MHA - FEM Assignment 2

Group 33
Nils Helgesson
Gabriel Wendel

November 2023

Task 1

Task 1 a) State the strong form for the stationary heat flow

Equation states the general strong form for stationary heat flow in 2D.

$$\mathrm{div}(\mathbf{q}t) = tQ \tag{1}$$
$$\mathbf{q} = -\mathbf{D}\nabla T \tag{2}$$

$\mathbf{D}$ is a matrix containing the material's heat conductive coefficient, k. For this problem we have isentropic heat flow, i.e. the material has no preferred direction.

$$\mathbf{D} = k \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} = k\mathbf{I}$$

There is no external heat supply according to the problem description, $Q = 0$ and we assume constant thickness, t. Equation x becomes

$$\mathrm{div}(\mathbf{q}t) = 0 \tag{3}$$

The geometry can be divided into four boundaries, $\Gamma_1, \Gamma_2, \Gamma_3$ and $\Gamma_{ins,sym}$, see Figure 1.

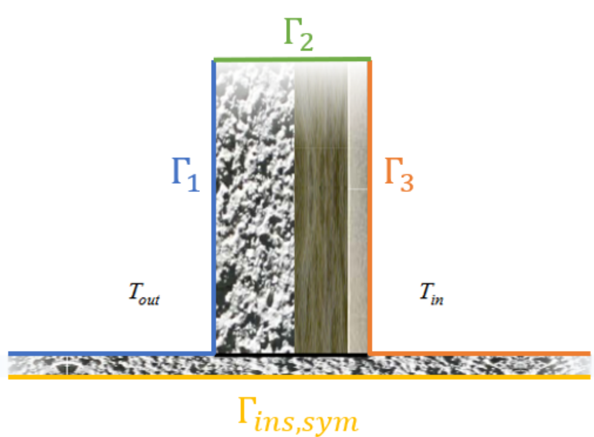


Figure 1: Wall divided into four boundaries $\Gamma_1, \Gamma_2, \Gamma_3$ and $\Gamma_{ins,sym}$

The problem is symmetric with regards to a horizontal axis and can be considered insulated along this symmetry line, i.e. no heat flow through the boundary, $q_n = 0$. Boundaries Γ_1 and Γ_3 are Robin boundaries, and they experience heat exchange between air and the wall. we assume that boundary Γ_2 is insulated.

$$
\begin{aligned}
&\Gamma_1 : q_n = \alpha(T - T_{out})\\
&\Gamma_2 : q_n = 0\\
&\Gamma_3 : q_n = \alpha(T - T_{in})\\
&\Gamma_{ins,sym} : q_n = 0
\end{aligned}
$$

Thus the strong form can be expressed as:

$$
\begin{cases}
\nabla \mathbf{q}^T t = 0\\
\Gamma_1 : q_n = \alpha(T - T_{out})\\
\Gamma_2 : q_n = 0\\
\Gamma_3 : q_n = \alpha(T - T_{in})\\
\Gamma_{ins,sym} : q_n = 0
\end{cases}
\tag{4}
$$

Task 1 b)

Derive the weak form

Similar to the 1D-case we multiply with a test function, $v(x, y)$, and integrate over the domain, Ω.

$$
\int_\Omega v \nabla^T(\mathbf{q}) t d\Omega = 0 \tag{5}
$$

Green-Gauss theorem is then applied on Equation 5 in order to differentiate the weight function instead of divergence operator $\mathbf{q}$:

$$
\int_\Omega v \nabla^T(\mathbf{q}) t d\Omega = -\int_\Omega (\nabla v)^T \mathbf{q} t d\Omega + \int_\Gamma^\Omega v \mathbf{q}^T \mathbf{n} t d\Gamma \tag{6}
$$

$$
\text{Where} \quad \mathbf{q}^T \mathbf{n} = q_n
$$

Left hand side of Equation 6 equals 0 according to Equation 5:

$$
\int_\Omega (\nabla v)^T \mathbf{q} t d\Omega = \int_\Gamma^\Omega v q_n t d\Gamma \tag{7}
$$

Insert expression for heat flux in Equation 2 (Fourier's law) into Equation 7:

$$
\int_\Omega \nabla v^T (\mathbf{D} \nabla T) t d\Omega = -\int_\Gamma^\Omega v q_n t d\Gamma \tag{8}
$$

