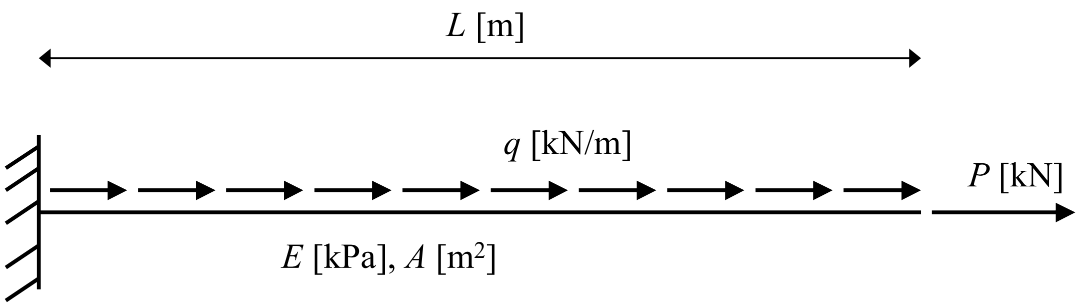
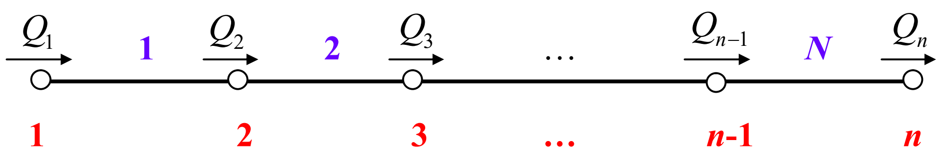
Project 1a: Statics of a bar structure – linear FE interpolation

Given is the bar with length L, clamped on the left end, subjected to the uniform load with intensity q as well as concentrated force P on the right end. The linear elastic material is assumed (with Young modulus E), with a rectangular cross-section (dimensions b and d).

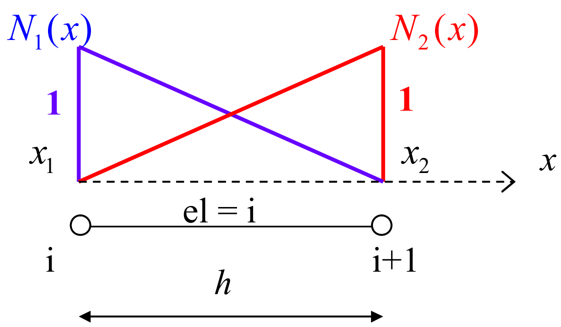


Apply FEM for determination of displacements Q as well as normal force S.

FEM discretization: n nodes, N elements, $N_{ss} = n$ degrees of freedom (horizontal displacements at nodes).



FEM approximation: two linear shape functions, $N_1(x)$, $N_2(x)$, with features $N_1(x_1) = 1$, $N_1(x_2) = 0$ and $N_2(x_1) = 0$, $N_2(x_2) = 1$



Formula for element stiffness matrix

$$\underset{[2\times2]}{\boldsymbol{K}^e} = E \cdot A \cdot \int_{x_1}^{x_2} \left(\boldsymbol{B}^e\right)^T \boldsymbol{B}^e dx \tag{1}$$

where $\boldsymbol{B}^e$ - matrix of element shape functions derivatives

$$\underset{[1\times2]}{\boldsymbol{B}^e} = \begin{bmatrix} \dfrac{dN_1(x)}{dx} & \dfrac{dN_2(x)}{dx} \end{bmatrix} \tag{2}$$

