \mu \left( \frac{\partial u_{\eta}}{\partial \xi} + \frac{\partial u_{\xi}}{\partial \eta} \right) \right] = - \frac{\partial p}{\partial \xi} + \frac{\partial \tau_{\xi \xi}}{\partial \xi} + \frac{\partial \tau_{\eta \xi}}{\partial \eta} + \frac{\partial \tau_{\xi \eta}}{\partial \xi} + \frac{\partial \tau_{\eta \eta}}{\partial \eta} + \rho \frac{\partial^2 U}{\partial \xi^2} + \rho \frac{\partial^2 U}{\partial \eta^2} + \frac{\partial S_{\xi}}{\partial \xi} + \frac{\partial S_{\eta}}{\partial \eta}
\]

(11.1)

y-momentum
\[
\frac{\partial}{\partial \xi} \left( \rho V u_{\xi} \right) + \frac{\partial}{\partial \eta} \left( \rho V u_{\eta} \right) + \frac{\partial}{\partial \xi} \left[ \mu \left( \frac{\partial u_{\xi}}{\partial \xi} + \frac{\partial u_{\eta}}{\partial \eta} \right) \right] + \frac{\partial}{\partial \eta} \left[ \mu \left( \frac{\partial u_{\eta}}{\partial \xi} + \frac{\partial u_{\xi}}{\partial \eta} \right) \right] = - \frac{\partial p}{\partial \eta} + \frac{\partial \tau_{\xi \xi}}{\partial \xi} + \frac{\partial \tau_{\eta \xi}}{\partial \eta} + \frac{\partial \tau_{\xi \eta}}{\partial \xi} + \frac{\partial \tau_{\eta \eta}}{\partial \eta} + \rho \frac{\partial^2 V}{\partial \xi^2} + \rho \frac{\partial^2 V}{\partial \eta^2} + \frac{\partial S_{\xi}}{\partial \xi} + \frac{\partial S_{\eta}}{\partial \eta}
\]

(11.2)

\[
\frac{\partial}{\partial \xi} \left( \rho \frac{\partial u_{\xi}}{\partial \xi} \right) + \frac{\partial}{\partial \eta} \left( \rho \frac{\partial u_{\eta}}{\partial \eta} \right) = - \frac{\partial p}{\partial \xi} + \frac{\partial \tau_{\xi \xi}}{\partial \xi} + \frac{\partial \tau_{\eta \xi}}{\partial \eta} + \frac{\partial \tau_{\xi \eta}}{\partial \xi} + \frac{\partial \tau_{\eta \eta}}{\partial \eta} + \rho \frac{\partial^2 u_{\xi}}{\partial \xi^2} + \rho \frac{\partial^2 u_{\xi}}{\partial \eta^2} + \frac{\partial S_{\xi}}{\partial \xi} + \frac{\partial S_{\eta}}{\partial \eta}
\]

(11.3)

Metric coefficients
\[
\begin{align*}
\frac{g^{12}}{J^2} & = \frac{(x_{\xi}^2 + x_{\eta}^2)}{J^2} \\
\frac{g^{11}}{J^2} & = \frac{(x_{\xi}^2 + x_{\eta}^2)}{J^2} \\
\frac{g^{22}}{J^2} & = \frac{(x_{\xi}^2 + x_{\eta}^2)}{J^2}
\end{align*}
\]

Jacobian of transformation
\[
J = \frac{x_{\xi} x_{\eta} - x_{\eta} x_{\xi}}{J^2}
\]

Contravariant velocities
\[
U = y_{\xi} u - x_{\eta} v \\
V = x_{\xi} u - y_{\eta} v
\]
Curvilinear grids - difficulties

- Body-fitted grids are structured, so grid refinement is generally not local.

- The refinement needed to resolve the boundary layers persists in the interior mesh, which represents a waste of computer storage.

---

Curvilinear grids - difficulties

- To generate curvilinear meshes it is necessary to map the physical geometry into a computational geometry.

  Mapping of physical geometry to computational geometry in structured meshes.

Mapping may be a tedious task if the domain is more complex.
Curvilinear grids - difficulties

- For complex geometries structured grid generation may be very difficult resulting in skewed cells, which can lead to stability problems for CFD solvers.

A structured non-orthogonal mesh for a pent-roof i.c. engine geometry

Problems encountered with structured grids:

- Still difficult and time consuming to generate
- If the geometry cannot be readily mapped into a rectangle (in 2D) or parallelepiped (3D) this can result in skewed grid lines.
- Unnecessary grid resolutions can result when mapping is difficult
- Mapping is sometimes impossible with complex 3D geometries with internal objects/parts.
Block-structured grids

- In block-structured grids the domain is sub-divided into regions, each of which has a structured mesh.
- Such meshes are more flexible than (single block) structured meshes.
- The block structured approach allows the use of fine grids in regions where greater resolution is required.

Block-structured mesh arrangement for an engine geometry, including inlet and exhaust ports, used in engine simulations with KIVA-3V
Unstructured grids

- There is no implicit structure of co-ordinate lines in unstructured grids.
- The mesh can easily be concentrated where necessary without wasting computer storage.
- Control volumes can be of any shape.

Unstructured grids

- A mixture of different cell shapes can be used.
- Unstructured mesh generation is fairly straightforward.
- Complex domains can be meshed.
- Mesh refinement and adaption (in regions with high gradients) are much more easier.
Discretisation in unstructured grids

There are two ways of defining CV’s in unstructured grids:

1. Cell-centered control volumes
2. Vertex-centered control volumes.

We will use cell-centered method.

(a) cell-centred control volumes  (b) vertex-centred control volumes

Discretisation in unstructured grids

The general transport equation (2.39) is

$$\frac{\partial (\rho \phi)}{\partial t} + \text{div}(\rho \phi \mathbf{v}) = \text{div}(\Gamma \text{ grad } \phi) + s_\phi$$  \hspace{1cm} (11.4)

Integrating Eq. (11.4) over a 3-D control volume

$$\int_{CV} \frac{\partial (\rho \phi)}{\partial t} dV + \int_{CV} \text{div}(\rho \phi \mathbf{v})dV = \int_{CV} \text{div}(\Gamma \text{ grad } \phi)dV + \int_{CV} s_\phi dV$$  \hspace{1cm} (11.5)

For a vector $\mathbf{a}$ Gauss’ divergence theorem states

$$\int_{CV} \text{div} \mathbf{a} dV = \int \mathbf{n} \cdot \mathbf{a} dA$$  \hspace{1cm} (11.6)

$n.a \rightarrow$ component of vector $\mathbf{a}$ in the direction of the outward unit vector $\mathbf{n}$ normal to infinitesimal surface element $dA$. 
Discretisation in unstructured grids

Application of Gauss’ divergence theorem to equation (11.1) gives

$$\frac{\partial}{\partial t} \left( \int_{cv} \rho \phi dV \right) + \int_A (\rho \phi \mathbf{v}) \cdot d\mathbf{A} = \int_A \mathbf{n} \cdot (\Gamma \ \text{grad} \ \phi) \cdot d\mathbf{A} + \int_{CV} s_\phi dV$$  \hspace{1cm} (11.7)

Using $\mathbf{A} = \mathbf{An}$, where $\mathbf{A} = A_i \mathbf{i} + A_j \mathbf{j} + A_k \mathbf{k}$, $A = |\mathbf{A}| = (A_x^2 + A_y^2 + A_z^2)^{1/2}$

$\mathbf{n} = n_x \mathbf{i} + n_y \mathbf{j} + n_z \mathbf{k}$ (unit vector normal to surface)

$\mathbf{A}$ = surface area vector

$A$ = magnitude of the surface area

$$\frac{\partial}{\partial t} \left( \int_{cv} \rho \phi dV \right) + \oint_A (\rho \phi \mathbf{v}) \cdot d\mathbf{A} = \oint_A (\Gamma \ \text{grad} \ \phi) \cdot d\mathbf{A} + \int_{CV} s_\phi dV$$  \hspace{1cm} (11.8)

Let us discretize each term separately

### Transient term

Using first order time derivative,

$$\frac{\partial}{\partial t} \left( \int_{cv} \rho \phi dV \right) \approx \frac{(\rho \phi')_p - (\rho \phi')_{p-1}}{\Delta t}$$

Superscript $n-1$ refers to previous time step

Terms with no superscript refer to present time.

Cell volume $V_p$ can be calculated by applying Gauss theorem on the identity $1 = \text{div}(\mathbf{x})$:

$$V = \int_v dV = \int_v \text{div}(\mathbf{x}) dV = \int_A x_i \cdot \mathbf{n} dA \approx \sum_f x_f A_f^x$$  \hspace{1cm} (11.9)

$f \rightarrow$ refers to cell faces. For the cell of node $P$ the summation is over the faces $f_1, f_2, f_3$ and $f_4$.

$A_f^x$ = $x$-component of the cell face surface vector.

$x_f$ = $x$-component of the position vector of the center of face $f$
Volumes and surface areas

\[ \mathbf{A}_f = A_f \mathbf{n} = A_{f}^i \mathbf{i} + A_{f}^j \mathbf{j} + A_{f}^k \mathbf{k} \]  

(11.10)

Instead of \( \mathbf{x}_i \), one can also use \( \mathbf{y}_j \) or \( \mathbf{z}_k \), that is:

\[ V \approx \sum_f y_f A_{f}^y \approx \sum_f z_f A_{f}^z \]

(11.11)

Surface areas

Consider the cell of node P. Let the top surface be \( f_1 \) with vertices \( v_1, v_2, v_3, v_4, v_5 \).

**Calculation of area of surface \( f_1 \):**

Decompose the surface \( f_1 \) into triangles with one common vertex. For example choose common vertex to be \( v_1 \).

The surface normal vector of the whole cell face \( f_1 \) is the sum of the surface vectors of all triangles.

\[ \mathbf{A}_1 = \frac{1}{2} \sum_{i=3}^{N_v} \left[ (\mathbf{r}_{i+1} - \mathbf{r}_i) \times (\mathbf{r}_i - \mathbf{r}_i) \right] \]

(11.12)

\( N_v \) = number of vertices in the cell face (5 vertices for face \( f_1 \))

\( \mathbf{r}_i \) = position vector of vertex \( i \)

Note that there are \((N_v - 2)\) triangles. Also note that the numbering of the vertices is **counter-clockwise** when looking from outside the cell.
The cell face center can be found by averaging the coordinates of the vertices of the face.

\[
(x_c)_f = \frac{1}{N_v} \sum_{i=1}^{N_v} x_i \\
(y_c)_f = \frac{1}{N_v} \sum_{i=1}^{N_v} y_i \\
(z_c)_f = \frac{1}{N_v} \sum_{i=1}^{N_v} z_i
\]

(11.13) (11.14) (11.15)

**Diffusion term**

The diffusion term is approximated as

\[
\oint_A \left( \Gamma \, \text{grad} \phi \right) \cdot d\mathbf{A} \approx \sum_{f=\text{nh}(P)} \Gamma_f \, \nabla \phi_f \cdot \mathbf{A}_f
\]

(11.17)

where the summation is over the faces of a cell. (over the faces \(f_1, f_2, f_4, f_4\) for cell \(P\) in the figure).

Define: \( \mathbf{PN} = r_N - r_P \)  (11.18)

\( \mathbf{r}_P, \mathbf{r}_N \): position vector of centroids of cells \(P\) and \(N\)

\( \mathbf{e}_{PN} \): unit vector along line \(PN\)

\( \mathbf{n} \): unit normal vector of surface \(f\)

\( \mathbf{A}_f \): surface area vector

We wish to approximate the diffusive flux \( D_f = \Gamma_f \nabla \phi_f \cdot \mathbf{A}_f \) at the surface.
Diffusion term

The gradient expression should involve centroids P and N of the cells.

