Finite Element Method

Introduction, 1D heat conduction

Siemens Wind Power A/S
External lecturer, Ph.D. Jesper Winther Stærdahl
Finite Element Method

Introduction, 1D heat conduction

Post doc. Johan Clausen
## Lecture plan Finite Element Method I

<table>
<thead>
<tr>
<th>Lecture</th>
<th>Description</th>
<th>References</th>
</tr>
</thead>
<tbody>
<tr>
<td>5:</td>
<td>Solve the exercise heat conduction problems from lectures 1-4 using the commercial FE software Abaqus</td>
<td>Self study</td>
</tr>
</tbody>
</table>
Form and expectations

☐ To give the participants an understanding of the basic elements of the finite element method as a tool for finding approximate solutions of linear boundary value problems. These will be exemplified with examples within stationary heat conduction. After the course the participants should be able to set up their own finite element solution of linear boundary value problems.

☐ We expect that you participate in the given lectures and exercises and ask questions when something is unclear.

☐ Many of the exercises are programming problems, which should be solved on a laptop. Preferably you should work in groups of two. Each group should bring a laptop with MatLab installed.

☐ We expect minor prior knowledge of scientific programming. Basic MatLab programming will be repeated during the lectures.

☐ You should bring pen and paper, and a copy of the lecture notes.
Choices we have made

- The lecture material is mainly slides. The referenced book is additional information. The book is not a beginners book, but can be used for additional information and also for advanced Finite Element problems. We believe the basics can be understood from the slides and attending the lectures.

- You will be programming into a rather extensive matlab toolbox developed by the lecturers. I.e. the overview might be lost in the beginning but you will have a working program, which also will be used in later courses. You will program a simple Finite Element program from nothing during the final lectures.

- We do everything in matlab. You will have your own program with the source code and all the problems this gives. We believe that in order to understand the Finite Element method you need to do some programming instead of just using commercial solvers.

- 4 full lectures including exercise time and 1 self study, instead of 5 short lectures without exercises.
Literature

- Lectures will be given from the slides. The theory and derivations are not directly based on the chapters in the book.
- The book is chosen as a good overall Finite Element book which can be used also for more advanced problems.
- During the lectures references will be given to relevant chapters and pages where the present subject is presented with Cook in parenthesis. E.g. (Cook: chapter 4). These pages are summarized in the lecture plan.
- Relatively cheap!
Beginners literature

- The formulations are based on this book, however, it is expensive and does not cover more advanced issues.
- What is covered in this book is essential what we cover during the lectures and on the slides.
- References to this book will be referenced in parenthesis with OP. E.g. (OP: chapter 4)
What is the Finite Element Method (FEM)?

- A numerical approach for solving partial differential equation, boundary value problems
  - Finite element method
  - Finite difference method
  - Finite volume method
  - Boundary element method
- An approxmative solution.
  - Simplification of geometry
  - Mesh dependent
  - Convergence issues
- Widely used within structural analysis
  - deformation
  - heat conduction
Why use FEM?

- General approach for solving complex structures.
  - Static undetermined systems causes no problems
  - Unsteady problems can be solved (not included in this course)
- Few basic building blocks (elements) can simulate complex problems
- Very suitable for computer solution.
  - Fast solutions
  - Many different load cases
Basic steps of the finite-element method (FEM)

1. Establish strong formulation
   - Partial differential equation
2. Establish weak formulation
   - Multiply with arbitrary field and integrate over element
3. Discretize over space
   - Mesh generation
4. Select shape and weight functions
   - Galerkin method
5. Compute element stiffness matrix
   - Local and global system
6. Assemble global system stiffness matrix
7. Apply nodal boundary conditions
   - temperature/flux/forces/forced displacements
8. Solve global system of equations
   - Solve for nodal values of the primary variables (displacements/temperature)
9. Compute temperature/stresses/strains etc. within the element
   - Using nodal values and shape functions
MatLab FE-program

- main.m (main program runs until "return" plot functions located at the bottom)
  - Coordinates.m (defines node coordinates, done by user)
  - Topology.m (defines the element topology, done by user)
  - BoundaryConditions.m (defines the BC, done by user)
  - calc_globdof.m (determines the global dof numbering, done by program)
  - Assemblering.m (assemble the global stiffness matrix, done by program)
  - solvequations.m (solve the system of equations for primary dof and forces, done by program)
- Plot functions
  - Visualize1Dheat.m (visualize the 1D heat problem with a temperature curve)
  - Visualize2D (visualize all 2D problems including 2D heat conduction and 2D structural elements)
  - Visualize3Dbeam_struct.m (visualize the 3D beam problem with geometry, deformation, beam normals, node and dof numbering)
  - Visualize3Dbeam_struct_secForce.m (visualize the 3D beam problem with geometry, deformation, beam normals, node numbering and section forces)
Exercise: Test that the program can run.