Formula for element load vector

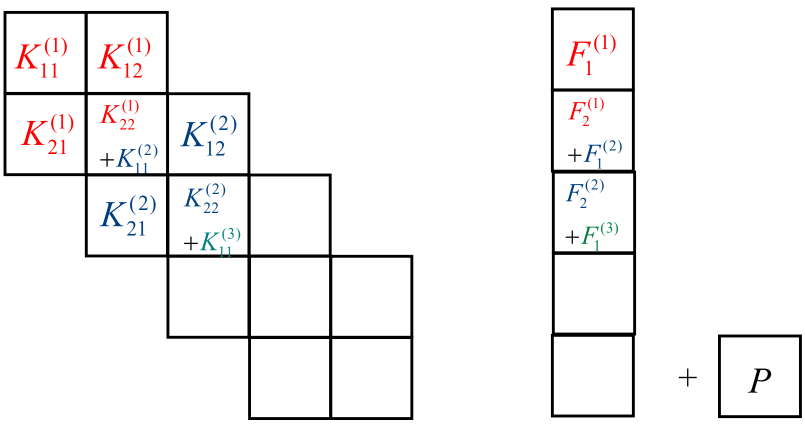
$$\underset{[2\times1]}{\boldsymbol{F}^e} = \int_{x_1}^{x_2} \left(\boldsymbol{N}^e\right)^T q \cdot dx \tag{3}$$

where $\boldsymbol{N}^e$ - matrix of element shape functions

$$\underset{[1\times2]}{\boldsymbol{N}^e} = \begin{bmatrix} N_1(x) & N_2(x) \end{bmatrix} \tag{4}$$

and q – uniform load intensity (constant or variable) along the element.

Assembly scheme of a global stiffness matrix $\underset{[N_{ss}\times N_{ss}]}{\boldsymbol{K}}$ and global load vector $\underset{[N_{ss}\times1]}{\boldsymbol{F}}$ (with P taken into account)



Fulfilment of a boundary condition $Q_1 = 0$

Solution of system of equations (determination of nodal displacements)

$$\underset{[N_{ss}\times1]}{\boldsymbol{Q}} \quad \rightarrow \quad \boldsymbol{K} \cdot \boldsymbol{Q} = \boldsymbol{F} \tag{5}$$

Determination of nodal normal forces S for each element

$$\underset{[2\times1]}{\boldsymbol{S}^e} = \boldsymbol{K}^e \cdot \boldsymbol{Q}^e - \boldsymbol{F}^e \tag{6}$$

Determination of a displacement and normal force for arbitrary point of element

$$u(x) = N_1(x) \cdot Q_1^e + N_2(x) \cdot Q_2^e \qquad (7)$$

$$s(x) = E \cdot A \cdot \left(\frac{dN_1(x)}{dx} \cdot Q_1^e + \frac{dN_2(x)}{dx} \cdot Q_2^e \right) \qquad (8)$$

Assignments

1. Assume data: L = 1 m, E = 207 GPa, b = d = 5 cm, q = $10e2 \cdot x$ kN/m (linearly dependent tensile load), P = - 1e2 kN (normal compression force).
2. Apply manual calculation assuming two finite elements.
3. Apply computer calculations using Matlab code for arbitrary number of elements, examine the correctness of manual calculations (for N = 2) by means of this code. Display results for N = 10.

Hints

In Matlab, use loop *for* as well as the following functions:

- for symbolic operations: *syms*, *diff*, *int, subs, eval*
- for matrix calculations: *zeros*
- for graphics: *figure*, *plot*, *title*, *axis*, *xlabel*, *ylabel, legend*, *hold*
- for FE computations (using Calfem package): *assem*, *solveq*.