By definition
\[ e_{PN} = \frac{PN}{d_{PN}} \quad d_{PN} = |PN| \quad (11.19) \]

The diffusion term at face f can be written as

\[ \Gamma_f \nabla \phi_f \cdot A_f = \Gamma_f \left( \frac{\partial \phi}{\partial n} \right)_f A_f \approx \Gamma_f \left( \frac{\phi_N - \phi_P}{PN \cdot n_f} \right) A_f = \Gamma_f \frac{\phi_N - \phi_P}{d_{PN} e_{PN} \cdot n_f} A_f \quad (11.20) \]

where \( A_f \) is the component of \( \mathbf{A}_f \) in the direction of line PN

\[ \phi_P \approx \phi_f + \nabla \phi_p \cdot PP' \quad (11.21) \]

\[ \phi_N \approx \phi_f + \nabla \phi_n \cdot NN' \quad (11.22) \]

Substituting (11.21) and (11.22) into Eqn. (11.20)

\[ \nabla \phi_f \cdot A_f = \frac{\phi_N - \phi_P}{d_{PN}} A_d + \frac{\nabla \phi_N \cdot NN' - \nabla \phi_P \cdot PP'}{d_{PN}} A_d \quad (11.29) \]

where

\[ A_d = \frac{\mathbf{A}_f \cdot \mathbf{A}_f}{\mathbf{A}_f \cdot \mathbf{PN}} \quad (11.30) \]

Orthogonal term is treated implicitly, whereas the cross diffusion term is treated as a source term in a deferred correction approach.
Calculation of Gradients

Calculation of Gradient at Nodal Points

An expression is required to calculate gradient at points $P$ and $N$ in equation (11.29). One popular method is to use Gauss’ rule as:

$$\nabla \phi = \frac{1}{V} \int \nabla \phi f dV = \frac{1}{V} \oint \nabla \phi dA = \frac{1}{V_P} \sum_{f \in P} \phi_f A_f$$  \hspace{1cm} (11.31)

where $\phi_f$ is calculated from

$$\phi_f = \phi_{f_0} + \nabla \phi_{f_0} \cdot f_f$$  \hspace{1cm} (11.32)

where

$$f_f = (1 - f_P) \phi_P + f_P \nabla \phi_N, \quad f_P = \frac{\mathbf{Pf} \cdot \mathbf{e}_{PN}}{(\mathbf{PN} \cdot \mathbf{PN})^{1/2}}$$

$\phi_{f_0}$ is calculated from

$$\phi_{f_0} = (1 - f_P) \nabla \phi_f + f_P \nabla \phi_N$$

$\mathbf{f} \cdot \mathbf{f} = \mathbf{Pf} - \mathbf{Pf_0} = \mathbf{Pf} - (\mathbf{Pf} \cdot \mathbf{e}_{PN}) \mathbf{e}_{PN}$

An iterative process is required to calculate gradients:

Step 1: Calculate gradient from Eq. (11.31)
Step 2: Calculate $\phi_{f_0}$ from Eq. (11.32)
Step 3: Repeat steps 1 and 2 until convergence. (4-5 repetitions required)

Diffusion source term

Note that in the general transport equation (11.4), only part of the diffusion term appears explicitly. The other part is buried in the source term:

$$\frac{\partial}{\partial t} (\rho \phi) + \text{div} (\rho \phi \mathbf{v}) = \text{div} (\Gamma \text{grad} \phi) + s_\phi$$

For example, in the x-momentum equation (2.32a) the diffusion term is

$$\frac{\partial}{\partial x} \left[ 2\mu \frac{\partial u}{\partial x} + \lambda \text{div} \mathbf{u} \right] + \frac{\partial}{\partial y} \left[ \mu \left( \frac{\partial u}{\partial y} + \frac{\partial v}{\partial x} \right) \right] + \frac{\partial}{\partial z} \left[ \mu \left( \frac{\partial u}{\partial z} + \frac{\partial w}{\partial x} \right) \right] + \frac{\partial}{\partial x} \left( \lambda \text{div} \mathbf{u} \right) \rightarrow \text{div} (\mu \text{grad} u)$$

$$= \text{div} (\mu \text{grad} u) + s_{Mx}$$

For incompressible fluids $\text{div} \mathbf{u} = 0$. If further, $\mu = \text{constant}$, then $s_{Mx} = 0$ (from discussion just after equation (2.34)).
However, sometimes \( \mu \) may be a function of temperature. Moreover, in the momentum equations of turbulence models, \( \mu \) is replaced by \( (\mu + \mu_t) \), where \( \mu_t \) is the turbulent viscosity, which is not constant at all. For those cases, \( S_{Mx} \) will not be zero.

The source term \( S_{Mx} \) can be expressed in vector form as

\[
S_{Mx} = \text{div}(\mu \nabla \vec{u} \cdot \vec{i}) = \text{div}(\mu \nabla \vec{u})
\]

where \( \nabla \vec{x} \) is the x-component of the gradient operator \( \vec{v} = \frac{\partial}{\partial x} \vec{i} + \frac{\partial}{\partial y} \vec{j} + \frac{\partial}{\partial z} \vec{k} \) is the velocity vector.

Integrating over the control volume gives:

\[
S_{Mx}^\text{diff} = \int_{CP} \text{div}(\mu \nabla \vec{u}) dV = \int_S (\mu \nabla \cdot \vec{u}) \cdot n dA \approx \sum_{f=\text{all}(P)} (\mu \nabla \cdot \vec{u})_f A'_f + (\mu \nabla \cdot \vec{v})_f A'_f + (\mu \nabla \cdot \vec{w})_f A'_f
\]

Similarly, the corresponding terms for the y- and z-momentum equations are

\[
S_{My}^\text{diff} = \int_{CP} \text{div}(\mu \nabla \vec{v}) dV = \int_S (\mu \nabla \cdot \vec{v}) \cdot n dA \approx \sum_{f=\text{all}(P)} (\mu \nabla \cdot \vec{v})_f A'_f + (\mu \nabla \cdot \vec{u})_f A'_f + (\mu \nabla \cdot \vec{w})_f A'_f
\]

\[
S_{Mz}^\text{diff} = \int_{CP} \text{div}(\mu \nabla \vec{w}) dV = \int_S (\mu \nabla \cdot \vec{w}) \cdot n dA \approx \sum_{f=\text{all}(P)} (\mu \nabla \cdot \vec{w})_f A'_f + (\mu \nabla \cdot \vec{u})_f A'_f + (\mu \nabla \cdot \vec{v})_f A'_f
\]

The gradients \( \nabla \vec{u}, \nabla \vec{v} \) and \( \nabla \vec{w} \) at the nodal points are found using Gauss rule (eqn.(11.31)) and their face values are found via linear interpolation using eqn. (11.30).

### Convection term

The convection term is approximated as

\[
\oint_d (\rho \vec{v} \cdot \vec{A}) \cdot dA \approx \sum_{f=1}^{n_{Pc}} (\rho \vec{v} \cdot \vec{A})_f \phi_f = \sum_{f=1}^{n_{Pc}} \hat{m}_f \phi_f \quad (11.33)
\]

where \( n_{Pc} \) is the number of faces of cell \( P \), (faces \( f_1, f_2, f_3, f_4 \) for cell \( P \) in the figure).

To calculate \( \hat{m}_f \), \( \vec{v} \), at the cell faces should be found. \( \vec{v} \) is calculated using the momentum interpolation method (MIM) of Rhie and Chow, to be discussed later.

Defining convective flux as

\[
F^c_f = \hat{m}_f \phi_f
\]

The convection term can also be expressed as

\[
\oint_d (\rho \vec{v} \cdot \vec{A}) \cdot dA \approx \sum_{f=1}^{n_{Pc}} F^c_f \quad (11.35)
\]
Calculation of $\phi_f$ for Convection Terms

**a) First order methods**

1) Upwind difference method (UD)

\[ \phi_f = \phi_p \quad \text{for} \quad \dot{m}_f > 0 \]
\[ \phi_f = \phi_N \quad \text{for} \quad \dot{m}_f < 0 \]  \hspace{1cm} (11.36)

The convective flux for the upwind difference method can be written as

\[ F_f^U = \max(\dot{m}_f, 0.)\phi_p + \min(\dot{m}_f, 0.)\phi_N \]  \hspace{1cm} (11.37)

**High Order Methods**

1) Linear Upwind Difference Scheme (LUD), or 2nd order Upwind Difference Scheme (SOU)

LUD formulation along a line in 1-D is:

\[ \phi_f = \phi_f^U + \left( \frac{\phi_p - \phi_U}{\Delta x} \right) \frac{1}{2} \Delta x \]

where subscript $U$ refers to upwind node. This can be extended to 3D space using Taylor series expansion around point $U$:

\[ \phi_f = \phi_f^U + \nabla \phi_U \cdot \mathbf{r}_U \]  \hspace{1cm} (11.38)

where $\phi_f^U = \phi_p$, $\nabla \phi = \nabla \phi_p$, $\mathbf{r}_U = \mathbf{r}_p = \mathbf{r}_f - \mathbf{r}_p$ for $\dot{m}_f > 0$

\[ \phi_f = \phi_N + \nabla \phi_N \cdot \mathbf{r}_N \quad \text{for} \quad \dot{m}_f < 0 \]  \hspace{1cm} (11.39)

In Eq. (11.39) $\nabla \phi_p$ and $\nabla \phi_N$ are calculated using Eq. (11.31)
Method of Deferred Correction

The convection term $F_f = m_f \phi_f$ for high order methods at the cell faces is implemented as the sum of the upwind value and other higher order terms which are evaluated at the previous iteration:

$$F_f = F_f^{(U)} + (F_f^{(H)} - F_f^{(U)})^{old}$$  \hspace{1cm} (11.40)

Subscript $U$ refers to upwind (\(\phi\) calculated using first order upwind method) Subscript $H$ refers to a high order method such as CD or LUD.

