CALFEM

for

Finite Element Method - Flow Analysis, VSMN25

Division of Structural Mechanics
Lund University

E-mail address:

`calfem@byggmek.lth.se`

Homepage:

`http://www.byggmek.lth.se/Calfem`

Contacts:

The Division of Structural Mechanics
Lund University
PO Box 118
SE–221 00 Lund
SWEDEN
Phone: +46 46 222 0000
Fax: +46 46 222 4420

Preface

CALFEM® is an interactive computer program for teaching the finite element method (FEM). The name CALFEM is an abbreviation of "Computer Aided Learning of the Finite Element Method". The program can be used for different types of structural mechanics problems and field problems.

CALFEM, the program and its built-in philosophy have been developed at the Division of Structural Mechanics, Lund University, starting in the late 70's. Many coworkers, former and present, have been engaged in the development at different stages.

This release represents the latest development of CALFEM. The functions for finite element applications are all MATLAB functions (.m-files) as described in the MATLAB manual. We believe that this environment increases the versatility and handling of the program and, above all, the ease of teaching the finite element method. CALFEM also works with Octave, presently with exception for some graphical functions.

Lund, September 20, 2019

The authors

Contents

1 Element functions

1.1 Introduction

The group of element functions contains functions for computation of element matrices and element forces for different element types. The element functions have been divided into the following groups

Spring element
Bar elements
Heat flow elements
Solid elements
Beam elements
Plate element

For each element type there is a function for computation of the element stiffness matrix $\mathbf{K}^e$. For most of the elements, an element load vector $\mathbf{f}^e$ can also be computed. These functions are identified by their last letter -e.

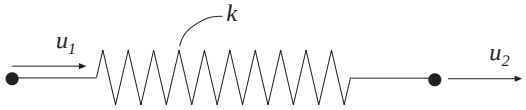
Using the function assem, the element stiffness matrices and element load vectors are assembled into a global stiffness matrix $\mathbf{K}$ and a load vector $\mathbf{f}$. Unknown nodal values of temperatures or displacements $\mathbf{a}$ are computed by solving the system of equations $\mathbf{Ka} = \mathbf{f}$ using the function solveq. A vector of nodal values of temperatures or displacements for a specific element is formed by the function extract.

When the element nodal values have been computed, the element flux or element stresses can be calculated using functions specific to the element type concerned. These functions are identified by their last letter -s.

For some elements, a function for computing the internal force vector is also available. These functions are identified by their last letter -f.

1.2 Spring element

The spring element, shown below, can be used for the analysis of one-dimensional spring systems and for a variety of analogous physical problems.



Quantities corresponding to the variables of the spring are listed in Table 1.

Problem type	Spring stiffness	Nodal displacement	Element force	Spring force
Spring	k	u	P	N
Bar	$\frac{EA}{L}$	u	P	N
Thermal conduction	$\frac{\lambda A}{L}$	T	$\bar{H}$	H
Electrical circuit	$\frac{1}{R}$	U	$\bar{I}$	I
Groundwater flow	$\frac{kA}{L}$	ϕ	$\bar{H}$	H
Pipe network	$\frac{\pi D^4}{128\mu L}$	p	$\bar{H}$	H

Table 1: *Analogous quantities*

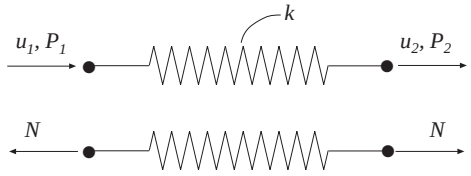
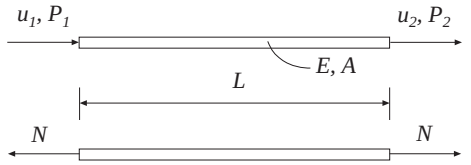
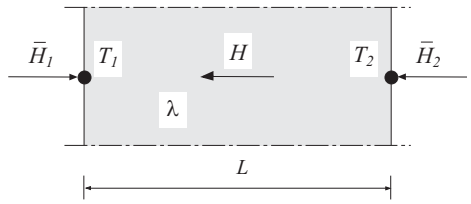
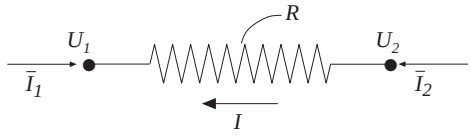
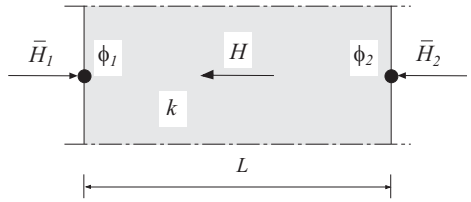
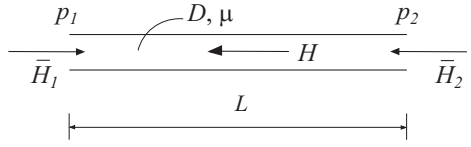
Interpretations of the spring element			
Problem type	Quantities	Designations	
Spring		k u P N	spring stiffness displacement element force spring force
Bar		L E A u P N	length modulus of elasticity area of cross section displacement element force normal force
Thermal conduction		L λ T $\bar{H}$ H	length thermal conductivity temperature element heat flow internal heat flow
Electrical circuit		R U $\bar{I}$ I	resistance potential element current internal current
Ground-water flow		L k ϕ $\bar{H}$ H	length permeability piezometric head element water flow internal water flow
Pipe network (laminar flow)		L D μ p $\bar{H}$ H	length pipe diameter viscosity pressure element fluid flow internal fluid flow

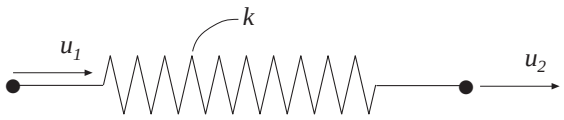
Table 2: *Quantities used in different types of problems*

The following functions are available for the spring element:

Spring functions	
spring1e	Compute element matrix
spring1s	Compute spring force

Purpose:

Compute element stiffness matrix for a spring element.



Syntax:

Ke=spring1e(ep)

Description:

spring1e provides the element stiffness matrix Ke for a spring element.

The input variable

$$\mathsf{ep} = [\ k\ ]$$

supplies the spring stiffness k or the analog quantity defined in Table 1.

Theory:

The element stiffness matrix $\mathbf{K}^e$, stored in Ke, is computed according to

$$\mathbf{K}^e = \begin{bmatrix} k & -k \\ -k & k \end{bmatrix}$$

where k is defined by ep.

Purpose:

Compute spring force in a spring element.

Syntax:

es=spring1s(ep,ed)

Description:

spring1s computes the spring force es in a spring element.

The input variable ep is defined in spring1e and the element nodal displacements ed are obtained by the function extract.

The output variable

$$\mathsf{es} = [\ N\ ]$$

contains the spring force N, or the analog quantity.

Theory:

The spring force N, or analog quantity, is computed according to

$$N = k\,[\ u_2 - u_1\ ]$$

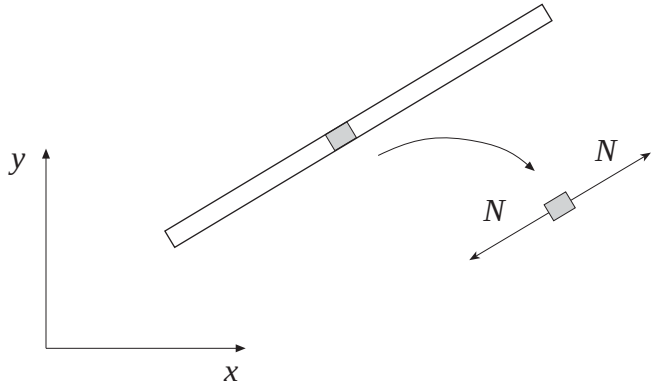
1.3 Bar elements

Bar elements are available for one, two, and three dimensional analysis. In this manual, only two dimensional elements are shown.

Two dimensional bar elements	
bar2e	Compute element matrix
bar2s	Compute normal force

Purpose:

Compute normal force in a two dimensional bar element.



Syntax:

es=bar2s(ex,ey,ep,ed)
es=bar2s(ex,ey,ep,ed,eq)
[es,edi]=bar2s(ex,ey,ep,ed,eq,n)
[es,edi,eci]=bar2s(ex,ey,ep,ed,eq,n)

Description:

bar2s computes the normal force in the two dimensional bar element bar2e.

The input variables ex, ey, and ep are defined in bar2e and the element nodal displacements, stored in ed, are obtained by the function extract. If distributed loads are applied to the element, the variable eq must be included. The number of evaluation points for section forces and displacements are determined by n. If n is omitted, only the ends of the bar are evaluated.

The output variables

$$\mathsf{es} = \begin{bmatrix} N(0) \\ N(\bar{x}_2) \\ \vdots \\ N(\bar{x}_{n-1}) \\ N(L) \end{bmatrix} \qquad \mathsf{edi} = \begin{bmatrix} u(0) \\ u(\bar{x}_2) \\ \vdots \\ u(\bar{x}_{n-1}) \\ u(L) \end{bmatrix} \qquad \mathsf{eci} = \begin{bmatrix} 0 \\ \bar{x}_2 \\ \vdots \\ \bar{x}_{n-1} \\ L \end{bmatrix}$$

contain the normal force, the displacement, and the evaluation points on the local $\bar{x}$-axis. L is the length of the bar element.

Theory:

The nodal displacements in global coordinates

$$\mathbf{a^e} = [\ \mathbf{u_1}\ \ \mathbf{u_2}\ \ \mathbf{u_3}\ \ \mathbf{u_4}\ ]^{\mathbf{T}}$$

are also shown in bar2e. The transpose of $\mathbf{a^e}$ is stored in ed.

The nodal displacements in local coordinates are given by

$$\bar{\mathbf{a}}^{\mathbf{e}} = \mathbf{G}\mathbf{a}^{\mathbf{e}}$$

where the transformation matrix $\mathbf{G}$ is defined in bar2e.

The displacement $u(\bar{x})$ and the normal force $N(\bar{x})$ are computed from

$$u(\bar{x}) = \mathbf{N}\bar{\mathbf{a}}^{\mathbf{e}} \quad + u_p(\bar{x})$$

$$N(\bar{x}) = D_{EA}\mathbf{B}\bar{\mathbf{a}}^{\mathbf{e}} \quad + N_p(\bar{x})$$

where

$$\mathbf{N} = \begin{bmatrix} 1 & \bar{x} \end{bmatrix} \mathbf{C}^{-1} = \begin{bmatrix} 1 - \frac{\bar{x}}{L} & \frac{\bar{x}}{L} \end{bmatrix}$$

$$\mathbf{B} = \begin{bmatrix} 0 & 1 \end{bmatrix} \mathbf{C}^{-1} = \frac{1}{L} \begin{bmatrix} -1 & 1 \end{bmatrix}$$

$$u_p(\bar{x}) = \quad -\frac{q_{\bar{x}}}{D_{EA}} \left(\frac{\bar{x}^2}{2} - \frac{L\bar{x}}{2} \right)$$

$$N_p(\bar{x}) = \quad -q_{\bar{x}} \left(\bar{x} - \frac{L}{2} \right)$$

where D_{EA}, L, $q_{\bar{x}}$ are defined in bar2e and

$$\mathbf{C}^{-1} = \begin{bmatrix} 1 & 0 \\ -\frac{1}{L} & \frac{1}{L} \end{bmatrix}$$

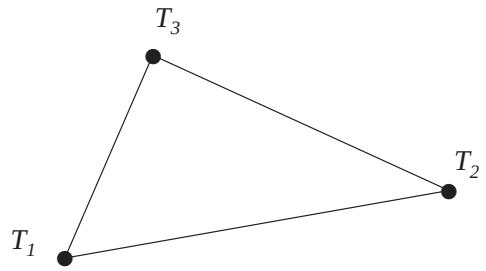
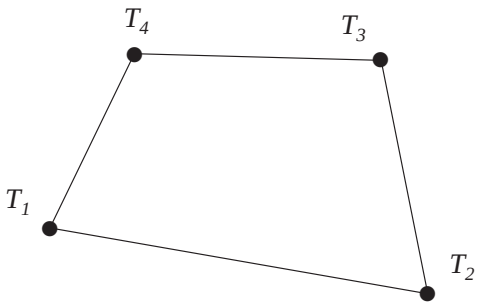
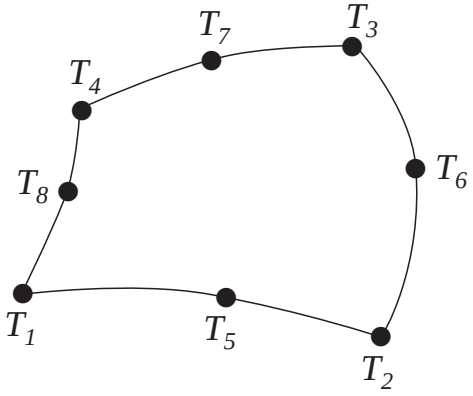
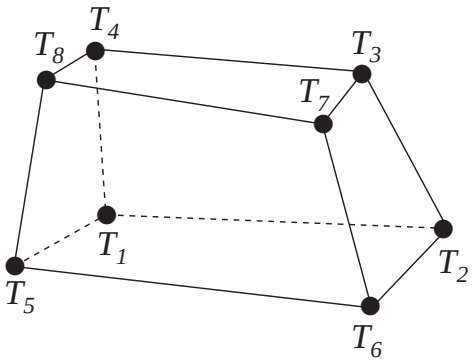
1.4 Heat flow elements

Heat flow elements are available for one, two, and three dimensional analysis. For one dimensional heat flow the spring element spring1 is used.

