MPY308

Clinical Engineering
&
Computational Mechanics

Recommended Reading:


Motivation

• Many physical phenomena are described by differential equations

• The equations are often too difficult to solve analytically, particularly for complex geometries or boundary conditions

• Solutions can be sought by the application of numerical methods.
Module Overview

This module describes some of the numerical techniques that can be used to explore physical systems, with illustrations from heat conduction, biomechanics and biofluid mechanics.

Finite differences will be introduced from first principles, and simple code will be developed to solve linear partial differential equations.

The theoretical basis of finite element methods will be briefly discussed, and the lectures will be supported by laboratory sessions in which the student will apply commercial codes to investigate problems in the medical sector.
Outline Syllabus

Introduction, objectives, history and development, commercial exploitation, typical applications of numerical methods in biomechanics and biofluid mechanics

Introduction to MATLAB, matrix manipulation. Presentation of results. Exercises.

One-dimensional heat conduction. Solution by finite difference and finite element schemes. Extension to 2D and 3D. Shape functions. Galerkin weighted residuals.

Laplace's equation. Numerical solution and practical applications.


Introduction to ANSYS. Pre-processing, solution and post-processing phases.

Practical aspects of finite element analysis.

Methodology

• In order to construct effective numerical schemes to solve the equations we need to understand the fundamental characteristics of the equations.

• We shall learn how to reduce the equations to a set of simultaneous equations. These can readily be written in matrix algebra.
Modern packages like MATLAB have a whole host of routines for efficient manipulation of matrices. They also feature brilliant graphics routines for displaying the results so that we can understand them!

We shall apply numerical schemes that we have constructed in MATLAB to solve problems relevant to the activities of a medical physicist.
Software - ANSYS

• We shall learn how to use ANSYS, a major commercial finite element analysis package, to solve problems including structural, fluid dynamic, electrical and thermal applications.

• We will construct models in ANSYS to study problems with a more complex geometrical shape.
Aims and Objectives

At the end of this module you should:

• Understand the role of numerical methods in the solution of real problems in biophysics and bioengineering;
• Be able to construct simple finite difference and finite element schemes for the solution of particular systems of differential equations;
• Understand the assumptions in the numerical schemes and the requirements for appropriate boundary conditions, constitutive equations and numerical formulations;
• Have a feel for the ‘size’ of a problem, the requirements for an appropriate numerical scheme, and understand how to go about choosing or developing software.
• Have completed a mini-project using MATLAB, ANSYS or FLOTRAN to investigate a biomedical system;
• Be aware of the requirements for comprehensive verification, validation and reporting of numerical solutions, and recognise these requirements in the production of a mini-project report;
Structure

• 22 x one hour lectures/ laboratories
• Private work, examples, assignments, reading ~ 50 hours

Assessment

<table>
<thead>
<tr>
<th>Assignment</th>
<th>%age</th>
<th>Hand In</th>
</tr>
</thead>
<tbody>
<tr>
<td>Assignment 1 (Matlab)</td>
<td>10 %</td>
<td>Wk 3</td>
</tr>
<tr>
<td>Assignment 2 (Finite Differences)</td>
<td>25 %</td>
<td>Wk 8</td>
</tr>
<tr>
<td>Mini-Project Assignment (ANSYS)</td>
<td>65 %</td>
<td>Wk 12</td>
</tr>
</tbody>
</table>
Laplace's Equation
Laplace's Equation

\[ \nabla^2 \phi = 0 \]

\[ \frac{\partial^2 \phi}{\partial x^2} + \frac{\partial^2 \phi}{\partial y^2} + \frac{\partial^2 \phi}{\partial z^2} = 0 \]

Applies to many physical problems including, for example, steady-state temperature distribution in a conductive body.
Laplace's Equation

\[ \nabla^2 \phi = 0 \]

- \( \phi \) depends only on the spatial coordinates \((x, y, z)\).

- \( \phi \) is a scalar. It might, for example, be temperature. Once we know the solution for \( \phi \) we can construct a colour map showing its distribution. This is often called a fringe or contour plot.
Laplace’s Equation

\[ \nabla^2 \phi = 0 \]

- \( \nabla \phi \) (we say \textbf{grad} \( \phi \)) is a vector. It represents a variation of \( \phi \) in space. We might plot it on a grid as an arrow at each point. The colour or length of the arrow might describe the magnitude of the vector and its direction will indicate the direction of the gradient of \( \phi \).

- What physical quantities might these arrows represent?

- \( \nabla^2 \phi \) (formally \text{div} (\text{grad} \( \phi \))) is a scalar again. In Laplace’s equation its value is zero everywhere in the solution domain.
Laplace's Equation in 1D

\[
\frac{\partial^2 \phi}{\partial x^2} = 0
\]

Describes steady state distribution of temperature along a bar.
Laplace's Equation in 1D \[ \frac{\partial^2 \phi}{\partial x^2} = 0 \]

Obviously we can solve this very easily using integration, but we will use it as a vehicle to develop our understanding of the numerical solution procedures.

It’s a second order (linear, homogeneous) differential equation, so to solve it we would expect to need two boundary conditions.

There are four possible boundary conditions - the value of the temperature and the gradient of the temperature at each end of the bar.

Not all pairs of boundary conditions are useful to us. We will explore this problem later.
Finite Differences
Spatial Discretisation of Domain

The basis of our numerical schemes will be to divide the domain up into sections and to seek the solution at points in the domain. We will cause these points nodes. In the 1D case we will have:

The fundamental unknowns in our numerical representation will be the temperatures at each node, $T_i$. If we are to solve the problem numerically, we need to go from the differential equation to a series of simultaneous equations.

How do we represent $\nabla^2 \phi$ in the discretised domain?
Finite Differences in 1D

How do we represent the gradient of temperature at node $i$?
Difference Expressions for Gradient

- **Backward difference**
  \[
  \left[ \frac{\partial T}{\partial x} \right]_i = \frac{T_i - T_{i-1}}{h}
  \]

- **Forward difference**
  \[
  \left[ \frac{\partial T}{\partial x} \right]_i = \frac{T_{i+1} - T_i}{h}
  \]

- **Central difference**
  \[
  \left[ \frac{\partial T}{\partial x} \right]_i = \frac{T_{i+1} - T_{i-1}}{2h}
  \]

Which, if any, is ‘most accurate’?
Difference Expressions for Curvature

Can you construct a difference expression for curvature?

\[ \frac{\partial^2 T}{\partial x^2} \]
Central Difference Expression for Curvature

\[ \left[ \frac{\partial T}{\partial x} \right]_{i-} = \frac{T_i - T_{i-1}}{h} \]

Gradient to left

\[ \left[ \frac{\partial T}{\partial x} \right]_{i+} = \frac{T_{i+1} - T_i}{h} \]

Gradient to right

\[ \left[ \frac{\partial^2 T}{\partial x^2} \right]_i = \frac{\partial}{\partial x} \left( \frac{\partial T}{\partial x} \right)_i = \frac{T_{i+1} - T_i}{h} - \frac{T_i - T_{i-1}}{h} \]

\[ \left[ \frac{\partial^2 T}{\partial x^2} \right]_i = \frac{\frac{T_{i-1} - 2T_i + T_{i+1}}{h^2}} {h^2} \]
Discretised Laplace's Equation

