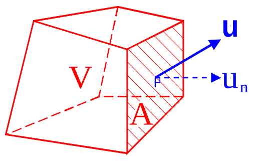

For a conserved physical quantity and an arbitrary control volume,

$$\textit{rate of change within control volume} \;+\; \textit{net outward flux} \;=\; \textit{source within cell} \qquad (1)$$

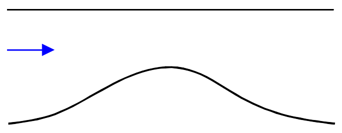


The total flux through a surface consists of *advection* (movement with the flow) and *diffusion* (net transport by random molecular or turbulent fluctuations). If ϕ is the amount of the conserved quantity per unit mass of fluid, then the generic *scalar-transport* (or *advection-diffusion*) *equation* may be written:

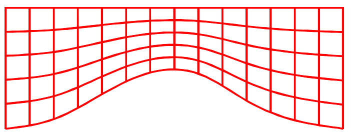
$$\underbrace{\frac{\mathrm{d}}{\mathrm{d}t}(\rho V\phi)}_{\textit{rate of change}} + \sum_{\textit{faces}}\Big(\underbrace{C\phi}_{\textit{advection}} - \underbrace{\Gamma\frac{\partial\phi}{\partial n}A}_{\textit{diffusion}}\Big) = \underbrace{SV}_{\textit{source}} \qquad (2)$$

The *finite-volume method* is a discretisation of (2). This Section focuses on *steady* flow.

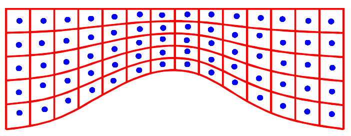
4.1 The Finite-Volume Method



(1) A flow geometry is defined.



(2) The flow domain is decomposed into a *computational mesh* or *grid* – a set of non-overlapping *control volumes* or *cells* – over which the integral equations are to be discretised.

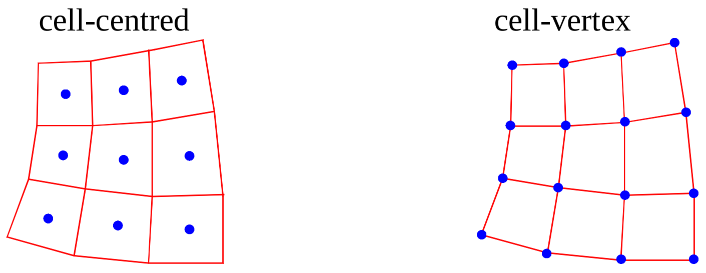


(3) The integral equations are *discretised* – i.e. approximated in terms of values at a set of nodes.

(4) The discretised equations are solved numerically.

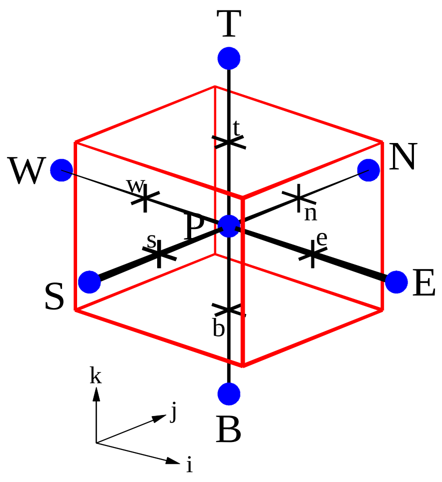
Computational meshes may be *structured* or *unstructured*, *cartesian* or *non-cartesian*.

The commonest configurations are *cell-centred* or *cell-vertex* storage. (In Section 5 it will be seen that it is not necessary to store all variables at the same location).



For simplicity, this course focuses on *structured*, *cartesian* meshes using *cell-centred* storage. In this arrangement the variable of interest, ϕ, is stored at *nodes* at the centres of control volumes. A typical 3-d control volume with its associated storage nodes is shown right. Relative to the cell centre (point P) the coordinate directions are commonly denoted west, east, south, north, bottom, top with:

- lower case w, e, s, n, b, t being used for cell *faces*;
- upper case W, E, S, N, B, T for adjacent *nodes*.



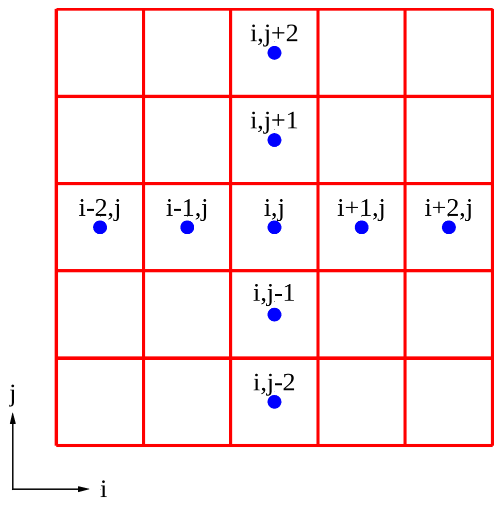
For a *cartesian* mesh these would usually correspond to $\pm x$, $\pm y$, $\pm z$ directions respectively.

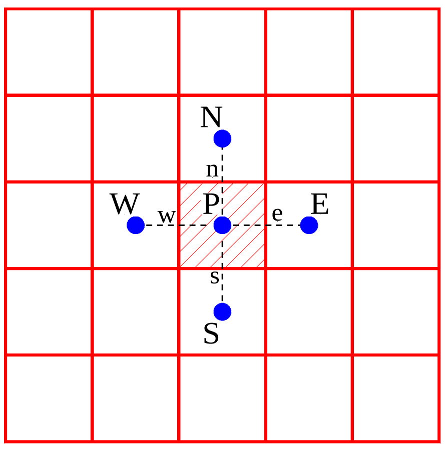
Cell-face areas will be denoted A_w, A_e, A_s, A_n, A_b, A_t. Volumes of cells will be denoted by V.

In 2 dimensions one can think of a single layer of cells having unit depth ($\Delta z = 1$).

For a *structured* mesh one may use, interchangeably, *ijk* indices for the nodes. Then,

$$\phi_P \equiv \phi_{ijk}, \qquad \phi_E \equiv \phi_{i+1\,jk}\,, \qquad \text{etc.}$$

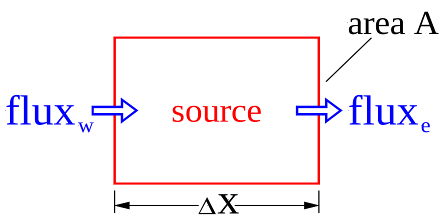




4.2 The One-Dimensional Advection-Diffusion Equation

Consider first the steady-state, 1-d advection-diffusion equation. This is worthwhile because:

- it greatly simplifies the analysis;
- it permits a hand solution of the discretised equations;
- subsequent generalisation to 2 and 3 dimensions is straightforward;
- in practice, discretisation of fluxes (advection and diffusion) is generally carried out coordinatewise, i.e. along *i*, *j* and *k* lines separately;
- many important theoretical problems are 1-d.



Conservation for one control-volume gives

$$flux_e - flux_w = source \tag{3}$$

where "*flux*" means the rate of transport through a cell face.

If ϕ is the amount per unit mass, then the total flux is the sum of advective and diffusive fluxes, where:

advective flux $= \rho u A \phi$

diffusive flux $= -\Gamma \dfrac{d\phi}{dx} A$

The advection-diffusion equation for ϕ is then:

$$\Big[\underbrace{\rho u A \phi}_{advection} \quad \underbrace{-\Gamma A \frac{d\phi}{dx}}_{diffusion} \Big]_w^e = \underbrace{S \Delta x}_{source} \tag{4}$$

(ρ = density, u = velocity, A = cross-sectional area, S = source per unit length).

