University of Nottingham
UK | CHINA | MALAYSIA

Computer Modelling Techniques

Numerical Methods

Lecture 1: short intro & solution of 1D steady diffusion equation

Mirco Magnini

Module specs

Module name	Computer Modelling Techniques MMME3086
Credits	20 (requires ~200 hrs study)
Year/Level	Level 3
Semester	Autumn Semester
Module Convenor	Dr Mirco Magnini
Teaching staff	**Dr Mirco Magnini** (Numerical Methods, NM) **Dr Chris Bennett, Dr Luke Parry** (Finite Element Analysis, FEA) **Dr Donald Giddings** (Computational Fluid Dynamics, CFD)

Educational Aims	To provide students with a basic knowledge and understanding of the mainstream computer modelling techniques used in modern engineering practice, including Finite Element and Computational Fluid Dynamics methods.

Module specs

MMME3086 covers the whole simulation procedure and the most popular simulation software for CFD (fluids) and FEA (solid):

Dr. Mirco Magnini
Numerical Methods: FV, fundamental aspects, Matlab
Teaching weeks 1-3; NM coursework

Dr. Chris Bennett
Finite Element Analysis: FE, solid mechanics, Abaqus
Teaching weeks 4-7; FEA coursework

Dr. Don Giddings
Computational Fluid Dynamics: FV, ANSYS
Teaching weeks 8-11; CFD coursework

Method of assessment

Coursework	Three marked coursework assignments: NM coursework (30%) FEA coursework (35%) CFD coursework (35%) • The assignments do not require supervision. • Time required to complete each assignment: 6 hours approx. • Submission deadline: 2 weeks after releasing the assignment • Educational objective: To gain some practical experience of the fundamentals of solving numerical problems and running FEA and CFD engineering software codes that are widely used in industry.

Timetable

Lectures: Thursday, 11-13, Coates Road Auditorium.

Seminars: Friday, 09-11, Coates Road Auditorium. **Bring your own laptop.**

Teaching week	Section	w/b	Lecture topic (2h)	Seminar session (2h)	Coursework dates
1	NM	02 Oct	Steady diffusion equation	1D steady finite-volume in Matlab	
2	NM	09 Oct	Solution of linear systems, unsteady diffusion equation	Gauss-Seidel method in Matlab	
3	NM	16 Oct	Solution of nonlinear equations, numerical integration	1D unsteady finite-volume in Matlab	**Mon 16 Oct: NM coursework release**
4	FEA	23 Oct	1D FE, Stiffness Matrices	Stiffness Matrix Assembly, and Solution	
5	FEA	30 Oct	Truss/Pin-jointed FE	Abaqus: Geo, Mesh	**Mon 30 Oct: NM coursework deadline**
6	FEA	06 Nov	Continuum Elements, Plates and Shells	Abaqus: BCs, Loading, Solution, Post-processing	**Thu 09 Nov: FEA coursework release**
7	FEA	13 Nov	Practical Notes on FE	FEA coursework support	
8	CFD	20 Nov	Practical introduction to CFD and a commercial code	Model creation and mesh generation.	**Thu 23 Nov: FEA coursework deadline**
9	CFD	27 Nov	Derive the Navier Stokes equations of fluid motion in 3D for an incompressible, steady flow	Setting up the models in CFD solution and achieving converged solution in Fluent	**Thu 30 Nov: CFD coursework release**
10	CFD	04 Dec	Demonstrate how the finite volume numerical method works in a 1D case	Post solution processing of CFD model and CW support	
11	CFD	11 Dec	CFD overview and revision	-	**Thu 14 Dec: CFD coursework deadline**

CMT – Part 1: Numerical Methods

Teacher – Mirco Magnini, UoN-webpage

Office – Coates B100a

Email – mirco.magnini@nottingham.ac.uk

Q&A – Forum in Moodle (NM)

Timetable

Teaching week	w/b	Lecture engagement (Thu, h11-13)	In-person seminar (Fri, h09-11)
1	02 Oct	L1: Steady 1D heat equation (2h)	Matlab: 1D steady finite-volume
2	09 Oct	L2: Solution of linear systems (1h), L3: Unsteady heat equation (1h)	Matlab: Gauss-Seidel method
3	16 Oct	L4: Solution of nonlinear equations (1h), L5: Numerical integration (1h)	Matlab: 1D unsteady finite-volume

Seminars – Coates Road Auditorium, **bring your laptop**!

NM **Coursework** – Released on Mon 16 Oct, deadline Mon 30 Oct

Scope of CMT – Part 1: Numerical Methods

- Learn how to discretise an ODE/PDE, **L1**
- Solve a discretised ODE/PDE, **seminar 1** (demo 1)
- Learn how to solve the resulting system of linear equations, **L2**
- Implement a solver for linear equations, **seminar 2** (demo 2)
- Learn how to discretise unsteady PDEs, **L3**
- Solve an unsteady PDE, **seminar 3** (demo 3)
- Learn how to solve nonlinear equations, **L4**
- Learn how to perform numerical integration, **L5**

Resources

Slides/notes/demos on Moodle cover the entire material for NM.

Matlab is installed in all computer rooms and Engineering Virtual Desktop. To install it in your own machine, follow the instructions on workspace. Consider reviewing the MATLAB Onramp training if you are not familiar with the software.

Books – If you wish to know more, my notes are based on:

- Numerical Heat Transfer and Fluid Flow, S. V. Patankar, Hemisphere Publisher, 1980
  https://nusearch.nottingham.ac.uk/permalink/f/gq7rlv/44NOTUK_ALMA21725380800005561
- Numerical Recipes, W. H. Press et al., Cambridge University Press, 2007
  https://nusearch.nottingham.ac.uk/permalink/f/11rbvif/44NOTUK_ALMA21106504400005561
- Applied Numerical Methods with MATLAB for engineers and scientists, S. C. Chapra, McGraw-Hill 2012
  https://nusearch.nottingham.ac.uk/permalink/f/11rbvif/44NOTUK_ALMA21106792720005561
- Numerical Analysis, R. L. Burden et al., Cengage Learning, 2016 (available online)
  https://nusearch.nottingham.ac.uk/permalink/f/1m5tnd/44NOTUK_ALMA51125623900005561
- An Introduction to Computational Fluid Dynamics, H. K. Versteeg et al., Pearson Prentice Hall, 2007
  https://nusearch.nottingham.ac.uk/permalink/f/1m5tnd/44NOTUK_ALMA21830489300005561

N.B. If you Google any of these books, it should not be hard to find free online versions for download

This week's menu (2h lecture + 2h seminar)

- ODEs/PDEs
- 1D steady-state diffusion problems
- The finite-volume method in 1D
- The finite-volume method in 2D – structured mesh
- The finite-volume method in 3D – structured mesh
- The finite-volume method – unstructured mesh
- Indicators of solution accuracy
- Seminar 1/demo 1: 1D steady FV method in Matlab (Worked examples 1,2&3)

Expected outcome: know how to discretise an ODE/PDE using the finite-volume method; use matlab to obtain the numerical solution; be able to use different tools to verify the accuracy of the solution.

Partial Differential Equations (PDEs)

The aim of this and next lecture is to develop a *numerical method* for the *discrete solution* of *Partial Differential Equations (PDEs)*.

