Computational Fluid Dynamics

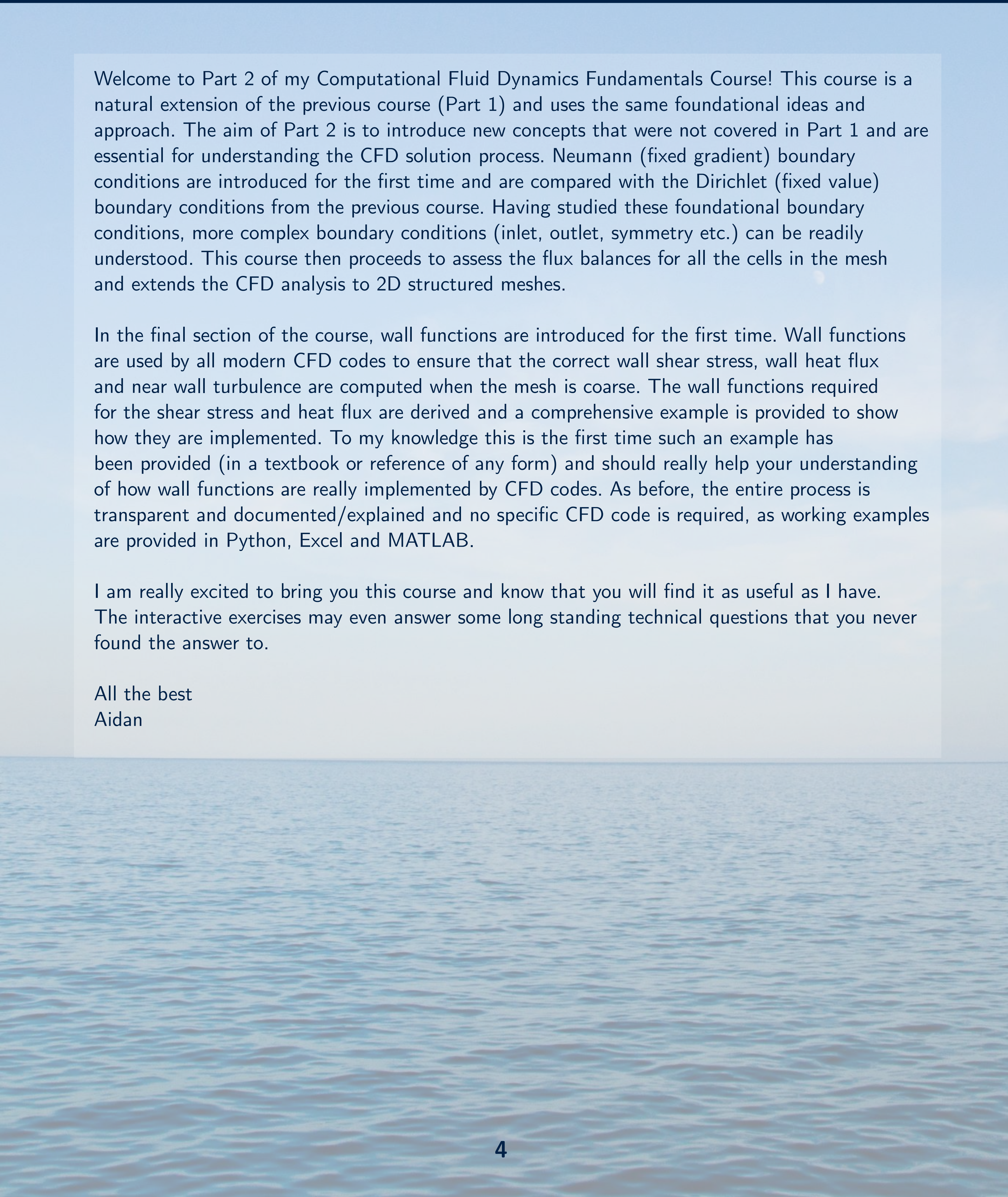
Fundamentals Course 2

Table of Contents

Foreward	4
How To Use This Course	5
Dirichlet and Neumann Boundary Conditions	7
Transport Equations in 2D	24
Wall Functions	42

Disclaimer

Foreward



Welcome to Part 2 of my Computational Fluid Dynamics Fundamentals Course! This course is a natural extension of the previous course (Part 1) and uses the same foundational ideas and approach. The aim of Part 2 is to introduce new concepts that were not covered in Part 1 and are essential for understanding the CFD solution process. Neumann (fixed gradient) boundary conditions are introduced for the first time and are compared with the Dirichlet (fixed value) boundary conditions from the previous course. Having studied these foundational boundary conditions, more complex boundary conditions (inlet, outlet, symmetry etc.) can be readily understood. This course then proceeds to assess the flux balances for all the cells in the mesh and extends the CFD analysis to 2D structured meshes.

In the final section of the course, wall functions are introduced for the first time. Wall functions are used by all modern CFD codes to ensure that the correct wall shear stress, wall heat flux and near wall turbulence are computed when the mesh is coarse. The wall functions required for the shear stress and heat flux are derived and a comprehensive example is provided to show how they are implemented. To my knowledge this is the first time such an example has been provided (in a textbook or reference of any form) and should really help your understanding of how wall functions are really implemented by CFD codes. As before, the entire process is transparent and documented/explained and no specific CFD code is required, as working examples are provided in Python, Excel and MATLAB.

I am really excited to bring you this course and know that you will find it as useful as I have. The interactive exercises may even answer some long standing technical questions that you never found the answer to.

All the best
Aidan

How To Use This Course

This course contains a comprehensive set of equations, explantations and diagrams, which are all contained in this PDF book. As you proceed through Chapter 1, 2 and 3, worked examples are provided. When instructed, it is advied that you open the example code (either in Microsoft Excel, MATLAB or in a suitable text editor/graphical user interface (GUI) for for Python). Here you will be able to examine the code, modify the input variables and run the CFD simuations yourself. This is where the majority of the learning is likely to take place and is highly encouraged for all readers.

All of the exercises can be completed using either the Excel, MATLAB or Python scripts. Use whichever approach is more appealing and straightforward for you to follow. Alternatively, you can use the equations that are provided in the text to write your own code/scripts to solve the equations! The aim of this course is not to develop knowledge of a specific language or CFD code, but to learn and observe the overall process. Hence, it is highly encouraged for you to take your preferred approach.

Chapter 1

Dirichlet and Neumann Boundary Conditions

1 Dirichlet and Neumann Boundary Conditions

In this chapter, the finite volume discretisation of the 1D diffusion equation from the previous course will be revisited. The diffusion equation will be used to introduce *Neumann* (fixed gradient) boundary conditions for the first time and revisit *Dirichlet* (fixed value) boundary conditions. While a general diffusion equation can be used to describe many transported quantities, the diffusion equation for thermal energy (temperature) will be considered in this course, as it is the easiest to physically interpret. Mathematically, the general scalar transport equation for thermal energy (temperature) for incompressible (low speed) flows is:

$$\underbrace{\frac{\partial\left(\rho c_p T\right)}{\partial t}}_{\text{Unsteady}}+\underbrace{\nabla\cdot\left(\rho c_p T\boldsymbol{U}\right)}_{\text{Convection}}=\underbrace{\nabla\cdot\left(k\nabla T\right)}_{\text{Diffusion}}+S \tag{1}$$

where T is the temperature, ρ is the density, c_p is the specific heat capacity at constant pressure, k is the thermal conductivity, $\boldsymbol{U}$ is the velocity vector and S is a source of thermal energy. Some readers may recognise this equation as the transport equation for *enthalpy*. To observe the transport equation for enthalpy (h), define $h = c_pT$ and substitute in to equation 1:

$$\frac{\partial(\rho h)}{\partial t}+\nabla\cdot(\rho\boldsymbol{U}h)=\nabla\cdot(k\nabla T)+S \tag{2}$$

However, this form of the thermal energy equation will not be used in this course, as temperature is an easier variable to interpret and follow than enthalpy.

The focus of this chapter will be the diffusion and source terms in equation 1, since the convection term was considered in the previous course and the unsteady term will be considered in a later course. Neglecting the convection and unsteady terms in equation 1, the diffusion equation for thermal energy is:

$$\cancelto{0}{\frac{\partial\left(\rho c_p T\right)}{\partial t}}+\cancelto{0}{\nabla\cdot\left(\rho c_p T\boldsymbol{U}\right)}=\nabla\cdot\left(k\nabla T\right)+S \tag{3}$$

$$0=\nabla\cdot\left(k\nabla T\right)+S \tag{4}$$

Expanding the gradient (∇) and divergence ($\nabla\cdot$) operators into Cartesian coordinates (x, y, z):

$$0=\frac{\partial}{\partial x}\left(k\frac{\partial T}{\partial x}\right)+\frac{\partial}{\partial y}\left(k\frac{\partial T}{\partial y}\right)+\frac{\partial}{\partial z}\left(k\frac{\partial T}{\partial z}\right)+S \tag{5}$$

To clearly demonstrate how Dirichlet and Neumann boundary conditions are implemented by CFD solvers, the diffusion equation will only be considered in 1D (the x direction) in this chapter. 2D geometries will be considered later in Chapter 2 of this course. In 1D, the diffusion equation for thermal energy (temperature) is:

$$0=\frac{\partial}{\partial x}\left(k\frac{\partial T}{\partial x}\right)+\cancelto{0}{\frac{\partial}{\partial y}\left(k\frac{\partial T}{\partial y}\right)}+\cancelto{0}{\frac{\partial}{\partial z}\left(k\frac{\partial T}{\partial z}\right)}+S \tag{6}$$

$$0=\frac{d}{dx}\left(k\frac{dT}{dx}\right)+S \tag{7}$$

The majority of modern CFD codes use the **finite volume method** to solve the transport equations for the various flow variables (velocity, temperature, pressure etc.). In the previous

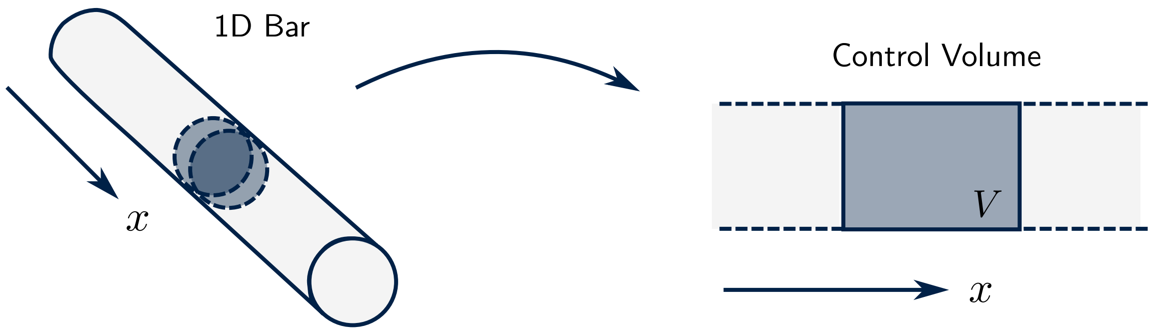


Figure 1: A 1D finite volume of fluid with volume V, which has been isolated from the bar.

course, the finite volume method was reviewed in detail. To avoid repetition, only a concise overview of the finite volume method will be provided here, to ensure that the derivation and solution presented is continuous and coherent. Further detail can be found in the previous course. The first stage in the finite volume method is to integrate the transport equation over a finite-sized control volume (cell). As shown in Figure 1, a 1D cell can be thought of as a slice through a thin 1D bar or rod. Integrating the 1D diffusion equation (equation 7) over this cell:

$$0 = \int_V \left[\frac{d}{dx}\left(k\frac{dT}{dx} \right) + S \right] \mathrm{d}V \tag{8}$$

Integration and addition are commutative operations (is doesn't matter what order they are carried out in). Hence, the finite volume integral can be split into two separate integrals and each one can then be considered in turn.

$$0 = \int_V \left[\frac{d}{dx}\left(k\frac{dT}{dx} \right) \right] \mathrm{d}V + \int_V [S]\, \mathrm{d}V \tag{9}$$

As shown in the previous course, Gauss's divergence theorem is used to replace the volume integral of the diffusion term with a surface integral. Recall (from the previous course) that the divergence theorem for a general vector field $\boldsymbol{B}$ is written as:

$$\int_V (\nabla \cdot \boldsymbol{B})\, \mathrm{d}V = \int_A (\boldsymbol{B} \cdot \hat{\boldsymbol{n}})\, \mathrm{d}A \tag{10}$$

where $\hat{\boldsymbol{n}}$ is the unit normal vector pointing out of the control volume and A is the surface area of the control volume. In 1D Cartesian coordinates, the divergence theorem can be written:

$$\int_V \left(\frac{\partial B_x}{\partial x} + \frac{\partial B_y}{\partial y} + \frac{\partial B_z}{\partial z} \right) \mathrm{d}V = \int_A (B_x n_x + B_y n_y + B_z n_z)\, \mathrm{d}A \tag{11}$$

$$\int_V \left(\frac{\partial B_x}{\partial x} + \cancelto{0}{\frac{\partial B_y}{\partial y}} + \cancelto{0}{\frac{\partial B_z}{\partial z}} \right) \mathrm{d}V = \int_A \left(B_x n_x + \cancelto{0}{B_y n_y} + \cancelto{0}{B_z n_z} \right) \mathrm{d}A \tag{12}$$

$$\int_V \left(\frac{\partial B_x}{\partial x} \right) \mathrm{d}V = \int_A (B_x n_x)\, \mathrm{d}A \tag{13}$$

For the heat diffusion equation $\boldsymbol{B} = k\nabla T$. Hence $B_x = k\ \partial T/\partial x$ in 1D. Applying the 1D divergence theorem to the 1D heat diffusion equation leads to:

$$0 = \int_A \left[k\frac{dT}{dx} n_x \right] \mathrm{d}A + \int_V [S] \mathrm{d}V \tag{14}$$

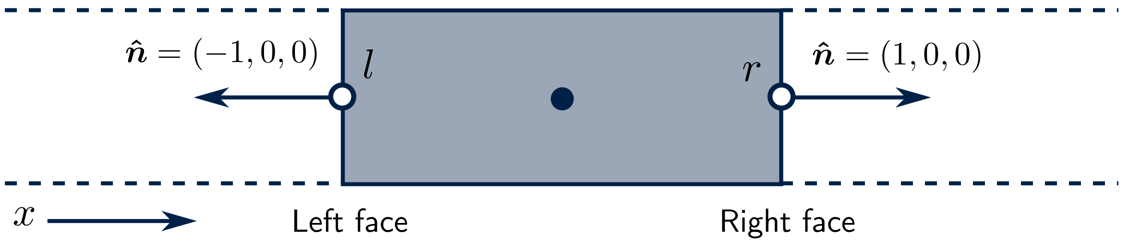

$\hat{\boldsymbol{n}} = (-1, 0, 0)$
l
r
$\hat{\boldsymbol{n}} = (1, 0, 0)$
x
Left face
Right face

Figure 2: A diagram to show the face normal vectors on the left and right faces of the 1D cell. The cell normal vectors always point out of the cell.

The source term is averaged over the control volume, so it can be moved outside the volume integral.

$$0 = \int_A \left[k \frac{dT}{dx} n_x\right] \mathrm{d}A + \overline{S} \int_V \mathrm{d}V \tag{15}$$

$$0 = \int_A \left[k \frac{dT}{dx} n_x\right] \mathrm{d}A + \overline{S}V \tag{16}$$

Recall that the unit normal vector (n_x) always points out of the cell. As shown in Figure 2, on the left face of the cell (l), the unit normal vector is negative. Conversely, the unit normal vector is positive on the right face of the cell (r).

$$0 = \left[kA \frac{dT}{dx}\right]_r - \left[kA \frac{dT}{dx}\right]_l + \overline{S}V \tag{17}$$

This finite volume discretisation is valid for all cells in the mesh. However, different simplifications are required for interior and boundary cells before the equations can be solved.

Interior Cells

Start with the general finite volume discretisation of the 1D diffusion equation.

$$0 = \left[kA \frac{dT}{dx}\right]_r - \left[kA \frac{dT}{dx}\right]_l + \overline{S}V \tag{18}$$

To simplify and solve this equation for the interior cells, the temperature gradient on the cell faces (l and r) needs to be expressed in terms of temperatures at the cell centroids (L, R and P). This is because the unknowns in the matrix equations are the temperatures at the cell centroids and other variables need to be expressed in terms of these temperatures. Hence, the temperature (and other variables) are calculated and stored at the cell centroids, rather than the nodes or faces. This approach is called a **cell-centred** finite volume method and is used by the CFD codes `OpenFOAM`, `ANSYS Fluent` and `Star CCM+` but not `ANSYS CFX` (which uses a **node based** finite volume method).

Throughout this course, the lower case subscripts (l and r) will be used to refer to cell faces, while upper-case subscripts (L, R and P) will be used to refer to cell centroids, as shown in Figure 3. This notation has been chosen because the temperatures at the cell centroids are the unknowns in the system. The necessary simplification of the gradients on the cell faces can be accomplished with linear interpolation, which is often called **central-differencing**. To

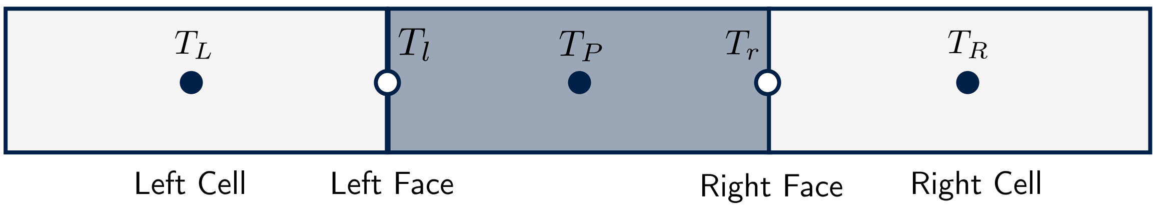


Figure 3: A diagram to show the difference between the temperatures on the cell faces (T_l and T_r) and the temperature at the cell centroids (T_L and T_R).

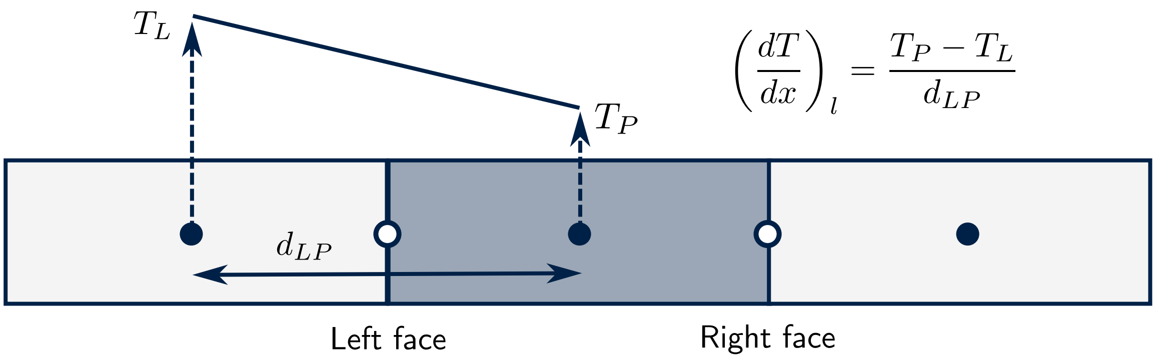


Figure 4: Central differencing (linear interpolation) of the temperature gradient on the left face of the cell using the values at the cell centroids of the interior cell (T_P) and the left cell (T_L).

help understand this simplification, remember that the spatial gradient of temperature can be thought of as:

$$\frac{dT}{dx} \sim \frac{\Delta T}{\Delta x} = \frac{\text{Change in Temperature}}{\text{Distance}} \tag{19}$$

As shown in Figure 4, the temperature gradient on the left face can be expressed using central differencing as:

$$\left(\frac{dT}{dx}\right)_l = \frac{T_P - T_L}{d_{LP}} \tag{20}$$

where d_{LP} is the distance between the cell centroids L and P. In a similar manner, the temperature gradient on the right face can also be expressed using central differencing:

$$\left(\frac{dT}{dx}\right)_r = \frac{T_R - T_P}{d_{PR}} \tag{21}$$

Substitute this simplification into the 1D diffusion equation (equation 18).

$$\left(k_r A_r \frac{T_R - T_P}{d_{PR}}\right) - \left(k_l A_l \frac{T_P - T_L}{d_{LP}}\right) + \overline{S}V = 0 \tag{22}$$

The 1D diffusion equation can now be solved for the temperatures at the cell centroids (T_L, T_R and T_P). To simplify this process, rearrange the equation and collect the terms in terms of temperature of the interior cell (T_P), temperature of the left cell (T_L) and the temperature of the right cell (T_R).

$$T_P\left(\frac{k_l A_l}{d_{LP}} + \frac{k_r A_r}{d_{PR}}\right) = T_L\left(\frac{k_l A_l}{d_{LP}}\right) + T_R\left(\frac{k_r A_r}{d_{PR}}\right) + \overline{S}V \tag{23}$$

At this stage, it is useful to introduce some new notation to simplify the finite volume discretisation. This new notation makes it easier to compare the finite volume discretisation of interior cells, boundary cells and add additional terms to the equation later on. The approach adopted in the previous course was to introduce the notation $D = k/d$. This quantity can be thought of as the diffusive flux of heat per unit area through the cell face and has units of W/m^{2}K. Using this notation, the finite volume discretiation becomes:

$$T_P \left(D_l A_l + D_r A_r\right) = T_L \left(D_l A_l\right) + T_R \left(D_r A_r\right) + \overline{S}V \tag{24}$$

For consistency with other equations that will be introduced later, write the above equation in the following form:

$$\begin{gathered} a_p T_P = a_L T_L + a_R T_R + S_u \\ T_P \underbrace{\left(D_l A_l + D_r A_r + 0\right)}_{a_p} = T_L \underbrace{\left(D_l A_l\right)}_{a_L} + T_R \underbrace{\left(D_r A_r\right)}_{a_R} + \underbrace{\overline{S}V}_{S_u} \end{gathered} \tag{25}$$

This equation is valid for all cells in the mesh, except for the boundary cells. The boundary cells have to be considered separately.