Dividing by Δx and taking the limit as $\Delta x \to 0$ gives a corresponding differential equation:

$$\frac{d}{dx}(\rho u A \phi - \Gamma A \frac{d\phi}{dx}) = S \tag{5}$$

Notes.

- This system is quasi-1-d in the sense that the cross-sectional area A may vary. To solve a truly 1-d problem, simply set $A = 1$. The differential equation is then

$$\frac{d}{dx}(\rho u \phi - \Gamma \frac{d\phi}{dx}) = S$$

- In this instance, ρ, u, Γ and S are assumed to be known. In the general CFD problem, u is itself the subject of a transport equation, which must be solved simultaneously.

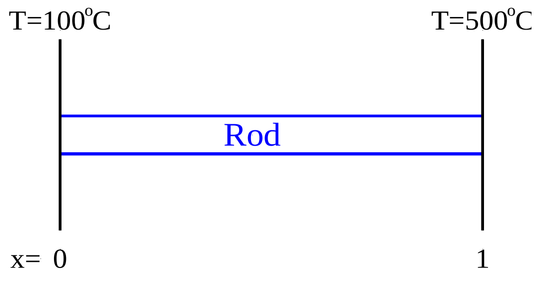
4.3 Classroom Examples

Example 1. (Pure Diffusion)

An insulated rod of length 1 m and 1 cm × 1 cm square cross-section has its end temperatures fixed at 100 ºC and 500 ºC as shown. The heat flux across any section of area A is given by

$$-k\frac{\mathrm{d}T}{\mathrm{d}x}A$$

where the conductivity $k = 1000$ W m^{-1} K^{-1}.



(a) Divide the rod into 5 control sections with nodes at the centre of each section and carry out a finite-volume analysis to find the temperature along the rod.

(b) Write down a differential equation for the temperature distribution along the rod.

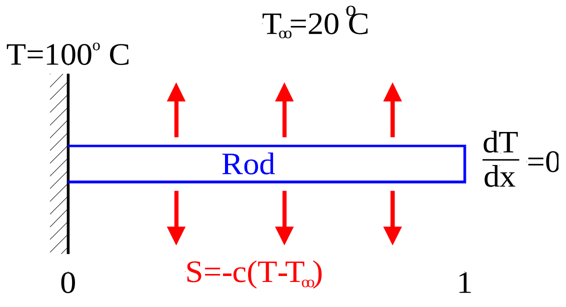
(c) Solve the governing differential equation analytically and compare with (a).

Example 2. (Diffusion + Sources)

The rod configuration is now changed such that the right-hand temperature is no longer fixed and the rod is allowed to cool along its length at a rate proportional to its difference from the ambient temperature (Newton's law of cooling); i.e. the heat loss per unit length is:

$$S = -c(T - T_\infty)$$

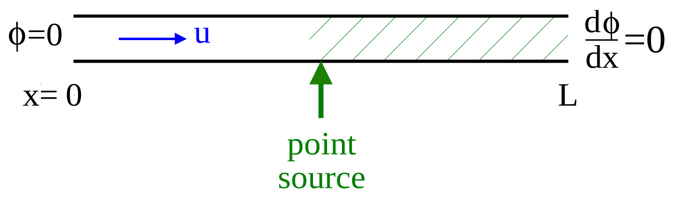
where the ambient temperature $T_\infty = 20$ ºC and the coefficient $c = 2.5$ W m^{-1} K^{-1}.



Repeat parts (a) – (c) of Example 1.

Example 3. (Advection + Diffusion + Sources)

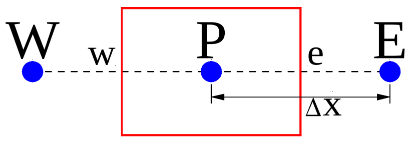
A pipe of cross-section $A = 0.01$ m^2 and length $L = 1$ m carries water (density $\rho = 1000$ kg m^{-3}) at velocity $u = 0.1$ m s^{-1}.



A faulty valve introduces a reactive chemical into the pipe half-way along its length at a rate of 0.01 kg s^{-1}. The diffusivity of the chemical in water is $\Gamma = 0.1$ kg m^{-1} s^{-1}. The chemical is subsequently broken down at a rate proportional to its concentration ϕ (mass of chemical per unit mass of water), this rate amounting to $-\gamma\phi$ per metre, where $\gamma = 0.5$ kg m^{-1} s^{-1}.

Assuming that the downstream boundary condition can be approximated by $\mathrm{d}\phi/\mathrm{d}x = 0$, set up a finite-volume calculation with 7 cells to estimate the concentration along the pipe using (a) central and (b) upwind differencing schemes for advection.

4.4 Discretising Diffusion

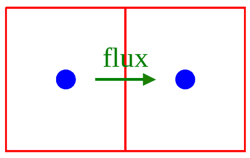


Gradient diffusion is almost invariably discretised by *central-differencing*:

$$-\Gamma A \frac{\mathrm{d}\phi}{\mathrm{d}x}\bigg|_e \quad \to \quad -(\Gamma A)_e \left(\frac{\phi_E - \phi_P}{\Delta x}\right) \tag{6}$$

Notes.

- In the finite-volume method, fluxes are required at cell *faces*, not nodes.
- This approximation for $(\mathrm{d}\phi/\mathrm{d}x)_e$ is *second-order accurate* in Δx (see later).
- If the diffusivity Γ varies then its value on the cell face must be obtained by interpolation.
- Equal weighting is applied to the two nodes on either side of the cell face. This is consistent with diffusion acting equally in all directions. Later on, we shall see that this contrasts strongly with advection, which has a definite directional bias.
- A similar expression must be used for the west face. This is necessary for *conservation*: flux *out* of one cell equals flux *into* the adjacent cell. This *conservative* property is where finite-volume methods score most strongly over finite-difference and finite-element methods.



It is common to define a *diffusive transfer coefficient D*. On the east face, for example,

$$D_e \equiv \left(\frac{\Gamma A}{\Delta x}\right)_e \tag{7}$$

Then,

$$-\Gamma \frac{\mathrm{d}\phi}{\mathrm{d}x} A\bigg|_e \quad \to \quad -D_e(\phi_E - \phi_P) \tag{8}$$

4.5 Discretising the Source Term

When the source is proportional to the amount of fluid, the total source strength for the cell is simply

$$(\textit{source per unit volume}) \times (\textit{volume})$$
$$= S \times V$$

S is the average source density (usually taken as the value at cell centre); V is the cell volume.

S often depends on the actual solution ϕ. For example, if ϕ is temperature, then Newton's law of cooling says that the rate of heat loss is proportional to the temperature difference over the surrounds, or $S = -c(\phi - \phi_\infty)$. The source term is conveniently decomposed into solution-independent and solution-dependent parts as

$$\textit{source} = b_P + s_P\phi_P \;, \qquad s_P \le 0 \tag{9}$$

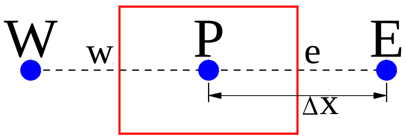
The reason for this particular linearisation will become apparent later.

4.6 Assembling the Algebraic Equations

When there is no flow (u = 0) the steady-state diffusion problem discretises as follows.

$$flux_e - flux_w = source$$

$$\Rightarrow \qquad -D_e(\phi_E - \phi_P) + D_w(\phi_P - \phi_W) = b_P + s_P\phi_p$$



Multiples of ϕ_P, ϕ_E and ϕ_W can be collected together to give

$$(D_E + D_W - s_P)\phi_P - D_e\phi_E - D_w\phi_W = b_P$$

or, in the notation that will be used hereafter:

$$\boxed{a_P\phi_P - \sum_{\substack{adjacent\\nodes}} a_F\phi_F = b_P} \qquad (10)$$

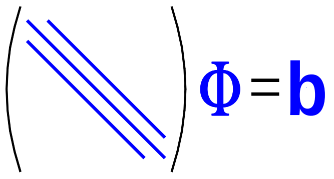
For pure diffusion the matrix coefficients are given by:

$$\begin{aligned} a_E &= D_e \\ a_W &= D_w \\ a_P &= a_E + a_W - s_P \end{aligned} \qquad (11)$$

(10) is a canonical form for the discretised scalar-transport equation. Each variable whose transport equation is to be solved will have a discretised equation of this form (with different values of the coefficients).

