1 Finite Volume Methods for Unstructured Grids

1.1 Introduction

The *finite volume* (FV) method starts with the integral form of a conservation equation:

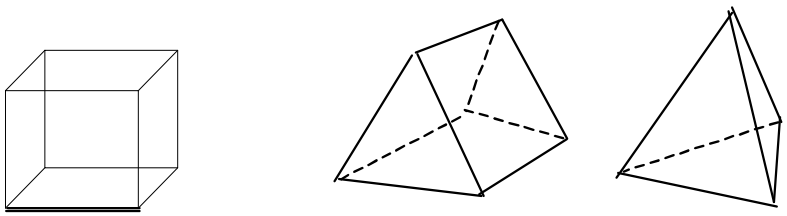
$$\frac{\partial}{\partial t}\int_\Omega \rho\,\phi\,d\Omega + \int_S \rho\,\phi\,\underline{v}.\underline{n}\,dS = \int_S \rho\,\gamma\,(\underline{grad}(\phi).\underline{n})dS + \int_\Omega q_\phi\,d\Omega \tag{1}$$

and is obtained by integrating:

$$\frac{\partial \rho\phi}{\partial t} + div(\underline{v}\,\rho\,\phi) = div(\rho\,\gamma\underline{grad}(\phi)) + q_\phi \tag{2}$$

over any *control volume* Ω bounded by the surface S, on which $\underline{n}$ is the normal directed toward the outside of Ω.

The integral conservation equation (1) is valid for any control volume Ω. It is applied to every cell of the mesh. These cells usually consist of rectangles or triangles in 2D, and hexahedra, prisms, or tetrahedra in 3D. More generally, any type of volume, bounded by N_s planar surfaces, is possible.



Hexahedra, Prisms, and Tetrahedra

The control volumes (CV) are noted T_i (considering Triangles as generic examples), the union of all CV ($i = 1, .., N_{CV}$) forming a partition of the entire flow domain. The surface integrals appearing in (1) when applied to the CV T_i is simply the sum of fluxes over the number Ns of planar surfaces separating T_i from the neighbour cells:

$$\int_{T_i} \rho\,\phi\,\underline{v}.\underline{n}\,dS = \sum_{j=1,Ns}\int_{S_{ij}} \rho\,\phi\,\underline{v}.\underline{n}\,dS$$

The contribution $\int_{S_{ij}} \rho\,\phi\,\underline{v}.\underline{n}\,dS$ in the summation $\sum_{j=1,Ns}$ is the flux of the quantity $\rho\,\phi$ carried by the flow velocity $\underline{v}$ across the surface S_{ij} between cells i and j. For cell P in figure 1, this writes:

$$\begin{aligned}\int_{cell\,P} div(\underline{v}\,\rho\,\phi)dS &= \int_{surface\ cell\,P} \rho\,\phi\,\underline{v}.\underline{n}\,dS \\ &= \int_{S_{PE}} \rho\,\phi\,\underline{v}.\underline{n}\,dS + \int_{S_{PN}} ... + \int_{S_{PW}} ... + \int_{S_{PS}} ...\end{aligned}$$

The calculation for one interfacial surface S_{ij} between cell i and cell j needs to be described only once, even though it will appear again when computing the fluxes into cell j. Indeed only the direction of the external normal is reversed, thus only the sign of the flux needs to be exchanged. For example on fig 1, the absolute value of flux $\int_{S_{PE}} \rho\,\phi\,\|\,\underline{v}.\underline{n}_{PE}\|\,dS_j$ across the boundary between the quadrangle centred on P and the triangle centred on E *must* be calculated only once, then it contributes to cell P with a + sign if $\underline{v}.\underline{n}_{PE}. > 0$ if $\underline{v}$ is pointing out of P ($\underline{n}_{PE}$ is pointing to the exterior of cell P). When considering cell E, the same expression will appear except that now $\underline{n}_{EP}$ is pointing to the exterior of cell E toward cell P hence a - sign is affected to the absolute flux if $\underline{v}.\underline{n}_{EP}. < 0$.