Away from boundaries (n-2 equations)

\[
\left[ \frac{\partial^2 T}{\partial x^2} \right] = \frac{T_{i-1} - 2T_i + T_{i+1}}{h^2} = 0
\]

\[
\left[ \frac{\partial^2 T}{\partial x^2} \right]_1 = \frac{T_0 - 2T_1 + T_2}{h^2} = 0
\]

\[
\left[ \frac{\partial^2 T}{\partial x^2} \right]_n = \frac{T_{n-1} - 2T_n + T_{n+1}}{h^2} = 0
\]

Contain ‘external’ nodes – need to replace these equations
Boundary Conditions

Options

1) Specify Temperature
   e.g. \( T_1 = 7 \)

2) Specify Gradient
   e.g. \( \left[ \frac{\partial T}{\partial x} \right]_1 = 10 \)

   forward difference
   \( \frac{T_2 - T_1}{h} = \left[ \frac{\partial T}{\partial x} \right]_1 \)
Can boundary gradient accuracy be improved by central differencing?

2) Specify Gradient

\[ \frac{T_2 - T_0}{2h} = \left[ \frac{\partial T}{\partial x} \right]_1 \rightarrow T_0 = T_2 - 2h \left[ \frac{\partial T}{\partial x} \right]_1 \]

**Central difference with Laplaces equation yields same boundary condition as forward difference**

\[ \frac{T_2 - 2h \left[ \frac{\partial T}{\partial x} \right]_1 - 2T_1 + T_2}{h^2} = 0 \]

\[ \frac{T_2 - T_1}{h} = \left[ \frac{\partial T}{\partial x} \right]_1 \]
Numerical Example

$T_1 = 20^\circ C$

$\left[ \frac{\partial T}{\partial x} \right]_5 = -30 \, ^\circ C/m$
Numerical Example

\[ T_1 = 20 \]

\[ \frac{T_1 - 2T_2 + T_3}{h^2} = 0 \]

\[ \frac{T_2 - 2T_3 + T_4}{h^2} = 0 \]

\[ \frac{T_3 - 2T_4 + T_5}{h^2} = 0 \]

\[ \frac{T_4 - T_5}{h} = 30 \]

\[ \left[ \frac{\partial T}{\partial x} \right]_5 = -30 \, ^\circ C/m \]
Numerical Example

\[ T_1 = 20 \]
\[ \begin{array}{cccccc}
1 & 0 & 0 & 0 & 0 & 0 \\
-1 & 2 & -1 & 0 & 0 & 0 \\
0 & -1 & 2 & -1 & 0 & 0 \\
0 & 0 & -1 & 2 & -1 & 0 \\
0 & 0 & 0 & -1 & 1 & 1
\end{array} \]
\[
\begin{bmatrix}
T_1 \\
T_2 \\
T_3 \\
T_4 \\
T_5
\end{bmatrix} = \begin{bmatrix}
20 \\
0 \\
0 \\
0 \\
-3
\end{bmatrix}
\]

\[ \left[ \frac{\partial T}{\partial x} \right]_5 = -30 \, ^\circ C/m \]
Numerical Example

\[ T_1 = 20 \]

\[ 0.1 \text{ m} \]

\[ \begin{bmatrix} \frac{\partial T}{\partial x} \end{bmatrix}_5 = -30 \degree \text{C/m} \]

\[
\begin{bmatrix}
1 & 0 & 0 & 0 & 0 \\
-1 & 2 & -1 & 0 & 0 \\
0 & -1 & 2 & -1 & 0 \\
0 & 0 & -1 & 2 & -1 \\
0 & 0 & 0 & -1 & 1 \\
\end{bmatrix}
\begin{bmatrix}
T_1 \\
T_2 \\
T_3 \\
T_4 \\
T_5 \\
\end{bmatrix} =
\begin{bmatrix}
20 \\
0 \\
0 \\
-3 \\
\end{bmatrix}
\]

\[ \{T\} = \begin{bmatrix}
20 \\
17 \\
14 \\
11 \\
8 \\
\end{bmatrix} \]

Solution methods … Gaussian elimination, Iterative, ….. you choose!
1D Central Difference Solver for Laplace's Equation

% 1D Central Difference Thermal Solver

nn=5; % Total number of nodes
L=0.4; % Length of domain
h=L/(nn-1); % Distance between nodes
bc1=0; % Boundary condition flag at node 1: 0 is temperature, 1 is gradient
T(1)=8; % ... and the value. T for temperature, dT for gradient
bcn=1; % Similarly at node n
dT(nn)=-30;
1D Central Difference Solver for Laplace's Equation

```matlab
k=zeros(nn,nn);    % Initialise 'stiffness' matrix
b=zeros(nn,1);      % Initialise right hand vector - 'load' vector

for i = 2 : nn-1     % Set up rows of stiffness matrix for all internal nodes
    k(i,i-1)=-1;
    k(i,i)=2;
    k(i,i+1)=-1;
end;
```

The matrix $k$ is a tridiagonal matrix:

$$
\begin{bmatrix}
1 & 0 & 0 & 0 & 0 \\
-1 & 2 & -1 & 0 & 0 \\
0 & -1 & 2 & -1 & 0 \\
0 & 0 & -1 & 2 & -1 \\
0 & 0 & 0 & -1 & 1
\end{bmatrix}
\begin{bmatrix}
T_1 \\
T_2 \\
T_3 \\
T_4 \\
T_5
\end{bmatrix}
= \begin{bmatrix} 20 \\ 0 \\ 0 \\ 0 \\ -3 \end{bmatrix}
$$
1D Central Difference Solver for Laplaces Equation

Temperature boundary condition
if bc1==0
  k(1,1)=1;
  b(1)=T(1); % Set up equation at node 1
else
  k(1,1)=1;
  k(1,2)=-1;
  b(1)=-h*dT(1);
end;

% Gradient boundary condition

% note the cost of generality.....
1D Central Difference Solver for Laplace's Equation

Temperature boundary condition
if bcn==0
    k(nn,nn)=1;
    b(nn)=T(nn);
else
    k(nn,nn-1)=-1;
    k(nn,nn)=1;
    b(nn)=h*dT(nn);
end;

Gradient boundary condition

% Set up equation at node n

\[
\begin{bmatrix}
1 & 0 & 0 & 0 & 0 \\
-1 & 2 & -1 & 0 & 0 \\
0 & -1 & 2 & -1 & 0 \\
0 & 0 & -1 & 2 & -1 \\
0 & 0 & 0 & -1 & 1
\end{bmatrix}
\begin{bmatrix}
T_1 \\
T_2 \\
T_3 \\
T_4 \\
T_5
\end{bmatrix}
= \begin{bmatrix}
20 \\
0 \\
0 \\
0 \\
-3
\end{bmatrix}
\]
1D Central Difference Solver for Laplaces Equation

if rank(k)<nn  % Check that the problem is well-posed
    'Error, k is singular'
end

T=k;  % Solve

This operation in Matlab is perfectly adequate for our current purposes: if we have large matrices we might reconsider!