A variety of important physical phenomena are described by the same differential equation as the heat flow problem. The heat flow element is thus applicable in modelling different physical applications. Table 3 below shows the relation between the primary variable **a**, the constitutive matrix **D**, and the load vector $\mathbf{f}_l$ for a chosen set of two dimensional physical problems.

Problem type	**a**	**D**	$\mathbf{f}_l$	Designation
Heat flow	T	λ_x , λ_y	Q	T = temperature λ_x , λ_y = thermal conductivity Q = heat supply
Groundwater flow	ϕ	k_x , k_y,	Q	ϕ = piezometric head k_x, k_y = permeabilities Q = fluid supply
St. Venant torsion	ϕ	$\frac{1}{G_{zy}}$, $\frac{1}{G_{zx}}$	2Θ	ϕ = stress function G_{zy}, G_{zx} = shear moduli Θ = angle of torsion per unit length

Table 3: *Problem dependent parameters*

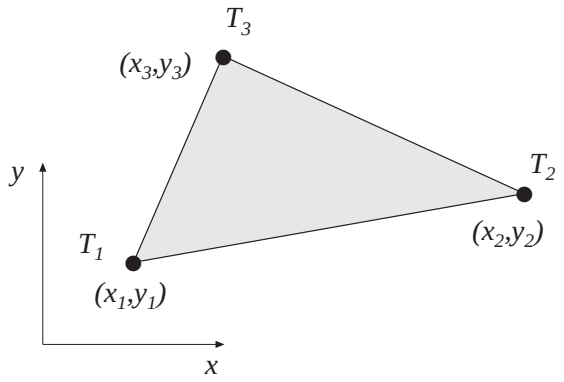
Heat flow elements	
 flw2te	 flw2qe flw2i4e
 flw2i8e	 flw3i8e

2D heat flow functions	
flw2te	Compute element matrices for a triangular element
flw2ts	Compute temperature gradients and flux
flw2qe	Compute element matrices for a quadrilateral element
flw2qs	Compute temperature gradients and flux
flw2i4e	Compute element matrices, 4 node isoparametric element
flw2i4s	Compute temperature gradients and flux
flw2i8e	Compute element matrices, 8 node isoparametric element
flw2i8s	Compute temperature gradients and flux

3D heat flow functions	
flw3i8e	Compute element matrices, 8 node isoparametric element
flw3i8s	Compute temperature gradients and flux

Purpose:

Compute element stiffness matrix for a triangular heat flow element.



Syntax:

Ke=flw2te(ex,ey,ep,D)
[Ke,fe]=flw2te(ex,ey,ep,D,eq)

Description:

flw2te provides the element stiffness (conductivity) matrix Ke and the element load vector fe for a triangular heat flow element.

The element nodal coordinates x_1, y_1, x_2 etc, are supplied to the function by ex and ey, the element thickness t is supplied by ep and the thermal conductivities (or corresponding quantities) k_{xx}, k_{xy} etc are supplied by D.

$$\mathsf{ex} = \begin{bmatrix} x_1 & x_2 & x_3 \end{bmatrix} \qquad \mathsf{ep} = \begin{bmatrix} t \end{bmatrix} \qquad \mathsf{D} = \begin{bmatrix} k_{xx} & k_{xy} \\ k_{yx} & k_{yy} \end{bmatrix}$$
$$\mathsf{ey} = \begin{bmatrix} y_1 & y_2 & y_3 \end{bmatrix}$$

If the scalar variable eq is given in the function, the element load vector fe is computed, using

$$\mathsf{eq} = \begin{bmatrix} Q \end{bmatrix}$$

where Q is the heat supply per unit volume.

Theory:

The element stiffness matrix $\mathbf{K}^e$ and the element load vector $\mathbf{f}_l^e$, stored in Ke and fe, respectively, are computed according to

$$\mathbf{K}^e = (\mathbf{C}^{-1})^T \int_A \bar{\mathbf{B}}^T \, \mathbf{D} \, \bar{\mathbf{B}} \, t \, dA \;\; \mathbf{C}^{-1}$$
$$\mathbf{f}_l^e = (\mathbf{C}^{-1})^T \int_A \bar{\mathbf{N}}^T \, Q \, t \, dA$$

with the constitutive matrix $\mathbf{D}$ defined by D.

The evaluation of the integrals for the triangular element is based on the linear temperature approximation $T(x, y)$ and is expressed in terms of the nodal variables T_1, T_2 and T_3 as

$$T(x, y) = \mathbf{N}^e \mathbf{a}^e = \bar{\mathbf{N}} \, \mathbf{C}^{-1} \mathbf{a}^e$$

where

$$\bar{\mathbf{N}} = [\ 1 \ \ x \ \ y\ ] \qquad \mathbf{C} = \begin{bmatrix} 1 & x_1 & y_1 \\ 1 & x_2 & y_2 \\ 1 & x_3 & y_3 \end{bmatrix} \qquad \mathbf{a}^e = \begin{bmatrix} T_1 \\ T_2 \\ T_3 \end{bmatrix}$$

and hence it follows that

$$\bar{\mathbf{B}} = \nabla \bar{\mathbf{N}} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \qquad \nabla = \begin{bmatrix} \dfrac{\partial}{\partial x} \\ \dfrac{\partial}{\partial y} \end{bmatrix}$$

Evaluation of the integrals for the triangular element yields

$$\mathbf{K}^e = (\mathbf{C}^{-1})^T\ \bar{\mathbf{B}}^T\ \mathbf{D}\ \bar{\mathbf{B}}\ \mathbf{C}^{-1}\ t\ A$$

$$\mathbf{f}_l^e = \frac{QAt}{3}\,[\ 1 \ \ 1 \ \ 1\ ]^T$$

where the element area A is determined as

$$A = \frac{1}{2}\det \mathbf{C}$$

Purpose:

Compute heat flux and temperature gradients in a triangular heat flow element.

Syntax:

[es,et]=flw2ts(ex,ey,D,ed)

Description:

flw2ts computes the heat flux vector es and the temperature gradient et (or corresponding quantities) in a triangular heat flow element.

The input variables ex, ey and the matrix D are defined in flw2te. The vector ed contains the nodal temperatures $\mathbf{a}^e$ of the element and is obtained by the function extract as

$$\mathsf{ed} = (\mathbf{a}^e)^T = [\ T_1 \ \ T_2 \ \ T_3\ ]$$

The output variables

$$\mathsf{es} = \mathbf{q}^T = [\ q_x \ q_y\ ]$$

$$\mathsf{et} = (\nabla T)^T = \left[\ \frac{\partial T}{\partial x} \ \ \frac{\partial T}{\partial y}\ \right]$$

contain the components of the heat flux and the temperature gradient computed in the directions of the coordinate axis.

Theory:

The temperature gradient and the heat flux are computed according to

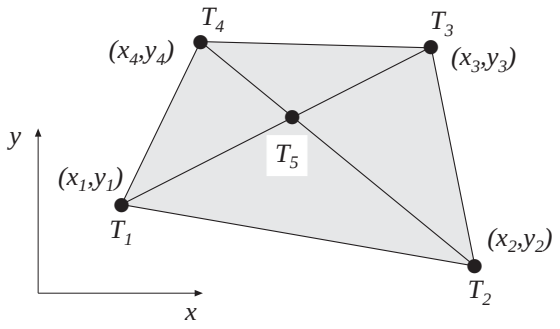
$$\nabla T = \bar{\mathbf{B}}\ \mathbf{C}^{-1}\ \mathbf{a}^e$$

$$\mathbf{q} = -\mathbf{D}\nabla T$$

where the matrices $\mathbf{D}$, $\bar{\mathbf{B}}$, and $\mathbf{C}$ are described in flw2te. Note that both the temperature gradient and the heat flux are constant in the element.

Purpose:

Compute element stiffness matrix for a quadrilateral heat flow element.



Syntax:

Ke=flw2qe(ex,ey,ep,D)
[Ke,fe]=flw2qe(ex,ey,ep,D,eq)

Description:

flw2qe provides the element stiffness (conductivity) matrix Ke and the element load vector fe for a quadrilateral heat flow element.

The element nodal coordinates x_1, y_1, x_2 etc, are supplied to the function by ex and ey, the element thickness t is supplied by ep and the thermal conductivities (or corresponding quantities) k_{xx}, k_{xy} etc are supplied by D.

$$\mathsf{ex} = \begin{bmatrix} x_1 & x_2 & x_3 & x_4 \end{bmatrix} \qquad \mathsf{ep} = \begin{bmatrix} t \end{bmatrix} \qquad \mathsf{D} = \begin{bmatrix} k_{xx} & k_{xy} \\ k_{yx} & k_{yy} \end{bmatrix}$$
$$\mathsf{ey} = \begin{bmatrix} y_1 & y_2 & y_3 & y_4 \end{bmatrix}$$

If the scalar variable eq is given in the function, the element load vector fe is computed, using

$$\mathsf{eq} = \begin{bmatrix} Q \end{bmatrix}$$

where Q is the heat supply per unit volume.

Theory:

In computing the element matrices, a fifth degree of freedom is introduced. The location of this extra degree of freedom is defined by the mean value of the coordinates in the corner points. Four sets of element matrices are calculated using flw2te. These matrices are then assembled and the fifth degree of freedom is eliminated by static condensation.

Purpose:

Compute heat flux and temperature gradients in a quadrilateral heat flow element.

Syntax:

[es,et]=flw2qs(ex,ey,ep,D,ed)
[es,et]=flw2qs(ex,ey,ep,D,ed,eq)

Description:

flw2qs computes the heat flux vector es and the temperature gradient et (or corresponding quantities) in a quadrilateral heat flow element.

The input variables ex, ey, eq and the matrix D are defined in flw2qe. The vector ed contains the nodal temperatures $\mathbf{a}^e$ of the element and is obtained by the function extract as

$$\mathsf{ed} = (\mathbf{a}^e)^T = [\ T_1 \ \ T_2 \ \ T_3 \ \ T_4\ ]$$

The output variables

$$\mathsf{es} = \mathbf{q}^T = [\ q_x \ q_y\ ]$$

$$\mathsf{et} = (\nabla T)^T = \left[\ \frac{\partial T}{\partial x} \ \ \frac{\partial T}{\partial y}\ \right]$$

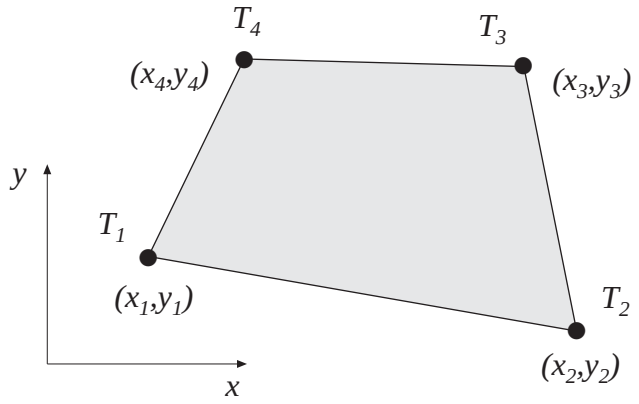
contain the components of the heat flux and the temperature gradient computed in the directions of the coordinate axis.

Theory:

By assembling four triangular elements as described in flw2te a system of equations containing 5 degrees of freedom is obtained. From this system of equations the unknown temperature at the center of the element is computed. Then according to the description in flw2ts the temperature gradient and the heat flux in each of the four triangular elements are produced. Finally the temperature gradient and the heat flux of the quadrilateral element are computed as area weighted mean values from the values of the four triangular elements. If heat is supplied to the element, the element load vector eq is needed for the calculations.

Purpose:

Compute element stiffness matrix for a 4 node isoparametric heat flow element.



Syntax:

Ke=flw2i4e(ex,ey,ep,D)
[Ke,fe]=flw2i4e(ex,ey,ep,D,eq)

Description:

flw2i4e provides the element stiffness (conductivity) matrix Ke and the element load vector fe for a 4 node isoparametric heat flow element.