- Start main.m
- Run the plotting of 1D heat conduction
  - Plotting functions are at the bottom of the program

- Output:
  - $u = [0 \ 2]^T$
  - $K = [1 \ 0 ; 0 \ 1]$

- Hints:
  - Open the m-file in the editor F5 runs the program until it reaches a "return"-command or a red circle next to the line numbers (debug mode)
    - To debug use F11 to advance one line (will enter sub-functions). Use F10 to advance one line without entering sub-functions
  - F9 runs the marked section
  - Enable cell-mode. Then ctrl-enter will run the present cell. Cells are divided by %%
Advanced plotting in MatLab using handles

- When a plot is generated in matlab corresponding handles are created.
  - a handle for the figure
  - a handle for the axis
  - a handle for each plot on the figure

- In this handle every information about the plot is defined
- Direct handles are created by putting a variable before the plot

\[
\text{>> } \text{hp} = \text{plot([0 1],[0 1])}
\]

- The information in the handle can be accessed by

\[
\text{>> get(hp)}
\]

- The information can be changed by

\[
\text{>> set(hp,'linewidth',3,'marker','o')}
\]

- new value, is put between ''
- if it is none numerical
Handle for figures and axes can be accessed either by putting a handle when generating them or by gcf (get-current-figure) or gca (get-current-axis) as handles:

```matlab
>> hf = figure
>> ha = axes
>> get(gcf)
>> get(gca)
>> get(hf)
>> get(ha)
```

- Changing values works as for the plot:
  ```matlab
  >> set(hf,'color',[1 0 0])
  >> set(ha,'fontsize',22)
  ```

- Position information, remember correct units:
  ```matlab
  >> set(hf,'units','centimeters','position',[5 5 10 10])
  ```
  - position = [lower_left_corner_x lower_left_corner_y width height]

- Activating a figure for adding axes or axes for plotting:
  ```matlab
  >> figure(hf)
  >> axes(hf)
  ```
Exercise: Make an advanced plot using handles

- figure 1 units in centimeters position lower left corner (5,5) size (10,10)
  - axis units in centimeters, position lower left corner (1,1) size (8,3)
  - xcolor and ycolor is red, figure color is black [r,g,b]=[0,0,0]
- figure 2 units in centimeters position lower left corner (15,5) size (10,10)
  - axis units in centimeters, position lower left corner (1,6) size (8,3)
  - xcolor and ycolor is red, figure color is black [r,g,b]=[0,0,0]

- Hint
  - you need to change units, position and color of the figure-handles
  - you need to change the units, position, xcolor and ycolor of the axis-handle
Make video using handles

- The coordinates for the part that moves are changed through the handle (xdata,ydata) and a 'drawnow' command is made for the plot to be updated

```matlab
figure
hold on
x = 0:0.001:5;
y = sin(x);

hp1 = plot(x,y);
hp2 = plot(x(1),y(1),'ro','markersize',15);
for j=1:size(x,2)
    set(hp2,'xdata',x(j),'ydata',y(j));
drawnow
end
```
Generating an avi-file (only every 100th frame is captured)

```matlab
p=1;
for j= 1:100:size(x,2);
    set(hp2,'xdata',x(j),'ydata',y(j));
    drawnow;
    Mov(p) = getframe(gcf);
    p=p+1;
end
movie2avi(Mov,'test2.avi','Quality',100,'compression','none');
```

Problem:
- Powerpoint cannot play the movie directly

Solution:
- re-code using e.g. virtualdub
- www.virtualdub.org
Plotting many lines in the same plot

- Every plot generates many informations, i.e. many plots become heavy
- Often a FEM solution is plotted using many line pieces because we don't know the geometry in advance. Hence, every element needs to be plotted separately
- One line, one handle independent of the number of points
- By introducing NaN (Not a Number) in the coordinates the line will be broken but still work as one line (i.e. one handle)
Try the following

```matlab
figure
hold on
hp1=plot([0 0],[0 1])
hp2=plot([1 1],[0 1])
hp3=plot([-0.2 1.4],[0.2 0.4])
```

```matlab
figure
hold on
hp4=plot([0 0 1 1 -0.2 1.4],[0 1 0 1 0.2 0.4])
```

```matlab
figure
hold on
hp4=plot([0 0 NaN 1 1 NaN -0.2 1.4],[0 1 NaN 0 1 NaN 0.2 0.4])
```

Try to mark the lines on the three figures

- figure 1: The lines can be marked separately, i.e. changed separately
- figure 2: One line is drawn and can only be changed as one, but this is not what we want. We don't want the connecting lines
- figure 3: The lines are drawn separately but are marked as one, i.e. one handle
Call a function given in a string variable (used in Assembling.m)

- Create 2 files cube.m and square.m
- cube.m

```matlab
function out=cube(x)
out = x^3;
```

- square.m

```matlab
function out=square(x)
out = x^2;
```

- Now try

```matlab
FuncName = 'cube'
Func = str2func(FuncName);
Output = Func(5)
```