An error trap ….
1D Central Difference Solver for Laplace's Equation

plot(T); % Plot results graphically

Improve this figure
- Title
- Label Axes (Units!)
- X axis should be position (m), not matrix entry
Observations on Matrix Structure

$T_1 = 20$

$\frac{\partial T}{\partial x}_5 = -30 \, ^\circ\text{C/m}$

Sparse

Tridiagonal

Storage, special solution methods?
Laplace's Equation in 2D

\[
\frac{\partial^2 \phi}{\partial x^2} + \frac{\partial^2 \phi}{\partial y^2} = 0
\]

Describes steady state distribution of temperature in 2d space
Domain Discretisation in 2D

Row $i + 1$

Row $i$

Row $i - 1$
Laplacian in 2D

\[
\begin{bmatrix}
\frac{\partial^2 T}{\partial x^2} \bigg|_{i,j}
\end{bmatrix} = \frac{T_{i,j-1} - 2T_{i,j} + T_{i,j+1}}{h^2}
\]

\[
\begin{bmatrix}
\frac{\partial^2 T}{\partial y^2} \bigg|_{i,j}
\end{bmatrix} = \frac{T_{i-1,j} - 2T_{i,j} + T_{i+1,j}}{h^2}
\]

\[
\nabla^2 \phi = \frac{T_{i,j-1} - 2T_{i,j} + T_{i,j+1}}{h^2} + \frac{T_{i-1,j} - 2T_{i,j} + T_{i+1,j}}{h^2}
\]
Laplacian in 2D

\[ \nabla^2 \phi = \frac{T_{i,j-1} - 2T_{i,j} + T_{i,j+1}}{h^2} + \frac{T_{i-1,j} - 2T_{i,j} + T_{i+1,j}}{h^2} \]
Numerical Example

\[ \frac{\partial T}{\partial y} = 200x^2 + 45x \]

\[ \frac{\partial T}{\partial x} = 100y + 50 \]

T = 0

T = 50x
Numerical Example

\[
\frac{\partial T}{\partial y} = 200x^2 + 45x
\]

\[
\frac{\partial T}{\partial x} = 100y + 50
\]

3 x 3 grid: unusual because only one internal node
**Numerical Example**

- **Grid size** $h = 0.2 \text{ m}$
- **Co-ordinates of nodes**

$$[xy] = \begin{bmatrix} 0 & 0 \\ 0.2 & 0 \\ 0.4 & 0 \\ 0 & 0.2 \\ 0.2 & 0.2 \\ 0.4 & 0.2 \\ 0 & 0.4 \\ 0.2 & 0.4 \\ 0.4 & 0.4 \end{bmatrix}$$
Numerical Example

\[ \frac{\partial T}{\partial y} = 200x^2 + 45x \]

\[ \frac{\partial T}{\partial x} = 100y + 50 \]

Compute specified temperatures and gradients

\[ \left\{ bc \right\} = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 1 \end{bmatrix} \quad ; \quad \left\{ T_{\text{spec}} \right\} = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 0 \\ 0 \end{bmatrix} \]

Set flags

\[ \left\{ \text{T} \right\} = \begin{bmatrix} ? \\ 10 \\ 20 \\ 0 \\ ? \\ ? \end{bmatrix} \quad ; \quad \left\{ \text{dT}_x \right\} = \begin{bmatrix} ? \\ 50 \\ ? \\ ? \\ ? \end{bmatrix} \quad ; \quad \left\{ \text{dT}_y \right\} = \begin{bmatrix} ? \\ 70 \\ ? \\ ? \end{bmatrix} \quad \begin{bmatrix} ? \\ ? \\ ? \\ ? \end{bmatrix} \quad \begin{bmatrix} ? \\ ? \\ ? \\ ? \end{bmatrix} \quad \begin{bmatrix} 50 \\ 17 \end{bmatrix} \]
% 2D Central Difference Thermal Solver

nnx=3; Lx=0.4;
nny=3; Ly=0.4;
h=Ly/(nny-1);   % Note that we are considering only square grids
nn=nnx*nny;

for i=1:nnx  % Calculate the co-ordinates of each node ....
    for j=1:nny
        n=i+nnx*(j-1);
        xy(n,1)=h*(i-1);  % ... and store in the array [xy]
        xy(n,2)=h*(j-1);
    end;
end;

Problem variables

Nodal co-ordinates
T=zeros(nn,1);  % Initialise the temperature vector
T_spec=zeros(nn,1); % ... and the 'specification' flag
bc1=0;                       % Specify the conditions on the x = 0 boundary ..
for n = 1 : nnx : nn-nnx+1
    T(n)=0;   T_spec(n)=1; end;
bc2=1;                       % ... and on the x = L boundary ..
for n = nnx : nnx : nn
    dTx(n)=100*xy(n,2)+50; end;
bc3=0;                       % ... and on the y = 0 boundary ..
for n = 1 : nnx
    T(n)=50*xy(n,1);   T_spec(n)=1; end;
bc4=1;                       % ... and on the y = L boundary.
for n = nn-nnx+1 : nn
    dTy(n)=200*xy(n,1)*xy(n,1)+45*xy(n,1); end;
2D Central Difference Solver for Laplace's Equation

```matlab
k=zeros(nn,nn);     % Initialise 'stiffness' matrix
b=zeros(nn,1);       % ... and 'load' vector

for i=2:nxn-1         % Set up rows of stiffness matrix for all internal nodes
    for j=2:nnn-1
        n=i+nnn*(j-1);
        k(n,n-nxn)=-1;
        k(n,n-1)=-1;
        k(n,n+1)=-1;
        k(n,n+nnn)=-1;
        k(n,n)=4;
    end;
end;
```

\[ \begin{bmatrix}
    0 & -1 & 0 & -1 & 4 & -1 & 0 & -1 & 0 \\
    T_1 & T_2 & T_3 & T_4 & T_5 & = & 0 \\
    T_6 & T_7 & T_8 & T_9
\end{bmatrix} \]
2D Central Difference Solver for Laplaces Equation

```matlab
for n = 1 : nnx : nn-nnx+1 % Set up equations on x=0 boundary
    if bc1==1
        k(n,n)=1;  k(n,n+1)=-1;  b(n)=-h*dTx(n);
    end;
end;
for n = nnx : nnx : nn % Set up equations on x=L boundary
    if bc2==1
        k(n,n)=1;  k(n,n-1)=-1;  b(n)=h*dTx(n);
    end;
end;
for n = 2 : nnx-1 % Set up equations on y=0 boundary
    if bc3==1
        k(n,n)=1;  k(n,n+nnx)=-1;  b(n)=-h*dTy(n);
    end;
end;
for n = nn-nnx+2 : nn-1 % Set up equations on y=L boundary
    if bc4==1
        k(n,n)=1;  k(n,n-nnx)=-1;  b(n)=h*dTy(n);
    end;
end;
```

Set up gradient boundary conditions

Row 9 overwritten ....
2D Central Difference Solver
for Laplace's Equation

for \( n = 1 : nn \)
    if \( T_{\text{spec}}(n) == 1 \)
        for \( j = 1 : nn \)
            \( k(n,j) = 0; \)
        end;
        \( k(n,n) = 1; \)
        \( b(n) = T(n); \)
    end;
end;

% Impose specified temperatures

\[
\begin{bmatrix}
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\end{bmatrix} \begin{bmatrix}
T_1 \\
T_2 \\
T_3 \\
T_4 \\
T_5 \\
T_6 \\
T_7 \\
T_8 \\
T_9 \\
\end{bmatrix} = \begin{bmatrix}
0 \\
10 \\
20 \\
0 \\
0 \\
70 \\
0 \\
17 \\
50 \\
\end{bmatrix}
\]

Rows 3 and 7 overwritten …..
2D Central Difference Solver for Laplaces Equation

if rank(k)<nn % Check that the problem is well-posed
    'Error, k is singular'
end

T=k\b;                % Solve

When setting up the equations some lines were overwritten by other lines.