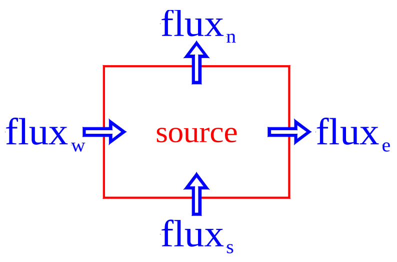
(10) is the discretised equation for one control volume. If the ϕ_P values at the centre of all cells are assembled into a vector, then the set of algebraic equations takes the form

$$\begin{pmatrix} \ddots & \ddots & & & 0 \\ \ddots & \ddots & \ddots & & \\ & -a_W & a_P & -a_E & \\ & & \ddots & \ddots & \ddots \\ 0 & & & \ddots & \ddots \end{pmatrix} \begin{pmatrix} \vdots \\ \vdots \\ \phi_P \\ \vdots \\ \vdots \end{pmatrix} = \begin{pmatrix} \vdots \\ \vdots \\ b_P \\ \vdots \\ \vdots \end{pmatrix} \qquad \text{or}$$



For a 1-d system this is *tri-diagonal*. If the coefficients are constants then it can be solved in one step by the *tri-diagonal matrix algorithm* (see the Appendix). If the coefficients are themselves dependent on the solution, then it must be solved iteratively.

4.7 Extension to 2 and 3 Dimensions



For a multi-dimensional flow the net flux out of a cell can be obtained by summing the *outward* fluxes through all faces.

For quadrilateral elements (in 2d) or hexahedral elements (in 3d) then the net *outward* flux from of a cell is simply the sum of the net fluxes through opposing sides and the general conservation equation may be written:

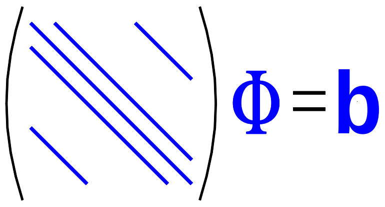
$$(flux_e - flux_w) + (flux_n - flux_s) + (flux_t - flux_b) = source \qquad (12)$$

The discretised equations are still assembled in the same matrix form

$$a_P \phi_P - \sum_F a_F \phi_F = b_P \tag{13}$$

with the summation simply being extended to include the other coordinate directions.

Equation (13) describes the discretisation of the integral conservation equation for one control volume. Combining these for every control volume forms a matrix equation for the set of nodal values $\{\phi_P\}$. For a 2-d flow this gives the banded matrix system shown. Additional outlying bands appear in 3 dimensions.



Thus, (10) or (13) has the same form in 1-, 2- or 3-d problems, but in multiple dimensions the summation is extended to the other (S, N, B, T) nodes and there is a corresponding increase in the number of non-zero diagonals in the assembled matrix equation. This has a considerable impact on the solution of the matrix equation.

Equation (13) formally describes the discretisation for a single control volume in an *unstructured* mesh. However, since the nodes do not have a simple *ijk* indexing pattern, the resulting matrix and the solution method are considerably more complex.

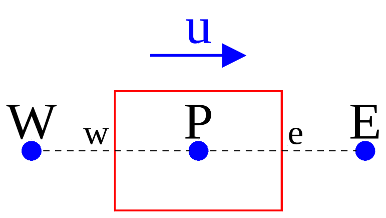
4.8 Advection Schemes (Part I)

Purely diffusive problems ($u = 0$) are of passing interest only. In typical engineering flow problems, the advection term far exceeds the diffusion term because the Reynolds number ($\text{Re} \equiv UL/\nu$ = ratio of inertial forces [mass × acceleration] to viscous forces) is very large.

The steady-state advection-diffusion equation is

$$flux_e - flux_w = source$$

$$\left[\rho u A \phi - \Gamma A \frac{\mathrm{d}\phi}{\mathrm{d}x} \right]_w^e = S\,V$$



Writing

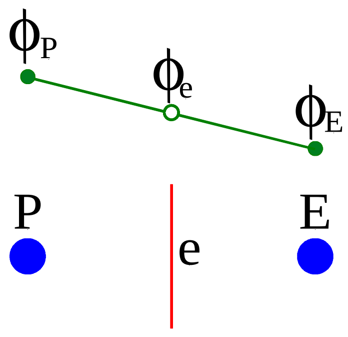
$$C_e = (\rho u A)_e = mass\;\;flux$$

and discretising the diffusion and source terms as before, the equation becomes

$$\underset{advection}{[C_e\phi_e - C_w\phi_w]} \quad - \quad \underset{diffusion}{[D_e(\phi_E - \phi_P) - D_w(\phi_P - \phi_W)]} \quad = \quad \underset{source}{b_P + s_P\phi_P} \tag{14}$$

The problem is … how to specify the **face** values ϕ_e and ϕ_w in terms of the values at adjacent **nodes**. The method of specifying these face values is called an *advection scheme* or *advection-differencing scheme*.

4.8.1 Central Differencing



At first sight the "obvious" answer would be linear interpolation (*central differencing*):

$$\phi_e = \tfrac{1}{2}(\phi_P + \phi_E)$$

This is second-order accurate for ϕ_e in terms of the mesh size Δx (see later). Substituting into (14) and rearranging gives a system of simultaneous equations – one for each control volume – of the form

$$a_P\phi_P - \sum_F a_F\phi_F = b_P$$

where the summation is over adjacent nodes and

$$\begin{aligned} a_E &= D_e - \tfrac{1}{2}C_e \\ a_W &= D_w + \tfrac{1}{2}C_w \\ a_P &= a_E + a_W - s_P + (C_e - C_w) \end{aligned} \qquad (15)$$

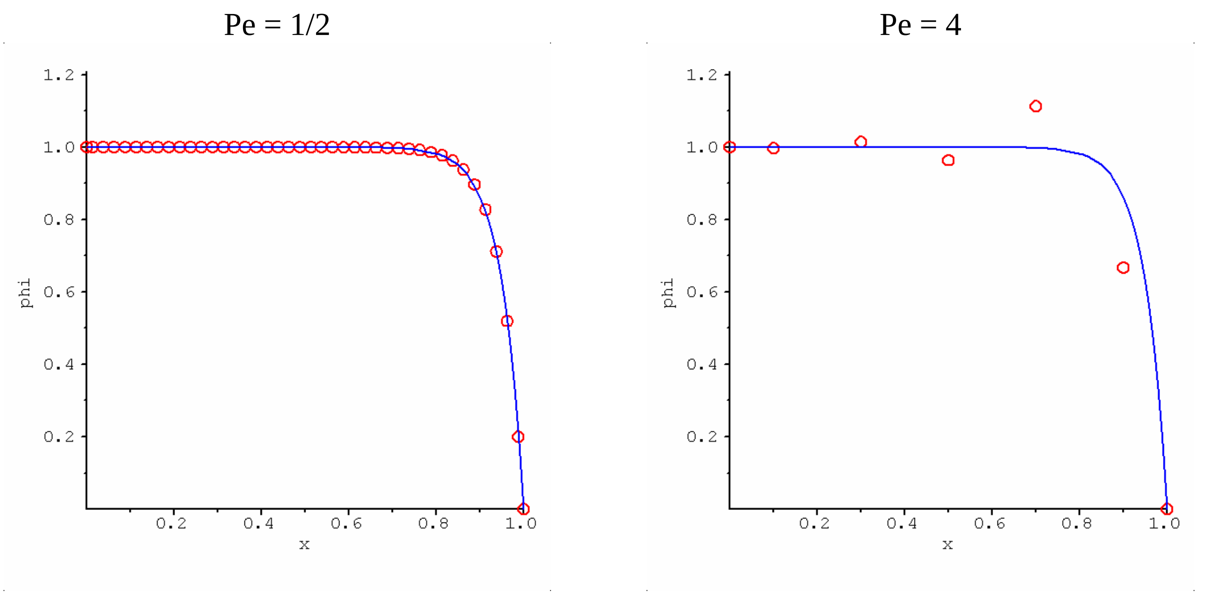
(Note that $C_e - C_w = 0$ to satisfy mass conservation, so that the expression for a_P can be simplified slightly.)