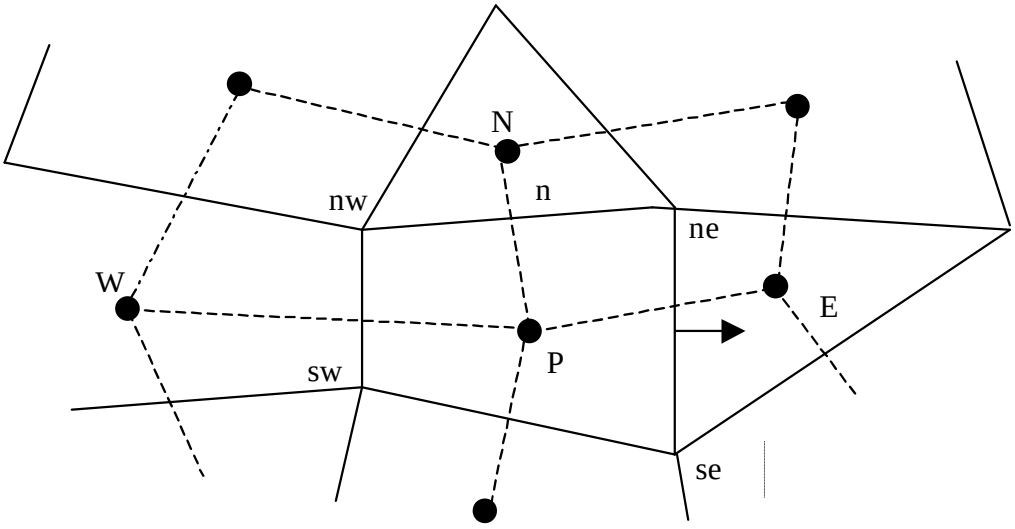


Figure 1: Combination of rectangular and triangular CV

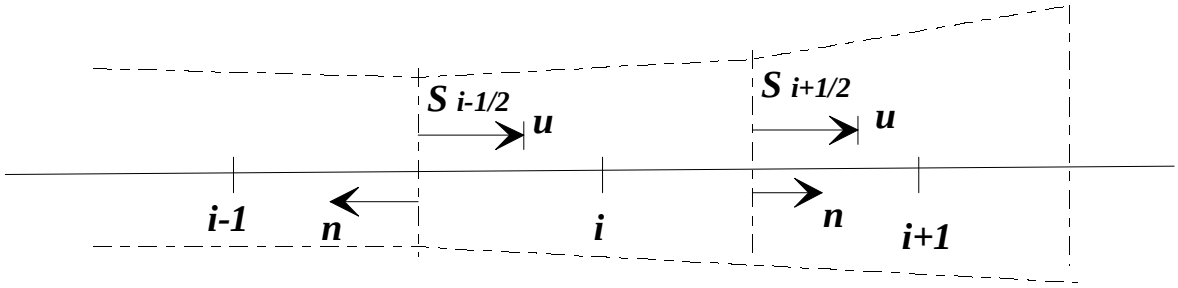


Figure 2: 1D convection problem

This is what makes the method absolutely conservative, for mass, momentum and scalars, however coarse the mesh. It is of course very appealing for industrial calculations where mesh refinement is not always possible.

However it is quite difficult to evaluate ϕ at the interface, since one knows only the *average* values $\overline{\phi_P}$ and $\overline{\phi_E}$ as is explained further. This step is often (hardly!) first order accurate on strongly distorted meshes. (It is usually quite accurate on regular meshes, i.e. second order, but the main reason for using unstructured FV methods is precisely that they can be applied to very complex geometries, where the mesh generators usually produce quite distorted cells)

1.2 1D example

The issues of FV against FD can be first outlined on the following 1D example of convection in a pipe with variable section as shown in figure 2.

$$\frac{\partial \phi}{\partial t} + div(\underline{v}\,\phi) = 0 \tag{3}$$

In 1D the velocity vector is simply $\underline{v} = (u, 0, 0)$ where u is assumed constant, then $\underline{v}\,\underline{n} = u$ if $\underline{n}$ is pointing to the right and $-u$ otherwise.

Let Ω the control volume centred on node i and bounded by surfaces $S_{i+1/2}$, $S_{i-1/2}$. The FV

formulation is then:

$$\frac{\partial}{\partial t}\int_{\Omega}\phi\, d\Omega + \int_{S}\phi.\underline{v\, n}\, dS = \frac{\partial \overline{\phi}^{i}}{\partial t}\left(x_{i+1/2}-x_{i-1/2}\right)\frac{S_{i+1/2}+S_{i-1/2}}{2} \tag{4}$$

$$+\,\phi\left(x_{i+1/2}\right)S_{i+1/2}(\underline{v\, n})_{i+1/2}+\phi\left(x_{i-1/2}\right)S_{i-1/2}(\underline{v\, n})_{i-1/2} \tag{5}$$

This formula would be valid for a pipe with variable section. For simplicity let us assume the sections constant and equal to unity. Then:

$$\frac{\partial \overline{\phi}^{i}}{\partial t}\left(x_{i+1/2}-x_{i-1/2}\right)+u\left[\phi\left(x_{i+1/2}\right)-\phi\left(x_{i-1/2}\right)\right]=0$$

note that $(\underline{v\, n})_{i+1/2}=+u$, while $(\underline{v\, n})_{i-1/2}=-u$ and $S_{i+1/2}=S_{i-1/2}\ =1$. Furthermore we have defined the *average* value of ϕ inside the CV as:

$$\overline{\phi}^{i}=\frac{1}{\left(x_{i+1/2}-x_{i-1/2}\right)}\int_{x_{i-1/2}}^{x_{i+1}/2}\phi.dx \tag{6}$$

One could also introduce the *average* value of the derivative:

$$\overline{\left(\frac{\partial\phi}{\partial x}\right)}^{i}=\frac{1}{\left(x_{i+1/2}-x_{i-1/2}\right)}\int_{x_{i-1/2}}^{x_{i+1}/2}\frac{d\phi}{dx}dx=\frac{\phi\left(x_{i+1/2}\right)-\phi\left(x_{i-1/2}\right)}{\left(x_{i+1/2}-x_{i-1/2}\right)} \tag{7}$$

Then the FV formulation can be seen as:

$$\left[\frac{\partial \overline{\phi}^{i}}{\partial t}+u\overline{\left(\frac{\partial\phi}{\partial x}\right)}^{i}\right]\left(x_{i+1/2}-x_{i-1/2}\right)=\frac{\partial \overline{\phi}^{i}}{\partial t}\left(x_{i+1/2}-x_{i-1/2}\right)+u\left[\phi\left(x_{i+1/2}\right)-\phi\left(x_{i-1/2}\right)\right] \tag{8}$$

This is not an *approximation*, but an *exact* representation of the physical process occurring within the segment $\left(x_{i+1/2}-x_{i-1/2}\right)$ centred around node x_i .