Does this change the answers?

Is there a ‘more accurate’ alternative?
% Reshape solution vector into a matrix with entries corresponding to
% the geometrical positions of the nodes
arrayT=reshape(T,nnx,nny)';

% Plot the results
% Figure 1 Surface plot of temperature
surf(arrayT);
% Figure 2 Shaded interpolated flat image
figure, pcolor(arrayT), shading interp, axis equal
% Compute the entries of the gradient vector
[gx,gy]=gradient(arrayT,h); gx=-gx; gy=-gy;
% Figure 3 Vector plot of gradient
figure, contour(arrayT),hold on, quiver(gx,gy),hold off;
2D Central Difference Solver for Laplace's Equation

Results for 11 x 11 grid
Heat Flux

The heat flux across a region of the boundary is given by the product of the material conductivity, the gradient normal to the boundary and the surface area of the boundary,

$$\text{Flux} = k \left[ \frac{\partial T}{\partial n} \right] dS$$

An adiabatic boundary is described by the zero flux condition, implying no heat flux across it.
A symmetry plane is one about which both the geometry and the loading are symmetrical. A symmetry plane must be an adiabatic boundary (zero flux, zero gradient).
An **antisymmetry** plane is one about which the geometry is symmetrical but the loading is antisymmetrical. The temperature on such a boundary must be zero.

Use of symmetry and/or antisymmetry conditions can produce significant savings in the size of the computational model.
Theoretical Fundamentals
Theoretical Fundamentals

Taylor series expansion of a function about a point

\[ f(x) = f(a) + f'(a)(x-a) + \frac{f''(a)(x-a)^2}{2!} + \cdots + \frac{f^{(n)}(a)(x-a)^n}{n!} + \cdots \]

If the value of a function and all of its derivatives are known at a point \( a \) then its value at all points \( x \) ‘in the neighbourhood of \( a \)’ can be computed.
Order of Accuracy of Finite Difference Approximations

Truncating at order $h^2$:

$$f(x + h) = f(x) + f'(x)(h) + O(h^2)$$

Rearranging:

$$f'(x) = \frac{f(x + h) - f(x)}{h} + O(h)$$

The forward difference scheme that we derived intuitively is ‘first order accurate’

What is the order of accuracy of the backward difference and of the central difference schemes that we wrote down?
Order of Accuracy of Finite Difference Approximations

Truncating at order $h^3$:

$$f(x + h) = f(x) + f'(x)(h) + \frac{f''(x)(h)^2}{2!} + \frac{f'''(x)(h)^3}{3!} + O(h)^4$$

$$f(x - h) = f(x) - f'(x)(h) + \frac{f''(x)(h)^2}{2!} - \frac{f'''(x)(h)^3}{3!} + O(h)^4$$

Adding:

$$f(x + h) + f(x - h) = 2f(x) + 2\frac{f''(x)(h)^2}{2} + O(h)^4$$

Rearranging:

$$f''(x) = \frac{f(x + h) + 2f(x) + f(x - h)}{h^2} + O(h)^2$$

..and so the ‘-1 2 -1’ scheme is ‘second order accurate’
Order of Accuracy of Finite Difference Approximations

Need higher accuracy? …. Write more terms!

\[
f(x + h) = f(x) + f'(x)(h) + \frac{f''(x)(h)^2}{2!} + \frac{f'''(x)(h)^3}{3!} + \frac{f''''(x)(h)^4}{4!} + O(h)^5
\]

\[
f(x - h) = f(x) - f'(x)(h) + \frac{f''(x)(h)^2}{2!} - \frac{f'''(x)(h)^3}{3!} + \frac{f''''(x)(h)^4}{4!} + O(h)^5
\]

Subtracting:

\[
f(x + h) - f(x - h) = 2f'(x)h + \frac{f'''(x)(h)^3}{3} + O(h)^5
\]

Similarly, writing \(f(x+2h)\) and \(f(x-2h)\) and subtracting:

\[
f(x + 2h) - f(x - 2h) = 4f'(x)h + \frac{8f'''(x)(h)^3}{3} + O(h)^5
\]

..and by eliminating the third derivative a fourth order accurate expression for the gradient is obtained
Transient Problems
1D transient heat conduction

\[ \frac{\partial^2 \phi}{\partial x^2} = \frac{1}{\alpha} \frac{\partial \phi}{\partial t} \]

\(\alpha\) is thermal diffusivity, \(k/\rho c\)  

What are its units?

\(k\) is thermal conductivity (W/mK)  
\(\rho\) is density (kg/m\(^3\))  
\(c\) is specific heat capacity (J/kgK)
1D transient heat conduction

\[ \frac{\partial^2 \phi}{\partial x^2} = \frac{1}{\alpha} \frac{\partial \phi}{\partial t} \]

One dimensional transient heat conduction

This is a parabolic second order partial differential equation. Such equations are ‘open’ and lend themselves to time marching schemes. Generally we have a set of initial conditions, and then we follow the evolution of the solution variables. The direction in which we follow the solution does not have to be time… but it often is.
Transient Solution Storage

\[ \frac{\partial^2 \phi}{\partial x^2} = \frac{1}{\alpha} \frac{\partial \phi}{\partial t} \]

One dimensional transient heat conduction

Store the solution as a matrix array

\[ \{\phi\} = \begin{bmatrix}
\phi_{1,1} & \phi_{1,2} & \cdots & \phi_{1,j} & \cdots & \phi_{1,n} \\
\phi_{2,1} & \phi_{2,2} & \cdots & \phi_{i-1,j} & \cdots & \phi_{i-1,n} \\
\vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\
\phi_{i,1} & \phi_{i,2} & \cdots & \phi_{i,j-1} & \phi_{i,j} & \phi_{i,j+1} & \cdots & \phi_{i,n} \\
\phi_{i+1,1} & \phi_{i+1,2} & \cdots & \phi_{i+1,j} & \cdots & \phi_{i+1,n} \\
\phi_{m,1} & \phi_{m,2} & \cdots & \phi_{m,j} & \cdots & \phi_{m,n}
\end{bmatrix} \]

Spatial solution at time \( j \)