As previously stated there is no heat transfer ($q_n = 0$) across the insulated boundaries Γ_2 and $\Gamma_{ins,sym}$. Thus integrals over these boundaries can be disregarded. Left hand side of Equation 8 can thus be written as:

$$
\begin{aligned}
\int_{\Gamma}^{\Omega} v q_n t d\Gamma &= \int_{\Gamma_1} v q_n t d\Gamma + \int_{\Gamma_2} v q_n t d\Gamma + \int_{\Gamma_3} v q_n t d\Gamma + \int_{\Gamma_{ins,sym}} v q_n t d\Gamma \\
&= \int_{\Gamma_1} v \alpha_w (T - T_{out}) t d\Gamma + \int_{\Gamma_3} v \alpha_w (T - T_{in}) t d\Gamma \qquad (9)
\end{aligned}
$$

Absence of essential (Dirichlet) boundaries results in the following expression of the weak form:

$$
\int_{\Omega} \nabla v^T (\mathbf{D} \nabla T) t d\Omega = -\int_{\Gamma_1} v \alpha_w (T - T_{out}) d\Gamma - \int_{\Gamma_3} v \alpha_w (T - T_{in}) d\Gamma
$$

Derive FE-form

Use linear shape function, $N(x, y)$, to approximate temperature:

$$
\begin{aligned}
T(x, y) &\approx T_h(x, y) = \mathbf{N}\mathbf{a} = \mathbf{a}^T \mathbf{N}^T \\
\nabla T(x, y) &\approx \nabla T_h(x, y) = \nabla(\mathbf{N}\mathbf{a}) = \mathbf{B}\mathbf{a} \qquad (10)
\end{aligned}
$$

The weight function can be determined using Galerkin's method:

$$
\begin{aligned}
v(x, y) &= \mathbf{N}\mathbf{c} = \mathbf{c}^T \mathbf{N}^T \\
\nabla v(x, y) &= \nabla \mathbf{N}\mathbf{c} = \mathbf{B}\mathbf{c} \qquad (11)
\end{aligned}
$$

The weak form can then be reformulated using $T_h(x, y)$ and $v(x, y)$ from Equation 10 and 11:

$$
\begin{aligned}
&\int_{\Omega} \mathbf{c}^T \mathbf{B}^T \mathbf{D} \mathbf{B} d\Omega \mathbf{a} = -\int_{\Gamma_1} \mathbf{c}^T \mathbf{N}^T \alpha_w (\mathbf{N}\mathbf{a} - T_{out}) d\Gamma - \int_{\Gamma_3} \mathbf{c}^T \mathbf{N}^T \alpha_w (\mathbf{N}\mathbf{a} - T_{in}) d\Gamma \\
\Rightarrow\ &\mathbf{c}^T \left[\int_{\Omega} \mathbf{B}^T \mathbf{D} \mathbf{B} d\Omega \mathbf{a} + \int_{\Gamma_1} \mathbf{N}^T \alpha_w (\mathbf{N}\mathbf{a} - T_{out}) d\Gamma + \int_{\Gamma_3} \mathbf{N}^T \alpha_w (\mathbf{N}\mathbf{a} - T_{in}) d\Gamma \right] = 0 \qquad (12)
\end{aligned}
$$

Since $\mathbf{c}$ is an arbitrary vector:

$$
\left[\underbrace{\int_{\Omega} \mathbf{B}^T \mathbf{D} \mathbf{B} d\Omega}_{\mathbf{K}} + \underbrace{\int_{\Gamma_1} \mathbf{N}^T \mathbf{N} \alpha d\Gamma + \int_{\Gamma_3} \mathbf{N}^T \mathbf{N} \alpha d\Gamma}_{\mathbf{K_c}} \right] \mathbf{a} = \underbrace{\int_{\Gamma_1} \mathbf{N}^T \alpha T_{out} d\Gamma + \int_{\Gamma_3} \mathbf{N}^T \alpha T_{in} d\Gamma}_{\mathbf{f_c}} \qquad (13)
$$

Thus the global FE-form can be stated as:

$$
(\mathbf{K} + \mathbf{K}_c)\mathbf{a} = \mathbf{f}_c \qquad (14)
$$

Task 1 c)

See appendix

Task 1 d)