Equation 23 can also be written using summation notation:

$$\begin{gathered} a_p T_P = \sum_N a_N T_N + S_u \\ a_p = \sum_N \frac{k_N A_N}{|\boldsymbol{d}_{PN}|}|\hat{\boldsymbol{n}}| \qquad a_N = \frac{k_N A_N}{|\boldsymbol{d}_{PN}|}|\hat{\boldsymbol{n}}| \qquad S_u = \overline{S}V \end{gathered} \tag{26}$$

where the summation is taken over the N cell neighbours. For 1D cells, $N = 2$. However, for 2D and 3D meshes, the summation notation becomes more useful as each cell has multiple cell neighbours and the finite volume discretisation can become quite long (this will be shown in Chapter 2)! Also notice that in the summation notation, the magnitude of the unit normal vector $|\hat{\boldsymbol{n}}|$ and the vector $|\boldsymbol{d}|$ are used. This is permissible because the unit normal vector and the vector $\boldsymbol{d}$ are parallel with eachother. In turn this is because the mesh used in this course is *structured* and *orthogonal*. Special treatment is required for *non-orthogonal* and *unstructured* meshes. However, the special treatment required for these meshes will not be considered in this course (for brevity).

Boundary Cell (Left) - Dirichlet Boundary Condition

Now the boundary conditions on the left hand boundary cell will be considered, starting with the Dirichlet (fixed temperature) condition. Figure 5 shows a schematic diagram of this cell, with a fixed temperature T_w applied at the wall. The general finite volume discretisation for the 1D diffusion equation is:

$$\left(kA\frac{dT}{dx}\right)_r - \left(kA\frac{dT}{dx}\right)_l + \overline{S}V = 0 \tag{27}$$

The right face of the boundary cell is connected to an interior cell. Hence, the same central differencing scheme for the temperature gradient from the previous section can be used for the right face. However, the left face is connected to a boundary. As shown in Figure 5, the temperature gradient term for the left face is:

$$\left(\frac{dT}{dx}\right)_l = \frac{T_P - T_w}{d_{LP}/2} \tag{28}$$

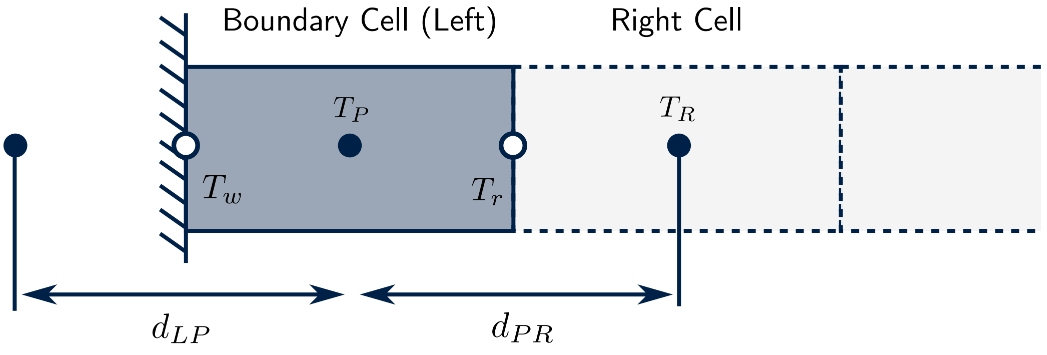


Figure 5: The left boundary cell with temperature T_P at its centroid. The shared face between the boundary cell and the right cell is at a temperature T_r and the wall has a temperature $T_l = T_w$.

The factor of $1/2$ is required as the distance from the cell centroid to the face is $1/2$ of d_{LP} (the distance from the cell centroid to the cell centroid of the adjacent cell). The finite volume discretisation of the 1D heat-diffusion equation for the left boundary cell is now:

$$\left(k_r A_r \frac{T_R - T_P}{d_{PR}}\right) - \left(k_l A_l \frac{T_P - T_w}{d_{LP}/2}\right) + \overline{S}V = 0 \tag{29}$$

Again, introduce the notation $D = k/d$ for the diffusive heat flux per unit area.

$$T_P\left(2D_l A_l + D_r A_r\right) = T_R\left(D_r A_r\right) + T_w\left(2D_l A_l\right) + \overline{S}V \tag{30}$$

For consistency with the interior cell, write in the standard form:

$$a_p T_p = a_L T_L + a_R T_R + S_u \tag{31}$$

$$T_P \underbrace{\left(0 + D_r A_r + 2D_l A_l\right)}_{a_P} = T_L \underbrace{(0)}_{a_L} + T_R \underbrace{\left(D_r A_r\right)}_{a_R} + \underbrace{T_w\left(2D_l A_l\right) + \overline{S}V}_{S_u} \tag{32}$$

For comparison with the interior cell, the boundary cell (left) has the following coefficients:

$$a_L = 0 \qquad a_R = D_r A_r \qquad a_p = a_L + a_R - S_p \tag{33}$$

$$S_P = -2D_l A_l \qquad S_u = \overline{S}V + T_w(2D_l A_l) \tag{34}$$

It follows that the boundary temperature T_w enters the equation through the source terms S_u and S_p. The left coefficient a_L is also set to zero, as this face is not connected to an interior cell. Using summation notation, the finite volume discretisation for the boundary cell can be written:

$$a_p T_P = \sum_N a_N T_N + S_u$$

$$a_p = \left(\sum_N \frac{k_N A_N}{|\boldsymbol{d}_{PN}|}|\hat{\boldsymbol{n}}|\right) + \frac{k_w A_w}{|\boldsymbol{d}_{LP}|/2}|\hat{\boldsymbol{n}}| \qquad a_N = \frac{k_N A_N}{|\boldsymbol{d}_{PN}|}|\hat{\boldsymbol{n}}|$$

$$S_u = \overline{S}V + T_w\left(\frac{k_w A_w}{|\boldsymbol{d}_{LP}|/2}|\hat{\boldsymbol{n}}|\right) \tag{35}$$

When using summation notation, it should be noted that $a_N = 0$ for the left boundary face and the summation for a_P does not include the left boundary face. The wall temperature enters the equation through the source terms S_u and S_p as before.

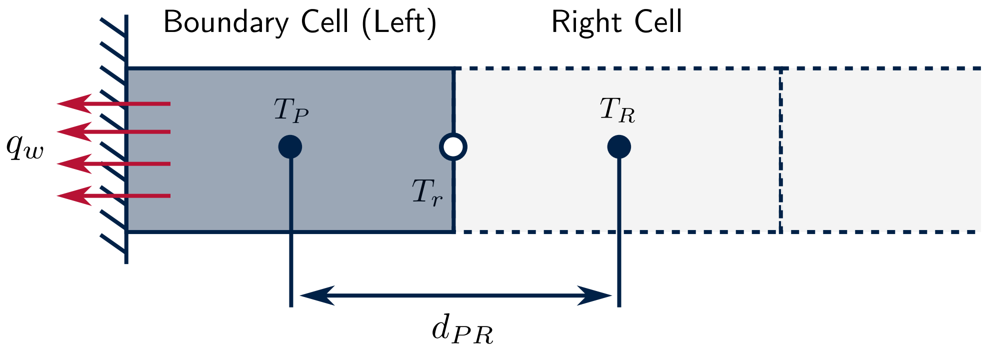


Figure 6: The left boundary cell with a temperature T_P at its centroid. The shared face between the boundary cell and the right cell is at a temperature T_r and a fixed heat flux of q_w is applied at the left boundary face. Positive q_w indicates heat passing out of the domain.

Boundary Cell (Left) - Neumann Boundary Condition

Figure 6 shows a schematic diagram of the same boundary cell, but with a fixed heat flux (per unit area) q_w applied at the left wall, instead of a fixed temperature T_w. Mathematically, the heat flux from the wall (per unit area) q_w is given by Fourier's law:

$$q_w = -k\nabla T \cdot \hat{\boldsymbol{n}} \qquad [\mathrm{W/m^2}] \tag{36}$$

$$q_w = -k\left(\frac{\partial T}{\partial x}, \frac{\partial T}{\partial y}, \frac{\partial T}{\partial z}\right) \cdot (n_x, n_y, n_z) \tag{37}$$

$$q_w = -k\left(\frac{\partial T}{\partial x}n_x + \frac{\partial T}{\partial y}n_y + \frac{\partial T}{\partial z}n_z\right) \tag{38}$$

where $\hat{\boldsymbol{n}}$ is the unit normal vector **out of the cell** into the wall. The negative sign is required to ensure that the heat flows in the opposite direction to the temperature gradient (high temperature to low temperature). In 1D, Fourier's Law for the wall heat flux reduces to:

$$q_w = -k\left(\frac{\partial T}{\partial x}n_x + \cancelto{0}{\frac{\partial T}{\partial y}}n_y + \cancelto{0}{\frac{\partial T}{\partial z}}n_z\right) \tag{39}$$

$$q_w = -k\frac{dT}{dx}n_x \qquad [\mathrm{W/m^2}] \tag{40}$$

Hence, the fixed heat flux boundary condition is a type of **Neumann** boundary condition, as the heat flux (q_w) is being used to set the temperature gradient at the wall. To understand how heat flux boundary conditions are applied, the finite volume discretisation for a general 1D cell needs to be revisited.

$$\underbrace{\left(kA\frac{dT}{dx}n_x\right)_r}_{\text{Heat Flux Right Face}} + \underbrace{\left(kA\frac{dT}{dx}n_x\right)_l}_{\text{Heat Flux Left Face}} + \overline{S}V = 0 \tag{41}$$

The underlined terms in the above equation actually represent the heat flux **out** of the left and right faces of the cell. To understand why, consider Fourier's Law in 1D across an interior face in the mesh:

$$q = -k\frac{dT}{dx}n_x \qquad [\mathrm{W/m^2}] \tag{42}$$

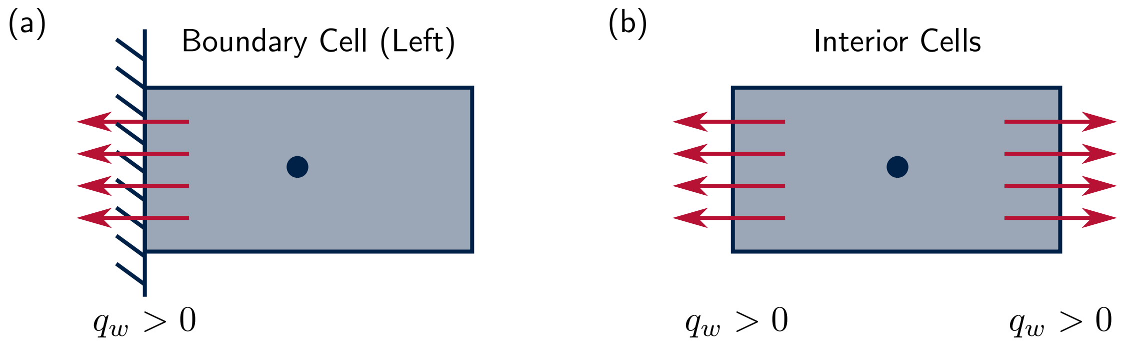


Figure 7: Diagram to show the sign convention for positive heat flux through the (a) boundary and (b) interior faces of the mesh.

Substitute this form of Fourier's Law into the finite volume discretisation for the interior cells (equation 41):

$$\underbrace{(-q_r A_r)} + \underbrace{(-q_l A_l)} + \overline{S}V = 0 \tag{43}$$

As the unit normal vector always points out of the cells, each underlined term in the above equation represents a heat flux **out** of the cell, across its faces (except for the source term $\overline{S}V$). Once again, the negative sign ensures that the heat flows in the opposite direction to the temperature gradient (high temperature to low temperature).

When considering the left boundary cell in Figure 6, a fixed heat flux q_w is applied at the left face.

$$q_w = -k_l \frac{dT}{dx} n_x \qquad [\mathrm{W/m^2}] \tag{44}$$

It should be noted that the heat flux out of the wall into the fluid is equal and opposite the heat flux out of the fluid into the wall. For consistency, the convention of positive heat flux passing **out of the fluid cell** into the wall will be taken as positive ($q_w > 0$), while heat flux passing out of the wall into the fluid will be taken as negative ($q_w < 0$). Figure 7 shows a schematic diagram to highlight the heat flux sign convention adopted for boundary and interior cells. Returning to Figure 6, the heat flux across the right face (interior face) of the left boundary cell can be calculated using central differencing from the previous section. Hence, the finite volume discretisation for this boundary cell becomes:

$$k_r A_r \left(\frac{T_R - T_P}{d_{PR}} \right) - q_w A_l + \overline{S}V = 0 \tag{45}$$

As the boundary heat flux per unit area (q_w) has units of W/m^2, it is multiplied by the face area to obtain the correct units of W. To simplify this equation further, the diffusive heat flux per unit area ($D = k/d$) will be introduced again.

$$D_r A_r (T_R - T_p) - q_w A_l + \overline{S}V = 0 \tag{46}$$

$$T_P\,(D_r A_r) = T_R (D_r A_r) - q_w A_l + \overline{S}V \tag{47}$$

For consistency with the interior cell, write in the standard form:

$$a_p T_p = a_L T_L + a_R T_R + S_u \tag{48}$$

$$T_P \underbrace{(0 + D_r A_r + 0)}_{a_P} = T_L \underbrace{(0)}_{a_L} + T_R \underbrace{(D_r A_r)}_{a_R} + \underbrace{-q_w A_l + \overline{S}V}_{S_u} \tag{49}$$

Hence, the boundary cell (left) has the following coefficients when a Neumann condition is applied:

$$a_L = 0 \qquad a_R = D_r A_r \qquad a_p = a_L + a_R - S_p \tag{50}$$

$$S_P = 0 \qquad S_u = \overline{S}V - q_w A_l \tag{51}$$

Using summation notation, the finite volume discretisation for boundary cells with Neumann boundary conditions can be written:

$$a_p T_P = \sum_N a_N T_N + S_u$$

$$a_p = \left(\sum_N \frac{k_N A_N}{|\boldsymbol{d}_{PN}|} |\hat{\boldsymbol{n}}| \right) \qquad a_N = \frac{k_N A_N}{|\boldsymbol{d}_{PN}|} |\hat{\boldsymbol{n}}| \qquad S_u = \overline{S}V - q_w A_l$$

As before, the coefficient $a_N = 0$ for boundary faces. Notice that unlike the Dirichlet boundary condition, the Neumann boundary condition (q_w) only makes a contribution to S_u and not a_P.

Summary of Coefficients

A summary of the finite volume coefficients is provided in the table below for interior and boundary cells. Notice that the only difference between the Dirichlet and Neumann boundary conditions is the coefficients S_u and S_p.

Dirichlet Boundary Condition

Cell Type	a_L	a_R	a_P	S_p	S_u
Boundary (L)	0	$D_r A_r$	$a_l + a_r - S_p$	$-2D_l A_l$	$T_w(2D_l A_l) + \overline{S}V$
Interior	$D_l A_l$	$D_r A_r$	$a_l + a_r - S_p$	0	$\overline{S}V$
Boundary (R)	$D_l A_l$	0	$a_l + a_r - S_p$	$-2D_r A_r$	$T_w(2D_r A_r) + \overline{S}V$

Neumann Boundary Condition

Cell Type	a_L	a_R	a_P	S_p	S_u
Boundary (L)	0	$D_r A_r$	$a_l + a_r - S_p$	0	$-q_w A_l + \overline{S}V$
Interior	$D_l A_l$	$D_r A_r$	$a_l + a_r - S_p$	0	$\overline{S}V$
Boundary (R)	$D_l A_l$	0	$a_l + a_r - S_p$	0	$-q_w A_r + \overline{S}V$

The summation notation can also be summarised in a table. However, rather than arranging by cell type, this table is arranged by the face type on each cell. The contribution of each face to the cell total is then given (noting that we sum over the N cell faces):

Summation Notation

Face Type	a_P	a_N	S_u
Interior	$\frac{k_N A_N}{\lvert\boldsymbol{d}_{PN}\rvert}\lvert\hat{\boldsymbol{n}}\rvert$	$\frac{k_N A_N}{\lvert\boldsymbol{d}_{PN}\rvert}\lvert\hat{\boldsymbol{n}}\rvert$	0
Dirichlet	$\frac{k_w A_w}{\lvert\boldsymbol{d}_{LP}\rvert/2}\lvert\hat{\boldsymbol{n}}\rvert$	0	$T_w\left(\frac{k_w A_w}{\lvert\boldsymbol{d}_{LP}\rvert/2}\lvert\hat{\boldsymbol{n}}\rvert\right)$
Neumann	0	0	$-q_w A_w$

If the above (summation notation) table is used to assemble the equations, it should be remembered that the additional volumetric source contribution ($\overline{S}V$), which is not included in the table, is still required.

Assemble and Solve the Equations

As shown in the previous course, once the coefficients (a_P, a_L, a_R, S_u) have been calculated, the finite volume equations can be assembled in matrix form. Start by writing an equation for each cell in the mesh. If the mesh has 5 cells, then the equations are:

Cell 1	Boundary Cell (Left)	$a_{P1}T_1 = a_{R1}T_2 + S_{u1}$
Cell 2	Interior Cell	$a_{P2}T_2 = a_{L2}T_1 + a_{R2}T_3 + S_{u2}$
Cell 3	Interior Cell	$a_{P3}T_3 = a_{L3}T_2 + a_{R3}T_4 + S_{u3}$
Cell 4	Interior Cell	$a_{P4}T_4 = a_{L4}T_3 + a_{R4}T_5 + S_{u4}$
Cell 5	Boundary Cell (Right)	$a_{P5}T_5 = a_{L5}T_4 + S_{u5}$

where the coefficients a_P, a_L, a_R and S_u are given in the summary in the previous section. Rearrange the equations, so that all the temperatures (T_P, T_L and T_R) are on the left hand side and the source terms (S_u) are on the right hand side.