Substituting $F_f^{(U)}$ from Eqn. (11.37) into Eqn. (11.40)

$$F_f = \max(m_f, 0.) \phi_f + \min(m_f, 0.) \phi_f + \left[ m_f \phi_f^{(H)} - \max(m_f, 0.) \phi_f - \min(m_f, 0.) \phi_f \right]^{old}$$

Then, convection term in Eqn. (11.35) becomes

$$\sum\limits_{f=1}^{n_f(P)} \left( \max(m_f, 0.) \phi_f + \min(m_f, 0.) \phi_f + \sum\limits_{f=1}^{n_f(P)} \left( m_f \phi_f^{(H)} - \max(m_f, 0.) \phi_f - \min(m_f, 0.) \phi_f \right)^{old} \right)$$

Source term

Source term in Eq. (11.8) is discretized as

$$S_p = \frac{1}{V} \int V p dV = \frac{1}{V} \sum\limits_{j=1}^{n_p} p_j A_j$$

For $x$-momentum equation:

$$S_p = -\frac{\partial p}{\partial x}$$

For $y$-momentum equation:

$$S_p = -\frac{\partial p}{\partial y}$$

where

$$\nabla p = \frac{\partial p}{\partial x} i + \frac{\partial p}{\partial y} j = (\nabla p) x i + (\nabla p) y j$$

Using Gauss’ rule (Eq. (11.31 with (11.32)):

$$\nabla p = \frac{1}{V} \sum\limits_{j=1}^{n_p} p_j \text{A}_j$$

\hspace{1cm} (Conservative form)

$$\nabla p = \frac{1}{V} \left( \sum\limits_{j=1}^{n_p} p_j \text{A}_j \right)^x + \frac{1}{V} \left( \sum\limits_{j=1}^{n_p} p_j \text{A}_j \right)^y$$

\hspace{1cm} (Nonconservative form)

$$\int s_y dV = s_y V_p = -\frac{\partial p}{\partial y} V_p = -\sum\limits_{j=1}^{n_p} p_j A_j^y$$

\hspace{1cm} for $x$-momentum equation

$$\int s_x dV = s_x V_p = -\frac{\partial p}{\partial x} V_p = -\sum\limits_{j=1}^{n_p} p_j A_j^x$$

\hspace{1cm} for $y$-momentum equation

I. Sezai – Eastern Mediterranean University
Substituting the discretized forms of the transient, convection and diffusion terms into the general equation (11.8)

$$\frac{(\rho \phi V)_{P} - (\rho \phi V)_{P-1}}{\Delta t} + \sum_{j=1}^{n(P)} \min (\dot{m}_{j}, 0.) \phi_{P} + \sum_{j=1}^{n(P)} \max (\dot{m}_{j}, 0.) \phi_{P} = \sum_{j=1}^{n(P)} \frac{\Gamma_{j} A_{j} D}{d_{Pn}} (\phi_{n} - \phi_{P})$$

$$+ \sum_{j=1}^{n(P)} \left[ \Gamma_{j} A_{j} (\nabla \phi_{j} \cdot \mathbf{N} N - \nabla \phi_{j} \cdot \mathbf{P P}) \right]_{n} - s_{b} V_{P}$$

$$- \sum_{j=1}^{n(P)} \left[ \dot{m}_{j} \phi_{j}'' - \max (\dot{m}_{j}, 0.) \phi_{P} - \min (\dot{m}_{j}, 0.) \phi_{N} \right]_{n-1}$$

(11.44)

Equation (11.44) can be written in the form of

$$a_{p} \phi_{p} = \sum_{N=n(P)} a_{p} \phi_{N} + S$$

(11.45)

$$S = S^{s} V_{s} + S^{t} V_{t} + S^{d} V_{d} + S^{p}$$ (source in differential equation + transient + deferred correction + pressure) terms

$$a_{s} = \frac{\Gamma_{j} A_{j}}{d_{Pn}} \min [\dot{m}_{j}, 0.]$$

$$a_{t} = \sum_{N=n(P)} a_{t} + a_{t}$$

$$S^{s} = \sum_{P} \left[ \Gamma_{j} A_{j} (\nabla \phi_{j} \cdot \mathbf{N} N - \nabla \phi_{j} \cdot \mathbf{P P}) \right]_{n}$$

$$S^{t} = \sum_{P} \left[ \nabla \phi_{j} \cdot \mathbf{N} N - \nabla \phi_{j} \cdot \mathbf{P P} \right]_{n}$$

$$S^{d} = \sum_{P} \left[ \nabla \phi_{j} \cdot \mathbf{N} N - \nabla \phi_{j} \cdot \mathbf{P P} \right]_{n}$$

$$S^{p} = \sum_{P} \left[ \nabla \phi_{j} \cdot \mathbf{N} N - \nabla \phi_{j} \cdot \mathbf{P P} \right]_{n}$$

$$\left( \frac{\partial \phi}{\partial t} \right)_{x} = \frac{\partial}{\partial x} \left( \rho \phi \frac{\partial \phi}{\partial x} - (\rho \phi V) \frac{\partial \phi}{\partial x} \right)$$

(11.46)

$$\left( \frac{\partial \phi}{\partial t} \right)_{y} = \frac{\partial}{\partial y} \left( \rho \phi \frac{\partial \phi}{\partial y} - (\rho \phi V) \frac{\partial \phi}{\partial y} \right)$$

$$= \sum_{j=1}^{n(P)} \min (\dot{m}_{j}, 0.) \phi_{j} - \max (\dot{m}_{j}, 0.) \phi_{N}$$

Equation 11.46 can be written in the form of

$$a_{t} \phi_{t} = \sum_{N=n(P)} a_{t} \phi_{N} + S$$

(11.45)

$$S = S^{s} V_{s} + S^{t} V_{t} + S^{d} V_{d} + S^{p}$$ (source in differential equation + transient + deferred correction + pressure) terms

$$a_{s} = \frac{\Gamma_{j} A_{j}}{d_{Pn}} \min [\dot{m}_{j}, 0.]$$

$$a_{t} = \sum_{N=n(P)} a_{t} + a_{t}$$

$$S^{s} = \sum_{P} \left[ \Gamma_{j} A_{j} (\nabla \phi_{j} \cdot \mathbf{N} N - \nabla \phi_{j} \cdot \mathbf{P P}) \right]_{n}$$

$$S^{t} = \sum_{P} \left[ \nabla \phi_{j} \cdot \mathbf{N} N - \nabla \phi_{j} \cdot \mathbf{P P} \right]_{n}$$

$$S^{d} = \sum_{P} \left[ \nabla \phi_{j} \cdot \mathbf{N} N - \nabla \phi_{j} \cdot \mathbf{P P} \right]_{n}$$

$$S^{p} = \sum_{P} \left[ \nabla \phi_{j} \cdot \mathbf{N} N - \nabla \phi_{j} \cdot \mathbf{P P} \right]_{n}$$

$$\left( \frac{\partial \phi}{\partial t} \right)_{x} = \frac{\partial}{\partial x} \left( \rho \phi \frac{\partial \phi}{\partial x} - (\rho \phi V) \frac{\partial \phi}{\partial x} \right)$$

(11.46)

$$\left( \frac{\partial \phi}{\partial t} \right)_{y} = \frac{\partial}{\partial y} \left( \rho \phi \frac{\partial \phi}{\partial y} - (\rho \phi V) \frac{\partial \phi}{\partial y} \right)$$

$$= \sum_{j=1}^{n(P)} \min (\dot{m}_{j}, 0.) \phi_{j} - \max (\dot{m}_{j}, 0.) \phi_{N}$$

Equation 11.46 can be written in the form of

$$a_{t} \phi_{t} = \sum_{N=n(P)} a_{t} \phi_{N} + S$$

(11.45)

$$S = S^{s} V_{s} + S^{t} V_{t} + S^{d} V_{d} + S^{p}$$ (source in differential equation + transient + deferred correction + pressure) terms

$$a_{s} = \frac{\Gamma_{j} A_{j}}{d_{Pn}} \min [\dot{m}_{j}, 0.]$$

$$a_{t} = \sum_{N=n(P)} a_{t} + a_{t}$$

$$S^{s} = \sum_{P} \left[ \Gamma_{j} A_{j} (\nabla \phi_{j} \cdot \mathbf{N} N - \nabla \phi_{j} \cdot \mathbf{P P}) \right]_{n}$$

$$S^{t} = \sum_{P} \left[ \nabla \phi_{j} \cdot \mathbf{N} N - \nabla \phi_{j} \cdot \mathbf{P P} \right]_{n}$$

$$S^{d} = \sum_{P} \left[ \nabla \phi_{j} \cdot \mathbf{N} N - \nabla \phi_{j} \cdot \mathbf{P P} \right]_{n}$$

$$S^{p} = \sum_{P} \left[ \nabla \phi_{j} \cdot \mathbf{N} N - \nabla \phi_{j} \cdot \mathbf{P P} \right]_{n}$$
In above equations superscripts:
\( n-1 \) refers to previous iteration values
\( o \) refers to previous time step values

In solving steady flows, Eq. (11.45) is relaxed as
\[
\frac{\alpha}{\alpha} \phi_p = \sum_{N=\text{all}(P)} a_n \phi_{n-1} + S + (1-\alpha) \frac{\alpha}{\alpha} \phi_o^{n-1} \tag{11.47}
\]

\( 0 \leq \alpha \leq 1 \) underrelaxation factor.

In that case, the terms in Eq. (11.46) are redefined as
\[
\begin{align*}
S & \leftarrow S + (1-\alpha) \frac{\alpha}{\alpha} \phi_o^{n-1} \\
\alpha_p & \leftarrow \frac{\alpha}{\alpha} 
\end{align*}
\tag{11.48}
\]
where superscript \( n-1 \) refers to the previous iteration value.

### MIM method

To find \( m_f \) in Eq.(11.34), velocity vector \( \mathbf{v}_f \) at cell faces are calculated using the momentum interpolation method (MIM) originally proposed by Rhie and Chow.

Rewriting the momentum equations for a 2D flow in the form of Eq. (11.45)
\[
\begin{align*}
\frac{\alpha}{\alpha} u_r &= \sum_{N=\text{all}(P)} a^{\alpha}_n u_{n} + a^{\alpha}_n b^{\alpha}_n - V(\nabla p)_r \mathbf{i} \\
\frac{\alpha}{\alpha} v_r &= \sum_{N=\text{all}(P)} a^{\alpha}_n v_{n} + a^{\alpha}_n b^{\alpha}_n - V(\nabla p)_r \mathbf{j}
\end{align*}
\tag{11.49}
\tag{11.50}
\]

where, \( b \) is the source terms excluding pressure source. Above Eqns can be written as,
\[
\begin{align*}
\{u_r\} - \{H[u_r]\} &= \{D[u_r]\} \begin{bmatrix} a^{\alpha} \\ 0 \end{bmatrix} \begin{bmatrix} (\nabla p)_r \mathbf{i} \\ (\nabla p)_r \mathbf{j} \end{bmatrix} \\
\{v_r\} - \{H[v_r]\} &= \{D[v_r]\} \begin{bmatrix} a^{\alpha} \\ 0 \end{bmatrix} \begin{bmatrix} (\nabla p)_r \mathbf{i} \\ (\nabla p)_r \mathbf{j} \end{bmatrix}
\end{align*}
\tag{11.51}
\]

Defining the vector forms of the above operators as
\[
H[\mathbf{v}]_r = \begin{bmatrix} \{H[u_r]\} \\ \{H[v_r]\} \end{bmatrix} \quad \mathbf{D}_r = \begin{bmatrix} \{D[u_r]\} & 0 \\ 0 & \{D[v_r]\} \end{bmatrix} \quad (\nabla p)_r = \begin{bmatrix} (\nabla p)_r \mathbf{i} \\ (\nabla p)_r \mathbf{j} \end{bmatrix}
\tag{11.52}
\tag{11.53}
\]

For point \( P \), the momentum equation in vector form becomes
\[
\mathbf{v}_f = H[\mathbf{v}]_r - \mathbf{D}_r (\nabla p)_r 
\tag{11.54}
\]
Similarly, for point N
\[ v_N = H[v]_N - D_N (\nabla p)_N \]
Eq. (11.54) can be written at the cell face \( f \) as
\[ v_f = H_f - D_f (\nabla p)_f \]  
(11.55)
Assuming that the term \( H_f \) can be approximated by using an interpolation between points P and N, that is
\[ H_f = \overline{H_f} \]
where the overbar indicates linear average between point P and N. Then, Eq. (11.55) becomes
\[ v_f = \overline{H_f} - D_f (\nabla p)_f \]  
(11.56)
The linearly averaged form of Eq. (11.55) is
\[ \overline{H_f} = \overline{v_f} + D_f \nabla p_f \]  
(11.57)
Using,
\[ D_f \nabla p_f = \overline{D_f} \nabla p_f \]
Eq (11.57) becomes,
\[ \overline{H_f} = \overline{v_f} + \overline{D_f} \nabla p_f \]  
(11.58)