- Change the first line and run the second and third again

```matlab
FuncName = 'square'
```

- We can call many different functions by changing a string variable
Ways to store data in matlab

- **Matrices:** use square brackets [] index in parenthesis after matrix name. e.g. A(row column) columns are separated by " " rows by "new line" or ";"
  - Advantages: when the same type and number of data should be stored, can count over index. Should ALWAYS be initialised (A = zeros(2,2);)

  ```
  A = [3 4 ; 5 6]
  A(2,1) = ?
  ```

- **Cells:** use curly brackets {}
  - Advantages: when the same type of data are stored but the number of data varies. Can count over index in both curly and squared brackets

  ```
  B = [3 4 5; 5 6 7]
  C{1} = A; C{2} = B;
  C{2}(2,1) = ?
  ```

- **Structures:** use "." in name
  - Advantages: When data size and type varies. You can call the data by name instead of just by index

  ```
  S.A = A; S.B = B; S.C = C; S.info = 'test'
  S.info(3) = ?
  ```
Example problem 1D stationary heat conduction (Cook: p21-22), (OP chapter 9)

\[ k = 5 \text{ J/(°Cms)}, \quad A = 10 \text{ m}^2 \]

\[ a, x = 2 \text{ m}, \quad T = 0°C \]

\[ b, x = 8 \text{ m}, \quad q = 15 \text{ J/(m}^2\text{s}) \]

- Constant area, A, thermal conductivity, k, and heat supply Q
- Boundary a: constant temperature T
- Boundary b: constant flux q
Step 1: Establish strong formulation for 1D heat conduction (OP: Chapter 4, p48-52)

\[ H(x) = \text{heat inflow [J/s]} \]
\[ A(x) = \text{area [m}^2]\]
\[ Q(x) = \text{heat supply [J/(sm)]} \]
Finite Element Method

Introduction, 1D heat conduction

- Balance or conservation equation
  - Heat inflow equal heat outflow

\[ H + Q \, dx = H + dH \quad \Rightarrow \quad \frac{dH}{dx} = Q \]

- By definition

\[ H = Aq \]

- Material property or constitutive relation

\[ q = -k \frac{dT}{dx} \]

- One-dimensional heat equation

\[ \frac{d}{dx} \left( Ak \frac{dT}{dx} \right) + Q = 0, \quad 0 \leq x \leq L \]

\[ H(x) = \text{heat inflow [J/s]} \]

\[ A(x) = \text{area [m}^2\text{]} \]

\[ Q(x) = \text{heat supply [J/(sm)]} \]

\[ q(x) = \text{flux [J/(sm}^2\text{)]} \]
Finite Element Method

Introduction, 1D heat conduction

- Second order differential equation needs two boundary conditions

\[ \frac{d}{dx} \left( Ak \frac{dT}{dx} \right) + Q = 0, \quad 0 \leq x \leq L \]

- Possible boundary conditions: temperature or temperature gradient (flux)

\[ T(a) = T_a, \quad \frac{dT}{dx} \bigg|_{x=b} = T'_b \quad \Rightarrow \quad q_b = -kT'_b \]

- This is the strong formulation for stationary 1D heat conduction
- Constant A, k, Q with \( T(a)=T_a \), \( q(b)=q_b \), analytical solution can be found (Differential equations and numerical methods 5th semester, Zill&Cullen)

\[ T(x) = \left( \frac{q_b}{k} - \frac{Qb}{Ak} \right) a + \frac{Qa^2}{2} + T_a - \left( \frac{q_b}{k} - \frac{Qb}{Ak} \right) x - \frac{Q}{2Ak} x^2 \]
Step 2: Establish weak formulation (OP: chapter 4, p56-62)

- **Strong form**

\[
\frac{d}{dx} \left( Ak \frac{dT}{dx} \right) + Q = 0, \quad 0 \leq x \leq L
\]

- Multiply with an arbitrary function \(v(x)\) (weight function) and integrate over the pertinent region

\[
\int_{0}^{L} v \left( \frac{d}{dx} \left( Ak \frac{dT}{dx} \right) + Q \right) dx = 0, \quad 0 \leq x \leq L
\]
Use integration by parts of the first term to obtain the same derivative of the weight function and primary variable $T$

$$\int_{a}^{b} f \frac{dg}{dx} \, dx = [fg]_{a}^{b} - \int_{a}^{b} \frac{df}{dx} \, g \, dx, \quad f = v, \quad g = Ak \frac{dT}{dx}$$

$$\int_{0}^{L} v \left( \frac{d}{dx} \left( Ak \frac{dT}{dx} \right) + Q \right) \, dx = 0$$

Weak formulation of 1D stationary heat conduction

$$\int_{0}^{L} \frac{dv}{dx} Ak \frac{dT}{dx} \, dx = \left[ vAk \frac{dT}{dx} \right]_{0}^{L} + \int_{0}^{L} vQ \, dx$$

- $v$ is not completely arbitrary since the above derivation should hold (i.e. $v$ should be one differentiable and defined in the region of integration)
Why are we interested in the weak form?