Ordinary Differential Equation (ODE): differential equation that contains one or more derivatives of an unknown function that depends on **only one variable**. It is linear if it is of the first degree in the unknown function and its derivatives, for example:

$$a_0(x)\phi + a_1(x)\frac{d\phi}{dx} + a_2(x)\frac{d^2\phi}{dx^2} + \cdots + a_n(x)\frac{d^n\phi}{dx^n} + b(x) = 0$$

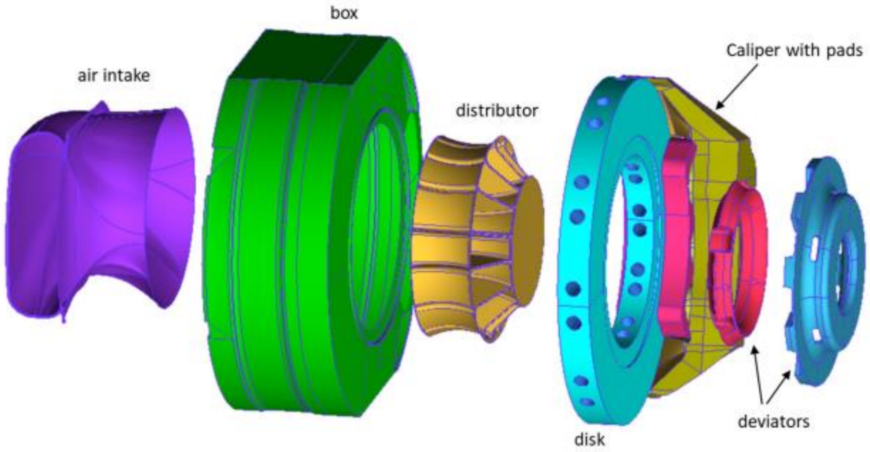
Partial Differential Equation (PDE): differential equation that contains one or more partial derivatives of an unknown function that depends on **at least two variables**. It is linear if it is of the first degree in the unknown function and its partial derivatives, for example:

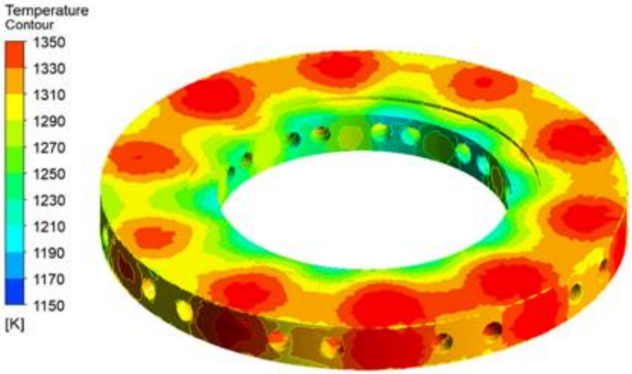
$$A(x,y)\frac{\partial^2\phi}{\partial x^2} + B(x,y)\frac{\partial^2\phi}{\partial x\,\partial y} + C(x,y)\frac{\partial^2\phi}{\partial y^2} + D(x,y,\phi) = 0$$

A PDE example: The heat equation

$$\boxed{\underbrace{\frac{\partial(\rho c_p T)}{\partial t}}_{\text{Variation of internal energy in the unit time}} = \underbrace{\frac{\partial}{\partial x}\left(\lambda\frac{\partial T}{\partial x}\right) + \frac{\partial}{\partial y}\left(\lambda\frac{\partial T}{\partial y}\right) + \frac{\partial}{\partial z}\left(\lambda\frac{\partial T}{\partial z}\right)}_{\text{Heat diffusion}} + \underbrace{S(x,y,z,t,T)}_{\text{Heat generation}}}$$

Why/where is it important? Thermal management enters many engineering problems, for example in a **racing car braking system**





Source: C. Cravero, D. Marsano, MDPI Energies 15 (2022) 2934.
https://www.mdpi.com/1996-1073/15/8/2934

The heat equation

How is it derived? Let's see a 2D example.

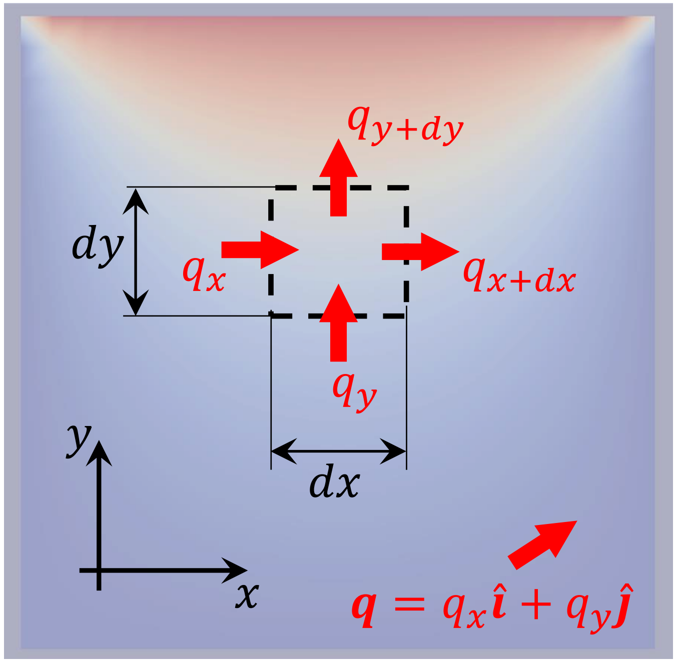
Energy balance:

Temporal variation of internal energy content =

Heat flux through boundaries + Internal generation

$$\frac{\partial(\rho c_p T)}{\partial t} dx\, dy =$$

$$(q_x - q_{x+dx})dy + (q_y - q_{y+dy})dx + S(x,y,t,T)\, dx\, dy$$


q_{y+dy}
dy
q_x
q_{x+dx}
q_y
y
dx
x
$\boldsymbol{q} = q_x\hat{\boldsymbol{\imath}} + q_y\hat{\boldsymbol{\jmath}}$

$$q_{x+dx} = q_x + \frac{\partial q_x}{\partial x} dx, \qquad q_{y+dy} = q_y + \frac{\partial q_y}{\partial y} dy$$

$$\Rightarrow \frac{\partial(\rho c_p T)}{\partial t} \cancel{dx\, dy} = -\frac{\partial q_x}{\partial x} \cancel{dx\, dy} - \frac{\partial q_y}{\partial y} \cancel{dx\, dy} + S(x,y,t,T)\, \cancel{dx\, dy}$$

Fourier law: $\boldsymbol{q} = -\lambda \nabla T = -\frac{\partial T}{\partial x}\hat{\boldsymbol{\imath}} - \frac{\partial T}{\partial y}\hat{\boldsymbol{\jmath}}$

$$\Rightarrow \boxed{\frac{\partial(\rho c_p T)}{\partial t} = \frac{\partial}{\partial x}\left(\lambda \frac{\partial T}{\partial x}\right) + \frac{\partial}{\partial y}\left(\lambda \frac{\partial T}{\partial y}\right) + S(x,y,t,T)}$$