Syntax of *assem*: [K,F]=assem(edof,K,Ke,F,Fe)
Syntax of *solveq*: [Q,R]=solveq(K,F,bc)

Code scheme:

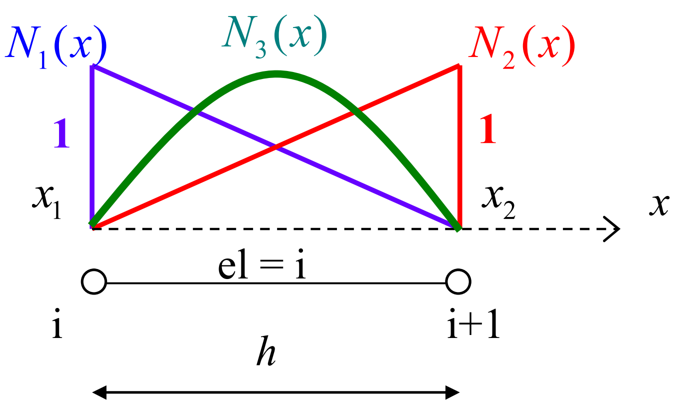
- open a new file, clear memory (*clear all*), the Command Window (*clc*) window and, close the graphics windows (*close all*),
- define the symbolic variable x,
- define data for the task (E, A, L, q, P),
- specify the number of elements (N)
- calculate: number of nodes (n), number of degrees of freedom (Nss), length of element (h)
- generate node coordinates (X),
- reset the global stiffness matrix (K),
- reset the global load vector (F),
- enter the value of the concentrated force into the corresponding element of the vector F,
- run the loop through the elements (i),
  - o specify the coordinate of the first node of the i-th element (x1),
  - o specify the coordinate of the second node of the i-th element (x2),
  - o enter the formula for the first shape function (N1),
  - o enter the formula for the second shape function (N2),
  - o calculate the shape function vector (Ne) [1x2],
  - o calculate the matrix of derivatives of the shape function (Be) [1x2],
  - o calculate element stiffness matrix (Ke) [2x2],
  - o calculate element load vector (Fe) [2x1],
  - o specify a vector [1x3] containing: the item number and its d.o.f. numbers (edof) for assembling purposes,

- close the loop through the elements,
- specify the boundary vector (bc) [1x2] - the blocked d.o.f. number and the displacement value,
- solve the system of FE equations (determine displacements and reactions),
- open the first graphics window,
- draw displacements,
- insert the title of the chart, place the legend and descriptions of the x and y axes,
- test the operation of the program for different numbers of N elements
- open the second graphics window, enable the hold function
- define the number of intermediate points in the element M = 10
- run the loop through the elements,
    - repeat the process of calculating the element's stiffness matrix (from the previous loop),
    - select the nodal displacements of the i-th element (Qe),
    - calculation of nodal forces (Se)
    - calculation of intermediate points in element (xe),
    - calculation of the normal force at intermediate points
    - draw a normal force in element,
- close the loop,
- insert the title of the graph and the descriptions of the x and y axes,
- test the operation of the program

Project 1b: Statics of a bar structure – parabolic FE interpolation

Matlab code, in its present code, working for statics, extend towards a parabolic interpolation in finite element, by means of hierarchical shape functions $N_1(x)$, $N_2(x)$ and $N_3(x)$, where $N_1(x)$ i $N_2(x)$ - standard linear shape functions from the previous variant, and $N_3(x)$ - bubble quadratic function:

$$N_3(x) = -(x - x_1)(x - x_2) \quad (9)$$

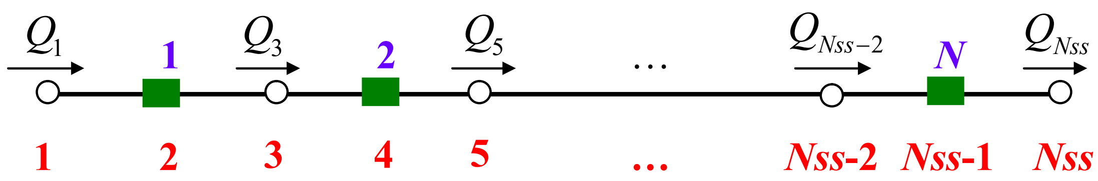


Element interpolation has the form (α_3^e - additional degree of freedom):

$$u(x) = N_1(x) \cdot Q_1^e + N_2(x) \cdot Q_2^e + N_3(x) \cdot \alpha_3^e \quad (10)$$

$$s(x) = E \cdot A \cdot \left(\frac{dN_1(x)}{dx} \cdot Q_1^e + \frac{dN_2(x)}{dx} \cdot Q_2^e + \frac{dN_3(x)}{dx} \cdot \alpha_3^e \right) \tag{11}$$