The element nodal coordinates x_1, y_1, x_2 etc, are supplied to the function by ex and ey. The element thickness t and the number of Gauss points n

$$(n \times n) \text{ integration points, } n = 1, 2, 3$$

are supplied to the function by ep and the thermal conductivities (or corresponding quantities) k_{xx}, k_{xy} etc are supplied by D.

$$\mathsf{ex} = [\, x_1 \;\; x_2 \;\; x_3 \;\; x_4 \,] \qquad \mathsf{ep} = [\, t \;\; n \,] \qquad \mathsf{D} = \begin{bmatrix} k_{xx} & k_{xy} \\ k_{yx} & k_{yy} \end{bmatrix}$$
$$\mathsf{ey} = [\, y_1 \;\; y_2 \;\; y_3 \;\; y_4 \,]$$

If the scalar variable eq is given in the function, the element load vector fe is computed, using

$$\mathsf{eq} = [\, Q \,]$$

where Q is the heat supply per unit volume.

Theory:

The element stiffness matrix $\mathbf{K}^e$ and the element load vector $\mathbf{f}_l^e$, stored in Ke and fe, respectively, are computed according to

$$\mathbf{K}^e = \int_A \mathbf{B}^{eT}\, \mathbf{D}\, \mathbf{B}^e\, t\, dA$$
$$\mathbf{f}_l^e = \int_A \mathbf{N}^{eT}\, Q\, t\, dA$$

with the constitutive matrix $\mathbf{D}$ defined by D.

The evaluation of the integrals for the isoparametric 4 node element is based on a temperature approximation $T(\xi, \eta)$, expressed in a local coordinates system in terms of the nodal variables T_1, T_2, T_3 and T_4 as

$$T(\xi, \eta) = \mathbf{N}^e \mathbf{a}^e$$

where

$$\mathbf{N}^e = [\ N_1^e\ \ N_2^e\ \ N_3^e\ \ N_4^e\ ] \qquad \mathbf{a}^e = [\ T_1\ \ T_2\ \ T_3\ \ T_4\ ]^T$$

The element shape functions are given by

$$N_1^e = \frac{1}{4}(1-\xi)(1-\eta) \qquad N_2^e = \frac{1}{4}(1+\xi)(1-\eta)$$
$$N_3^e = \frac{1}{4}(1+\xi)(1+\eta) \qquad N_4^e = \frac{1}{4}(1-\xi)(1+\eta)$$

The $\mathbf{B}^e$-matrix is given by

$$\mathbf{B}^e = \nabla \mathbf{N}^e = \begin{bmatrix} \dfrac{\partial}{\partial x} \\ \dfrac{\partial}{\partial y} \end{bmatrix} \mathbf{N}^e = (\mathbf{J}^T)^{-1} \begin{bmatrix} \dfrac{\partial}{\partial \xi} \\ \dfrac{\partial}{\partial \eta} \end{bmatrix} \mathbf{N}^e$$

where $\mathbf{J}$ is the Jacobian matrix

$$\mathbf{J} = \begin{bmatrix} \dfrac{\partial x}{\partial \xi} & \dfrac{\partial x}{\partial \eta} \\ \dfrac{\partial y}{\partial \xi} & \dfrac{\partial y}{\partial \eta} \end{bmatrix}$$

Evaluation of the integrals is done by Gauss integration.

Purpose:

Compute heat flux and temperature gradients in a 4 node isoparametric heat flow element.

Syntax:

[es,et,eci]=flw2i4s(ex,ey,ep,D,ed)

Description:

flw2i4s computes the heat flux vector es and the temperature gradient et (or corresponding quantities) in a 4 node isoparametric heat flow element.

The input variables ex, ey, ep and the matrix D are defined in flw2i4e. The vector ed contains the nodal temperatures $\mathbf{a}^e$ of the element and is obtained by extract as

$$\mathsf{ed} = (\mathbf{a}^e)^T = [\ T_1 \ \ T_2 \ \ T_3 \ \ T_4\ ]$$

The output variables

$$\mathsf{es} = \bar{\mathbf{q}}^T \qquad = \begin{bmatrix} q_x^1 & q_y^1 \\ q_x^2 & q_y^2 \\ \vdots & \vdots \\ q_x^{n^2} & q_y^{n^2} \end{bmatrix}$$

$$\mathsf{et} = (\bar{\nabla} T)^T = \begin{bmatrix} \dfrac{\partial T}{\partial x}^1 & \dfrac{\partial T}{\partial y}^1 \\ \dfrac{\partial T}{\partial x}^2 & \dfrac{\partial T}{\partial y}^2 \\ \vdots & \vdots \\ \dfrac{\partial T}{\partial x}^{n^2} & \dfrac{\partial T}{\partial y}^{n^2} \end{bmatrix} \qquad \mathsf{eci} = \begin{bmatrix} x_1 & y_1 \\ x_2 & y_2 \\ \vdots & \vdots \\ x_{n^2} & y_{n^2} \end{bmatrix}$$

contain the heat flux, the temperature gradient, and the coordinates of the integration points. The index n denotes the number of integration points used within the element, cf. flw2i4e.

Theory:

The temperature gradient and the heat flux are computed according to

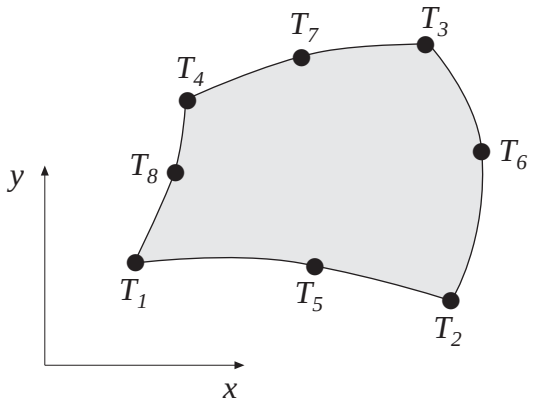
$$\nabla T = \mathbf{B}^e\, \mathbf{a}^e$$

$$\mathbf{q} = -\mathbf{D} \nabla T$$

where the matrices $\mathbf{D}$, $\mathbf{B}^e$, and $\mathbf{a}^e$ are described in flw2i4e, and where the integration points are chosen as evaluation points.

Purpose:

Compute element stiffness matrix for an 8 node isoparametric heat flow element.



Syntax:

Ke=flw2i8e(ex,ey,ep,D)
[Ke,fe]=flw2i8e(ex,ey,ep,D,eq)

Description:

flw2i8e provides the element stiffness (conductivity) matrix Ke and the element load vector fe for an 8 node isoparametric heat flow element.

The element nodal coordinates x_1, y_1, x_2 etc, are supplied to the function by ex and ey. The element thickness t and the number of Gauss points n

$$(n \times n) \text{ integration points, } n = 1, 2, 3$$

are supplied to the function by ep and the thermal conductivities (or corresponding quantities) k_{xx}, k_{xy} etc are supplied by D.

$$\mathsf{ex} = [\ x_1\ \ x_2\ \ x_3\ \ \ldots\ \ x_8\ ] \qquad \mathsf{ey} = [\ y_1\ \ y_2\ \ y_3\ \ \ldots\ \ y_8\ ]$$

$$\mathsf{ep} = [\ t\ \ n\ ] \qquad \mathsf{D} = \begin{bmatrix} k_{xx} & k_{xy} \\ k_{yx} & k_{yy} \end{bmatrix}$$

If the scalar variable eq is given in the function, the vector fe is computed, using

$$\mathsf{eq} = [\ Q\ ]$$

where Q is the heat supply per unit volume.

Theory:

The element stiffness matrix $\mathbf{K}^e$ and the element load vector $\mathbf{f}_l^e$, stored in Ke and fe, respectively, are computed according to

$$\mathbf{K}^e = \int_A \mathbf{B}^{eT}\, \mathbf{D}\, \mathbf{B}^e\, t\, dA$$
$$\mathbf{f}_l^e = \int_A \mathbf{N}^{eT}\, Q\, t\, dA$$

with the constitutive matrix $\mathbf{D}$ defined by D.

The evaluation of the integrals for the 2D isoparametric 8 node element is based on a temperature approximation $T(\xi,\eta)$, expressed in a local coordinates system in terms of the nodal variables T_1 to T_8 as

$$T(\xi,\eta) = \mathbf{N}^e\mathbf{a}^e$$

where

$$\mathbf{N}^e = [\, N_1^e \;\; N_2^e \;\; N_3^e \;\; \ldots \;\; N_8^e \,] \qquad \mathbf{a}^e = [\, T_1 \;\; T_2 \;\; T_3 \;\; \ldots \;\; T_8 \,]^T$$

The element shape functions are given by

$$N_1^e = -\frac{1}{4}(1-\xi)(1-\eta)(1+\xi+\eta) \qquad N_5^e = \frac{1}{2}(1-\xi^2)(1-\eta)$$
$$N_2^e = -\frac{1}{4}(1+\xi)(1-\eta)(1-\xi+\eta) \qquad N_6^e = \frac{1}{2}(1+\xi)(1-\eta^2)$$
$$N_3^e = -\frac{1}{4}(1+\xi)(1+\eta)(1-\xi-\eta) \qquad N_7^e = \frac{1}{2}(1-\xi^2)(1+\eta)$$
$$N_4^e = -\frac{1}{4}(1-\xi)(1+\eta)(1+\xi-\eta) \qquad N_8^e = \frac{1}{2}(1-\xi)(1-\eta^2)$$

The $\mathbf{B}^e$-matrix is given by

$$\mathbf{B}^e = \nabla\mathbf{N}^e = \begin{bmatrix} \dfrac{\partial}{\partial x} \\ \dfrac{\partial}{\partial y} \end{bmatrix} \mathbf{N}^e = (\mathbf{J}^T)^{-1} \begin{bmatrix} \dfrac{\partial}{\partial \xi} \\ \dfrac{\partial}{\partial \eta} \end{bmatrix} \mathbf{N}^e$$

where $\mathbf{J}$ is the Jacobian matrix

$$\mathbf{J} = \begin{bmatrix} \dfrac{\partial x}{\partial \xi} & \dfrac{\partial x}{\partial \eta} \\ \dfrac{\partial y}{\partial \xi} & \dfrac{\partial y}{\partial \eta} \end{bmatrix}$$

Evaluation of the integrals is done by Gauss integration.

Purpose:

Compute heat flux and temperature gradients in an 8 node isoparametric heat flow element.

Syntax:

[es,et,eci]=flw2i8s(ex,ey,ep,D,ed)

Description:

flw2i8s computes the heat flux vector es and the temperature gradient et (or corresponding quantities) in an 8 node isoparametric heat flow element.

The input variables ex, ey, ep and the matrix D are defined in flw2i8e. The vector ed contains the nodal temperatures $\mathbf{a}^e$ of the element and is obtained by the function extract as

$$\mathsf{ed} = (\mathbf{a}^e)^T = [\ T_1\ \ T_2\ \ T_3\ \ \ldots\ \ T_8\ ]$$

The output variables

$$\mathsf{es} = \bar{\mathbf{q}}^T \qquad = \begin{bmatrix} q_x^1 & q_y^1 \\ q_x^2 & q_y^2 \\ \vdots & \vdots \\ q_x^{n^2} & q_y^{n^2} \end{bmatrix}$$

$$\mathsf{et} = (\bar{\nabla} T)^T = \begin{bmatrix} \dfrac{\partial T^1}{\partial x} & \dfrac{\partial T^1}{\partial y} \\ \dfrac{\partial T^2}{\partial x} & \dfrac{\partial T^2}{\partial y} \\ \vdots & \vdots \\ \dfrac{\partial T^{n^2}}{\partial x} & \dfrac{\partial T^{n^2}}{\partial y} \end{bmatrix} \qquad \mathsf{eci} = \begin{bmatrix} x_1 & y_1 \\ x_2 & y_2 \\ \vdots & \vdots \\ x_{n^2} & y_{n^2} \end{bmatrix}$$

contain the heat flux, the temperature gradient, and the coordinates of the integration points. The index n denotes the number of integration points used within the element, cf. flw2i8e.

Theory:

The temperature gradient and the heat flux are computed according to

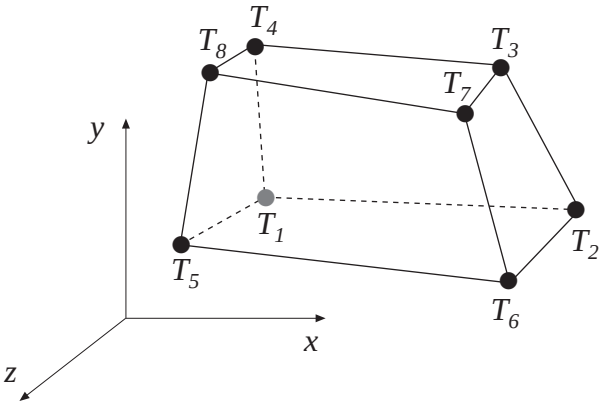
$$\nabla T = \mathbf{B}^e\, \mathbf{a}^e$$

$$\mathbf{q} = -\mathbf{D} \nabla T$$

where the matrices $\mathbf{D}$, $\mathbf{B}^e$, and $\mathbf{a}^e$ are described in flw2i8e, and where the integration points are chosen as evaluation points.

