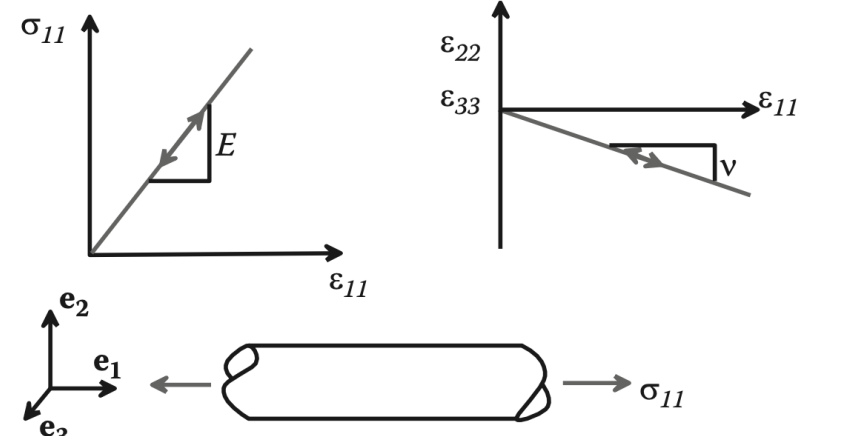


Chapter 2 - The Finite Element (FE) Method for Static Linear Elasticity

2.1 Derivation and implementation of a basic 2D FE code with triangular constant strain elements

1. Linear elastic material behavior

What is the basic idea of a linear elastic material? It has a uniaxial stress-strain response (valid only for small strains) as shown in the figure. The stress-strain law for the material may be expressed in matrix form as:

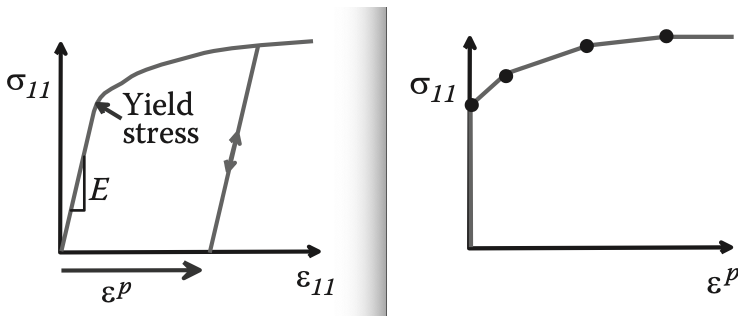


$$(2.1) \quad \begin{bmatrix} \epsilon_{11} \\ \epsilon_{22} \\ \epsilon_{33} \\ \epsilon_{12} \\ \epsilon_{13} \\ \epsilon_{23} \end{bmatrix} = \frac{1}{E} \begin{bmatrix} 1 & -\nu & -\nu & 0 & 0 & 0 \\ -\nu & 1 & -\nu & 0 & 0 & 0 \\ -\nu & -\nu & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1+\nu & 0 & 0 \\ 0 & 0 & 0 & 0 & 1+\nu & 0 \\ 0 & 0 & 0 & 0 & 0 & 1+\nu \end{bmatrix} \begin{bmatrix} \sigma_{11} \\ \sigma_{22} \\ \sigma_{33} \\ \sigma_{12} \\ \sigma_{13} \\ \sigma_{23} \end{bmatrix} + \alpha \Delta T \begin{bmatrix} 1 \\ 1 \\ 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

where E and ν are Young's modulus and Poisson's ratio for the material, whereas α denotes the thermal expansion coefficient. Typical values for steel, for example, are $E = 210 \text{ GN/m}^2$, $\nu = 0.33$, and $\alpha = 5 \times 10^{-6} \text{ K}^{-1}$.

2.2 Elastic-plastic material behavior

The uniaxial stress-strain curve for an elastic-plastic solid looks something like the one shown in the figure. The material behaves elastically until a critical stress (known as the yield stress) is reached. If yield is exceeded, the material deforms permanently. The yield stress of the material generally increases with plastic strain: this behavior is known as strain hardening. The conditions necessary to initiate yielding under multiaxial loading are specified by a *yield criterion*, such as the *von Mises* or *Tresca criteria*. These yield criteria can be built into the FE code. The strain hardening behavior of a material is approximated by allowing the yield stress to increase with plastic strain. The nonlinear variation of yield stress with plastic strain for a material is usually specified by representing it as a series of straight lines, as shown in the figure.

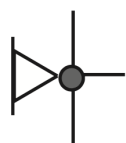


2.3 Boundary condition

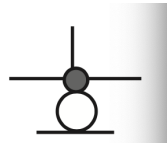
Boundary conditions are used to specify the loading applied to a solid. There are several ways to apply loads to a finite element mesh.

1. Displacement boundary conditions

The displacements at any node on the boundary or within the solid can be specified. One may prescribe u_1, u_2, u_3 or all of those. For example, to stretch a 2D block of material vertically while allowing it to expand or contract freely horizontally, we would apply boundary constraints to the top and bottom surface as shown in the figure for making the finite element program able to find a unique displacement field solution. Notice that you cannot directly apply a rotation to a node attached to a 2D or 3D solid. Rotations can, however, be applied to the nodes attached to certain special types of element, such as beams, plates, and shells, as well as rigid surfaces.



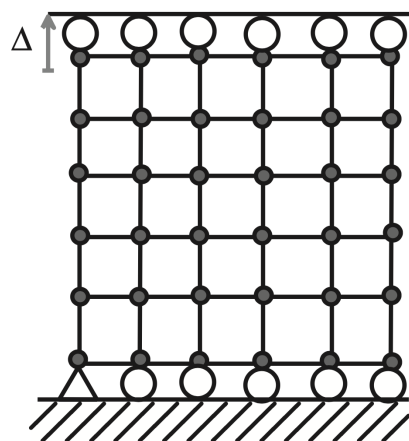
Built in
(or encastre)
 u_1, u_2 prescribed



Roller
 u_2 prescribed



Roller
 u_1 prescribed



2. Symmetry conditions

Finite element codes need to automatically enforce symmetry and antisymmetry boundary conditions.