Modifications in FE algorithm should affect larger dimensions of element matrices ($[3\times3]$) and element vectors ($[1\times3]$ or $[3\times1]$). However on the level of the entire system, it is required to assume additional mathematical (no physical meaning) degree of freedom α, assigned to each element.



Hints
Make a copy of a current file. Save a copy under a proper name and provide required modifications. Examine the code for specified number of elements, compare results of project 1a and 1b.

Project 1c: Dynamics of a bar structure – self vibrations

The bar from the previous task is subjected to longitudinal free vibration (no external load). Find the first four natural frequencies and the corresponding vibrations. The calculations should be carried out for two variants of mass distribution of the bar - concentrated mass and continuous mass distribution.

In the FEM analysis, a consistent (full) mass matrix is used, calculated as (for an element)

$$\underset{[2\times2]}{\boldsymbol{M}^e} = \rho \cdot A \cdot \int_{x_1}^{x_2} \left(\boldsymbol{N}^e\right)^T \boldsymbol{N}^e dx \tag{12}$$

where ρ denotes the mass density. The mass matrices of individual elements are aggregated similarly to stiffness matrices.
In addition to the consistent matrix (i.e., having off-diagonal elements), diagonal matrices of masses can be used, resulting from point distribution of mass (similar to the problems of structural mechanics)

$$\underset{[2\times2]}{\boldsymbol{M}^{pun}} = \rho \cdot A \cdot \frac{h^e}{2} \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \tag{13}$$

Aggregation of the stiffness and mass matrices leads to a generalized eigen problem at the level of the entire system (bar)

$$\left(\boldsymbol{K} - \omega^2 \boldsymbol{M}\right)\boldsymbol{Q} = 0 \tag{14}$$

where ω - angular frequency of free vibrations. Using angular frequency, one may determine the ordinary frequency f as well as the period T

$$\omega = 2\pi f = \frac{2\pi}{T} \tag{15}$$

Assignments

For the bar from task 1a (linear interpolation), assume the mass density $\rho = 7500 \frac{kg}{m^3}$. To the existing program, add procedures that allow to determine the first frequencies and forms of natural vibration, based on matrices of continuous and concentrated mass distributions. Draw vibration forms, based upon the results of FEM.

Hints

Calculations for the mass matrix can be organized in the same loop as for the stiffness matrix - appropriate global matrices should be prepared (zeroed) before the loop

```
Mcons = zeros(
Mdiag = zeros(
```

and then, within the loop, build the appropriate element matrices based on the formulas (12), (13) and aggregate them to the global matrix (*assem* function).

In the further part of the program, in addition to the previously mentioned functions, use *eigen* function (from the Calfem toolbox), solving a generalized eigen problem. Syntax of the function: **[L, Q] = eigen (K, M, bc)**, where L - eigenvalues, Q - eigenvectors (in columns), K - global stiffness matrix, M - global mass matrix, bc - vector containing blocked numbers degrees of freedom (i.e., the first column of matrix bc for static calculations).