Purpose:

Compute element stiffness matrix for an 8 node isoparametric element.



Syntax:

Ke=flw3i8e(ex,ey,ez,ep,D)
[Ke,fe]=flw3i8e(ex,ey,ez,ep,D,eq)

Description:

flw3i8e provides the element stiffness (conductivity) matrix Ke and the element load vector fe for an 8 node isoparametric heat flow element.

The element nodal coordinates x_1, y_1, z_1 x_2 etc, are supplied to the function by ex, ey and ez. The number of Gauss points n

$$(n \times n \times n) \text{ integration points, } n = 1, 2, 3$$

are supplied to the function by ep and the thermal conductivities (or corresponding quantities) k_{xx}, k_{xy} etc are supplied by D.

$$\mathsf{ex} = [\ x_1\ \ x_2\ \ x_3\ \ \ldots\ \ x_8\ ] \qquad \mathsf{ey} = [\ y_1\ \ y_2\ \ y_3\ \ \ldots\ \ y_8\ ] \qquad \mathsf{ez} = [\ z_1\ \ z_2\ \ z_3\ \ \ldots\ \ z_8\ ]$$

$$\mathsf{ep} = [\ n\ ] \qquad \mathsf{D} = \begin{bmatrix} k_{xx} & k_{xy} & k_{xz} \\ k_{yx} & k_{yy} & k_{yz} \\ k_{zx} & k_{zy} & k_{zz} \end{bmatrix}$$

If the scalar variable eq is given in the function, the element load vector fe is computed, using

$$\mathsf{eq} = [\ Q\ ]$$

where Q is the heat supply per unit volume.

Theory:

The element stiffness matrix $\mathbf{K}^e$ and the element load vector $\mathbf{f}_l^e$, stored in Ke and fe, respectively, are computed according to

$$\mathbf{K}^e = \int_V \mathbf{B}^{eT}\ \mathbf{D}\ \mathbf{B}^e\ dV$$

$$\mathbf{f}_l^e = \int_V \mathbf{N}^{eT}\ Q\ dV$$

with the constitutive matrix $\mathbf{D}$ defined by D.

The evaluation of the integrals for the 3D isoparametric 8 node element is based on a temperature approximation $T(\xi, \eta, \zeta)$, expressed in a local coordinates system in terms of the nodal variables T_1 to T_8 as

$$T(\xi, \eta, \zeta) = \mathbf{N}^e \mathbf{a}^e$$

where

$$\mathbf{N}^e = [\, N_1^e \;\; N_2^e \;\; N_3^e \;\; \ldots \;\; N_8^e \,] \qquad \mathbf{a}^e = [\, T_1 \;\; T_2 \;\; T_3 \;\; \ldots \;\; T_8 \,]^T$$

The element shape functions are given by

$$N_1^e = \frac{1}{8}(1-\xi)(1-\eta)(1-\zeta) \qquad N_2^e = \frac{1}{8}(1+\xi)(1-\eta)(1-\zeta)$$
$$N_3^e = \frac{1}{8}(1+\xi)(1+\eta)(1-\zeta) \qquad N_4^e = \frac{1}{8}(1-\xi)(1+\eta)(1-\zeta)$$
$$N_5^e = \frac{1}{8}(1-\xi)(1-\eta)(1+\zeta) \qquad N_6^e = \frac{1}{8}(1+\xi)(1-\eta)(1+\zeta)$$
$$N_7^e = \frac{1}{8}(1+\xi)(1+\eta)(1+\zeta) \qquad N_8^e = \frac{1}{8}(1-\xi)(1+\eta)(1+\zeta)$$

The $\mathbf{B}^e$-matrix is given by

$$\mathbf{B}^e = \nabla \mathbf{N}^e = \begin{bmatrix} \dfrac{\partial}{\partial x} \\ \dfrac{\partial}{\partial y} \\ \dfrac{\partial}{\partial z} \end{bmatrix} \mathbf{N}^e = (\mathbf{J}^T)^{-1} \begin{bmatrix} \dfrac{\partial}{\partial \xi} \\ \dfrac{\partial}{\partial \eta} \\ \dfrac{\partial}{\partial \zeta} \end{bmatrix} \mathbf{N}^e$$

where $\mathbf{J}$ is the Jacobian matrix

$$\mathbf{J} = \begin{bmatrix} \dfrac{\partial x}{\partial \xi} & \dfrac{\partial x}{\partial \eta} & \dfrac{\partial x}{\partial \zeta} \\ \dfrac{\partial y}{\partial \xi} & \dfrac{\partial y}{\partial \eta} & \dfrac{\partial y}{\partial \zeta} \\ \dfrac{\partial z}{\partial \xi} & \dfrac{\partial z}{\partial \eta} & \dfrac{\partial z}{\partial \zeta} \end{bmatrix}$$

Evaluation of the integrals is done by Gauss integration.

Purpose:

Compute heat flux and temperature gradients in an 8 node isoparametric heat flow element.

Syntax:

[es,et,eci]=flw3i8s(ex,ey,ez,ep,D,ed)

Description:

flw3i8s computes the heat flux vector es and the temperature gradient et (or corresponding quantities) in an 8 node isoparametric heat flow element.

The input variables ex, ey, ez, ep and the matrix D are defined in flw3i8e. The vector ed contains the nodal temperatures $\mathbf{a}^e$ of the element and is obtained by the function extract as

$$\mathsf{ed} = (\mathbf{a}^e)^T = [\ T_1\ \ T_2\ \ T_3\ \ \ldots\ \ T_8\ ]$$

The output variables

$$\mathsf{es} = \bar{\mathbf{q}}^T = \begin{bmatrix} q_x^1 & q_y^1 & q_z^1 \\ q_x^2 & q_y^2 & q_z^2 \\ \vdots & \vdots & \vdots \\ q_x^{n^3} & q_y^{n^3} & q_z^{n^3} \end{bmatrix}$$

$$\mathsf{et} = (\bar{\nabla} T)^T = \begin{bmatrix} \frac{\partial T}{\partial x}^1 & \frac{\partial T}{\partial y}^1 & \frac{\partial T}{\partial z}^1 \\ \frac{\partial T}{\partial x}^2 & \frac{\partial T}{\partial y}^2 & \frac{\partial T}{\partial z}^2 \\ \vdots & \vdots & \vdots \\ \frac{\partial T}{\partial x}^{n^3} & \frac{\partial T}{\partial y}^{n^3} & \frac{\partial T}{\partial z}^{n^3} \end{bmatrix} \qquad \mathsf{eci} = \begin{bmatrix} x_1 & y_1 & z_1 \\ x_2 & y_2 & z_2 \\ \vdots & \vdots & \vdots \\ x_{n^3} & y_{n^3} & z_{n^3} \end{bmatrix}$$

contain the heat flux, the temperature gradient, and the coordinates of the integration points. The index n denotes the number of integration points used within the element, cf. flw3i8e.

Theory:

The temperature gradient and the heat flux are computed according to

$$\nabla T = \mathbf{B}^e\, \mathbf{a}^e$$

$$\mathbf{q} = -\mathbf{D} \nabla T$$

where the matrices $\mathbf{D}$, $\mathbf{B}^e$, and $\mathbf{a}^e$ are described in flw3i8e, and where the integration points are chosen as evaluation points.

2 System functions

2.1 Introduction

The group of system functions comprises functions for the setting up, solving, and elimination of systems of equations. The functions are separated in two groups:

Static system functions
Dynamic system functions

Static system functions concern the linear system of equations

$$\mathbf{K}\mathbf{a} = \mathbf{f}$$

where $\mathbf{K}$ is the global stiffness matrix and $\mathbf{f}$ is the global load vector. Often used static system functions are assem and solveq. The function assem assembles the global stiffness matrix and solveq computes the global displacement vector $\mathbf{a}$ considering the boundary conditions. It should be noted that $\mathbf{K}$, $\mathbf{f}$, and $\mathbf{a}$ also represent analogous quantities in systems others than structural mechanical systems. For example, in a heat flow problem $\mathbf{K}$ represents the conductivity matrix, $\mathbf{f}$ the heat flow, and $\mathbf{a}$ the temperature.

Dynamic system functions are related to different aspects of linear dynamic systems of coupled ordinary differential equations according to

$$\mathbf{C}\,\dot{\mathbf{d}} + \mathbf{K}\,\mathbf{d} = \mathbf{f}$$

for first-order systems and

$$\mathbf{M}\ddot{\mathbf{d}} + \mathbf{C}\dot{\mathbf{d}} + \mathbf{K}\mathbf{d} = \mathbf{f}$$

for second-order systems. First-order systems occur typically in transient heat conduction and second-order systems occur in structural dynamics.

2.2 Static system functions

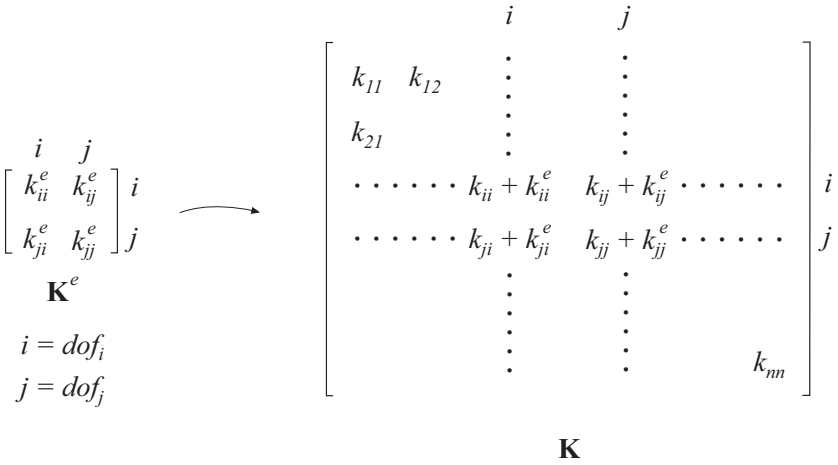
The group of static system functions comprises functions for setting up and solving the global system of equations. It also contains a function for eigenvalue analysis, a function for static condensation, a function for extraction of element displacements from the global displacement vector and a function for extraction of element coordinates.

The following functions are available for static analysis:

Static system functions	
assem	Assemble element matrices
coordxtr	Extract element coordinates from a global coordinate matrix.
eigen	Solve a generalized eigenvalue problem
extract	Extract values from a global vector
red	Reduce the size of a square matrix
solveq	Solve a system of equations
statcon	Perform static condensation
Dynamic system function	
step1	Carry out step-by-step integration in first-order systems

Purpose:

Assemble element matrices.



Syntax:

K=assem(edof,K,Ke)
[K,f]=assem(edof,K,Ke,f,fe)

Description:

assem adds the element stiffness matrix $\mathbf{K}^e$, stored in Ke, to the structure stiffness matrix $\mathbf{K}$, stored in K, according to the topology matrix edof.

The element topology matrix edof is defined as

$$\text{edof} = [el \quad \underbrace{dof_1 \quad dof_2 \quad \ldots \quad dof_{ned}}_{\text{global dof.}}]$$

where the first column contains the element number, and the columns 2 to $(ned + 1)$ contain the corresponding global degrees of freedom (ned = number of element degrees of freedom).

