Chapter 3
Analysis of bars

3.1 A bar element

Consider the two-node bar finite element shown in figure 3.1, with constant cross-section (area A) and length $L = 2a$. The bar element can undergo only axial stresses σ_x, which are uniform in every cross-section.

The work stored as strain energy dU is obtained as

$$dU = \frac{1}{2}\sigma_x \epsilon_x A dx \tag{3.1}$$

The total strain energy is given by

$$U = \frac{1}{2}\int_{-a}^{a} \sigma_x \epsilon_x A dx \tag{3.2}$$

By assuming a linear elastic behaviour of the bar material, we can write

$$\sigma_x = E\epsilon_x \tag{3.3}$$

where E is the modulus of elasticity. Therefore the strain energy can be expressed as

$$U = \frac{1}{2}\int_{-a}^{a} EA\epsilon_x^2 dx \tag{3.4}$$

Strains ϵ_x can be related with the axial displacements u as

$$\epsilon_x = \frac{du}{dx} \tag{3.5}$$

By substituting (3.5) into (3.4) we then obtain

$$U = \frac{1}{2}\int_{-a}^{a} EA\left(\frac{du}{dx}\right)^2 dx \tag{3.6}$$

A.J.M. Ferreira, *MATLAB Codes for Finite Element Analysis: Solids and Structures,* Solid Mechanics and Its Applications 157,

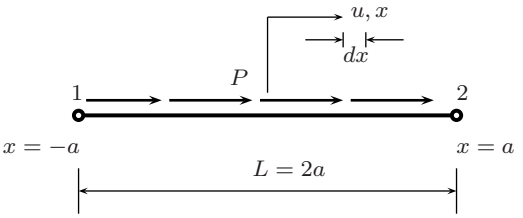


Fig. 3.1 A bar element in its local coordinate system

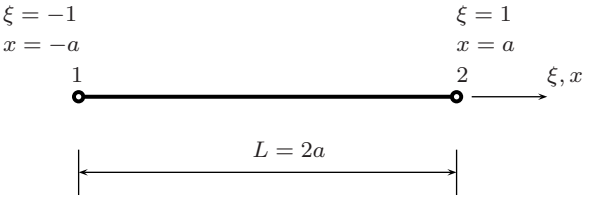


Fig. 3.2 A two-node bar element

If we consider p as the applied forces by unit length, the virtual external work at each element is

$$\delta W = \int_{-a}^{a} p\delta u dx \tag{3.7}$$

Let's consider now a two-noded finite element, as illustrated in figure 3.2. The axial displacements can be interpolated as

$$u = N_1(\xi)u_1 + N_2(\xi)u_2 \tag{3.8}$$

where the shape functions are defined as

$$N_1(\xi) = \frac{1}{2}(1-\xi); \quad N_2(\xi) = \frac{1}{2}(1+\xi) \tag{3.9}$$

in the natural coordinate system $\xi \in [-1, +1]$. The interpolation (3.8) can be defined in matrix form as

$$\mathbf{u} = \begin{bmatrix} N_1 & N_2 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} = \mathbf{N}\mathbf{u}^e \tag{3.10}$$

The element strain energy is now expressed as

$$U = \frac{1}{2}\int_{-a}^{a} EA\left(\frac{du}{dx}\right)^2 dx = \frac{1}{2}\int_{-1}^{1} \frac{EA}{a^2}\left(\frac{du}{d\xi}\right)^2 a d\xi = \frac{1}{2}\mathbf{u}^{eT}\int_{-1}^{1} \frac{EA}{a}\mathbf{N}'^T\mathbf{N}' d\xi \mathbf{u}^e \tag{3.11}$$

where $\mathbf{N}' = \dfrac{du}{d\xi}$, and

$$U = \frac{1}{2}\mathbf{u}^{eT}\mathbf{K}^e\mathbf{u}^{\mathbf{e}} \tag{3.12}$$

The element stiffness matrix, $\mathbf{K}^e$, is given by

$$\mathbf{K}^e = \frac{EA}{a}\int_{-1}^{1} \mathbf{N}'^T \mathbf{N}' d\xi \tag{3.13}$$

The integral is evaluated in the natural system, after a transformation of coordinates $x = a\xi$, including the evaluation of the jacobian determinant, $|J| = \frac{dx}{d\xi} = a$.