Mass flow rate
where
\[ \overline{v_f} = (1-f_p)v_p + f_p v_N \]
\[ \overline{D_f} = (1-f_p)D_p + f_p D_N \]
\[ \nabla p_f = (1-f_p)(\nabla p)_p + f_p (\nabla p)_N \]
\[ \overline{D_f} = \overline{D_f} \]
where, \( f_p \) is the geometric interpolation factor defined as
\[ f_p = \frac{P \cdot e_{PN}}{PN} \]
or preferably:
\[ f_p = \frac{P \cdot e_{PN}}{PN} = \frac{P \cdot e_{PN}}{(PN \cdot PN)^{1/2}} \]
A more accurate interpolated value can be found using Eqn. (11.32).
Substituting eq (11.58) into eq (11.56)
\[ v_f = \overline{v_f} - \left( \overline{D_f} \nabla p_f - \overline{D_f} \nabla p_f \right) \]  
(11.59)
Mass flow rate:
\[ \dot{m}_f = (\rho \nu \cdot A)_f \]
Substituting Eq. (11.59) for \( v_f \):
\[ \dot{m}_f = \rho_f \overline{v_f} \cdot A_f - (\rho_f \overline{D_f} \nabla p_f \cdot A_f - \rho_f \overline{D_f} \nabla p_f \cdot A_f) \]  
(11.60)
The term $\nabla p_f \cdot A_f$ is discretized using Eq. (11.29)

$$
\nabla \phi_f \cdot A_f = \frac{\phi_{n} - \phi_{p} \cdot A_f}{d_{PN}} + \frac{\nabla \phi_{n} \cdot \text{NN}^\top \nabla \phi_{p} \cdot \text{PP}^\top}{d_{PN}} A_f
$$

(11.61)

where the overbar denotes linear interpolation between the two cells straddling the surface $f$.

Substituting into Eq. (11.60)

$$
\dot{m}_f = \rho_f \overline{v_f} \cdot A_f + \rho_f \overline{D_f (\nabla p_f) \cdot A_f} - \rho_f \overline{P_f} \left( \frac{P_n - P_p \cdot A_f}{d_{PN}} + \frac{\nabla P_n \cdot \text{NN}^\top \nabla P_p \cdot \text{PP}^\top}{d_{PN}} A_f \right)
$$

(11.62)

or using $\overline{A_f} = \frac{A_f \cdot A_f}{A_f \cdot PN}$

$$
\dot{m}_f = \rho_f \overline{v_f} \cdot A_f + \rho_f \overline{D_f (\nabla p_f) \cdot A_f} - \rho_f \overline{P_f} \left( p_n - p_p \cdot A_f \right) - \rho_f \overline{\phi_f} \left( \nabla P_n \cdot \text{NN}^\top \nabla P_p \cdot \text{PP}^\top \right) A_f
$$

(11.63)

where $a_f \overline{\phi_f} = \rho_f (\overline{D_f A_f}) \cdot A_f / A_f \cdot PN$ (from Eq. (11.77))

### SIMPLEx Algorithm

Consider the momentum equations in Eq. (11.56) rewritten at the face $f$ as

$$
v_f - \mathbf{H}[v]_f = -\mathbf{D}_f (\nabla p)_f
$$

(11.64)

Let $p', u', v'$ be the correction needed to correct the guessed pressure and velocity fields, i.e.

$$
p = p^* + p'
$$

and

$$
v = v^* + v'
$$

(11.65)

For a guessed pressure field $p^*$ the corresponding velocity can be written as

$$
v'_f - \mathbf{H}[v']_f = -\mathbf{D}_f (\nabla p^*)_f
$$

(11.66)

Subtracting Eq. (11.66) from Eq. (11.64) yields

$$
v'_f - \mathbf{H}[v']_f = -\mathbf{D}_f (\nabla p')_f
$$

(11.67)
SIMPLE Algorithm

In SIMPLE method the second term on the left hand side of Eq. (11.67) is neglected. Then, Eq. (11.67) becomes

$$v'_f = -D_f \nabla p'_f$$  \hspace{1cm} (11.68)

The discretized continuity equation is

$$\left(\frac{\rho_p - \rho'_p}{\Delta} \right) V_p + \sum_{f=\text{in}(P)} \tilde{m}_f = 0 \quad \text{where} \quad \tilde{m}_f = \rho_v v'_f \cdot A_f$$  \hspace{1cm} (11.69)

or

$$\left(\frac{\rho_p - \rho'_p}{\Delta} \right) V_p + \sum_{f=\text{in}(P)} \tilde{m}^*_f + \sum_{f=\text{out}(P)} \tilde{m}'_f = 0$$  \hspace{1cm} (11.70)

where

$$\tilde{m}'_f = (\rho v'_f) \cdot A_f$$

Substituting $v'_f$ from Eq. (11.68)

$$\tilde{m}'_f = (\rho v'_f) \cdot A_f = -\rho_f (D_f \nabla p'_f) \cdot A_f$$  \hspace{1cm} (11.71)

The term $\nabla p'_f \cdot A_f$ is discretized using Eq. (11.29)

$$\nabla p'_f \cdot A_f = \frac{A_f \cdot A_f}{A_f \cdot P N} (p'_N - p'_P) + \frac{A_f \cdot A_f}{A_f \cdot P N} (\nabla p'_N \cdot N N' - \nabla p'_P \cdot P P')$$  \hspace{1cm} (11.72)

Substituting Eq. (11.72) into Eq. (11.71) gives

$$\tilde{m}'_f = -\rho_f (D_f A_f) \cdot A_f (p'_N - p'_P) - \rho_f (D_f A_f) \cdot A_f (\nabla p'_N \cdot N N' - \nabla p'_P \cdot P P')$$  \hspace{1cm} (11.73)

Substituting Eq. (11.73) into Eq. (11.70), the continuity equation becomes

$$\sum_{f=\text{in}(P)} \rho_f (D_f A_f) \cdot A_f (p'_N - p'_P) = \sum_{f=\text{in}(P)} \tilde{m}^*_f + \frac{\rho_p - \rho'_p}{\Delta} V_p$$

$$- \sum_{f=\text{out}(P)} \rho_f (D_f A_f) \cdot A_f (\nabla p'_N \cdot N N' - \nabla p'_P \cdot P P')$$  \hspace{1cm} (11.74)

Equation (11.74) is also called the pressure correction equation.
The pressure correction equation (11.74) can be written as:

\[
a_p^p b_p^p = \sum_{N=nb(P)} a_N^p p_N^p + b_N^p
\]

(11.77)

\[
a_N^p = \frac{\rho_f (D_j A_j \cdot A_f)}{A_j \cdot P_N}
\]

\[
a_P^p = \sum_{N=nb(P)} a_N^p
\]

\[
b_P^p = -\frac{(\rho_P^0 - \rho_P^p)}{\Delta t} V_P - \sum_{f=nb(P)} m_i^f + \sum_{f=nb(P)} \frac{\rho_f (D_j A_j \cdot A_f)}{A_f \cdot P_N} (\nabla p_N^p \cdot NN^p - \nabla p_P^p \cdot PP^p)
\]

The last term on the right hand side of \( b_P \) term can be neglected.

**SIMPLE Algorithm**

Mass correction given by Eqn (11.73) can be written as

\[
\dot{m}_j = -a_N^p (p_N^p - p_P^p) - a_N^p (\nabla p_N^p \cdot NN^p - \nabla p_P^p \cdot PP^p)
\]

(11.79)

After solving the \( p' \) field from Eq. (11.77) the mass flow rate at the surfaces are corrected by using Eq. (11.79)

\[
\dot{m}_j = \dot{m}_j^p - a_N^p (p_N^p - p_P^p) - a_N^p (\nabla p_N^p \cdot NN^p - \nabla p_P^p \cdot PP^p)
\]

(11.80)

The nodal velocities are corrected as

\[
\mathbf{v}_p = \mathbf{v}_P^p + \mathbf{v}_P = \mathbf{v}_P - D_j \nabla p_P^p
\]

(11.81)

and the pressure field is corrected by using

\[
p_P = p_P^p + \alpha_p p_P^p
\]

(11.82)

\( \alpha_p \) = pressure under-relaxation factor
**SIMPLE Algorithm**

- **Step 1:** Solve the discretized transport equation (11.47) for $\phi = u$ and $\phi = v$

$$\frac{d\phi}{\alpha} = \sum_{n=0}^{N-1} a_n \phi + S + (1 - \alpha) \frac{d\phi}{\alpha}$$

- **Step 2:** Calculate mass flow rate at cell faces (Eqn. (11.63))

$$\dot{m}_f = \rho_f \vec{v}_f \cdot \vec{A}_f + \rho_f \frac{\bar{D}_f}{(\nabla p)_f} \cdot \vec{A}_f - a_f^\alpha (p_N - p_P) - a_f^\alpha (\nabla P_N \cdot \vec{N} - \nabla P_P \cdot \vec{P})$$

- **Step 3:** Solve pressure correction equation (11.77)

$$\alpha \frac{d\phi}{p} = \sum_{N=0}^{N-1} a_n^\phi p_n^\phi + b_P^\phi$$

- **Step 4:** Correct velocities and pressure at points P using Eqn’s (11.81), (11.82)

$$v = v^*_p + v^* = v^*_p - D_p \nabla p^*_p \quad p^*_p = p^*_p + \alpha p^*_p$$

- **Step 5:** Correct mass flow rate using equation (11.80)

$$\dot{m}_f = \dot{m}_f^* - a^\phi (p^*_N - p^*_P) - a^\phi (\nabla p^*_N \cdot \vec{N} - \nabla p^*_P \cdot \vec{P})$$

- **Step 6:** Solve the discretized transport equation (11.47) for other unknowns (i.e. for $\phi = \text{Temperature}$)

$$\frac{d\phi}{\alpha} = \sum_{n=0}^{N-1} a_n \phi + S + (1 - \alpha) \frac{d\phi}{\alpha}$$

- **Step 7:** Repeat steps 1 to 6 until convergence.

**Modifications to original MIM method**

In the original MIM method proposed by Rhie and Chow, using equations (11.56-11.57) for the face values $\mathbf{H}_f$ and $\mathbf{D}_f$, makes the solution to depend on the relaxation parameter $\alpha$ as well as the time step size $\Delta t$. To overcome this, Eqns (11.49-11.50) are written as (Bo Yu et al., *Numer. Heat Transf.*, Part B, v42, pp. 141-166, 2002):

$$\mathbf{v}_f = \mathbf{H}_f - \frac{\alpha}{\mathbf{a}_f} V_f (\nabla p)_f + (1 - \alpha) \mathbf{v}_f^{n-1} + \frac{\alpha}{\mathbf{a}_f} \mathbf{a}_f^\phi \mathbf{v}_f^\phi + \frac{\alpha}{\mathbf{a}_f} V_f (\mathbf{s}^{\text{body}})_f$$

where

$$\mathbf{H}_f = \alpha \mathbf{a}_f \left[ \sum_{n=0}^{N-1} \mathbf{a}_n \mathbf{v}_n \right] + (\mathbf{S}^{bc})_f + (\mathbf{S}^{bc})_f$$

Note that the source terms resulting from relaxation, unsteady term and body forces has been separated from $\mathbf{H}$ term. $\mathbf{S}^{bc}$ is the source due to boundary conditions.