Cell 1	Boundary Cell (Left)	$a_{P1}T_1 - a_{R1}T_2 = S_{u1}$
Cell 2	Interior Cell	$-a_{L2}T_1 + a_{P2}T_2 - a_{R2}T_3 = S_{u2}$
Cell 3	Interior Cell	$-a_{L3}T_2 + a_{P3}T_3 - a_{R3}T_4 = S_{u3}$
Cell 4	Interior Cell	$-a_{L4}T_3 + a_{P4}T_4 - a_{R4}T_5 = S_{u4}$
Cell 5	Boundary Cell (Right)	$-a_{L5}T_4 + a_{P5}T_5 = S_{u5}$

These equations can now be written concisely in matrix form ($\boldsymbol{AT} = \boldsymbol{B}$):

$$\begin{bmatrix} a_{P1} & -a_{R1} & 0 & 0 & 0 \\ -a_{L2} & a_{P2} & -a_{R2} & 0 & 0 \\ 0 & -a_{L3} & a_{P3} & -a_{R3} & 0 \\ 0 & 0 & -a_{L4} & a_{P4} & -a_{R4} \\ 0 & 0 & 0 & -a_{L5} & a_{P5} \end{bmatrix} \begin{bmatrix} T_1 \\ T_2 \\ T_3 \\ T_4 \\ T_5 \end{bmatrix} = \begin{bmatrix} S_{u1} \\ S_{u2} \\ S_{u3} \\ S_{u4} \\ S_{u5} \end{bmatrix} \tag{52}$$

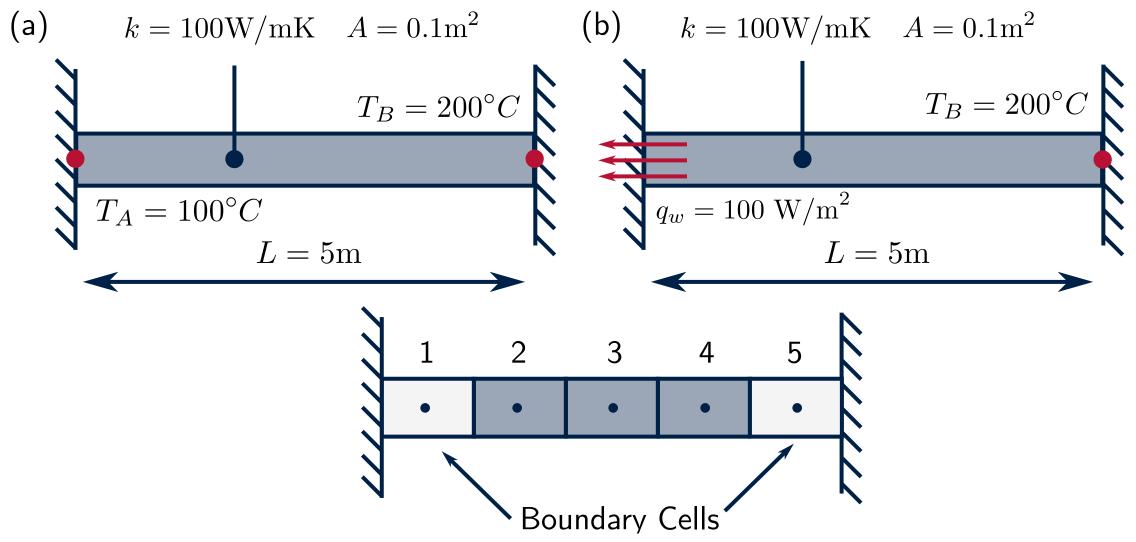

(a)
$k = 100\mathrm{W/mK}$ $A = 0.1\mathrm{m}^2$
$T_B = 200°C$
$T_A = 100°C$
$L = 5\mathrm{m}$
(b)
$k = 100\mathrm{W/mK}$ $A = 0.1\mathrm{m}^2$
$T_B = 200°C$
$q_w = 100\ \mathrm{W/m}^2$
$L = 5\mathrm{m}$
1
2
3
4
5
Boundary Cells

Figure 8: An example problem to demonstrate 1D heat diffusion in a bar. Two cases will be considered: (a) a fixed temperature (T_A) of $100°C$ at the left end of the bar and (b) a fixed heat flux (q_w) of 100 W/m^2 at the left end of the bar.

Notice that the matrices have a diagonal banded structure. This is because the mesh is structured (the cells are ordered in a regular pattern) and the coefficients represent the connectivity to the neighbouring cells. The banded structure of the matrices will change slightly in later chapters, when 2D meshes are considered.

Commercial CFD solvers populate the matrices by calculating the coefficients (a_l, a_p and a_r) automatically for the user and then solve the matrix equations. In the next section, the entire process will be demonstrated with an example problem. A mesh will be defined, the coefficients will be calculated and then the matrices will be constructed and solved.

Example Problem - Heat Diffusion in a Bar

To demonstrate the difference between the Neumann and Dirichlet boundary conditions an example problem will now be considered. Consider 1D steady-state diffusion of heat in a bar, as shown in Figure 8 (b). For ease of comparison, this example problem has the same geometry, mesh and material properties as the previous course. However, at the left hand end, a heat flux boundary condition will be applied (Figure 8 (b)), rather than a fixed temperature boundary condition (Figure 8 (a)), as was applied in the previous course. The bar has a length of 5m, a cross-sectional area of 0.1 m^2 and a thermal conductivity of 100 W/mK. The temperature at the right end of the bar (T_B) is $200°C$. There is a constant heat source ($\overline{S}$) of 1000 W/m^3 in the bar. At the left end of the bar, a fixed heat flux (q_w) of 100 W/m^2 will be applied, rather than a fixed temperature (T_A) of 100°C, as was applied in the previous course.

Step 1: Divide the Geometry into a Mesh

For the example in Figure 8, divide the geometry into a mesh of 5 cells of equal length. The length of each cell (L_{cell}) is given by:

$$L_{\text{cell}} = \frac{L}{N} = \frac{5}{5} = 1\text{m} \tag{53}$$

Because the cells are uniformly distributed and have equal size, the distance between cell centroids d is equivalent to the length of the cells. Hence:

$$d_{LP} = d_{PR} = d = 1\text{m} \tag{54}$$

Step 2: Assign Material Properties

The thermal conductivity k and the cross-sectional area A are the same for every cell in the mesh. Hence, the parameter DA is given by:

$$DA = \frac{kA}{d} = \frac{100 * 0.1}{1} = 10 \text{ [W/K]} \tag{55}$$

The volumetric heat source in each cell is given by:

$$\overline{S}V = \overline{S}AL_{\text{cell}} = 1000 * 0.1 * 1 = 100 \text{ [W]} \tag{56}$$

Step 3: Calculate Matrix Coefficients

Now calculate the coefficients required to assemble the matrices. The most straightforward way is to fill in the table of coefficients:

Neumann Boundary Conditions (at the left end of the bar)

	a_L	a_R	S_p	S_u	a_P
Boundary (Left)	0	10	**0**	**90**	**10**
Interior	10	10	0	100	20
Boundary (Right)	10	0	-20	4100	30

For comparison, the matrix coefficients for the same problem with a Dirichlet boundary condition (fixed temperature) at the left end are shown below. The differences between the two sets of matrix coefficients are highlighted in red.

Dirichlet Boundary Conditions (at the left end of the bar)

	a_L	a_R	S_p	S_u	a_P
Boundary (Left)	0	10	**-20**	**2100**	**30**
Interior	10	10	0	100	20
Boundary (Right)	10	0	-20	4100	30

Alternatively, the summation notation summary table can be filled in (noting that an additional source $\overline{S}V = 100$W is required in each cell):

Dirichlet Boundary Conditions (at the left end of the bar)

Face Type	a_P	a_N	S_u
Interior	10	10	0
Dirichlet (Left)	20	0	2000
Dirichlet (Right)	20	0	4000
Neumann (Left)	0	0	-10

Step 4: Assemble the Matrices

Assign the coefficients to their correct locations in the matrix. Note that the final matrices will be identical, regardless of the method used to calculate the coefficients (summation notation, $D = k/d$, or otherwise).

For the Neumann (fixed heat flux) boundary condition at the left end:

$$\begin{bmatrix} \mathbf{10} & -10 & 0 & 0 & 0 \\ -10 & 20 & -10 & 0 & 0 \\ 0 & -10 & 20 & -10 & 0 \\ 0 & 0 & -10 & 20 & -10 \\ 0 & 0 & 0 & -10 & 30 \end{bmatrix} \begin{bmatrix} T_1 \\ T_2 \\ T_3 \\ T_4 \\ T_5 \end{bmatrix} = \begin{bmatrix} \mathbf{90} \\ 100 \\ 100 \\ 100 \\ 4100 \end{bmatrix} \tag{57}$$

and for the Dirichlet boundary condition (fixed temperature) at the left end:

$$\begin{bmatrix} \mathbf{30} & -10 & 0 & 0 & 0 \\ -10 & 20 & -10 & 0 & 0 \\ 0 & -10 & 20 & -10 & 0 \\ 0 & 0 & -10 & 20 & -10 \\ 0 & 0 & 0 & -10 & 30 \end{bmatrix} \begin{bmatrix} T_1 \\ T_2 \\ T_3 \\ T_4 \\ T_5 \end{bmatrix} = \begin{bmatrix} \mathbf{2100} \\ 100 \\ 100 \\ 100 \\ 4100 \end{bmatrix} \tag{58}$$

Once again, the differences between the Dirichlet and Neumann boundary conditions are highlighted in red.

Step 5: Solve the Equations

Now that the matrices have been assembled, the matrix equation can solved with an appropriate *iterative method*. As with the previous course, different algorithms to solve the matrix equation $\boldsymbol{AT} = \boldsymbol{B}$ will not be considered, as details can be found in any comprehensive text on linear algebra. Instead, the default algorithms used by Excel and Python will be used, as this is not the focus of this course.

Run the Example Problem Yourself!

Now, open either the Excel spreadsheets, the Python source code or the MATLAB code and solve the problem with the Neumann boundary condition (fixed heat flux) yourself. For the Dirichlet boundary condition, you can use the source code from the previous course.

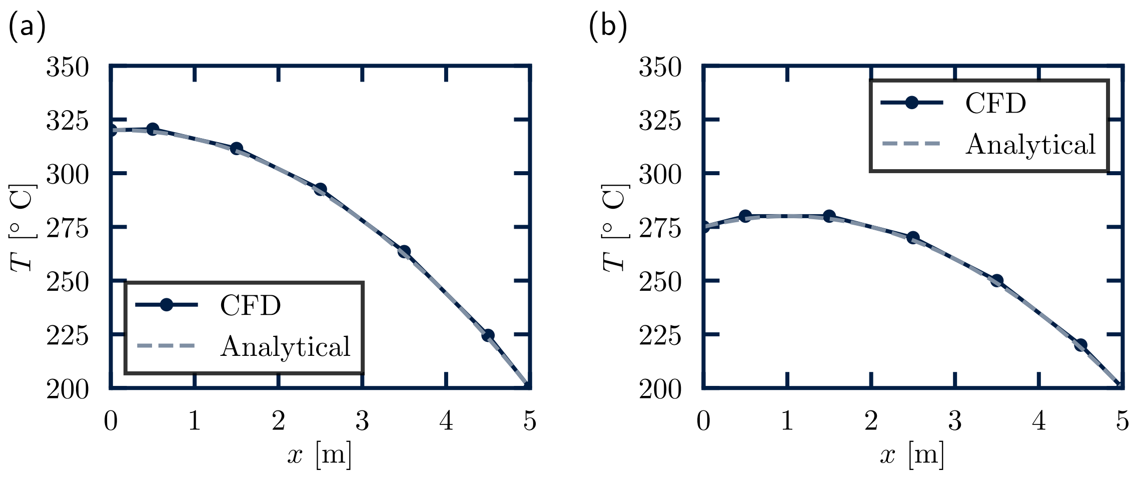


Figure 9: Temperature variation along the 1D bar with a heat flux of (a) 100 W/m^2 and (b) 1000 W/m^2 applied at the left end.

Excel	`neumannBoundaryConditions.xlsx`
Python	`neumannBoundaryConditions.py`
MATLAB	`neumannBoundaryConditions.m`

Examine the calculation of the coefficients, the assembly of the matrices and run the code. You can even try changing the strength of the heat flux at the left hand end of the bar (q_w), or even set an adiabatic condition (zero heat flux, $q_w = 0$).

Results

Figure 9 shows the temperature distribution in the bar with a heat flux of 100 W/m^2 applied at the left hand end. For comparison, the analytical solution is also shown:

$$T = \frac{\overline{S}}{2k}\left(L^2 - x^2\right) + \frac{q_w}{k}(x - L) + T_B \tag{59}$$

By changing the heat flux at the left boundary from 100 W/m^2 to 1000 W/m^2, Figure 9(b) shows that the temperature of the left end of the bar is not fixed. The heat flux q_w implicitly sets the temperature gradient at the left end of the bar (as the thermal conductivity k is fixed) and the temperature profile develops to match this gradient.

$$q_w = -k\frac{dT}{dx}n_x \tag{60}$$

As a result, when a heat flux boundary condition is applied in a CFD code, the boundary temperature on that surface develops as part of the solution. To calculate the boundary temperature from the CFD solution (T_w), the temperature must be extrapolated from the temperature at the cell centroid T_1. Central differencing can be used to extrapolate the boundary temperature from the temperature at the cell centroids.

$$q_w A_L = -k_L A_L\left(\frac{T_1 - T_A}{d_{LP}/2}\right)n_x \qquad D_l A_l = \frac{k_L A_L}{d_{LP}} \tag{61}$$

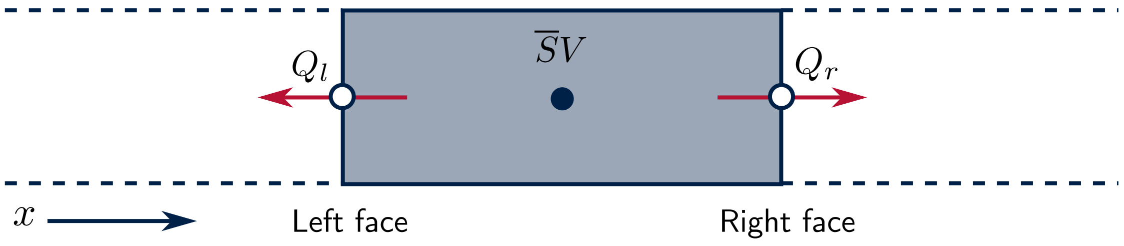


Figure 10: A diagram to show the the heat flux out of the cell across the left (Q_l) and right (Q_r) faces of the cell.

$$T_A = T_1 + \frac{q_A A_L}{2 D_l A_L n_x} \tag{62}$$

The use of central differencing to extrapolate and calculate the boundary temperature can be found in both the Excel spreadsheet and Python source code. Note that for the left boundary face, the unit normal vector (n_x) points in the negative x direction, so $n_x = -1$.

Heat Balancing

Once a CFD solution has been computed, a heat balance can be written for each of the cells in the mesh as an additional check of the solution. To understand the heat balance, start with the general finite volume discretisation for the interior cells:

$$k_l A_l \left(\frac{T_P - T_L}{d_{LP}} \right) n_x + k_r A_r \left(\frac{T_R - T_P}{d_{PR}} \right) n_x + \overline{S}V = 0 \tag{63}$$

The heat flux out of the left face of the cell (Q_l) is:

$$Q_l = -k_l A_l \left(\frac{T_P - T_L}{d_{LP}} \right) n_x \qquad \text{[W]} \tag{64}$$

and the heat flux out of the right face of the cell (Q_r) is:

$$Q_r = -k_r A_r \left(\frac{T_R - T_P}{d_{PR}} \right) n_x \qquad \text{[W]} \tag{65}$$

The negative sign is required to ensure that the heat flows in the opposite direction to the temperature gradient and the unit normal vector ensures that heat fluxes out of the cell are positive. Substitute these definitions into the general finite volume discretisation for the interior cell:

$$-Q_l - Q_r + \overline{S}V = 0 \tag{66}$$

For the left boundary cell, the heat flux balance is the same. However, it should be noted that Q_l is based on a distance of $d_{LP}/2$ (the distance from the centroid to the wall) not d_{LP}.

$$-Q_l - Q_r + \overline{S}V = 0 \tag{67}$$

Likewise, for the right boundary face:

$$-Q_l - Q_r + \overline{S}V = 0 \tag{68}$$

Physically, these equations state that the sum of the heat fluxes into the cell (positive heat sources $\overline{S}$ generate heat in the cell) is equal to zero. The equation will now be evaluated explicitly, using the temperatures that were computed at the cell centroids. Remember that this is a post-processing operation and the temperatures are now known. As the matrix solvers are (generally) iterative, there will be some error in solving the finite volume equations. This error in the heat balance in each cell is given by:

$$\text{Error} = -Q_l - Q_r + \overline{S}V \tag{69}$$

The table below summarises the heat balance for each cell in the mesh:

Cell	Q_l [W]	Q_r [W]	$\overline{S}V$ [W]	Error
1	**10**	90	100	0
2	-90	190	100	0
3	-190	290	100	0
4	-290	390	100	0
5	-390	**490**	100	0

Total heat flux out of the bar = 10 + 490 = 500W
Total heat generated in the bar = 100 + 100 + 100 + 100 + 100 = 500W

The heat balance is useful as we can easily observe that the total heat flux out of the left and right faces of the bar (highlighted in red in the table), balances the total heat generated in the bar (500W). As both heat fluxes are positive, this indicates that heat is flowing out of the bar at both ends, with the majority of the heat flowing out of the right face as it is at a lower temperature (see Figure 9). As a further check, we can also see that the heat flux boundary condition has been applied correctly to the left boundary cell (cell 1):

$$Q_A = q_w A = 100 * 0.1 = 10 \text{ W} \tag{70}$$

It may be noted that for this problem, the error in the heat balance in every cell is zero. This is because a direct matrix solver has been used to solve the equations. However, real CFD codes use iterative matrix solvers and a small error will generally exists in every cell in the mesh. This error is often called the **residual** of the equation. As a vector, the root-mean-square and maximum values of the vector are often computed and reported to the user (instead of the entire vector) as a measure of the convergence of the solution.

$$\text{RMS} = \sqrt{\frac{1}{5}\left(\text{Error } 1^2 + \text{Error } 2^2 + \text{Error } 3^2 + \text{Error } 4^2 + \text{Error } 5^2\right)} \tag{71}$$

$$\text{MAX} = \max\left(\text{Error } 1, \text{Error } 2, \text{Error } 3, \text{Error } 4, \text{Error } 5\right) \tag{72}$$

These are the values that are typically shown to the user as convergence graphs in the graphical user interface (GUI) of the CFD code.