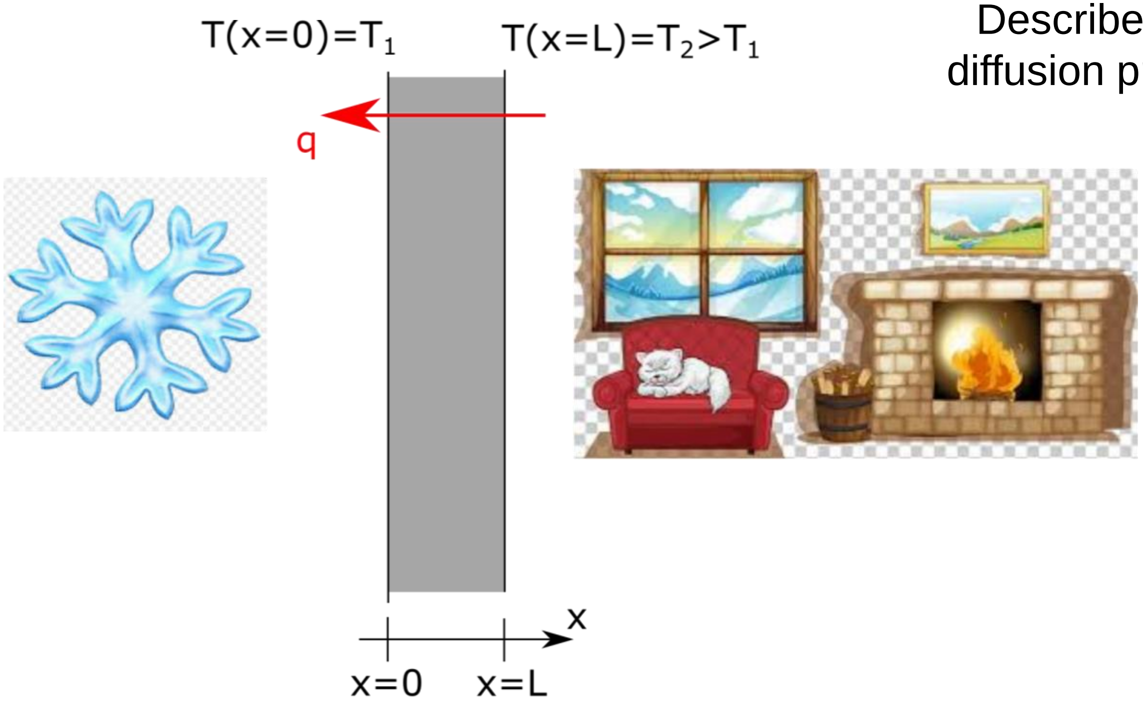
ρ: density [kg/m^3]; c_p: specific heat [J/(kg*K)]; λ: thermal conductivity [W/(m*K)]; q: heat flux [W/m^2]; S: source term [W/m^3]

1D steady-state heat equation

- 1D: the problem depends only on one spatial coordinate
- Steady-state: the problem is independent of time

The heat equation becomes an ODE.

$$\underbrace{\frac{d}{dx}\left(\lambda\frac{dT}{dx}\right)}_{\text{Describes the diffusion process}} + \underbrace{S(x,T)}_{\text{Describes creation/destruction}} = 0$$



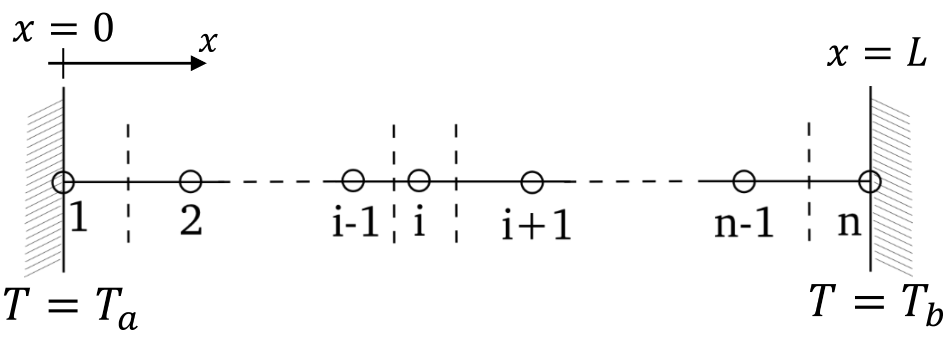
Many transport processes in engineering can be described by this equation! More in the notes…

Any solution method for this ODE will be applicable to different engineering problems!

The discretisation concept

Differential problem to be solved:
1D steady-state heat conduction

$$\boxed{\begin{array}{l}\dfrac{d}{dx}\left(\lambda\dfrac{dT}{dx}\right)+S(x,T)=0, \quad 0<x<L \\[2ex] T(x=0)=T_a, \qquad T(x=L)=T_b\end{array}}$$



An analytical method seeks for the continuous *T(x)* satisfying the problem $\forall x \in [0, L]$

A numerical method seeks for an approximate solution $T_1,\ldots,T_n$ valid only at discrete locations $x_1,\ldots,x_n$ → if we have *n* unknowns T_i, we need ***n* equations**!

In general, we prefer to solve systems of algebraic linear equations of the kind:

$$\begin{cases} a_{1,1}T_1 + a_{1,2}T_2 + \cdots + a_{1,i}T_i + \cdots + a_{1,n}T_n = b_1 \\ \vdots \\ a_{n,1}T_1 + a_{n,2}T_2 + \cdots + a_{n,i}T_i + \cdots + a_{n,n}T_n = b_n \end{cases} \longrightarrow \mathbf{A\times T=B}$$

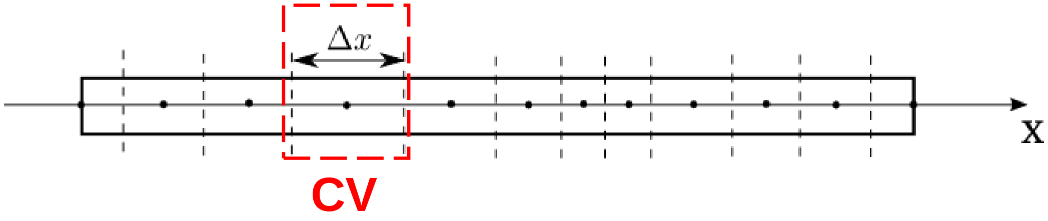
Therefore, the specific numerical method adopted (FD,FV,FE,…) identifies:

- How to derive the discretization equation from the initial differential equation
- How to express derivatives as a function of the node values $T_1,\ldots,T_n$
- How to deal with nonlinear terms

The finite-volume method – 1D

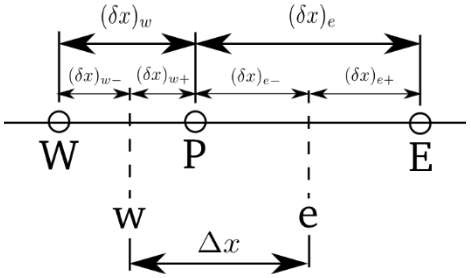
Equation to be solved:
1D steady-state heat conduction

$$\boxed{\frac{d}{dx}\left(\lambda\frac{dT}{dx}\right)+S(x,T)=0}$$



From the differential to the discretized equation (internal nodes):