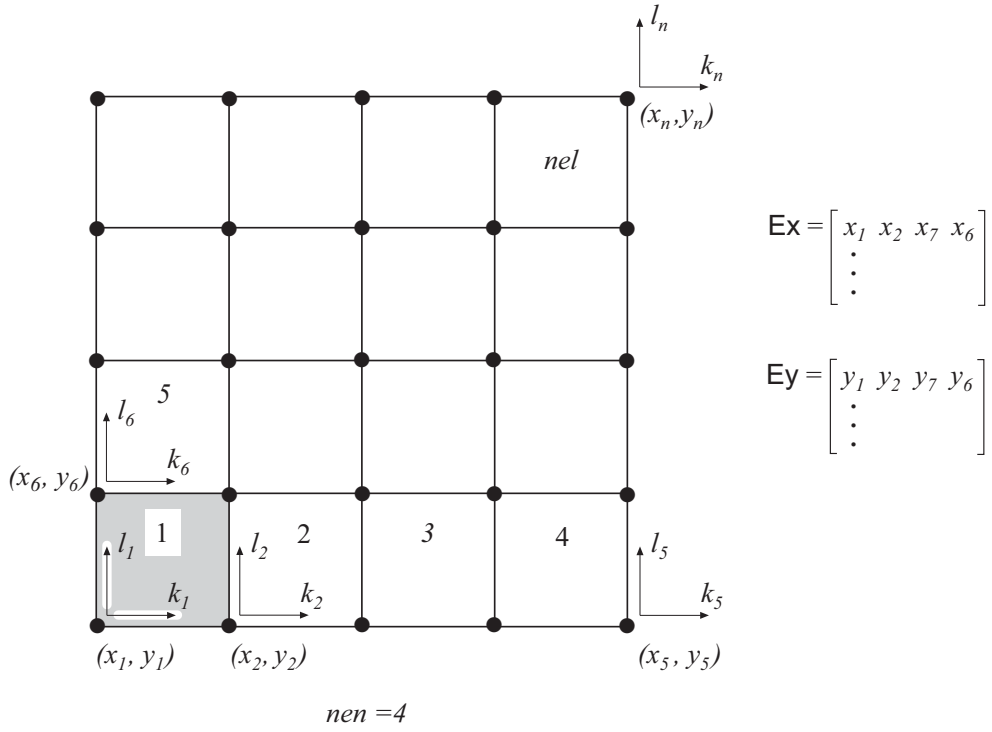
In the case where the matrix $\mathbf{K}^e$ is identical for several elements, assembling of these can be carried out simultaneously. Each row in Edof then represents one element, i.e. nel is the total number of considered elements.

$$\text{Edof} = \left.\begin{bmatrix} el_1 & dof_1 & dof_2 & \ldots & dof_{ned} \\ el_2 & dof_1 & dof_2 & \ldots & dof_{ned} \\ \vdots & \vdots & \vdots & & \vdots \\ el_{nel} & dof_1 & dof_2 & \ldots & dof_{ned} \end{bmatrix}\right\} \textit{one row for each element}$$

If fe and f are given in the function, the element load vector $\mathbf{f}^e$ is also added to the global load vector $\mathbf{f}$.

Purpose:

Extract element coordinates from a global coordinate matrix.



Syntax:

[Ex,Ey,Ez]=coordxtr(Edof,Coord,Dof,nen)

Description:

coordxtr extracts element nodal coordinates from the global coordinate matrix Coord for elements with equal numbers of element nodes and dof's.

Input variables are the element topology matrix Edof, defined in assem, the global coordinate matrix Coord, the global topology matrix Dof, and the number of element nodes nen in each element.

$$\text{Coord} = \begin{bmatrix} x_1 & y_1 & [z_1] \\ x_2 & y_2 & [z_2] \\ x_3 & y_3 & [z_3] \\ \vdots & \vdots & \vdots \\ x_n & y_n & [z_n] \end{bmatrix} \qquad \text{Dof} = \begin{bmatrix} k_1 & l_1 & \ldots & m_1 \\ k_2 & l_2 & \ldots & m_2 \\ k_3 & l_3 & \ldots & m_3 \\ \vdots & \vdots & \ldots & \vdots \\ k_n & l_n & \ldots & m_n \end{bmatrix} \qquad \text{nen} = [\ nen\ ]$$

The nodal coordinates defined in row i of Coord correspond to the degrees of freedom of row i in Dof. The components k_i, l_i and m_i define the degrees of freedom of node i, and n is the number of global nodes for the considered part of the FE-model.

The output variables Ex, Ey, and Ez are matrices defined according to

$$\mathsf{Ex} = \begin{bmatrix} {x_1}^1 & {x_2}^1 & {x_3}^1 & \ldots & {x_{nen}}^1 \\ {x_1}^2 & {x_2}^2 & {x_3}^2 & \ldots & {x_{nen}}^2 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ {x_1}^{nel} & {x_2}^{nel} & {x_3}^{nel} & \ldots & {x_{nen}}^{nel} \end{bmatrix}$$

where row i gives the x-coordinates of the element defined in row i of Edof, and where nel is the number of considered elements.

The element coordinate data extracted by the function coordxtr can be used for plotting purposes and to create input data for the element stiffness functions.

Purpose:

Solve the generalized eigenvalue problem.

Syntax:

L=eigen(K,M)
L=eigen(K,M,b)
[L,X]=eigen(K,M)
[L,X]=eigen(K,M,b)

Description:

eigen solves the eigenvalue problem

$$| \mathsf{K} - \lambda \mathsf{M} | = 0$$

where K and M are square matrices. The eigenvalues λ are stored in the vector L and the corresponding eigenvectors in the matrix X.

If certain rows and columns in matrices K and M are to be eliminated in computing the eigenvalues, b must be given in the function. The rows (and columns) that are to be eliminated are described in the vector b defined as

$$\mathsf{b} = \begin{bmatrix} dof_1 \\ dof_2 \\ \vdots \\ dof_{nb} \end{bmatrix}$$

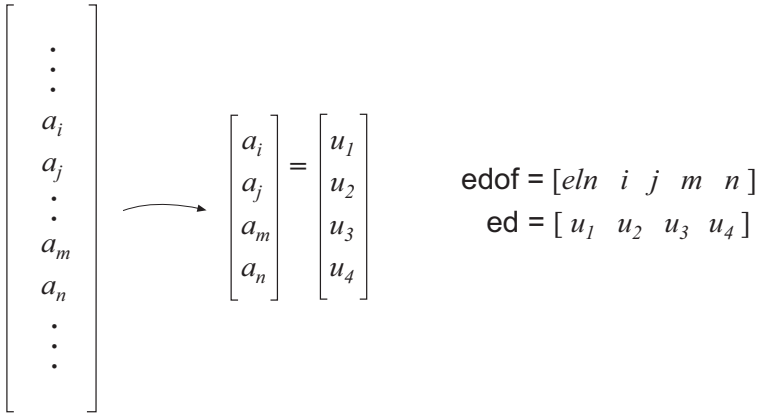
The computed eigenvalues are given in order ranging from the smallest to the largest. The eigenvectors are normalized in order that

$$\mathsf{X}^T \mathsf{M} \mathsf{X} = \mathbf{I}$$

where $\mathbf{I}$ is the identity matrix.

Purpose:

Extract element nodal quantities from a global solution vector.



Syntax:

ed=extract(edof,a)

Description:

extract extracts element displacements or corresponding quantities $\mathbf{a}^e$ from the global solution vector $\mathbf{a}$, stored in a.

Input variables are the element topology matrix edof, defined in assem, and the global solution vector a.

The output variable

$$\mathsf{ed} = (\mathbf{a}^e)^T$$

contains the element displacement vector.

If Edof contains more than one element, Ed will be a matrix

$$\mathsf{Ed} = \begin{bmatrix} (\mathbf{a}^e)_1^T \\ (\mathbf{a}^e)_2^T \\ \vdots \\ (\mathbf{a}^e)_{nel}^T \end{bmatrix}$$

where row i gives the element displacements for the element defined in row i of Edof, and nel is the total number of considered elements.

Example:

For the two dimensional beam element, the extract function will extract six nodal displacements for each element given in Edof, and create a matrix Ed of size ($nel \times 6$).

$$\mathsf{Ed} = \begin{bmatrix} u_1 & u_2 & u_3 & u_4 & u_5 & u_6 \\ u_1 & u_2 & u_3 & u_4 & u_5 & u_6 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ u_1 & u_2 & u_3 & u_4 & u_5 & u_6 \end{bmatrix}$$

Purpose:

Reduce the size of a square matrix by omitting rows and columns.

Syntax:

B=red(A,b)

Description:

B=red(A,b) reduces the square matrix A to a smaller matrix B by omitting rows and columns of A. The indices for rows and columns to be omitted are specified by the column vector b.

Examples:

Assume that the matrix A is defined as

$$\mathsf{A} = \begin{bmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 9 & 10 & 11 & 12 \\ 13 & 14 & 15 & 16 \end{bmatrix}$$

and b as

$$\mathsf{b} = \begin{bmatrix} 2 \\ 4 \end{bmatrix}$$

The statement B=red(A,b) results in the matrix

$$\mathsf{B} = \begin{bmatrix} 1 & 3 \\ 9 & 11 \end{bmatrix}$$

Purpose:

Solve equation system.

Syntax:

a=solveq(K,f)
a=solveq(K,f,bc)
[a,r]=solveq(K,f,bc)

Description:

solveq solves the equation system

$$\mathsf{K}\,\mathsf{a} = \mathsf{f}$$

where K is a matrix and a and f are vectors.

The matrix K and the vector f must be predefined. The solution of the system of equations is stored in a vector a which is created by the function.

If some values of a are to be prescribed, the row number and the corresponding values are given in the boundary condition matrix

$$\mathsf{bc} = \begin{bmatrix} dof_1 & u_1 \\ dof_2 & u_2 \\ \vdots & \vdots \\ dof_{nbc} & u_{nbc} \end{bmatrix}$$

where the first column contains the row numbers and the second column the corresponding prescribed values.

If r is given in the function, support forces are computed according to

$$\mathsf{r} = \mathsf{K}\,\mathsf{a} - \mathsf{f}$$

Purpose:

Reduce system of equations by static condensation.

Syntax:

[K1,f1]=statcon(K,f,b)

Description:

statcon reduces a system of equations

$$\mathsf{K}\,\mathsf{a} = \mathsf{f}$$

by static condensation.

The degrees of freedom to be eliminated are supplied to the function by the vector

$$\mathsf{b} = \begin{bmatrix} dof_1 \\ dof_2 \\ \vdots \\ dof_{nb} \end{bmatrix}$$

where each row in b contains one degree of freedom to be eliminated.

The elimination gives the reduced system of equations

$$\mathsf{K}_1\,\mathsf{a}_1 = \mathsf{f}_1$$

where K_1 and f_1 are stored in K1 and f1 respectively.

Purpose:

Compute the dynamic solution to a set of first order differential equations.

Syntax:

[D,V]=step1(K,C,d0,ip,snap,f,bc)

Description:

step1 computes at equal time steps the solution to a set of first order differential equations of the form

$$\begin{aligned} \mathbf{C}\dot{\mathbf{d}} + \mathbf{K}\mathbf{d} &= \mathbf{f}(x,t), \\ \mathbf{d}(0) &= \mathbf{d}_0. \end{aligned}$$

The command solves transient field problems. In the case of heat conduction, K and C represent the $n \times n$ conductivity and capacity matrices, respectively. d is the temperature and v=$\dot{d}$ is the time derivative of the temperature.

The initial conditions are given by the vector d0 containing initial values of d. The time integration procedure is governed by the parameters given in the vector ip defined as

$$\mathsf{ip} = [dt\ T\ \alpha]$$

where dt specifies the length of the time increment in the time stepping scheme, T total time and α the time integration constant; see [1]. The parameter *snap* denotes at which times snapshots of the solution will be stored in D and V. The following table lists frequently used values of α:

$\alpha = 0$	Forward difference; forward Euler,
$\alpha = \frac{1}{2}$	Trapezoidal rule; Crank-Nicholson,
$\alpha = 1$	Backward difference; backward Euler.

The matrix f contains the time-dependent load vectors. If no external loads are active, the matrix corresponding to f should be replaced by []. The matrix bc contains the time-dependent prescribed values of the field variable d. If no field variables are prescribed the matrix corresponding to bc should be replaced by []. Matrix f is organized in the following manner:

$$\mathsf{f} = \begin{bmatrix} \text{time history of the load at } dof_1 \\ \text{time history of the load at } dof_2 \\ \vdots \\ \text{time history of the load at } dof_n \end{bmatrix}.$$

The dimension of f is

$$(\text{number of degrees-of-freedom}) \times (\text{number of timesteps} + 1).$$

The matrix bc is organized in the following manner:

$$\mathsf{bc} = \begin{bmatrix} dof_1 & \text{time history of the field variable} \\ dof_2 & \text{time history of the field variable} \\ \vdots & \vdots \\ dof_{m_2} & \text{time history of the field variable} \end{bmatrix}.$$

The dimension of bc is

(number of dofs with prescribed field values) $\times$ (number of timesteps + 2).

The time history functions can be generated using the command gfunc. If all the values of the time histories of f or bc are kept constant, these values need to be stated only once. In this case the number of columns in f is one and in bc two.

It is highly recommended to define f as sparse (a MATLAB built-in function). In most cases only a few degrees-of-freedom are affected by the exterior load, and hence the matrix contains only few non-zero entries.

The computed snapshots of $\mathbf{d}$ and $\dot{\mathbf{d}}$ are stored in D and V, respectively, one column for each requested time according to snap. The dimension of D and V is $ndofs \times$ (number of snapshots + 1).

3 Statements and macros

Statements describe algorithmic actions that can be executed. There are two different types of control statements, conditional and repetitive. The first group defines conditional jumps whereas the latter one defines repetition until a conditional statement is fulfilled. Macros are used to define new functions to the MATLAB or CALFEM structure, or to store a sequence of statements in an .m-file.

Control statements	
if	Conditional jump
for	Initiate a loop
while	Define a conditional loop

Macros	
function	Define a new function
script	Store a sequence of statements

4 Graphics functions

The group of graphics functions comprises functions for element based graphics. Mesh plots, displacements, section forces, flows, iso lines and principal stresses can be displayed. The functions are divided into two dimensional, and general graphics functions.

