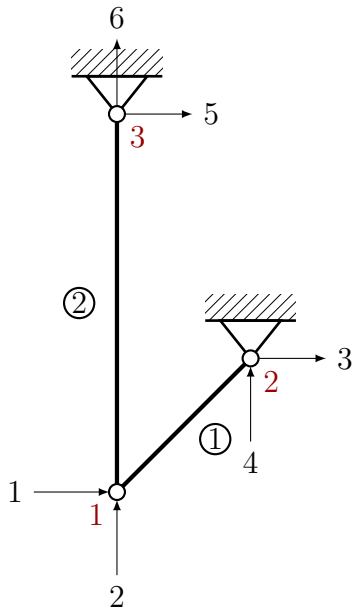
# Problem set 5 - solutions

## Problem 1

a) Degree of freedom classification:




| DOF index | Description |
|---|---|
| 1 | $q_{1x}$ |
| 2 | $q_{1y}$ |
| 3 | $q_{2x}$ |
| 4 | $q_{2y}$ |
| 5 | $q_{3x}$ |
| 6 | $q_{3y}$ |

Each node contributes two degrees of freedom (horizontal and vertical displacements). Thus, the system has six DOFs. Nodes 2 and 3 are fully fixed, thus $q_{2x} = q_{2y} = q_{3x} = q_{3y} = 0$, the only free DOFs are 1 and 2, the horizontal and vertical displacements of node 1.

b) Reduced system formulation: the full stiffness and loads vectors are respectively a $6 \times 6$ matrix and a $6 \times 1$ vector. However, since the unconstrained DOFs are indexed by 1 and 2, these are the only lines and columns we shall compute and include in the reduced stiffness matrix and applied loads vector. Bars parameters are:

| Elements | Orientations (${}^e\theta$) | Lengths (${}^e\ell$) |
|---|---|---|
| 1 | 45° | 5 m |
| 2 | 90° | 10 m |

Therefore, after applying boundary conditions, the two reduced element stiffness matrices in global

coordinates are:

$$
{}^1\mathbf{K} = \frac{EA}{{}^1\ell}\begin{bmatrix} \cos^2({}^1\theta) & \sin({}^1\theta)\cos({}^1\theta) \\ \sin({}^1\theta)\cos({}^1\theta) & \sin^2({}^1\theta) \end{bmatrix} = \frac{EA}{10}\begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}
$$

$$
{}^2\mathbf{K} = \frac{EA}{{}^2\ell}\begin{bmatrix} \cos^2({}^2\theta) & \sin({}^2\theta)\cos({}^2\theta) \\ \sin({}^2\theta)\cos({}^2\theta) & \sin^2({}^2\theta) \end{bmatrix} = \frac{EA}{10}\begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix}
$$

As we assemble we obtain the reduced stiffness matrix:

$$
\mathbf{K} = {}^1\mathbf{K} + {}^2\mathbf{K} = \frac{EA}{10}\begin{bmatrix} 1 & 1 \\ 1 & 2 \end{bmatrix}.
$$

The reduced applied loads vector $\mathbf{f}$ can be computed directly without having to estimate the two elementary loads vectors ${}^1\mathbf{f}$ and ${}^2\mathbf{f}$. We have

$$
\mathbf{f} = \begin{bmatrix} -25000 \\ 0 \end{bmatrix}.
$$

c) Displacement calculation: solving the reduced linear system

$$
\frac{EA}{10}\begin{bmatrix} 1 & 1 \\ 1 & 2 \end{bmatrix}\begin{bmatrix} q_{1x} \\ q_{1y} \end{bmatrix} = \begin{bmatrix} -25000 \\ 0 \end{bmatrix}
$$

yields to (static) displacements for node 1:

$$
q_{1x} = -\frac{500'000}{EA}, \quad \text{and} \quad q_{1y} = \frac{250'000}{EA}.
$$

The displacements indicate that node 1 experiences negative horizontal and positive vertical displacements; this behavior is consistent with the direction of the applied load and the geometry of the structure.