The corresponding equation at the face is

$$\mathbf{v}_f = \mathbf{H}_f - \frac{\alpha}{\mathbf{a}_f} V_f (\nabla p)_f + (1 - \alpha) \mathbf{v}_f^{n-1} + \frac{\alpha}{\mathbf{a}_f} (\mathbf{a}_f^\phi) \mathbf{v}_f^\phi + \frac{\alpha}{\mathbf{a}_f} V_f (\mathbf{s}^{\text{body}})_f$$

or

$$\mathbf{v}_f = \mathbf{H}_f - \mathbf{D}_f (\nabla p)_f + (1 - \alpha) \mathbf{v}_f^{n-1} + \frac{\alpha}{\mathbf{a}_f} (\mathbf{a}_f^\phi) \mathbf{v}_f^\phi + \mathbf{D}_f (\mathbf{s}^{\text{body}})_f$$
Modifications to original MIM method

Following the procedure as in Rhie and Chow method, the face velocity can be expressed as

\[
\mathbf{v}_f = \mathbf{v}_f + \left[ \mathbf{D}_f \nabla \rho_f - \mathbf{D}_f \mathbf{v}_f \right] + (1 - \alpha) \left[ \mathbf{v}^{n-1}_f - \mathbf{v}_f^{n-2} \right] \\
+ \alpha \left( \mathbf{a}_f \right)^{-1} \left[ \mathbf{a}_f^{n-1} - \mathbf{a}_f^{n-2} \right] \left[ \mathbf{v}_f^{n-1} - \mathbf{v}_f^{n-2} \right] \\
+ \left[ \mathbf{D}_f (s_{body})_f - \mathbf{D}_f (s_{body})^{n-1}_f \right]
\]

where

\[
\frac{1}{a_f} = (1 - f_p) \frac{1}{a_p} + f_p \frac{1}{a_N}
\]

\[
a_f^o = \frac{\rho \mathbf{v}_f}{\Delta t}
\]

Then, using Eqn. (d), mass flow rate through face \( f \) can be written as

\[
\dot{m}_f = \dot{m}_f^{RC} + \left( 1 - \alpha \right) \left[ \dot{m}_f^{n-1} - \rho_f \mathbf{v}_f^{n-1} \cdot \mathbf{A}_f \right] \\
+ \frac{\rho_f}{\Delta t} \left( \mathbf{D}_f \mathbf{A}_f \right) \cdot \mathbf{A}_f \left[ \dot{m}_f^{n-1} - \left( \rho_f \alpha \left( \mathbf{a}_f \right)^{-1} \left( \mathbf{a}_f^{n-1} \right) \mathbf{v}_f^{n-1} \right) \cdot \mathbf{A}_f \right] \\
+ \rho_f \left[ \mathbf{D}_f \left( s_{body} \right)_f - \mathbf{D}_f \left( s_{body} \right)^{n-1}_f \right] \cdot \mathbf{A}_f
\]

where

\[
\dot{m}_f^{RC} = \text{mass flow rate calculated from Eqn. (11.63) using Rhie-Chow method.}
\]

\[
\dot{m}_f^o = \text{mass flow rate at face } f \text{ during previous time step.}
\]

\[
\dot{m}_f^{n-1} = \text{mass flow rate at face } f \text{ during previous iteration step.}
\]

Note that \( \dot{m}_f^{n-1} \) is used during the current iteration so that no extra storage is needed for \( \dot{m}_f^{n-1} \). However, storage is needed for \( \dot{m}_f^o \).
Modifications to original MIM method

The linearly averaged value of \((s_{\text{body}})^{\text{body}}\) at a face \(f\) is estimated using

\[
(s_{\text{body}})^{\text{body}} = (1 - f_P)(s_{\text{body}})^P + f_P(s_{\text{body}})^N
\]

and the face value is estimated as:

\[
(s_{\text{body}})^f = \frac{1}{2}((s_{\text{body}})^P + (s_{\text{body}})^N) + \frac{1}{2}((\nabla(s_{\text{body}})^P \cdot \mathbf{r}_{PF} + \nabla(s_{\text{body}})^N \cdot \mathbf{r}_{NF})
\]

where the gradient at the nodal point \(P\) can be calculated using the Gauss’ rule.

The use of the neighboring cell source values for the estimation of \((s_{\text{body}})^f\) is expected to circumvent the convergence difficulties encountered in flows with large source terms.

It should be noted that Eqn. (d) is equivalent to Eqn. (c). However, Eqn (d) may be preferred since extra storage is required for \(H_f\) in Eqn (c).

Gradient reconstruction using Least-Squares method

- One popular method for calculating the gradient at points \(P\) and \(N\) is to use Gauss’ method given by Eqn’s (11.31-11.32). Another possible way is to use least-squares method.
- Consider a control volume and its neighbors:

  ![Diagram of control volume and neighbors](image)

  In LUD scheme for the case \(m_f > 0\) the value of \(\phi\) at a face \(f\) can be calculated from Eqn. (11.39) as:

  \[
  \phi_f = \phi_P + \nabla \phi_P \cdot \mathbf{r}_{PF}
  \]

  A similar equation can be written between \(P\) and its neighbor points \(A, B, C\) and \(D\):
\[
\phi_i = \phi_0 + \left( \frac{\partial \phi}{\partial x} \right)_0 (x_i - x_0) + \left( \frac{\partial \phi}{\partial y} \right)_0 (y_i - y_0)
\]

or

\[
\phi_i = \phi_0 + \left( \frac{\partial \phi}{\partial x} \right)_0 \Delta x_i + \left( \frac{\partial \phi}{\partial y} \right)_0 \Delta y_i
\]

where \( i \rightarrow 1, 2, 3, 4, \ldots \), or \( A, B, C, D, \ldots \)

For each node around “0” \((0 \rightarrow P)\) we have

\[
\phi_i - \phi_0 = \left( \frac{\partial \phi}{\partial x} \right)_0 \Delta x_i + \left( \frac{\partial \phi}{\partial y} \right)_0 \Delta y_i
\]

\[
\phi_i - \phi_1 = \left( \frac{\partial \phi}{\partial x} \right)_0 \Delta x_1 + \left( \frac{\partial \phi}{\partial y} \right)_0 \Delta y_1
\]

\[
\phi_i - \phi_1 = \left( \frac{\partial \phi}{\partial x} \right)_0 \Delta x_2 + \left( \frac{\partial \phi}{\partial y} \right)_0 \Delta y_2
\]

\[
\phi_i - \phi_1 = \left( \frac{\partial \phi}{\partial x} \right)_0 \Delta x_3 + \left( \frac{\partial \phi}{\partial y} \right)_0 \Delta y_3
\]

This set of equations can be written in matrix form as

\[
\begin{bmatrix}
\Delta x_1 & \Delta y_1 \\
\Delta x_2 & \Delta y_2 \\
\Delta x_3 & \Delta y_3 \\
\vdots & \vdots \\
\Delta x_N & \Delta y_N
\end{bmatrix}
\begin{bmatrix}
\left( \frac{\partial \phi}{\partial x} \right)_0 \\
\left( \frac{\partial \phi}{\partial y} \right)_0
\end{bmatrix}
= \\
\begin{bmatrix}
\phi_1 - \phi_0 \\
\phi_2 - \phi_1 \\
\phi_3 - \phi_2 \\
\vdots \\
\phi_N - \phi_{N-1}
\end{bmatrix}
\]

This represents an overdetermined system of equations, in the form

\[
AX = B
\]

which may be solve for \( X = \left[ (\partial \phi/\partial x)_0, (\partial \phi/\partial y)_0 \right] \) using least-squares approach. Multiplying both sides of the eqn by the transpose \( A^T \)

\[
A^T = \begin{bmatrix}
\Delta x_1 & \Delta x_2 & \Delta x_3 & \ldots & \Delta x_N \\
\Delta y_1 & \Delta y_2 & \Delta y_3 & \ldots & \Delta y_N
\end{bmatrix}
\]

we obtain

\[
A^T X = A^T B
\]
or

\[
\begin{bmatrix}
 a_{11} & a_{12} \\
 a_{21} & a_{22}
\end{bmatrix}
\begin{bmatrix}
 \frac{\partial \phi}{\partial x} \\
 \frac{\partial \phi}{\partial y}
\end{bmatrix}_0
= 
\begin{bmatrix}
 b_1 \\
 b_2
\end{bmatrix}
\]

where

\[
a_{11} = (\Delta x_1)^2 + (\Delta x_2)^2 + \ldots + (\Delta x_N)^2
\]

\[
a_{12} = a_{21} = (\Delta x_1)(\Delta y_1) + (\Delta x_2)(\Delta y_2) + (\Delta x_3)(\Delta y_3) + \ldots + (\Delta x_N)(\Delta y_N)
\]

\[
a_{22} = (\Delta y_1)^2 + (\Delta y_2)^2 + \ldots + (\Delta y_N)^2
\]

\[
b_1 = (\Delta x_1)(\phi_1 - \phi_0) + (\Delta x_2)(\phi_2 - \phi_0) + (\Delta x_3)(\phi_3 - \phi_0) + \ldots + (\Delta x_N)(\phi_N - \phi_0)
\]

\[
b_2 = (\Delta y_1)(\phi_1 - \phi_0) + (\Delta y_2)(\phi_2 - \phi_0) + (\Delta y_3)(\phi_3 - \phi_0) + \ldots + (\Delta y_N)(\phi_N - \phi_0)
\]

It can be observed that the matrix \( A^T A \) is a 2 2 matrix which can be easily inverted to solve for \( X \), i.e.

\[
X = (A^T A)^{-1} A^T B
\]

For 3D problems, \( X = [(\partial \phi / \partial x)_0, (\partial \phi / \partial y)_0, (\partial \phi / \partial z)_0] \) and \( A^T A \) is a 3 3 matrix.

Performing the inversion we obtain

\[
(\nabla \phi)_0 = 
\begin{bmatrix}
 (\nabla \phi)_0 \\
 (\nabla \phi)_0 \\
 (\nabla \phi)_0
\end{bmatrix} = 
\begin{bmatrix}
 \sum_{j=1}^{\text{total}} C_j^1 (\phi_j - \phi_0) \\
 \sum_{j=1}^{\text{total}} C_j^2 (\phi_j - \phi_0) \\
 \sum_{j=1}^{\text{total}} C_j^3 (\phi_j - \phi_0)
\end{bmatrix}
\]

where the C terms are

\[
C_j^1 = \alpha_{j,1} = \frac{r_{j,1}}{r_{1,1}} \alpha_{j,2} + \psi \alpha_{j,3}
\]

\[
C_j^2 = \alpha_{j,2} = \frac{r_{j,2}}{r_{2,2}} \alpha_{j,3}
\]

\[
C_j^3 = \alpha_{j,3}
\]

and

\[
\alpha_{j,1} = \frac{\Delta x_j}{r_{1,1}}, \quad \alpha_{j,2} = \frac{1}{r_{2,2}} \left( \Delta y_j - \frac{r_{j,2}}{r_{1,1}} \Delta x_j \right), \quad \alpha_{j,3} = \frac{1}{r_{3,3}} \left( \Delta z_j - \frac{r_{j,3}}{r_{2,2}} \Delta y_j + \psi \Delta x_j \right), \quad \psi = \frac{r_{j,2} r_{j,3}}{r_{1,1}}
\]
and
\[ r_{11} = \sqrt{\sum_{j=1}^{\text{NB}(0)} (\Delta x_j)^2}, \quad r_{12} = 1 - \frac{1}{r_{11}} \sum_{j=1}^{\text{NB}(0)} (\Delta x_j)(\Delta y_j) \]
\[ r_{13} = \frac{1}{r_{11}} \sum_{j=1}^{\text{NB}(0)} (\Delta x_j)(\Delta z_j), \quad r_{22} = \sqrt{\sum_{j=1}^{\text{NB}(0)} (\Delta y_j)^2} - r_{12}^2 \]
\[ r_{23} = \frac{1}{r_{22}} \sum_{j=1}^{\text{NB}(0)} (\Delta y_j)(\Delta z_j) - r_{22}r_{13}, \quad r_{33} = \sqrt{\sum_{j=1}^{\text{NB}(0)} (\Delta z_j)^2} - (r_{13}^2 + r_{23}^2) \]

where NB(0) refers to neighbor nodes around “0” (0 -> P).