Chapter 2

Transport Equations in 2D

2 Transport Equations in 2D

The heat diffusion equation will now be solved in 2D (x and y) rather than just 1D (x) using the finite volume method. This same approach can be used to integrate and solve any transport equation (momentum, turbulence, species concentration etc.) under consideration. Starting with the heat diffusion equation in vector notation:

$$0 = \nabla \cdot (k \nabla T) + S \tag{73}$$

In 2D Cartesian coordinates, the derivatives can be expanded to give:

$$0 = \frac{\partial}{\partial x}\left(k\frac{\partial T}{\partial x}\right) + \frac{\partial}{\partial y}\left(k\frac{\partial T}{\partial y}\right) + S \tag{74}$$

Following the same approach as the previous chapter, integrate the equation over a finite control volume (cell) with volume V.

$$0 = \int_V \left[\frac{\partial}{\partial x}\left(k\frac{\partial T}{\partial x}\right) + \frac{\partial}{\partial y}\left(k\frac{\partial T}{\partial y}\right) + S\right] \mathrm{d}V \tag{75}$$

Once again, split the integral into diffusion and source components. This is permissible as integration and addition are commutative operations (they can be performed in any order without changing the result).

$$0 = \int_V \left[\frac{\partial}{\partial x}\left(k\frac{\partial T}{\partial x}\right) + \frac{\partial}{\partial y}\left(k\frac{\partial T}{\partial y}\right)\right] \mathrm{d}V + \int_V [S]\,\mathrm{d}V \tag{76}$$

As before, the source term is averaged over the control volume, so that it can be moved outside the integral.

$$0 = \int_V \left[\frac{\partial}{\partial x}\left(k\frac{\partial T}{\partial x}\right) + \frac{\partial}{\partial y}\left(k\frac{\partial T}{\partial y}\right)\right] \mathrm{d}V + \overline{S}\int_V \mathrm{d}V \tag{77}$$

$$0 = \int_V \left[\frac{\partial}{\partial x}\left(k\frac{\partial T}{\partial x}\right) + \frac{\partial}{\partial y}\left(k\frac{\partial T}{\partial y}\right)\right] \mathrm{d}V + \overline{S}V \tag{78}$$

The volume integral of the diffusion term can be simplified using Gauss's divergence theorem. Recall (from the previous course) that the divergence theorem for a general vector field $\boldsymbol{B}$ is written as:

$$\int_V (\nabla \cdot \boldsymbol{B})\,\mathrm{d}V = \int_A (\boldsymbol{B} \cdot \hat{\boldsymbol{n}})\,\mathrm{d}A \tag{79}$$

$$\int_V \left(\frac{\partial B_x}{\partial x} + \frac{\partial B_y}{\partial y} + \frac{\partial B_z}{\partial z}\right) \mathrm{d}V = \int_A (B_x n_x + B_y n_y + B_z n_z)\,\mathrm{d}A \tag{80}$$

where $\hat{\boldsymbol{n}}$ is the unit normal vector pointing out of the control volume and A is the surface area of the control volume. In 2D, the divergence theorem can be written:

$$\int_V \left(\frac{\partial B_x}{\partial x} + \frac{\partial B_y}{\partial y}\right) \mathrm{d}V = \int_A (B_x n_x + B_y n_y)\,\mathrm{d}A \tag{81}$$

For the 2D heat diffusion equation $\boldsymbol{B} = k\nabla T$. Hence $B_x = k\ \partial T/\partial x$ and $B_y = k\ \partial T/\partial y$. Applying the 2D divergence theorem to the 2D heat diffusion equation leads to:

$$0 = \int_A \left[k\frac{\partial T}{\partial x}n_x + k\frac{\partial T}{\partial y}n_y\right] \mathrm{d}A + \overline{S}V \tag{82}$$

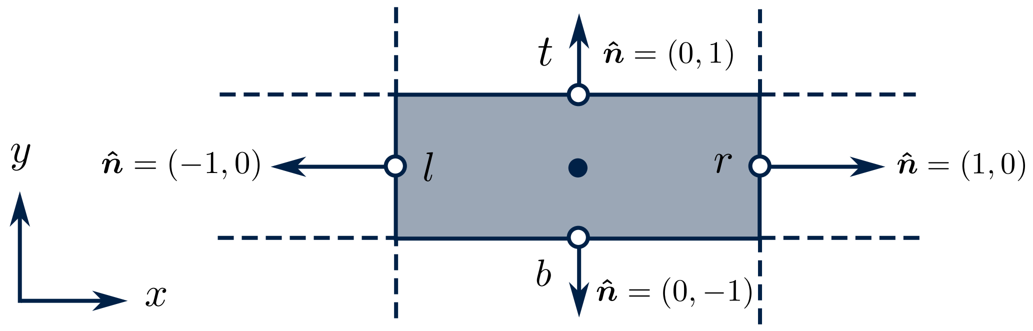

t
$\hat{\boldsymbol{n}} = (0,1)$
y
$\hat{\boldsymbol{n}} = (-1,0)$
l
r
$\hat{\boldsymbol{n}} = (1,0)$
b
$\hat{\boldsymbol{n}} = (0,-1)$
x

Figure 11: A diagram to show the face normal vectors on the left (l), right (r), top (t) and bottom (b) faces of the 2D cell. The cell normal vectors always point out of the cell.

The cell has a finite number of faces (N). For a quadrilateral cell $N = 4$, while for a triangular cell $N = 3$. Hence, the surface integral over the entire surface can be replaced with an integral over each of the N faces of the cell.

$$0 = \sum_N \int_A \left[k\frac{\partial T}{\partial x} n_x + k\frac{\partial T}{\partial y} n_y \right] \mathrm{d}A + \overline{S}V \tag{83}$$

In the second-order finite volume method, the flow quantities all vary linearly across the face. The integral across the face can therefore be reduced to the value at the centre of the face (a constant value).

$$0 = \sum_N \left[k\frac{\partial T}{\partial x} n_x + k\frac{\partial T}{\partial y} n_y \right] \int_A \mathrm{d}A + \overline{S}V \tag{84}$$

$$0 = \sum_N \left[k\frac{\partial T}{\partial x} n_x + k\frac{\partial T}{\partial y} n_y \right] A + \overline{S}V \tag{85}$$

It may be noted that this is equivalent to:

$$0 = \sum_N \left[kA \left(\nabla T \cdot \hat{\boldsymbol{n}} \right) \right] + \overline{S}V \tag{86}$$

To simplify the equation further, consider the 2D quadrilateral cell in Figure 11. The cell has a left face (l), a right face (r), a top face (t) and a bottom face (b). The unit normal vectors always point out of the cell. As the x direction is positive left to right and the y direction is positive bottom to top, the unit normal vectors on each face are:

Face	n_x	n_y
Left (l)	-1	0
Right (r)	1	0
Bottom (b)	0	-1
Top (t)	0	1

Hence, the finite volume discretisation can be simplified further:

$$0 = \left(kA\frac{\partial T}{\partial x} \right)_r - \left(kA\frac{\partial T}{\partial x} \right)_l + \left(kA\frac{\partial T}{\partial y} \right)_t - \left(kA\frac{\partial T}{\partial y} \right)_b + \overline{S}V \tag{87}$$

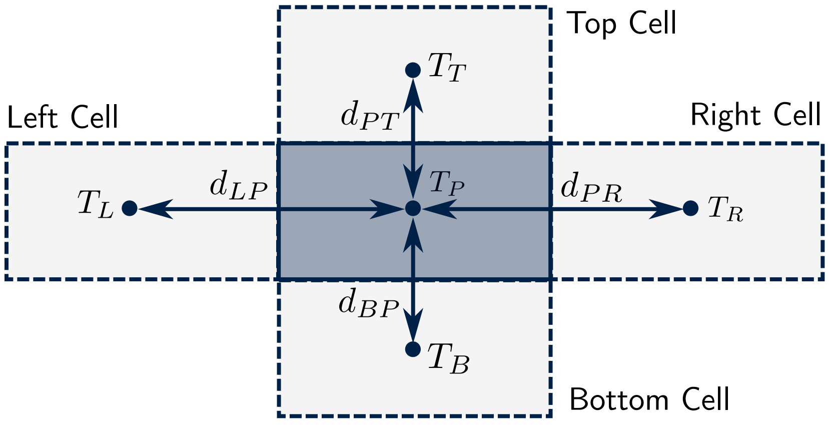


Figure 12: A diagram to show the distance between the cell centroid and the left cell (d_{LP}), right cell (d_{PR}), bottom cell (d_{BP}) and top cell (d_{PT}).

This discretisation is valid for all cells in the mesh that are the same shape and orientation as the quadrilateral cell in Figure 11 (a regular structured mesh). To simplify further, the interior and boundary cells need to be considered separately.

Interior Cells

Starting with the general finite volume discretisation for the 2D quadrilateral cell:

$$0 = \left(kA\frac{\partial T}{\partial x}\right)_r - \left(kA\frac{\partial T}{\partial x}\right)_l + \left(kA\frac{\partial T}{\partial y}\right)_t - \left(kA\frac{\partial T}{\partial y}\right)_b + \overline{S}V \tag{88}$$

Central differencing will be used for all of the diffusion terms. Using the notation in Figure 12 for the distances between the cell centroids (d):

$$0 = k_r A_r \frac{T_R - T_P}{d_{PR}} - k_l A_l \frac{T_P - T_L}{d_{LP}} + k_t A_t \frac{T_T - T_P}{d_{PT}} - k_b A_b \frac{T_P - T_B}{d_{BP}} + \overline{S}V \tag{89}$$

As before, the lowercase subscript notation (l, r, b, t) is used to refer to the faces of the cell and the uppercase subscript notation (L, R, B, T, P) is used to refer to the cell centroids. For simplicity, introduce the diffusive flux per unit area $D = k/d$, which has units of W/m^{2}K.

$$0 = D_r A_r (T_R - T_P) - D_l A_l (T_P - T_L) + D_t A_t (T_T - T_P) - D_b A_b (T_P - T_B) + \overline{S}V \tag{90}$$

Rearrange and group the terms by cell centroid temperatures $(T_P, T_R, T_L, T_T, T_B)$.

$$\begin{aligned} T_P \left[D_l A_l + D_r A_r + D_b A_b + D_t A_t\right] = T_L \left[D_l A_l\right] + T_R \left[D_r A_r\right] \\ + T_B \left[D_b A_b\right] + T_T \left[D_t A_t\right] + \overline{S}V \end{aligned} \tag{91}$$

Manipulate the equation slightly (add a zero to the T_P bracket) to express in standard form.

$$\begin{aligned} T_P \underbrace{\left[D_l A_l + D_r A_r + D_b A_b + D_t A_t + 0\right]}_{a_p} = T_L \underbrace{\left[D_l A_l\right]}_{a_L} + T_R \underbrace{\left[D_r A_r\right]}_{a_R} \\ + T_B \underbrace{\left[D_b A_b\right]}_{a_B} + T_T \underbrace{\left[D_t A_t\right]}_{a_B} + \underbrace{\overline{S}V}_{S_u} \end{aligned} \tag{92}$$

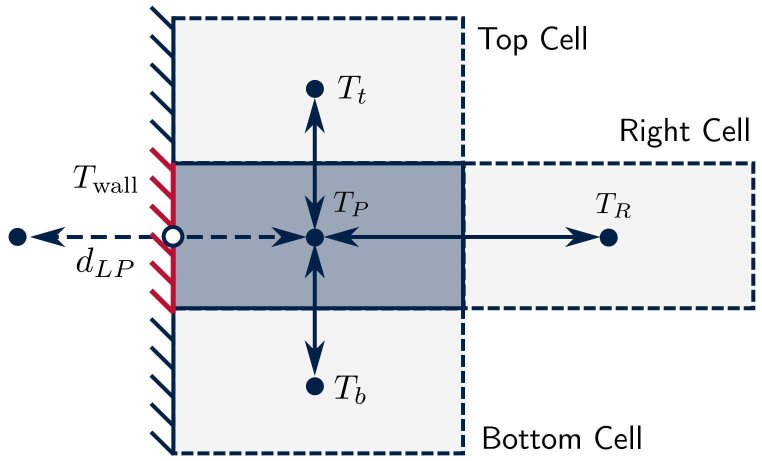


Figure 13: The boundary cell at the left of the domain with temperature T_P at its centroid. The wall (the left face of the cell) is at a fixed temperature T_{wall} and is shown in red.

The standard form is similar to the standard form used for the 1D diffusion equation.

$$\begin{gathered} a_p T_P = a_L T_L + a_R T_R + a_B T_B + a_T T_T + S_u \qquad \text{[2D]} \\ a_p T_P = a_L T_L + a_R T_R + S_u \qquad \text{[1D]} \end{gathered} \tag{93}$$

With the following coefficients for interior cells:

$$\begin{gathered} a_p = a_L + a_R + a_T + a_B - S_p \\ a_L = D_l A_l \qquad a_R = D_r A_r \qquad a_B = D_b A_b \qquad a_T = D_t A_t \\ S_p = 0 \qquad S_u = \overline{S} V \end{gathered} \tag{94}$$

Using summation notation (from the previous chapter), the finite volume discretisation is identical to the 1D discretisation from the previous chapter.

$$a_p T_P = \sum_N a_N T_N + S_u$$

$$a_p = \sum_N \frac{k_N A_N}{|\boldsymbol{d}_{PN}|} |\hat{\boldsymbol{n}}| \qquad a_N = \frac{k_N A_N}{|\boldsymbol{d}_{PN}|} |\hat{\boldsymbol{n}}| \qquad S_u = \overline{S} V \tag{95}$$

The summation notation becomes more useful as the number of faces (N) increases. Hence, it is often used by real CFD codes (and given in user manuals) as cells in 2D and 3D meshes may have many faces and it is easier to keep track of all the terms in the finite volume discretisation.

Boundary Cells - Dirichlet Boundary Conditions

Figure 13 shows a boundary cell on the left of the domain, which will be used to simplify the finite volume discretisation for boundary cells. Starting with the general finite volume discretisation for the 2D heat diffusion equation:

$$0 = \left(kA\frac{\partial T}{\partial x}\right)_r - \left(kA\frac{\partial T}{\partial x}\right)_l + \left(kA\frac{\partial T}{\partial y}\right)_t - \left(kA\frac{\partial T}{\partial y}\right)_b + \overline{S} V \tag{96}$$

The right, top and bottom faces are connected to interior cells. Hence, the heat diffusion through these faces can be simplified using the same treatment as the interior cells.

$$0 = D_r A_r (T_R - T_P) - \left(kA \frac{\partial T}{\partial x} \right)_l + D_t A_t (T_T - T_P) - D_b A_b (T_P - T_B) + \overline{S}V \tag{97}$$

In this chapter, fixed value (Dirichlet) boundary conditions will be applied on the left face. However, the method used in the previous chapter for Neumann boundary conditions could be applied instead if desired. The left boundary face is at a temperature T_{wall} and the distance to the wall is half the distance to the next cell centroid. Hence the heat diffusion through the left face is:

$$\left(kA \frac{\partial T}{\partial x} \right)_l = k_l A_l \frac{T_P - T_{\text{wall}}}{d_{LP}/2} = 2 D_l A_l (T_p - T_{\text{wall}}) \tag{98}$$

Substitute into the finite volume discretisation:

$$0 = D_r A_r (T_R - T_P) - 2 D_l A_l (T_p - T_{\text{wall}}) + D_t A_t (T_T - T_P) - D_b A_b (T_P - T_B) + \overline{S}V \tag{99}$$

Rearrange and group the terms by cell centroid temperatures $(T_P, T_R, T_L, T_T, T_B)$.

$$\begin{aligned} T_P \left[0 + D_r A_r + D_b A_b + D_t A_t + 2 D_l A_l \right] = {} & T_L \left[0 \right] + T_R \left[D_r A_r \right] \\ & + T_B \left[D_b A_b \right] + T_T \left[D_t A_t \right] \\ & + T_{\text{wall}} \left[2 D_l A_l \right] + \overline{S}V \end{aligned} \tag{100}$$

The equation is now in standard form:

$$T_P \underbrace{\left[0 + D_r A_r + D_b A_b + D_t A_t + 2 D_l A_l \right]}_{a_p} = T_L \underbrace{\left[0 \right]}_{a_L} + T_R \underbrace{\left[D_r A_r \right]}_{a_R} \tag{101}$$

$$+ T_B \underbrace{\left[D_b A_b \right]}_{a_B} + T_T \underbrace{\left[D_t A_t \right]}_{a_T} \tag{102}$$

$$+ \underbrace{T_{\text{wall}} \left[2 D_l A_l \right] + \overline{S}V}_{S_u} \tag{103}$$

With the following coefficients:

$$\begin{gathered} a_p = a_L + a_R + a_T + a_B - S_p \\ a_L = 0 \qquad a_R = D_r A_r \qquad a_B = D_b A_b \qquad a_T = D_t A_t \\ S_p = -2 D_l A_l \qquad S_u = T_{\text{wall}} (2 D_l A_l) + \overline{S}V \end{gathered} \tag{104}$$

Using summation notation (from the previous chapter), the finite volume discretisation is identical to the 1D discretisation from the previous chapter.

$$\begin{gathered} a_p T_P = \sum_N a_N T_N + S_u \\ a_p = \left(\sum_N \frac{k_N A_N}{|\boldsymbol{d}_{PN}|} |\hat{\boldsymbol{n}}| \right) + \frac{k_w A_w}{|\boldsymbol{d}_{LP}|/2} |\hat{\boldsymbol{n}}| \qquad a_N = \frac{k_N A_N}{|\boldsymbol{d}_{PN}|} |\hat{\boldsymbol{n}}| \\ S_u = \overline{S}V + T_{\text{wall}} \left(\frac{k_w A_w}{|\boldsymbol{d}_{LP}|/2} |\hat{\boldsymbol{n}}| \right) \end{gathered} \tag{105}$$

Once again it should be remembered that when using the summation notation $a_N = 0$ for the boundary faces and the summation for a_P does not include the boundary faces.

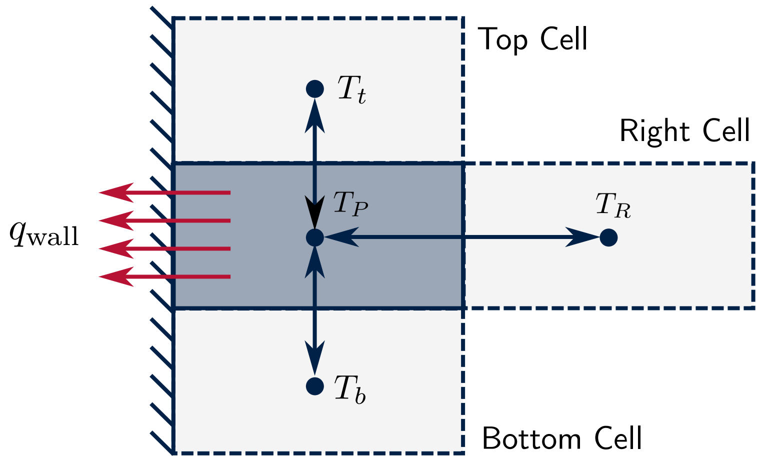


Figure 14: The boundary cell at the left of the domain with temperature T_P at its centroid. A fixed heat flux q_{wall} is applied at the wall (the left face of the cell).

Boundary Cells - Neumann Boundary Conditions

Now consider the case where a fixed heat flux (Neumann boundary condition) is applied to the left boundary face, as shown in Figure 14. Starting with the general finite volume discretisation for the 2D heat diffusion equation:

$$0 = \left(kA\frac{\partial T}{\partial x}\right)_r - \left(kA\frac{\partial T}{\partial x}\right)_l + \left(kA\frac{\partial T}{\partial y}\right)_t - \left(kA\frac{\partial T}{\partial y}\right)_b + \overline{S}V \tag{106}$$

The right, top and bottom faces are connected to interior cells. Hence, the heat flux through these faces can be simplified using the same treatment as the interior cells.

$$0 = D_r A_r (T_R - T_P) - \left(kA\frac{\partial T}{\partial x}\right)_l + D_t A_t (T_T - T_P) - D_b A_b (T_P - T_B) + \overline{S}V \tag{107}$$

With a Neumann boundary condition, the heat flux per unit area **out** of the left face of the cell is:

$$q_{\text{wall}} = -k_l \frac{\partial T}{\partial x} n_x \tag{108}$$

Note that the unit normal vector for this cell $n_x = -1$. Hence:

$$q_{\text{wall}} = k_l \frac{\partial T}{\partial x} \tag{109}$$

Hence, the finite volume discretisation now becomes:

$$0 = D_r A_r (T_R - T_P) - q_{\text{wall}} A_l + D_t A_t (T_T - T_P) - D_b A_b (T_P - T_B) + \overline{S}V \tag{110}$$

Rearrange and group the terms by cell centroid temperatures $(T_P, T_R, T_L, T_T, T_B)$.

$$\begin{aligned} T_P \left[0 + D_r A_r + D_b A_b + D_t A_t + 0\right] = T_L \left[0\right] + T_R \left[D_r A_r\right] \\ + T_B \left[D_b A_b\right] + T_T \left[D_t A_t\right] \\ - q_{\text{wall}} A_l + \overline{S}V \end{aligned} \tag{111}$$

The equation is now in standard form:

$$
\begin{aligned}
T_P \underbrace{[0 + D_r A_r + D_b A_b + D_t A_t + 0]}_{a_p} = T_L \underbrace{[0]}_{a_L} &+ T_R \underbrace{[D_r A_r]}_{a_R} \\
&+ T_B \underbrace{[D_b A_b]}_{a_B} + T_T \underbrace{[D_t A_t]}_{a_T} \\
&+ \underbrace{-q_{\mathsf{wall}} A_l + \overline{S} V}_{S_u}
\end{aligned} \tag{112}
$$

With the following coefficients:

$$
\begin{gathered}
a_p = a_L + a_R + a_T + a_B - S_p \\
a_L = 0 \qquad a_R = D_r A_r \qquad a_B = D_b A_b \qquad a_T = D_t A_t \\
S_p = 0 \qquad S_u = -q_{\mathsf{wall}} A_l + \overline{S} V
\end{gathered} \tag{113}
$$

Using summation notation (from the previous chapter), the finite volume discretisation is identical to the 1D discretisation from the previous chapter.

$$
a_p T_P = \sum_N a_N T_N + S_u
$$

$$
a_p = \left(\sum_N \frac{k_N A_N}{|\boldsymbol{d}_{PN}|} |\hat{\boldsymbol{n}}| \right) \qquad a_N = \frac{k_N A_N}{|\boldsymbol{d}_{PN}|} |\hat{\boldsymbol{n}}| \qquad S_u = \overline{S} V - q_w A_w
$$

Once again it should be remembered that when using the summation notation $a_N = 0$ for the boundary faces and the summation for a_P does not include the boundary faces.