3. Prescribed forces

Any node in a finite element mesh may be subjected to a prescribed force. The nodal force is a vector and is specified by its three (or two for 2D) components, (F_1, F_2, F_3) . Notice that there is no direct way to apply a moment to a solid in FEM; you would need to do this by applying two point forces a small distance apart or by applying contact loading, as outlined below. Moments can be applied to some special types of element, such as shells, plates, or beams.

4. Distributed loads

A solid may be subjected to distributed pressure or traction acting on its boundary. Examples include aerodynamic loading, or hydrostatic fluid pressure. Distributed traction is a vector quantity, with physical dimensions of *force per unit area in 3D*, and *force per unit length in 2D*. To model this type of loading in a finite element program, distributed loads may be applied to the *face* of any element.

5. Default boundary condition at boundary nodes

If no displacements or forces are prescribed at a boundary node and no distributed loads act on any element faces connected to that node, then the node is assumed to be free of external force.

6. Body forces

External body forces may act on the interior of a solid. Examples of body forces include gravitational loading, or electromagnetic forces. Body force is a vector quantity, with physical dimensions of force per unit volume. To model this type of loading in a finite element program, body forces may be applied to the interior of any element.

7. Contact

Probably the most common way to load a solid is through contact with another solid. Special procedures are available for modeling contact between solids. These will be discussed in a separate section in the course.

8. Load history

In some cases, one may want to apply a cycle of load to a solid. In this case, the prescribed loads and displacements must be specified as a function of time.

2.4 Constraint

You may sometimes need to use more complicated boundary conditions than simply constraining the motion or loads applied to a solid. Some examples might include the following:

1. Connecting different element types (e.g., beam elements to solid elements)
2. Enforcing periodic boundary conditions
3. Constraining a boundary to remain flat
4. Approximating the behavior of mechanical components such as welds, bushings, bolted joints, etc.

You can do this by defining constraints in an analysis. At the most basic level, constraints can simply be used to enforce prescribed relationships between the displacements or velocities of individual nodes in the mesh. You can also specify relationships between motion of groups of nodes.

2.5 Geometry linearity and nonlinearity

1. Geometry linearity

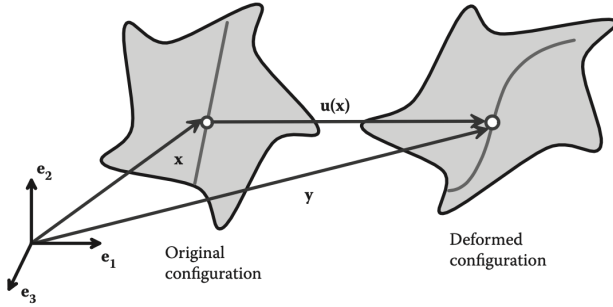
One can simplify the calculation of internal forces in a system by neglecting its shape changes, which is small, when solving the equations of equilibrium. For example, when you solve the problem of a truss system, you usually calculate forces in each member based on the roughly undeformed shape of the structure. You can use the same idea to simplify calculations involving slightly deformed solids. In such case of problems, if the materials have linear stress-strain relations, the FE calculation is linear.

2. Geometry nonlinearity

If one anticipate that material might stretch by more than approximately 10%, one expect that some part of the solid might rotate by more than about 10° , or one want to calculate buckling loads for the system of interest, the finite geometry changes in the computation should be considered. This will automatically make your calculation nonlinear, even if all the materials have linear stress-strain relations.

2.6 Basic finite element program

The goal of this section is to provide some insight into the theory and algorithms that are coded in a finite element program. Here, we will implement only the simplest possible finite element code: specifically, we will develop an FEM to solve a 2D (*plane stress* or *plane strain*) static boundary value problem in linear elasticity, as shown in the figure.



1. We are given the following:

- (1) The shape of the solid in its unloaded condition R .
- (2) Boundary conditions for displacements $\mathbf{u}^*(x)$ on a portion $\partial_1 R$ or tractions \mathbf{t}^* on a portion $\partial_2 R$ of the boundary of R .

2. We make the following assumptions:

- (1) The solid is an isotropic, linear elastic solid with Young's modulus E and Poisson's ratio ν .
- (2) Plane strain or plane stress deformation.
- (3) The solid is at constant temperature (no thermal strains).
- (4) Body forces of the solid are neglected.

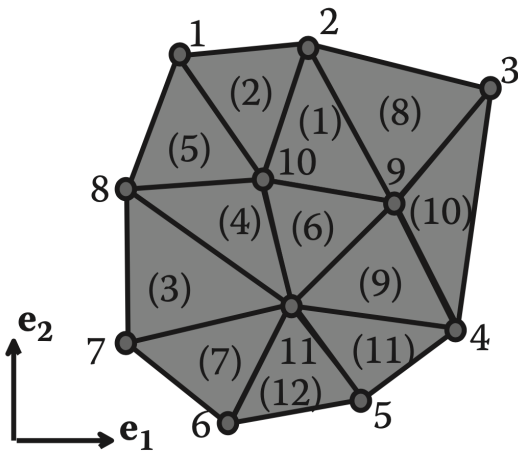
3. Four steps

We then want to find a displacement field \mathbf{u}_i satisfying the usual field equations and boundary conditions, which will be discussed later. There are four steps which are based on *the principle of minimum potential energy*:

- (1) A finite element mesh is constructed to interpolate the displacement field.
- (2) The strain energy in each element is calculated in terms of the displacements of each node.
- (3) The potential energy of tractions acting on the solid's boundary is added.
- (4) The displacement field is calculated by minimizing the potential energy.

4. Finite element mesh and element connectivity

For simplicity, the elements are designed to be three-noded triangles, as shown in the figure. The nodes are numbered $1, 2, \dots, N = 11$, whereas the elements are numbered $1, 2, \dots, L = 12$. Element numbers are shown in parentheses.



(1) The position of the a -th node: $\mathbf{x}_i^{(a)}$ ($i = 1, 2$ for 2D and $i = 1, 2, 3$ for 3D problems)

(2) The element connectivity: Specifying the node numbers attached to each element. For example, in the figure, the connectivity for element 1 is (10, 9, 2), for element 2 it is (10, 2, 1), etc.