However several *approximations* will need to the introduced:

- Interpolation: $\phi\left(x_{i+1/2}\right)$ needs to be interpolated from the local values which are stored at nodes x_{i+1} and x_i.
- *Local* values of ϕ at node x_i are not even know, since the FV formulation gives equations for the *average* value over the cell:

$$\int_{x_{i-1/2}}^{x_{i+1}/2}\frac{\partial\phi}{\partial t}dx=\frac{\partial}{\partial t}\int_{x_{i-1/2}}^{x_{i+1}/2}\phi.dx=\frac{\partial \overline{\phi}^{i}}{\partial t} \tag{9}$$

  With a constant grid spacing Δx, (8) is identical to a FD discretisation if the following approximation are introduced:

$$\phi(x_i)\approx\overline{\phi}^{i} \tag{10}$$

and:

$$\phi\left(x_{i+1/2}\right)\approx\frac{\phi\left(x_{i+1}\right)+\phi\left(x_i\right)}{2} \tag{11}$$

i.e. equation (8) is then equivalent to the centred FD:

$$\frac{\partial\phi}{\partial t}+u\frac{\partial\,\phi}{\partial x}\approx\frac{\partial\phi_i}{\partial t}+u\frac{\phi\left(x_{i+1}\right)-\phi\left(x_{i-1}\right)}{2\Delta x} \tag{12}$$

Although FV methods are currently very appealing for their simplicity and possible use on any type of unstructured grid, these methods still contain discretisation errors, which are less obviously exhibited than the truncation errors of finite differences.

Question: What is the order of the approximation $\phi(x_i)\approx\overline{\phi}^{i}$?

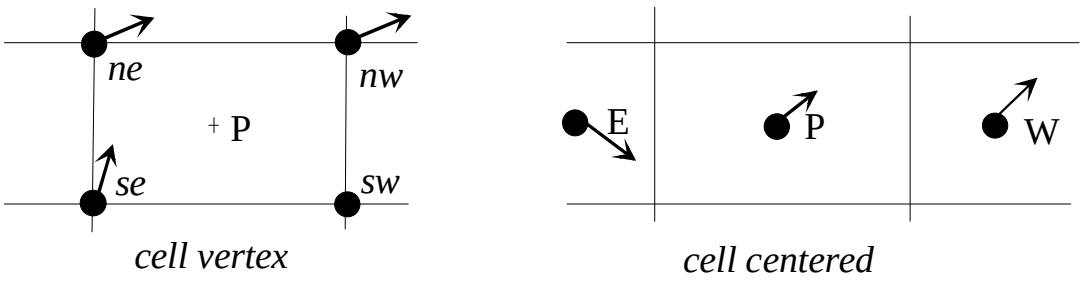


Figure 3: two possibilities for storage of variables

1.3 Cell Centred and Cell Vertex Storage

Figure 3 shows the two possibilities for the location of the discretized variables:

- The *cell-centred* approach where $\overline{\phi}$ is stored at the centre of the cell ($\overline{\phi}_P, \overline{\phi}_E, \overline{\phi}_W$...are stored). This makes the approximation of the volume integral straightforward, using the mid-point rule:

$$\int_{Cell_P} \rho\, \phi\, d\Omega = \rho_P\, \overline{\phi}_P\, \Omega_P$$

while for the convective fluxes an interpolation of the variables at the interface, e is necessary.

$$\int_{S_e} \rho\, \phi\, \underline{v}.\underline{n}\, dS = \left[\rho\, \phi\, \underline{v}\right]_e .\underline{n}\, S_e$$

with for instance at the lowest order:

$$\left[\rho\, \phi\, \underline{v}\right]_e \approx \frac{1}{2}\left(\left[\rho\, \overline{\phi}\, \underline{v}\right]_P + \left[\rho\, \overline{\phi}\, \underline{v}\right]_E\right) \tag{13}$$

This however introduces some error since ordinarily (except for regular grids) the surface centre e is not exactly the middle of cell centres P and E.

Note that $\underline{n}\, S_e$ is a constant geometric factor, computed and stored once for all at the beginning of the calculation.

- The *cell-vertex* approach where ϕ is stored at the cell corners (ϕ_{ne}, ϕ_{se}, ϕ_{sw}, ϕ_{nw} are stored). This makes the approximation of the convective fluxes straightforward,

$$\int_{S_e} \rho\, \phi\, \underline{v}.\underline{n}\, dS = \frac{1}{2}\left(\left[\rho\, \phi\, \underline{v}\right]_{ne} + \left[\rho\, \phi\, \underline{v}\right]_{se}\right) .\underline{n}\, S_e$$

Since ϕ_{ne} and $\underline{v}_{ne}$ is exactly the storage location of the variables, the above is now exactly the trapezoidal rule for the discretisation of the surface integral, even if the cell is very distorted. The trade-off is that the volume integral is now given by:

$$\int_{\Omega_P} \rho\, \phi\, d\Omega = \frac{1}{4}\left(\left[\rho\, \phi\right]_{ne} + \left[\rho\ \phi\right]_{se} + \left[\rho\, \phi\right]_{sw} + \left[\rho\ \phi\right]_{nw}\right)\, \Omega_P$$

However, since we need to introduce this expression in time derivative, i.e. at time level t^n and t^{n+1} a complex system of coupled implicit equations results (similar to the mass matrix in Finite Elements).

On the contrary, the cell centred choice simply leads to:

$$\frac{\partial}{\partial t}\int_{\Omega} \rho\, \phi\, d\Omega = \frac{\left(\rho_P\, \phi_P\right)^{n+1} - \left(\rho_P\, \phi_P\right)^{n}}{\Delta t}\Omega_P$$

which is obviously much simpler.

1.4 Dual mesh, Finite Volume-Elements

Actually, in the cell-vertex case, new control volumes (called dual mesh, see figure 4) are often constructed around the cell vertex, onto which the cell-centred approach is then applied. To compute the fluxes along the new CV boundaries, variables need to be interpolated from the cell vertices (where they are stored). There seems to be no difference with the cell centred approach except that we have created a new "dual mesh", but since this dual mesh is not given by the mesh generator, but by the FV code, it can be defined in an way that will reduce the error in the approximation of the interpolation of the value on the surface (13). In the figure above for example, the dual mesh is constructed from the centres of the triangles (*) and the middles of segments (+). Gradients and interpolated values at these dual nodes are then easily obtained, but the number of sub elements across which the fluxes must be computed is very large (10 on the above example, compared to the 3 sides of the original triangular mesh).