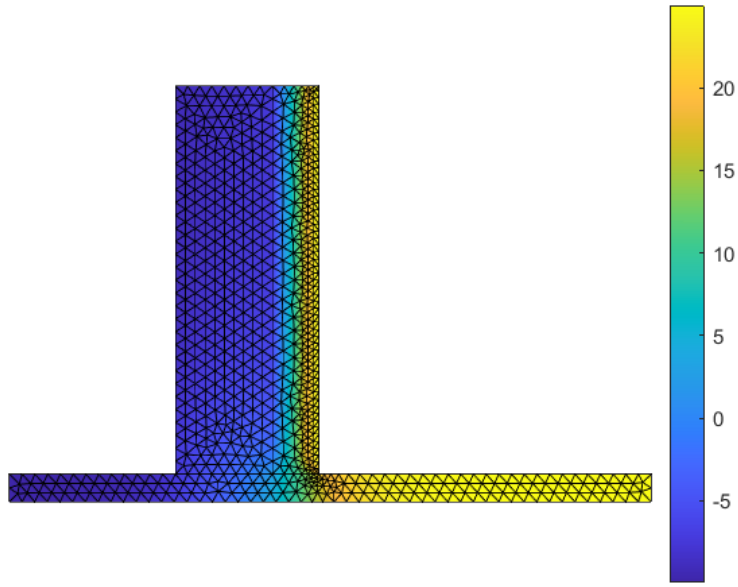


Figure 2: Temperature distribution through the wall, using the finer mesh.

Task 1 e)

The heat flow out of the wall is given by:

$$Q(\Delta T) = \bar{q}(\Delta T)H + \psi \Delta T \tag{15}$$

Using the FE model we calculate $Q(\Delta T) = Q_{out}$ along the outer boundary. The calculations are done element-wise and then summed up. Note, since Q is defined positive if heat is added, we will get a negative sign on the heat outflow. But for transmittance calculations we want the absolute value.

$$Q^e = \alpha_e L_e t \left(\frac{T_i^e + T_j^e}{2} - T_{ext,e} \right) \tag{16}$$

$$Q_{out}^{half} = \sum Q_i^e \tag{17}$$

$$Q_{out} = 2Q_{out}^{half} \tag{18}$$

$$Q_{out}^{coarse} = 41.79W/m \tag{19}$$

$$Q_{out}^{fine} = 41.86W/m \tag{20}$$

$$\tag{21}$$

The analytic solution for q is given by:

$$\bar{q}(\Delta T) = \frac{\Delta T}{R_{ekv}} \tag{22}$$

$$R_{ekv} = \frac{1}{\alpha} + \frac{h_1}{k_1} + \frac{h_2}{k_2} + \frac{h_3}{k_3} + \frac{1}{\alpha} \tag{23}$$

$$q_{analytic} = 7.25W/m^2 \tag{24}$$

We can now calculate the the linear thermal transmittance as

$$\psi = \frac{Q(\Delta T) - \bar{q}(\Delta T)H}{\Delta T}$$
$$\psi_{coarse} = 0.885\ W/mK$$
$$\psi_{fine} = 0.883\ W/mK$$

Task 2 (voluntary)

To establish a benchmark for comparing the heat flow obtained through MATLAB in Task 1, a simulation was conducted in ANSYS.

Figure 3 illustrates half of the wall, meshed with a triangular mesh and a mesh size of 1×10^{-2} m.

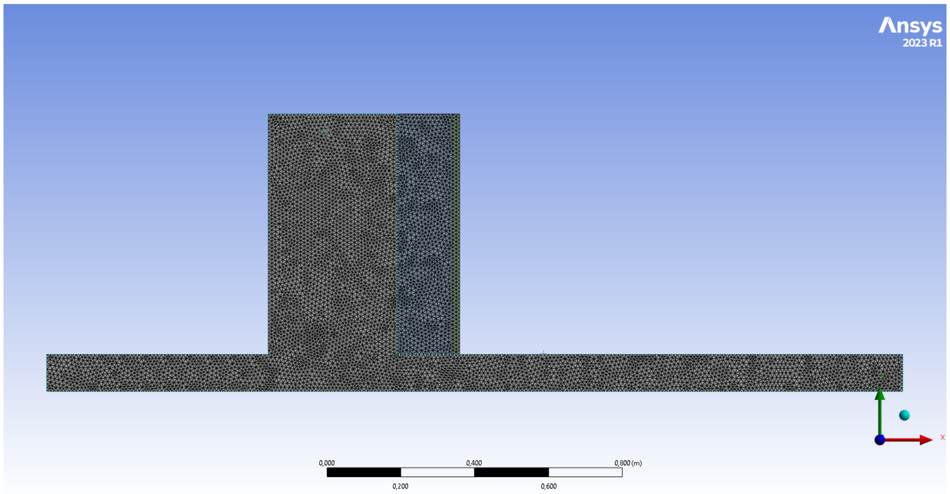


Figure 3: Triangular mesh with mesh size 1e-2m.