Summary of Coefficients

The tables below summarise the coefficients for each of the boundary and interior cells in the mesh when Dirichlet and Neumann boundary conditions are applied.

Dirichlet Boundary Condition (2D)

	a_L	a_R	a_B	a_T	S_p	S_u
Interior	$D_L A_L$	$D_R A_R$	$D_B A_B$	$D_T A_T$	0	$\overline{S} V$
Boundary (L)	0	$D_R A_R$	$D_B A_B$	$D_T A_T$	$-2 D_L A_L$	$T_{\mathsf{wall}}(2 D_L A_L) + \overline{S} V$
Boundary (R)	$D_L A_L$	0	$D_B A_B$	$D_T A_T$	$-2 D_R A_R$	$T_{\mathsf{wall}}(2 D_R A_R) + \overline{S} V$
Boundary (B)	$D_L A_L$	$D_R A_R$	0	$D_T A_T$	$-2 D_B A_B$	$T_{\mathsf{wall}}(2 D_B A_B) + \overline{S} V$
Boundary (T)	$D_L A_L$	$D_R A_R$	$D_T A_T$	0	$-2 D_T A_T$	$T_{\mathsf{wall}}(2 D_T A_T) + \overline{S} V$

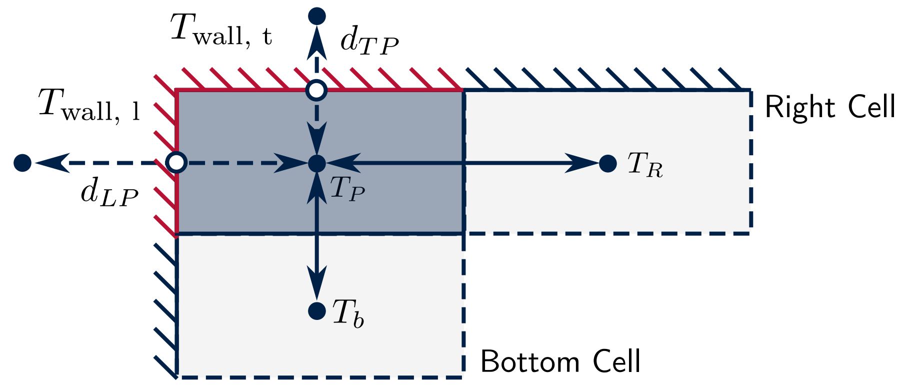


Figure 15: The boundary cell at the top-left of the domain with temperature T_P at its centroid. The top and left walls are at fixed temperatures of $T_{\text{wall, l}}$ and $T_{\text{wall, t}}$

Neumann Boundary Condition (2D)

	a_L	a_R	a_B	a_T	S_p	S_u
Interior	$D_L A_L$	$D_R A_R$	$D_B A_B$	$D_T A_T$	0	$\overline{S}V$
Boundary (L)	0	$D_R A_R$	$D_B A_B$	$D_T A_T$	0	$-q_{\text{wall}} A_l + \overline{S}V$
Boundary (R)	$D_L A_L$	0	$D_B A_B$	$D_T A_T$	0	$-q_{\text{wall}} A_r + \overline{S}V$
Boundary (B)	$D_L A_L$	$D_R A_R$	0	$D_T A_T$	0	$-q_{\text{wall}} A_B + \overline{S}V$
Boundary (T)	$D_L A_L$	$D_R A_R$	$D_T A_T$	0	0	$-q_{\text{wall}} A_T + \overline{S}V$

The summation notation can also be summarised in a table. Following the previous chapter, this table gives the contribution of each face to the cell total:

Summation Notation

Face Type	a_P	a_N	S_u
Interior	$\frac{k_N A_N}{\lvert\boldsymbol{d}_{PN}\rvert}\lvert\hat{\boldsymbol{n}}\rvert$	$\frac{k_N A_N}{\lvert\boldsymbol{d}_{PN}\rvert}\lvert\hat{\boldsymbol{n}}\rvert$	0
Dirichlet	$\frac{k_w A_w}{\lvert\boldsymbol{d}_{LP}\rvert/2}\lvert\hat{\boldsymbol{n}}\rvert$	0	$T_w\left(\frac{k_w A_w}{\lvert\boldsymbol{d}_{LP}\rvert/2}\lvert\hat{\boldsymbol{n}}\rvert\right)$
Neumann	0	0	$-q_{\text{wall}} A_w$

It should be emphasised once again that with the summation notation, the contribution of the volumetric source term $(\overline{S}V)$ also needs to be included in the total for S_u.

Cells with Multiple Boundary Faces

For 2D and 3D meshes, some boundary cells may have more than 1 boundary face. Furthermore, the boundary conditions on these boundary faces may be different. Figure 15 shows an example cell with 2 boundary faces. For this cell, the finite volume discretisation is:

$$0 = \left(kA\frac{\partial T}{\partial x}\right)_r - \left(kA\frac{\partial T}{\partial x}\right)_l + \left(kA\frac{\partial T}{\partial y}\right)_t - \left(kA\frac{\partial T}{\partial y}\right)_b + \overline{S}V \tag{114}$$

The right and bottom faces are connected to interior cells. Hence, central differencing can be used to simplify the heat flux through these faces.

$$0 = k_r A_r \frac{T_R - T_P}{d_{PR}} - \left(kA\frac{\partial T}{\partial x}\right)_l + \left(kA\frac{\partial T}{\partial x}\right)_t - k_b A_b \frac{T_P - T_B}{d_{BP}} + \overline{S}V \tag{115}$$

Introduce the diffusive flux per unit $D = k/d$, as normal.

$$0 = D_r A_r (T_R - T_P) - \left(kA\frac{\partial T}{\partial x}\right)_l + \left(kA\frac{\partial T}{\partial x}\right)_t - D_b A_b (T_P - T_B) + \overline{S}V \tag{116}$$

If the boundary faces are at fixed temperatures of $T_{\text{wall, l}}$ on the left and $T_{\text{wall, t}}$ on the top, then central differencing can also be used on the boundary faces.

$$0 = D_r A_r (T_R - T_P) - 2D_l A_l (T_P - T_{\text{wall, l}}) + 2D_t A_t (T_{\text{wall, t}} - T_P) - D_b A_b (T_P - T_B) + \overline{S}V \tag{117}$$

Rearrange and collect the temperatures at the cell centroids (T_P, T_R, T_T).

$$\begin{aligned} T_P \left[0 + D_r A_r + D_b A_b + 0 + 2D_l A_l + 2D_t A_t\right] = {} & T_L \left[0\right] + T_R \left[D_r A_r\right] \\ & + T_B \left[D_b A_b\right] + T_T \left[0\right] + \overline{S}V \\ & + T_{\text{wall, l}} \left(2D_l A_l\right) + T_{\text{wall, t}} \left(2D_t A_t\right) \end{aligned} \tag{118}$$

The equation is in standard form:

$$a_p = a_L + a_R + a_T + a_B - S_p \tag{119}$$

With the following coefficients:

$$\begin{gathered} a_L = 0 \qquad a_R = D_r A_r \qquad a_B = D_b A_b \qquad a_T = 0 \\ S_p = -2D_l A_l - 2D_t A_t \qquad S_u = \overline{S}V + T_{\text{wall, 1}}(2D_l A_l) + T_{\text{wall, 2}}(2D_t A_t) \end{gathered} \tag{120}$$

Hence, when a cell has multiple boundary faces the contribution of those faces to $a_p = 0$. These faces then add their contribution to S_p and S_u instead.

When using summation notation, additional summation symbols are used in the definition of a_p and S_u to emphasise that additional contributions are added for each boundary face M that the cell has. Otherwise the equations are identical.

$$\begin{gathered} a_p T_P = \sum_N a_N T_N + S_u \\ a_p = \left(\sum_N \frac{k_N A_N}{|\boldsymbol{d}_{PN}|}|\hat{\boldsymbol{n}}|\right) + \left(\sum_M \frac{k_w A_w}{|\boldsymbol{d}_{LP}|/2}|\hat{\boldsymbol{n}}|\right) \qquad a_N = \frac{k_N A_N}{|\boldsymbol{d}_{PN}|}|\hat{\boldsymbol{n}}| \\ S_u = \overline{S}V + \sum_M T_{\text{wall}} \left(\frac{k_w A_w}{|\boldsymbol{d}_{LP}|/2}|\hat{\boldsymbol{n}}|\right) \end{gathered} \tag{121}$$

Cells with multiple boundary faces will be considered further in the example problem in this chapter.

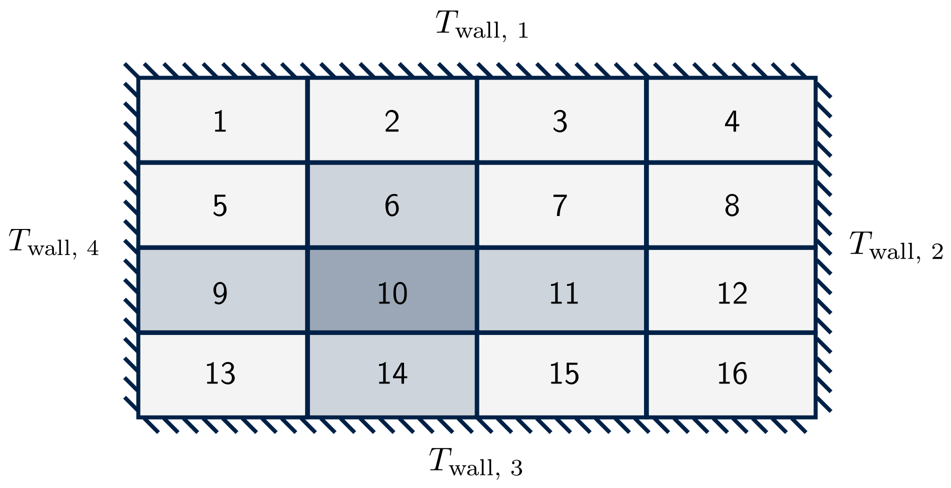


Figure 16: Cell numbering scheme for a 4x4 mesh. The mesh stencil around cell 10 is highlighted.

Assembling the Matrices

When applying the finite volume method in 2D and 3D, a cell numbering scheme must be adopted to ensure that the correct cells are connected together. Figure 16 shows an example numbering scheme for a mesh with 16 cells (4 x 4). For each cell in the mesh, it useful to visualise a stencil of 4 cells that surround the cell under consideration (also shown in Figure 16). This stencil allows us to identify the neighbouring cells more easily. As shown in the previous course, an algebraic equation is written for each cell in the mesh individually. For the mesh in Figure 16, the algebraic equations are:

$$
\begin{aligned}
a_{p1}T_1 &= a_{r2}T_2 + a_{b5}T_5 + S_{u1} \\
a_{p2}T_2 &= a_{l1}T_1 + a_{r3}T_3 + a_{b6}T_6 + S_{u2} \\
a_{p3}T_3 &= a_{l2}T_2 + a_{r4}T_4 + a_{b7}T_7 + S_{u3} \\
a_{p4}T_4 &= a_{l3}T_3 + a_{b8}T_8 + S_{u4} \\
a_{p5}T_5 &= a_{t1}T_1 + a_{r6}T_6 + a_{b9}T_9 + S_{u5} \\
a_{p6}T_6 &= a_{l2}T_2 + a_{t5}T_5 + a_{r7}T_7 + a_{b10}T_{10} + S_{u6} \\
a_{p7}T_7 &= a_{l3}T_3 + a_{t6}T_6 + a_{r8}T_8 + a_{b11}T_{11} + S_{u7} \\
a_{p8}T_8 &= a_{l4}T_4 + a_{t7}T_7 + a_{b12}T_{12} + S_{u8} \\
a_{p9}T_9 &= a_{r5}T_5 + a_{t10}T_{10} + a_{b13}T_{13} + S_{u9} \\
a_{p10}T_{10} &= a_{l6}T_6 + a_{r9}T_9 + a_{t11}T_{11} + a_{b14}T_{14} + S_{u10} \\
a_{p11}T_{11} &= a_{l7}T_7 + a_{r10}T_{10} + a_{t12}T_{12} + a_{b15}T_{15} + S_{u11} \\
a_{p12}T_{12} &= a_{l8}T_8 + a_{t11}T_{11} + a_{b16}T_{16} + S_{u12} \\
a_{p13}T_{13} &= a_{t9}T_9 + a_{r14}T_{14} + S_{u13} \\
a_{p14}T_{14} &= a_{l10}T_{10} + a_{t13}T_{13} + a_{r15}T_{15} + S_{u14} \\
a_{p15}T_{15} &= a_{l11}T_{11} + a_{t14}T_{14} + a_{r16}T_{16} + S_{u15} \\
a_{p16}T_{16} &= a_{l12}T_{12} + a_{t15}T_{15} + S_{u16}
\end{aligned} \tag{122}
$$

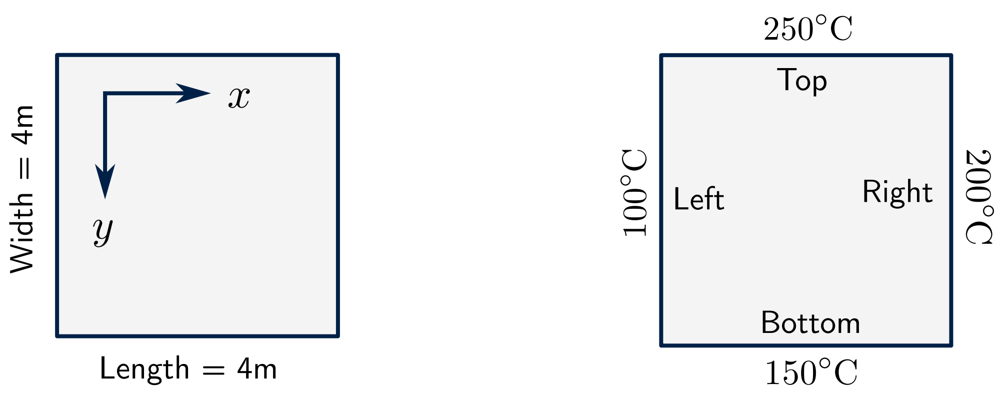


Figure 17: Example geometry used for heat diffusion in a 2D plate.

As before, the equations are then arranged into matrix form.

$$
\boldsymbol{A} = \begin{bmatrix}
a_{p1} & -a_{r2} & 0 & 0 & -a_{b5} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
-a_{l1} & a_{p2} & -a_{r3} & 0 & 0 & -a_{b6} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & -a_{l2} & a_{p3} & -a_{r4} & 0 & 0 & -a_{b7} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & -a_{l3} & a_{p4} & -a_{r5} & 0 & 0 & -a_{b8} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
-a_{t1} & 0 & 0 & -a_{l4} & a_{p5} & -a_{r6} & 0 & 0 & -a_{b9} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & -a_{t2} & 0 & 0 & -a_{l5} & a_{p6} & -a_{r7} & 0 & 0 & -a_{b10} & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & -a_{t3} & 0 & 0 & -a_{l6} & a_{p7} & -a_{r8} & 0 & 0 & -a_{b11} & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & -a_{t4} & 0 & 0 & -a_{l7} & a_{p8} & -a_{r9} & 0 & 0 & -a_{b12} & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & -a_{t5} & 0 & 0 & -a_{l8} & a_{p9} & -a_{r10} & 0 & 0 & -a_{b13} & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & -a_{t6} & 0 & 0 & -a_{l9} & a_{p10} & -a_{r11} & 0 & 0 & -a_{b14} & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & -a_{t7} & 0 & 0 & -a_{l10} & a_{p11} & -a_{r12} & 0 & 0 & -a_{b15} & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & -a_{t8} & 0 & 0 & -a_{l11} & a_{p12} & -a_{r13} & 0 & 0 & -a_{b16} \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -a_{t9} & 0 & 0 & -a_{l12} & a_{p13} & -a_{r14} & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -a_{t10} & 0 & 0 & -a_{l13} & a_{p14} & -a_{r15} & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -a_{t11} & 0 & 0 & -a_{l14} & a_{p15} & -a_{r16} \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -a_{t12} & 0 & 0 & -a_{l15} & a_{p16}
\end{bmatrix}
$$

$$
\begin{aligned}
\boldsymbol{T} = [T_1 \quad T_2 \quad T_3 \quad T_4 \quad T_5 \quad T_6 \quad T_7 \quad T_8 \quad T_9 \quad T_{10} \quad T_{11} \quad \ldots \\
\ldots \quad T_{12} \quad T_{13} \quad T_{14} \quad T_{15} \quad T_{16}]^T
\end{aligned} \tag{123}
$$

$$
\begin{aligned}
\boldsymbol{B} = [S_{u1} \quad S_{u2} \quad S_{u3} \quad S_{u4} \quad S_{u5} \quad S_{u6} \quad S_{u7} \quad S_{u8} \quad S_{u9} \quad S_{u10} \quad S_{u11} \quad \ldots \\
\ldots \quad S_{u12} \quad S_{u13} \quad S_{u14} \quad S_{u15} \quad S_{u16}]^T
\end{aligned} \tag{124}
$$

In the same manner as the 1D flow case, the $\boldsymbol{A}$ matrix has a diagonal banded structure. However, rather than having 2 off-diagonal bands (representing the left and right faces of the cell), the matrix now has 4 bands. The additional bands represent the connectivity to the top and bottom cells.

Example Problem - Heat Diffusion in a 2D Plate

Consider steady-state diffusion of heat in a 2D plate, as shown in Figure 17. The plate has a length of 4m, a width of 4m, a thickness of 0.1m and a thermal conductivity of 100 W/mK.

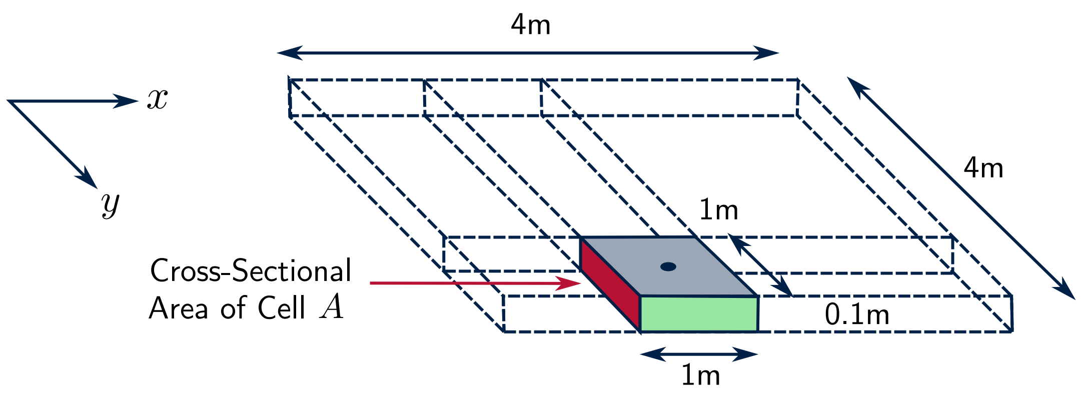


Figure 18: A diagram to show the dimensions of a cell in the mesh. The cross-sectional area that is used to calculate the heat fluxes in the x direction is highlighted in red and the cross-sectional area that is used to calculate the heat fluxes in the y direction is highlighted in green.