For triangles and quadrangles (hexa and tetrahedra in 3D) it is convenient to use the well-known shape functions of the finite element method.

$$\phi(\underline{x}) = \sum \phi_i \; p_i(\underline{x}) \tag{14}$$

where $p_i(\underline{x})$ is a polynomial whose value is 1 when $\underline{x} = \underline{x}_i$ and 0 on the other vertices $\underline{x}_j$

In opposition to traditional finite elements, there is no variational formulation, i.e. no multiplication by shape functions, equation (1) is simply used as it stands, and the shape functions are only used for interpolations. This is called the Finite-Volume/Finite-Element (FV/FE) method.

To sum up, the cell centred FV approach uses the original mesh (provided by the mesh generator) as control volumes and stores variables at their centre. The cell vertex FV-FE approach stores the variables on the original mesh nodes, and reconstructs new control volumes around them, in a way that will facilitate (or make more accurate) the calculation of the interface fluxes.

A disadvantage of the cell-vertex storage is that it involves a significantly larger number of interfacial fluxes to compute. Also, the stored variables lie exactly on the boundary, which does not simplify the use of e.g. "wall functions" in turbulent flows where Neuman conditions are imposed rather than Dirichlet conditions.

In the following, we consider only cell-centred discretisation.

1.5 Diffusive Fluxes and Gradient Reconstruction

The Diffusive flux is calculated in the same manner, except that an approximation of the *gradients* at the cell face centre "e" is now needed. The integral of the gradient on the cell face S_e is approximated by the value at the middle of this face, time the surface projected onto the normal.

$$\int_{cell\ P} div(\rho\,\gamma\,\underline{grad}(\phi))d\Omega \;=\; \int_{surface\ cell\ P} \rho\,\gamma\,\underline{grad}(\phi).\underline{n}\,dS \tag{15}$$

$$\int_{S_e} \gamma\,\underline{grad}(\phi)\,.\underline{n}\,dS \;=\; \left[\gamma\,\underline{grad}(\phi)\right]_e.\underline{n}\,S_e \tag{16}$$

The gradient at the cell centre, is defined by the Gauss theorem:

$$\int_\Omega \frac{\partial\phi}{\partial x_i}\,d\Omega = \int_S \phi\,\underline{e_i}.\underline{n}\,dS = \sum_{c=1,Nfaces} \phi_c S_c \underline{e_i}.\underline{n} \tag{17}$$

where the summation is performed on all interface centres, c, (for a rectangle $c = e, n, w, s$). The geometric coefficient, $S_c\underline{e_i}.\underline{n} = S_c^i$, is the area of the interface surface projected on the $\underline{e_i}$ base vector of the Cartesian frame ($\underline{e_x}$ is the unit vector along the x axis, the procedure is repeated for $i = x, y$, then z).

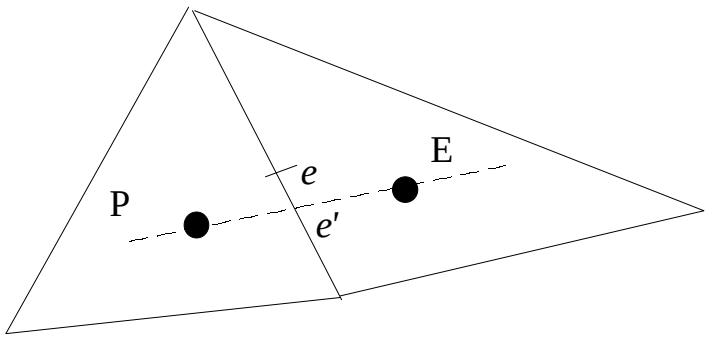


Figure 4: center of face e, and intersection e'

Next the volume integral on the left of (17) is evaluated using the value at the CV centre:

$$\int_\Omega \frac{\partial \phi}{\partial x_i}\, d\Omega = \left[\frac{\partial \phi}{\partial x_i}\right]_P \Omega \tag{18}$$

Thus

$$\left[\frac{\partial \phi}{\partial x_i}\right]_P = \frac{1}{\Omega} \sum_{c=1,Nfaces} \phi_c S_c^i \tag{19}$$

If the grid is orthogonal, then ϕ_e for instance is obtained by linear interpolation from the CV centres

$$\phi_e \cong \alpha \phi_P + (1-\alpha)\phi_E \tag{20}$$

using the ratio of lengths $\alpha = L_{eE}/L_{PE}$

Once the gradients at the centres are known using (19), then in turn they can be interpolated to give the gradients at the faces:

$$\begin{aligned} \left[\underline{grad}\phi\right]_e &= \alpha \left[\underline{grad}\phi\right]_P + (1-\alpha)\left[\underline{grad}\phi\right]_E \\ \left[\frac{\partial \phi}{\partial x_i}\right]_e &= \alpha \frac{1}{\Omega_P} \sum_{c=cell\ P\ faces} \phi_c S_c^i + (1-\alpha)\frac{1}{\Omega_E} \sum_{c=cell\ E\ faces} \phi_c S_c^i \end{aligned} \tag{21}$$

The problem, as shown in figure 22, is that for a non orthogonal grid, the interpolation (20) is only correct for point e' , the intersection of line PE with the east face. Thus is must be corrected by:

$$\phi_{e'} \cong \alpha \phi_P + (1-\alpha)\phi_E \tag{22}$$

$$\phi_e = \phi_{e'} + \underline{e'e}.\left[\underline{grad}\phi\right]_e \tag{23}$$

Unfortunately, the gradients are not known at this stage.