Material properties for concrete, plaster and mineral wool was implemented. We assumed same heat transfer coefficients, k, as in Task 2, see Table 1. Figure 4 depicts the temperature distribution through the wall.

Table 1: Heat transfer coefficients used in Task 1 and Task 2.

Material	**Heat transfer coefficient, k** [W/m^2/K]
Concrete	1.5
Mineral wool	0.035
Plaster	0.17

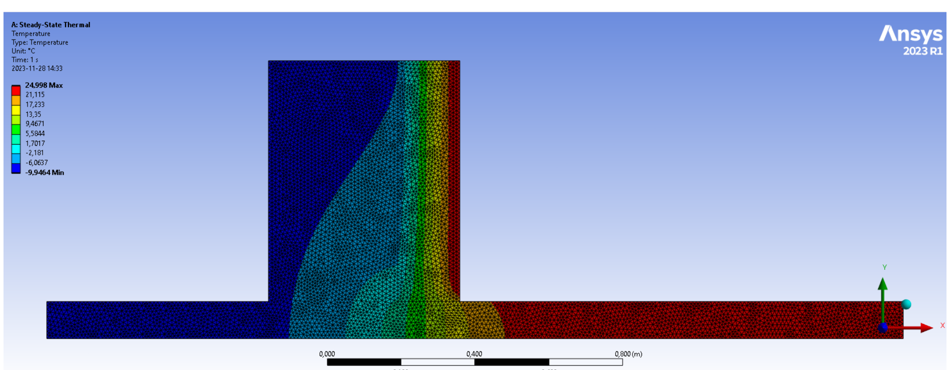


Figure 4: Temperature distribution.

Figure 5 illustrates heta flux through the wall.

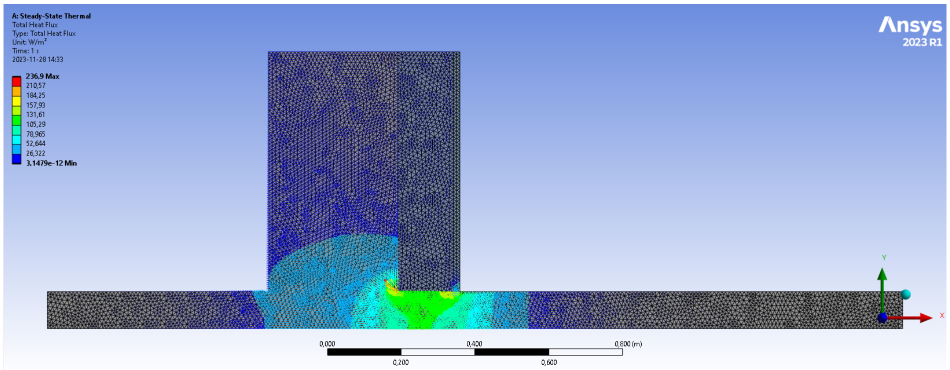
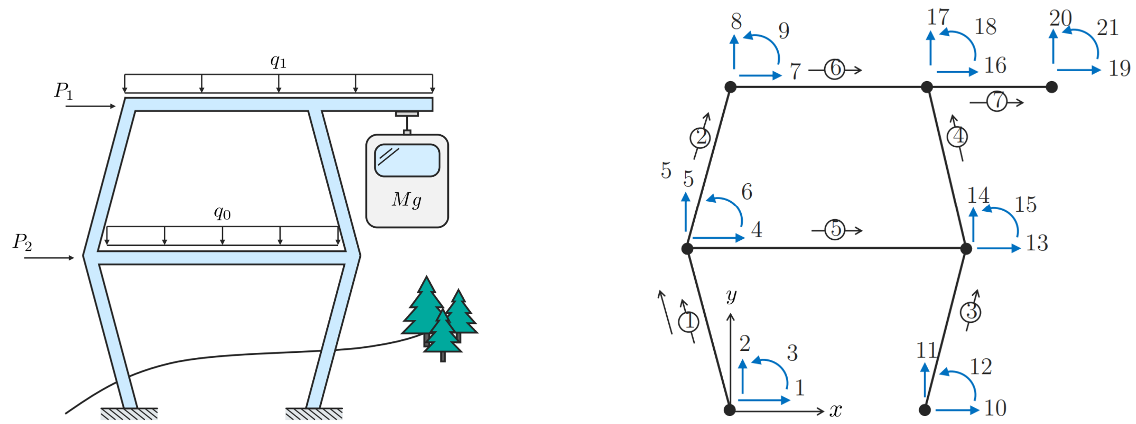


Figure 5: Heat flux displayed with arrows.