In this element the derivatives of the shape functions are

$$\frac{dN_1}{d\xi} = -\frac{1}{2}; \quad \frac{dN_2}{d\xi} = \frac{1}{2} \tag{3.14}$$

In this case, the stiffness matrix can be given in explicit form as

$$\mathbf{K}^e = \frac{EA}{a}\int_{-1}^{1} \begin{bmatrix} -\frac{1}{2} \\ \\ \frac{1}{2} \end{bmatrix} \begin{bmatrix} -\frac{1}{2} & \frac{1}{2} \end{bmatrix} d\xi = \frac{EA}{2a}\begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix} \tag{3.15}$$

By using $L = 2a$ we obtain the same stiffness matrix as in the direct method presented in the previous chapter.

The virtual work done by the external forces is defined as

$$\delta W^e = \int_{-a}^{a} p\delta u dx = \int_{-1}^{1} p\delta u a d\xi = \delta \mathbf{u}^{eT} a \int_{-1}^{1} p\mathbf{N}^T d\xi \tag{3.16}$$

or

$$\delta W^e = \delta \mathbf{u}^{eT} \mathbf{f}^e \tag{3.17}$$

where the vector of nodal forces that are equivalent to distributed forces is given by

$$\mathbf{f}^e = a\int_{-1}^{1} p\mathbf{N}^T d\xi = \frac{ap}{2}\int_{-1}^{1} \begin{bmatrix} 1-\xi \\ 1+\xi \end{bmatrix} d\xi = ap\begin{bmatrix} 1 \\ 1 \end{bmatrix} \tag{3.18}$$

For a system of bars, the contribution of each element must be assembled. For example in the bar of figure 3.3, we consider five nodes and four elements. In this case the structure vector of displacements is given by

$$\mathbf{u}^T = \begin{bmatrix} u_1 & u_2 & u_3 & u_4 & u_5 \end{bmatrix} \tag{3.19}$$

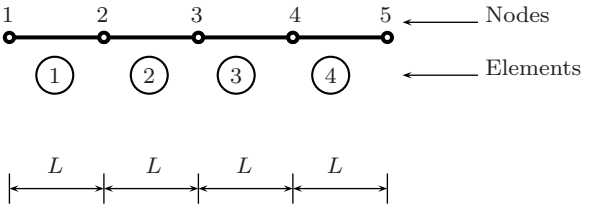


Fig. 3.3 Bar discretized into four elements

Summing the contribution of all elements, we obtain the strain energy and the energy done by the external forces as

$$U = \frac{1}{2}\mathbf{u}^T \sum_{e=1}^{4} \mathbf{K}^e \mathbf{u} = \frac{1}{2}\mathbf{u}^T \mathbf{K}\mathbf{u} \tag{3.20}$$

$$\delta W = \delta\mathbf{u}^T \sum_{e=1}^{4} \mathbf{f}^e = \delta\mathbf{u}^T \mathbf{f} \tag{3.21}$$

where $\mathbf{K}$ and $\mathbf{f}$ are the structure stiffness matrix and the force vector, respectively.

The stiffness matrix is then assembled as

$$\mathbf{K} = \frac{EA}{L}\left\{ \underbrace{\begin{bmatrix} 1 & -1 & 0 & 0 & 0 \\ -1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}}_{\text{element 1}} + \underbrace{\begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & -1 & 0 & 0 \\ 0 & -1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}}_{\text{element 2}} + \ldots \right\} = \frac{EA}{L}\begin{bmatrix} 1 & -1 & 0 & 0 & 0 \\ -1 & 2 & -1 & 0 & 0 \\ 0 & -1 & 2 & -1 & 0 \\ 0 & 0 & -1 & 2 & -1 \\ 0 & 0 & 0 & -1 & 1 \end{bmatrix} \tag{3.22}$$

whereas the vector of equivalent forces is given by

$$\mathbf{f} = ap\begin{bmatrix} 1 \\ 2 \\ 2 \\ 2 \\ 1 \end{bmatrix} \tag{3.23}$$

We then obtain a global system of equations

$$\mathbf{K}\mathbf{u} = \mathbf{f} \tag{3.24}$$

to be solved after the imposition of the boundary conditions as explained before.

