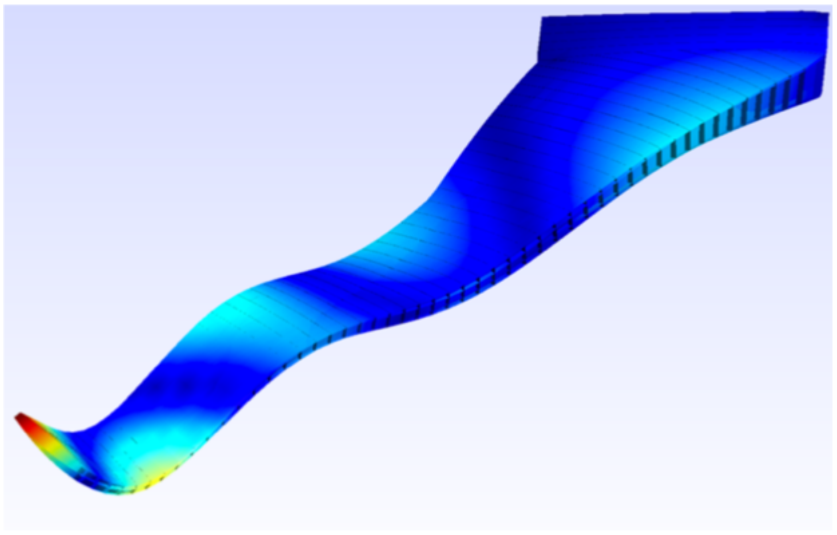
VFEA PART 2: Finite Element Analysis

Joseph Morlier

October 4, 2024



Contents

FISA2A - Principle of Virtual work

Prof. Joseph Morlier

October 9, 2024

1 Principle of minimum potential energy

"The minimum potential energy corresponds to stable static equilibrium."

This is an alternative view to force balance.

Potential energy = PE = Strain energy + Work potential

i.e.,

PE = SE + WP

Strain energy = deformation energy stored in a structure

Work potential = negative of the work done by the external forces

When PE is in terms of the displacements, we minimize PE with respect to the displacements.

A simple example used to understand the principle of minimum potential energy

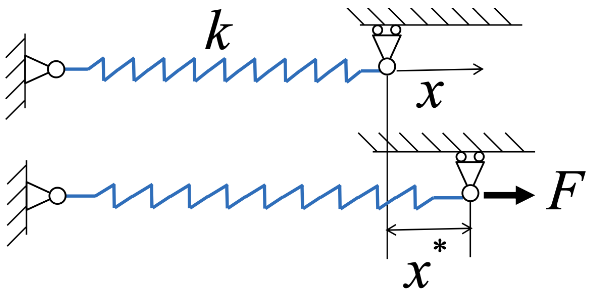


Figure 1: A simple problem

We know from force balance:

$$F = kx^*$$
$$\text{Min}_x\, PE = \frac{1}{2}kx^2 - Fx$$
$$\frac{\partial PE}{\partial x} = kx - F = 0$$

so, $F = kx^*$

More generally:
Let $\mathbf{K}$ be the stiffness matrix and $\mathbf{u}$ be the displacement vector.

$$\begin{aligned}
&\text{Strain energy } = \text{SE} = \frac{1}{2}\mathbf{u}^{\mathrm{T}}\mathrm{Ku} \\
&\text{Work potential } = \text{ WP } = -\mathbf{p}^{\mathrm{T}}\mathbf{u} \\
&\text{Potential energy } = \text{SE} + \text{PE} = \frac{1}{2}\mathbf{u}^{\mathrm{T}}\mathbf{Ku} - \mathbf{p}^{\mathrm{T}}\mathbf{u} \\
&\text{Min}_{\mathbf{u}}\, PE = \frac{1}{2}\mathbf{u}^{T}\mathbf{Ku} - \mathbf{p}^{T}\mathbf{u} \\
&\frac{\partial PE}{\partial \mathbf{u}} = \mathbf{Ku} - \mathbf{p} = 0 \Rightarrow \underset{n\times n}{\mathbf{K}}\underset{n\times 1}{\mathbf{u}} = \underset{n\times 1}{\mathbf{p}}
\end{aligned}$$

2 Principle of virtual work

"External virtual work is equal to the internal virtual work."