- The FE-method is based on the weak form.
- The FE-method is an approximate method, i.e. in the strong form we need an approximation to the second derivative of the temperature, where only the first derivative is needed in the weak form at the cost of introducing an arbitrary field.
- The weak form holds when discontinuities occur, the strong form requires a modification (see OP: p60-62)
Finite Element Method
Introduction, 1D heat conduction

Step 3: Discretize over space

- Original problem
  \[ k = 5 \text{ J/(°Cms)}, \; A = 10 \text{ m}^2 \]
  \( a, x = 2 \text{ m}, T = 0^\circ C \)
  \( b, x = 8 \text{ m}, q = 15 \text{ J/(m}^2\text{s)} \)
  \( Q = 100 \text{ J/(sm)} \)

- Discretized problem. Define: nodes, unknown (degree-of-freedom dof) numbering, element numbering, element topology (which nodes define the element)

Nodes: \( n_1, n_2, n_3, n_4 \)
Coordinate: \( x = 2, 4, 6, 8 \)
Elements: \( E, E_2, E_3 \)
Dof: \( T_1, T_2, T_3, T_4 \)
Node number: \( n_1, n_2, n_3, n_4 \)
Step 4: Select shape and weight functions
(OP: chapter 7, p90-94, 98-106)

- The finite element is an approximative method
  - The discrete values $T_i$ of the unknowns are evaluated at the nodes
  - The continuous field $T(x)$ is interpolated using shape functions.
Approximation requirements

- **Convergence criteria**
  - In the limit when the elements are infinitely small, the approximation should be infinitely close to the exact solution

- **Completeness requirements**
  - The approximation must be able to represent an arbitrary constant temperature gradient
  - The approximation must be able to represent an arbitrary constant temperature
  - i.e. the approximation of the temperature should include terms corresponding to a linear polynomial

- **Compatibility or conforming requirement**
  - The approximation of the temperature over element boundaries must be continues

- convergence = completeness + compatibility
Assuming nodal values to be known
- Linear variation of temperature allows a constant temperature gradient
- Simplest one-dimensional element (OP: p98-99)

\[ T(x) = -\frac{1}{L} (x - x_j)T_i + \frac{1}{L} (x - x_i)T_j \]

- Test

\[
T(x_i) = -\frac{1}{L} (x_i - x_j)T_i + \frac{1}{L} (x_i - x_i)T_j = T_i \\
L = x_j - x_i \\
T(x_j) = -\frac{1}{L} (x_j - x_j)T_i + \frac{1}{L} (x_j - x_i)T_j = T_j
\]

- Matrix notation

\[ T(x) = N\mathbf{a}, \quad N = \begin{bmatrix} -\frac{1}{L} (x - x_j) & \frac{1}{L} (x - x_i) \end{bmatrix}, \quad \mathbf{a} = \begin{bmatrix} T_i \\ T_j \end{bmatrix} \]
- Weight functions (see OP: chapter 8, p142-156)

\[ v(x) = V(x)c, \quad V = [V_1 \ V_2 \ \ldots \ V_n], \quad c = \begin{bmatrix} c_1 \\ c_2 \\ \vdots \\ c_n \end{bmatrix} \]

- FE-method uses the Galerkin method (OP: p152-153)
  - weight functions = shape functions

\[ V(x) = N(x) \]

- Transpose of a scalar field equals the field

\[ v(x) = Nc = c^T N^T \]
Step 5: Compute element stiffness matrix

- If the weak formulation holds for the entire field, it also holds for part of the field, i.e. integration is done over one element

\[ \int_{x_i}^{x_j} \frac{dv}{dx} Ak \frac{dT}{dx} dx = -[vAq]_{x_i}^{x_j} + \int_{x_i}^{x_j} vQ dx \]

- Insert the temperature field and arbitrary field into the weak formulation

\[ T(x) = N(x)a, \quad N(x) = \left[-\frac{1}{L}(x-x_j) \quad \frac{1}{L}(x-x_i)\right], \quad a = \begin{bmatrix} T_i \\ T_j \end{bmatrix} \]

\[ v(x) = c^T N^T(x) \]

\[ c^T \int_{x_i}^{x_j} \frac{dN^T}{dx} Ak \frac{dN}{dx} dx a = -c^T [N^T Aq]_{x_i}^{x_j} + c^T \int_{x_i}^{x_j} N^T Q dx \]

- \( c^T \) cancels from the equation!!!!
Finite Element Method

Introduction, 1D heat conduction

In compact form

\[ Ka = f \]

\[
K = \int_{x_i}^{x_j} B^T A_k B \, dx, \quad B = \frac{dN}{dx}
\]

\[
f = -[N^T A_q]_{x_i}^{x_j} + \int_{x_i}^{x_j} N^T Q \, dx
\]

\[
a = \begin{bmatrix} T_i \\ T_j \end{bmatrix}
\]
Exercise: Compute by hand the element stiffness matrix and force vector