1. Discretisation of the domain into control volumes (CV)
2. Integration over the CV:



CV cross-section area

$$\int_V \frac{d}{dx}\left(\lambda\frac{dT}{dx}\right)dV+\int_V S(x,T)dV=0,\qquad dV=Adx,$$

Average value of *S* within the CV

$$\int_w^e \frac{d}{\cancel{dx}}\left(\lambda\frac{dT}{dx}\right)A\cancel{dx}+\int_w^e S(x,T)Adx=0 \longrightarrow \left[\lambda\frac{dT}{dx}\right]_w^e+\bar{S}[x]_w^e=0$$

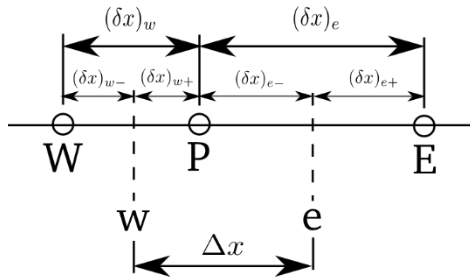
$$\left(\lambda\frac{dT}{dx}\right)_e-\left(\lambda\frac{dT}{dx}\right)_w+\bar{S}\Delta x=0$$

How do we discretise the derivatives?

We need an assumption for the behavior of $T(x)$ between the CV-centres

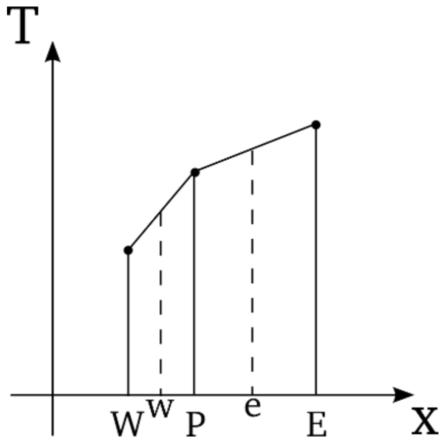
The finite-volume method – 1D

$\frac{dT}{dx}$?



Taylor expansion around generic x_i:

$$T(x) = T(x_i) + (x - x_i)\frac{dT}{dx}\bigg|_{x_i} + \frac{(x - x_i)^2}{2}\frac{d^2T}{dx^2}\bigg|_{x_i} + \frac{(x - x_i)^3}{6}\frac{d^3T}{dx^3}\bigg|_{x_i} + higher\ order$$



Linear profile assumption: we retain terms until 2nd-order

Truncation error

$$T(x_P) = T(x_e) - (\delta x)_{e-}\frac{dT}{dx}\bigg|_{x_e} + \frac{(\delta x)_{e-}^2}{2}\frac{d^2T}{dx^2}\bigg|_{x_e} - \frac{(\delta x)_{e-}^3}{6}\frac{d^3T}{dx^3}\bigg|_{x_e}$$

$$T(x_E) = T(x_e) + (\delta x)_{e+}\frac{dT}{dx}\bigg|_{x_e} + \frac{(\delta x)_{e+}^2}{2}\frac{d^2T}{dx^2}\bigg|_{x_e} + \frac{(\delta x)_{e+}^3}{6}\frac{d^3T}{dx^3}\bigg|_{x_e}$$

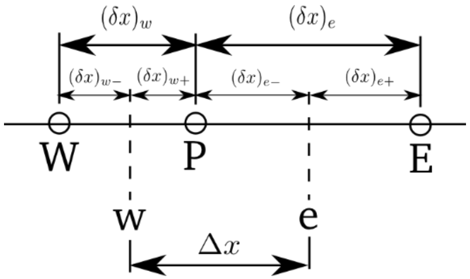
$$\xrightarrow[=(\delta x)_{e+}]{(\delta x)_{e-}} \qquad \frac{dT}{dx}\bigg|_{x_e} = \frac{T(x_E) - T(x_P)}{(\delta x)_e} - \frac{(\delta x)_e^2}{24}\frac{d^3T}{dx^3}\bigg|_e$$

Truncation error $O[(\delta x)_e^2 T_e''']$: order of accuracy of the approximation

Higher-order profiles (e.g. quadratic) exist, but require more points

The finite-volume method – 1D

$$\boxed{\frac{d}{dx}\left(\lambda\frac{dT}{dx}\right)+S(x,T)=0}$$



Approximation of derivatives and linearisation of source term:

$$\left(\lambda\frac{dT}{dx}\right)_e=\lambda_e\frac{T_E-T_P}{\delta x_e},\qquad \left(\lambda\frac{dT}{dx}\right)_w=\lambda_w\frac{T_P-T_W}{\delta x_w},\qquad \bar{S}=S_P T_P+S_C$$

$$\left(\lambda\frac{dT}{dx}\right)_e-\left(\lambda\frac{dT}{dx}\right)_w+\bar{S}\Delta x=0 \longrightarrow \lambda_e\frac{T_E-T_P}{\delta x_e}-\lambda_w\frac{T_P-T_W}{\delta x_w}+(S_P T_P+S_C)\Delta x=0$$

$$-\frac{\lambda_w}{\delta x_w}T_W+\left(\frac{\lambda_w}{\delta x_w}+\frac{\lambda_e}{\delta x_e}-S_P\Delta x\right)T_P-\frac{\lambda_e}{\delta x_e}T_E=S_C\Delta x$$

Final linear algebraic equation

$$\longrightarrow \boxed{a_W T_W+a_P T_P+a_E T_E=b}$$

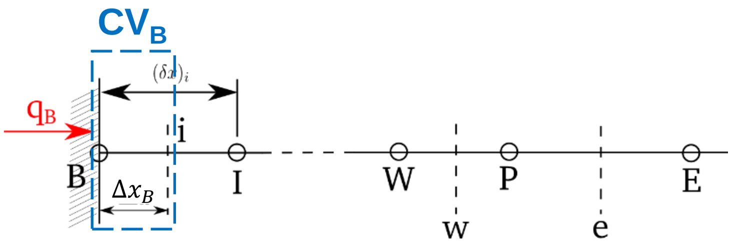
$$a_W=-\frac{\lambda_w}{\delta x_w},\qquad a_P=\frac{\lambda_w}{\delta x_w}+\frac{\lambda_e}{\delta x_e}-S_P\Delta x,\qquad a_E=-\frac{\lambda_e}{\delta x_e},\qquad b=S_C\Delta x$$

This was valid for all internal CVs, but what about boundary CVs?