d) Stress evaluation: the approximated axial stress in each element is computed using the formula:

$$
{}^e\sigma^h = \frac{E}{{}^e\ell}\begin{bmatrix} -\cos({}^e\theta) & -\sin({}^e\theta) & \cos({}^e\theta) & \sin({}^e\theta) \end{bmatrix}\begin{bmatrix} q_{ix} \\ q_{iy} \\ q_{jx} \\ q_{jy} \end{bmatrix}
$$

where we suppose that, in general, element $e$ is connected to nodes numbered as $i$ and $j$. For element 1 ($\theta = 45°$, $\ell = 5$ m):

$$
{}^1\sigma^h = \frac{E}{5}\begin{bmatrix} -\frac{\sqrt{2}}{2} & -\frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} \end{bmatrix}\begin{bmatrix} -500'000/EA \\ 250'000/EA \\ 0 \\ 0 \end{bmatrix} = \frac{25'000\sqrt{2}}{A}
$$

For element 2 ($\theta = 90°$, $\ell = 10$ m):

$$
{}^2\sigma^h = \frac{E}{10}\begin{bmatrix} 0 & -1 & 0 & 1 \end{bmatrix}\begin{bmatrix} -500'000/EA \\ 250'000/EA \\ 0 \\ 0 \end{bmatrix} = -\frac{25'000}{A}
$$

The computed stresses indicate that element 1 experiences a tensile stress, while element 2 is under compressive stress. This is consistent with the overall structural behavior: the applied horizontal load at node 1 pulls on element 1, stretching it, while simultaneously pushing down against the vertical support, placing element 2 in compression. These stress signs reflect the internal force directions that balance the applied load and maintain equilibrium.

## Problem 2

The MATLAB code provided below solves the transient analysis of a two-dimensional steel truss using the finite element method. The approach follows standard steps in structural dynamics: material definition, finite element modeling, modal analysis, and time-domain response.

### Material properties

The code begins by defining the material properties of structural steel:

```
clear all

% E: modulus of elasticity
E = 21010^9;
% A: area of cross section
A = 6.4510-4;
% EA: axial stiffness
EA = E*A;
% rho: density
rho = 7800;
% rhoA: mass per unit length
rhoA = rho*A;
% Applied force
f = 22000;
```

### Geometry and mesh connectivity

The truss geometry is defined through a node coordinate matrix and a connectivity matrix. The nodes are arranged in a simple triangular configuration, and elements are formed between nodes via the connectivity table. A plotting function `draw2Dtruss` is used to visually confirm the mesh.

```
nodesCoordinates = 6*[0 0; 0 1; 1 0];
connectivity = [1 2; 2 3; 1 3];

numberOfNodes = size(nodesCoordinates,1);
GDof = 2*numberOfNodes;

draw2Dtruss(nodesCoordinates, connectivity, ...
'LineColor', 'r', ...
'LineWidth', 2, ...
'MarkerSize', 8, ...
'ShowNodeNumbers', true);
```

### Global stiffness and consistent mass matrices

The next step involves the computation of the global stiffness matrix and global consistent mass matrix using helper functions `formStiffness2Dtruss` and `formConsistentMass2Dtruss`. These matrices represent the elastic and inertial properties of the system, respectively.

```
% compute and assemble the structure stiffness matrix
stiffness = formStiffness2Dtruss(GDof, connectivity, nodesCoordinates, EA);

% compute and assemble the structure mass matrix
mass = formConsistentMass2Dtruss(GDof, connectivity, nodesCoordinates, rhoA);
```

### Applying boundary conditions and solving the eigenproblem

Boundary conditions are applied by specifying the constrained degrees of freedom. Node 1 is fully fixed, and node 4 is constrained in the $y$-direction, leaving three free DOFs.

```
prescribedDof = transpose([1 2 6]);
activeDof = setdiff(transpose((1:GDof)), prescribedDof);
stiffness_freeDofs = stiffness(activeDof,activeDof);
mass_freeDofs = mass(activeDof,activeDof);

[modal_matrix,frequencies] = computeFrequenciesAndModes(GDof,prescribedDof,stiffness,
                                mass,0);
```

Notice that normal modes are only relative values which may be scaled of normalized to some extent as a matter of choice. Use the following normalization convention is in force:

$$n_i = \frac{\phi_i}{\sqrt{\phi_i^T \mathbf{M} \phi_i}}$$

The vectors $n_i$ are referred as normalized modal vectors.