The graphs below show the solution of an advection-diffusion problem with no sources for the two combinations

Pe = 1/2	(advection « diffusion)
Pe = 4	(advection » diffusion)

where the *Peclet number* Pe is defined by

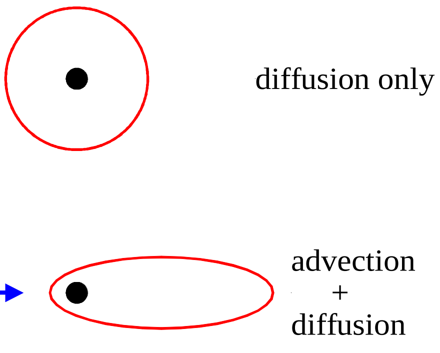
$$\text{Pe} = \frac{C}{D}\left(\text{i.e.}\,\frac{advection}{diffusion}\right) = \frac{\rho u \Delta x}{\Gamma} \qquad (16)$$



In the first instance the solution is very good (consistent with second-order accuracy).

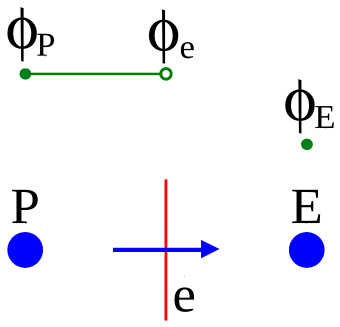
In the second case there are pronounced "wiggles" in what should be a perfectly smooth solution. What is wrong?

Mathematically, when the cell Peclet number Pe is bigger than 2, one of the matrix coefficients becomes negative, meaning that, for example, an *increase* in the corresponding nodal value would lead to a *decrease* in the value at the central node.



Physically, the advection process is *directional*; it transports properties only in the direction of the flow. By contrast, the central-differencing formula assigns *equal weight* to both upwind and downwind nodes.

4.8.2 Upwind Differencing



In *upwind differencing* ϕ_{face} is simply taken as the value at the upwind node; e.g.

$$\phi_e = \begin{cases} \phi_P & (\text{if } u > 0) \\ \phi_E & (\text{if } u < 0) \end{cases}$$

This is only first-order accurate in Δx, but preserves the directional nature of advection. The alternatives are usually summarised as

$$\phi_{face} = \phi_U$$

where subscript U simply denotes the *upwind* node for that face.

With this scheme, the algebraic equation for each control volume takes the canonical form

$$a_P \phi_P - \sum_F a_F \phi_F = b_P$$

where

$$\begin{aligned} a_E &= D_e + \max(0, -C_e) \\ a_W &= D_w + \max(0, C_w) \\ a_P &= a_E + a_W - s_P \end{aligned} \qquad (17)$$

(If the "max" bit confuses you, set $C_e = C_w$ and consider separately the two cases where this is positive or negative.)

When applied to the same advection-diffusion problem as the central-differencing scheme it is found that:

- when $\text{Pe} = \dfrac{C}{D} = \dfrac{\rho u \Delta x}{\Gamma} = \dfrac{1}{2}$ the upwind-differencing scheme is not as accurate as central differencing; this is to be expected from its order of accuracy;
- when $\text{Pe} = 4$ the upwind-differencing solution is not particularly accurate, but the "wiggles" have disappeared.

So there appears to be a pay-off – accuracy versus boundedness (absence of wiggles). The next sections examine the desirable properties of discretisation schemes, the constraints that they impose upon the matrix coefficients and more advanced advection schemes that are both accurate and bounded (wiggle-free).

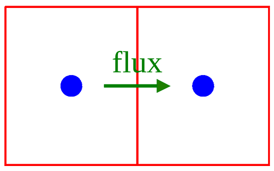
4.9 Discretisation Properties

(i) *Consistent*

The discretised equations are equivalent to the continuum equations in the limit as the grid size tends to zero.

e.g. $\frac{\phi_E - \phi_P}{\Delta x}$ is a consistent approximation for $\frac{\partial \phi}{\partial x}$ (by the definition of a derivative).

(ii) *Conservative*



Achieved by consistent expressions for fluxes through the faces of adjacent control volumes; i.e.

- what goes **out** of one cell must go **into** the adjacent cell;
- fluxes are associated with **faces**, not **nodes**.

This is built into the finite-volume method.

(iii) *Transportive*

Directional influence borne out in an advection scheme. In practice this means a higher weighting to node(s) on the upstream side of a face.

(iv) *Bounded*

In an advection-diffusion problem without sources the solution is bounded by the maximum and minimum values of the flow variable at surrounding nodes.

(v) *Stable*

This determines whether it is possible to obtain a solution – it says nothing about its accuracy. It means that small errors do not grow in the course of the solution procedure.

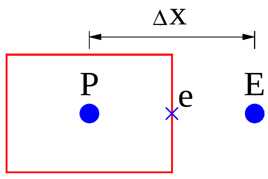
(vi) *Order*

Refers to how fast the truncation error diminishes as the grid size Δ is reduced. If, on a uniform grid of spacing Δ, the error in some numerical scheme is proportional to Δ^n as $\Delta \to 0$ then that scheme is said to be *of order n*.

Order can usually be established by the leading-order error term in a Taylor-series expansion (see below). Note that "error" refers to the theoretical *truncation error* in this expansion and ignores any computational *round-off error* (computers can only store numbers to a finite number of significant figures).

Higher accuracy can be obtained by using more nodes in an approximation – one node permits schemes of at most first-order accuracy, two permit at most second-order accuracy and so on.

The more accurate a scheme then, in principle, the greater the reduction in numerical error as the grid is made finer or, conversely, the less nodes required to resolve the flow to a given accuracy. However, high-order schemes tend to require more computational calculations and often have boundedness problems.



The order of differencing schemes for diffusion and advection can be established by Taylor-series expansions for the nodal values about the cell **face**. e.g. for the nodes either side of the east face:

$$\phi_E = \phi_e \quad + \quad (\frac{\Delta x}{2})\frac{\partial \phi}{\partial x}\bigg|_e \quad + \quad \frac{1}{2!}(\frac{\Delta x}{2})^2 \frac{\partial^2 \phi}{\partial x^2}\bigg|_e \quad + \quad \frac{1}{3!}(\frac{\Delta x}{2})^3 \frac{\partial^3 \phi}{\partial x^3}\bigg|_e \quad + \quad \cdots \qquad (18)(a)$$

$$\phi_P = \phi_e \quad - \quad (\frac{\Delta x}{2})\frac{\partial \phi}{\partial x}\bigg|_e \quad + \quad \frac{1}{2!}(\frac{\Delta x}{2})^2 \frac{\partial^2 \phi}{\partial x^2}\bigg|_e \quad - \quad \frac{1}{3!}(\frac{\Delta x}{2})^3 \frac{\partial^3 \phi}{\partial x^3}\bigg|_e \quad + \quad \cdots \qquad (18)(b)$$

Subtracting (18)(a) – (b) gives:

$$\phi_E - \phi_P = \Delta x \frac{\partial \phi}{\partial x}\bigg|_e \quad + \quad 0 \quad + \quad \frac{1}{24}\Delta x^3 \frac{\partial^3 \phi}{\partial x^3}\bigg|_e \quad + \quad \cdots$$

$$\Rightarrow \qquad \frac{\phi_E - \phi_P}{\Delta x} = \frac{\partial \phi}{\partial x}\bigg|_e + O(\Delta x^2)$$

Hence, as the error term is $O(\Delta x^2)$, $\frac{\phi_E - \phi_P}{\Delta x}$ is a second-order approximation for $\frac{\partial \phi}{\partial x}\bigg|_e$.