The problem, in order to compute the gradients at the cell CV centres, is that the values of ϕ on the cell faces are needed, and in order to obtain these, when the mesh is not orthogonal, the interpolations from the CV centres to the centres of the faces involve the gradients. This "gradient reconstruction" is an implicit relation that can be solved interatively.

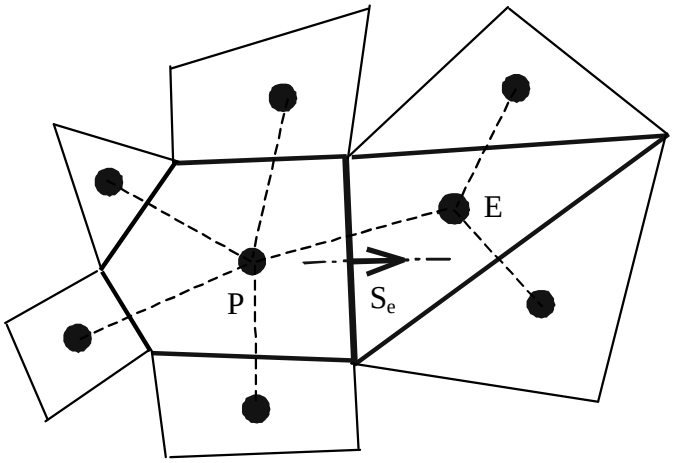
After the initial sequence (20), (21), (22), one can iterate over (21) and (22).

This can be a very time consuming process. For stability reasons, the diffusion step of the Navier Stokes equation solver is usually implicit, and solved iteratively by a Gauss-Seidel or conjugate gradient method. Within each of these iterations, new values of ϕ_P are obtained, and so new "gradient reconstruction" steps must be performed. Overall, this is an iterative process, within an iterative process. This is clearly a disadvantage compared to finite elements where the matrices are clearly defined at the beginning, separate from the linear system resolution.

Alternatively, on can proceed "as if" the grid was orthogonal, for the implicit part of the code, and use values of the gradient from the previous time step to correct the non-orthogonality in the interpolation, as explained further.

Indeed, there is yet an additional problem which is that the process of first calculating the gradients at the cell centres, then interpolating them on the cell faces leads to a very large discretisation molecule (a Laplacian calculated on cell P involves all neighbours of cell P, but also all neighbours of neighbours).

Using values in all the neighbouring cells of P, $\left[\underline{grad}\,\phi\right]_P$ is computed, then values of all neighbours of cell E are needed to obtain $\left[\underline{grad}\,\phi\right]_E$. Next, $\left[\underline{grad}\,\phi\right]_e$ is given by interpolation from the two previous. This involves a very large molecule as seen on the figure below, and moreover the same procedure must be extended to all faces of P. Also, the iteration between (21) and (22) is not local, restricted to cell P, but must be carried over all cells of the mesh, i.e. compute (21) for all cells, then (22) for all cells, and repeat.



CV involved in the calculation of $\left[\underline{grad}\,(\phi)\right]_e$

Instabilities resulting from a too large dicretisation molecule can be illustrated by comparison to the more familiar Finite Differences on a rectangular Cartesian grid. The projection of the north and south interfaces in the x direction is zero, while that of the east and west interfaces is simply:

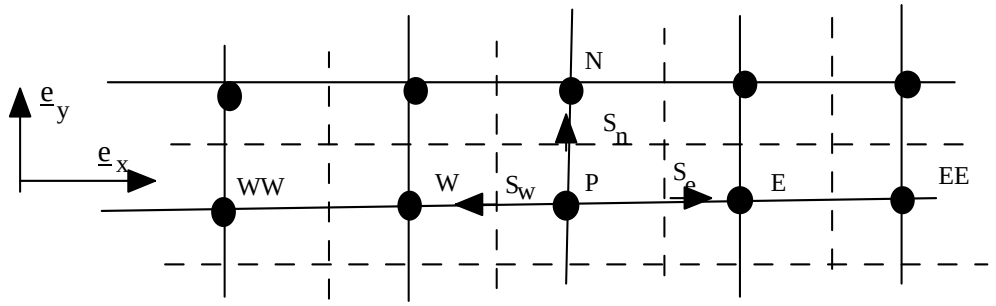


Figure 5:

$$S_n^x = S_n \underline{e_x}.\underline{n} = S_n \underline{e_x}.\underline{e_y} = 0$$
$$S_e^x = S_e \underline{e_x}.\underline{n} = S_e \underline{e_x}.\underline{e_x} = S_e = \Delta y$$
$$S_s^x = 0 \quad \text{and} \quad S_w^x = S_w \underline{e_x}.(-\underline{e_x}) = -\Delta y$$

Thus

$$\left[\frac{\partial \phi}{\partial x}\right]_P = \frac{1}{\Delta x \Delta y} \sum_c \phi_c S_c^i = \frac{1}{\Delta x \Delta y} \left(\phi_e \Delta y - \phi_w \Delta y\right)$$

when the interface values are replaced by the interpolation of the nodal values, this results in:

$$\left[\frac{\partial \phi}{\partial x}\right]_P = \frac{(\phi_E - \phi_W)}{2\Delta x}$$

which is the standard second order central difference. Fortunately no gradient reconstruction iterations are needed here since the grid-lines are orthogonal. In the momentum equation of cell P will appear the difference between east and west fluxes (resulting from the volume integrated laplacian):

$$\begin{aligned}
\left[\frac{\partial \phi}{\partial x}\right]_e - \left[\frac{\partial \phi}{\partial x}\right]_w &= \frac{1}{2}\left(\left[\frac{\partial \phi}{\partial x}\right]_E + \left[\frac{\partial \phi}{\partial x}\right]_P\right) - \frac{1}{2}\left(\left[\frac{\partial \phi}{\partial x}\right]_P + \left[\frac{\partial \phi}{\partial x}\right]_W\right) \\
&= \frac{1}{2}\left(\frac{(\phi_{EE} - \phi_P)}{2\Delta x} - \frac{(\phi_P - \phi_{WW})}{2\Delta x}\right) \\
&= \frac{\phi_{EE} - 2\phi_P - \phi_{WW}}{4\Delta x} \\
&= \frac{\phi_{j+2} - 2\phi_j - \phi_{j-2}}{4\Delta x}
\end{aligned}$$