5. Global displacement vector

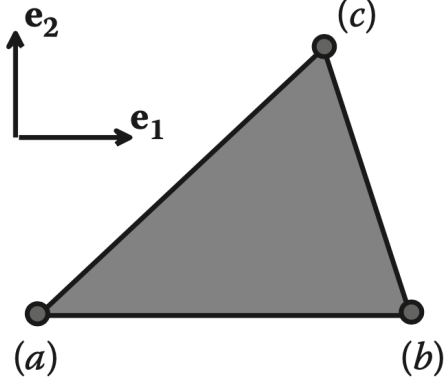
In FEM, the displacement field by *interpolating* between values at the nodes, as follows. Let $\mathbf{u}_i^{(a)}$ denotes the unknown displacement vector at nodes $a = 1, 2, \dots, N$. In the FE code, the displacements for a plane stress or plane strain problem are normally stored as a column vector like the one shown below:

$$(2.2) \mathbf{u} = \begin{bmatrix} u_1^{(1)} & u_2^{(1)} & u_1^{(2)} & u_2^{(2)} & u_1^{(3)} & u_2^{(3)} & \dots \end{bmatrix}$$

The unknown displacement components are going to be determined by minimizing the potential energy of the solid (will be discussed later).

6. Element interpolation function

As mentioned above, since the potential energy of solid needs to be calculated, we have to compute the displacements within each element. This is done by interpolation. Consider a triangular element, with nodes a, b, c at its corners, as shown in the figure. Let $x_i^{(a)}, x_i^{(b)}, x_i^{(c)}$ denote the coordinates of the corners. Define the element interpolation functions (also known as *shape functions*) as follows:



$$(2.3) N_a(x_1, x_2) = \frac{(x_2 - x_2^{(b)})(x_1^{(c)} - x_1^{(b)}) - (x_1 - x_1^{(b)})(x_2^{(c)} - x_2^{(b)})}{(x_2^{(a)} - x_2^{(b)})(x_1^{(c)} - x_1^{(b)}) - (x_1^{(a)} - x_1^{(b)})(x_2^{(c)} - x_2^{(b)})}$$

$$(2.4) N_b(x_1, x_2) = \frac{(x_2 - x_2^{(c)})(x_1^{(a)} - x_1^{(c)}) - (x_1 - x_1^{(c)})(x_2^{(a)} - x_2^{(c)})}{(x_2^{(b)} - x_2^{(c)})(x_1^{(a)} - x_1^{(c)}) - (x_1^{(b)} - x_1^{(c)})(x_2^{(a)} - x_2^{(c)})}$$

$$(2.5) N_c(x_1, x_2) = \frac{(x_2 - x_2^{(a)})(x_1^{(b)} - x_1^{(a)}) - (x_1 - x_1^{(a)})(x_2^{(b)} - x_2^{(a)})}{(x_2^{(c)} - x_2^{(a)})(x_1^{(b)} - x_1^{(a)}) - (x_1^{(c)} - x_1^{(a)})(x_2^{(b)} - x_2^{(a)})}$$

These shape functions are constructed so that

- (1) They vary linearly with position within the element, and
- (2) Each shape function has a value of one at one of the nodes and is zero at the other two.

We then write:

$$(2.6) u_i(x_1, x_2) = u_i^{(a)} N_a(x_1, x_2) + u_i^{(b)} N_b(x_1, x_2) + u_i^{(c)} N_c(x_1, x_2)$$

Of course, the shape functions above are valid only for three-noded triangular elements; other elements have more complicated interpolation functions.

7. Element strains, stresses, and strain energy density

We can now compute the strain distribution within the element and hence determine the strain energy density. Because we are solving a plane strain problem, the only non-zero strains are $\varepsilon_{11}, \varepsilon_{22}, \varepsilon_{12}$. It is convenient to express the results in matrix form, as follows:

$$(2.7) \boldsymbol{\varepsilon} = \mathbf{B} \mathbf{u}^{elem.} = \begin{bmatrix} \varepsilon_{11} \\ \varepsilon_{22} \\ 2\varepsilon_{12} \end{bmatrix} = \begin{bmatrix} \frac{\partial N_a}{\partial x_1} & 0 & \frac{\partial N_b}{\partial x_1} & 0 & \frac{\partial N_c}{\partial x_1} & 0 \\ 0 & \frac{\partial N_a}{\partial x_2} & 0 & \frac{\partial N_b}{\partial x_2} & 0 & \frac{\partial N_c}{\partial x_2} \\ \frac{\partial N_a}{\partial x_2} & \frac{\partial N_a}{\partial x_1} & \frac{\partial N_b}{\partial x_2} & \frac{\partial N_b}{\partial x_1} & \frac{\partial N_c}{\partial x_2} & \frac{\partial N_c}{\partial x_1} \end{bmatrix} \begin{bmatrix} u_1^{(a)} \\ u_2^{(a)} \\ u_1^{(b)} \\ u_2^{(b)} \\ u_1^{(c)} \\ u_2^{(c)} \end{bmatrix}$$

Note that, for linear triangular elements, the matrix of shape function derivatives \mathbf{B} is constant. It depends only on the coordinates of the corners of the element and does not vary with position within the element. That is the reason for generally calling triangular elements as *constant strain elements*.