The finite-volume method – 1D

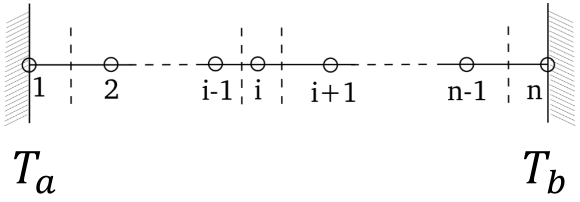
Discretised equations for the boundary CVs - **Dirichlet boundary conditions**:

The temperature at the left boundary is known, for example we know that $T(x=0)=T_a$. Then we can write a “dummy” equation of the kind:



$$\boxed{a_B T_B + a_I T_I = b, \qquad a_B = 1, a_I = 0, b = T_a}$$

Or, with our usual notation:



1st CV: $a_P T_P + a_E T_E = b,$ with $a_P = 1,\ a_E = 0,\ b = T_a$

nth CV: $a_W T_W + a_P T_P = b,$ with $a_P = 1,\ a_W = 0,\ b = T_b$

Discretised equations for the boundary CVs - **Neumann boundary conditions**:

Instead of the temperature, at the left boundary we know the heat flux q_B. This sets a condition on the temperature gradient, because owing to the Fourier’s law:

$$q = -\lambda \frac{dT}{dx}$$

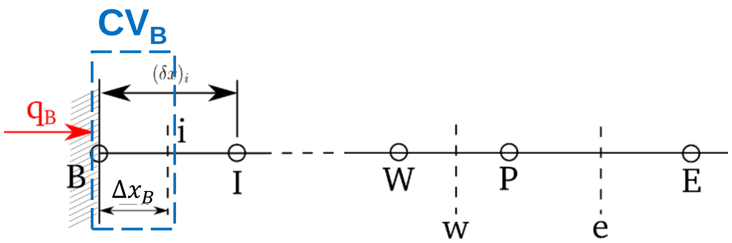
The finite-volume method – 1D

Discretised equations for the boundary CVs - **Neumann boundary conditions**:

Instead of the temperature, at the left boundary we know the heat flux q_B. This sets a condition on the temperature gradient, because owing to the Fourier's law:

$$q = -\lambda \frac{dT}{dx}$$

We derive a discretisation equation for CV_B by integrating the diffusion equation as done before:



$$\int_B^i \frac{d}{dx}\left(\lambda \frac{dT}{dx}\right) A\, dx + \int_B^i S(x,T) A\, dx = 0 \longrightarrow \left[\lambda \frac{dT}{dx}\right]_B^i + \bar{S}[x]_B^i = 0$$

$$\left(\lambda \frac{dT}{dx}\right)_i \overset{q_B}{\boxed{-\left(\lambda \frac{dT}{dx}\right)_B}} + \bar{S}\Delta x_B = 0 \longrightarrow \lambda_i \frac{T_I - T_B}{\delta x_i} + q_B + (S_P T_B + S_C)\Delta x_B = 0$$

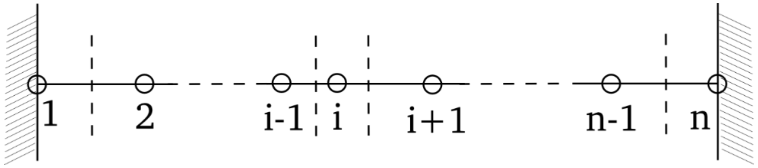
$$\boxed{a_B T_B + a_I T_I = b, \qquad a_B = \frac{\lambda_i}{\delta x_i} - S_P \Delta x_B, \qquad a_I = -\frac{\lambda_i}{\delta x_i}, \qquad b = S_C \Delta x_B + q_B}$$

Or, with our usual notation:

$$a_P T_P + a_E T_E = b, \qquad a_P = \frac{\lambda_e}{\delta x_e} - S_P \Delta x_B, a_E = -\frac{\lambda_e}{\delta x_e}, b = S_C \Delta x_B + q_B$$

The finite-volume method – 1D

Assembling the linear system of equations:



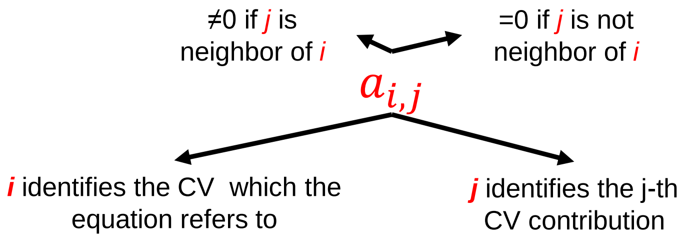
1st CV: $a_{1,P}T_{1,P} + a_{1,E}T_{1,E} = b_1 \;\Rightarrow\; a_{1,1}T_1 + a_{1,2}T_2 = b_1$

ith CV: $a_{i,W}T_{i,W} + a_{i,P}T_{i,P} + a_{i,E}T_{i,E} = b_i \;\Rightarrow\; a_{i,i-1}T_{i-1} + a_{i,i}T_i + a_{i,i+1}T_{i+1} = b_i$

nth CV: $a_{n,W}T_{n,W} + a_{n,P}T_{n,P} = b_n \;\Rightarrow\; a_{n,n-1}T_{n-1} + a_{n,n}T_n = b_n$

$$\begin{pmatrix}
a_{1,1} & a_{1,2} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \cdots & 0 \\
a_{2,1} & a_{2,2} & a_{2,3} & 0 & 0 & 0 & 0 & 0 & 0 & \cdots & 0 \\
0 & a_{3,2} & a_{3,3} & a_{3,4} & 0 & 0 & 0 & 0 & 0 & \cdots & 0 \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
0 & \cdots & 0 & a_{i-1,i-2} & a_{i-1,i-1} & a_{i-1,i} & 0 & 0 & 0 & \cdots & 0 \\
0 & \cdots & 0 & 0 & a_{i,i-1} & a_{i,i} & a_{i,i+1} & 0 & 0 & \cdots & 0 \\
0 & \cdots & 0 & 0 & 0 & a_{i+1,i} & a_{i+1,i+1} & a_{i+1,i+2} & 0 & \cdots & 0 \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
0 & \cdots & 0 & 0 & 0 & 0 & 0 & a_{n-2,n-3} & a_{n-2,n-2} & a_{n-2,n-1} & 0 \\
0 & \cdots & 0 & 0 & 0 & 0 & 0 & 0 & a_{n-1,n-2} & a_{n-1,n-1} & a_{n-1,n} \\
0 & \cdots & 0 & 0 & 0 & 0 & 0 & 0 & 0 & a_{n,n-1} & a_{n,n}
\end{pmatrix}
\times
\begin{pmatrix} T_1 \\ T_2 \\ T_3 \\ \vdots \\ T_{i-1} \\ T_i \\ T_{i+1} \\ \vdots \\ T_{n-2} \\ T_{n-1} \\ T_n \end{pmatrix}
=
\begin{pmatrix} b_1 \\ b_2 \\ b_3 \\ \vdots \\ b_{i-1} \\ b_i \\ b_{i+1} \\ \vdots \\ b_{n-2} \\ b_{n-1} \\ b_n \end{pmatrix}$$

A×T=B: linear algebraic system with ***n*** equations
A: **tridiagonal** *n*×*n* matrix, **T**,**B**: *n*×*1* vectors



≠0 if *j* is neighbor of *i*

=0 if *j* is not neighbor of *i*

$a_{i,j}$

i identifies the CV which the equation refers to

j identifies the j-th CV contribution

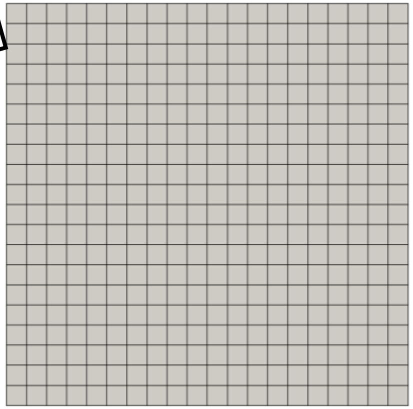
For **numerical stability**, it is good if:
- Diagonal coefficients are **positive**
- Off-diagonal coefficients are **negative**