On the last line we have switched to finite difference notation to show that even-nodes are linked only to even-nodes, (and odd nodes to only odd). This is similar to the checkerboard oscillations that can appear for the discretisation of pressure on a collocated grid. In practice however this mainly appears on structured grids. As soon as a few triangles (or non cubic elements in 3D) are introduced, the solutions tend to be recoupled (imagine a checkerboard where two cells are lumped together).

Nevertheless, the above construction results in a broad stencil, that, when reaching the stage linear solving linear system leads to a matrix with many non-zero terms. Since we will be interested in solving the diffusion implicitly, for stability reasons, this matrix will be difficult to inverse. For a non regular grids, the above "pseudo laplacian" results in:

$$\left[\frac{\partial \phi}{\partial x}\right]_e - \left[\frac{\partial \phi}{\partial x}\right]_w = a\phi_{j+2} + b\phi_{j+1} + c\phi_j + d\phi_{j+1} + d\phi_{j+2}$$

When 2 or 3D problems and non orthogonality are considered, many more points are involved. It is thus desirable to seek a more compact approximation to the fluxes.

Note that when the control volumes are either rectangles or equilateral triangles, the centre of mass lies also along the normal directions to the interface centres, so that it is possible to directly compute the gradient along the normal:

$$\left[\frac{\partial \phi}{\partial n}\right]_e = \frac{\phi_E - \phi_P}{L_{PE}} \tag{24}$$

where L_{PE} is the distance between CV centres. This can then directly be cast in the flux integral:

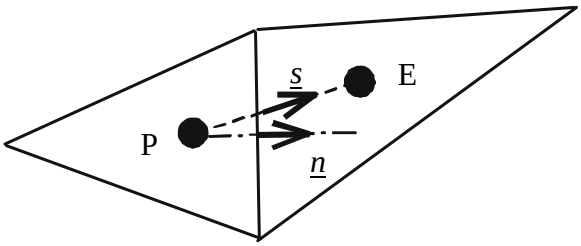
$$\int_{S_e} \underline{grad}(\phi) \,.\underline{n}\, dS = \left[\frac{\partial \phi}{\partial n}\right]_e . S_e$$

For the case of regular grids we recover the usual discretisation of the Laplacian[1].

$$\left[\frac{\partial \phi}{\partial x}\right]_e - \left[\frac{\partial \phi}{\partial x}\right]_w = \frac{\phi_E - 2\phi_P + \phi_W}{\Delta x} = \frac{\phi_{j+1} - 2\phi_j + \phi_{j-1}}{\Delta x}$$

This obviously leads to a much more compact discretisation, which moreover is free of odd-even oscillations. The problem is that we are in fact calculating the gradient along the direction $\underline{s}$ connecting the CV centres, instead of $\underline{n}$. However we can use it as a first approximation and correct it with the more accurate (but expensive) gradient reconstruction by Gauss theorem method.

[1] mutiplied by Δx (here in 1D) because we started from the volume integrated form of convection diffusion equaitons



$$\int_{S_e} \underline{grad}(\phi) \,.\underline{n}\, dS = \underbrace{\frac{\phi_E - \phi_P}{L_{PE}} S_e}_{implicit} + \underbrace{\left[\underline{grad}\,\phi\right]_e .(\underline{n} - \underline{s}) S_e}_{\text{explicit interpolation}}$$

The correction tends to zero when $\underline{n}$ and $\underline{s}$ are nearly aligned, which is expected from a "good quality" CFD mesh. We can then use "older" values of $\underline{grad}\,\phi$ from the previous iteration or time-step. This is called *deferred correction* for non orthogonality. It is a less expensive method because the linear system to be solved is easier as the matrix corresponding to the implicit terms has a better conditioning. Recall that when solving a system $A.X = B$ by iterative methods (by Residual Minimisation methods such as GMRES, or simpler Gauss-Seidel methods) the matrix-vector $A.X$ is computed many times, but the right hand side term, B collecting all explicit terms (including the deferred correction) is only computed once.

A different type of correction for non-orthogonality, called "least-squares gradient reconstruction" is given in the appendix.

References

[1] L. Davidson, "A pressure correction Method for Unstructured Arbitrary Control Volumes," International Journal for Numerical Methods in Fluids 22, no. 4 (1996): 265-281.

[2] H. Jasak and A.D. Gosman, "Automatic resolution control for the finite-volume method, part 2: Adaptive mesh refinement and coarsening," Numerical Heat Transfer, Part B: Fundamentals 38, no. 3 (2000): 257-271.

[3] S. Muzaferija and D. Gosman, "Finite-Volume CFD Procedure and Adaptive Error Control Strategy for Grids of Arbitrary Topology," Journal of Computational Physics 138, no. 2 (December 1997): 766-787.