Two dimensional graphics functions	
eldraw2	Draw undeformed finite element mesh
eldisp2	Draw deformed finite element mesh
elflux2	Plot flux vectors
eliso2	Draw isolines for nodal quantities
elprinc2	Plot principal stresses
scalfact2	Determine scale factor
scalgraph2	Draw graphic scale

General graphics functions in MATLAB	
plot	Plot lines and points in 2D space
fill	Draw filled 2D polygons
axis	Axis scaling and appearance
clf	Clear current figure
figure	Create figures
grid	Grid lines
hold	Hold current graph
print	Print graph or save graph to file
text	Add text to current plot
title	Titles for 2D and 3D plots
xlabel, ylabel, zlabel	Axis labels for 2D and 3D plots

Purpose:

Draw the undeformed mesh for a two dimensional structure.

Syntax:

eldraw2(Ex,Ey)
eldraw2(Ex,Ey,plotpar)
eldraw2(Ex,Ey,plotpar,elnum)

Description:

eldraw2 displays the undeformed mesh for a two dimensional structure.

Input variables are the coordinate matrices Ex and Ey formed by the function coordxtr.

The variable plotpar sets plot parameters for linetype, linecolor and node marker.

plotpar = [*linetype* *linecolor* *nodemark*]

linetype =	1	solid line	*linecolor* =	1	black
	2	dashed line		2	blue
	3	dotted line		3	magenta
				4	red

nodemark =	1	circle
	2	star
	0	no mark

Default is solid black lines with circles at nodes.

Element numbers can be displayed at the center of the element if a column vector elnum with the element numbers is supplied. This column vector can be derived from the element topology matrix Edof,

elnum=Edof(:,1)

i.e. the first column of the topology matrix.

Limitations:

Supported elements are bar, beam, triangular three node, and quadrilateral four node elements.

Purpose:

Draw the deformed mesh for a two dimensional structure.

Syntax:

[sfac]=eldisp2(Ex,Ey,Ed)
[sfac]=eldisp2(Ex,Ey,Ed,plotpar)
eldisp2(Ex,Ey,Ed,plotpar,sfac)

Description:

eldisp2 displays the deformed mesh for a two dimensional structure.

Input variables are the coordinate matrices Ex and Ey formed by the function coordxtr, and the element displacements Ed formed by the function extract.

The variable plotpar sets plot parameters for linetype, linecolor and node marker.

plotpar=[*linetype* *linecolor* *nodemark*]

linetype =	1	solid line	*linecolor* =	1	black
	2	dashed line		2	blue
	3	dotted line		3	magenta
				4	red

nodemark =	1	circle
	2	star
	0	no mark

Default is dashed black lines with circles at nodes.

The scale factor sfac is a scalar that the element displacements are multiplied with to get a suitable geometrical representation. The scale factor is set automatically if it is omitted in the input list.

Limitations:

Supported elements are bar, beam, triangular three node, and quadrilateral four node elements.

Purpose:

Draw element flow arrows for two dimensional elements.

Syntax:

[sfac]=elflux2(Ex,Ey,Es)
[sfac]=elflux2(Ex,Ey,Es,plotpar)
elflux2(Ex,Ey,Es,plotpar,sfac)

Description:

elflux2 displays element heat flux vectors (or corresponding quantities) for a number of elements of the same type. The flux vectors are displayed as arrows at the element centroids. Note that only the flux vectors are displayed. To display the element mesh, use eldraw2.

Input variables are the coordinate matrices Ex and Ey, and the element flux matrix Es defined in flw2ts or flw2qs.

The variable plotpar sets plot parameters for the flux arrows.

plotpar=[*arrowtype* *arrowcolor*]

arrowtype =	1	solid	*arrowcolor* =	1	black
	2	dashed		2	blue
	3	dotted		3	magenta
				4	red

Default, if plotpar is omitted, is solid black arrows.

The scale factor sfac is a scalar that the values are multipied with to get a suitable arrow size in relation to the element size. The scale factor is set automatically if it is omitted in the input list.

Limitations:

Supported elements are triangular 3 node and quadrilateral 4 node elements.

Purpose:

Display element iso lines for two dimensional elements.

Syntax:

eliso2(Ex,Ey,Ed,isov)
eliso2(Ex,Ey,Ed,isov,plotpar)

Description:

eliso2 displays element iso lines for a number of elements of the same type. Note that only the iso lines are displayed. To display the element mesh, use eldraw2.

Input variables are the coordinate matrices Ex and Ey formed by the function coordxtr, and the element nodal quantities (e.g displacement or energy potential) matrix Ed defined in extract.

If isov is a scalar it determines the number of iso lines to be displayed. If isov is a vector it determines the values of the iso lines to be displayed (number of iso lines equal to length of vector isov).

isov = [$isolines$]
isov = [$isovalue(1)$ ... $isovalue(n)$]

The variable plotpar sets plot parameters for the iso lines.

plotpar=[$linetype$ $linecolor$ $textfcn$]

$arrowtype$ =	1	solid	$arrowcolor$ =	1	black
	2	dashed		2	blue
	3	dotted		3	magenta
				4	red

$textfcn$ =	0	the iso values of the lines will not be printed
	1	the iso values of the lines will be printed at the iso lines
	2	the iso values of the lines will be printed where the cursor indicates

Default is solid, black lines and no iso values printed.

Limitations:

Supported elements are triangular 3 node and quadrilateral 4 node elements.

Purpose:

Determine scale factor for drawing computational results.

Syntax:

[sfac]=scalfact2(ex,ey,ed)
[sfac]=scalfact2(ex,ey,ed,rat)

Description:

scalfact2 determines a scale factor sfac for drawing computational results, such as displacements, section forces or flux.

Input variables are the coordinate matrices ex and ey and the matrix ed containing the quantity to be displayed. The scalar rat defines the ratio between the geometric representation of the largest quantity to be displayed and the element size. If rat is not specified, 0.2 is used.

Purpose:

Draw a Graphic scale.

Syntax:

scalgraph2(sfac,magnitude)
scalgraph2(sfac,magnitude,plotpar)

Description:

scalgraph2 draws a graphic scale to visualise the magnitude of displayed computational results. The input variable sfac is a scale factor determined by the function scalfact2 and the variable

magnitude = [S x y]

specifies the value corresponding the length of the graphic scale S, and (x, y) are the coordinates of the starting point. If no coordinates are given the starting point will be (0,-0.5).

The variable plotpar sets the the graphic scale colour.

plotpar=[*colour*]

colour = 1 black
2 blue
3 magenta
4 red

5 User's Manual, examples

5.1 Introduction

This set of examples is defined with the ambition to serve as a User's Manual. The examples are written as .m-files (script files) and supplied together with the CALFEM functions.

The User's Manual examples are separated into three groups:

Static analysis
Dynamic analysis
Nonlinear analysis

In this manual, only static examples are shown. The static linear examples illustrate finite element analysis of different structures loaded by stationary loads.

5.2 Static analysis

This section illustrates some linear static finite element calculations. The examples deal with structural problems as well as field problems such as heat conduction.

Static analysis	
exs1	Linear spring system
exs2	One-dimensional heat flow
exs3	Plane truss
exs4	Plane truss analysed using loops
exs8	Two dimensional diffusion

The introductory example exs1 illustrates the basic steps in the finite element method for a simple structure of one-dimensional linear springs. The linear spring or analogy element is also used in example exs2 to solve a problem of heat conduction in a wall. A plane truss consisting of three bars is analysed in exs3. In example exs4 a plane truss consisting of ten bars is analysed using loops. First the analysis is performed by defining coordinate data for each element directly, and then it is shown how this data can be obtained from global coordinate data. A simply supported beam is analysed in example exs5. Element forces and the support forces are calculated. A two dimensional plane frame is analysed in example exs6. A structure built up from both beams and bars is analysed in example exs7. Graphics facilities are also explained in examples exs6, exs7, and exs8.

Note: The examples listed above are supplied as .m-files under the directory examples. The example files are named according to the table.

Purpose:

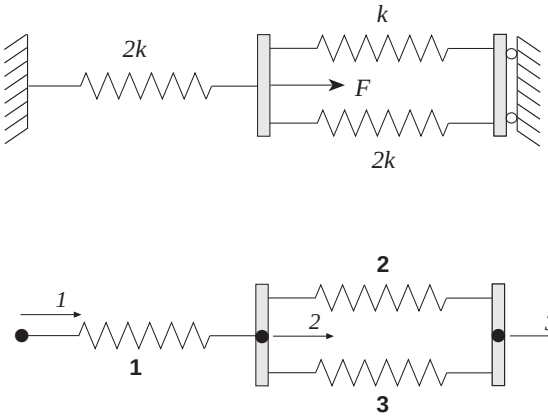
Show the basic steps in a finite element calculation.

Description:

The general procedure in linear finite element calculations is carried out for a simple structure. The steps are

- define the model
- generate element matrices
- assemble element matrices into the global system of equations
- solve the global system of equations
- evaluate element forces

Consider the system of three linear elastic springs, and the corresponding finite element model. The system of springs is fixed in its ends and loaded by a single load F.



The computation is initialized by defining the topology matrix Edof, containing element numbers and global element degrees of freedom,

```
>> Edof=[1  1 2;
         2  2 3;
         3  2 3];
```

the global stiffness matrix K (3×3) of zeros,

```
>> K=zeros(3,3)

K =
     0     0     0
     0     0     0
     0     0     0
```

and the load vector f (3×1) with the load $F = 100$ in position 2.

```
>> f=zeros(3,1);  f(2)=100

f =
      0
    100
      0
```

Element stiffness matrices are generated by the function spring1e. The element property ep for the springs contains the spring stiffnesses k and $2k$ respectively, where $k = 1500$.

```
>> k=1500;  ep1=k;  ep2=2*k;
>> Ke1=spring1e(ep1)

Ke1 =
        1500       -1500
       -1500        1500

>> Ke2=spring1e(ep2)

Ke2 =
        3000       -3000
       -3000        3000
```

The element stiffness matrices are assembled into the global stiffness matrix K according to the topology.

```
>> K=assem(Edof(1,:),K,Ke2)

K =
        3000       -3000           0
       -3000        3000           0
           0           0           0

>> K=assem(Edof(2,:),K,Ke1)

K =
        3000       -3000           0
       -3000        4500       -1500
           0       -1500        1500
```

```
>> K=assem(Edof(3,:),K,Ke2)

K =
        3000       -3000           0
       -3000        7500       -4500
           0       -4500        4500
```

The global system of equations is solved considering the boundary conditions given in bc.

```
>> bc= [1 0; 3 0];
>> [a,r]=solveq(K,f,bc)

a =
         0
    0.0133
         0

r =

  -40.0000
         0
  -60.0000
```

Element forces are evaluated from the element displacements. These are obtained from the global displacements a using the function extract.

```
>> ed1=extract(Edof(1,:),a)

ed1 =
      0      0.0133

>> ed2=extract(Edof(2,:),a)

ed2 =
      0.0133      0

>> ed3=extract(Edof(3,:),a)

ed3 =
      0.0133      0
```

The spring forces are evaluated using the function spring1s.

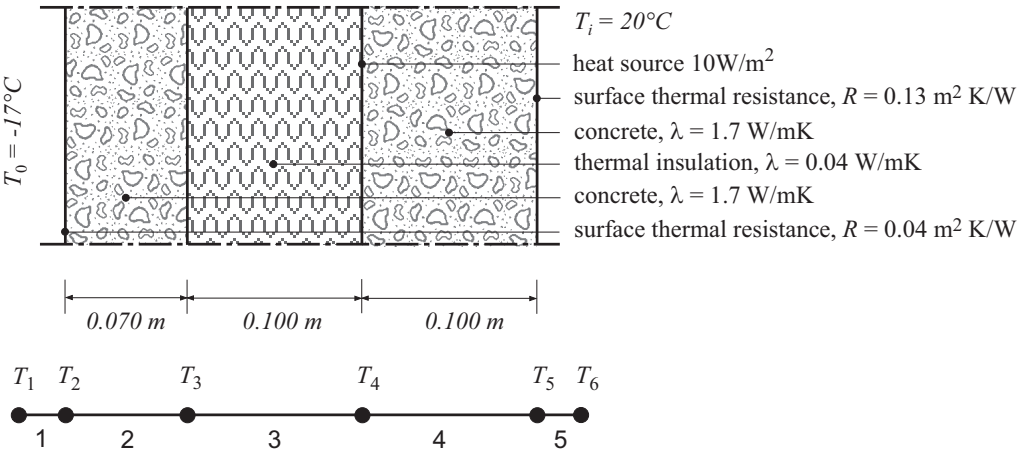
```
>> es1=spring1s(ep2,ed1)

es1 =
       40

>> es2=spring1s(ep1,ed2)

es2 =
      -20

>> es3=spring1s(ep2,ed3)

es3 =
      -40
```

Purpose:

Analysis of one-dimensional heat flow.