Temporal solution at position \( i \)
One discretisation option

\[ \{\phi\} = \begin{bmatrix}
\phi_{1,1} & \phi_{1,2} & \cdots & \phi_{1,j} & \cdots & \phi_{1,n} \\
\phi_{2,1} & \phi_{2,2} & \cdots & \cdots & \cdots & \cdots \\
\cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\
\phi_{m,1} & \cdots & \cdots & \cdots & \cdots & \phi_{m,n}
\end{bmatrix} \]

\[ \frac{\phi_{i-1,j} - 2\phi_{i,j} + \phi_{i+1,j}}{h^2} = \left[ \frac{\partial^2 \phi}{\partial x^2} \right] \]

Temporal solution at position \( i \)

Spatial solution at time \( j \)

Central difference in space

Forward difference in time

\[ \frac{\partial \phi}{\partial t} = \frac{\phi_{i,j+1} - \phi_{i,j}}{dt} \]
Discretisation

\[ \frac{\partial^2 \phi}{\partial x^2} = \frac{1}{\alpha} \frac{\partial \phi}{\partial t} \]

Central difference in space

\[ \frac{\phi_{i-1,j} - 2 \phi_{i,j} + \phi_{i+1,j}}{h^2} = \frac{\phi_{i,j+1} - \phi_{i,j}}{\alpha \, dt} \]

Forward difference in time

\[ \phi_{i,j+1} = \phi_{i,j} + \frac{\alpha \, dt}{h^2} \cdot (\phi_{i-1,j} - 2 \phi_{i,j} + \phi_{i+1,j}) \]
An Explicit Scheme

\[
\frac{\partial^2 \phi}{\partial x^2} = \frac{1}{\alpha} \frac{\partial \phi}{\partial t}
\]

Central difference in space

Forward difference in time

\[
\phi_{i,j+1} = \left(1 - 2 \frac{\alpha \, dt}{h^2}\right) \phi_{i,j} + \frac{\alpha \, dt}{h^2} \cdot (\phi_{i-1,j} + \phi_{i+1,j})
\]

Given the solution at any time \( j \), the solution at time \( j+1 \) can be calculated immediately from this explicit equation.

No matrix inversion is required……… it’s computationally inexpensive!
Bar of length 0.4 m, initially at temperature zero, has one end raised in temperature by 0.1°C/s for 200 seconds, and then the temperature there is maintained at 20°C. The temperature at the other end is maintained at zero throughout. The diffusivity is $1 \times 10^{-5}$ m$^2$/s.

Compute the temperature distribution at 10 second intervals at the ends and at three equi-spaced internal points, until 400 s have elapsed.
Numerical Example

Initial Temperature = 0 everywhere

What do you expect to happen?

Do we need boundary conditions at the right hand end? … what would happen without?
Numerical Example

\[
\{T\} = \begin{bmatrix}
0 & \cdots & 0 \\
0 & \cdots & 0 \\
0 & \cdots & 0 \\
0 & \cdots & 0 \\
0 & \cdots & 0 \\
\end{bmatrix}
\]

Initial Temperature = 0 everywhere

\(T_5 = 0\)
Numerical Example

\[ T_1 \]

\[ T_5 = 0 \]

\[ \phi_{i,j+1} = \left( 1 - 2 \frac{\alpha \, dt}{h^2} \right) \phi_{i,j} + \frac{\alpha \, dt}{h^2} \cdot (\phi_{i-1,j} + \phi_{i+1,j}) \]

\[ \alpha = 1 \times 10^{-5} \, \text{m}^2/\text{s}, \, dt = 10 \, \text{s}, \, h = 0.1 \, \text{m} \]

\[ \phi_{i,j+1} = 0.98 \phi_{i,j} + 0.01 (\phi_{i-1,j} + \phi_{i+1,j}) \]

An explicit scheme
Numerical Example

At \( t = 10 \) s

\[
\phi_{1,2} = 0.1 \cdot t = 0.1 \times 10 = 1
\]

\[
\phi_{2,2} = 0.98 \phi_{2,1} + 0.01(\phi_{1,1} + \phi_{3,1})
\]

\[
\phi_{3,2} = 0.98 \phi_{3,1} + 0.01(\phi_{2,1} + \phi_{4,1})
\]

\[
\phi_{4,2} = 0.98 \phi_{4,1} + 0.01(\phi_{3,1} + \phi_{5,1})
\]

\[
\phi_{5,2} = 0
\]
Numerical Example

At $t = 20$ s

$\phi_{1,3} = 0 \cdot 2t = 0 \cdot 2 \times 10 = 2$

$\phi_{2,3} = 0.98 \phi_{2,2} + 0.01(\phi_{1,2} + \phi_{3,2})$

$\phi_{3,3} = 0.98 \phi_{3,2} + 0.01(\phi_{2,2} + \phi_{4,2})$

$\phi_{4,3} = 0.98 \phi_{4,2} + 0.01(\phi_{3,2} + \phi_{5,2})$

$\phi_{5,3} = 0$

$T_1$

$T_5 = 0$

\[
\begin{bmatrix}
0 & 1 & 2 & \ldots & \ldots & \ldots \\
0 & 0 & 0.01 & \ldots & \ldots & \ldots \\
0 & 0 & 0 & \ldots & \ldots & \ldots \\
0 & 0 & 0 & \ldots & \ldots & \ldots \\
0 & 0 & 0 & \ldots & \ldots & \ldots \\
\end{bmatrix}
\]
At $t = 30$ s

\[
\begin{align*}
\phi_{1,4} &= 0 \cdot 3t = 0 \cdot 3 \times 10 = 3 \\
\phi_{2,4} &= 0.98 \phi_{2,3} + 0.01(\phi_{1,3} + \phi_{3,3}) \\
\phi_{3,4} &= 0.98 \phi_{3,3} + 0.01(\phi_{2,3} + \phi_{4,3}) \\
\phi_{4,4} &= 0.98 \phi_{4,3} + 0.01(\phi_{3,3} + \phi_{5,3}) \\
\phi_{5,4} &= 0
\end{align*}
\]
Numerical Example

..... and so we go on .......

\[
\{T\} = \begin{bmatrix}
0 & 1 & 2 & 3 & 4 & 5 & \ldots \\
0 & 0 & 0.01 & 0.0298 & 0.059205 & 0.098025 & \ldots \\
0 & 0 & 0 & 0.0001 & 0.000396 & 0.000980 & \ldots \\
0 & 0 & 0 & 0 & 0.000001 & 0.000005 & \ldots \\
0 & 0 & 0 & 0 & 0 & 0 & \ldots
\end{bmatrix}
\]
Numerical Example

Solution at start, 200s (heating time) and 400s (end)

\[
\{T\} = \begin{bmatrix}
0 & \ldots & 20 & \ldots & 20 \\
0 & \ldots & 1.694 & \ldots & 4.509 \\
0 & \ldots & 0.097 & \ldots & 0.602 \\
0 & \ldots & 0.004 & \ldots & 0.056 \\
0 & \ldots & 0 & \ldots & 0 \\
\end{bmatrix}
\]
Numerical Example

Make a note of the temperature at the mid-node \((x=0.2)\) at \(t = 400\)s

Experiment with changing the grid size \((h)\)

….. and with changing the timestep \((dt)\)

what do you find ?

Convergence ?