For full analysis (with consistent mass matrix):

```
[L1,Q1] = eigen(
Freq1 =
```

and for simplified analysis (with diagonal mass matrix)

```
[L2,Q2] = eigen(
Freq2 =
```

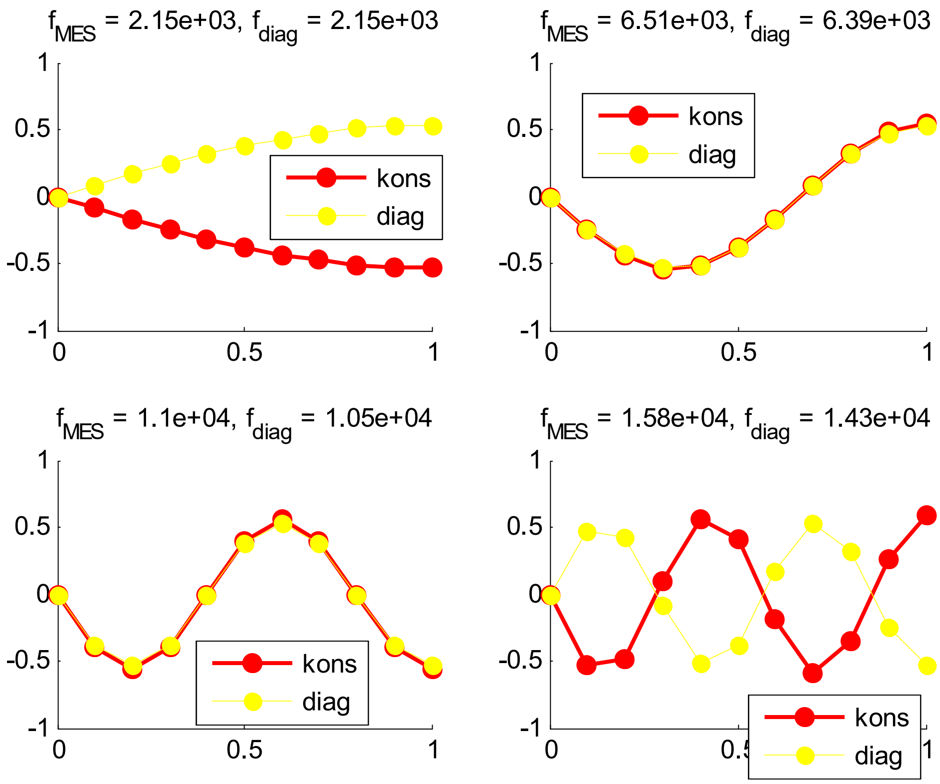
Variables Freq1 i Freq2 denote the frequencies of the free vibrations, which should be determined from (15).

Scheme of part of a code, drawing vibration forms

```
figure(
form_number = min([4 N]);
for i = % loop over form number
    subplot(2,2,i);
```

```
    title(['f_F_E_M = ' num2str(Freq1(i),3) ...
        ', f_d_i_a_g = ' num2str(Freq2(i),3)]);
    hold on
    plot( % Graph of a vibration form Q1 for Mcons matrix
    plot( % Graph of a vibration form Q2 for Mdiag matrix
    legend('cons','diag');
end
```

Exemplary solution – comparison of the first four vibration forms



Project 1d: Dynamics of a bar structure – forced vibrations

Assume the time-dependent load and use the Newmark method to determine axial displacements at arbitrary time. Draw a history of displacement at the selected point $x = L$. Make calculations for the approximation variant with linear shape functions.

FE equations of forced vibrations (without damping)

$$\boldsymbol{M}\ddot{\boldsymbol{u}} + \boldsymbol{K}\boldsymbol{u} = \boldsymbol{f}(t) \tag{16}$$

where

- $\boldsymbol{u} = \boldsymbol{u}(x,t)$ - displacements,
- $\boldsymbol{f} = \boldsymbol{f}(t)$ - load,
- $\boldsymbol{M}$ - mass matrix,
- $\boldsymbol{K}$ - stiffness matrix.