However, this is not the case for most elements. Now, we can compute the strain energy density within the element. Begin by computing the stresses within the element. For plane strain deformation, we have the following matrix form of $\boldsymbol{\sigma} = \mathbf{D} \boldsymbol{\varepsilon}$:

$$(2.8) \begin{bmatrix} \sigma_{11} \\ \sigma_{22} \\ \sigma_{12} \end{bmatrix} = \frac{E}{(1+\nu)(1-2\nu)} \begin{bmatrix} 1-\nu & \nu & 0 \\ \nu & 1-\nu & 0 \\ 0 & 0 & \frac{1-2\nu}{2} \end{bmatrix} \begin{bmatrix} \varepsilon_{11} \\ \varepsilon_{22} \\ 2\varepsilon_{12} \end{bmatrix}$$

For plane stress, the result is:

$$(2.9) \begin{bmatrix} \sigma_{11} \\ \sigma_{22} \\ \sigma_{12} \end{bmatrix} = \frac{E}{(1-\nu^2)} \begin{bmatrix} 1 & \nu & 0 \\ \nu & 1 & 0 \\ 0 & 0 & \frac{1-\nu}{2} \end{bmatrix} \begin{bmatrix} \varepsilon_{11} \\ \varepsilon_{22} \\ 2\varepsilon_{12} \end{bmatrix}$$

Since the strain energy density is related to the stresses and strains by $U = \sigma_{ij}\epsilon_{ij} / 2$. This can be written in matrix form as:

$$(2.10) \quad U^{elem} = \frac{1}{2} \boldsymbol{\epsilon}^T \boldsymbol{\sigma} = \frac{1}{2} \boldsymbol{\epsilon}^T \mathbf{D} \boldsymbol{\epsilon} = \frac{1}{2} \mathbf{u}^{elemT} \mathbf{B}^T \mathbf{D} \mathbf{B} \mathbf{u}^{elem}$$

Because \mathbf{B} is constant within an element, we merely need to multiply the strain energy density by the area of the element with unit thickness, which can be computed from the coordinates of the element corners as follows:

$$(2.11) \quad A^{elem} = \frac{1}{2} |(x_1^b - x_1^a)(x_2^c - x_2^a) - (x_1^c - x_1^a)(x_2^b - x_2^a)|$$

Hence, the total strain energy of the element is:

$$(2.12) \quad W^{elem} = \frac{1}{2} \mathbf{u}^{elemT} A^{elem} \mathbf{B}^T \mathbf{D} \mathbf{B} \mathbf{u}^{elem}$$

8. Element stiffness matrix

From Eq. (2.12), the element stiffness matrix can be defined as:

$$(2.13) \quad \mathbf{K}^{elem} = A^{elem} \mathbf{B}^T \mathbf{D} \mathbf{B}$$

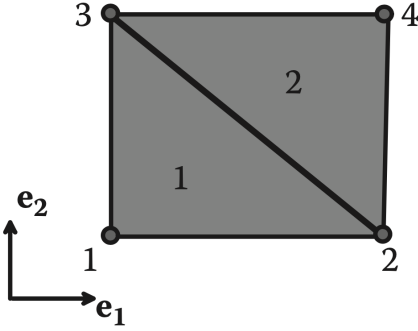
Observe that, because the material property matrix \mathbf{D} is symmetric, the element stiffness matrix is also symmetric:

9. Global stiffness matrix

(1) The total strain energy of the solid may be computed by adding together the strain energy of each element:

$$(2.14) \quad W = \sum_{elem} W^{elem} = \frac{1}{2} \sum_{elem} \mathbf{u}^{elemT} \mathbf{K}^{elem} \mathbf{u}^{elem}$$

(2) It is more convenient to express W in terms of the vector \mathbf{u} , which contains all the nodal displacements, rather than using \mathbf{u}^{elem} to the displacements for each element. For example, the strain energy for the simple two-element mesh shown in the figure is:



$$(2.15) \quad W = \frac{1}{2} \begin{bmatrix} u_1^{(1)} & u_2^{(1)} & u_1^{(2)} & u_2^{(2)} & u_1^{(3)} & u_2^{(3)} \end{bmatrix} \begin{bmatrix} k_{11}^{(1)} & k_{12}^{(1)} & \dots & k_{16}^{(1)} \\ k_{21}^{(1)} & k_{22}^{(1)} & & \\ \vdots & & \ddots & \\ k_{61}^{(1)} & & & k_{66}^{(1)} \end{bmatrix} \begin{bmatrix} u_1^{(1)} \\ u_2^{(1)} \\ u_1^{(2)} \\ u_2^{(2)} \\ u_1^{(3)} \\ u_2^{(3)} \end{bmatrix} \\ + \frac{1}{2} \begin{bmatrix} u_1^{(2)} & u_2^{(2)} & u_1^{(3)} & u_2^{(3)} & u_1^{(4)} & u_2^{(4)} \end{bmatrix} \begin{bmatrix} k_{11}^{(2)} & k_{12}^{(2)} & \dots & k_{16}^{(2)} \\ k_{21}^{(2)} & k_{22}^{(2)} & & \\ \vdots & & \ddots & \\ k_{61}^{(2)} & & & k_{66}^{(2)} \end{bmatrix} \begin{bmatrix} u_1^{(2)} \\ u_2^{(2)} \\ u_1^{(3)} \\ u_2^{(3)} \\ u_1^{(4)} \\ u_2^{(4)} \end{bmatrix}$$

(3) We could add the missing terms to each element displacement vector:

(2.16)