Implementation of the least-squares gradient reconstruction requires that six \( r \) values \((r_{11}, r_{22}, r_{33}, r_{12}, r_{13}, r_{23})\) be stored at each node P.

The calculation needs to be performed only once for each node.

---

**Gradient reconstruction using Least-Squares with weighting coefficients**

In the above least-squares method each neighbor node contributes equally to the gradient reconstruction irrespective of its distance to the central point P.

The accuracy of the method may deteriorate on highly distorted meshes.

It is possible to give different weights to the contribution of each neighbor node to gradient reconstruction. This yield the following set of equations for 2D problems:

\[
\begin{bmatrix}
    w_1 \Delta x_1 & w_1 \Delta y_1 \\
    w_2 \Delta x_2 & w_2 \Delta y_2 \\
    \vdots & \vdots \\
    w_N \Delta x_N & w_N \Delta y_N \\
\end{bmatrix}
\begin{bmatrix}
    \frac{\partial \phi}{\partial x} \\
    \frac{\partial \phi}{\partial y} \\
\end{bmatrix}
= \begin{bmatrix}
    w_1 (\phi_1 - \phi_0) \\
    w_2 (\phi_2 - \phi_0) \\
    \vdots \\
    w_N (\phi_N - \phi_0) \\
\end{bmatrix}
\]

where \( w_i \) is the weighting factor of neighbor node \( i \).
The \( w_i \)'s can be selected as a function of the geometry or the solution.

- Geometrical weights can be chosen in the form
  \[ w_i = \frac{1}{|r_i^t - r_p^t|} \quad t = 0,1,2. \]
- For inverse distance weighting \( t = 1 \), \( w_i = 1/ \sqrt{\Delta x_i^2 + \Delta y_i^2 + \Delta z_i^2} \)
- The gradient using the weighted least-squares reconstruction becomes

\[
\nabla \phi_0 = \left[ \begin{array}{c}
(\nabla^2 \phi)_0 \\
(\nabla^2 \phi)_0 \\
(\nabla^2 \phi)_0 \\
\end{array} \right] = \left[ \begin{array}{c}
\frac{\partial \phi}{\partial x} \\
\frac{\partial \phi}{\partial y} \\
\frac{\partial \phi}{\partial z} \\
\end{array} \right]_0 \left[ \begin{array}{c}
\sum_{j=1}^{NB(0)} C_{j} w_j (\phi_j - \phi_0) \\
\sum_{j=1}^{NB(0)} C_{j} w_j (\phi_j - \phi_0) \\
\sum_{j=1}^{NB(0)} C_{j} w_j (\phi_j - \phi_0) \\
\end{array} \right]
\]

where the \( C \) terms are the same as that of the unweighted method:

\[
C_j^1 = \alpha_{j,1} - \frac{r_{12}}{r_{11}} \alpha_{j,2} + \psi \alpha_{j,3}, \quad C_j^2 = \alpha_{j,2} - \frac{r_{23}}{r_{22}} \alpha_{j,3}, \quad C_j^3 = \alpha_{j,3}
\]

The \( \alpha \) and \( r \) terms are

\[
\alpha_{j,1} = \frac{w_j \Delta x_j}{r_{11}}, \quad \alpha_{j,2} = \frac{1}{r_{22}} \left( w_j \Delta y_j - \frac{r_{12}}{r_{11}} w_j \Delta x_j \right), \quad \alpha_{j,3} = \frac{1}{r_{33}} \left( w_j \Delta z_j - \frac{r_{13}}{r_{11}} w_j \Delta x_j + \psi w_j \Delta y_j \right), \quad \psi = \frac{r_{12} r_{33} - r_{13} r_{23}}{r_{11} r_{22}}
\]

\[
\begin{align*}
 r_{11} &= \sqrt{\sum_{j=1}^{NB(0)} \left( w_j \Delta x_j \right)^2}, & r_{12} &= \frac{1}{r_{11}} \sum_{j=1}^{NB(0)} \left( w_j \Delta x_j \right) \left( w_j \Delta y_j \right), & r_{22} &= \sqrt{\sum_{j=1}^{NB(0)} \left( w_j \Delta y_j \right)^2}, \\
 r_{13} &= \frac{1}{r_{22}} \sum_{j=1}^{NB(0)} \left( w_j \Delta y_j \right) \left( w_j \Delta z_j \right) - r_{12} r_{33}, & r_{23} &= \sqrt{\sum_{j=1}^{NB(0)} \left( w_j \Delta z_j \right)^2} - (r_{13}^2 + r_{33}^2).
\end{align*}
\]
For Cartesian grids, for \( u_e > 0 \), using a TVD scheme, \( \phi_e \) may be written as

\[
\phi_e = \phi_P + \frac{1}{2} \psi(r)(\phi_E - \phi_P)
\]

where

\[
r = \frac{\phi_P - \phi_W}{\phi_E - \phi_P}
\]

\( \psi \) = flux limiter function

\( \psi = 0 \) for UD scheme

\( \psi = 1 \) for CD scheme

\( \psi = r \) for LUD scheme

\( \psi = 0.25r + 0.75 \) for QUICK scheme

However, in unstructured grids, for face \( f_1 \) of cell \( P \), the upwind point \( U \) is not available.

There are methods to construct \( U \), but these are costly.

In the absence of \( \phi_U \), the method proposed by Darwish and Moukalled (2003) will be used.

Since the index based notation used above is not suitable for unstructured grids, a more appropriate notation proposed by Darwish and Moukalled (2003) will be adopted.

Using this notation:

\[
\phi_f = \phi_C + \frac{1}{2} \psi(r_f)(\phi_D - \phi_C)
\]

where

\[
r_f = \frac{\phi_L - \phi_1}{\phi_D - \phi_C}
\]

The main difficulty in using TVD schemes in unstructured grids lies in the need for defining a virtual node \( U \).
Exact \( r \) Formulation

Returning to the definition of \( r \):

\[
 r_j = \frac{\phi_D - \phi_U}{\phi_D - \phi_C} = \frac{\phi_D + (\phi_C - \phi_U)}{\phi_D - \phi_C} = \frac{(\phi_D - \phi_U) - (\phi_D - \phi_C)}{\phi_D - \phi_C} = \left( \frac{\phi_D - \phi_U}{\phi_D - \phi_C} \right) - 1
\]

(11.86)

Note that \( D \) and \( C \) are the nodes straddling the interface and thus are readily available.

For node \( U \) we can write:

\[
 (\phi_D - \phi_U) = \nabla \phi_C \cdot r_{UD} = (2\nabla \phi_C \cdot r_{CD})
\]

(11.87)

since \( r_{UD} = 2r_{CD} \) (\( C \) is at the center of the \( UD \) segment). Other positions of \( U \) could also be chosen but with a loss of accuracy as the nodal gradient yields a second order accuracy only when the difference is centered at \( C \).

The formulation of \( r \) becomes

\[
 r_j = \frac{(2\nabla \phi_U \cdot r_{CD}) - (\phi_D - \phi_U)}{\phi_D - \phi_C} = \frac{(2\nabla \phi_U \cdot r_{CD})}{\phi_D - \phi_C} - 1
\]

(11.88)

After calculating \( r_j \) from Eq. (11.88) find \( \psi(r_j) \) from a TVD scheme (chapter 5) and then calculate \( \phi_f \) from Eq. (11.84).

---

Note that this \( r \)-formulation can also be used for any high order scheme without TVD. For example for QUICK scheme:

\[
 \psi(r_f) = 0.25r_f + 0.75
\]

Then, the application of high order convection schemes with TVD in a deferred correction manner, is added as a source term to \( S_{dc} \) in accordance with Eq. (11.46) as

\[
 S_{dc}^{n+1} = - \left[ \sum_{f=1}^{n_f} \left( \hat{m}_f \phi_f^H - \max(\hat{m}_f, 0.)\phi_f - \min(\hat{m}_f, 0.)\phi_f \right) \right]^{n+1}
\]

(11.89)

where, \( \phi_f^H \) is the \( \phi_f \) value computed from Eq. (11.84).
TVD Schemes in Unstructured Grids: 2) Gradient Limiters

- For structured grids flux limiters are used. For unstructured grids, the **gradient or slope limiters are used**. Instead of limiting the flux at the cell face, in this method the gradient \( \nabla \phi_P \) at cell center \( P \), is limited.

- Barth and Jespersen (1989) introduced the first gradient limiter for unstructured grids.

- The scheme consists of finding the value of \( \phi \) in a CV by limiting the gradient such that no extrema of \( \phi \) is formed:

\[
\phi(\mathbf{r}) = \phi_P + \Phi \nabla \phi_P \cdot (\mathbf{r} - \mathbf{r}_P) \quad (11.90)
\]

- \( \Phi \) = gradient limiter for the control volume at \( P \).

- That is, to prevent formation of extrema in the CV, \( \Phi \) limits the gradient such that the value of \( \phi(\mathbf{r}) \) calculated in the CV (and at the faces) should satisfy the following:

- \( \phi(\mathbf{r}) < \max(\phi_P, \phi_{Ni}), \quad i = 1, n \) and

- \( \phi(\mathbf{r}) > \min(\phi_P, \phi_{Ni}), \quad i = 1, n \)

Using the limited gradient \( \Phi \nabla \phi \), then the value of \( \phi \) at any cell face is found from

\[
\phi_f = \phi_P + \Phi \nabla \phi_P \cdot \mathbf{r}_{PF} \quad (11.91)
\]

where \( \mathbf{r}_{PF} \) is the position vector from \( P \) to the face center \( f \).

Point \( f \) is called the **reconstruction point**. (\( \phi_f \) is reconstructed from its gradient at \( P \))

However it should be noted that reconstruction points may also be chosen to be the vertices of the face, which is common in vertex-based unstructured grid methods.

The advantage of gradient limiters over the flux limiters is that, not only the convection fluxes, but diffusion fluxes are also limited, since these fluxes contain gradients of \( \phi \).
Barth and Jespersen’s gradient limiter

The following procedure is used by **Barth and Jespersen**:  
1) Find the largest negative ($\Delta_{\min} = \phi_{\min} - \phi_P$) and positive ($\Delta_{\max} = \phi_{\max} - \phi_P$) difference between the neighbors and the current CV.