### Mode normalization and orthogonality checks

The eigenvectors computed by the `eig` function in MATLAB are not guaranteed to follow a particular normalization convention. In the context of structural dynamics, however, it is often convenient to normalize the mode shapes with respect to the mass matrix such that:

$$\mathbf{\Phi}^T \mathbf{M} \mathbf{\Phi} = \mathbf{I},$$

This normalization ensures that the modal matrix $\mathbf{\Phi}$, which collects the mode shapes as columns, satisfies:

$$\mathbf{\Phi}^T \mathbf{K} \mathbf{\Phi} = \mathbf{\Lambda},$$

where $\mathbf{\Lambda}$ is the diagonal matrix of eigenvalues $\lambda_i = \omega_i^2$. The following code performs this mass-normalization of the mode shapes and verifies the orthogonality relations with respect to both the mass and stiffness matrices. These checks confirm that the system of modal coordinates is decoupled and can be treated as independent single-degree-of-freedom systems during transient analysis.

```
modes_normalized = zeros(size(modal_matrix));
for modeNumber = 1:size(modal_matrix,2)
norm = sqrt(transpose(modal_matrix(:,modeNumber)) * mass_freeDofs *
        modal_matrix(:,modeNumber));
modes_normalized(:,modeNumber) = 1 / norm * modal_matrix(:,modeNumber);
end

% Check orthogonality with respect to stiffness matrix
transpose(modal_matrix) * stiffness_freeDofs * modal_matrix
frequencies.^2

% Check orthogonality with respect to mass matrix
transpose(modal_matrix) * mass_freeDofs * modal_matrix
```

**Transient analysis**

The transient response of the structure is evaluated in modal coordinates using the modal superposition method. The transformation $\mathbf{q}(t) = \mathbf{\Phi}\,\mathbf{z}(t)$ yields the modal force vector, which drives each of the uncoupled second-order ordinary differential equations (ODEs) of the form:

$$\mathbf{I}\ddot{\mathbf{z}}(t) + \mathbf{\Lambda}\mathbf{z}(t) = \mathbf{p}$$

where $\mathbf{\Lambda} = \mathrm{diag}(\lambda_1^2, \lambda_2^2, \lambda_3^2)$ and the modal force is

$$\mathbf{p} = \mathbf{\Phi}^T\mathbf{f}.$$

The modal equations are given in components by

$$\ddot{z}_i(t) + \omega_i^2 z_i(t) = p_i.$$

Hence the uncoupled system of equations is

$$\begin{cases} \ddot{z}_1(t) + 506400\, z_1(t) = p_1 \\ \ddot{z}_2(t) + 224360\, z_2(t) = p_2 \\ \ddot{z}_3(t) + 324770\, z_3(t) = p_3 \end{cases}$$

where

```
p = transpose(modal_matrix) * [f; 0; 0];
```

The solution of the above equations for zero initial conditions $\mathbf{z}(0) = \dot{\mathbf{z}}(0) = 0$ is given by

$$\begin{cases} z_1(t) = \frac{p_1}{\omega_1^2}(1 - \cos\left(\omega_1^2 t\right)) \\ z_2(t) = \frac{p_2}{\omega_3^2}(1 - \cos\left(\omega_2^2 t\right)) \\ z_3(t) = \frac{p_3}{\omega_2^2}(1 - \cos\left(\omega_3^2 t\right)) \end{cases}$$

```
syms t
z = vpa(p ./ frequencies.^2 .* (1 - cos(frequencies.^2 * t)),3);
q = vpa(modal_matrix * z, 3)
```

The time response at the nodal coordinates is calculated from $\mathbf{q}(t) = \Phi\,\mathbf{z}(t)$.

### Plotting the time responses

We now compute and plot the time responses $q_1(t)$, $q_2(t)$, and $q_3(t)$ over a given time interval.

```
% Define time vector
t_vals = linspace(0, 0.1, 1000);
q_time = zeros(3, length(t_vals));

for i = 1:length(t_vals)
z_t = p ./ frequencies.^2 .* (1 - cos(frequencies .* t_vals(i)));
q_time(:, i) = modal_matrix * z_t;
end

% Plot q_1(t)
figure;
plot(t_vals, q_time(1, :), 'LineWidth', 2);
xlabel('Time (s)'); ylabel('Displacement q_1(t) [m]');
title('Time Response of q_1(t)'); grid on;

% Plot q_2(t)
figure;
plot(t_vals, q_time(2, :), 'LineWidth', 2);
xlabel('Time (s)'); ylabel('Displacement q_2(t) [m]');
title('Time Response of q_2(t)'); grid on;

% Plot q_3(t)
figure;
plot(t_vals, q_time(3, :), 'LineWidth', 2);
xlabel('Time (s)'); ylabel('Displacement q_3(t) [m]');
title('Time Response of q_3(t)'); grid on;
```