$$\begin{aligned}
W = & \frac{1}{2} \begin{bmatrix} u_1^{(1)} & u_2^{(1)} & u_1^{(2)} & u_2^{(2)} & u_1^{(3)} & u_2^{(3)} & u_1^{(4)} & u_2^{(4)} \end{bmatrix} \begin{bmatrix} k_{11}^{(1)} & k_{12}^{(1)} & \dots & k_{16}^{(1)} & 0 & 0 \\ k_{21}^{(1)} & k_{22}^{(1)} & & & 0 & 0 \\ \vdots & & \ddots & & 0 & 0 \\ k_{61}^{(1)} & & & k_{66}^{(1)} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} u_1^{(1)} \\ u_2^{(1)} \\ u_1^{(2)} \\ u_2^{(2)} \\ u_1^{(3)} \\ u_2^{(3)} \\ u_1^{(4)} \\ u_2^{(4)} \end{bmatrix} \\
& + \frac{1}{2} \begin{bmatrix} u_1^{(1)} & u_2^{(1)} & u_1^{(2)} & u_2^{(2)} & u_1^{(3)} & u_2^{(3)} & u_1^{(4)} & u_2^{(4)} \end{bmatrix} \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & k_{11}^{(1)} & k_{12}^{(1)} & \dots & k_{16}^{(1)} \\ 0 & 0 & k_{21}^{(1)} & k_{22}^{(1)} & & \\ 0 & 0 & \vdots & & \ddots & \\ 0 & 0 & k_{61}^{(1)} & & & k_{66}^{(1)} \end{bmatrix} \begin{bmatrix} u_1^{(1)} \\ u_2^{(1)} \\ u_1^{(2)} \\ u_2^{(2)} \\ u_1^{(3)} \\ u_2^{(3)} \\ u_1^{(4)} \\ u_2^{(4)} \end{bmatrix} \\
& = \frac{1}{2} \begin{bmatrix} u_1^{(1)} & u_2^{(1)} & u_1^{(2)} & u_2^{(2)} & u_1^{(3)} & u_2^{(3)} & u_1^{(4)} & u_2^{(4)} \end{bmatrix} \begin{bmatrix} k_{11}^{(1)} & k_{12}^{(1)} & k_{13}^{(1)} & k_{14}^{(1)} & k_{15}^{(1)} & k_{16}^{(1)} & 0 & 0 \\ k_{21}^{(1)} & k_{22}^{(1)} & k_{23}^{(1)} & k_{24}^{(1)} & k_{25}^{(1)} & k_{26}^{(1)} & 0 & 0 \\ k_{31}^{(1)} & k_{32}^{(1)} & k_{33}^{(1)} + k_{11}^{(2)} & k_{34}^{(1)} + k_{12}^{(2)} & k_{35}^{(1)} + k_{13}^{(2)} & k_{36}^{(1)} + k_{14}^{(2)} & k_{15}^{(2)} & k_{16}^{(2)} \\ k_{41}^{(1)} & k_{42}^{(1)} & k_{43}^{(1)} + k_{21}^{(2)} & k_{44}^{(1)} + k_{22}^{(2)} & k_{45}^{(1)} + k_{23}^{(2)} & k_{46}^{(1)} + k_{24}^{(2)} & k_{25}^{(2)} & k_{26}^{(2)} \\ k_{51}^{(1)} & k_{52}^{(1)} & k_{53}^{(1)} + k_{31}^{(2)} & k_{54}^{(1)} + k_{32}^{(2)} & k_{55}^{(1)} + k_{33}^{(2)} & k_{56}^{(1)} + k_{34}^{(2)} & k_{35}^{(2)} & k_{36}^{(2)} \\ k_{61}^{(1)} & k_{62}^{(1)} & k_{63}^{(1)} + k_{41}^{(2)} & k_{64}^{(1)} + k_{42}^{(2)} & k_{65}^{(1)} + k_{43}^{(2)} & k_{66}^{(1)} + k_{44}^{(2)} & k_{45}^{(2)} & k_{46}^{(2)} \\ 0 & 0 & k_{51}^{(2)} & k_{52}^{(2)} & k_{53}^{(2)} & k_{54}^{(2)} & k_{55}^{(2)} & k_{56}^{(2)} \\ 0 & 0 & k_{61}^{(2)} & k_{62}^{(2)} & k_{63}^{(2)} & k_{64}^{(2)} & k_{65}^{(2)} & k_{66}^{(2)} \end{bmatrix} \begin{bmatrix} u_1^{(1)} \\ u_2^{(1)} \\ u_1^{(2)} \\ u_2^{(2)} \\ u_1^{(3)} \\ u_2^{(3)} \\ u_1^{(4)} \\ u_2^{(4)} \end{bmatrix}
\end{aligned}$$

Therefore, we can write:

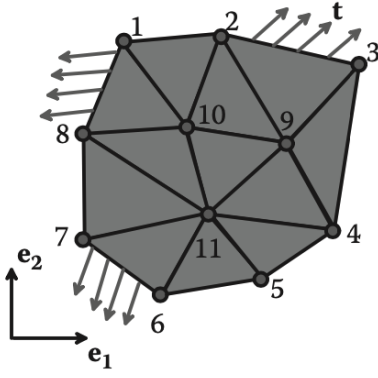
$$(2.17) \quad W = \frac{1}{2} \mathbf{u}^T \mathbf{K} \mathbf{u}$$

where \mathbf{K} is known as the *global stiffness* matrix. It is the sum of all the element stiffness matrices. Because the element stiffness matrix is symmetric, the global stiffness matrix must also be symmetric. To assemble the global stiffness matrix for a plane strain or plane stress mesh with N nodes, we use the following procedure:

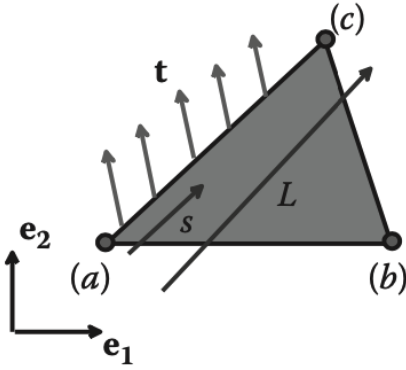
- (1) Note that, for N nodes, there will be $2N$ unknown displacement components (two at each node). Therefore, we start by setting up storages for a ($2N \times 2N$) global stiffness matrix:
- (2) Next, begin a loop over the elements.
- (3) For the current element, add the element stiffness matrix to the global stiffness matrix, using the following procedure. Considering the information of element connectivity, let $n_a^{(j)}$ ($a = 1, 2, 3$, and $j = 1, 2$) denote the *global numbers* of the three corner nodes of the j -th element (i.e., $n_1^{(1)} = 1, n_2^{(1)} = 2, n_3^{(1)} = 3$ for the first element or $n_1^{(2)} = 2, n_2^{(2)} = 3, n_3^{(2)} = 4$ for the second element), and for the j -th element:
 - Loop 1: $a = 1$ to 3
 - Loop 2: $i = 1$ to 2
 - Loop 3: $b = 1$ to 3
 - Loop 4: $k = 1$ to 2
 - $ir = 2(n_a^{(j)} - 1) + i$
 - $ic = 2(n_b^{(j)} - 1) + k$
 - $K_{2(a-1)+i, 2(b-1)+k}^{elem} \rightarrow K_{ir, ic}$
 - End of Loop 3
 - End of Loop 2
 - End of Loop 1
- (4) Proceed to the next element.