Description:

Consider a wall built up of concrete and thermal insulation. The outdoor temperature is $-17°$C and the temperature inside is 20°C. At the inside of the theral insulation there is a heat source yielding 10 W/m^2.



The wall is subdivided into five elements and the one-dimensional spring (analogy) element spring1e is used. Equivalent spring stiffnesses are $k_i = \lambda A/L$ for thermal conductivity and $k_i = A/R$ for thermal surface resistance. Corresponding spring stiffnesses per m^2 of the wall are:

$$
\begin{array}{lrcrl}
k_1 = & 1/0.04 & = & 25.0 & \text{W/K} \\
k_2 = & 1.7/0.070 & = & 24.3 & \text{W/K} \\
k_3 = & 0.040/0.100 & = & 0.4 & \text{W/K} \\
k_4 = & 1.7/0.100 & = & 17.0 & \text{W/K} \\
k_5 = & 1/0.13 & = & 7.7 & \text{W/K}
\end{array}
$$

A global system matrix K and a heat flow vector f are defined. The heat source inside the wall is considered by setting $f_4 = 10$. The element matrices Ke are computed using spring1e, and the function assem assembles the global stiffness matrix.

The system of equations is solved using solveq with considerations to the boundary conditions in bc. The prescribed temperatures are $T_1 = -17°$C and $T_6 = 20°$C.

```
>> Edof=[1  1 2
         2  2 3;
         3  3 4;
         4  4 5;
         5  5 6];
```

```
>> K=zeros(6);
>> f=zeros(6,1);  f(4)=10

f =

     0
     0
     0
    10
     0
     0

>> ep1=[25];  ep2=[24.3];
>> ep3=[0.4];  ep4=[17];
>> ep5=[7.7];

>> Ke1=spring1e(ep1);       Ke2=spring1e(ep2);
>> Ke3=spring1e(ep3);       Ke4=spring1e(ep4);
>> Ke5=spring1e(ep5);

>> K=assem(Edof(1,:),K,Ke1);   K=assem(Edof(2,:),K,Ke2);
>> K=assem(Edof(3,:),K,Ke3);   K=assem(Edof(4,:),K,Ke4);
>> K=assem(Edof(5,:),K,Ke5);

>> bc=[1 -17; 6 20];

>> [a,r]=solveq(K,f,bc)

a =

  -17.0000
  -16.4384
  -15.8607
   19.2378
   19.4754
   20.0000

r =

  -14.0394
    0.0000
   -0.0000
         0
    0.0000
    4.0394
```

The temperature values T_i in the node points are given in the vector a and the boundary flows in the vector r.

After solving the system of equations, the heat flow through the wall is computed using extract and spring1s.

```
>> ed1=extract(Edof(1,:),a);
>> ed2=extract(Edof(2,:),a);
>> ed3=extract(Edof(3,:),a);
>> ed4=extract(Edof(4,:),a);
>> ed5=extract(Edof(5,:),a);

>> q1=spring1s(ep1,ed1)

q1 =

   14.0394

>> q2=spring1s(ep2,ed2)

q2 =

   14.0394

>> q3=spring1s(ep3,ed3)

q3 =

   14.0394

>> q4=spring1s(ep4,ed4)

q4 =

    4.0394

>> q5=spring1s(ep5,ed5)

q5 =

    4.0394
```

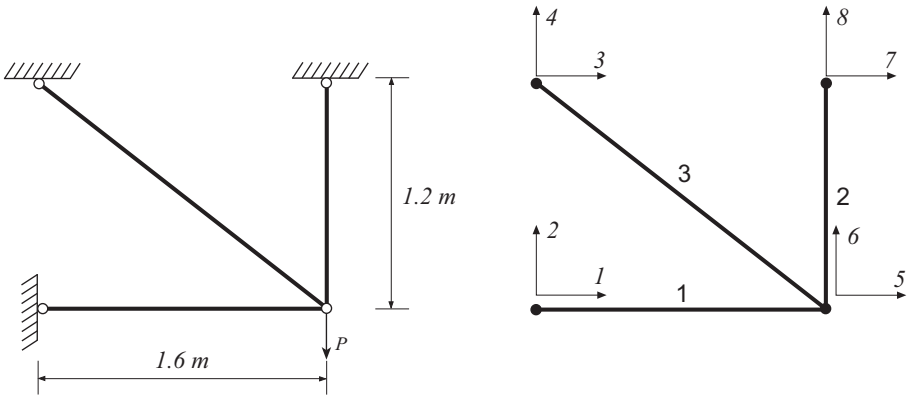
The heat flow through the wall is $q = 14.0$ W/m^2 in the part of the wall to the left of the heat source, and $q = 4.0$ W/m^2 in the part to the right of the heat source.

Purpose:

Analysis of a plane truss.

Description:

Consider a plane truss consisting of tree bars with the properties $E = 200$ GPa, $A_1 = 6.0 \cdot 10^{-4}$ m^2, $A_2 = 3.0 \cdot 10^{-4}$ m^2 and $A_3 = 10.0 \cdot 10^{-4}$ m^2, and loaded by a single force $P = 80$ kN. The corresponding finite element model consists of three elements and eight degrees of freedom.



The topology is defined by the matrix

```
>> Edof=[1   1  2  5  6;
         2   5  6  7  8;
         3   3  4  5  6];
```

The stiffness matrix K and the load vector f, are defined by

```
>> K=zeros(8);
   f=zeros(8,1); f(6)=-80e3;
```

The element property vectors ep1, ep2 and ep3 are defined by

```
>> E=2.0e11;
>> A1=6.0e-4;      A2=3.0e-4;      A3=10.0e-4;
>> ep1=[E A1];     ep2=[E A2];     ep3=[E A3];
```

and the element coordinate vectors ex1, ex2, ex3, ey1, ey2 and ey3 by

```
>> ex1=[0 1.6];     ex2=[1.6 1.6];     ex3=[0 1.6];
>> ey1=[0 0];       ey2=[0 1.2];       ey3=[1.2 0];
```

The element stiffness matrices Ke1, Ke2 and Ke3 are computed using bar2e.

```
>> Ke1=bar2e(ex1,ey1,ep1)

Ke1 =

  1.0e+007 *

    7.5000         0   -7.5000         0
         0         0         0         0
   -7.5000         0    7.5000         0
         0         0         0         0

>> Ke2=bar2e(ex2,ey2,ep2)

Ke2 =

  1.0e+007 *

         0         0         0         0
         0    5.0000         0   -5.0000
         0         0         0         0
         0   -5.0000         0    5.0000

>> Ke3=bar2e(ex3,ey3,ep3)

Ke3 =

  1.0e+007 *

    6.4000   -4.8000   -6.4000    4.8000
   -4.8000    3.6000    4.8000   -3.6000
   -6.4000    4.8000    6.4000   -4.8000
    4.8000   -3.6000   -4.8000    3.6000
```

Based on the topology information, the global stiffness matrix can be generated by assembling the element stiffness matrices

```
>> K=assem(Edof(1,:),K,Ke1);
>> K=assem(Edof(2,:),K,Ke2);
>> K=assem(Edof(3,:),K,Ke3)
```

```
K =

  1.0e+008 *

  Columns 1 through 7

    0.7500         0         0         0   -0.7500         0         0
         0         0         0         0         0         0         0
         0         0    0.6400   -0.4800   -0.6400    0.4800         0
         0         0   -0.4800    0.3600    0.4800   -0.3600         0
   -0.7500         0   -0.6400    0.4800    1.3900   -0.4800         0
         0         0    0.4800   -0.3600   -0.4800    0.8600         0
         0         0         0         0         0         0         0
         0         0         0         0         0   -0.5000         0

  Column 8

         0
         0
         0
         0
         0
   -0.5000
         0
    0.5000
```

Considering the prescribed displacements in bc, the system of equations is solved using the function solveq, yielding displacements a and support forces r.

```
>> bc= [1 0;2 0;3 0;4 0;7 0;8 0];
>> [a,r]=solveq(K,f,bc)

a =

  1.0e-002 *

         0
         0
         0
         0
   -0.0398
   -0.1152
         0
         0
```

```
r =

  1.0e+004 *

    2.9845
         0
   -2.9845
    2.2383
    0.0000
    0.0000
         0
    5.7617
```

The vertical displacement at the point of loading is 1.15 mm. The section forces es1, es2 and es3 are calculated using bar2s from element displacements ed1, ed2 and ed3 obtained using extract.

```
>> ed1=extract(Edof(1,:),a);
>> N1=bar2s(ex1,ey1,ep1,ed1)

N1 =

-2.9845e+004

>> ed2=extract(Edof(2,:),a);
>> N2=bar2s(ex2,ey2,ep2,ed2)

N2 =

  5.7617e+004

>> ed3=extract(Edof(3,:),a);
>> N3=bar2s(ex3,ey3,ep3,ed3)

N3 =

  3.7306e+004
```

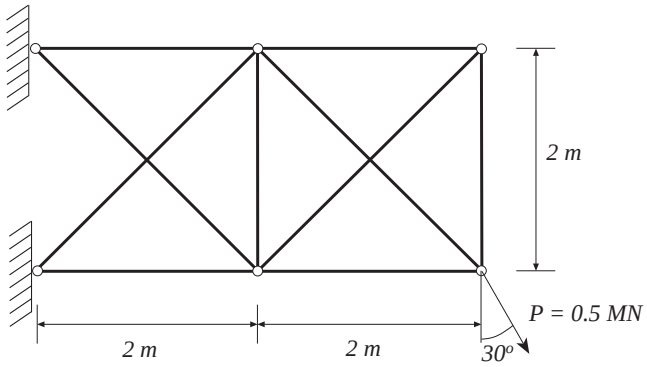
i.e., the normal forces are $N_1 = -29.84$ kN, $N_2 = 57.62$ kN and $N_3 = 37.31$ kN.

Purpose:

Analysis of a plane truss.

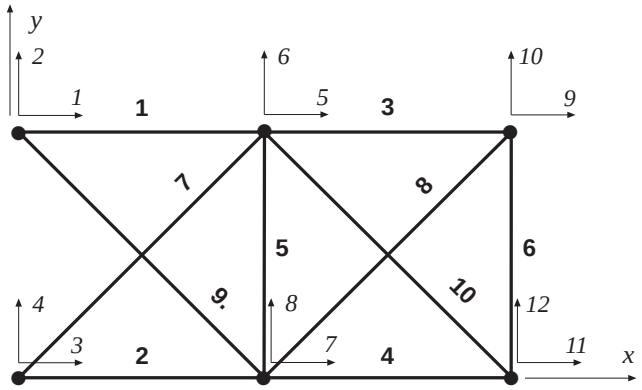
Description:

Consider a plane truss, loaded by a single force $P = 0.5$ MN.



$A = 25.0 \cdot 10^{-4}$ m^2
$E = 2.10 \cdot 10^{5}$ MPa

The corresponding finite element model consists of ten elements and twelve degrees of freedom.



The topology is defined by the matrix

```
>> Edof=[1   1  2  5  6;
         2   3  4  7  8;
         3   5  6  9 10;
         4   7  8 11 12;
         5   7  8  5  6;
         6  11 12  9 10;
         7   3  4  5  6;
         8   7  8  9 10;
         9   1  2  7  8;
        10   5  6 11 12];
```

A global stiffness matrix K and a load vector f are defined. The load P is divided into x and y components and inserted in the load vector f.

```
>> K=zeros(12);
>> f=zeros(12,1);  f(11)=0.5e6*sin(pi/6);  f(12)=-0.5e6*cos(pi/6);
```

The element matrices Ke are computed by the function bar2e. These matrices are then assembled in the global stiffness matrix using the function assem.

```
>> A=25.0e-4;    E=2.1e11;   ep=[E A];

>> Ex=[0 2;
       0 2;
       2 4;
       2 4;
       2 2;
       4 4;
       0 2;
       2 4;
       0 2;
       2 4];

>> Ey=[2 2;
       0 0;
       2 2;
       0 0;
       0 2;
       0 2;
       0 2;
       0 2;
       2 0;
       2 0];
```

All the element matrices are computed and assembled in the loop

```
>> for i=1:10
      Ke=bar2e(Ex(i,:),Ey(i,:),ep);
      K=assem(Edof(i,:),K,Ke);
   end;
```

The displacements in a and the support forces in r are computed by solving the system of equations considering the boundary conditions in bc.

```
>> bc=[1 0;2 0;3 0;4 0];
>> [a,r]=solveq(K,f,bc)
```

```
a =

         0
         0
         0
         0
    0.0024
   -0.0045
   -0.0016
   -0.0042
    0.0030
   -0.0107
   -0.0017
   -0.0113

r =

  1.0e+005 *

   -8.6603
    2.4009
    6.1603
    1.9293
    0.0000
   -0.0000
   -0.0000
   -0.0000
    0.0000
    0.0000
    0.0000
    0.0000
```

The displacement at the point of loading is $-1.7 \cdot 10^{-3}$ m in the x-direction and $-11.3 \cdot 10^{-3}$ m in the y-direction. At the upper support the horizontal force is -0.866 MN and the vertical 0.240 MN. At the lower support the forces are 0.616 MN and 0.193 MN, respectively.