3.2 Numerical integration

The integrals arising from the variational formulation can be solved by numerical integration, for example by Gauss quadrature. In this section we present the Gauss method for the solution of one dimensional integrals. We consider a function $f(x),\ x \in [-1, 1]$. In the Gauss method, the integral

$$I = \int_{-1}^{1} f(x)dx \tag{3.25}$$

Table 3.1 Coordinates and weights for Gauss integration

n	$\pm x_i$	W_i
1	0.0	2.0
2	0.5773502692	1.0
3	0.774596697	0.5555555556
	0.0	0.8888888889
4	0.86113663116	0.3478548451
	0.3399810436	0.6521451549

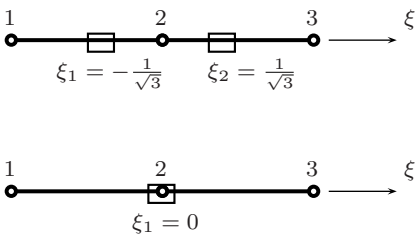


Fig. 3.4 One dimensional Gauss quadrature for two and one Gauss points

is replaced by a sum of p Gauss points, in which the function at those points is multiplied by some weights, as in

$$I = \int_{-1}^{1} f(x)dx = \sum_{i=1}^{p} f(x_i)W_i \tag{3.26}$$

where W_i is the i-th point weight. In table 3.1 the coordinates and weights of the Gauss technique are presented. This technique is exact for a $2n-1$ polynomial if we use at least n Gauss points. In figure 3.4 the Gauss points positions are illustrated.

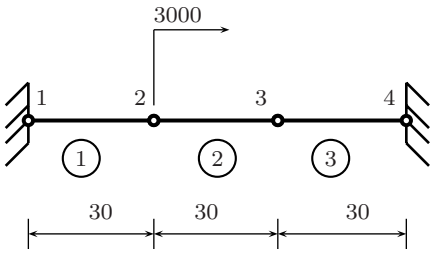
3.3 An example of isoparametric bar

MATLAB code problem2.m solves the bar problem illustrated in figure 3.5, in which the modulus of elasticity is $E = 30e6$, and the area of the cross-section is $A = 1$.

The code problem2.m considers an isoparametric formulation.

```
%.....................................................................
% MATLAB codes for Finite Element Analysis
```

Fig. 3.5 Clamped bar subjected to point load, problem2.m




```
% problem2.m
% antonio ferreira 2008

% clear memory
clear all

% E; modulus of elasticity
% A: area of cross section
% L: length of bar
E  = 30e6;A=1;EA=E*A; L  = 90;

% generation of coordinates and connectivities
% numberElements: number of elements
numberElements=3;
% generation equal spaced coordinates
nodeCoordinates=linspace(0,L,numberElements+1);
xx=nodeCoordinates;
% numberNodes: number of nodes
numberNodes=size(nodeCoordinates,2);

% elementNodes: connections at elements
ii=1:numberElements;
elementNodes(:,1)=ii;
elementNodes(:,2)=ii+1;

% for structure:
    % displacements: displacement vector
    % force : force vector
    % stiffness: stiffness matrix
displacements=zeros(numberNodes,1);
force=zeros(numberNodes,1);
stiffness=zeros(numberNodes,numberNodes);
```

```
% applied load at node 2
force(2)=3000.0;

% computation of the system stiffness matrix
for e=1:numberElements;
  % elementDof: element degrees of freedom (Dof)
  elementDof=elementNodes(e,:) ;
  nn=length(elementDof);
  length_element=nodeCoordinates(elementDof(2))...
      -nodeCoordinates(elementDof(1));
  detJacobian=length_element/2;invJacobian=1/detJacobian;

  % central Gauss point (xi=0, weight W=2)
  [shape,naturalDerivatives]=shapeFunctionL2(0.0);
     Xderivatives=naturalDerivatives*invJacobian;

% B matrix
B=zeros(1,nn);  B(1:nn)  = Xderivatives(:);
  stiffness(elementDof,elementDof)=...
      stiffness(elementDof,elementDof)+B'*B*2*detJacobian*EA;
end

% boundary conditions and solution
% prescribed dofs
fixedDof=find(xx==min(nodeCoordinates(:)) ...
    | xx==max(nodeCoordinates(:)))';
prescribedDof=[fixedDof]
% free Dof : activeDof
activeDof=setdiff([1:numberNodes]',[prescribedDof]);

% solution
GDof=numberNodes;
displacements=solution(GDof,prescribedDof,stiffness,force);

% output displacements/reactions
outputDisplacementsReactions(displacements,stiffness,...
    numberNodes,prescribedDof)
```