Stability ?
Numerical Example

Make a note of the temperature at the mid-node \((x=0.2)\) at \(t = 400s\)

*Experiment with changing the grid size \((h)\) \([dt=10s]\)*

<table>
<thead>
<tr>
<th>(h)</th>
<th>(T(0.2, 400))</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.1</td>
<td>0.602194</td>
</tr>
<tr>
<td>0.05</td>
<td>0.313677</td>
</tr>
<tr>
<td>0.025</td>
<td>0.222071</td>
</tr>
<tr>
<td>0.0125</td>
<td>???</td>
</tr>
<tr>
<td>0.00625</td>
<td></td>
</tr>
<tr>
<td>0.003125</td>
<td></td>
</tr>
<tr>
<td>0.0015625</td>
<td></td>
</tr>
</tbody>
</table>
## Numerical Example

Make a note of the temperature at the mid-node \((x=0.2)\) at \(t = 400s\)

*Experiment with changing the grid size \((h)\) \([dt=1s]\)*

<table>
<thead>
<tr>
<th>(h)</th>
<th>(T(0.2, 400))</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.1</td>
<td>0.624258</td>
</tr>
<tr>
<td>0.05</td>
<td>0.339762</td>
</tr>
<tr>
<td>0.025</td>
<td>0.250611</td>
</tr>
<tr>
<td>0.0125</td>
<td>0.226146</td>
</tr>
<tr>
<td>0.00625</td>
<td>0.219859</td>
</tr>
<tr>
<td>0.003125</td>
<td>????</td>
</tr>
<tr>
<td>0.0015625</td>
<td></td>
</tr>
</tbody>
</table>
Numerical Example

Make a note of the temperature at the mid-node \((x=0.2)\) at \(t = 400s\)

*Experiment with changing the grid size \((h)\) \([dt=0.1s]\)*

<table>
<thead>
<tr>
<th>h</th>
<th>(T(0.2, 400))</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.1</td>
<td>0.626448</td>
</tr>
<tr>
<td>0.05</td>
<td>0.342338</td>
</tr>
<tr>
<td>0.025</td>
<td>0.253414</td>
</tr>
<tr>
<td>0.0125</td>
<td>0.229034</td>
</tr>
<tr>
<td>0.00625</td>
<td>0.222770</td>
</tr>
<tr>
<td>0.003125</td>
<td>0.221193</td>
</tr>
<tr>
<td>0.0015625</td>
<td>0.220798</td>
</tr>
</tbody>
</table>
Numerical Example

Make a note of the temperature at the mid-node \((x=0.2)\) at \(t = 400\)s

*Experiment with changing the timestep \((dt)\) \([h=0.00625m]\)*

<table>
<thead>
<tr>
<th>(dt)</th>
<th>(T(0.2, 400))</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.0</td>
<td>????</td>
</tr>
<tr>
<td>2.0</td>
<td>0.020717</td>
</tr>
<tr>
<td>1.0</td>
<td>0.219859</td>
</tr>
<tr>
<td>0.5</td>
<td>0.221477</td>
</tr>
<tr>
<td>0.25</td>
<td>0.222285</td>
</tr>
<tr>
<td>0.125</td>
<td>0.222689</td>
</tr>
<tr>
<td>0.0625</td>
<td>0.222891</td>
</tr>
</tbody>
</table>
Explicit Scheme Stability Limit?

\[
\frac{\partial^2 \phi}{\partial x^2} = \frac{1}{\alpha} \frac{\partial \phi}{\partial t}
\]

Central difference in space
Forward difference in time

\[
\phi_{i,j+1} = \left(1 - 2 \frac{\alpha \ dt}{h^2}\right) \phi_{i,j} + \frac{\alpha \ dt}{h^2} \left(\phi_{i-1,j} + \phi_{i+1,j}\right)
\]

What happens when \( \frac{\alpha \ dt}{h^2} > \frac{1}{2} \)?

Go back to your tables and enter the value of \( \frac{\alpha \ dt}{h^2} \)

What do you see? … and what’s the ‘right’ answer ???

Accuracy?
Alternative Discretisation?

\[ \frac{\partial^2 \phi}{\partial x^2} = \frac{1}{\alpha} \frac{\partial \phi}{\partial t} \]

Central difference in space \rightarrow \text{Backward difference in time}

\[ \frac{\phi_{i-1,j} - 2\phi_{i,j} + \phi_{i+1,j}}{h^2} = \frac{\phi_{i,j} - \phi_{i,j-1}}{\alpha \, dt} \]

\[ -\phi_{i-1,j} + \left( 2 + \frac{h^2}{\alpha \, dt} \right) \phi_{i,j} - \phi_{i+1,j} = \phi_{i,j-1} \]
An Implicit Scheme

\[
\frac{\partial^2 \phi}{\partial x^2} = \frac{1}{\alpha} \frac{\partial \phi}{\partial t}
\]

Central difference in space

Backward difference in time

\[
-\phi_{i-1,j} + \left( 2 + \frac{h^2}{\alpha \, dt} \right) \phi_{i,j} - \phi_{i+1,j} = \phi_{i,j-1}
\]

This looks rather like our scheme for 1D Laplaces equation

(….. additional term on main diagonal and entry on right)

Triangular matrix at every timestep……… computationally expensive!

….. but are there compensations?
Numerical Example

Repeat the tests performed with the explicit scheme, completing the same tables

Experiment with changing the grid size (h)

….. and with changing the timestep (dt)

what do you find ?

Convergence ?

Stability ?

We still need to deal with ‘Accuracy’
More Discretisation Schemes

There are very many variants on these schemes, many of which bear the names of their developers (e.g. Crank-Nicholson, ….). The difference between them is the way in which the equations are discretised.

We seek to improve the accuracy and stability of explicit schemes and the accuracy of implicit schemes.

For nonlinear problems we seek also to improve the rate of convergence.
Explicit v Implicit

Explicit
- Fast computation at each timestep
- Trivially parallelisable
- Only conditionally stable
- Might need very small timestep

Implicit
- Unconditionally stable
- Computationally expensive per timestep
- .. long timesteps stable
  ..but might be inaccurate!

What spatial and temporal resolution does the physics demand?
What type of computational resource is available?
Now choose!
Finite Elements
Principles of Finite Element Analysis

In finite element analysis, like finite difference methods, we describe the solution in terms of the values of the unknowns at discrete points (NODES) in the domain.

Unlike finite difference methods, we make direct assumptions about the magnitudes of the unknown variables everywhere. Their values at all points within each ELEMENT are assumed to be known functions of the values at the nodes of the element.

The (unknown) values of the solution variables at the nodes are called the DEGREES OF FREEDOM (DOFs).

We attempt to ensure that some measure of the solution error is minimised over the whole domain.
A 1D, 2 DOF Element

How does $\phi$ vary over the element?
Linear 1D, 2 DOF Element

The most natural choice is a linear variation
Linear 1D, 2 DOF Element
Linear 1D, 2 DOF Element

\[ \phi = \frac{(x - x_j)}{L_e} \cdot \Phi_i \]

\[ \phi = \frac{(x - x_i)}{L_e} \cdot \Phi_j \]

\[ \phi = N_i \Phi_i + N_j \Phi_j = [N]\{\Phi\} \]
The Shape Functions

The functions $N$ are called *shape functions*.