Alternatively, adding (18)(a) – (b) gives:

$$\phi_P + \phi_E = 2\phi_e \quad + \quad \frac{1}{4}\Delta x^2 \frac{\partial^2 \phi}{\partial x^2}\bigg|_e \quad + \quad \cdots$$

$$\Rightarrow \qquad \tfrac{1}{2}(\phi_P + \phi_E) = \phi_e + O(\Delta x^2)$$

Again, as the error terms are $O(\Delta x^2)$, the central differencing formula $\tfrac{1}{2}(\phi_P + \phi_E)$ is a second-order approximation for ϕ_e. On the other hand, the Upwind-differencing approximations ϕ_P or ϕ_E (depending on the direction of the flow) are first-order accurate.

It can be shown that truncation errors in the advection terms of:

- odd order lead to *numerical diffusion* – smearing out of discontinuities – e.g. first-order upwind differencing;
- even order lead to *dispersion* – recognised by under- and overshoots or "wiggles"; e.g. second-order central differencing.

Schemes of low-order accuracy – e.g. Upwind – lead to substantial numerical diffusion in 2- and 3-d calculations when the velocity vector is not aligned with the grid lines.

4.10 Constraints on the Matrix Coefficients

The steady-state advection-diffusion equation takes the form

$$\textit{net outward flux} = \textit{source}$$

and leads to algebraic equations for each control volume of the form

$$a_P \phi_P - \sum_F a_F \phi_F = b_P$$

where the summation is over all adjacent nodes. There is one set of matrix coefficients a_P, $\{a_F\} = \{a_W, a_E, a_S, a_N, a_B, a_T\}$, b_P for each internal node. The required properties of discretisation schemes place a number of constraints on the matrix coefficients.

<u>Without Sources</u>

For each control volume *boundedness* requires that, with no sources, the value at each node must lie between the minimum and maximum values at adjacent nodes. For linear schemes (i.e. those whose coefficients do not depend on the solution) this requires:

$a_F \geq 0$ for all F (“*positive coefficients*”)

$$a_P \geq \sum_F a_F$$

(Actually, boundedness alone requires that all coefficients are of the same sign, but in this context it invariably means positive). It is the contravening of the positivity condition on the matrix coefficients that leads the central-differencing scheme to produce “wiggles”.

When there are no sources, the differential equation involves only derivatives of ϕ and hence the discretisation must admit the solution ϕ = constant. This requires that, in the absence of sources, the second of these constraints is actually stricter:

$$a_P = \sum_F a_F \qquad \text{(“\textit{sum of neighbouring coefficients}”)}$$

<u>With Sources</u>

When source terms are added they can be linearised as $b_P + s_P \phi_P$. If the diagonal source term ($s_P\phi_P$) is switched to the LHS then the diagonal coefficient becomes

$$a_P = \sum_F a_F - s_P$$

Numerical stability requires negative feedback; i.e.

$s_P \leq 0$ (“*negative-slope linearisation of the source term*”)

If this condition and the positivity of the a_F is maintained then

$$\frac{\sum |a_F|}{a_P} \leq 1 \qquad \text{(“\textit{diagonal dominance}”)}$$

The last condition is, in fact, a necessary requirement of many solution algorithms.

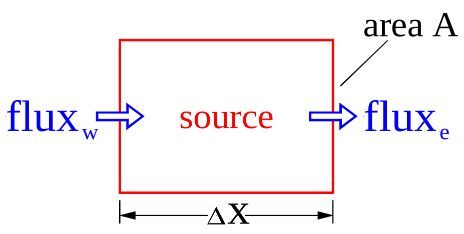
To sum up, the requirements of boundedness and stability place the following constraints on the matrix coefficients:

positive coefficients:	$a_F \geq 0$ for all F
negative-slope linearisation of the source term:	$\textit{source} = b_P + s_P \phi_P, \quad s_P \leq 0$
sum of neighbouring coefficients:	$a_P = \sum_F a_F - s_P$

4.11 Advection Schemes (Part II)

With an understanding of the desirable properties of a differencing scheme it is now possible to examine more advanced advection schemes.

4.11.1 Exponential Scheme



The 1-d advection diffusion equation is

$$\frac{\mathrm{d}}{\mathrm{d}x}(\rho u A\phi - \Gamma A\frac{\mathrm{d}\phi}{\mathrm{d}x}) = source$$

or, equivalently,

$$flux_e - flux_w \equiv \left[\rho u A\phi - \Gamma A\frac{\mathrm{d}\phi}{\mathrm{d}x}\right]_w^e = S\ \Delta x \qquad (19)$$

If there are no sources (S = 0) then the total flux must be constant:

$$flux = \rho u A\phi - \Gamma A\frac{\mathrm{d}\phi}{\mathrm{d}x} = constant$$

This can be solved exactly between values ϕ_P and ϕ_E at adjacent nodes to give

$$flux_e = C\{\phi_P + \frac{\phi_P - \phi_E}{e^{\mathrm{Pe}} - 1}\}$$

where the Peclet number is

$$\mathrm{Pe} = \frac{C}{D} \quad = \frac{\rho u \Delta x}{\Gamma}$$

and

$$C \quad = \rho u A \quad = \text{mass flux}$$
$$D \quad = \frac{\Gamma A}{\Delta x} \quad = \text{diffusive transfer coefficient}$$

.
With a similar expression for the west face, one obtains

$$flux_e - flux_w = a_P\phi_P - \sum a_F\phi_F$$

where

$$a_W = \frac{Ce^{\mathrm{Pe}}}{e^{\mathrm{Pe}} - 1}, \qquad a_E = \frac{C}{e^{\mathrm{Pe}} - 1}, \qquad a_P = a_E + a_W \qquad (20)$$

Assessment

- Conservative by construction.
- Transportive, because if Pe » 1 (advection » diffusion) then, for example,

  $$(flux)_e \approx C\phi_P$$

  depends primarily on the upstream node.
- Bounded: all a_F are positive and a_P is the sum of the neighbouring coefficients.

This scheme – by construction – gives the **exact** solution for **zero sources** and **constant velocity and diffusivity**, but this is something we could have found analytically anyway.

The scheme has never really found favour because:

- the scheme isn't exact when u or Γ vary, or if there are sources, or in 2-d or 3-d flow;
- exponentials are extremely expensive to compute.

4.11.2 Hybrid Scheme (Spalding (1972)

Based on a piecewise linear approximation to the exponential scheme. Amounts to:

- central differencing if $|\text{Pe}| \le 2$
- upwind differencing (with zero diffusion) if $|\text{Pe}| > 2$.

e.g. for $u > 0$:

$$flux_e - flux_w = a_P\phi_P - \sum a_F\phi_F$$

where:

$$\begin{cases} a_W = C_w/2 + D_w\ , & a_E = -C_e/2 + D_e & \text{if } \text{Pe} \equiv C/D \le 2 \\ a_W = C_w, & a_E = 0 & \text{if } \text{Pe} > 2 \end{cases} \qquad (21)$$

$$a_P = a_E + a_W$$

and, for each face (e or w),

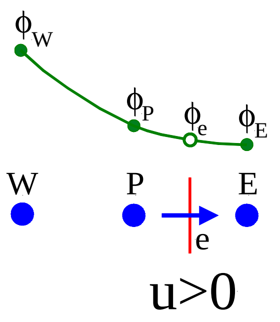
$$C = \rho uA, \qquad D = \frac{\Gamma A}{\Delta x}, \qquad \text{Pe} = \frac{C}{D} = \frac{\rho u \Delta x}{\Gamma}.$$

Assessment
The scheme is conservative, transportive and bounded.