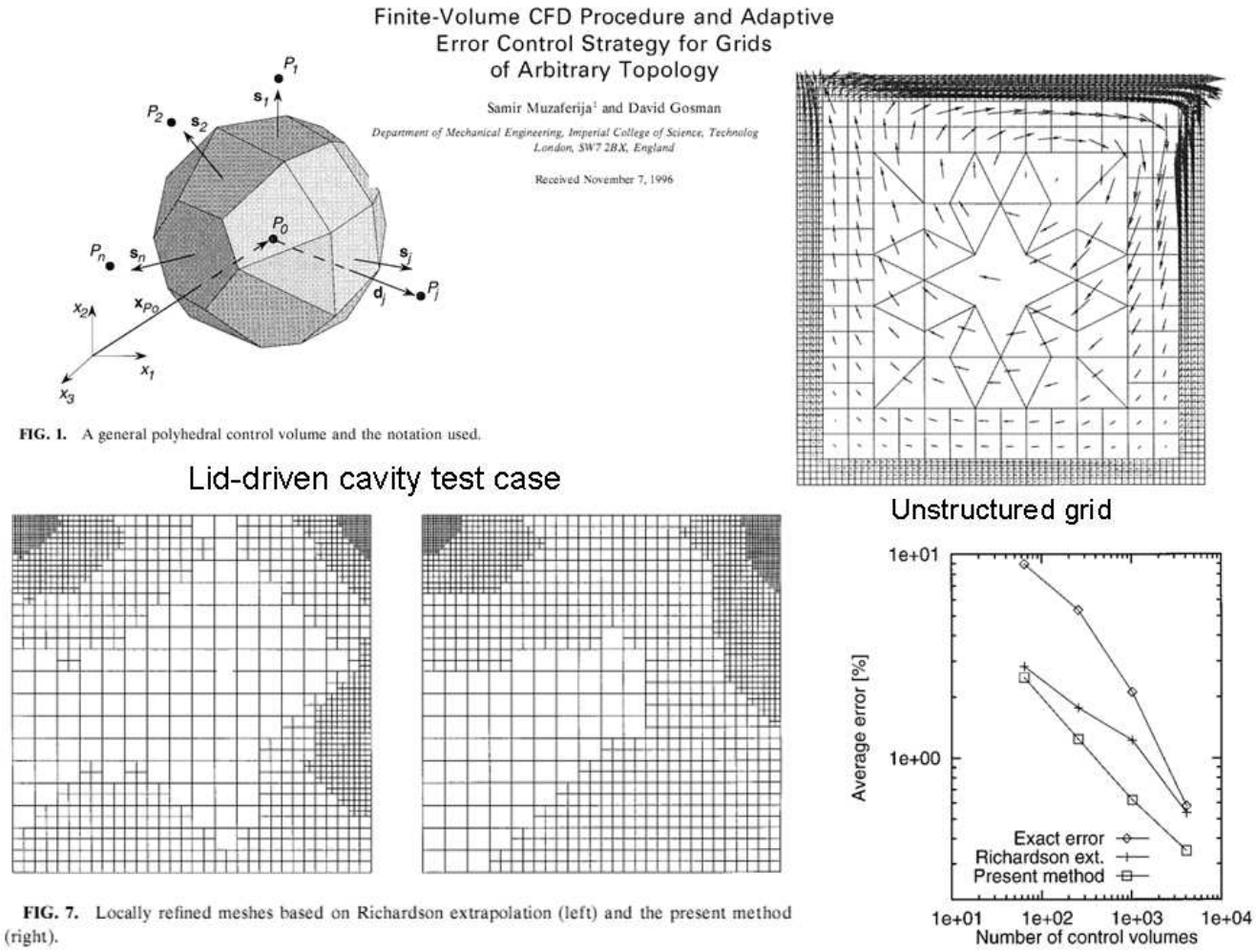


Figure 6:

2 APPENDIX

2.1 Least square method for Gradient Reconstruction on a triangle

1. In 2D consider a triangular cell P and its 3 neighbours, A, B, C.

2. Introduce a linear extrapolation for ϕ from its value ϕ_P at centre P to centre A and equate this to ϕ_A . Do the same for ϕ_B, ϕ_C :

$$\phi_A = \phi_P + (x_{1(A)} - x_{1(P)}) \left[\frac{\partial \phi}{\partial x_1}\right]_P + (x_{2(A)} - x_{2(P)}) \left[\frac{\partial \phi}{\partial x_2}\right]_P \tag{25}$$

$$\phi_B = \phi_P + (x_{1(B)} - x_{1(P)}) \left[\frac{\partial \phi}{\partial x_1}\right]_P + (x_{2(B)} - x_{2(P)}) \left[\frac{\partial \phi}{\partial x_2}\right]_P \tag{26}$$

$$\phi_C = \phi_P + (x_{1(C)} - x_{1(P)}) \left[\frac{\partial \phi}{\partial x_1}\right]_P + (x_{2(C)} - x_{2(P)}) \left[\frac{\partial \phi}{\partial x_2}\right]_P \tag{27}$$

3. Could these relations be used to compute $\left[\frac{\partial \phi}{\partial x_1}\right]_P$ and $\left[\frac{\partial \phi}{\partial x_2}\right]_P$ from ϕ_A, ϕ_B , and ϕ_C ?

We can write the previous equations as a linear system: $\underline{\underline{A}}\ \ \underline{X} = \underline{F}$, with:

$$\underline{X} = \begin{bmatrix} \frac{\partial \phi}{\partial x_1} \\ \frac{\partial \phi}{\partial x_2} \end{bmatrix};\ \underline{F} = \begin{bmatrix} \phi_A - \phi_P \\ \phi_B - \phi_P \\ \phi_C - \phi_P \end{bmatrix};\ \underline{\underline{A}} = \begin{bmatrix} dx_{1(A)} & dx_{2(A)} \\ dx_{1(B)} & dx_{2(B)} \\ dx_{1(C)} & dx_{2(C)} \end{bmatrix};\ dx_{1(PA)} = x_{1(A)} - x_{1(P)}$$

We have N=3 equations for only d=2 unknowns, $\left[\frac{\partial \phi}{\partial x_1}\right]_P$ and $\left[\frac{\partial \phi}{\partial x_2}\right]_P$. In general $\underline{\underline{A}}$ has N lines and d columns, where N is the number of neighbouring cells sharing an interface with cell P, and d is = 2 or 3 in 2D and 3D space respectively. For the most common case of hexa cells in 3D (bricks), $N = 6$ and $d = 3$. So there is thus no solution to over-constrained $\underline{\underline{A}}\ \ \underline{X} = \underline{F}$ problem, but a least-squares method can be used.