The temperature on the sides of the plate are fixed at 100°C, 150°C, 200°C and 250°C on the left, bottom, right and top faces respectively. There is a constant heat source of 1000 W/m^3 in the plate. The temperature field in the plate is governed by the 2D steady-state diffusion equation.

$$\frac{\partial}{\partial x}\left(k\frac{\partial T}{\partial x}\right)+\frac{\partial}{\partial y}\left(k\frac{\partial T}{\partial y}\right)+S=0 \tag{125}$$

Step 1: Divide the Geometry into a Mesh

For the example in Figure 17, divide the geometry into a mesh of 16 quadrilateral cells (4 cells in each direction). These cells have equal length and equal width. The length of each cell (L_{cell}) is given by:

$$L_{\text{cell}}=\frac{L}{N}=\frac{4}{4}=1\text{m} \tag{126}$$

Because the cells are uniformly distributed and have equal size, the distance between cell centroids d is equivalent to the length of the cells. Hence:

$$d_{LP}=d_{PR}=d_{BP}=d_{PT}=d=1\text{m} \tag{127}$$

As shown in Figure 18, the cross-sectional area of all of the cells (in the x and y directions) is:

$$A=1.0*0.1=0.1\text{m}^2 \tag{128}$$

Step 2: Assign Material Properties

The thermal conductivity k and the cross-sectional area A are the same for every cell in the mesh. Hence, the parameter DA is given by:

$$DA=\frac{kA}{d}=\frac{100*0.1}{1}=10\ [\text{W/K}] \tag{129}$$

The heat source in each cell is given by:

$$\overline{S}V=\overline{S}L_{\text{cell}}Wt=1000*1*1*0.1=100\ [\text{W}] \tag{130}$$

Step 3: Calculate Matrix Coefficients

Now calculate the coefficients required to assemble the matrices. The most straightforward way is to fill in the table of coefficients:

Cell	a_L	a_R	a_b	a_t	S_p	S_u	a_p
1	0	10	10	0	-40	7100	60
2	10	10	10	0	-20	5100	50
3	10	10	10	0	-20	5100	50
4	10	0	10	0	-40	9100	60
5	0	10	10	10	-20	2100	50
6	10	10	10	10	0	100	40
7	10	10	10	10	0	100	40
8	10	0	10	10	-20	4100	50
9	0	10	10	10	-20	2100	50
10	10	10	10	10	0	100	40
11	10	10	10	10	0	100	40
12	10	0	10	10	-20	4100	50
13	0	10	0	10	-40	5100	60
14	10	10	0	10	-20	3100	50
15	10	10	0	10	-20	3100	50
16	10	0	0	10	-40	7100	60

Alternatively, the summation notation summary table can be filled in (noting that an additional source $\overline{S}V = 100$W is required in each cell):

Face Type	a_P	a_N	S_u
Interior	10	10	0
Dirichlet (Left)	20	0	2000
Dirichlet (Bottom)	20	0	3000
Dirichlet (Right)	20	0	4000
Dirichlet (Top)	20	0	5000

Step 4: Assemble the Matrices

Assign the coefficients to their correct location in the matrix.

$$\boldsymbol{A} = \begin{bmatrix} 60 & -10 & 0 & 0 & -10 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -10 & 50 & -10 & 0 & 0 & -10 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -10 & 50 & -10 & 0 & 0 & -10 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -10 & 60 & 0 & 0 & 0 & -10 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -10 & 0 & 0 & 0 & 50 & -10 & 0 & 0 & -10 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -10 & 0 & 0 & -10 & 40 & -10 & 0 & 0 & -10 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -10 & 0 & 0 & -10 & 40 & -10 & 0 & 0 & -10 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -10 & 0 & 0 & -10 & 50 & 0 & 0 & 0 & -10 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -10 & 0 & 0 & 0 & 50 & -10 & 0 & 0 & -10 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -10 & 0 & 0 & -10 & 40 & -10 & 0 & 0 & -10 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -10 & 0 & 0 & -10 & 40 & -10 & 0 & 0 & -10 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & -10 & 0 & 0 & -10 & 50 & 0 & 0 & 0 & -10 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -10 & 0 & 0 & 0 & 60 & -10 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -10 & 0 & 0 & -10 & 50 & -10 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -10 & 0 & 0 & -10 & 50 & -10 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -10 & 0 & 0 & -10 & 60 \end{bmatrix} \tag{131}$$

$$\boldsymbol{T} = [T_1 \quad T_2 \quad T_3 \quad T_4 \quad T_5 \quad T_6 \quad T_7 \quad T_8 \quad T_9 \quad T_{10} \quad T_{11} \quad \ldots$$
$$\ldots \quad T_{12} \quad T_{13} \quad T_{14} \quad T_{15} \quad T_{16}]^T \tag{132}$$

$$\boldsymbol{B} = [7100 \quad 5100 \quad 5100 \quad 9100 \quad 2100 \quad 100 \quad 100 \quad 4100 \quad 2100 \quad 100 \quad 100 \quad \ldots$$
$$\ldots \quad 4100 \quad 5100 \quad 3100 \quad 3100 \quad 7100]^T \tag{133}$$

Step 5: Solve the Equations

Now that the matrices have been assembled, the matrix equation can solved with an appropriate *iterative method*. As with the previous course, different algorithms to solve the matrix equation $\boldsymbol{AT} = \boldsymbol{B}$ will not be considered and the default algorithms used by Excel and Python will be used instead.

Run the Example Problem Yourself!

Now, open either the Excel spreadsheets, the Python source code or the MATLAB code and solve the problem yourself.

Excel	`solve2DDiffusionEquation.xlsx`
Python	`solve2DDiffusionEquation.py`
MATLAB	`solve2DDiffusionEquation.m`

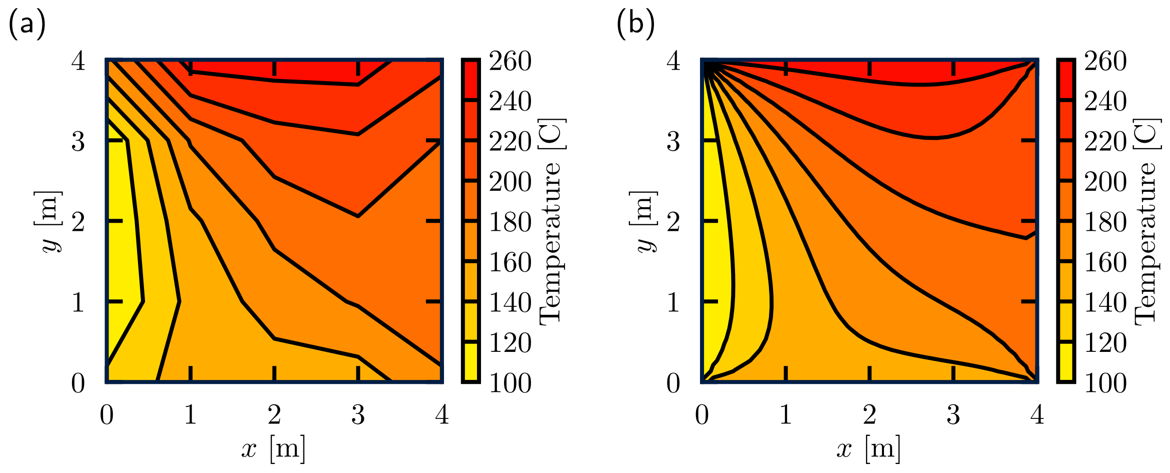


Figure 19: Temperature variation across the 2D plate with (a) a coarse mesh and (b) a fine mesh.

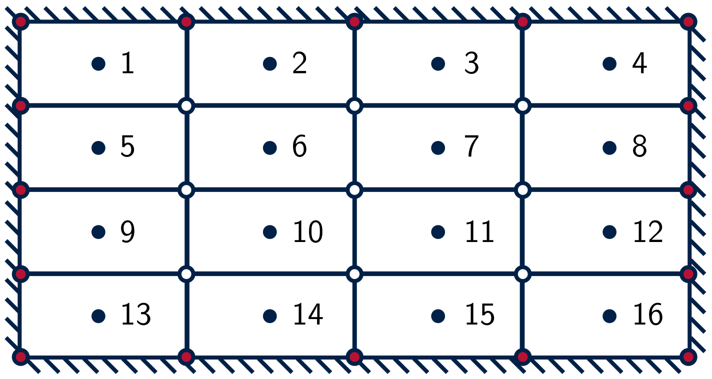


Figure 20: A diagram to show the centroids numbering scheme used in the mesh. The centroids are shown as dark blue circles, the boundary nodes as red circles and the interior nodes as white circles.

Results

Figure 19 shows the temperature variation across the plate with (a) a coarse mesh and (b) a fine mesh. To plot the temperature variation as a contour plot, it is more convenient to plot the temperature at the mesh **nodes** rather than the mesh **centroids**. As shown in Figure 20, this is because the boundary temperatures are specified at the nodes (shown in red) and these would be missed if a contour plot was generated from the mesh centroids (shown in blue).

The CFD solution is computed at the cell centroids. Hence, the interior nodal values (shown in white in Figure 20) have to be computed by interpolating between the centroid values (shown in blue). This interpolation has already been carried out in the Excel spreadsheet and the python source code for you.

Heat Balancing

Now that a CFD solution has been computed, a heat balance can be written for every cell in the mesh. In the same manner as the previous chapter, the sum of the heat fluxes into the cell and the heat generated in the cell must be equal to zero for energy to be conserved.

$$\begin{aligned}0 &= k_r A_r\left(\frac{T_R - T_P}{d_{PR}}\right)n_x + k_l A_l\left(\frac{T_P - T_L}{d_{LP}}\right)n_x \\ &+ k_t A_t\left(\frac{T_T - T_P}{d_{PT}}\right)n_y + k_b A_b\left(\frac{T_P - T_B}{d_{BP}}\right)n_y + \overline{S}V\end{aligned} \tag{134}$$

From Fourier's Law, the heat conduction across each face is given by:

$$\begin{aligned}Q_l &= -k_l A_l\left(\frac{T_P - T_L}{d_{LP}}\right)n_x \qquad Q_r = -k_r A_r\left(\frac{T_R - T_P}{d_{PR}}\right)n_x \\ Q_b &= -k_b A_b\left(\frac{T_P - T_B}{d_{BP}}\right)n_y \qquad Q_t = -k_t A_t\left(\frac{T_T - T_P}{d_{PT}}\right)n_y\end{aligned} \tag{135}$$

The negative sign is required to ensure that the heat flows in the opposite direction to the temperature gradient and the unit normal vectors ensure that positive heat flues are out of the cell. By substituting Fourier's Law (equation 135) into the finite volume discretisation (equation 134), the heat balance for each cell can be written in concise form as:

$$-Q_r - Q_l - Q_t - Q_b + \overline{S}V = 0 \tag{136}$$

$$\sum_N (-Q) + \overline{S}V = 0 \tag{137}$$

The error in the heat flux balance for 2D quadrilateral cells can therefore be written as:

$$\text{Error} = -Q_r - Q_l - Q_t - Q_b + \overline{S}V \tag{138}$$

The table below summarises the heat balance for every cell in the mesh. The heat fluxes across the boundaries of the plate are highlighted in blue.

Cell Heat Flux Balance Table

Cell	Q_l [W]	Q_r [W]	Q_t [W]	Q_b [W]	$\overline{S}V$ [W]	Error
1	**1575.0**	-427.3	**-1425.0**	377.3	100	0
2	427.3	-109.2	**-570.4**	352.3	100	0
3	109.2	74.4	**-351.9**	268.3	100	0
4	-74.4	**499.4**	**-500.6**	175.6	100	0
5	**820.4**	-452.3	-377.3	109.2	100	0
6	452.3	-193.3	-352.3	193.3	100	0
7	193.3	-18.3	-268.3	193.3	100	0
8	18.3	**148.1**	-175.6	109.2	100	0
9	**601.9**	-368.3	-109.2	-24.4	100	0
10	368.3	-193.3	-193.3	118.3	100	0
11	192.3	-102.3	-193.3	202.3	100	0
12	102.3	**-70.4**	-109.2	177.3	100	0
13	**650.6**	-225.6	24.4	**-349.4**	100	0
14	225.6	-109.2	-118.3	**101.9**	100	0
15	109.2	-127.3	-202.3	**320.4**	100	0
16	127.3	**-425.0**	-177.3.	**575.0**	100	0

The cell heat flux balance table can also be used to calculate the total heat flux out of each of the faces of the plate (across all the boundaries). By summing the heat flux out of each of the boundary faces (highlighted in blue in the table above):

Heat Flux Summary Table

Boundary	Heat Flux Out [W]	Total [W]
Left	1575.0 + 820.4 + 601.9 + 650.6	3647.9
Right	499.4 + 148.1 -70.4 -425.0	152.1
Bottom	-349.4 + 101.9 +320.4 +575.0	647.9
Top	-1425 -570.4 - 351.9 - 500.6	-2847.9
Total		**1600**

Once again positive heat fluxes indicate heat travelling out of the plate across the boundary, while negative heat fluxes indicate heat travelling into the plate. As the coldest face, the left face of the plate ($100°$C) experiences the largest heat flux out of the plate. Conversely, the top face of the plate experiences a negative heat flux as heat is drawn into the plate by the hot boundary temperature ($250°$C). The total of these heat fluxes (1600W) balances the total heat generated in the plate (1600W), so energy is conserved. As the mesh is refined, the same total heat flux (1600W) is conserved. However, the temperature distribution in the plate and the heat flux across each of the individual cells changes as a result of the mesh refinement.

Chapter 3

Wall Functions

3 Wall Functions

In the second order finite volume method, the flow variables (temperature, velocity, pressure etc.) vary linearly across the cell. Hence, when there are steep gradients in the flow variables, many cells are required to resolve the profiles with sufficient accuracy. As shown in Figure 21, the gradients are particularly steep close to the wall. This is due to the boundary layer that develops on a solid surface in contact with the flow. In order to resolve the steep gradients near the wall accurately, many thin cells are required normal to the wall. The gradients across the cell that is immediately adjacent to the wall are particularly important, as the gradients across this cell determine the wall shear stress and the wall heat flux. In general, the wall shear stress τ_w and the wall heat flux q_w are given by Newton's Law of Viscosity and Fourier's Law of Heat Conduction:

$$\frac{\tau_w}{\rho} = -\nu \left.\frac{\partial U}{\partial y}\right|_{y=0} \qquad \frac{q_w}{\rho c_p} = -\alpha \left.\frac{\partial T}{\partial y}\right|_{y=0} \tag{139}$$

where ρ is the fluid density, ν is the kinematic viscosity of the fluid, α is the thermal diffusivity of the fluid, U is the velocity component parallel with the wall, T is the temperature and y is the direction normal to the wall. As the profile is non-linear (the gradient changes with distance from the wall), the gradient is evaluated at the wall ($y = 0$). The minus sign is required because the wall shear stress acts in the opposite direction to the velocity profile and heat flux is in the opposite direction to the temperature gradient (high temperature to low temperature).

When generating a mesh for CFD simulations, a fine mesh should be used with many thin cells normal to the wall. If the mesh is sufficiently fine, then the piecewise linear variation that is computed by the CFD code has a sufficient number of nodes to accurately reproduce the real flow profile normal to the wall. However, for some flow simulations it is not possible to use many thin cells normal to the wall, as this would lead to a high cell count (and sometimes poor cell quality). For these flow scenarios, the cell adjacent to the wall is too large to accurately reproduce the flow profile. As shown in Figure 22, the velocity and temperature gradients at the wall ($\partial U/\partial y$ and $\partial T/\partial y$) will be incorrect if the cell is too large. This will lead to inaccuracies in the wall shear stress and wall heat flux (equation 139), unless the CFD code is corrected appropriately. The focus of this Chapter is the method used to correct the flow at the wall when the cells adjacent to the wall are too large.

Background Theory

In order to compute the correct wall shear stress, the **product** of the near wall kinematic viscosity and the velocity gradient at the wall must be correct.

$$\frac{\tau_w}{\rho} = -\nu \left.\frac{\partial U}{\partial y}\right|_{y=0} = -\text{Kinematic Viscosity} \times \text{Velocity Gradient at the Wall} \tag{140}$$

Hence, even though the gradient at the wall ($\partial U/\partial y$) is incorrect, if the near wall kinematic viscosity (ν) is modified appropriately, then the wall shear stress will be correct. This approach of modifying the near wall kinematic viscosity and the near wall thermal diffusivity is used by the majority of modern CFD codes to correct the solution when the cells are too large.

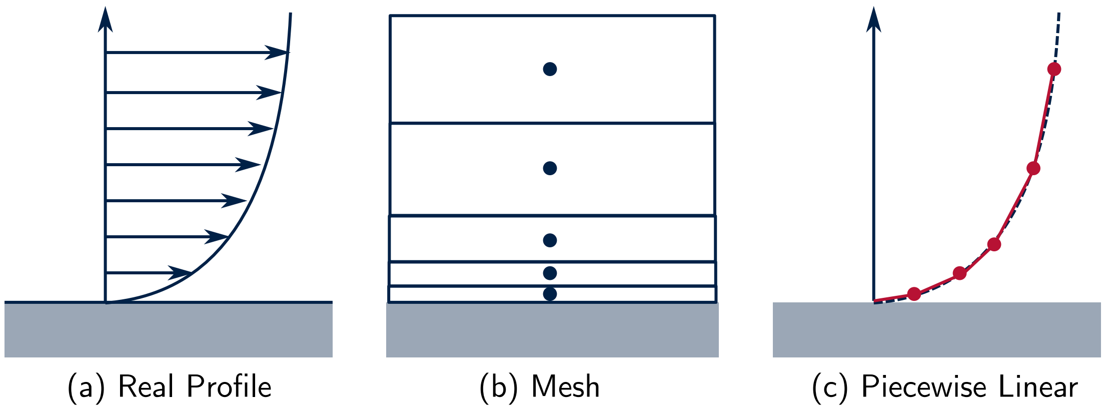


Figure 21: A diagram to show the approximation of a real flow profile (velocity or temperature) by a finite volume mesh. The finite volume method uses a piecewise linear approximation of the real profile.

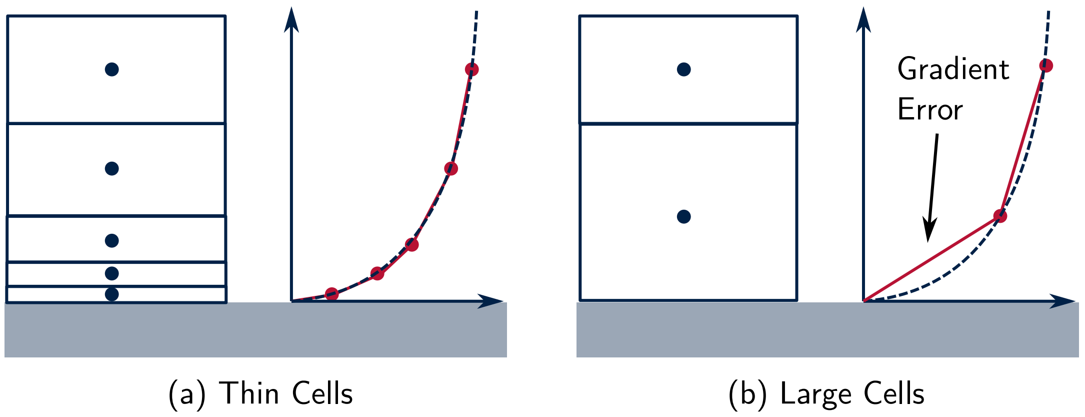


Figure 22: A diagram to show the difference in the piecewise linear approximation of the near wall velocity or temperature profile when (a) thin and (b) large cells are used. The gradient at the wall is incorrect when large cells are used.