$N_i$ describes how the parameter ($\phi$ in this case) varies over the element when it has a value of 1 at node $i$ and zero at all other nodes.
Methods of Weighted Residuals

For field problems the discrete numerical equations are usually formulated by minimisation of some measure of the error over the domain.

Taking Laplace's equation in 1D as an example:

\[
\frac{\partial^2 \phi}{\partial x^2} = 0
\]
Methods of Weighted Residuals

In each element we have assumed the value of $\phi_e$ as a function of its values at the nodes

$$
\phi_e = [N]_e \{\Phi\}_e
$$

Hence within each element:

$$
\frac{\partial^2 \phi}{\partial x^2} = \frac{\partial^2 ([N]_e \{\phi\}_e)}{\partial x^2} = \left[ \frac{\partial^2 N}{\partial x^2} \right]_e \{\phi\}_e
$$
Methods of Weighted Residuals

..but if Laplace's equation is to be satisfied we require that this function be zero at all points in the domain.

The amount by which the equation is not satisfied at any point is called the residual

In this case, within each element:

\[ R = \frac{\partial^2 \phi}{\partial x^2} - 0 = \left[ \frac{\partial^2 N}{\partial x^2} \right]_e \{\phi\}_e \]
Methods of Weighted Residuals

We might take one measure of the error in the solution to be the sum of the errors over the whole domain ……

\[ \int R \, dx = \sum_{\text{elements}} \left( \int_{x_i}^{x_j} \left[ \frac{\partial^2 N}{\partial x^2} \right]_e \{\phi\}_e \, dx \right) \]

A little thought might persuade us that this is not a particularly good measure of error… not least because huge + and – errors will cancel.
More generally, and more appropriately, we might take a weighted sum of the residuals

\[
\int W(x) R \, dx = \sum_{\text{elements}} \left( \int_{x_i}^{x_j} W(x) \left[ \frac{\partial^2 N}{\partial x^2} \right]_e \{\phi\}_e \, dx \right)
\]

where \( W(x) \) is a weighting factor
Galerkin’s Method of Weighted Residuals

In Galerkin’s method we take the weight to be the displacement itself

\[
\int NR \, dx = \sum_{\text{elements}} \left( \int_{x_i}^{x_j} \{\phi\}_e^T [N]_e^T \frac{\partial^2 N}{\partial x^2} e \{\phi\}_e \, dx \right)
\]

The order and manner in which the terms are written is chosen for convenience and consistency with systems with more DOFs at each node.
Galerkin’s Method of Weighted Residuals

The solution vectors are independent of $x$ and can be taken outside the integral, and indeed of the summation

$$\int_{x} NR \, dx = \sum_{\text{elements}} \left( \int\!\!\int_{x_{i}}^{x_{j}} \left[ \frac{\partial^{2} N}{\partial x^{2}} \right]_{e} dx \right) \cdot \{\phi\}^{T} \{\phi\}$$

Now if this quantity is to be minimised, its differential with respect to each of the entries in the nodal solution vector must be zero.
Galerkin’s Method of Weighted Residuals

Differentiating the error measure with respect to each of the solution variables \( \phi_i \) in turn:

\[
\sum_{\text{elements}} \left( \int_{x_i}^{x_j} [N]_e^T \frac{\partial^2 N}{\partial x^2} e \right) d x \cdot \{\phi\} = \{0\}
\]

This matrix equation represents \( n \) simultaneous equations in the \( n \) unknowns.
Galerkin’s Method of Weighted Residuals

The second order differential can be eliminated by integration by parts

\[
\int_{x_i}^{x_j} \left[ [N]_e^T \frac{\partial^2 N}{\partial x^2} \right]_e \, dx = \left[ [N]_e^T \frac{\partial N}{\partial x} \right]_e^{x_j} - \int_{x_i}^{x_j} \left[ \frac{\partial N}{\partial x} \right]_e^T \left[ \frac{\partial N}{\partial x} \right]_e \, dx
\]

The first term is a boundary condition, and will be dealt with separately.

The second term we write as:

\[
[k]_e = \int_{x_i}^{x_j} [B]_e^T [B]_e \, dx
\]

where \([B]_e = \left[ \frac{\partial N}{\partial x} \right]_e\)
The Stiffness Matrix

$[k]$ is usually called the stiffness matrix, because it was first derived (using a different approach…) for structural problems.

For the linear 1D element:

$$[B]_e = \left[ \frac{\partial N}{\partial x} \right]_e = \begin{bmatrix} -\frac{1}{L_e} & \frac{1}{L_e} \end{bmatrix} = \frac{1}{L_e} \begin{bmatrix} -1 & 1 \end{bmatrix}$$

$$[k]_e = \int_{x_i}^{x_j} [B]_e^T [B]_e \, dx = \frac{1}{L_e} \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix}$$
System Stiffness Matrix

The system stiffness matrix is assembled by summing over the elements

$$[k]_{global} = \sum_{elements} ([k]_e)$$
The Boundary Conditions

In the absence of heat sources or sinks along the length of the bar, the heat conduction from the right of one element must equal that into the left of the next one, and the relevant boundary conditions for the system are those at the two ends.

At the left hand end:

\[
\left[ \frac{\partial \phi}{\partial x} \right]_1 = \frac{\Phi_2 - \Phi_1}{L_1}
\]

From the Galerkin formulation:

\[
\left[ N \right]^T \left[ \frac{\partial N}{\partial x} \right] \left\{ \Phi_1 \right\} = \frac{1}{L_e^2} \begin{bmatrix} -L_e & L_e \\ 0 & 0 \end{bmatrix} \left\{ \Phi_2 \right\} = \begin{bmatrix} \Phi_2 - \Phi_1 \\ L_e \end{bmatrix}
\]

At the right hand end:

\[
\left[ \frac{\partial \phi}{\partial x} \right]_{n-1} = \frac{\Phi_n - \Phi_{n-1}}{L_{n-1}}
\]
Numerical Example

\[ T_1 = 20^\circ C \]

\[ \left[ \frac{\partial T}{\partial x} \right]_5 = -30 \, ^\circ C/m \]

\[ 0.4 \, m \]

\[ 0.1 \, m \]
Numerical Example

\[ T_1 = 20 \]

\[ 0.1 \text{ m} \]

\[ \frac{\partial T}{\partial x}_5 = -30 \, ^\circ\text{C/m} \]

\[ [k]_1 = \frac{1}{L_e} \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix} = \frac{1}{0.1} \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix} \]

\[ [k]_2 = \frac{1}{0.1} \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix} \]

\[ [k]_3 = \frac{1}{0.1} \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix} \]

\[ [k]_4 = \frac{1}{0.1} \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix} \]
Numerical Example

\[ T_1 = 20 \]

\[ \begin{bmatrix} 1 & -1 & 0 & 0 & 0 \\ -1 & 1+1 & -1 & 0 & 0 \\ 0 & -1 & 1+1 & -1 & 0 \\ 0 & 0 & -1 & 1+1 & -1 \\ 0 & 0 & 0 & -1 & 1 \end{bmatrix} \]

\[ \frac{\partial T}{\partial x} \bigg|_5 = -30 \, ^\circ C/m \]
Numerical Example

There are no internal sources or sinks and so the ‘forces’ on internal nodes are zero.