- Assume constant A, k, Q
- \( q(x_i) = q_i, \ q(x_j) = q_j \)
- Need the derivative of the shape functions

\[
K = \int_{x_i}^{x_j} B^T A k B \, dx, \quad B = \frac{dN}{dx}, \quad K = [2 \times 2]
\]

\[
f = -[N^T Aq]_{x_i}^{x_j} + \int_{x_i}^{x_j} N^T Q \, dx, \quad f = [2 \times 1]
\]

\[
N(x) = \begin{bmatrix} -\frac{1}{L}(x - x_j) & \frac{1}{L}(x - x_i) \end{bmatrix} \quad L = x_j - x_i
\]
Local and global coordinates (OP: p191-193)

- Integration and shape functions are defined in a local coordinate system, i.e. along the beam axis.
- Often we wish to normalize the integration. In this case it is not necessary but for the case of argument and introduction to 2D and numerical integration.

Global coordinate system

Local coordinate system

Integration and shape functions are defined in a local coordinate system, i.e. along the beam axis. Often we wish to normalize the integration. In this case it is not necessary but for the case of argument and introduction to 2D and numerical integration.
Transformation between coordinate systems

\[ x = \frac{L}{2} \xi + \frac{1}{2} (x_i + x_j) \]

Test

\[ L = x_j - x_i \]

\[ \xi = -1 \quad \Rightarrow \quad x = -\frac{1}{2} (x_j - x_i) + \frac{1}{2} (x_i + x_j) = x_i \]

\[ \xi = 0 \quad \Rightarrow \quad x = \frac{1}{2} (x_i + x_j) \]

\[ \xi = 1 \quad \Rightarrow \quad x = \frac{1}{2} (x_j - x_i) + \frac{1}{2} (x_i + x_j) = x_j \]
Integration in local coordinate system

\[ x = \frac{L}{2} \xi + \frac{1}{2} (x_i + x_j) \quad \Rightarrow \quad dx = \frac{L}{2} d\xi \]

\[ N(x) = \begin{bmatrix} -\frac{1}{L}(x - x_j) & \frac{1}{L}(x - x_i) \end{bmatrix} \quad \Rightarrow \quad N(\xi) = \begin{bmatrix} \frac{1}{2}(1 - \xi) & \frac{1}{2}(1 + \xi) \end{bmatrix} \]

\[ B = \frac{dN}{dx} = \frac{dN}{d\xi} \frac{d\xi}{dx} = \begin{bmatrix} -\frac{1}{2} & \frac{1}{2} \end{bmatrix} \frac{2}{L} = \begin{bmatrix} -1 & 1 \end{bmatrix} \frac{1}{L} \]

\[ K = \int_{x_i}^{x_j} B^T A k B dx = \frac{L}{2} \int_{-1}^{1} B^T A k B d\xi \]

- constant A and k

\[ K = \frac{Ak}{2L} \int_{-1}^{1} \begin{bmatrix} -1 \\ 1 \end{bmatrix} \begin{bmatrix} -1 & 1 \end{bmatrix} d\xi = \frac{Ak}{2L} \int_{-1}^{1} \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix} d\xi = \frac{Ak}{L} \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix} \]
Exercise: Program the shape functions

$$\mathbf{N}(x) = \begin{bmatrix} -\frac{1}{L}(x - x_j) & \frac{1}{L}(x - x_i) \end{bmatrix}$$

- make a function `shape_1d_heat.m`

```matlab
function [N,B] = shape_1d_heat(x,eCoord)

% Shape functions for 1D heat element
% INPUT
% x = coordinate where N and B is evaluated [1x1]
% eCoord = [x1 y1 z1 ; x2 y2 z2] Nodal coordinates
% OUTPUT
% N = shape functions evaluated at coordinate x [1x2]
% B = derivative of shape functions evaluated at coordinate x [1x2]
```
step 6: Assemble global system stiffness matrix

(OP: p184-191)

- Global system of equations

\[Ka = f, \quad a = \begin{bmatrix} T_1 \\ T_2 \\ \vdots \\ T_{n_{dof}} \end{bmatrix}, \quad K = [n_{dof} \times n_{dof}], \quad f = [n_{dof} \times 1]\]

- The global system stiffness matrix \([n_{dof} \times n_{dof}]\) is assembled from all the element stiffness matrices \([2x2]\) according to the global numbering of the degrees-of-freedom.
Example: 4 dofs, 3 elements

\[ A_1, k_1, L_1 \quad A_2, k_2, L_2 \quad A_3, k_3, L_3 \]

\[ T_1 \quad T_2 \quad T_3 \quad T_4 \]

\[ E_1 \quad E_2 \quad E_3 \]

\[ n_1 \quad n_2 \quad n_3 \quad n_4 \]

\[ x_1 \quad x_2 \quad x_3 \quad x_4 \]