10. Boundary loading

We have now found a way to compute the strain energy for a finite element mesh. Next, we need to compute the boundary term in the potential energy. Considering tractions acting on a finite element, as shown in the figure, and boundary loading will be specified as follows:



- (1) The element on which the loading acts.
 (2) The face of the element that is loaded.
 (3) The traction vector \mathbf{t} (force per unit area) that acts on the face of the element. The traction is assumed to be *constant* on the face of any one element.
 Now, we compute the contribution to the potential energy attributable to the traction acting on the face of one element. For the element shown in the figure, the contribution to the potential energy would be:



$$(2.18) \quad P = - \int_0^L t_i u_i ds$$

$$(2.19) \quad u_i = u_i^{(a)} \left(1 - \frac{s}{L} \right) + u_i^{(c)} \frac{s}{L}$$

Because the tractions are uniform along the element edge,

$$(2.20) \quad \begin{aligned} P^{elem} &= -t_i u_i^{(a)} \int_0^L \left(1 - \frac{s}{L} \right) ds - t_i u_i^{(c)} \int_0^L \frac{s}{L} ds \\ &= -t_i u_i^{(a)} \frac{L}{2} - t_i u_i^{(c)} \frac{L}{2} \\ &= - \left[t_1 \frac{L}{2} \quad t_2 \frac{L}{2} \quad t_1 \frac{L}{2} \quad t_2 \frac{L}{2} \right] \cdot \begin{bmatrix} u_1^{(a)} & u_2^{(a)} & u_1^{(c)} & u_2^{(c)} \end{bmatrix} \end{aligned}$$

Define $\mathbf{r}^{face} \equiv [t_1 L/2 \quad t_2 L/2 \quad t_1 L/2 \quad t_2 L/2]$ and $\mathbf{u}^{face} \equiv [u_1^{(a)} \quad u_2^{(a)} \quad u_1^{(c)} \quad u_2^{(c)}]$, the potential energy of the element can be abbreviated as:

$$(2.21) \quad P^{elem} = -\mathbf{r}^{face} \cdot \mathbf{u}^{face}$$

Similar to the concept of assembling global stiffness matrix of a solid, the total contribution to the potential energy attributable to boundary loading on all element faces is:

$$(2.22) \quad P = \sum P^{elem} = - \sum \mathbf{r}^{face} \cdot \mathbf{u}^{face} = -\mathbf{r} \cdot \mathbf{u}$$

where \mathbf{r} is the *global residual force vector* corresponding to the DOFs of the global displacement vector \mathbf{u} . The global residual force vector for a mesh with N nodes is assembled by looping over the loaded elements. Recalling that $n_a^{(j)}$ ($a = 1, 2, 3$, and $j = 1, 2$) denote the global numbers of the three corner nodes of the j -th element, and for a loaded edge of the j -th element:

Loop 1: $a = 1$ to 2

Loop 2: $i = 1$ to 2

$$ir = 2(n_a^{(j)} - 1) + i$$

$$r_{2(a-1)+i}^{edge} \rightarrow r_{ir}$$

End of Loop 2

End of Loop 1

11. Prescribed displacements

For case that some displacements are prescribed in a finite element mesh which is subjected to loads, the global stiffness matrix and global residual force vector are still assembled exactly as described in the preceding section. They are then modified to enforce the displacement constraint. The procedure is best illustrated using an following example. Suppose that the finite element equations after assembly have the form (derivation will be discussed in the Section 7):

$$(2.23) \quad \begin{bmatrix} k_{11} & k_{12} & \dots & k_{1\ 2N} \\ k_{21} & k_{22} & & k_{2\ 2N} \\ \vdots & & \ddots & \\ k_{2N\ 1} & k_{2N\ 2} & & k_{2N\ 2N} \end{bmatrix} \begin{bmatrix} u_1^{(1)} \\ u_2^{(1)} \\ \vdots \\ u_2^{(N)} \end{bmatrix} = \begin{bmatrix} r_1 \\ r_2 \\ \vdots \\ r_{2N} \end{bmatrix}$$

To prescribe displacements for any node, we simply replace the equation for the appropriate degrees of freedom with the corresponding constraint. For example, to enforce $u_2^{(1)} = \Delta$, we could modify the above finite element equations to:

$$(2.24) \quad \begin{bmatrix} k_{11} & k_{12} & \dots & k_{1\ 2N} \\ 0 & 1 & & 0 \\ \vdots & & \ddots & \\ k_{2N\ 1} & k_{2N\ 2} & & k_{2N\ 2N} \end{bmatrix} \begin{bmatrix} u_1^{(1)} \\ u_2^{(1)} \\ \vdots \\ u_2^{(N)} \end{bmatrix} = \begin{bmatrix} r_1 \\ \Delta \\ \vdots \\ r_{2N} \end{bmatrix}$$

Thus, the equation for $u_2^{(1)}$ has been replaced with the constraint $u_2^{(1)} = \Delta$. This procedure works, but it has the disadvantage that the modified stiffness matrix is no longer symmetric. It is preferable to modify the stiffness and residual further, to retain symmetry. To do so, we eliminate the constrained degrees of freedom from all rows of the stiffness matrix. Consider the modified matrix equation (Eq. 2.24), and the stiffness matrix can retain symmetric by setting each entry in the second column (apart from the diagonal) to zero:

$$(2.25) \quad \begin{bmatrix} k_{11} & 0 & \dots & k_{1\ 2N} \\ 0 & 1 & & 0 \\ \vdots & & \ddots & \\ k_{2N\ 1} & 0 & & k_{2N\ 2N} \end{bmatrix} \begin{bmatrix} u_1^{(1)} \\ u_2^{(1)} \\ \vdots \\ u_2^{(N)} \end{bmatrix} = \begin{bmatrix} r_1 - k_{12} \Delta \\ \Delta \\ \vdots \\ r_{2N} - k_{2N\ 2} \Delta \end{bmatrix}$$

12. Solution

The result of Sections 6.1 through 6.11 is a set of simultaneous linear equations of the form:

$$(2.26) \quad \mathbf{K}^{mod} \mathbf{u} = \mathbf{r}$$

These can be solved for the unknown displacements \mathbf{u} using standard techniques (e.g., *Gaussian elimination* or *iterative techniques*). An important feature of the FEM equations is that the stiffness matrix is sparse; that is to say, only a small number of entries in the matrix are nonzero. Consequently, special schemes are used to store and factor the equations, which avoid having to store large numbers of zeros.