The hybrid scheme remained extremely popular in commercial codes for a long time because it was stable and robust. However, most flows of interest operate in the high-advection/low-diffusion regime, where this scheme amounts to first-order upwinding (with no diffusion). Modern CFD practitioners seek much higher accuracy.

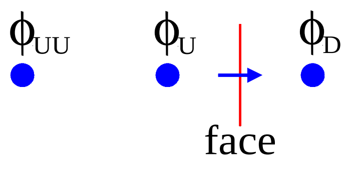
Patankar also developed a *power-law* approximation to the exponential scheme, to overcome the heavy-handed switch-off of diffusion at $\text{Pe} = 2$. However, “powers” are as computationally expensive as exponentials.

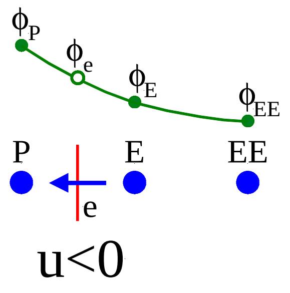
4.11.3 QUICK – QUadratic Interpolation for Convective Kinematics (Leonard, 1979)



Fits a quadratic polynomial through 3 nodes to get 3rd-order accuracy.

For each cell face, QUICK uses the nodes either side of the cell face, plus a further upwind node depending on the direction of the flow; the situation for the east face is shown right.

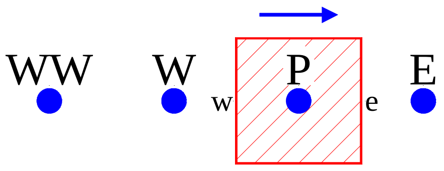




To emphasise the conservation property which associates fluxes with cell *faces*, not nodes, we shall, in future, for all such *three-point* schemes use the notation ϕ_D, ϕ_U and ϕ_{UU} for the Downwind, Upwind and Upwind-Upwind nodes at any particular face:

By fitting a quadratic polynomial to these nodes the QUICK scheme gives:

$$\phi_{face} = -\tfrac{1}{8}\phi_{UU} + \tfrac{3}{4}\phi_U + \tfrac{3}{8}\phi_D \qquad (22)$$



e.g. if $u > 0$, then:

$$\phi_e = -\tfrac{1}{8}\phi_W + \tfrac{3}{4}\phi_P + \tfrac{3}{8}\phi_E$$
$$\phi_w = -\tfrac{1}{8}\phi_{WW} + \tfrac{3}{4}\phi_W + \tfrac{3}{8}\phi_P$$

Hence:

$$\begin{aligned}(flux)_e - (flux)_w &= [C_e\phi_e - D_e(\phi_E - \phi_P)] - [C_w\phi_w - D_w(\phi_P - \phi_W)] \\ &= a_P\phi_P - a_E\phi_E - a_W\phi_W - a_{WW}\phi_{WW}\end{aligned}$$

where

$$a_{WW} = -\tfrac{1}{8}C_w,$$
$$a_W = \tfrac{3}{4}C_w + \tfrac{1}{8}C_e + D_w\,,$$
$$a_E = -\tfrac{3}{8}C_e + D_e\,,$$
$$a_P = a_{WW} + a_W + a_E + (C_e - C_w)$$

Assessment

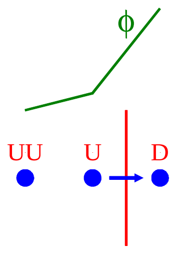
- 3rd-order accurate.
- Conservative by construction.
- Transportive, because upwind bias is built into selection of the third node.
- Not bounded; (for example, if $u > 0$ then a_E is negative).

Despite boundedness not being guaranteed (which can be a severe problem in turbulent flows, where k and ε are required to be positive – see later) the high-order accuracy of the QUICK scheme make it popular and widely-used.

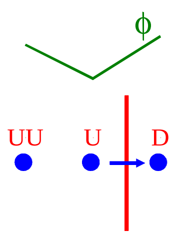
4.11.4 Flux-Limited (Monotonic) Schemes

Hitherto we have only seen schemes where the matrix coefficients a_F are constants (i.e. independent of the solution ϕ). It can be proved that the only unconditionally-bounded scheme of this type is Upwind differencing – and this is merely first-order accurate. Schemes, such as QUICK, which rely on fitting a higher-order polynomial through several points, are prone to generate cell-face values which lie outside the interpolating values ϕ_D, ϕ_U, ϕ_{UU}. To prevent this, modern schemes employ *solution-dependent limiters*, which enforce boundedness whilst trying to retain high-order accuracy wherever possible.

For three-point schemes, ϕ is said to be *monotonic increasing* or *monotonic decreasing* if $\phi_{UU} < \phi_U < \phi_D$ or $\phi_{UU} > \phi_U > \phi_D$ respectively. A necessary condition for boundedness is that the schemes must default to first-order upwinding (i.e. $\phi_{face} = \phi_U$) if ϕ is not locally monotonic (either increasing or decreasing).



monotonic



non-monotonic

Two such schemes used in in-house software are the following. In both cases, monotonic variation in ϕ may be gauged by whether the changes in ϕ between successive nodes (ϕ_{UU} and ϕ_U, ϕ_U and

ϕ_D) have the same sign; i.e.

$$monotone \Leftrightarrow (\phi_D - \phi_U)(\phi_U - \phi_{UU}) > 0$$

Where monotone, the fraction of the total variation which occurs between UU and U nodes is

$$r = \frac{\phi_U - \phi_{UU}}{\phi_D - \phi_U}$$

UMIST scheme (Upstream Monotonic Interpolation for Scalar Transport – Lien and Leschziner, 1993). This is a limited variant of QUICK which is third-order accurate where monotone:

$$\phi_{face} = \begin{cases} \phi_U + (\phi_D - \phi_U)\min(1, r, \tfrac{1}{8} + \tfrac{3}{8}r, \tfrac{3}{8} + \tfrac{1}{8}r) & \text{if monotone} \\ \phi_U & \text{otherwise} \end{cases}$$

Harmonic scheme (Van Leer, 1974)
Second-order accurate where monotone.

$$\phi_{face} = \begin{cases} \phi_U + \dfrac{(\phi_D - \phi_U)(\phi_U - \phi_{UU})}{(\phi_D - \phi_{UU})} & \text{if monotone} \\ \phi_U & \text{otherwise} \end{cases}$$

The choice of these two examples is (obviously!) parochial. Many other equally-good schemes exist. The important points about these schemes are that they are (a) bounded, but (b) non-linear (i.e. the resulting matrix coefficients depend on, and hence change with, the solution ϕ). The last property means that an **iterative** numerical solution is inevitable.