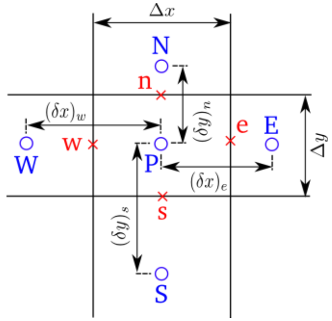
The finite-volume method – 2D structured

$$\boxed{\frac{\partial}{\partial x}\left(\lambda\frac{\partial T}{\partial x}\right)+\frac{\partial}{\partial y}\left(\lambda\frac{\partial T}{\partial y}\right)+S(x,y,T)=0}$$

No need to know 😅

A 2D structured mesh





Finite-volume discretization (structured mesh):

$$\boxed{a_N T_N + a_W T_W + a_P T_P + a_E T_E + a_s T_S = b}$$

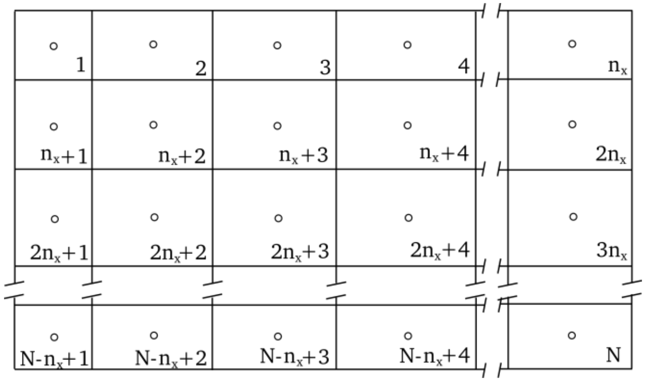
Assembling the linear system

Quadrilateral mesh with n_x×n_y=n control volumes.

CV neighbor connectivity:

P→i, E→i+1, W→i-1, N→i-n_x, S→i+n_x

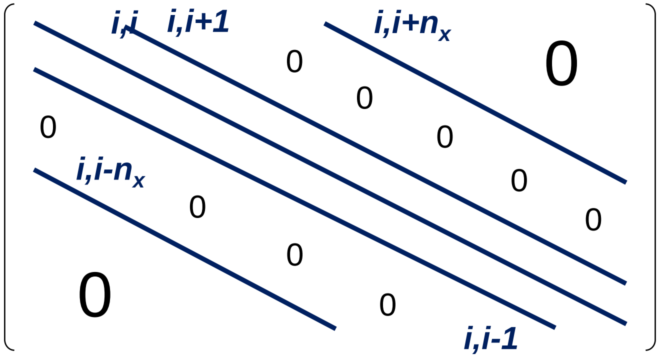
ith CV: $a_{i,i-n_x}T_{i-n_x}+a_{i,i-1}T_{i-1}+a_{i,i}T_i+a_{i,i+1}T_{i+1}+a_{i,i+n_x}T_{i+n_x}=b_i$



1	2	3	4	n_x
n_x+1	n_x+2	n_x+3	n_x+4	$2n_x$
$2n_x+1$	$2n_x+2$	$2n_x+3$	$2n_x+4$	$3n_x$
$N-n_x+1$	$N-n_x+2$	$N-n_x+3$	$N-n_x+4$	N

CV are consecutively numbered from left to right and from top to bottom

A=

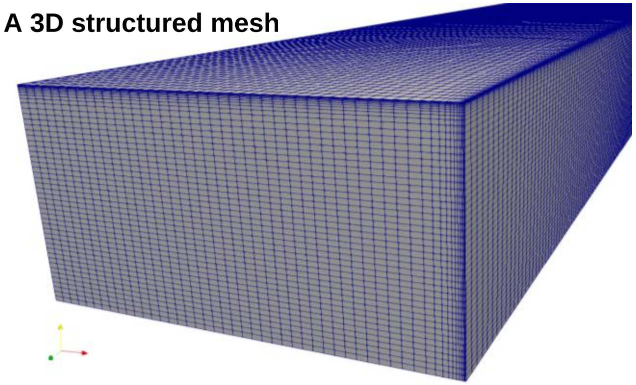


A×T=B: linear algebraic system with n eqs.
A: *n*×*n* matrix with 5 non-zero diagonals,
T,**B**: *n*×*1* vectors

The finite-volume method – 3D structured

$$\frac{\partial}{\partial x}\left(\lambda\frac{\partial T}{\partial x}\right)+\frac{\partial}{\partial y}\left(\lambda\frac{\partial T}{\partial y}\right)+\frac{\partial}{\partial z}\left(\lambda\frac{\partial T}{\partial z}\right)+S(x,y,z,T)=0$$

A 3D structured mesh

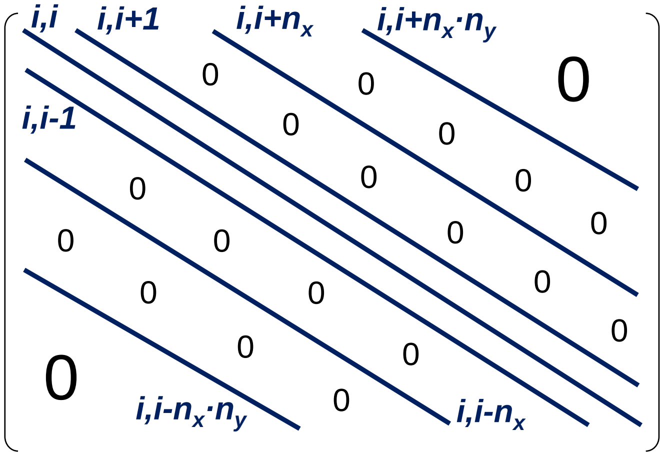


Consider a 3D domain with $n_x \times n_y \times n_z = n$ control volumes.

CV neighbor connectivity: $T \longrightarrow i-n_x \cdot n_y$, $B \longrightarrow i+n_x \cdot n_y$

ith CV: $a_{i,i-n_x\cdot n_y}T_{i-n_x\cdot n_y} + a_{i,i-n_x}T_{i-n_x} + a_{i,i-1}T_{i-1} + a_{i,i}T_i + a_{i,i+1}T_{i+1} + a_{i,i+n_x}T_{i+n_x} + a_{i,i+n_x\cdot n_y}T_{i+n_x\cdot n_y} = b_i$

A=



A×T=B: linear algebraic system with n equations
A: *n*×*n* matrix with 7 non-zero diagonals
T,**B**: *n*×*1* vectors

No need to know 😅

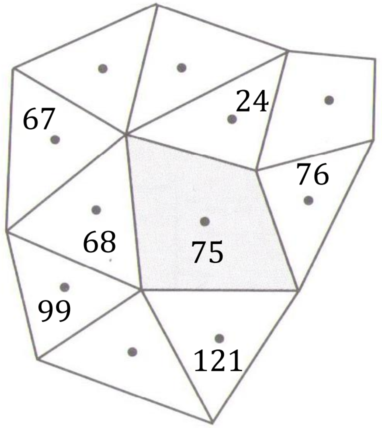
Advantages in using a structured mesh**:**