13. Postprocessing

Once the displacements \mathbf{u} have been computed, the strain in each element can be computed, and so the stress distribution can be deduced (stress recovery). The procedure is as follows:

$$(2.27) \quad \boldsymbol{\epsilon} = \mathbf{B} \mathbf{u}^{elem}$$

$$(2.28) \quad \begin{bmatrix} \epsilon_{11} \\ \epsilon_{22} \\ 2\epsilon_{12} \end{bmatrix} = \begin{bmatrix} \frac{\partial N_a}{\partial x_1} & 0 & \frac{\partial N_b}{\partial x_1} & 0 & \frac{\partial N_c}{\partial x_1} & 0 \\ 0 & \frac{\partial N_a}{\partial x_2} & 0 & \frac{\partial N_b}{\partial x_2} & 0 & \frac{\partial N_c}{\partial x_2} \\ \frac{\partial N_a}{\partial x_2} & \frac{\partial N_a}{\partial x_1} & \frac{\partial N_b}{\partial x_2} & \frac{\partial N_b}{\partial x_1} & \frac{\partial N_c}{\partial x_2} & \frac{\partial N_c}{\partial x_1} \end{bmatrix} \begin{bmatrix} u_1^{(a)} \\ u_2^{(a)} \\ u_1^{(b)} \\ u_2^{(b)} \\ u_1^{(c)} \\ u_2^{(c)} \end{bmatrix}$$

The stresses can then be determined from the stress-strain equations (Eq. 2.8 or Eq. 2.9):

$$(2.29) \quad \boldsymbol{\sigma} = \mathbf{D} \boldsymbol{\epsilon}$$

2.7 Minimizing the potential energy

Now we are going to derive the finite element equations after assembly. For the potential energy of a finite element mesh:

$$(2.30) \quad \begin{aligned} V &= \frac{1}{2} \mathbf{u}^T \mathbf{K} \mathbf{u} - \mathbf{r}^T \mathbf{u} \\ &= \frac{1}{2} \sum_{j=1}^{2N} u_j \sum_{i=1}^{2N} K_{ji} u_i - \sum_{j=1}^{2N} r_j u_j \end{aligned}$$

By minimizing V :

$$(2.31) \quad \frac{\partial V}{\partial u_k} = \frac{1}{2} \sum_{i=1}^{2N} K_{ki} u_i + \frac{1}{2} \sum_{j=1}^{2N} u_j K_{jk} - r_k = 0$$

where since

$$(2.32) \quad \frac{\partial u_j}{\partial u_k} = \begin{cases} 1, & j = k \\ 0, & j \neq k \end{cases}$$

Furthermore, this can be simplified by noting that \mathbf{K} is symmetric:

$$(2.33) \quad \frac{\partial V}{\partial u_k} = \frac{1}{2} \sum_{i=1}^{2N} K_{ki} u_i + \frac{1}{2} \sum_{j=1}^{2N} u_j K_{kj} - r_k = \sum_{i=1}^{2N} K_{ki} u_i - r_k = 0$$

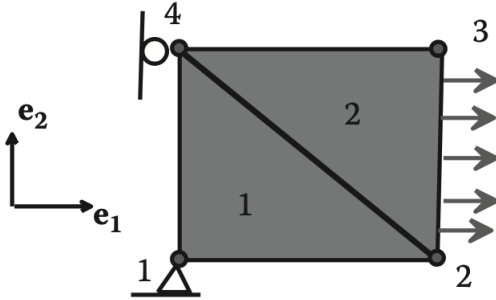
$$(2.34) \Rightarrow \mathbf{K} \mathbf{u} = \mathbf{r}$$

This is a system of $2N$ simultaneous linear equations for the $2N$ unknown nodal displacements. Typically, the system of equations can be solved by inverting the global stiffness matrix. Moreover, standard computational techniques such as Gaussian elimination, Cholesky factorization, or conjugate gradient methods may be used to solve the matrix equations.

2.8 Demonstration of finite element analysis code

1. Finite element analysis demonstration

This Section will demonstrate the implementations of the finite element analysis (FEA) procedures discussed before. First of all, the FEA code should read an input file. Consider a two-dimensional rectangular block in the stress state (deformed configuration) of *plane strain* with Young's modulus 100 GPa and Poisson's ratio 0.3 is meshed with two elements. Node 1 is pinned, node 4 is constrained against horizontal motion, and the right-hand face of element 2 is subjected to a constant horizontal traction with magnitude 10 kN/mm^2 . A very simple input file "input.ipt" for the illustrative mesh with its unit thickness having just two elements is shown below:



```
*PARAMETER
num-dim: 2
*MATPROP
b-plane-strain: 1
young's-modulus: 100.0
poisson's-ratio: 0.3
*NODE
num-node: 4
nodal-coord:
0.0 0.0
1.0 0.0
1.0 1.0
0.0 1.0
*ELEMENT
num-elem: 2
num-elem-node: 3
elem-conn:
1 2 4
2 3 4
*BOUNDARY
num-prescribed-disp: 3
node#-dof#-disp:
```

```

1 1 0.0
1 2 0.0
4 1 0.0
num-prescribed-load: 1
elem#-face#-trac:
2 1 10.0 0.0

```

In the above input file, the information can be mostly self-explanatory. One should note that:

- (1) Nodes are numbered sequentially; thus, node **1** has coordinates $(0.0, 0.0)$, node **2** has coordinates $(1.0, 0.0)$, etc.
- (2) The element connectivity specifies the node numbers attached to each element, using a counterclockwise numbering convention. It doesn't matter which node you use for the first one, as long as all the others are ordered in a counterclockwise sense around the element. For example, you could use $(2, 4, 1)$ instead of $(1, 2, 4)$ for the connectivity of the first element.
- (3) To fix motion of a node, you need to enter the node number, the degree of freedom that is being constrained (**1** for horizontal, **2** for vertical), and a value for the prescribed displacement.
- (4) To specify tractions acting on an element face, you need to enter ① the element number, ② the face number of the element, and ③ the horizontal and vertical components of traction acting on the face. The face numbering scheme is determined by the element connectivity, as follows. Face **1** has the first and second nodes as end points, face **2** has the second and third nodes, and face **3** has the third and first nodes as end points. Because connectivity for element **2** was entered as $(2, 3, 4)$, face **1** of this element has nodes numbered **2** and **3**, face **2** connects nodes numbered **3** and **4**, whereas face **3** connects nodes numbered **4** and **1**.