4.12 Implementation of Higher-Order Advection Schemes

The general steady-state scalar transport equation is

$$\sum_{faces} (\underbrace{C\phi}_{advection} - \underbrace{\Gamma \frac{\partial \phi}{\partial n} A}_{diffusion}) = \underbrace{SV}_{source} \tag{23}$$

If C and $D = \Gamma A/\Delta$ are the **outward** mass flux and diffusion transport coefficient on each cell face, then, with the standard discretisation for diffusion and sources, (23) becomes

$$\sum_{faces} [C\phi_{face} + D(\phi_P - \phi_F)] = b_P + s_P \phi_P$$

where F denotes an adjacent node and P is the index of the cell-centre node. Since $\sum C = 0$ (by mass conservation), it is convenient to subtract $\sum C\phi_P$ from both sides:

$$\sum_{faces} [C(\phi_{face} - \phi_P) + D(\phi_P - \phi_F)] = b_P + s_P \phi_P \tag{24}$$

An *advection scheme* (Upwind, Central, QUICK, …) is needed to specify the cell-face value ϕ_{face}. Because many matrix-solution algorithms require positive coefficients, it is common practice to separate ϕ_{face} into the *upwind* value ϕ_U (because this is guaranteed to produce positive coefficients) and the *departure* from upwind differencing, $\phi_{face} - \phi_U$; i.e.:

$$\begin{aligned} C(\phi_{face}-\phi_P) &= \underbrace{C(\phi_U-\phi_P)}_{upwind} \qquad +\underbrace{C(\phi_{face}-\phi_U)}_{correction} \\ &= \max(-C,0)(\phi_P-\phi_F) \quad +C(\phi_{face}-\phi_U) \end{aligned}$$

The first part gives rise to a positive matrix coefficient a_F, whilst the latter part is transferred to the RHS (source side) of the equation as what is called a *deferred correction*:

$$\sum_F a_F(\phi_P-\phi_F) = b_P + s_P\phi_P \underbrace{-\sum_{faces} C(\phi_{face}-\phi_U)}_{deferred\ correction}$$

Then

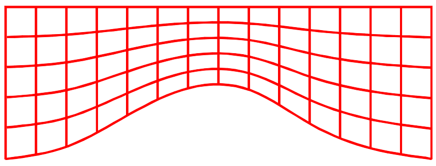
$$a_P\phi_P - \sum a_F\phi_F = \hat{b}_P \tag{25}$$

where

$$a_F = \max(-C,0)+D$$
$$a_P = \sum a_F - s_P$$
$$\hat{b}_P = b_P - \sum C(\phi_{face}-\phi_U)$$

The deferred correction has to be calculated from the current value of ϕ and is treated explicitly (i.e. held constant during an iteration).

4.13 Curvilinear Meshes



Non-cartesian coordinate systems are called *curvilinear*. The coordinates are denoted (ξ,η,ζ) or (ξ_i) and the direction of the coordinate lines varies with position in space.

Coordinate systems in which coordinate lines cross at right angles are termed *orthogonal*. Examples other than cartesian include cylindrical and spherical polar coordinates.

For a cartesian mesh the flux through the east face is given by:

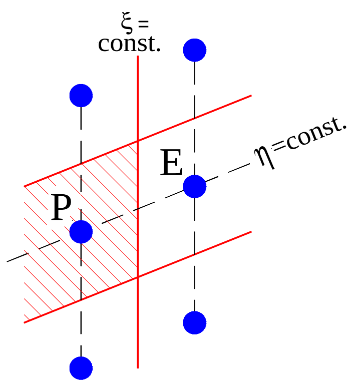
$$flux_e = (\rho uA)\phi - \Gamma\frac{\partial\phi}{\partial x}A \quad = C_e\phi_e - D_e(\phi_E-\phi_P) \tag{26}$$

where

$$\begin{aligned} C_e &= (\rho uA)_e &&= \text{mass flux} \\ D_e &= \left(\frac{\Gamma A}{\Delta x}\right)_e &&= \text{diffusion transfer coefficient} \end{aligned} \tag{27}$$

with similar expressions for the other faces. Orthogonal meshes can be treated in the same way: the direction normal to a cell face coincides with a coordinate line and fluxes can be discretised in a "coordinate-wise" fashion.

However, most curvilinear systems are *non-orthogonal*, with coordinate lines not crossing at right angles. The direction normal to one face is not necessarily along a coordinate line, and hence the diffusive flux, which depends on $\partial\phi/\partial n$, cannot be approximated solely in terms of the nodes either side of the face. For example, in 2 dimensions, if the east face (ξ = constant) of a control volume coincides with x = constant, the normal derivative is



$$\frac{\partial\phi}{\partial x}=\frac{\partial\phi}{\partial\xi}\frac{\partial\xi}{\partial x}+\frac{\partial\phi}{\partial\eta}\frac{\partial\eta}{\partial x}$$

The "diagonal derivative" $\partial\phi/\partial\xi$ can be discretised as $(\phi_E-\phi_P)/\Delta\xi$, but the derivative parallel to the cell face, $\partial\phi/\partial\eta$, depends on the values at nodes other E and P.

In general, non-orthogonal meshes require:

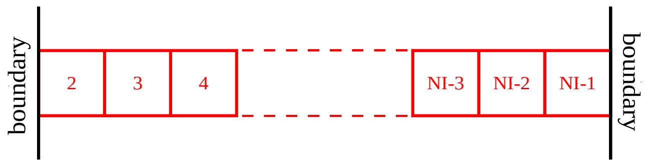
- the "off-diagonal" components of the diffusive flux (that containing $\partial\phi/\partial\eta$ in the above example) to be transferred to the source side of the equation and treated explicitly;
- the storage of all *metric* components $\partial\xi/\partial x$, $\partial\eta/\partial y$, … (9 components in 3d).

Thus, non-orthogonal meshes are inherently more computationally-intensive. However, their added geometric flexibility makes them desirable in general-purpose solvers,

4.14 Boundary Conditions

The most common types of boundary condition are:

- ϕ specified (*Dirichlet* boundary conditions);
  e.g. $u = 0$ at a wall, or temperature fixed at some surface;
- $\partial\phi/\partial n$ specified (*Neumann* boundary conditions).
  e.g. $\partial\phi/\partial n = 0$ on a symmetry plane, or at an outflow boundary.



In the finite-volume method, both types of boundary condition can be implemented by transferring the boundary flux to the source term.

For a cell abutting a boundary:

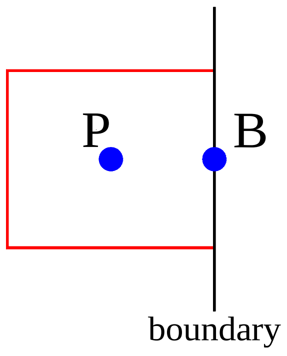
$$\sum_{\substack{not\\boundary}} flux + flux_{boundary} = sources$$

$$\Downarrow$$

$$a_P\phi_P-\sum_{\substack{not\\boundary}} a_F\phi_F = b_P - flux_{boundary}$$

Thus, there are two modifications:

- the a_F coefficient in the direction of the boundary is set to zero;
- the boundary flux is subtracted from the source terms.



If *flux*(ϕ) is specified on the boundary, then this is immediate. If ϕ itself is fixed on the boundary then:

$$flux(\phi)=-\Gamma A\left(\frac{\phi_B-\phi_P}{\Delta}\right)=-D_B(\phi_B-\phi_P)$$

To subtract this flux from the source term requires a simple change of coefficients:

$$b_P\to b_P+D_B\phi_B\ ,\qquad s_P\to s_P-D_B \tag{28}$$

4.15 Solution of the Algebraic Equations

The discretisation of a single scalar transport equation over a single control volume produces an algebraic equation of the form

$$a_P \phi_P - \sum a_F \phi_F = b_P$$

where the summation is over adjacent nodes. Combining the equations for all control volumes produces a set of simultaneous equations, i.e. a matrix equation

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

where $\boldsymbol{\phi}$ is the vector of nodal values, $\{\phi_P\}$, and **A** is *sparse* (i.e. has only a few non-zero diagonals). Many algebraic methods have been used to tackle this problem; only a few are mentioned below.