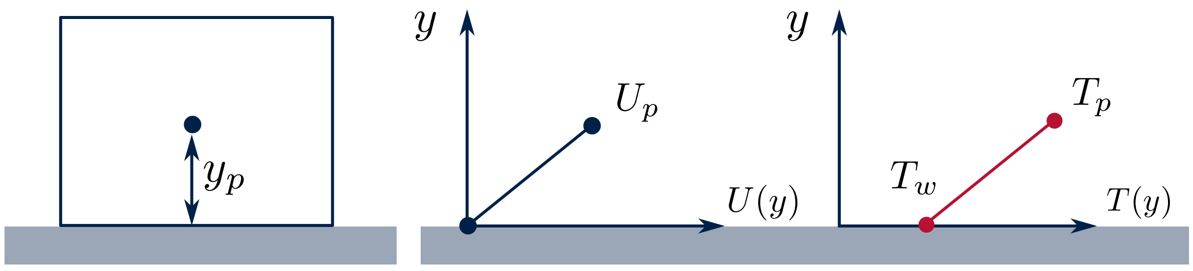


Figure 23: A diagram to show the linear variation of velocity and temperature across the wall adjacent cell in the mesh.

The velocity and temperature variation across the cell are **always** linear. Hence, as shown in Figure 23, the wall shear stress and wall heat flux computed by the CFD code are:

$$\tau_w = -\rho \nu_w \frac{U_p}{y_p} \qquad q_w = -\rho c_p \alpha_w \left(\frac{T_p - T_w}{y_p} \right) \tag{141}$$

where y_p is the distance from the wall to the wall adjacent cell centroid, U_p is the velocity at the cell centroid, T_p is the temperature at the cell centroid and T_w is the temperature of

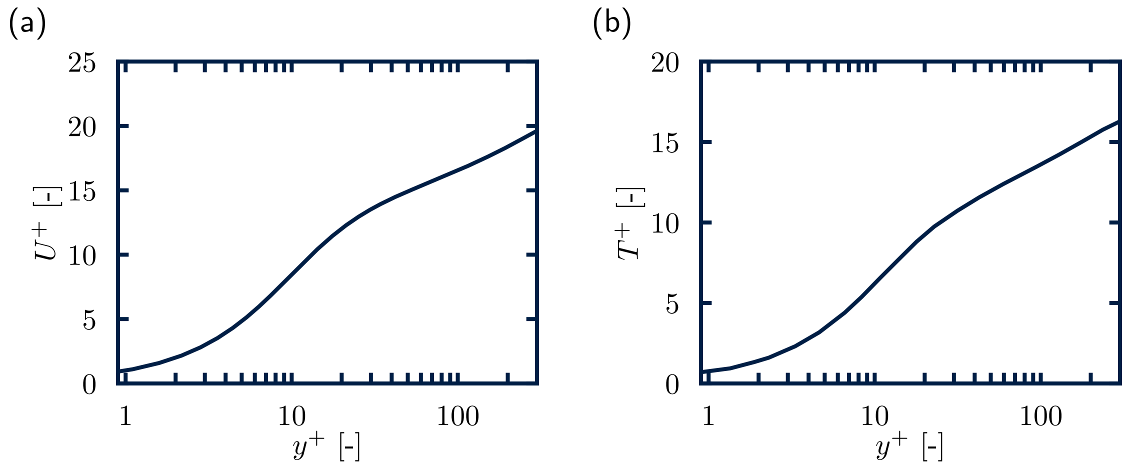


Figure 24: Experimental measurements of (a) the velocity profile and (b) the temperature profile normal to the wall in a turbulent flow of air over a flat plate.

the wall. The kinematic viscosity ν_w and the thermal diffusivity α_w are now denoted with a subscript w and will be described as the **near wall** kinematic viscosity (ν_w) and the **near wall** thermal diffusivity (α_w). The change in notation is to emphasise that these quantities have been modified. The modification is required to ensure that the product of these quantities and the (incorrect) velocity and temperature gradients produces the correct wall shear stress and wall heat flux. The modification to the near wall kinematic viscosity and the near wall thermal diffusivity is called a **wall function**, as the modification is only carried out in the cells that are adjacent to the wall.

At this stage it should be emphasised that a wall function is a modification to ν_w and α_w in the wall adjacent cell. The wall function is not a modification to the velocity and temperature profiles (despite sometimes being called a velocity or temperature wall function in the literature) because the variation across the cell in a CFD code is always linear. The modifications to α_w and ν_w may result in a change in the velocity and temperature solution fields, but the velocity and temperature in the wall adjacent cell are not set directly by a wall function.

Wall Functions for ν_w and α_w

In order to propose wall functions for ν_w and α_w, the real variation of velocity and temperature between the cell centroid and the wall is required first. These are the actual velocity and temperature profiles that we are trying to model and are non-linear. Figure 24 shows the real variation of velocity and temperature normal to the wall, which were extracted from experimental measurements of fully developed turbulent flow between 2 parallel plates. Further experimental measurements found this profile to be relatively universal (independent of Reynolds number and streamwise pressure gradient) close to the wall. Hence, the profile is often called **'The Universal Law of the Wall'**. Mathematically, the experimental data in Figure 24 can be reasonably approximated by the following function:

$$U^+ = \begin{cases} y^+ & y^+ < 11.25 \\ \frac{1}{\kappa}\log\left(Ey^+\right) & y^+ > 11.25 \end{cases} \tag{142}$$

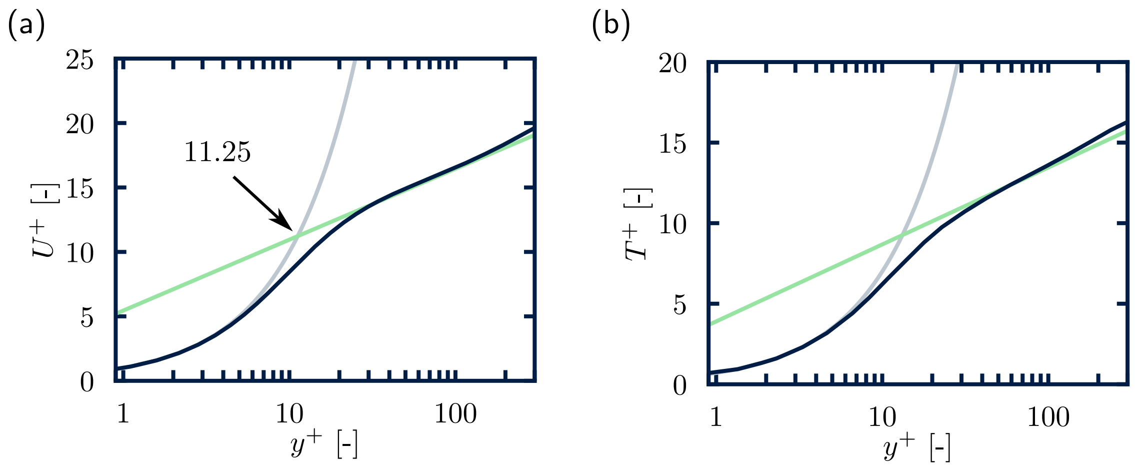


Figure 25: Experimental measurements of (a) the velocity profile and (b) the temperature profile normal to the wall in a turbulent flow of air. The light blue and light green lines show the mathematical functions that are fitted to the data for $y^+ < y_L^+$ and $y^+ > y_L^+$ respectively.

where $\kappa = 0.4187$ and $E = 9.793$ are empirical constants that were fitted to the data. As shown in Figure 25, these profiles are less accurate at reproducing the experimental data when $5 < y^+ < 30$ (the buffer region). Hence, it is normally recommended to ensure that $y^+ < 5$ or $y^+ > 30$ for the final solution to be accurate.

The two profiles intersect where:

$$y^+ = \frac{1}{\kappa}\log\left(Ey^+\right) \tag{143}$$

$$y^+ - \frac{1}{\kappa}\log\left(Ey^+\right) = 0 \qquad \longrightarrow \qquad f(y^+) = 0 \tag{144}$$

Substituting in the empirical coefficients $E = 9.7983$ and $\kappa = 0.4187$ and solving for y^+ with a root-finding algorithm (like the bisection method or Newton-Raphson method) results in an intersection point of $y^+ = 11.25$. This process of root finding will be demonstrated later in the example problem. A similar mathematical function can also be fitted to the temperature profile.

$$T^+ = \begin{cases} Pr\ y^+ & y^+ < y_L^+ \\ Pr_t\left(\frac{1}{\kappa}\log\left(Ey^+\right) + P\right) & y^+ > y_L^+ \end{cases} \tag{145}$$

where $Pr = \nu/\alpha$ is the molecular Prandtl number, Pr_t is the turbulent Prandtl number (0.85) and P is an empirical function of Pr and Pr_t.

$$P = 9.24\left[\left(\frac{Pr}{Pr_t}\right)^{3/4} - 1\right]\left[1 + 0.28e^{-0.007Pr/Pr_t}\right] \tag{146}$$

The reason for the differences between the velocity and temperature wall functions is that (unlike the velocity profile) the dimensionless temperature profile is **different for different fluids**. The difference is captured by the molecular Prandtl number Pr. For water and air for example, the molecular Prandtl numbers are:

	$Pr = \nu/\alpha$
Air	0.71
Water	5.68

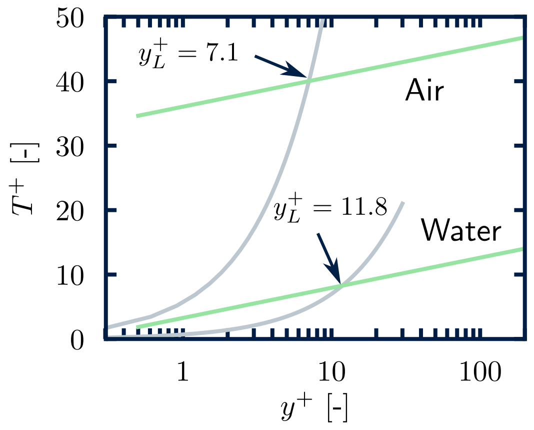


Figure 26: A diagram to shown the differences in the temperature profiles for water and air.

A value of $Pr < 1$ indicates that thermal energy diffuses into the flow faster than momentum. This means that the thermal boundary layer is thinner than the velocity boundary layer when $Pr < 1$. A consequence of the temperature profile being different for different fluids is that the intersection between the 2 profiles does not occur at 11.25 (as it does for the velocity profile). The intersection point y_L^+ can be determined by equating the profiles in equation 145.

$$Pr\ y^+ = Pr_t \left(\frac{1}{\kappa}\log\left(Ey^+\right) + P\right) \tag{147}$$

$$Pr\ y^+ - Pr_t \left(\frac{1}{\kappa}\log\left(Ey^+\right) + P\right) = 0 \qquad \longrightarrow \qquad f(y^+) = 0 \tag{148}$$

Substitute in the values of Pr for the fluid of interest. Then solve the non-linear equation for y^+ using a root-finding algorithm (this process will be demonstrated later in the example problem). This is the intersection point of the two curves (y_L^+) and is different for different fluids.

	Pr	y_L^+
Air	0.71	7.1
Water	5.68	11.8

Figure 26 compares the temperature profiles for water and air. Unlike the velocity profile (which is the same for water and air), the profiles are different and have a different intersection point. This difference in the profiles will also be reflected in the wall function for α_w. Once again it should be emphasised that the mathematical functions for velocity and temperature are **not applied directly** in the CFD code. The equations are actually used to derive equations for the near wall kinematic viscosity ν_w and the near wall thermal diffusivity α_w. This derivation will be demonstrated in the next section. It is these wall functions for ν_w and α_w that are actually applied in the CFD code.

y^+ and $y^\star$

The velocity and temperature profiles in equations 142 and 145 are expressed in dimensionless wall units.

$$U^+ = \frac{U}{u_\tau} \qquad y^+ = \frac{\rho u_\tau y}{\mu} \qquad T^+ = \frac{(T_w - T_p)\rho c_p u_\tau}{q_w} \tag{149}$$

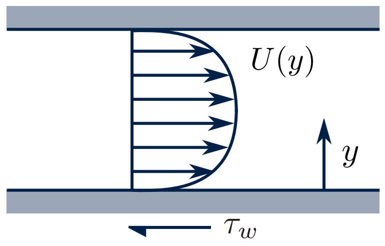


Figure 27: A schematic diagram of the experimental set up that was used to derive the velocity profiles normal to the wall.

The reason for using dimensionless units is so that the profiles are universal and can be applied to any flow scenario (flow over an aerofoil, flow in a pipe, flow over a flat plate etc.), as long as we are close to the wall. This is a reasonable theory, as the flow close to the wall at small scales should be universal, regardless of the freestream flow and the overall shape of the geometry.

However, to define these dimensionless groups, appropriate choices of velocity scale (u_τ) and length scale (y) are required. Since the flow is close to the wall, the appropriate length scale y is the distance normal to the wall. For the velocity scale, it would not be appropriate to use the free-stream velocity (U_∞), as the flow is close to the wall where viscous effects are dominant and the velocity is much lower than the freestream velocity. However, the velocity at the wall itself is zero due to the no-slip condition, so this cannot be used as a velocity scale either. The original approach in the experiments was to chose a velocity scale based on the (square-root of the) wall shear stress. This velocity scale (u_τ) is called the friction-velocity.

$$u_\tau = \sqrt{\frac{|\tau_w|}{\rho}} \qquad [\text{m/s}] \tag{150}$$

However, the problem with using a friction velocity based on the wall shear stress is that the wall shear stress is zero at separation points. This would result in a friction velocity of 0, which would create difficulties for the CFD code. An alternative is to use a friction velocity based on the turbulent kinetic energy k (which would have been more difficult to attain in the experiments).

$$u_\tau = \sqrt{C_\mu^{1/2} k} \qquad C_\mu = 0.09 \tag{151}$$

A dimensionless velocity scale based on the turbulent kinetic energy is denoted as $y^\star$ rather than y^+ by CFD codes.

$$U^\star = \frac{U}{u_\tau} \qquad y^\star = \frac{\rho u_\tau y}{\mu} \qquad T^\star = \frac{(T_w - T_p)\rho c_p u_\tau}{q_w} \qquad u_\tau = \sqrt{C_\mu^{1/2} k} \tag{152}$$

In most scenarios, y^+ and $y^\star$ are almost identical and either can be used. However, the majority of modern CFD codes prefer to use $y^\star$ in their calculations.

Deriving a Wall Function for ν_w

We now have a mathematical model for the velocity profile normal to the wall:

$$U^+ = \begin{cases} y^+ & y^+ < 11.25 \\ \frac{1}{\kappa}\log\left(Ey^+\right) & y^+ > 11.25 \end{cases} \tag{153}$$

Replace the dimensionless variables with the actual variables ($U^+ = U/u_\tau$ and $y^+ = \rho u_\tau y/\mu$):

$$\begin{aligned} \frac{U}{u_\tau} &= \frac{\rho u_\tau y}{\mu} \qquad y^+ < 11.25 \\ \frac{U}{u_\tau} &= \frac{1}{\kappa}\log\left(E\frac{\rho u_\tau y}{\mu}\right) \qquad y^+ > 11.25 \end{aligned} \tag{154}$$

The trick to calculating the wall shear stress τ_w from these profiles is to write:

$$\frac{U}{u_\tau} = \frac{U u_\tau}{u_\tau^2} \tag{155}$$

Then recall that $u_\tau = \sqrt{|\tau_w|/\rho}$ and therefore $u_\tau^2 = -\tau_w/\rho$

$$\frac{U}{u_\tau} = \frac{U u_\tau}{u_\tau^2} = -\frac{U\rho u_\tau}{\tau_w} \tag{156}$$

The velocity profile then becomes:

$$\begin{aligned} -\frac{U\rho u_\tau}{\tau_w} &= \frac{\rho u_\tau y}{\mu} \qquad y^+ < 11.25 \\ -\frac{U\rho u_\tau}{\tau_w} &= \frac{1}{\kappa}\log\left(E\frac{\rho u_\tau y}{\mu}\right) \qquad y^+ > 11.25 \end{aligned} \tag{157}$$

Rearrange for the wall shear stress (τ_w):

$$\begin{aligned} \tau_w &= -\mu\frac{U}{y} \qquad y^+ < 11.25 \\ \tau_w &= -\frac{U\rho u_\tau}{\frac{1}{\kappa}\log\left(E\frac{\rho u_\tau y}{\mu}\right)} \qquad y^+ > 11.25 \end{aligned} \tag{158}$$

This equation gives the **real** wall shear stress that is exhibited by a turbulent velocity profile over a flat plate. In a CFD code, the velocity profile between the wall adjacent cell centroid and the wall is always linear. As shown in Figure 23, the velocity at the cell centroid is U_p and the cell is a height y_p normal to the wall. Hence, the wall shear stress **in the CFD code** is always:

$$\tau_w = -\rho\nu \left.\frac{\partial U}{\partial y}\right|_{y=0} = -\rho\nu_w\frac{U_p}{y_p} \tag{159}$$

To ensure that the CFD code always computes the correct wall shear stress, equate equation 159 with equation 158 (noting that the velocity and wall normal distance in equation 158 are now evalauted at the cell centroid, $U = U_p$ and $y = y_p$).

$$\begin{aligned} -\rho\nu_w\frac{U_p}{y_p} &= -\rho\nu\frac{U_p}{y_p} \qquad y^+ < 11.25 \\ -\rho\nu_w\frac{U_p}{y_p} &= -\frac{U_p\rho u_\tau}{\frac{1}{\kappa}\log\left(E\frac{\rho u_\tau y_p}{\mu}\right)} \qquad y^+ > 11.25 \end{aligned} \tag{160}$$

Rearrange for ν_w.

$$\begin{aligned} \nu_w &= \nu \qquad y^+ < 11.25 \\ \nu_w &= \frac{u_\tau y_p}{\frac{1}{\kappa}\log\left(E\frac{\rho u_\tau y_p}{\mu}\right)} \qquad y^+ > 11.25 \end{aligned} \tag{161}$$

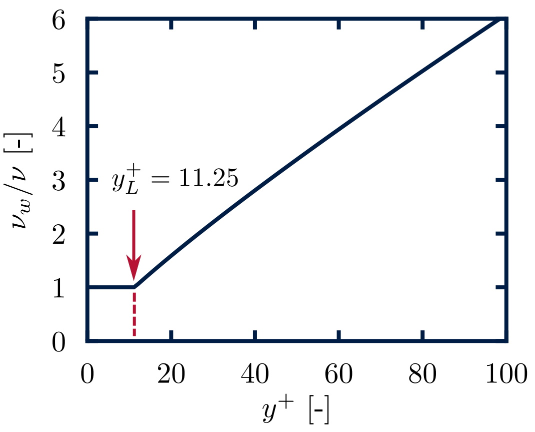
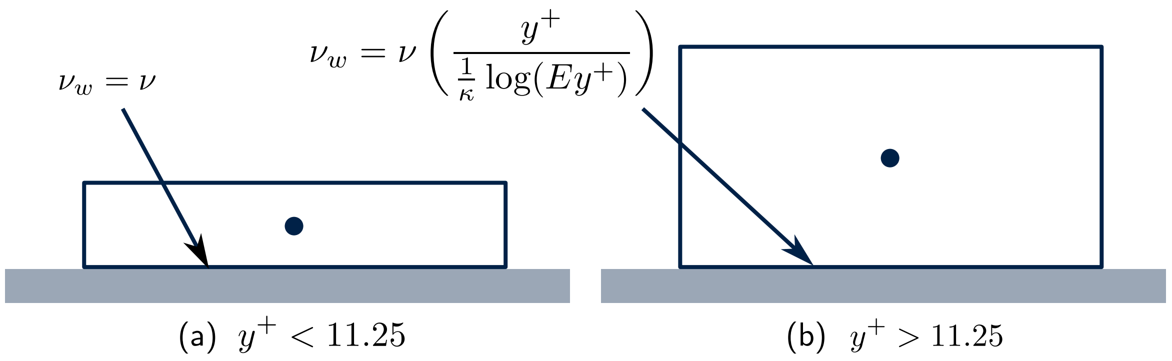


Figure 28: The variation of near wall kinematic viscosity ν_w with y^+



$\nu_w = \nu$
$\nu_w = \nu \left(\frac{y^+}{\frac{1}{\kappa}\log(Ey^+)} \right)$
(a) $y^+ < 11.25$
(b) $y^+ > 11.25$

Figure 29: A diagram to shown the difference in wall treatment when (a) $y^+ < 11.25$ and (b) $y^+ > 11.25$

Make a final substitution ($y^+ = y_p u_\tau/\nu$) to simplify the equation:

$$\boxed{\nu_w = \begin{cases} \nu & y^+ < 11.25 \\ \nu \left(\dfrac{y^+}{\frac{1}{\kappa}\log\left(Ey^+\right)} \right) & y^+ > 11.25 \end{cases}} \tag{162}$$

Equation 162 is the **wall function** for the near wall kinematic viscosity ν_w and is plotted in Figure 28. Physically, the equation states that if the wall adjacent cell is thin enough ($y^+ < 11.25$), then the near wall kinematic viscosity is set equal to the molecular viscosity of the fluid (ν). This will result in the correct wall shear stress as the true velocity profile is linear. However, if the cell is large ($y^+ > 11.25$), then the CFD code assumes that the velocity profile between the wall adjacent cell centroid and the wall is linear. This is not correct, as the real velocity profile is non-linear. However, if the near wall kinematic viscosity is increased using equation 162, the the **product** of the near wall kinematic viscosity and the (incorrect) velocity gradient will yield the correct wall shear stress.

$$\frac{\tau_w}{\rho} = -\text{Near Wall Kinematic Viscosity} \times \text{Velocoity Gradient} \tag{163}$$

Figure 29 shows a diagram to illustrate this process. The astute reader will notice that

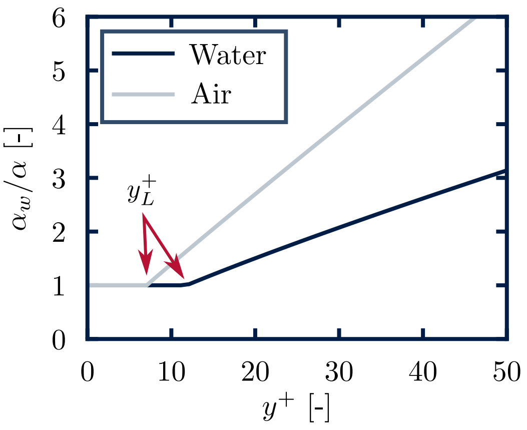


Figure 30: The variation of the near wall thermal diffusivity α_w with y^+ for air and water.

equation 162 can be written concisely as:

$$\nu_w = \nu * \left[\frac{y^+}{f(y^+)}\right] \tag{164}$$

where $f(y^+)$ is given by equation 153. Hence, if a new function for U^+ is proposed (replacing equation 142), then the wall function for ν_w can be deduced directly from the above equation.