The nodal coordinates are obtained by an equal-spaced division of the domain, using `linspace`.

```
% generation of coordinates and connectivities
% numberElements: number of elements
numberElements=3;
```

```
% generation equal spaced coordinates
nodeCoordinates=linspace(0,L,numberElements+1);
```

The connectivities are obtained by a vectorized cycle

```
% elementNodes: connections at elements
ii=1:numberElements;
elementNodes(:,1)=ii;
elementNodes(:,2)=ii+1;
```

We use a Gauss quadrature with one central point $\xi = 0$ and weight 2. The evaluation of the stiffness matrix involves the integral (3.15) by

```
  stiffness(elementDof,elementDof)=...
      stiffness(elementDof,elementDof)+B'*B*2*detJacobian*EA;
```

where $\mathbf{B}$ is a matrix with the derivatives of the shape functions

```
% B matrix
B=zeros(1,nn);  B(1:nn)  = Xderivatives(:);
  stiffness(elementDof,elementDof)=...
      stiffness(elementDof,elementDof)+B'*B*2*detJacobian*EA;
```

The shape function and derivatives with respect to natural coordinates are computed in function shapeFunctionL2.m.

```
% ..........................................................
    function [shape,naturalDerivatives]=shapeFunctionL2(xi)

    % shape function and derivatives for L2 elements
    % shape : Shape functions
    % naturalDerivatives: derivatives w.r.t. xi
    % xi: natural coordinates (-1 ... +1)

    shape=([1-xi,1+xi]/2)';

    naturalDerivatives=[-1;1]/2;

    end % end function shapeFunctionL2
```

The function (solution.m) will be used in the remaining of the book. This function computes the displacements of any FE system in the forthcoming problems.

```
%.....................................................................

function displacements=solution(GDof,prescribedDof,stiffness,
    force)
% function to find solution in terms of global displacements
activeDof=setdiff([1:GDof]', ...
    [prescribedDof]);
U=stiffness(activeDof,activeDof)\force(activeDof);
displacements=zeros(GDof,1);
displacements(activeDof)=U;
```

3.4 Problem 2, using MATLAB struct

Another possible code using MATLAB structures would be problem2Structure.m, as:

```
%.....................................................................

% MATLAB codes for Finite Element Analysis
% problem2Structure.m
% antonio ferreira 2008

% clear memory
clear all

% p1 : structure
p1=struct();

% E; modulus of elasticity
% A: area of cross section
% L: length of bar
E  = 30e6;A=1;EA=E*A; L  = 90;

% generation of coordinates and connectivities
% numberElements: number of elements
p1.numberElements=3;
% generation equal spaced coordinates
p1.nodeCoordinates=linspace(0,L,p1.numberElements+1);
```

```
p1.xx=p1.nodeCoordinates;
% numberNodes: number of nodes
p1.numberNodes=size(p1.nodeCoordinates,2);

% elementNodes: connections at elements
ii=1:p1.numberElements;
p1.elementNodes(:,1)=ii;
p1.elementNodes(:,2)=ii+1;

% GDof: total degrees of freedom
p1.GDof=p1.numberNodes;

% % numberElements: number of Elements
% p1.numberElements=size(p1.elementNodes,1);
%
% % numberNodes: number of nodes
% p1.numberNodes=4;

% for structure:
    % displacements: displacement vector
    % force : force vector
    % stiffness: stiffness matrix
p1.displacements=zeros(p1.GDof,1);
p1.force=zeros(p1.GDof,1);
p1.stiffness=zeros(p1.GDof);

% applied load at node 2
p1.force(2)=3000.0;

% computation of the system stiffness matrix
for e=1:p1.numberElements;
  % elementDof: element degrees of freedom (Dof)
  elementDof=p1.elementNodes(e,:) ;
  nn=length(elementDof);
  length_element=p1.nodeCoordinates(elementDof(2))...
      -p1.nodeCoordinates(elementDof(1));
  detJacobian=length_element/2;invJacobian=1/detJacobian;

  % central Gauss point (xi=0, weight W=2)
  shapeL2=shapeFunctionL2Structure(0.0);
     Xderivatives=shapeL2.naturalDerivatives*invJacobian;

     % B matrix
```

```
  B=zeros(1,nn);  B(1:nn)  = Xderivatives(:);
  p1.stiffness(elementDof,elementDof)=...
      p1.stiffness(elementDof,elementDof)+B'*B*2*detJacobian*EA;
end

% prescribed dofs
p1.prescribedDof=find(p1.xx==min(p1.nodeCoordinates(:)) ...
    | p1.xx==max(p1.nodeCoordinates(:)))';

% solution
p1.displacements=solutionStructure(p1)

% output displacements/reactions
outputDisplacementsReactionsStructure(p1)
```