2) Compute the value of $\phi_{f_i}$ for face $i$ from the unlimited expression 
   \[ \phi_{f_i} = \phi_P + \nabla \phi_P \cdot \mathbf{r}_{f_i} \]  \hspace{1cm} (11.92)

3) Compute a maximum allowable value of $\Phi_{fi}$ for each face $i$ of CV: 
   \[ \Phi_{fi} = \begin{cases} 
   1, & \text{if } \Delta_{\max} > 0 \\
   \min \left( \frac{\Delta_{\min}}{\Delta_{\max}}, 1 \right), & \text{if } \Delta_{\max} < 0 \\
   1, & \text{if } \Delta_{\max} = 0 
   \end{cases} \]  \hspace{1cm} (11.93)

   where 
   \[ \Delta_{\min} = \phi_{\max} - \phi_P, \quad \Delta_{\max} = \phi_{\min} - \phi_P, \quad \Delta_{\min} = \phi_P - \phi_P \]

   \[ \phi_{\max} = \max(\phi_P, \phi_N), \quad i = 1, n \]

   \[ \phi_{\min} = \min(\phi_P, \phi_N), \quad i = 1, n \]

4) Select 
   \[ \Phi_{BJ} = \min(\Phi_{f_i}), \quad i = 1, n \]  \hspace{1cm} (11.94)

   where $n$ = number of faces of the CV

   That is, gradient limiter of Barth-Jespersen, $\Phi_{BJ}$, is defined as the minimum of all the neighboring face limiters $\Phi_{f_i}$.

   A compact formulation of Barth-Jespersen’s limiter is given by Wang (1998) as 
   \[ \Phi_{BJ} = \min \left[ 1, \left( \frac{\max_{i=1,n} |\phi_{Ni} - \phi_P|}{\max_{i=1,n} |\phi_{f_i} - \phi_P|} \right) \left( \frac{\min_{i=1,n} |\phi_{Ni} - \phi_P|}{\min_{i=1,n} |\phi_{f_i} - \phi_P|} \right) \right] \]  \hspace{1cm} (11.95)
The function $\Phi_{fi}$ proposed by Barth and Jespersen is non-differentiable. This adversely affects the convergence of the solver.

Venkatakrishnan, (1995) proposed modifications that replace the discontinuous limiter function with a smoothly varying function.

While the new limiter does not strictly enforce monotonicity of the reconstructed fluxes, it has better convergence properties.

$$\Phi_{fi}(y) = \frac{y^2 + 2y}{y^2 + y + 2} \quad (11.96)$$

where

$$y = \frac{\Delta_{\text{max}}}{\Delta_{fi}}$$

Although this new function is differentiable, it has the disadvantage of reducing the gradient even in regions where no new extrema are formed (i.e. for $y < 2$ as can be seen in the following figure).

Venkatakrishnan’s smooth approximation to $\min(1, y)$

However, for smooth flows (where $\phi$ is nearly uniform) on a uniform mesh, $y$ is expected to be $\cong 2$ and this method introduces an error which is on the order of truncation.

To avoid applying the limiter in regions of nearly uniform flow Venkatakrishnan introduced a modification to the method.
The modification eliminates the effect of the limiter when

\( \phi_f - \phi_P \leq (K \Delta x)^{1.5} \)  \hspace{1cm} (11.97)

where \( \Delta x \) is a characteristic length for the CV (\( \Delta x = (V_{CV})^{1/3} \)).

- \( K \) is a tuneable parameter (\( K = 0.1 - 2 \), but usually \( K = 0.3 \) is used).

- The modified limiter of Venkatakrishnan is

\[
\Phi_{\beta} = \begin{cases} 
\frac{1}{\Delta_{\beta}} \left( \frac{\Delta_{\text{max}}^2 + \varepsilon^2}{\Delta_{\beta}^2 + 2\Delta_{\beta}^2 + \Delta_{\max}^2} \right) & \text{if } \Delta_{\beta} > 0 \\
1 & \text{if } \Delta_{\beta} = 0 \\
\frac{1}{\Delta_{\beta}} \left( \frac{\Delta_{\text{min}}^2 + \varepsilon^2}{\Delta_{\beta}^2 + 2\Delta_{\beta}^2 + \Delta_{\min}^2} \right) & \text{if } \Delta_{\beta} < 0
\end{cases}
\]  \hspace{1cm} (11.98)

where \( \varepsilon^2 = (K \Delta x)^3 \).

With \( K > 0 \), this modification does not maintain strict monotonicity.

**Limiter of Michalac and Ollivier-Gooch**

- Venkatakrishnan’s limiter does not provide sufficient accuracy even in smooth regions without any local extrema.

- While the Barth-Jespersen limiter does satisfy these conditions, convergence to steady state solution is difficult due to lack of differentiability of \( \text{min}(1, y) \).

- Michalac and Olliver-Gooch (2008) proposed a new approximation for \( \text{min}(1, y) \) in the limiter function of Barth-Jespersen that will be monotone and differentiable:

\[
\Phi_{\beta} = \begin{cases} 
P(y) & y < y_t \\
1 & y \geq y_t
\end{cases}
\]  \hspace{1cm} (11.99)

where \( 1 < y_t < 2 \) is a threshold and \( P(y) \) is a polynomial satisfying

\[
P_y(0) = 0, \quad P_y(y_t) = 1
\]

\[
\frac{dP}{dy}(0) = 1, \quad \frac{dP}{dy}(y_t) = 0
\]
A third degree polynomial of the form

\[ P(y) = a_3 y^3 + a_2 y^2 + a_1 y + a_0 \]  

(11.100)
can satisfy the above requirements with coefficients:

\[ a_3 = \frac{(y_t - 2)}{y_t^3}, \quad a_2 = -\frac{(1 + 3a_3 y_t^2)}{2(y_t)}, \quad a_1 = 1, \quad a_0 = 0 \]

The resulting limiters for \( y_t = 1.5, 1.75 \) are plotted in the above figure. For \( y_t = 1.5 \) accuracy is better. For \( y_t = 1.75 \) convergence is better.

### Modifications to Limiter in uniform regions

- In regions of flow which are nearly uniform, extrema will occur frequently in the transient solution.
- Therefore, it is desirable to switch-off the limiter in those regions.
- Michalac and Ollivier-Gooch suggest to switch the limiter off when
  \[ \Delta \phi \equiv (\Delta_{\text{max}} - \Delta_{\text{min}}) < (K\Delta x)^{3/2} \]  

(11.101)

- To maintain differentiability, a modified limiter, \( \Phi_{BJ\text{-mod}} \), is proposed:
  \[ \Phi_{BJ\text{-mod}} = \sigma + (1 - \sigma)\Phi_{BJ} \]  

(11.102)

where \( \Phi_{BJ} \) is Barth-Jespersen’s limiter calculated from Eqn. (11.94) and \( \sigma \) is

\[
\sigma = \begin{cases} 
1 & \text{for } (\Delta \phi)^2 \leq (K\Delta x)^3 \\
(\Delta \phi)^2 & \text{for } (K\Delta x)^3 < (\Delta \phi)^2 < 2(K\Delta x)^3 \\
0 & \text{for } (\Delta \phi)^2 \geq 2(K\Delta x)^3 
\end{cases}
\]

(11.103)

where the transition function \( s(\eta) \) is

\[ s(\eta) = 2\eta^3 - 3\eta^2 + 1, \quad \eta = \frac{(\Delta \phi)^2 - (K\Delta x)^3}{(K\Delta x)^3} \]  

(11.104)
The transition function $s(\eta)$ is used to smoothly disable the limiter in nearly uniform meshes. In this way, the cells with $\sigma = 1$ have $\Phi_{\text{new}} = 1$. As a result $\Phi$ need not be calculated in nearly uniform flow regions since from (11.102)

$$\Phi_{\text{BJ-mod}} = \sigma + (1 - \sigma)\Phi_{\text{BJ}}$$

This decreases the computer time.

### Sensitivity of results to $K$

Consider transonic flow at Mach 0.8 across an airfoil shown in the figure. (Michalac and Olliver-Gooch (2009)). The sensitivity of total pressure loss across the shock near the leading edge of the airfoil, to the tuning parameter $K$, is shown in the figure below.

The modified limiter exhibits almost no sensitivity to $K$ before the shock. Downstream the shock the results using $K = 0.25$ and $K = 1$ are indistinguishable.

Venkatakrishnan’s limiter, $\Phi_V$

Barth-Jespersen’s modified limiter, $\Phi_{\text{BJ-mod}}$
Unstructured Mesh generation

- Commercial CFD packages such as FLUENT, CFX, STAR-CD, PHOENICS, and FLOW3D have their own mesh generation tools together with their geometry modelers.
- They also have facilities to import data from solid modeler packages such as PARASOLID, PRO/ENGINEER, SOLID EDGE, SOLID WORKS, and UNIGRAPHICS.

There are also free mesh generation packages such as GMSH and SALOME. Some of these free mesh generation packages can also import geometry files from solid modelers.

A geometric model of the solution domain is first developed. Then, meshes are created on this geometric model. Salome seems to have a better GUI (graphics user interface) and is more user friendly than GMSH for the time being (as of 2012).

The mesh file created by the mesh generator software consists of two parts which give information for nodes and elements. That is:

1) $x$, $y$, and $z$ coordinates of each node in the mesh
2) list of nodes of each element together with its element type and region (zone) number.
An example drawing created by GMSH

The drawing of a square domain created by GMSH is shown in the figure. The physical region number of each boundary is denoted by numbers from 1 to 4. 1 → bottom, 2 → right, 3 → top, 4 → left boundaries. The surface is indicated by dashed lines and has a physical region number of 6.

The mesh of drawing 1 created by GMSH

There are 13 nodes (blue colors) and 24 elements. First 8 elements are boundary elements (line elements, elm type = 1). Elements 9-24 are interior elements (triangular elements, elm type = 2).

In nodes data: 1st column is node number, columns 2-4 are \(x\), \(y\), \(z\) coordinates of nodes.

In elements data: first column is element number, column 2 → elm type, column 3 → number of tags, columns 4-5 → tags, (tag1 → physical region number, tag 2 → elementary region number), the rest of columns (columns 6-7 or 6-8) → nodes of element.
For an unstructured grid, the element connectivity cannot be easily built, so this additional information is generally added to the data file.

The element information typically includes the element type or shape, and the list of nodes for each element.

Unstructured grid element numbering convention of CGNS (CFD General Notation System) will be given here.

CGNS provides a general, portable, and extensible standard for the storage and retrieval of CFD analysis data.

In an unstructured zone, the nodes are ordered from 1 to \( N \), where \( N \) is the number of nodes in the zone.

An element is defined as a group of one or more nodes, where each node is represented by its index.

The elements are indexed from 1 to \( M \) within a zone, where \( M \) is the total number of elements defined for the zone.

CGNS supports eight element shapes – points, lines, triangles, quadrangles, tetrahedra, pentahedra, pyramids, and hexahedra.

Elements describing a volume are referred to as 3-D elements.

Those defining a surface are 2-D elements.

Line and point elements are called 1-D and 0-D elements, respectively.

In a 3-D unstructured mesh, the cells are defined using 3-D elements, while the boundary patches may be described using 2-D elements.

Each element shape may have a different number of nodes, depending on whether linear or quadratic interpolation is used.