Heat flow calculated in ANSYS = ± 14.534W. As the geometry represents only half of the wall, this value must be doubled to account for the entire wall, $Q_{ANSYS} = 29.068$ W. This is somewhat comparable to the heat flow calculated in MATLAB ($41.79, \mathrm{W}$). The variation could be attributed to differences in meshing quality, as we employed a finer mesh in ANSYS.

Task 3

The gondola can be represented by the frame structure in Figure 6b. There are two point forces, P_1 and P_2, two distributed loads, q_1 and q_0, as well as a concentrated force generated by the weight of the passengers and the gondola, Mg.



(a) Gondola (b) Frame structure

Figure 6: Gondola and the frame structure with degrees of freedom for each node.

Task 3 a)

Element stiffness matrix, $\mathbf{K_e}$, and element load vector, $\mathbf{f_e}$, for the two dimensional beam element was computed using CALFEM `beam2e` function and assembled in the global stiffness matrix and global load vector using CALFEM `assem` function. Displacement was then determined using `solveeq` given boundary conditions (fixed points). Maximum displacement was obtained by taking the maximum absolute value of `a`:

```
max_disp = max(abs(a));
```

Maximum displacement = 0.2606 m.

Task 3 b)

To plot element displacement over the frame structure CALFEM `eldisp2` was used.

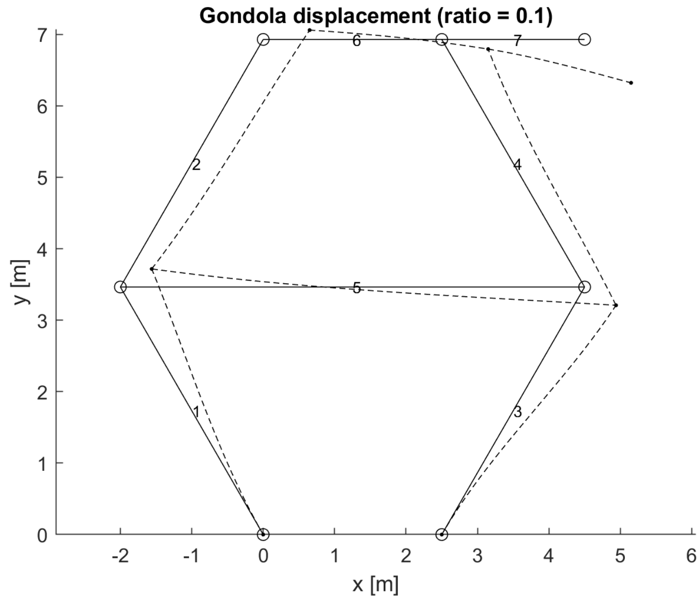


Figure 7: Displacement of the gondola with scale ratio 0.1.

Task 3 c)

In order to determine sectional forces, normal force, shear force and bending moment, function `beam2s` was used for each evaluation point on each beam element, given element node coordinates, properties, displacement and number of evaluation points. Sectional forces for the whole structure was plotted using `eldia2`. Figure 8 - 10 illustrates normal force, shear force and bending moment for the frame structure. Scaling factors are 5e-5, 3e-5 and 2e-5 respectively.

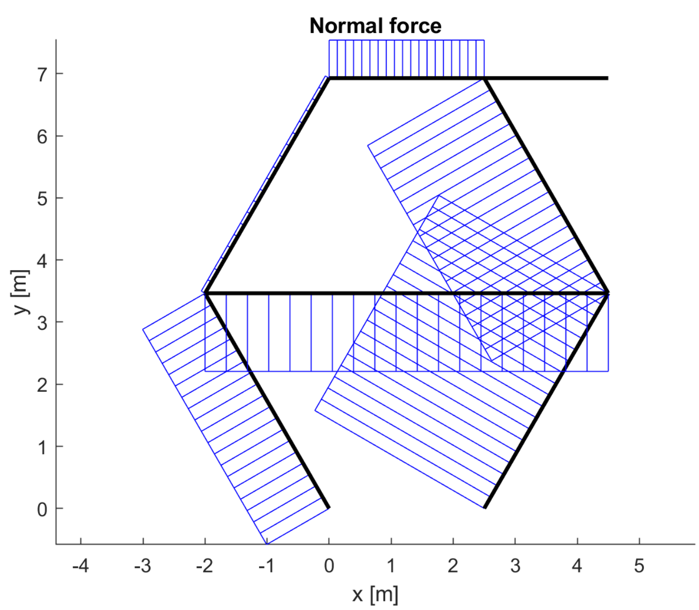


Figure 8: Normal Force, scaling factor $= 5 \times 10^{-5}$.