Normal forces are evaluated from element displacements. These are obtained from the global displacements a using the function extract. The normal forces are evaluated using the function bar2s.

```
ed=extract(Edof,a);
```

```
>> for i=1:10
      N(i,:)=bar2s(Ex(i,:),Ey(i,:),ep,ed(i,:));
   end
```

The obtained normal forces are

```
>> N

N =

  1.0e+005 *

    6.2594
   -4.2310
    1.7064
   -0.1237
   -0.6945
    1.7064
   -2.7284
   -2.4132
    3.3953
    3.7105
```

The largest normal force $N = 0.626$ MN is obtained in element 1 and is equivalent to a normal stress $\sigma = 250$ MPa.

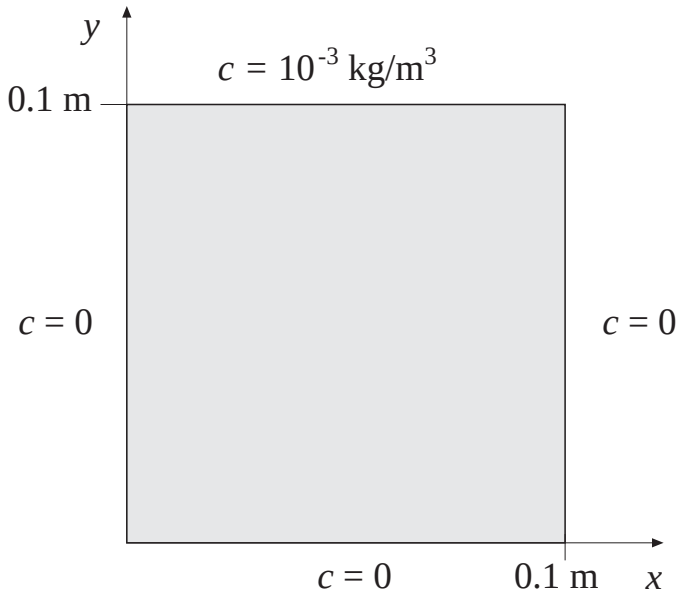
To reduce the quantity of input data, the element coordinate matrices Ex and Ey can alternatively be created from a global coordinate matrix Coord and a global topology matrix Coord using the function coordxtr, i.e.

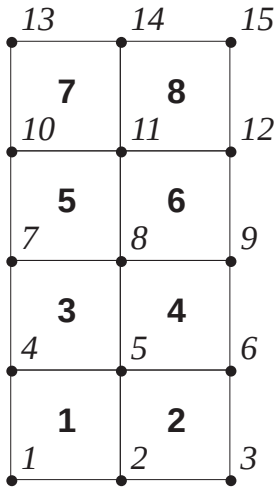
```
>> Coord=[0 2;
          0 0;
          2 2;
          2 0;
          4 2;
          4 0];

>> Dof=[ 1  2;
         3  4;
         5  6;
         7  8;
         9 10;
        11 12];

>> [ex,ey]=coordxtr(Edof,Coord,Dof,2);
```

Purpose:

Analysis of two dimensional diffusion.

Description:





Description:

Consider a filter paper of square shape. Three sides are in contact with pure water and the fourth side is in contact with a solution of concentration $c = 1.0 \cdot 10^{-3}$ kg/m^3. The length of each side is 0.100 m. Using symmetry, only half of the paper has to be analyzed. The paper and the corresponding finite element mesh are shown. The following boundary conditions are applied

$c(0, y) = c(x, 0) = c(0.1, y) = 0$
$c(x, 0.1) = 10^{-3}$

The element topology is defined by the topology matrix

```
>> Edof=[1   1  2  5  4
         2   2  3  6  5
         3   4  5  8  7
         4   5  6  9  8
         5   7  8 11 10
         6   8  9 12 11
         7  10 11 14 13
         8  11 12 15 14];
```

The system matrices, i.e. the stiffness matrix K and the load vector f, are defined by

```
>> K=zeros(15);   f=zeros(15,1);
```

Because of the same geometry, orientation, and constitutive matrix for all elements, only one element stiffness matrix Ke has to be computed. This is done by the function flw2qe.

```
>> ep=1;     D=[1 0; 0 1];

>> ex=[0 0.025 0.025 0];     ey=[0 0 0.025 0.025];

>> Ke=flw2qe(ex,ey,ep,D)

>> Ke =

    0.7500   -0.2500   -0.2500   -0.2500
   -0.2500    0.7500   -0.2500   -0.2500
   -0.2500   -0.2500    0.7500   -0.2500
   -0.2500   -0.2500   -0.2500    0.7500
```

Based on the topology information, the global stiffness matrix is generated by assembling this element stiffness matrix Ke in the global stiffness matrix K

```
>> K=assem(Edof,K,Ke);
```

Finally, the solution is calculated by defining the boundary conditions bc and solving the system of equations. The boundary condition at dof 13 is set to $0.5{\cdot}10^{-3}$ as an average of the concentrations at the neighbouring boundaries. Concentrations a and unknown boundary flows r are computed by the function solveq.

```
>> bc=[1 0;2 0;3 0;4 0;7 0;10 0;13 0.5e-3;14 1e-3;15 1e-3];

>> [a,r]=solveq(K,f,bc);
```

The element flows q are calculated from element concentration Ed

```
>> Ed=extract(Edof,a);

>> for i=1:8
     Es=flw2qs(ex,ey,ep,D,Ed(i,:));
   end
```

Results

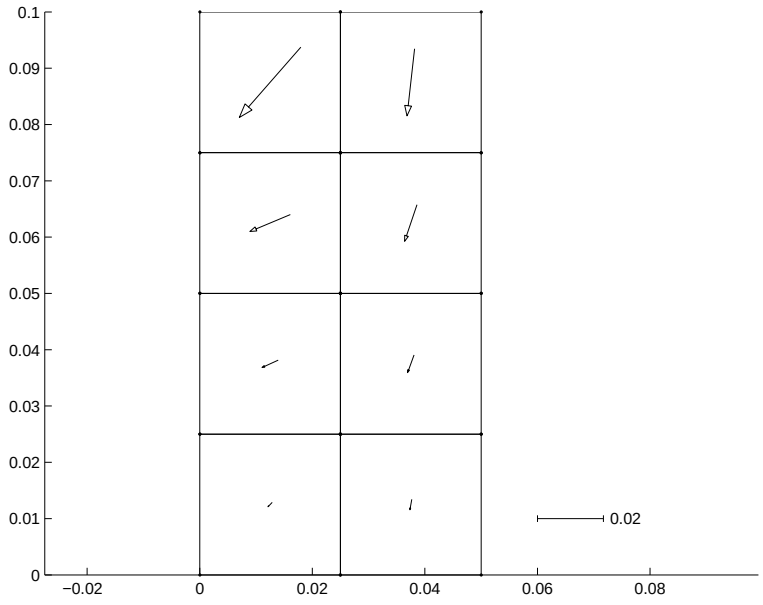
```
a=                    r=
  1.0e-003 *            1.0e-003 *

         0              -0.0165
         0              -0.0565
         0              -0.0399
         0              -0.0777
    0.0662               0.0000
    0.0935                    0
         0              -0.2143
    0.1786               0.0000
    0.2500               0.0000
         0              -0.6366
    0.4338               0.0000
    0.5494              -0.0000
    0.5000               0.0165
    1.0000               0.7707
    1.0000               0.2542
```

```
Es =
   -0.0013   -0.0013
   -0.0005   -0.0032
   -0.0049   -0.0022
   -0.0020   -0.0054
   -0.0122   -0.0051
   -0.0037   -0.0111
   -0.0187   -0.0213
   -0.0023   -0.0203
```

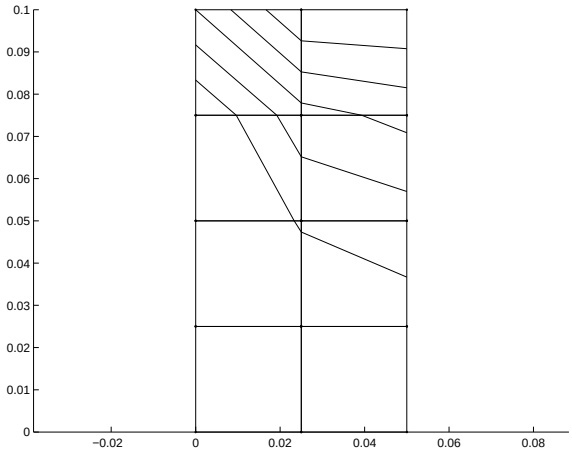
The following .m-file shows an alternative set of commands to perform the diffusion analysis of exs8. By use of global coordinates, an FE-mesh is generated. Also plots with flux-vectors and contour lines are created.

```
% ----- System matrices -----

K=zeros(15);  f=zeros(15,1);
Coord=[0     0    ; 0.025 0    ; 0.05  0
       0     0.025; 0.025 0.025; 0.05  0.025
       0     0.05 ; 0.025 0.05 ; 0.05  0.05
       0     0.075; 0.025 0.075; 0.05  0.075
       0     0.1  ; 0.025 0.1  ; 0.05  0.1  ];
Dof=[1; 2; 3
     4; 5; 6
     7; 8; 9
    10;11;12
    13;14;15];

% ----- Element properties, topology and coordinates -----

ep=1; D=[1 0;0 1];
Edof=[1   1  2  5  4
      2   2  3  6  5
      3   4  5  8  7
      4   5  6  9  8
      5   7  8 11 10
      6   8  9 12 11
      7  10 11 14 13
      8  11 12 15 14];
[Ex,Ey]=coordxtr(Edof,Coord,Dof,4);

% ----- Generate FE-mesh -----

eldraw2(Ex,Ey,[1 3 0],Edof(:,1));
pause; clf;

% ----- Create and assemble element matrices -----

for i=1:8
  Ke=flw2qe(Ex(i,:),Ey(i,:),ep,D);
  K=assem(Edof(i,:),K,Ke);
end;

% ----- Solve equation system -----

bc=[1 0;2 0;3 0;4 0;7 0;10 0;13 0.5e-3;14 1e-3;15 1e-3];
```

```
[a,r]=solveq(K,f,bc)

% ----- Compute element flux vectors -----

 Ed=extract(Edof,a);
 for i=1:8
   Es(i,:)=flw2qs(Ex(i,:),Ey(i,:),ep,D,Ed(i,:))
 end

% ----- Draw flux vectors and contour lines -----
 sfac=scalfact2(Ex,Ey,Es,0.5);
 eldraw2(Ex,Ey,[1,3,0]);
 elflux2(Ex,Ey,Es,[1,4],sfac);
 pltscalb2(sfac,[2e-2 0.06 0.01],4);
 pause; clf;
 eldraw2(Ex,Ey,[1,3,0]);
 eliso2(Ex,Ey,Ed,5,[1,4]);
```



Flux vectors



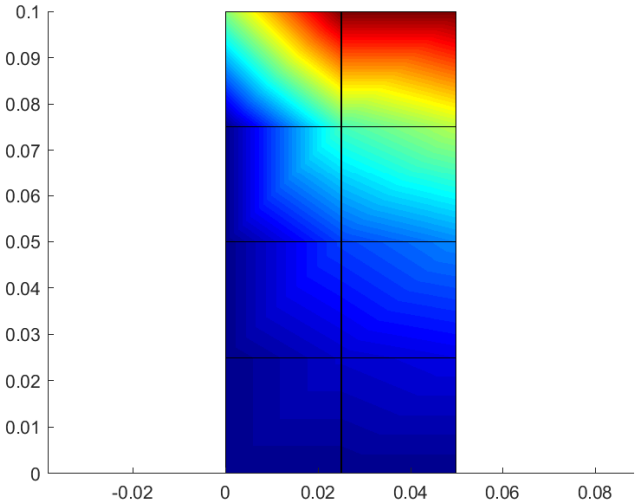
Contour lines

In the upper left corner, the contour lines should physically have met at the corner point. However, the drawing of the contour lines for the planqe element follows the numerical approximation. A finer element mesh will bring the contour lines closer to the corner point.

Along the symmetry line, the contour lines should physically be perpendicular to the boundary. This will also be improved with a finer element mesh.

With the MATLAB functions colormap and fill a filled contour of the concentrations can be plotted.

```
colormap('jet{\'})
fill(Ex',Ey',Ed')
axis equal
```



Filled contour

Index

Index