- No need to store CV neighbour connectivity → less disk space
- Faster access to the matrix elements → the linear solver is faster
- Solvers specific for these matrix structures are more efficient than general purpose ones

The finite-volume method – Unstructured mesh

No need to know 😅

$$\boxed{\nabla \cdot (\lambda \nabla T) + S(\boldsymbol{x}, T) = 0}$$



68th CV: $a_{68,67}T_{67} + a_{68,68}T_{68} + a_{68,75}T_{75} + a_{68,99}T_{99} = b_{68}$

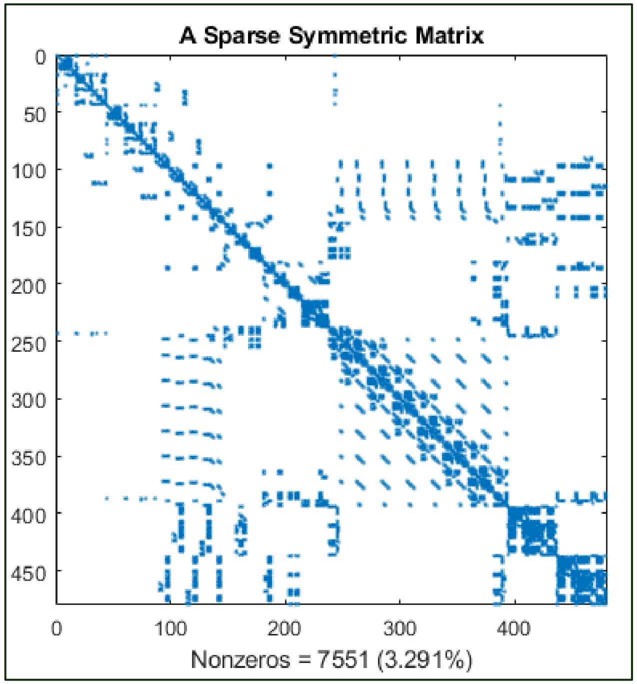
75th CV: $a_{75,24}T_{24} + a_{75,68}T_{68} + a_{75,75}T_{75} + a_{75,76}T_{76} + a_{75,121}\,T_{121} = b_{75}$

A: *n*×*n* **sparse** matrix; the blue dots in the figure indicate elements $a_{i,j} \neq 0$.

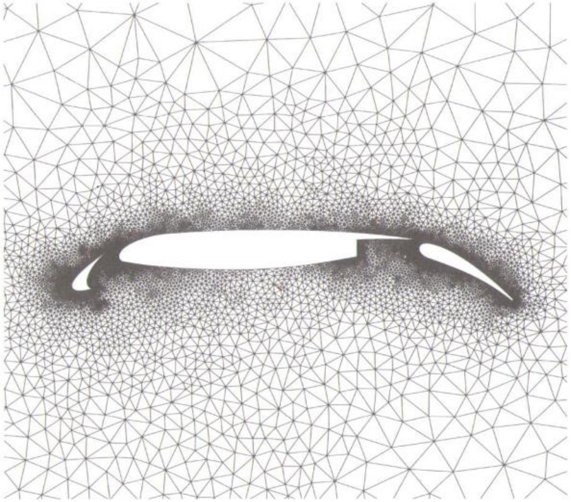
Remember that:

$a_{i,j} \neq 0$ if T_i depends (is neighbor of) on T_j

$a_{i,j} = 0$ if T_i does not depend (not a neighbor) on T_j.



https://uk.mathworks.com/help/matlab/math/sparse-matrix-reordering.html



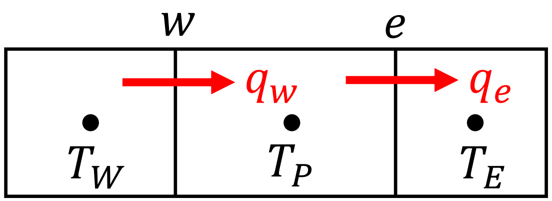
An unstructured mesh

Now**:**

- Easier to mesh complex domains, but…
- …need to store CV neighbour connectivity
- Slower linear solver

How to judge solution accuracy?

- ✓ Is the solution bounded, for instance between the boundary values?
- ✓ Are the boundary conditions respected?
- ✓ Compare with theory/experiments (if available)
- ✓ Residual error: $R = \dfrac{|\boldsymbol{B} - \boldsymbol{A} \times \boldsymbol{T}|}{|diag(\boldsymbol{A}) \times \boldsymbol{T}|} < tol$
- ✓ Energy balance: at steady-state, if $\bar{S} = 0$, for each cell $q_w - q_e = 0$



$$q_w - q_e = -\left[\lambda \frac{dT}{dx}\right]_w + \left[\lambda \frac{dT}{dx}\right]_e = 0$$

➡ $-\lambda_w \dfrac{T_P - T_W}{\delta x_w} + \lambda_e \dfrac{T_E - T_P}{\delta x_e} = 0$

➡ Total error (rescaled): $error = \dfrac{\sum_{i=1}^{i=n} |q_{i,w} - q_{i,e}|}{q_{ref}}$

q_{ref}: arbitrary, e.g. $q_{ref} = \sum_{i=1}^{i=n} |q_{i,w}|$, or $q_{ref} = \lambda |T_b - T_a| / L$

How to judge solution accuracy?

✓ As you refine/coarsen the mesh, is the convergence order of the solution respected?

Remember our starting point: $\boxed{\left(\lambda \frac{dT}{dx}\right)_e - \left(\lambda \frac{dT}{dx}\right)_w + \bar{S}\Delta x = 0}$

To express the derivatives, we discretised them using a 2nd-order scheme:

$$\left.\frac{dT}{dx}\right|_{x_e} = \frac{T(x_E) - T(x_P)}{(\delta x)_e} + O[(\delta x)_e^2 T_e'''] \qquad \left.\frac{dT}{dx}\right|_{x_w} = \frac{T(x_P) - T(x_W)}{(\delta x)_w} + O[(\delta x)_w^2 T_w''']$$

So that our discretization equation became:

$$\lambda_e \frac{T_E - T_P}{\delta x_e} - \lambda_w \frac{T_P - T_W}{\delta x_w} + \bar{S}\Delta x + O[(\delta x)^2 T'''] = 0 \qquad O[(\delta x)^2 T''']\text{: truncation error}$$

$O[(\delta x)^2 T''']$ does not actually appear in the discretisation equation, but is the intrinsic error of our numerical solution. As we refine the mesh and decrease δx, the truncation error decreases and our solution converges to the exact solution. If the overall truncation error is $O[(\delta x)^2]$, our solution is expected to converge to the exact solution with an order 2, i.e. if we halve the node spacing, the error should be 4 times smaller. To verify this, we must know the exact solution. We need to test different values of δx, and display a log-log plot of the error vs δx. We will see this in Worked example 2.