Typically, the output file "output.opt" from your FEA code should contain the results shown below:

```

*NODE
node#-u1-u2:
1 0.000 0.000
2 0.091 0.000
3 0.091 -0.039
4 0.000 -0.039
*ELEMENT
elem#-e11-e22-e12-s11-s22-s12
1 0.091 -0.039 0.000 10.000 0.000 0.000
2 0.091 -0.039 0.000 10.000 0.000 0.000

```

2. Program architecture

- (1) Read the input file.
- (2) Loop over all elements for assembling the global stiffness matrix from each element stiffness matrix.
- (3) Loop over all loaded element faces for assembling the global residual force vector from each loaded element face force vector.
- (4) Modify the global stiffness matrix and the global residual force vector based on prescribed displacements.
- (5) Solve the finite element equations after assembly.
- (6) Postprocess the strain and stress components of the elements from the displacement results, and generate the output file.
- (7) Visualization the nodal displacement, element stress, and element strain distributions.

3. Example FEA code written in MATLAB

(1) *ReadInput* function

```

function [ndime, nnode, nele, nelnd, npres, ntrac, mate, coor, conn, pres, trac] = ReadInput(infile)
    cellarray = textscan(infile, '%s');
    ndime = str2num(cellarray{1}{3});
    mate = zeros(3,1);
    mate(1) = str2num(cellarray{1}{6});
    mate(2) = str2num(cellarray{1}{8});
    mate(3) = str2num(cellarray{1}{10});
    nnode = str2num(cellarray{1}{13});
    ind = 14;
    coor = zeros(ndime, nnode)
    for i = 1:nnode
        for j = 1:ndime
            ind = ind+1;
            coor(j,i) = str2num(cellarray{1}{ind});
        end
    end
    .
    .
    .
    conn = zeros(nelnd, nele);
    .

```

```

.
.
pres = zeros(3,npres);
.
.
.
trac = zeros(2+ndime,ntrac);
.
.
.
end

```

(2) *ElemStif* function

```

function kel = ElemStif(iel,mate,coor,conn)
    x1a = coor(1,conn(1,iel));
    x2a = coor(2,conn(1,iel));
    x1b = coor(1,conn(2,iel));
    x2b = coor(2,conn(2,iel));
    x1c = coor(1,conn(3,iel));
    x2c = coor(2,conn(3,iel));
    B = zeros(3,6);
    B(1,1) = -(x2c-x2b)/((x2a-x2b)*(x1c-x1b)-(x1a-x1b)*(x2c-x2b));
    B(1,2) = 0;
    B(1,3) = -(x2a-x2c)/((x2b-x2c)*(x1a-x1c)-(x1b-x1c)*(x2a-x2c));
    .
    .
    .
    ael = . . .;
    if(mate(1) == 1)
        D = [1-mate(3) mate(3) 0; mate(3) 1-mate(3) 0; 0 0 (1-2*mate(3)) / 2];
        D = D*mate(2)/(1+mate(3))/(1-2*mate(3));
    else
        D = [1 mate(3) 0; mate(3) 1 0; 0 0 (1-mate(3)) / 2];
        D = D*mate(2)/(1-mate(3)*mate(3));
    end
    kel = ael*B.'*D*B;
end

```

(3) *GlobStif* function

```

function kglob = GlobStif(ndime,nnode,nelem,nelnd,mate,coor,conn)
    kglob = zeros(ndim*nnode,ndim*nnode);
    for j = 1:nelem
        kel = ElemStif(j,mate,coor,conn);
        for a = 1:nelnd
            for i = 1:ndime
                for b = 1:nelnd
                    for k = 1:ndime
                        ir = ndime*(conn(a,j)-1)+i;
                        ic = ndime*(conn(b,j)-1)+k;
                        kglob(ir,ic) = kglob(ir,ic)+kel(ndime*(a-1)+i,ndime*(b-1)+k);
                    end
                end
            end
        end
    end
end

```

(4) *ElemTrac* function

```

function rel = ElemTrac(itrac,coor,conn,trac)
    iel= trac(1,itrac);
    if(trac(2,itrac) == 1)

```

```

    x1a = coor(1,conn(1,iel));
    x2a = coor(2,conn(1,iel));
    x1b = coor(1,conn(2,iel));
    x2b = coor(2,conn(2,iel));
elseif(trac(2,itrac) == 2)
    x1a = coor(1,conn(2,iel));
    x2a = coor(2,conn(2,iel));
    x1b = coor(1,conn(3,iel));
    x2b = coor(2,conn(3,iel));
else
    x1a = coor(1,conn(3,iel));
    x2a = coor(2,conn(3,iel));
    x1b = coor(1,conn(1,iel));
    x2b = coor(2,conn(1,iel));
end
lel = ...;
rel = zeros(4,1);
rel(1) = trac(3)*lel / 2;
rel(2) = trac(4)*lel / 2;
.
.
end

```

(5) *GlobTrac* function

```

function rglob = GlobTrac(ndime,nnode,nelem,nelnd,ntrac,mate,coor,conn,trac)
rglob = zeros(ndim*nnode,1);
for j = 1:ntrac
    rel = ElemTrac(j,coor,conn,trac);
    iel = trac(1,j);
    ia = zeros(2,1);
    if(trac(2,j) == 1)
        ia(1) = 1;
        ia(2) = 2;
    elseif(trac(2,j) == 2)
        ia(1) = 2;
        ia(2) = 3;
    else
        ia(1) = 3;
        ia(2) = 1;
    end
    for a = 1:2
        for i = 1:2
            ir = ndime*(conn(ia(a),iel)-1)+i;
            rglob(ir) = rglob(ir)+rel(ndime*(a-1)+i);
        end
    end
end
end
end

```