From node number the dof-number is found

From node coordinates the length is defined

Need information (input) about
- node coordinates
- element definition or topology (which nodes are used, section (A) and material (k))
Finite Element Method

Introduction, 1D heat conduction

- Local dof numbering vs. global dof numbering

\[ A_1, k_1, L_1 \quad A_2, k_2, L_2 \quad A_3, k_3, L_3 \]

Local numbering

Global numbering

- Local system of equations, see exercise slide 36

\[
\frac{A_i k_i}{L_i} \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix} \begin{bmatrix} T_1 \\ T_2 \end{bmatrix} = \frac{Q L_i}{2} \begin{bmatrix} 1 \\ 1 \end{bmatrix} - A_i \begin{bmatrix} q_1 \\ -q_2 \end{bmatrix}
\]
Global system of equations

In a linear system we can add stiffnesses and loads (in this case heat supply)

\[
\begin{bmatrix}
- & - & - & - \\
- & - & - & - \\
- & - & - & - \\
- & - & - & - \\
\end{bmatrix}
\begin{bmatrix}
T_1 \\
T_2 \\
T_3 \\
T_4 \\
\end{bmatrix}
= 
\begin{bmatrix}
- \\
- \\
- \\
- \\
\end{bmatrix}
- 
\begin{bmatrix}
- \\
- \\
- \\
- \\
\end{bmatrix}
\]

Element 1: Global numbering

\[
\frac{A_1 k_1}{L_1} \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix} \begin{bmatrix} T_1 \\ T_2 \end{bmatrix} = \frac{QL_1}{2} \begin{bmatrix} 1 \\ 1 \end{bmatrix} - A_1 \begin{bmatrix} q_1 \\ -q_2 \end{bmatrix}
\]

\[
\begin{bmatrix}
\frac{A_1 k_1}{L_1} & - \frac{A_1 k_1}{L_1} & - & - \\
- \frac{A_1 k_1}{L_1} & \frac{A_1 k_1}{L_1} & - & - \\
- & - & - & - \\
- & - & - & - \\
\end{bmatrix}
\begin{bmatrix}
T_1 \\
T_2 \\
T_3 \\
T_4 \\
\end{bmatrix}
= 
\begin{bmatrix}
\frac{QL}{2} \\
\frac{QL}{2} \\
- \\
- \\
\end{bmatrix}
- 
\begin{bmatrix}
A_1 q_1 \\
-A_1 q_2 \\
- \\
- \\
\end{bmatrix}
\]
Finite Element Method

Introduction, 1D heat conduction

Element 2: Global numbering

\[
\frac{A_2 k_2}{L_2} \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix} \begin{bmatrix} T_2 \\ T_3 \end{bmatrix} = \frac{QL_2}{2} \begin{bmatrix} 1 \\ 1 \end{bmatrix} - A_2 \begin{bmatrix} q_2 \\ -q_3 \end{bmatrix}
\]

\[
\begin{bmatrix}
\frac{A_1 k_1}{L_1} \\
-\frac{A_1 k_1}{L_1} \\
\frac{A_2 k_2}{L_2} \\
-\frac{A_2 k_2}{L_2} \\
0 \\
0
\end{bmatrix}
\begin{bmatrix} T_1 \\ T_2 \\ T_3 \\ T_4 \end{bmatrix} = \begin{bmatrix}
\frac{QL_1}{2} \\
\frac{QL_2}{2} \\
\frac{QL_2}{2} \\
\frac{QL_3}{2} \\
0 \\
0
\end{bmatrix} + \begin{bmatrix}
A_1 q_1 \\
-A_1 q_2 + A_2 q_2 \\
-A_2 q_3 \\
-A_2 q_3 + A_3 q_3 \\
0 \\
0
\end{bmatrix}
\]

Element 3: Global numbering

\[
\frac{A_3 k_3}{L_3} \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix} \begin{bmatrix} T_3 \\ T_4 \end{bmatrix} = \frac{QL_3}{2} \begin{bmatrix} 1 \\ 1 \end{bmatrix} - A_3 \begin{bmatrix} q_3 \\ -q_4 \end{bmatrix}
\]

\[
\begin{bmatrix}
\frac{A_1 k_1}{L_1} \\
-\frac{A_1 k_1}{L_1} \\
\frac{A_2 k_2}{L_2} \\
-\frac{A_2 k_2}{L_2} \\
0 \\
0
\end{bmatrix}
\begin{bmatrix} T_1 \\ T_2 \\ T_3 \\ T_4 \end{bmatrix} = \begin{bmatrix}
\frac{QL_1}{2} \\
\frac{QL_2}{2} \\
\frac{QL_2}{2} \\
\frac{QL_3}{2} \\
0 \\
0
\end{bmatrix} + \begin{bmatrix}
A_1 q_1 \\
-A_1 q_2 + A_2 q_2 \\
-A_2 q_3 + A_3 q_3 \\
-A_2 q_3 + A_3 q_3 \\
0 \\
0
\end{bmatrix}
\]
The assembling procedure is as follows

1. Determine the local stiffness matrix
2. determine the global number of dof corresponding to the local dof for the element
3. add the components of the local stiffness matrix to the rows and columns of the global stiffness matrix corresponding to the global dof numbers
4. repeat 1-3 until all contributions from all elements have been added.