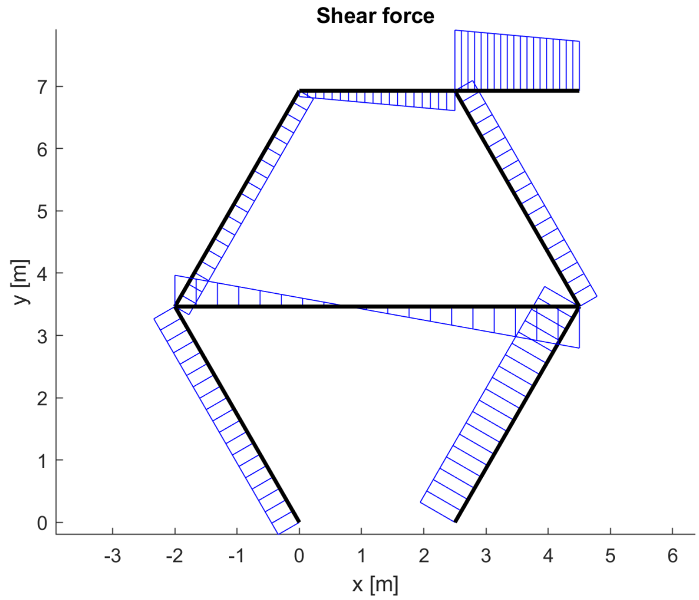


Figure 9: Shear Force, scaling factor = 3×10^{-5}.

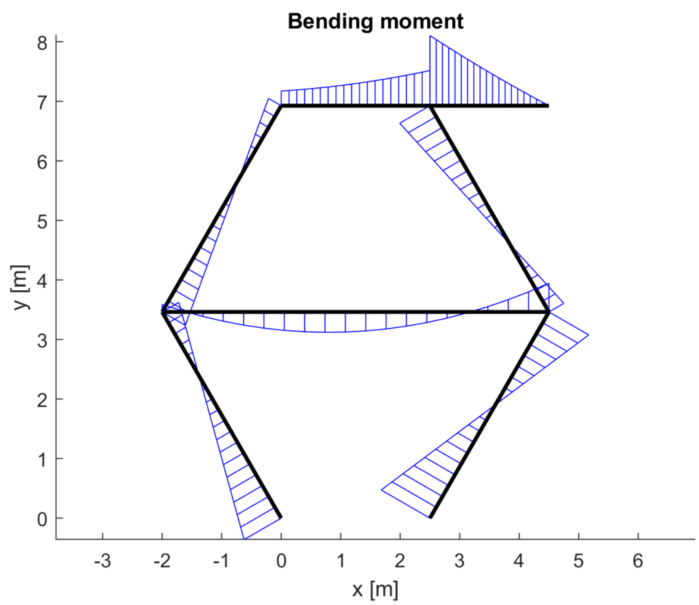


Figure 10: Bending moment, scaling factor = 2×10^{-5}.

Task 3 d)

Maximum bending moment was determined by taking the maximum absolute value of the calculated bending moments, see Appendix.

Maximum bending moment = 5.8920e+04 Nm.

Task 3 e)

Bending stress was calculated for each evaluation point on every beam element using Equation 25, ignoring contribution from normal force. Maximum bending stress was found at both top and bottom of the cross-section (`sig_upper_max` = `sig_lower_max`, see Appendix). Inertia, denoted as I, varies depending on the beam type, whether it's HEA100 or HEA120.

$$\sigma = \frac{Mz}{I} \tag{25}$$

Maximum bending stress = 6.5040e+08 Pa

To determine factor of safety, n, Equation 26 was used. Maximum bending moment occurs at element number 3, see frame structure in Figure 6b.

$$n = \frac{\sigma_y}{\sigma_{max}} \qquad \text{where } \sigma_y = 250 \text{ MPa} \tag{26}$$

Factor of safety, $n = 0.3844$.