For this we multiply both sides of the linear system by the transpose $\underline{\underline{A^T}}$:

$$\underline{\underline{A^T}}.\underline{\underline{A}}\ \ \underline{X} = \underline{\underline{A^T}}\,\underline{F}$$

The gradient of any variable can then be evaluated by

$$\underline{X} = (\underline{\underline{A^T}}.\underline{\underline{A}})^{-1}\underline{\underline{A^T}}\,\underline{F}$$

The components of gradient in cell P for any variable ϕ are obtained by:

$$[\underline{X}(\phi)]_P = \underline{\underline{M_P}}\,.\underline{F}(\phi_P, \phi_A, \phi_B, ...)$$

$(\underline{\underline{A^T}}.\underline{\underline{A}})^{-1}$ is a $d*d$ symmetric matrix. $\underline{\underline{M}} = (\underline{\underline{A^T}}.\underline{\underline{A}})^{-1}\underline{\underline{A^T}}$ is a $d*N$ matrix that depends only on the mesh geometry. $\underline{\underline{M}}$ only needs to be computed once and stored at the beginning of the simulation. $\underline{F}(\phi_P, \phi_A, \phi_B, ...)$ is simply the differences $\phi_A - \phi_P$ .. .This is a simple direct method, relatively low cost compared to the iterative "Gauss formula" method presented earlier. It is however less accurate on stretched or skewed meshes.

1. **Application:** Develop explicitly the above calculations for a triangle in 2D
2. **Question:** In Muzaferia & Gosman (1997) the least-squares method is introduce in a very brief manner:

" A least squares fit of relation $\phi(x) = \phi(P) + \underline{\nabla \phi_P}(\underline{x} - \underline{x_P})$ to the set of nearest neighbour values is proposed to calculate $\underline{\nabla \phi_P}$

$$\underline{\nabla \phi_P} = \underline{\underline{G}}^{-1}\underline{h}$$

$$g_{kl} = \sum_{J=A,B,C...}^{N} d_J^k d_J^l \quad \text{defines coefficients of matrix } \underline{\underline{G}}$$

$$h_k = \sum_{J=A,B,C...}^{N} (\phi_J - \phi_P) d_J^k \qquad \text{defines vector } \underline{h}$$

where k is the k^{th} Cartesian component of vector $(\underline{x_J} - \underline{x_P})$, l is the l^{th} Cartesian component, and summation is over N nearest neighbours"

How do $\underline{\underline{G}}$ and $\underline{h}$ in the Muzaferia and Gosman paper relate to the current $\underline{\underline{A}}$ and $\underline{F}$ notations?

1. **Answer:**

With $\underline{\underline{G}} = \underline{\underline{A^T}}.\underline{\underline{A}}$ and $\underline{h} = \underline{\underline{A^T}}\,\underline{F}$ we have: $\underline{\nabla \phi_P} = \underline{\underline{G}}^{-1}\underline{h}$

$$g_{kl} = \sum_{J=A,B,C...}^{N} d_J^k d_J^l \quad ; \quad g_{11} = dx_{PA}^2 + dx_{PB}^2 + dx_{PC}^2$$

$$h_k = \sum_{J=A,B,C...}^{N} (\phi_J - \phi_P) d_J^k \quad ; \quad h_1 = dx_{PA}\,(\phi_A - \phi_P) + dx_{PB}\,(\phi_B - \phi_P) + dx_{PC}\,(\phi_C - \phi_P)$$

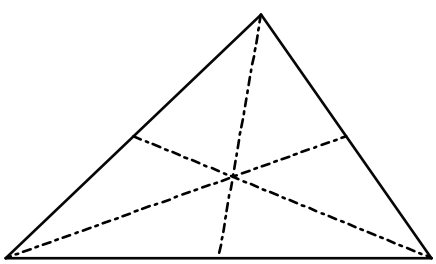
2.1.1 "Center" of a cell?

- *Centroid:*

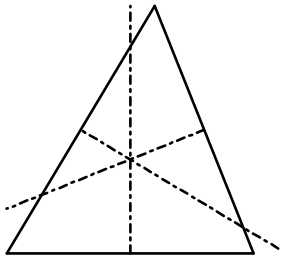
Consider the case of a triangular mesh. The three medians of a triangle intersect in a common point, called the *centroid* of the triangle. A *median* of a triangle is a line segment that has as its end points a vertex of the triangle and the midpoint of the opposite side of the triangle.

- *Circumcentre:*

The three *perpendicular bisectors* of the sides of a triangle intersect in a common point, called the *circumcentre* of the triangle.



Centroid



Circumcentre

For the storage of the variables the ideal choice for the calculation of volume integrals is the centroid, while the best choice for the diffusive fluxes is obviously the circumcentre. The deferred correction is the price we need to pay in order to allow arbitrary shapes of control volumes. For equilateral triangles, (or other equilateral polygons) this correction is not needed since centroid and circumcentre are identical. Since distorted cells are ordinarily only necessary near the boundaries, it would be more efficient to mesh most of the domain with regular polygons, and to mark the irregular polygons in order to apply the deferred correction only on those cells. Unfortunately this feature is not yet present in standard meshing packages.