In MatLab this is done in assembling.m

\[ K_{\text{gDof,gDof}} = K_{\text{gDof,gDof}} + K_e; \]
In order to be able to create a global system matrix we need to give information about:

- Material for each element
- Section dimensions for each element
- Coordinates for each node (and a numbering)
- Topology for each element (which nodes are in the element)
  - dof numbering is given from the node numbering and number of dofs per node (one in this case)

See `calc_globdof.m` for numbering of global dof:

- `GlobDof = [nDof1 GDof1 GDof2 ...]`
  - dof numbering of each node
- `nDof = total number of Dof`
Exercise: Define the following problem in the program

\[ k = 5 \text{ J}/(\text{°Cms}), \ A = 10 \text{ m}^2 \]

\[ a, x = 2 \text{ m}, T = 0\text{°C} \quad b, x = 8 \text{ m}, q = 15 \text{ J}/(\text{m}^2\text{s}) \]

- Discretize into 3 elements
  - change coordinates.m, topology.m, material and section definitions
  - Explain how the structure of section and material is defined
- plot the solution

\[ T_1 \quad E_1 \quad T_2 \quad E_2 \quad T_3 \quad E_3 \quad T_4 \]

\[ n_1 \quad n_2 \quad n_3 \quad n_4 \]

\[ x = 2 \quad x = 4 \quad x = 6 \quad x = 8 \]
Exercise: Enter the correct stiffness matrix into the program

- Look through assemblering.m and identify the steps in slide 46
  - Used functions:
    - `elemInfo`: provides element type, section number, Name of stiffness function (see slide 20), coordinates for the element, global Dof numbering
    - `secType`: provides the name of the section function giving the constitutive relation
- Modify `K_1D_heat.m`
  - `Ke = K_1D_heat(eCoord, ConstRel)`
  - Input `eCoord`, `ConstRel`
  - Output `Ke [2x2]`
  - Output `K [4x4]`

\[
K = \begin{bmatrix}
25 & -25 \\
-25 & 25 \\
-25 & 50 & -25 & 0 \\
0 & -25 & 50 & -25 & 0 \\
0 & 0 & -25 & 25
\end{bmatrix}
\]

\[
K = \begin{bmatrix}
25 & -25 & 0 & 0 \\
-25 & 25 & -25 & 0 \\
0 & -25 & 50 & -25 & 0 \\
0 & 0 & -25 & 25
\end{bmatrix}
\]
Step 7: Apply nodal boundary conditions
temperature/flux

Why do we need boundary conditions?

\[ Ka = f, \quad a = \begin{bmatrix} T_1 \\ T_2 \\ \vdots \\ T_{ndof} \end{bmatrix}, \quad K = [ndof \times ndof], \quad f = \begin{bmatrix} f_1 \\ f_2 \\ \vdots \\ f_{ndof} \end{bmatrix} \]

- \( K \) is known from material and geometric input \([ndof \times ndof]\)
- \( a \) contains the unknown temperatures \([ndof \times 1]\)
- \( f \) contains the unknown loads (heat supply) \([ndof \times 1]\)
- i.e. 2 x ndof unknowns but only ndof equations
- We need to define ndof of the unknowns,
- We define either temperature or load in each node and solve for the remaining unknowns, i.e. for the temperature where the load is defined and for the load where the temperature is defined.
The force vector, see slide 45

\[ f = \int_{x_i}^{x_j} N^T Q dx - [N^T Aq]_{x_i}^{x_j} \]

\[ f = \begin{bmatrix} \frac{QL_1}{2} \\ \frac{QL_1^2}{2} + \frac{QL_2}{2} \\ \frac{QL_2^2}{2} + \frac{QL_3}{2} \\ \frac{QL_3^2}{2} \end{bmatrix} + \begin{bmatrix} A_1 q_1 \\ 0 \\ 0 \\ -A_3 q_4 \end{bmatrix} \]

Q is the heat supply assumed constant, \( q_i \) are the boundary nodal fluxes
Exercise: Determine the boundary conditions and enter them into the program