Newmark method:

- given initial data: $\boldsymbol{u}_0$, $\boldsymbol{v}_0 = \dot{\boldsymbol{u}}_0$, $\boldsymbol{a}_0 = \ddot{\boldsymbol{u}}_0 = \boldsymbol{M}^{-1}\left(\boldsymbol{f}_0 - \boldsymbol{K}\boldsymbol{u}_0\right)$
- acceleration:

$$\boldsymbol{a}_{k+1} = \left(\boldsymbol{M} + \frac{1}{2}\Delta t^2 \boldsymbol{K}\right)^{-1} \left(\boldsymbol{f}_{k+1} - \boldsymbol{K}\cdot\left(\boldsymbol{u}_k + \Delta t \cdot \boldsymbol{v}_k\right)\right) \tag{17}$$

- velocity:

$$\boldsymbol{v}_{k+1} = \boldsymbol{v}_k + \Delta t \cdot \boldsymbol{a}_{k+1} \tag{18}$$

- displacement:

$$\boldsymbol{u}_{k+1} = \boldsymbol{u}_k + \Delta t \cdot \boldsymbol{v}_k + \frac{1}{2}\Delta t^2 \boldsymbol{a}_{k+1} \tag{19}$$

Assume the following data:

- load frequency : 0.8 of a minimal frequency for natural vibrations, determined by FEM for the consistent mass matrix,
- angular load frequency ω according to (15),
- dynamic load according to $\boldsymbol{f}(t) = \boldsymbol{A}_f \sin\left(\omega \cdot t\right)$, where load amplitude $\boldsymbol{A}_f = \boldsymbol{F}$ - static load vector,
- initial time $t_0 = 0$,
- the final time $t_1 = 0.1$,
- number of time steps $N_t = 500$,
- length of a time step $\Delta t = \dfrac{t_1 - t_0}{N_t - 1}$

Attention! In the case of unstable results, reduce the final time or increase the number of time steps, respectively.

The scheme of the last part of the program:

```
%% FORCED VIBRATIONS - NEWMARK METHOD

% enforcement of boundary conditions in mass and stiffness matrices
Mcons(1, :) = []; % deletion of the first row from the mass matrix
Mcons( % removal of the first column from the mass matrix
K( % removal of the first row from the stiffness matrix
K( % removal of the first column from the stiffness matrix

% load definition
Load_freq = % load ordinary frequency
W_load = % load angular frequency
Load_amp = F( % elements of the static load vector from 2 to Nss
t_0 = % initial time (of the beginning of the process)
t_1 = % the final time of the process

% discretization of the time axis
Nt = % number of time steps
dt = % length of the time step
```

```
% Newmark method
Q = zeros(Nss-1,1); % initial displacement of nodes from 2 to Nss
Qv = zeros( % initial node velocities from 2 to Nss
Qa = zeros( % initial accelerations of nodes from 2 to Nss
Q_hist = zeros( % history of Nt displacements of nodes from 1 to Nss
q_hist = zeros( % history of Nt displacements of the node x = L
t_hist = zeros( % history of time changes

t_hist(1) = t_0;
q_hist(1) = Q(Nss-1);
Q_hist(1, :) = [0; Q]';
time = t_0;

for i = % loop over time steps from 2 to Nt
    time = % time in the current step
    Qa_pop = Qa;
    Qv_pop = Qv;
    Q_pop = Q;
    Qa = ;% new acceleration value
    Qv = ;% new speed value
    Q = ;% new displacement value

    t_hist i) = ;% saves the current time
    q_hist(i) = ;% save the displacement of the last node
    Q_hist(i, :) = [0; Q]'; % save displacements of all nodes
end

% animation of a vibrating bar
figure(4);
title('
xlabel('
ylabel('
for i = % loop over time steps from 1 to Nt
    plot( % node displacement graph at the next time of time (on the x-axis: X, on
the y-axis: i-th column of the matrix Q)
    axis([0 L min(min(Q_hist)) max(max(Q_hist))]); % fixed scale of the chart
    pause(0.001)
end

% displacement history graph of the last node
figure(5);
plot( % displacement history graph of the last node
hold on
grid on
title('
legend('
xlabel('
ylabel('
```