Appendix 1

Listing 1: MATLAB code for Task 1

```
%% MHA021 FEM - Assigment 2
close all;
clear all;
clc;

%% Task 1

load('Mesh_data.mat')

%indata
h1=0.35;
h2=0.15;
h3=0.018;
h4=0.2;

% Temperatures
T_in=25;
T_out=-10;
T=[T_out T_in]

alpha=10;

% Lengths [m]
L1=1.2;
L2=0.6;
H=1.5;

% Heat transfer coefficient [W/m^2/K]
k1=1.5; %concrete
k2=0.035; %insulation
k3=0.17; %plaster
k=[k1 k2 k3]

thickness=1;

K=zeros(NoDofs);

% Assemble K elementwise
for element = 1:NoElem
    D=k(matrlIndex(element)).*eye(2);

    Ke = flw2te(Ex(element,:),Ey(element,:),thickness,D);
    K = assem(Edof(element,:),K,Ke);
end

%Assemble K with Kc and fb

NoBoundary=length(boundaryEdof)
Kc=zeros(NoDofs);
fb=zeros(NoDofs,1);

for element= 1:NoBoundary
    ex=boundaryEx(element,:);
```

```
    ey=boundaryEy(element,:);
    Tamb=T(boundaryMaterial( element, 2 ));
    [Kce, fce] = convecte(ex, ey, alpha, thickness, Tamb);
    [Kc, fb]=assem(boundaryEdof(element,:),Kc,Kce,fb,fce);
end

a=solveq(K+Kc,fb)

% plot
figure(1)
ed=extract(Edof,a);
fill(Ex',Ey',ed')
colormap parula
colorbar

axis equal
axis off

%Linear transmittance
Q_out=0
outside_boundary_index=find(boundaryMaterial(:,2)==1)
for i=1:length(outside_boundary_index)
    ex=boundaryEx(i,:);
    ey=boundaryEy(i,:);
    i_index=boundaryEdof(i,2);
    j_index=boundaryEdof(i,3);
    T_i=a(i_index);
    T_j=a(j_index);
    L_ex=ex(2)-ex(1);
    L_ey=ey(2)-ey(1);
    L_e=sqrt(L_ex^2+L_ey^2);
    Tamb=T(boundaryMaterial(i, 2 ));
    Q_e=alpha*L_e*thickness*((T_i+T_j)/2-Tamb);
    Q_out=Q_out+Q_e;
end
Q_out=2*Q_out;
%analytic solution
R_conv=1/(alpha);
R_1=h1/(k1);
R_2=h2/(k2);
R_3=h3/(k3);
R_ekv=2*R_conv+R_1+R_2+R_3;
q_analytic=(T_in-T_out)/R_ekv

psi=(abs(Q_out)-q_analytic*H)/(T_in-T_out)
```