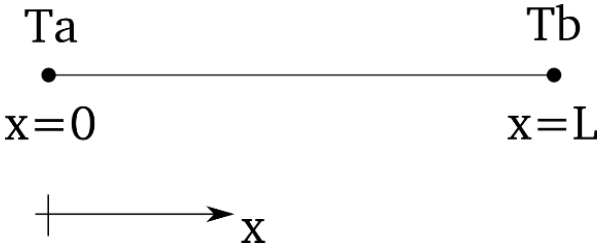
What to take home from today’s lecture

- How to discretise the 1D steady-state heat conduction equation using FV
- How to estimate the order of accuracy of the solution based on the truncation error of the discretisation schemes used
- How to derive the linear system of equations for the 1D problem
- How to impose Dirichlet or Neumann conditions
- How to assemble the system of linear equations
- How to judge the accuracy of the solution
- How to use Matlab to solve the linear system and obtain the solution

Worked example 1

Implement a FV code in Matlab to solve the 1D steady heat conduction problem:

$$\boxed{\frac{d}{dx}\left(\lambda\frac{dT}{dx}\right)+S(x,T)=0}$$



with:

- λ constant
- Zero source term, $S(x,T)=0$

Parameters:

- L=1 m
- n=21 equidistant nodes
- λ=400 W/(m·K)
- T_a=300 K, T_b=320 K

Boundary conditions:

- T(x=0)=T_a
- T(x=L)=T_b

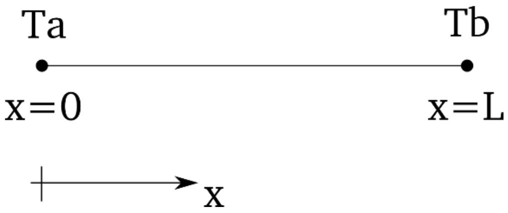
To solve the linear system, make use of matlab's "backslash" operator: T=A\B, (https://uk.mathworks.com/help/matlab/ref/mldivide.html).

Perform the solution accuracy tests suggested in the previous slides. The problem has the following analytical solution:

$$T(x)=T_a+\frac{T_b-T_a}{L}x$$

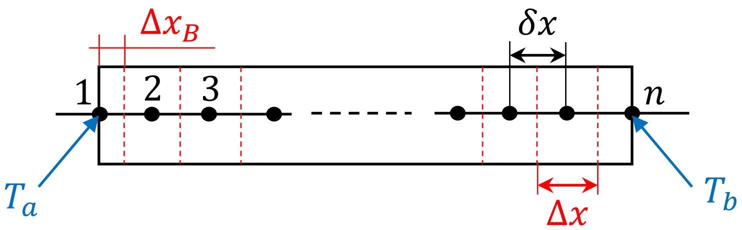
Worked example 1

We should know now that solving the problem numerically means to solve the linear system **A×T=B**, so we need to generate the matrix **A** and vector **B**, then we will use Matlab’s functions to solve the system.



Let’s start defining the geometry.

21 equidistant nodes give:

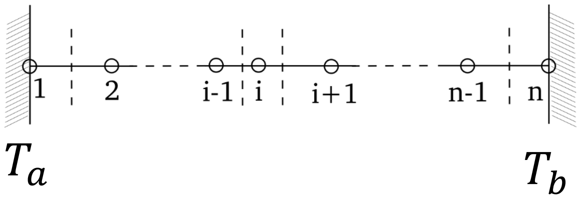


$$\delta x = \Delta x = \frac{L}{n-1} = \frac{1\,m}{21-1} = 0.05\,m$$

Whereas for the two boundary control volumes: $\Delta x_B = \dfrac{\Delta x}{2} = 0.025\,m$

Next, we begin filling matrix **A** and vector **B**

Worked example 1



1st CV: $a_{1,P}T_{1,P} + a_{1,E}T_{1,E} = b_1 \;\Rightarrow\; a_{1,1}T_1 + a_{1,2}T_2 = b_1$

→ $a_{1,1} = 1,\qquad a_{1,2} = 0,\qquad b_1 = T_a$

ith CV: $a_{i,W}T_{i,W} + a_{i,P}T_{i,P} + a_{i,E}T_{i,E} = b_i \;\Rightarrow\; a_{i,i-1}T_{i-1} + a_{i,i}T_i + a_{i,i+1}T_{i+1} = b_i$

→ $a_{i,i-1} = -\dfrac{\lambda}{\delta x},\qquad a_{i,i} = \dfrac{2\lambda}{\delta x},\qquad a_{i,i+1} = -\dfrac{\lambda}{\delta x},\qquad b_i = 0$

nth CV: $a_{n,W}T_{n,W} + a_{n,P}T_{n,P} = b_n \;\Rightarrow\; a_{n,n-1}T_{n-1} + a_{n,n}T_n = b_n$

→ $a_{n,n-1} = 0,\qquad a_{n,n} = 1,\qquad b_n = T_b$

$$\begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & \cdots & 0 & -\dfrac{\lambda}{\delta x} & \dfrac{2\lambda}{\delta x} & -\dfrac{\lambda}{\delta x} & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & \cdots & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \times \begin{pmatrix} T_1 \\ \vdots \\ T_i \\ \vdots \\ T_n \end{pmatrix} = \begin{pmatrix} T_a \\ \vdots \\ 0 \\ \vdots \\ T_b \end{pmatrix}$$

Continues in the notes…

Worked example 2

Repeat Worked example 1, but now with a nonzero source term:

$$S_C = 5000\ \frac{W}{m^3}, \qquad S_P = -100\frac{W}{m^3K}$$

The associated ODE: $\lambda T'' + S_C + S_P T = 0$

has the following analytical solution:

$$T(x) = c_1 e^{\mu_1 x} + c_2 e^{\mu_2 x} - \frac{S_c}{S_p} \qquad \mu_{1,2} = \pm\sqrt{-\frac{S_p}{\lambda}}$$

$$c_1 = \frac{T_b - \left(\frac{S_c}{S_p} + T_a\right) e^{\mu_2 L} + \frac{S_c}{S_p}}{e^{\mu_1 L} - e^{\mu_2 L}} \qquad c_2 = T_a + \frac{S_c}{S_p} - c_1$$

The only difference with example 1 is that now:

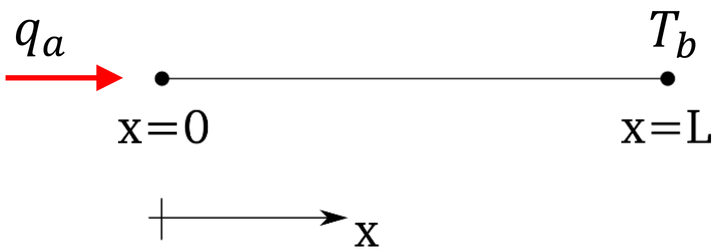
$$a_{i,i} = \frac{2\lambda}{\delta x} - S_p \Delta x, \qquad b_i = S_c \Delta x,$$

Continues in the notes…

Worked example 3

Let's now see the case of Neumann boundary conditions. Solve:

$$\boxed{\frac{d}{dx}\left(\lambda\frac{dT}{dx}\right)+S(x,T)=0}$$



with:

- λ constant
- Nonzero source term, $S_C = 50000\ W/m^3, S_P = 0$

Parameters:

- L=1 m
- n=21 equidistant nodes
- λ=400 W/(m·K)

Boundary conditions:

- $q_a=10^4$ W/m^2
- $T(x=L)=T_b=320$ K

The problem has the following analytical solution:

$$T(x)=T_b+\frac{q}{\lambda}(L-x)+\frac{S_C}{2\lambda}(L^2-x^2)$$

Continues in the notes…