The code needs some change in function shapeFunctionL2Structure.m, as follows:

```
% .....................................................
    function shapeL2=shapeFunctionL2Structure(xi)

    % shape function and derivatives for L2 elements
    % shape : Shape functions
    % naturalDerivatives: derivatives w.r.t. xi
    % xi: natural coordinates (-1 ... +1)

    shapeL2=struct()
    shapeL2.shape=([1-xi,1+xi]/2)';
    shapeL2.naturalDerivatives=[-1;1]/2;

    end % end function shapeFunctionL2
```

Results are placed in structure p1, as before.

```
>> p1

p1 =

     numberElements: 3
    nodeCoordinates: [0 30 60 90]
                 xx: [0 30 60 90]
        numberNodes: 4
       elementNodes: [3x2 double]
```

```
              GDof: 4
     displacements: [4x1 double]
             force: [4x1 double]
         stiffness: [4x4 double]
     prescribedDof: [2x1 double]
```

We can obtain detailed information on p1, for example, displacements:

```
>> p1.displacements

ans =

         0
    0.0020
    0.0010
         0
```

3.5 Problem 3

Another problem involving bars and springs is illustrated in figure 3.6. The MATLAB code for this problem is problem3.m, using direct stiffness method.

```
%.....................................................................

% MATLAB codes for Finite Element Analysis
% problem3.m
% ref: D. Logan, A first couse in the finite element method,
% third Edition, page 121, exercise P3-10
% direct stiffness method
% antonio ferreira 2008
```

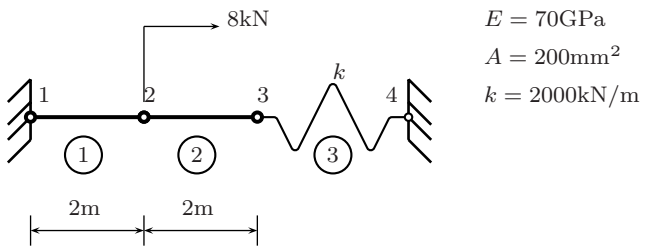


Fig. 3.6 Illustration of problem 3, problem3.m

```
% clear memory
clear all

% E; modulus of elasticity
% A: area of cross section
% L: length of bar
% k: spring stiffness
E=70000;A=200;k=2000;

% generation of coordinates and connectivities
% numberElements: number of elements
numberElements=3;
numberNodes=4;
elementNodes=[1 2; 2 3; 3 4];
nodeCoordinates=[0 2000 4000 4000];
xx=nodeCoordinates;

% for structure:
    % displacements: displacement vector
    % force : force vector
    % stiffness: stiffness matrix
displacements=zeros(numberNodes,1);
force=zeros(numberNodes,1);
stiffness=zeros(numberNodes,numberNodes);

% applied load at node 2
force(2)=8000;

% computation of the system stiffness matrix
for e=1:numberElements;
 % elementDof: element degrees of freedom (Dof)
 elementDof=elementNodes(e,:) ;
 L=nodeCoordinates(elementDof(2))-nodeCoordinates(elementDof(1));
  if e<3
      ea(e)=E*A/L;
  else
      ea(e)=k;
  end
  stiffness(elementDof,elementDof)=...
      stiffness(elementDof,elementDof)+ea(e)*[1 -1;-1 1];
end
% boundary conditions and solution
```

```
% prescribed dofs
prescribedDof=[1;4];
% free Dof : activeDof
activeDof=setdiff([1:numberNodes]',[prescribedDof]);

% solution
GDof=4;
displacements=solution(GDof,prescribedDof,stiffness,force);

% output displacements/reactions
outputDisplacementsReactions(displacements,stiffness,...
    numberNodes,prescribedDof)
```