It should also be noted that $y^\star$ can be used in place of y^+ in the wall function for near wall kinematic viscosity.

$$\nu_w = \begin{cases} \nu & y^\star < 11.25 \\ \nu\left(\dfrac{y^\star}{\frac{1}{\kappa}\log\left(Ey^\star\right)}\right) & y^\star > 11.25 \end{cases} \tag{165}$$

The majority of CFD codes (ANSYS Fluent, ANSYS CFX, Star CCM+) use the $y^\star$ formulation instead of y^+. OpenFOAM offers the choice of both: `nutUWallFunction` (for y^+) and `nutkWallFunction` (for $y^\star$).

Wall Function for α_w

The wall function for the near wall thermal diffusivity (α_w) takes a similar form to ν_w:

$$\boxed{\alpha_w = \begin{cases} \alpha & y^+ < y_L^+ \\ \alpha\left[\dfrac{Pr\ y^+}{Pr_t\left(\frac{1}{\kappa}\log\left(Ey^+\right)+P\right)}\right] & y^+ > y_L^+ \end{cases}} \tag{166}$$

For brevity, the derivation will not be provided here, as it follows an identical process to the wall function for ν_w in the previous section. The wall function for α_w is shown in Figure 30 for air and water. In the same manner as ν_w, for $y^+ < y_L^+$, α_w takes the value of the molecular thermal diffusivity α. For $y^+ > y_L^+$, the thermal diffusivity is increased, to ensure that the product of the near wall thermal diffusivity and the temperature gradient gives the correct wall heat flux. Notice (in Figure 30) that air and water have different values of y_L^+

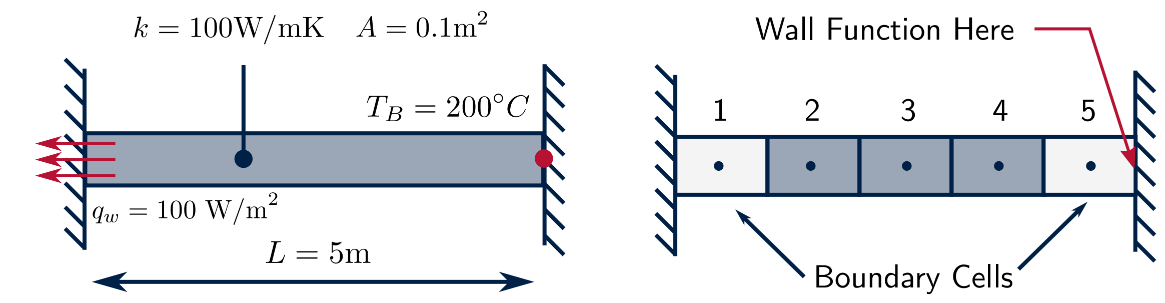


Figure 31: An example problem to demonstrate 1D heat diffusion in a bar. A wall function is applied at the right hand end (where the temperature is fixed).

and the rate of increase of α_w with y^+ is greater for water than air. In general, the shape of the wall function will be different for every fluid, depending on Pr. The wall function for ν_w however, is universal for all fluids.

Now that the wall functions for ν_w and α_w have been presented, the remainder of the chapter will demonstrate how the wall functions are incorporated into CFD codes. This demonstration will be carried through an example problem.

Example Problem: Heat Diffusion in a Bar

Consider 1D steady-state diffusion of heat in a bar, as shown in Figure 31. The left end of the bar has a fixed heat flux (q_{wall}) of 100 W/m^2, while the right end of the bar is at a fixed temperature of (T_B) 200°C. There is a constant heat source ($\overline{S}$) of 1000 W/m3 applied to the bar. The bar has a density of 8000 kg/m^3 and a specific heat capacity of 500 J/kg K. For ease of comparison, this example problem has the same geometry, mesh and boundary conditions as the previous chapter. However, unlike the previous chapter, a wall function for the thermal diffusivity α_w will be applied at the right hand end (where the fixed temperature is applied). To evaluate the wall function, the material is assumed to have a Prandtl number (Pr) of 0.71 and the right boundary cell has a $y^+ = 30$.

Step 1: Divide the Geometry into a Mesh

For the example in Figure 31, divide the geometry into a mesh of 5 cells of equal length. The length of each cell (L_{cell}) is given by:

$$L_{\text{cell}} = \frac{L}{N} = \frac{5}{5} = 1\text{m} \tag{167}$$

Because the cells are uniformly distributed and have equal size, the distance between cell centroids d is equivalent to the length of the cells. Hence:

$$d_{LP} = d_{PR} = d = 1\text{m} \tag{168}$$

Step 2: Evaluate the Wall Function

Before the material properties can be assigned, the wall function needs to be evaluated. As the wall function for α_w is being applied (equation 166), the first stage is to evaluate the empirical function P. For a molecular Prandtl number of 0.71 and a turbulent Prandtl number of 0.85, the function P evaluates as:

$$P = 9.24 \left[\left(\frac{Pr}{Pr_t} \right)^{3/4} - 1 \right] \left[1 + 0.28 e^{-0.007 Pr / Pr_t} \right] = -1.491 \tag{169}$$

Now the intersection point between the two sections of the wall function (y_L^+) needs to be calculated. This requires the solution of the following non-linear equation for y^+ (equation 148):

$$Pr\ y^+ - Pr_t \left(\frac{1}{\kappa} \log \left(E y^+ \right) + P \right) = 0 \tag{170}$$

This non-linear equation can be solved with any root finding algorithm (Bisection, Newton-Raphson etc.). The Newton-Raphson method gives rapid convergence and will be used here. To use the Newton-Raphson method, define:

$$f = Pr\ y^+ - Pr_t \left(\frac{1}{\kappa} \log \left(E y^+ \right) + P \right) \tag{171}$$

$$\frac{df}{dy^+} = Pr - \frac{Pr_t}{\kappa y^+} \tag{172}$$

The Newton-Raphson iteration proceeds by evaluating:

$$y_{i+1}^+ = y_i^+ - \frac{f}{df/dy^+} \tag{173}$$

Starting from an initial guess (y_0^+) of 11.0, the Newton-Raphson iteration gives:

Iteration	y_i^+	y_{i+1}^+
1	11.000	11.806
2	11.806	11.796
3	11.796	11.796

Hence, the solution of the Newton Raphson procedure gives $y_L^+ = 11.796$ when $Pr = 0.71$. Now that y_L^+ has been evaluated, α_w can be calculated (using equation 166).

$$\alpha_w = \begin{cases} \alpha & y^+ < y_L^+ \\ \alpha \left[\dfrac{Pr\ y^+}{Pr_t \left(\frac{1}{\kappa} \log \left(E y^+ \right) + P \right)} \right] & y^+ > y_L^+ \end{cases} \tag{174}$$

In this example, $y^+ = 30$. Hence, $\alpha_w / \alpha = 2.074$ and the near wall thermal diffusivity will be approximately double the thermal diffusivity of the interior cells.

Step 2: Assign Material Properties

The wall function is applied to the molecular thermal diffusivity α. The molecular thermal diffusivity is defined as:

$$\alpha = \frac{k}{\rho c_p} \tag{175}$$

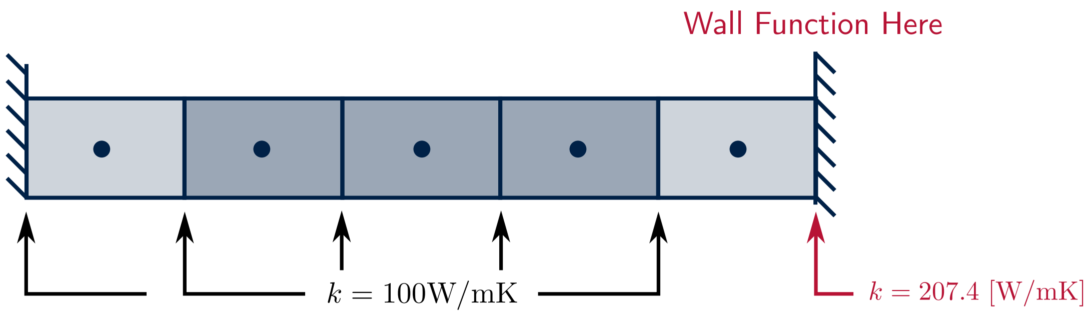


Figure 32: Thermal conductivity of each of the faces in the mesh when a wall function is applied to the right end of the bar.

where k is the thermal conductivity, ρ is the material density and c_p is the specific heat capacity. For the example problem considered here:

$$\alpha = \frac{k}{\rho c_p} = \frac{100}{8000 * 500} = 2.5 \times 10^{-5} \text{ [m}^2\text{/s]} \tag{176}$$

This is the thermal diffusivity for all the interior cells and interior faces in the mesh. For the right boundary face, the thermal diffusivity is modified by the wall function:

$$\frac{\alpha_w}{\alpha} = 2.074 \qquad \alpha_w = 5.185 \times 10^{-5} \text{ [m}^2\text{/s]} \tag{177}$$

As the thermal diffusivity of the right boundary face is now α_w, then the thermal conductivity of the face k_w is also modified.

$$k_w = \rho c_p \alpha_w = 8000 * 500 * 5.185 \times 10^{-5} = 207.4 \text{ [W/mK]} \tag{178}$$

It follows that the thermal conductivity of all the faces in the mesh is constant (100 W/mK), except for the right boundary face, where k modified by the wall function. The variation of thermal conductivity across the mesh is shown in Figure 32. For the interior cells in the mesh, the diffusive heat flux across the cell faces DA is given by:

$$DA = \frac{kA}{d} = \frac{100 * 0.1}{1} = 10 \text{ [W/K]} \tag{179}$$

For the right boundary cell, the diffusive heat flux is evaluated using the modified thermal conductivity (k_w).

$$DA = \frac{k_w A}{d} = \frac{200 * 0.1}{1} = 20.74 \text{ [W/K]} \tag{180}$$

The heat source in each cell is not affected by the wall function. It is given by:

$$\overline{S}V = \overline{S}AL_{\text{cell}} = 1000 * 0.1 * 1 = 100 \text{ [W]} \tag{181}$$

Step 3: Calculate the Matrix Coefficients

The matrix coefficients for 1D heat diffusion in a bar with fixed heat flux (Neumann) boundary conditions at the left end and fixed temperature (Dirichlet) boundary conditions at the right end are:

Matrix Coefficients

	a_L	a_R	a_P	S_p	S_u
Boundary (L)	0	$D_R A_R$	$a_l + a_r - S_p$	0	$-q_A A_L + \overline{S}V$
Interior	$D_L A_L$	$D_R A_R$	$a_l + a_r - S_p$	0	$\overline{S}V$
Boundary (R)	$D_L A_L$	0	$a_l + a_r - \boldsymbol{S_P}$	$-2\boldsymbol{D_R}A_R$	$T_B(2\boldsymbol{D_R}A_R) + \overline{S}V$

However, as a wall function is applied to the right face of the right boundary cell, the coefficient D_R (highlighted in **red** in the table above) is modified by the wall function. The other coefficients remain unchanged by the wall function. Filling in the table with the calculated coefficients:

Matrix Coefficients

	a_L	a_R	a_P	S_p	S_u
Boundary (L)	0	10	10	0	90
Interior	10	10	20	0	100
Boundary (R)	10	0	**51.5**	**-41.5**	**8395.9**

It is clear that the wall function enters the matrix equations through the diagonal coefficient a_p and the source term S_u of the cell that contains the boundary face. The coefficients in the other cells remain unchanged.

Using summation notation, the wall function modifies the face contribution of the right boundary face through k_w. No other changes are made to the face contributions of the other faces.

Summation Notation

Face Type	a_P	a_N	S_u
Interior	10	10	0
Neumann (Left)	0	0	-10
Dirichlet (Right, No Wall Function)	**20**	0	**4000**
Dirichlet (Right, Wall Function)	**41.5**	0	**8295.9**

Step 4: Assemble and Solve the Matrices

As a reminder, the matrices are populated in the following manner:

$$\begin{bmatrix} a_{p1} & -a_{r1} & 0 & 0 & 0 \\ -a_{l2} & a_{p2} & -a_{r2} & 0 & 0 \\ 0 & -a_{l3} & a_{p3} & -a_{r3} & 0 \\ 0 & 0 & -a_{l4} & a_{p4} & -a_{r4} \\ 0 & 0 & 0 & -a_{l5} & a_{p5} \end{bmatrix} \begin{bmatrix} T_1 \\ T_2 \\ T_3 \\ T_4 \\ T_5 \end{bmatrix} = \begin{bmatrix} S_{u1} \\ S_{u2} \\ S_{u3} \\ S_{u4} \\ S_{u5} \end{bmatrix} \tag{182}$$

Filling in the known coefficients from the table above:

$$\begin{bmatrix} 10 & -10 & 0 & 0 & 0 \\ -10 & 20 & -10 & 0 & 0 \\ 0 & -10 & 20 & -10 & 0 \\ 0 & 0 & -10 & 20 & -10 \\ 0 & 0 & 0 & -10 & \mathbf{51.5} \end{bmatrix} \begin{bmatrix} T_1 \\ T_2 \\ T_3 \\ T_4 \\ T_5 \end{bmatrix} = \begin{bmatrix} 90 \\ 100 \\ 100 \\ 100 \\ \mathbf{8395.9} \end{bmatrix} \tag{183}$$

For the case when a wall function is not applied, $D_R A_R = kA/d = 10$ W/mK and the matrices reduce back to the same matrices from the previous chapter.

$$\begin{bmatrix} 10 & -10 & 0 & 0 & 0 \\ -10 & 20 & -10 & 0 & 0 \\ 0 & -10 & 20 & -10 & 0 \\ 0 & 0 & -10 & 20 & -10 \\ 0 & 0 & 0 & -10 & \mathbf{30} \end{bmatrix} \begin{bmatrix} T_1 \\ T_2 \\ T_3 \\ T_4 \\ T_5 \end{bmatrix} = \begin{bmatrix} 90 \\ 100 \\ 100 \\ 100 \\ \mathbf{4100} \end{bmatrix} \tag{184}$$

Hence, the effect of the wall function is localised to the cell where the wall function is applied.

Run the Example Problem Yourself!

Now, open either the Excel spreadsheets, the Python source code or the MATLAB code and solve the problem for yourself.

Excel	`wallFunction.xlsx`
Python	`wallFunction.py`
MATLAB	`wallFunction.m`

Examine the calculation of the coefficients, the assembly of the matrices and run the code. A fixed value of $y^+ = 30$ has been set initially. Try changing the value of y^+ and observe the changes in the solution. Note that in a real CFD code, y^+ changes automatically in response to changes in the mesh. In this problem, we are specifying the value of y^+ **directly** instead, so that we can observe the changes in solution for a **fixed mesh**. Hence, the solution is not strictly accurate, but is useful for demonstration purposes.

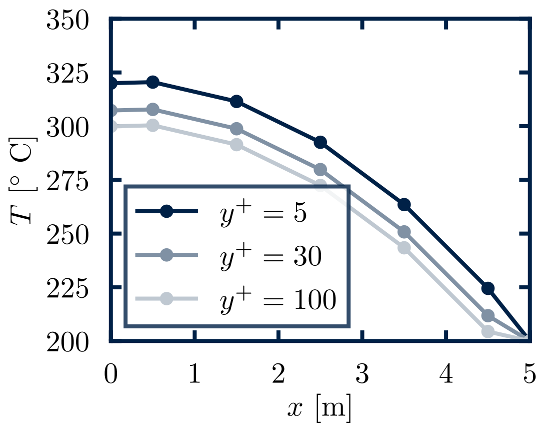


Figure 33: Temperature variation along the 1D bar with different values of y^+ on the right boundary face.

Results

Figure 33 shows the temperature variation along the bar, for different values of y^+ on the right boundary face. Notice that the temperature of the right boundary face is fixed at 200°C. However, the temperature of the wall adjacent cell centroid (T_5) is modified by the wall function. This results in a change in shape of the entire temperature profile.

The temperature of the wall adjacent cell centroid (T_5) is not modified because the wall function changes its temperature directly. The temperature of the cell is modified because the wall function modifies the thermal conductivity of the right face of this cell from 100 W/mK to 207.4 W/mK. In order to maintain the same heat flux from the wall (q_w), the temperature gradient (dT/dx) is reduced to counterbalance the increase in k_w. The table below summarises the heat flux balance for every cell in the mesh for a y^+ of (a) 5 and (b) 30. It is clear that the heat flux through the faces of each of the cells remains unchanged by the wall function, but the temperature profile does change (see Figure 33).

Heat Flux Balance

(a) $y^+ = 5$

Cell	Q_l [W]	Q_r [W]	$\overline{S}V$ [W]	Error
1	**10**	90	100	0
2	-90	190	100	0
3	-190	290	100	0
4	-290	390	100	0
5	-390	**490**	100	0

Heat flux out of the bar = 490 + 10 = 500W
Heat generated in the bar = 100 + 100 + 100 + 100 + 100 = 500W

(b) $y^+ = 30$

Cell	Q_l [W]	Q_r [W]	$\overline{S}V$ [W]	Error
1	**10**	90	100	0
2	-90	190	100	0
3	-190	290	100	0
4	-290	390	100	0
5	-390	**490**	100	0

Heat flux out of the bar = 490 + 10 = 500W
Heat generated in the bar = 100 + 100 + 100 + 100 + 100 = 500W

As the heat flux from the left end of the bar is fixed by the boundary condition (10W) and 500W is generated in the bar, 490W must pass out of the bar at the right end. Hence, as the wall function modifies α_w, the temperature gradient at the wall changes, so that the heat flux remains the same (490W).