Therefore the name of each type of element is composed of two parts; the first part identifies the element shape, and the second part the number of nodes.
## Element Types in CGNS

<table>
<thead>
<tr>
<th>Dimensionality</th>
<th>Shape</th>
<th>Linear Interpolation</th>
<th>Quadratic Interpolation</th>
</tr>
</thead>
<tbody>
<tr>
<td>0-D</td>
<td>Point</td>
<td>NODE</td>
<td>NODE</td>
</tr>
<tr>
<td>1-D</td>
<td>Line</td>
<td>BAR_2</td>
<td>BAR_3</td>
</tr>
<tr>
<td>2-D</td>
<td>Triangle</td>
<td>TRI_3</td>
<td>TRI_6</td>
</tr>
<tr>
<td>Quadrangle</td>
<td></td>
<td>QUAD_4</td>
<td>QUAD_8, QUAD_9</td>
</tr>
<tr>
<td>3-D</td>
<td>Tetrahedron</td>
<td>TETRA_4</td>
<td>TETRA_10</td>
</tr>
<tr>
<td>Pyramid</td>
<td></td>
<td>PYRA_5</td>
<td>PYRA_14</td>
</tr>
<tr>
<td>Pentahedron</td>
<td></td>
<td>PENTA_6</td>
<td>PENTA_15, PENTA_18</td>
</tr>
<tr>
<td>Hexahedron</td>
<td></td>
<td>HEXA_8</td>
<td>HEXA_20, HEXA_27</td>
</tr>
</tbody>
</table>

### 1-D line elements

**Face Definition**

<table>
<thead>
<tr>
<th>Corner nodes</th>
<th>Mid node</th>
</tr>
</thead>
<tbody>
<tr>
<td>E1</td>
<td>N1, N2</td>
</tr>
<tr>
<td></td>
<td>N3</td>
</tr>
</tbody>
</table>
2-D (surface) elements

CGNS supports two shapes of 2-D elements - triangles and quadrangles.

**Edge Definition**

<table>
<thead>
<tr>
<th>Oriented edges</th>
<th>Corner nodes</th>
<th>Mid node</th>
</tr>
</thead>
<tbody>
<tr>
<td>E1</td>
<td>N1,N2</td>
<td>N4</td>
</tr>
<tr>
<td>E2</td>
<td>N2,N3</td>
<td>N5</td>
</tr>
<tr>
<td>E3</td>
<td>N3,N1</td>
<td>N6</td>
</tr>
</tbody>
</table>

**Face Definition**

<table>
<thead>
<tr>
<th>Face</th>
<th>Corner nodes</th>
<th>Mid-edge nodes</th>
<th>Oriented edges</th>
</tr>
</thead>
<tbody>
<tr>
<td>F1</td>
<td>N1,N2,N3,N4</td>
<td>N5,N6,N7,N8</td>
<td>E1,E2,E3</td>
</tr>
</tbody>
</table>

Grid point identification number. Integer ≥ 0 or blank, and no two values may be the same. Grid points N1, N2, and N3 are in consecutive order about the triangle.

Edge identification number.

Face identification number.

**Quadrilateral elements**

<table>
<thead>
<tr>
<th>E1</th>
<th>N1,N2</th>
<th>N5</th>
</tr>
</thead>
<tbody>
<tr>
<td>E2</td>
<td>N2,N3</td>
<td>N6</td>
</tr>
<tr>
<td>E3</td>
<td>N3,N4</td>
<td>N7</td>
</tr>
<tr>
<td>E4</td>
<td>N4,N1</td>
<td>N8</td>
</tr>
</tbody>
</table>

**Face Definition**

<table>
<thead>
<tr>
<th>Face</th>
<th>Corner nodes</th>
<th>Mid-edge nodes</th>
<th>Mid-face node</th>
<th>Oriented edges</th>
</tr>
</thead>
<tbody>
<tr>
<td>F1</td>
<td>N1,N2,N3,N4</td>
<td>N5,N6,N7,N8</td>
<td>N9</td>
<td>E1,E2,E3,E4</td>
</tr>
</tbody>
</table>

I. Sezai – Eastern Mediterranean University
CGNS supports four different shapes of 3-D elements - tetrahedra, pyramids, pentahedra, and hexahedra.

### 3-D Volume Elements

#### 3-D Volume Elements

**TETRA_4**

**TETRA_10**

<table>
<thead>
<tr>
<th>Edge Definition</th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Oriented edges</strong></td>
<td><strong>Corner nodes</strong></td>
<td><strong>Mid node</strong></td>
<td></td>
</tr>
<tr>
<td>E1</td>
<td>N1,N2</td>
<td>N5</td>
<td></td>
</tr>
<tr>
<td>E2</td>
<td>N2,N3</td>
<td>N6</td>
<td></td>
</tr>
<tr>
<td>E3</td>
<td>N3,N1</td>
<td>N7</td>
<td></td>
</tr>
<tr>
<td>E4</td>
<td>N1,N4</td>
<td>N8</td>
<td></td>
</tr>
<tr>
<td>E5</td>
<td>N2,N4</td>
<td>N9</td>
<td></td>
</tr>
<tr>
<td>E6</td>
<td>N3,N4</td>
<td>N10</td>
<td></td>
</tr>
</tbody>
</table>

**Face Definition**

<table>
<thead>
<tr>
<th>Face</th>
<th>Corner nodes</th>
<th>Mid-edge nodes</th>
<th>Oriented edges</th>
</tr>
</thead>
<tbody>
<tr>
<td>F1</td>
<td>N1,N3,N2</td>
<td>N7,N6,N5</td>
<td>E3,-E2,-E1</td>
</tr>
<tr>
<td>F2</td>
<td>N1,N2,N4</td>
<td>N5,N9,N8</td>
<td>E1,E5,-E4</td>
</tr>
<tr>
<td>F3</td>
<td>N2,N3,N4</td>
<td>N6,N10,N9</td>
<td>E2,E6,-E5</td>
</tr>
<tr>
<td>F4</td>
<td>N3,N1,N4</td>
<td>N7,N8,N10</td>
<td>E3,E4,-E6</td>
</tr>
</tbody>
</table>

**Pyramid elements**

**PYRA_5**

**PYRA_14**

<table>
<thead>
<tr>
<th>Edge Definition</th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Oriented edges</strong></td>
<td><strong>Corner nodes</strong></td>
<td><strong>Mid node</strong></td>
<td></td>
</tr>
<tr>
<td>E1</td>
<td>N1,N2</td>
<td>N6</td>
<td></td>
</tr>
<tr>
<td>E2</td>
<td>N2,N3</td>
<td>N7</td>
<td></td>
</tr>
<tr>
<td>E3</td>
<td>N3,N4</td>
<td>N8</td>
<td></td>
</tr>
<tr>
<td>E4</td>
<td>N4,N1</td>
<td>N9</td>
<td></td>
</tr>
<tr>
<td>E5</td>
<td>N1,N5</td>
<td>N10</td>
<td></td>
</tr>
<tr>
<td>E6</td>
<td>N2,N5</td>
<td>N11</td>
<td></td>
</tr>
<tr>
<td>E7</td>
<td>N3,N5</td>
<td>N12</td>
<td></td>
</tr>
<tr>
<td>E8</td>
<td>N4,N5</td>
<td>N13</td>
<td></td>
</tr>
</tbody>
</table>

**Face Definition**

<table>
<thead>
<tr>
<th>Face</th>
<th>Corner nodes</th>
<th>Mid-face nodes</th>
<th>Oriented edges</th>
</tr>
</thead>
<tbody>
<tr>
<td>F1</td>
<td>N1,N4,N3,N2</td>
<td>N9,N8,N7,N6</td>
<td>E4,-E3,-E2,-E1</td>
</tr>
<tr>
<td>F2</td>
<td>N1,N2,N5</td>
<td>N6,N11,N10</td>
<td>E1,E6,-E5</td>
</tr>
<tr>
<td>F3</td>
<td>N2,N3,N5</td>
<td>N7,N12,N11</td>
<td>E2,E7,-E6</td>
</tr>
<tr>
<td>F4</td>
<td>N3,N4,N5</td>
<td>N8,N13,N12</td>
<td>E3,E8,-E7</td>
</tr>
<tr>
<td>F5</td>
<td>N4,N1,N5</td>
<td>N9,N10,N13</td>
<td>E4,E5,-E8</td>
</tr>
</tbody>
</table>
Pentahedral elements

<table>
<thead>
<tr>
<th>Edge Definition</th>
<th>Corner nodes</th>
<th>Mid node</th>
</tr>
</thead>
<tbody>
<tr>
<td>E1</td>
<td>N1, N2</td>
<td>N7</td>
</tr>
<tr>
<td>E2</td>
<td>N1, N3</td>
<td>N8</td>
</tr>
<tr>
<td>E3</td>
<td>N1, N1</td>
<td>N9</td>
</tr>
<tr>
<td>E4</td>
<td>N1, N4</td>
<td>N10</td>
</tr>
<tr>
<td>E5</td>
<td>N2, N5</td>
<td>N11</td>
</tr>
<tr>
<td>E6</td>
<td>N3, N6</td>
<td>N12</td>
</tr>
<tr>
<td>E7</td>
<td>N4, N5</td>
<td>N13</td>
</tr>
<tr>
<td>E8</td>
<td>N5, N6</td>
<td>N14</td>
</tr>
<tr>
<td>E9</td>
<td>N6, N4</td>
<td>N15</td>
</tr>
</tbody>
</table>

Hexahedral elements

<table>
<thead>
<tr>
<th>Edge Definition</th>
<th>Corner nodes</th>
<th>Mid node</th>
</tr>
</thead>
<tbody>
<tr>
<td>E1</td>
<td>N1, N2</td>
<td>N9</td>
</tr>
<tr>
<td>E2</td>
<td>N2, N3</td>
<td>N10</td>
</tr>
<tr>
<td>E3</td>
<td>N3, N4</td>
<td>N11</td>
</tr>
<tr>
<td>E4</td>
<td>N4, N1</td>
<td>N12</td>
</tr>
<tr>
<td>E5</td>
<td>N1, N5</td>
<td>N13</td>
</tr>
<tr>
<td>E6</td>
<td>N2, N6</td>
<td>N14</td>
</tr>
<tr>
<td>E7</td>
<td>N3, N7</td>
<td>N15</td>
</tr>
<tr>
<td>E8</td>
<td>N4, N8</td>
<td>N16</td>
</tr>
<tr>
<td>E9</td>
<td>N5, N6</td>
<td>N17</td>
</tr>
<tr>
<td>E10</td>
<td>N6, N7</td>
<td>N18</td>
</tr>
<tr>
<td>E11</td>
<td>N7, N8</td>
<td>N19</td>
</tr>
<tr>
<td>E12</td>
<td>N8, N5</td>
<td>N20</td>
</tr>
</tbody>
</table>

Face Definition

<table>
<thead>
<tr>
<th>Face</th>
<th>Corner nodes</th>
<th>Mid-edge nodes</th>
<th>Mid-face node</th>
<th>Oriented edges</th>
</tr>
</thead>
<tbody>
<tr>
<td>F1</td>
<td>N1, N2, N3, N4, N7, N11</td>
<td>N12, N10</td>
<td>N16</td>
<td>E1, E4, E2, E3, E5, E6, E7, E9</td>
</tr>
<tr>
<td>F2</td>
<td>N2, N3, N6, N5, N8, N12, N14</td>
<td>N11</td>
<td>N17</td>
<td>E2, E6, E8, E5</td>
</tr>
<tr>
<td>F3</td>
<td>N3, N1, N4, N6, N9, N10, N15, N18</td>
<td>N12</td>
<td>N18</td>
<td>E3, E4, E9, E6</td>
</tr>
<tr>
<td>F4</td>
<td>N1, N3, N2, N9, N8, N7</td>
<td>N13, N14, N15</td>
<td>N19</td>
<td>E5, E2, E1</td>
</tr>
<tr>
<td>F5</td>
<td>N4, N5, N6, N13, N14, N15</td>
<td>N17, N18</td>
<td>N20</td>
<td>E7, E8, E9</td>
</tr>
</tbody>
</table>

I. Sezai – Eastern Mediterranean University
References in addition to that of the textbook