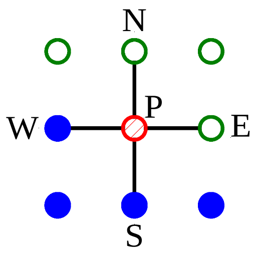
4.15.1 Matrix Solution Algorithms

Gaussian Elimination

This is a direct (i.e. not iterative) method. It consists of a sequence of row operations to obtain zeros below the diagonal ("upper-triangular" matrix).

Inefficient: tends to fill in the sparse matrix **A**.
OK for small hand calculations, but not recommended for large systems of equations.

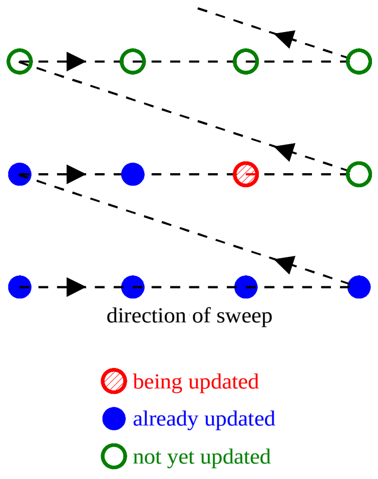
Gauss-Seidel



Iterative update scheme:

$$\phi_P = \frac{1}{a_P}(b_P + \sum a_F \phi_F^*)$$

through successive control volumes; (the asterisk * denotes a "latest" value).

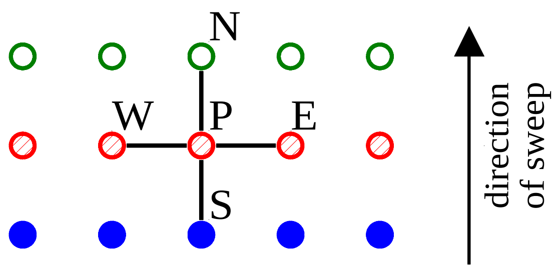


Gauss-Seidel is simple to code and is often used for unstructured grids. However, it tends to converge slowly and may require substantial under-relaxation (see below).

Line-Iterative Procedures ("Line Gauss-Seidel")

Along any one coordinate line, the system is tri-diagonal; e.g. in the *i*-direction:

$$-a_W \phi_W + a_P \phi_P - a_E \phi_E = b_P - \sum_{\substack{not\\W,E}} a_F \phi_F^*$$



and hence *a whole line can be updated at one go* by the tri-diagonal matrix algorithm. This means that information can propagate right across the domain in one iteration, rather than (as in Gauss-Seidel) one node at a time. A typical single iteration would consist of applying the update for each successive *i* line, then for each successive *j* line, then for each successive *k* line (in 3d).

This is probably the most popular method for block-structured grids, and is the basis of most of our in-house research codes.

4.15.2 Convergence Criteria

In all iterative methods the iteration is stopped when the residual error becomes less than some small, pre-defined tolerance. "Residual error" is a sum over the errors of all control volumes; e.g.

sum of absolute residuals: $Error = \sum_{cells} |res|$

root-mean-square (rms) error: $Error = \sqrt{\frac{1}{N}\sum_{cells}(res)^2}$

where the *residual* is the error in satisfying the algebraic equations for any one cell:

$$res = a_P\phi_P - \sum_F a_F\phi_F - b_P$$

4.15.3 Under-Relaxation

If the algebraic methods described above are applied to the non-linear, coupled set of fluid-dynamical variables ($\phi = u, v, w, \ldots$) then the change in variables at each iteration may be large enough to cause numerical instability. To overcome this, *under-relaxation* reduces the change at each iteration (without affecting the final, converged, solution).

For the equation set

$$a_P\phi_P - \sum a_F\phi_F = b_P$$

the iterative update can be written

$$\phi_P = \phi_P^{old} + \Delta\phi_P\ , \qquad \Delta\phi_P = \frac{\sum a_F\phi_F + b_P}{a_P} - \phi_P^{old}$$

If, instead, only a fraction $\alpha\Delta\phi_P$ of the update is applied then

$$\phi_P = \phi_P^{old} + \alpha\Delta\phi_P = \alpha\left(\frac{\sum a_F\phi_F + b_P}{a_P}\right) + (1-\alpha)\phi_P^{old}$$

which can be rearranged as

$$a_P\phi_P - \sum \alpha a_F\phi_F = \alpha b_P + (1-\alpha)a_P\phi_P^{old}$$

Hence, under-relaxation can be achieved by a simple change of coefficients:

$$\begin{aligned} a_F &\to \alpha a_F \\ b_P &\to \alpha b_P + (1-\alpha)a_P\phi_P^{old} \end{aligned} \qquad (29)$$

Note that this also makes the matrix equations more diagonally dominant (i.e. $\frac{\sum a_F}{a_P}$ smaller).

4.16 Summary

- The integral conservation law for a scalar ϕ takes the form
  $$\textit{rate of change} \quad + \quad \textit{net outward flux} \quad = \quad \textit{source}$$

- "*flux*" ≡ rate of transport through a surface and consists of:
  advection – movement with the flow (sometimes called *convection*);
  diffusion – net movement by random molecular or turbulent fluctuations.

- Discretisation of the scalar-transport equation yields an algebraic equation of form
  $$a_P\phi_P - \sum_F a_F\phi_F = b_P$$
  for each control volume, where the summation is over adjacent nodes.

- The collection of these simultaneous equations yields a matrix equation with *limited bandwidth* (i.e. few non-zero diagonals), typically solved by iterative methods such as Gauss-Seidel or line-Gauss-Seidel.

- Source terms are linearised as
  $$b_P + s_P\phi_P \ , \quad s_P \le 0 \, .$$

- Diffusive fluxes are usually discretised by central differencing; e.g.
  $$-\Gamma\frac{\partial\phi}{\partial x}A \quad \to \quad -\frac{\Gamma A}{\Delta x}(\phi_F - \phi_P)$$

- *Advection schemes* are means of approximating ϕ on cell faces in order to compute advective fluxes. They include Upwind, Central, Exponential, Hybrid, QUICK, and flux-limited schemes.

- General principles desired in a discretisation scheme:
  consistency
  conservativeness
  boundedness
  stability
  transportiveness
  accuracy

- The above conditions impose certain constraints on the matrix equations:
  $$a_F \ge 0 \text{ for all } F$$
  $$s_P \le 0$$
  $$a_P = \sum_F a_F - s_P$$

- To ensure positive coefficients (and implement non-linear schemes), advective fluxes are often decomposed into
  "*Upwind*" + "*deferred correction*"
  with the latter being transferred to the source term and treated explicitly.

- For general curvilinear meshes it is necessary to store the metrics ($\partial\xi_i/\partial x_j$) of the coordinate transform and transfer part of the diffusive flux to the source term.

- Boundary conditions can be implemented by transferring boundary fluxes to the source terms.

- Under-relaxation is usually required to solve coupled and/or non-linear equations.

(**** *MSc course only* ****)

Appendix: Tri-Diagonal Matrix Algorithm

For the system of equations:

$$-a_i\phi_{i-1} + b\phi_i - c_i\phi_{i+1} = s_i \ , \quad i = 1, \ldots, N-1 \qquad\qquad \phi_0 \text{ and } \phi_N \text{ given}$$

<u>Forward pass</u>:

$$P_0 = 0, \quad Q_0 = \phi_0$$

$$P_i = \frac{c_i}{b_i - a_i P_{i-1}}, \qquad Q_i = \frac{s_i + a_i Q_{i-1}}{b_i - a_i P_{i-1}}, \qquad i = 1, \ldots, N-1$$

<u>Backward pass</u>:

$$\phi_i = P_i\phi_{i+1} + Q_i, \qquad i = N-1, \ldots, 1$$

Guaranteed to converge if $a_i > 0$, $c_i > 0$, $a_i + c_i \le b_i$ for all i.