The isoparametric version for the problem illustrated in figure 3.6 is given in problem3a.m.

```
%.....................................................................

% MATLAB codes for Finite Element Analysis
% problem3a.m
% ref: D. Logan, A first couse in the finite element method,
% third Edition, page 121, exercise P3-10
% with isoparametric formulation
% antonio ferreira 2008

% clear memory
clear all

% E; modulus of elasticity
% A: area of cross section
% L: length of bar
E  = 70000;A=200;EA=E*A;k=2000;

% generation of coordinates and connectivities
numberElements=3;
numberNodes=4;
elementNodes=[1 2; 2 3; 3 4];
nodeCoordinates=[0 2000 4000 4000];
xx=nodeCoordinates;

% for structure:
```

```
    % displacements: displacement vector
    % force : force vector
    % stiffness: stiffness matrix
displacements=zeros(numberNodes,1);
force=zeros(numberNodes,1);
stiffness=zeros(numberNodes,numberNodes);

% applied load at node 2
force(2)=8000.0;

% computation of the system stiffness matrix
for e=1:numberElements;
  % elementDof: element degrees of freedom (Dof)
  elementDof=elementNodes(e,:) ;

  if e<3 % bar elements
  nn=length(elementDof);
  length_element=nodeCoordinates(elementDof(2))...
      -nodeCoordinates(elementDof(1));
  detJacobian=length_element/2;invJacobian=1/detJacobian;
% central Gauss point (xi=0, weight W=2)
  % central Gauss point (xi=0, weight W=2)
  [shape,naturalDerivatives]=shapeFunctionL2(0.0);
     Xderivatives=naturalDerivatives*invJacobian;

% B matrix
B=zeros(1,nn);  B(1:nn)  = Xderivatives(:);
  ea(e)=E*A;
  stiffness(elementDof,elementDof)=...
      stiffness(elementDof,elementDof)+B'*B*2*detJacobian*ea(e);
  else % spring element
       stiffness(elementDof,elementDof)=...
      stiffness(elementDof,elementDof)+k*[1 -1;-1 1];
  end
end

% boundary conditions and solution
prescribedDof=[1;4];
GDof=4;

% solution
displacements=solution(GDof,prescribedDof,stiffness,force);
```

```
% output displacements/reactions
outputDisplacementsReactions(displacements,stiffness,...
    numberNodes,prescribedDof)
```

For both codes, the solution is the same and matches the analytical solution presented in Logan [11]. The displacements at nodes 2 and 3 are 0.935 and 0.727 mm, respectively. The reactions at the supports 1 and 4 are −6.546 and −1.455 kN, respectively.

The code problem3Structure.m is equivalent to problem3.m, but using MATLAB structures.

```
%.....................................................................

% MATLAB codes for Finite Element Analysis
% problem3Structure.m
% antonio ferreira 2008

% clear memory
clear all

% p1 : structure
p1=struct();

% E; modulus of elasticity
% A: area of cross section
% L: length of bar
% k: spring stiffness
E=70000;A=200;k=2000;

% generation of coordinates and connectivities
% numberElements: number of elements
p1.numberElements=3;
p1.numberNodes=4;
p1.elementNodes=[1 2; 2 3; 3 4];
p1.nodeCoordinates=[0 2000 4000 4000];
p1.xx=p1.nodeCoordinates;

% GDof: total degrees of freedom
p1.GDof=p1.numberNodes;

% for structure:
```

```
    % displacements: displacement vector
    % force : force vector
    % stiffness: stiffness matrix
p1.displacements=zeros(p1.GDof,1);
p1.force=zeros(p1.GDof,1);
p1.stiffness=zeros(p1.GDof);

% applied load at node 2
p1.force(2)=8000.0;

% computation of the system stiffness matrix
for e=1:p1.numberElements;
  % elementDof: element degrees of freedom (Dof)
  elementDof=p1.elementNodes(e,:) ;
  L=p1.nodeCoordinates(elementDof(2))-...
      p1.nodeCoordinates(elementDof(1));
  if e<3
      ea(e)=E*A/L;
  else
      ea(e)=k;
  end
  p1.stiffness(elementDof,elementDof)=...
      p1.stiffness(elementDof,elementDof)+ea(e)*[1 -1;-1 1];
end

% prescribed dofs
p1.prescribedDof=[1;4];

% solution
p1.displacements=solutionStructure(p1)

% output displacements/reactions
outputDisplacementsReactionsStructure(p1)
```