The first and last equations can be replaced by the boundary conditions as for the finite difference method

\[
\begin{bmatrix}
1 & 0 & 0 & 0 & 0 \\
-1 & 2 & -1 & 0 & 0 \\
0 & -1 & 2 & -1 & 0 \\
0 & 0 & -1 & 2 & -1 \\
0 & 0 & 0 & -1 & 1 \\
\end{bmatrix}
\begin{bmatrix}
T_1 \\
T_2 \\
T_3 \\
T_4 \\
T_5 \\
\end{bmatrix}
= \begin{bmatrix}
20 \\
0 \\
0 \\
0 \\
-3 \\
\end{bmatrix}
\]

\[
\left[ \frac{\partial T}{\partial x} \right]_5 = -30 \, ^{\circ}\text{C/m}
\]

…. and in this case the system of equations is identical to that derived by the finite difference approach
The 1D Heat equation

Laplaces equation represents the steady state distribution of temperature in the absence of heat sources or sinks.

For the solutions so far we assumed that there was no source or sink except at the boundary, and treated the boundary separately.

More generally:

$$k \frac{\partial^2 \phi}{\partial x^2} + Q = 0$$

Working through the same procedure as illustrated previously, the finite element method will ‘naturally’ yield the full system equations, although special treatment will be required for positions at which the temperature is fixed.
Fundamental to the finite element method is the description of the variation of the governing parameters over the whole of the element in terms of their values at the nodes.

Such descriptions can be difficult to write for elements of complex shape.

It is attractive to produce the analytical derivations in an idealised space, with a mapping to and from real space described by a Jacobian for each element.

This is the basis of the parametric element formulations.
A 2D Isoparametric Element

\[ \phi = N_i \Phi_i + N_j \Phi_j + N_k \Phi_k + N_l \Phi_l = [N]\{\Phi\} \]
A 2D Isoparametric Element

\[ N_l = \frac{1}{4}((1 - \xi)(1 + \eta)) \quad : \quad N_k = \frac{1}{4}((1 + \xi)(1 + \eta)) \]

\[ N_i = \frac{1}{4}((1 - \xi)(1 - \eta)) \quad : \quad N_j = \frac{1}{4}((1 + \xi)(1 - \eta)) \]
From Real to Parametric Space

We need to describe the relationship between the real space in which the element is defined and the idealised, parameterised space in which its shape function is defined.
The Jacobian Matrix

The mapping is described in terms of the Jacobian, $[J]$

By the chain rule of differentiation:

$$\frac{\partial}{\partial \xi} = \frac{\partial x}{\partial \xi} \cdot \frac{\partial}{\partial x} + \frac{\partial y}{\partial \xi} \cdot \frac{\partial}{\partial y}$$

$$\frac{\partial}{\partial \eta} = \frac{\partial x}{\partial \eta} \cdot \frac{\partial}{\partial x} + \frac{\partial y}{\partial \eta} \cdot \frac{\partial}{\partial y}$$

$$\begin{bmatrix}
\frac{\partial}{\partial \xi} \\
\frac{\partial}{\partial \eta}
\end{bmatrix} = \begin{bmatrix}
\frac{\partial x}{\partial \xi} & \frac{\partial y}{\partial \xi} \\
\frac{\partial x}{\partial \eta} & \frac{\partial y}{\partial \eta}
\end{bmatrix} \begin{bmatrix}
\frac{\partial}{\partial x} \\
\frac{\partial}{\partial y}
\end{bmatrix} = [J] \begin{bmatrix}
\frac{\partial}{\partial x} \\
\frac{\partial}{\partial y}
\end{bmatrix}$$
To evaluate the Jacobian we need to know how the \( x \) and \( y \) co-ordinates are related to the \( \xi \) and \( \eta \) ones.

The most common assumption (isoparametric element) is that the relationship is exactly the same as that which describes the distribution of the solution parameters.

\[
x = N_i x_i + N_j x_j + N_k x_k + N_l x_l = [N]\{x\}
\]
\[
y = N_i y_i + N_j y_j + N_k y_k + N_l y_l = [N]\{y\}
\]
The [B] Matrix

\[ \begin{bmatrix} \frac{\partial \phi}{\partial x} \\ \frac{\partial \phi}{\partial y} \end{bmatrix} = [J]^{-1} \begin{bmatrix} \frac{\partial \phi}{\partial \xi} \\ \frac{\partial \phi}{\partial \eta} \end{bmatrix} = [J]^{-1} \begin{bmatrix} \frac{\partial N_i}{\partial \xi} & \frac{\partial N_j}{\partial \xi} & \frac{\partial N_k}{\partial \xi} & \frac{\partial N_l}{\partial \xi} \\ \frac{\partial N_i}{\partial \eta} & \frac{\partial N_j}{\partial \eta} & \frac{\partial N_k}{\partial \eta} & \frac{\partial N_l}{\partial \eta} \end{bmatrix} \begin{bmatrix} \Phi_1 \\ \Phi_2 \\ \Phi_3 \\ \Phi_4 \end{bmatrix} \]
The Element Stiffness Matrix

For the 2D version of Laplace's equation, the element stiffness matrix is:

\[
[k]_e = \int_{\text{Area}} [B]^T [B] \, dx \, dy
\]

[B] is defined in parametric co-ordinates, and it is natural to perform the integral in these co-ordinates

\[
[k]_e = \int_{-1}^{1} \int_{-1}^{1} [B]^T [B] \, \det[J] \, d\xi \, d\eta
\]
Numerical Integration

\[
[k]_e = \int_{-1}^{1} \int_{-1}^{1} [B]^T [B] \det[J] \, d\xi \, d\eta
\]

In the general case the Jacobian, and hence the \([B]\) matrix and \(\det[J]\), vary over the element

The stiffness matrix is usually computed using a numerical integration procedure

‘Gaussian Quadrature’ is commonly used for this purpose

2 x 2 Gaussian quadrature is accurate for a cubic polynomial

The procedure becomes inaccurate when the element becomes ‘too distorted’

There are many measures of distortion
Stiffness Matrix for a Square Element

\[
\begin{align*}
\mathbf{J} &= \begin{bmatrix}
\frac{1}{2} & 0 & 0 \\
0 & 1 & 0 \\
0 & 0 & 1
\end{bmatrix} \\
[k]_e &= \frac{1}{6} \begin{bmatrix}
4 & -1 & -2 & -1 \\
-1 & 4 & -1 & -2 \\
-2 & -1 & 4 & -1 \\
-1 & -2 & -1 & 4
\end{bmatrix}
\end{align*}
\]

Check it!
The global stiffness matrix is:

sparse ..... 
suggests special solution techniques

singular ..... 
requires attention to boundary conditions before solution
Test Problems
Structural Analysis Problem

Stress concentration at hole in plate?
Structural Analysis Problem

10 mm
R=2.5 mm
50 mm

E = 200000 MPa
υ = 0.3
Blood Flow Problem
70% (by area) Axisymmetric Stenosis

Not to Scale

Q = 0.5 litres/min
ρ = 1050 kg/mm³
μ = 0.004 Pa.s