Appendix 2

Listing 2: MATLAB code for Task 3

```
%% Task 3

% Lengths [m]
L1 = 4;
L2 = 2.5;
L3 = 2;

% Angle
alpha = 30;      % [deg]

% Weight of gondola plus passengers
m = 3000;        % [kg]
g = 9.82;        % [m/s^2]

% Point force [N]
P1 = 4000;
P2 = 6000;
P_m = m*g;

% Load [Nm^-1]
q_0 = 6000;
q_1 = 3000;

% Young's modulus
E = 210e9;       % [Pa]

% Beam areas [HEA100 HEA120] [m^2]
A = [21.24*10^-4 25.34*10^-4];

% Beam moment of inertia [HEA100 HEA120] [kg*m^2]
I = [349.2*10^-8 606.2*10^-8];

% Number of beams
num_HEA100_beams = 4;
num_HEA120_beams = 3;

% Matrices containing beam properties
HEA100_prop = repmat([E A(1) I(1)], num_HEA100_beams, 1);
HEA120_prop = repmat([E A(2) I(2)], num_HEA120_beams, 1);

% Ep = [E A I]
Ep = zeros(7,3);

% Beam 1,2,3 and 4 are HEA100 beams
Ep(1:4, :) = HEA100_prop;
% Beam 5,6 and 7 are HEA120 beams
Ep(5:end,:) = HEA120_prop;

% Load vector q_e rows = number of beams
q_e = zeros(7,2);
% Beam 5,6 and 7 experience loads
q_e(5,:) = [0,-q_0];
q_e(6:7,:) = [0,-q_1; 0,-q_1];
```

```matlab


% Elements dofs
Edof = [1 1:6
        2 4:9
        3 10:15
        4 13:18
        5 4 5 6 13 14 15
        6 7 8 9 16 17 18
        7 16:21];

Dof = [1:3
       4:6
       7:9
       10:12
       13:15
       16:18
       19:21];

% Coordinates
Coords = [0 0; -L1*sind(alpha) L1*cosd(alpha);0 2*(L1*cosd(alpha));L2 0;...
    L2+L1*sind(alpha) L1*cosd(alpha);L2 2*(L1*cosd(alpha));L2+L3 2*(L1*cosd(alpha))];

% Element coordinates Ex, Ey
[Ex, Ey] = coordxtr(Edof, Coords, Dof, 2);

% Illustrate the frame construction
figure(1)
eldraw2(Ex, Ey, [1 1 1], Edof(:,1))
hold on

% Initialize load vector and stiffness matrix
f = zeros(21,1);
K = zeros(21);

% Compute element stiffness matrix and element load vector for a two
% dimensional beam element by using CALFEM function "beam2e"
for i=1:length(Edof)
    % Loop over all elements and construct element stiffness matrix and element load vector
    [Ke,fe] = beam2e(Ex(i,:), Ey(i,:), Ep(i,:), q_e(i,:));
    % Assembly in global stiffness matrix and global load vector
    [K,f] = assem(Edof(i,:), K, Ke, f, fe);
end

% Add point forces to load vector
f(4) = P2;
f(7) = P1;
f(20) = -P_m;

% Boundary conditions
bc = [1 0; 2 0; 3 0; 10 0; 11 0; 12 0];

% Solve for displacement, a
```

```
a = solveq(K, f, bc);

% Maximum displacement
max_disp = max(abs(a));

% Extract all degrees of freedom for each element
Ed = extract_dofs(Edof, a);

sfac = scalfact2(Ex, Ey, Ed, 0.1);

plotpar = [2 1 0];

% Plot displacement
figure(1)
hold on
eldisp2(Ex, Ey, Ed, plotpar, sfac)
xlabel('x [m]')
ylabel('y [m]')
title('Gondola displacement (ratio = 0.1)')

n = 20; % number of evaluation points along the beam

% Compute sectional forces along the element [N V M]
% column 1 is normal forces, column 2 is th shear force and column 3
    is
% the bending moment
for i=1:length(Edof)
    SectionalForces(i).es = beam2s(Ex(i,:), Ey(i,:), Ep(i,:), Ed(i,:),
        q_e(i,:), n);
end

% Scaling factors
sfacs = [5e-5, 3e-5, 2e-5];

% Titles for the three plots
titles = {'Normal force', 'Shear force', 'Bending moment'};

% plot sectional forces
for j=1:3
    figure(j+1)
    for i=1:length(Edof)
        eldia2(Ex(i, :), Ey(i, :), SectionalForces(i).es(:,j), [2 1],
            sfacs(j));
        hold on
    end
    xlabel('x [m]')
    ylabel('y [m]')
    title(titles{j})
end

% Maximum bending moment
max_M_beam = zeros(length(Edof),1);
% Loop over structure, max_M_beam = max bending moment for every beam
for i=1:length(Edof)
    max_M_beam(i) = max(abs(SectionalForces(i).es(:,3)));
end
```

```
% Max bending moment in structure
max_M = max(abs(max_M_beam));

% z = h/2 for HEA100 and HEA120 beam [m^3]
z = [4.8e-2 5.7e-2];

% Initialize stress matrices, rows = number of beams, columns = number of
% evalution points
sig_upper = zeros(length(Edof), n);
sig_lower = zeros(length(Edof), n);

% Start with HEA100 beam
k = 1;

% Calculate stress on every beam
for i = 1:length(SectionalForces)
    % At every evaluation point on the beam
    % If beam number > 4 => HEA120 beam
    if i > 4
        k = 2;
    end
    for j = 1:n
        % Naviers formula, excluding normal force, sigma = Mz/I
        sig_upper(i, j) = SectionalForces(i).es(j, 3) / I(k) * z(k);
        sig_lower(i, j) = -SectionalForces(i).es(j, 3) / I(k) * z(k);
    end
end

% Sigma_max and element number
[sig_upper_max, pos_upper] = max(abs(sig_upper(:)));
[sig_lower_max, pos_lower] = max(abs(sig_lower(:)));

% Where does the max bending stress occur, upper or lower
if sig_upper_max > sig_lower_max
    sig_max = sig_upper_max;
else
    sig_max = sig_lower_max;
end

% Yield limit of the material [Pa]
sig_yield = 250*10^6;

% Determine factor of safety sigma_yield/sigma_max [-]
factor_of_safety = sig_yield/sig_max;
```