- modify BoundaryCondition.m
- Type 1 is a temperature, type 2 is a heat supply
- Calculate the load vector by hand and enter nodal values

\[ k = 5 \text{ J/(°Cm)}, \quad A = 10 \text{ m}^2 \]

\[ a, \ x = 2 \text{ m}, \ T = 0^\circ \text{C} \]

\[ b, \ x = 8 \text{ m}, \ q = 15 \text{ J/(m}^2\text{s)} \]

\[ Q = 100 \text{ J/(sm)} \]

\[ f = \begin{bmatrix} \frac{QL_1}{2} + \frac{QL_2}{2} \\ \frac{QL_2}{2} + \frac{QL_3}{2} \\ \frac{QL_3}{2} \end{bmatrix} + \begin{bmatrix} A_1q_1 \\ 0 \\ 0 \end{bmatrix} \]
Step 8: Solve global system of equations

- Global system with boundary conditions entered

\[
\begin{bmatrix}
K_{11} & K_{12} & K_{13} & K_{14} \\
K_{21} & K_{22} & K_{23} & K_{24} \\
K_{31} & K_{32} & K_{33} & K_{34} \\
K_{41} & K_{42} & K_{43} & K_{44}
\end{bmatrix}
\begin{bmatrix}
a \\
T_2 \\
T_3 \\
T_4
\end{bmatrix}
= 
\begin{bmatrix}
f_1 \\
b \\
c \\
d
\end{bmatrix}
\]

- Left hand unknowns are called primary unknowns (temperature)
- First we identify the equations with primary unknowns (in this case row 2-4)
- The defined temperatures are multiplied with their respective columns of \( \mathbf{K} \) and transferred to the right-hand side

\[
\begin{bmatrix}
K_{22} & K_{23} & K_{24} \\
K_{32} & K_{33} & K_{34} \\
K_{42} & K_{43} & K_{44}
\end{bmatrix}
\begin{bmatrix}
T_2 \\
T_3 \\
T_4
\end{bmatrix}
= 
\begin{bmatrix}
b \\
c \\
d
\end{bmatrix}
- a
\begin{bmatrix}
K_{21} \\
K_{31} \\
K_{41}
\end{bmatrix}
\]
The reduced system can be solved

\[ \mathbf{a}_{\text{red}} = \mathbf{K}^{-1}_{\text{red}} \mathbf{f}_{\text{red}} \]

Finally, the unknown heat supplies can be determined from the remaining equations.

The reduced solutions should enter the full solution, i.e. the boundary conditions and solved temperature should be combined in one vector for plotting.

This is done in MatLab in solveequations.m

\[ \text{ured} = \text{Kred}\backslash\text{Fred} \]
Step 9: Compute temperature/fluxes within elements

- Nodal values of dofs (temperature) are given from the solution of the global system.
- Internal values are determined using shape functions and nodal values corresponding to the relevant element. See slide 32.

\[ T(x) = \mathbf{N}(x) \mathbf{a}, \quad \mathbf{N}(x) = \begin{bmatrix} -\frac{1}{L} (x - x_j) & \frac{1}{L} (x - x_i) \end{bmatrix}, \quad \mathbf{a} = \begin{bmatrix} T_i \\ T_j \end{bmatrix} \]

- Fluxes are determined from the derivative of the temperature. See slide 24.

\[ q(x) = -k \frac{dT(x)}{dx} = -k \frac{d\mathbf{N}(x)}{dx} \mathbf{a} = -k \mathbf{B}(x) \mathbf{a}, \quad \mathbf{B}(x) = \frac{1}{L} \begin{bmatrix} -1 & 1 \end{bmatrix} \]

- I.e. internal values are determined from nodal values multiplied with shape functions evaluated at the respective coordinate.
Overall steps in MatLab FE-program

- **Input**
  - section, material, node coordinates, topology, boundary conditions
- **Generate global dof numbering**
- **Assemble global stiffness matrix**
  - loop over elements
    - element section and material gives the constitutive relation
    - Calculate element stiffness matrix from coordinates and constitutive relation
    - global numbering of element dofs
    - add element stiffness matrix to the system stiffness matrix according to the global dof numbering
- **Setup the global system of equations by introducing boundary conditions and solve for dofs and forces**
- **Post processing**
  - determine internal fields (temperature/flux) from nodal values and shape functions
  - visualize results
Exercise: Get familiar with the program

- Change boundary conditions
- Change topology
- Change material and sections
  - try with different material and sections in the same analysis
- Change the discretization (element coordinates)
  - should all be on the x-axis, i.e. y- and z-coordinate equal zero.
- Compare with analytical solution, slide 25.
- Determine the temperature variation of the following problem. Discretize into 5 elements, for the area use the height at the middle of the element (thickness=1.0 m)

\[ x = 0 \text{ m}, \ h(0) = 1.0 \text{ m}, \ T = 0^\circ\text{C} \]

\[ x = [0.15 \ 0.45 \ 0.75 \ 1.05 \ 1.35] \]

\[ h = [0.87 \ 0.66 \ 0.53 \ 0.46 \ 0.47] \]

\[ k = 5 \text{ J/(°C m)}, \ t = 1.0 \text{ m} \]

\[ h(x) = 0.4x^2 - 0.93x + 1 \]
Thank you for your attention