This is an alternative view to force balance and the principle of minimum potential energy.

External virtual work = EVW =	work done by the external real forces undergoing virtual displacements
Internal virtual work = IVW =	work done by the real internal forces undergoing virtual displacements

External virtual work = EVW =	work done by external virtual forces undergoing real displacements
Internal virtual work = IVW =	work done by internal virtual forces undergoing real displacements

$EVW = IVW$ is a powerful analytical tool.
It is a thought experiment either with virtual displacements or virtual forces.
We know from force balance:

$$\begin{aligned}
\text{Min}_x\, PE &= \tfrac{1}{2}kx^2 - Fx \\
\tfrac{\partial PE}{\partial x} &= kx - F = 0
\end{aligned}$$

So, $F = kx^*$
Imagine virtual displacement δx

$$\left.\begin{aligned} EVW &= F\delta x \\ IVW &= kx^*\delta x \end{aligned}\right\} EVW = IVW \Rightarrow F = kx^*$$

Imagine virtual force δF

$$\left.\begin{aligned} EVW &= \delta F \quad x^* \\ IVW &= k\delta x \quad x^* \end{aligned}\right] \quad \Rightarrow \delta F(F/k) = k(\delta F/k)x^* \Rightarrow F/k = x^*$$

More generally:

$$\underset{n\times n}{\mathbf{K}}\,\mathbf{u}\overset{=}{_{n\times 1}}\mathbf{p}_{n\times 1}$$

Imagine virtual displacements $\delta\mathbf{u}$

$$\begin{aligned}
&EVW = \mathbf{p}^T\delta\mathbf{u} \\
&IVW = (\mathbf{Ku})^T\delta\mathbf{u} = \mathbf{u}^T\mathbf{K}\delta\mathbf{u} \\
&EVW = IVW \Rightarrow \mathbf{p}^T\delta\mathbf{u} = \mathbf{u}^T\mathbf{K}\delta\mathbf{u} \\
&\Rightarrow \mathbf{p}^T = \mathbf{u}^T\mathbf{K} \\
&\Rightarrow \mathbf{Ku} = \mathbf{p}
\end{aligned}$$

$$IVW = (\mathbf{Ku})^T\delta\mathbf{u} = \mathbf{u}^T\mathbf{K}\delta\mathbf{u} \quad \text{because } \mathbf{K} \text{ is symmetric.}$$

3 Clapeyron's theorem

Clayperon's theorem At static equilibrium, the mean compliance is equal to twice the strain energy.
$MC = 2 * SE$

$EVW = \mathbf{p}^T \delta \mathbf{u} \quad IVW = (\mathbf{K}\mathbf{u}^*)^T \delta \mathbf{u} = \delta \mathbf{u}^T \mathbf{K}\mathbf{u}$

$$EVW = IVW$$
$$\Rightarrow \mathbf{p}^T \delta \mathbf{u} = \delta \mathbf{u}^T \mathbf{K}\mathbf{u}^*$$

Make virtual displacement equal to real equilibrium displacement.

$$\mathbf{p}^T \mathbf{u}^* = \mathbf{u}^* \mathbf{K}\mathbf{u}^* \Rightarrow MC = 2^* SE$$

4 spring or bar in Statics

Many of us have seen the development of spring and bar element stiffness matrices in finite element analysis equilibrium of forces at end nodes and the stiffness relation between axial (normal) forces and relative displacements between the two nodes of the element.

The assembly of elements for the structure is then formed using displacement compatibility (continuity) at connected nodes to write a matrix system representing the nodal equilibrium equations.

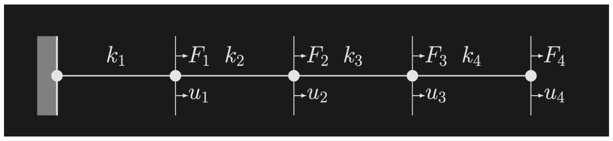


Figure 2: A 4 Springs problem

This direct approach to writing explicit force equilibrium equations expressed at nodes has limited application in practice since the steps require careful freebody diagrams at each node and are only applicable for concentrated forces at nodes.

5 Why?

To handle virtually any complication thrown our way (pun intended), pros turn to the fundamental principle of virtual work to replace the principle of direct force equilibrium. In fact, once you have a firm grasp of this principle, you can brag that you also know the discrete form of the variational equation, which is equivalent to the discrete 'weak form' of the boundary value problem, which, when generalized to a continuum, forms the basis of the finite element method.

But How? As a prelude to the development of the finite element method formulation for structures and solid bodies, it is a good practice to revisit the principle of virtual work in simple settings. This allows us to gain insights into the fundamental principles before diving into the details of the more general matrix structural or finite element analysis.

Let's start with simple examples of linear springs in series, which are models of elastic bar stiffness $k = EA/L$.

6 Begin simple

To begin, what could be easier than a single spring (uniform elastic bar) with stiffness k ?

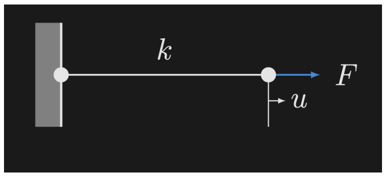


Figure 3: A simple spring problem

- To use the principle of virtual work, we apply a virtual displacement after equilibrium has been reached by full application of the load F. - The virtual work of the load is

$$F \cdot \delta u,$$

while the virtual work of the internal spring force is,

$$F_s \cdot \delta u = ku \cdot \delta u$$

The total virtual work for both external and internal forces, is set to zero for the stationary condition for static equilibrium.

$$\delta V = (\Sigma F) \cdot \delta u = (F - ku) \cdot \delta u = 0$$

Rearranging, the principle of virtual work states that the internal virtual work is balanced by external virtual work,

$$ku \cdot \delta u = F \cdot \delta u$$

For any non-zero δu, this gives the equilibrium equation

$$F - ku = 0, \quad \rightarrow ku = F$$

which is solved for $u = F/k$ at equilibrium.

We will discover the true power of the principle of virtual work when we extend to more than one spring and generalize to a continuum.

7 Two spring in series

Consider two springs/bars connected in series, fixed at the left end, with external forces F_1 and F_2.

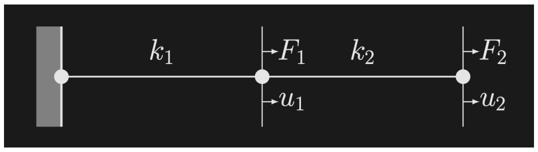


Figure 4: A 2 Springs problem

Our goal is to determine displacements u_1 and u_2 at equilibrium. - At equilibrium, $\delta V = 0$, and the internal and external virtual work are balanced, $\delta V_{\text{internal}} = \delta V_{\text{external}}$:

$$k_1 u_1 \cdot \delta u_1 + k_2 (u_1 - u_2) \cdot \delta u_1 + k_2 (u_2 - u_1) \cdot \delta u_2 = F_1 \delta u_1 + F_2 \delta u_2$$

- Collecting terms multiplied by virtual displacements, we have,

$$\underbrace{(k_1 u_1 + k_2 (u_1 - u_2) - F_1)}_{0} \cdot \delta u_1 + \underbrace{(k_2 (u_2 - u_1) - F_2)}_{0} \cdot \delta u_2 = 0$$

This equation must be satisfied for any nonzero δu_1 and δu_2, and thus the terms in braces must be zero independently, resulting in the two force equilibrium equations associated with nodes 1 and 2:

$$\begin{aligned} (k_1 + k_2)\, u_1 - k_2 u_2 &= F_1 \\ -k_2 u_1 + k_2 u_2 &= F_2 \end{aligned}$$

Given external forces F_1 and F_2, using substitution, we can solve these two coupled simultaneous equations for displacements at equilibrium,

$$\begin{aligned} u_1 &= \frac{1}{k_1}\,(F_1 + F_2) \\ u_2 &= u_1 + \frac{1}{k_2} F_2 = \frac{F_1}{k_1} + \frac{F_2}{k_{\text{eqv}}} \end{aligned}$$

The displacement at the end u_2 can be expressed in terms of the equivalent stiffness k_{eqv} for two springs in series:

$$\frac{1}{k_{\text{eqv}}} = \frac{1}{k_1} + \frac{1}{k_2}, \quad k_{\text{eqv}} = \frac{k_1 k_2}{k_1 + k_2}$$

But How?

To move towards a more systematic approach, let's express the total virtual work at equilibrium in Equation (2) as the inner product of two vectors:

$$\delta V = [\delta u_1, \delta u_2]\, (\underbrace{\left(\begin{matrix} F_1 \\ F_2 \end{matrix} \right\} - \begin{bmatrix} k_1 + k_2 & -k_2 \\ -k_2 & k_2 \end{bmatrix} \left[\begin{matrix} u_1 \\ u_2 \end{matrix} \right\}}_{[0,0]^T}) = 0$$

Since this equation must be satisfied for all non-zero $[\ \delta u_1, \delta u_2]$, this requires the column vector in the braces to be the zero vector, resulting in the linear algebraic matrix system,

$$\underbrace{\begin{bmatrix} k_1 + k_2 & -k_2 \\ -k_2 & k_2 \end{bmatrix} \begin{Bmatrix} u_1 \\ u_2 \end{Bmatrix}}_{\text{Sum of Internal forces}} = \underbrace{\begin{Bmatrix} F_1 \\ F_2 \end{Bmatrix}}_{\text{external forces}}$$

Each row of the stiffness matrix represents the equilibrium of forces balancing internal axial forces at nodes with external nodal forces.
Each column represents the stiffness associated with each nodal displacement.

Expanding the rows in this matrix system, we find that we have the same two equilibrium equations in terms of nodal displacements u_1 and u_2 as expected.

Given F_1, F_2, this system can be solved using linear algebra methods such as inverting the matrix together with matrix-vector multiplication with the applied force vector or Gauss-elimination to eliminate variables until a single equation for a single unknown is obtained, then back-substitution to the solve for remaining unknowns.

The solution of the matrix system for $[u_1, u_2]$ is the same as that obtained by the substitution method in the expanded equations found earlier.

8 Generalizing with a 4 springs problem

Given the system defined hereafter.

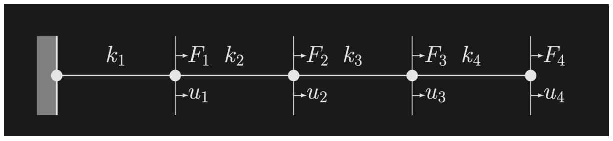


Figure 5: A 4 Springs problem

$$[\delta u_1, \delta u_2, \delta u_3, \delta u_4] \left(\begin{bmatrix} k_1+k_2 & -k_2 & 0 & 0 \\ -k_2 & k_2+k_3 & -k_3 & 0 \\ 0 & -k_3 & k_3+k_4 & -k_4 \\ 0 & 0 & -k_4 & k_4 \end{bmatrix} \begin{Bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \end{Bmatrix} - \begin{Bmatrix} F_1 \\ F_2 \\ F_3 \\ F_4 \end{Bmatrix} \right) =$$

$$\underbrace{\begin{bmatrix} k_1+k_2 & -k_2 & 0 & 0 \\ -k_2 & k_2+k_3 & -k_3 & 0 \\ 0 & -k_3 & k_3+k_4 & -k_4 \\ 0 & 0 & -k_4 & k_4 \end{bmatrix} \begin{Bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \end{Bmatrix}}_{\text{Sum of Internal forces}} = \underbrace{\begin{Bmatrix} F_1 \\ F_2 \\ F_3 \\ F_4 \end{Bmatrix}}_{\text{external forces}}$$

9 Minimizing a potential energy

Since we already know the principle of virtual work, we don't need to use total potential energy to find the equations to solve at equilibrium. However, many of us are comfortable with potential energy from our previous studies of the energy and work on a linear spring in our introductory physics class.

We will see that minimizing the total potential energy is equivalent to the variational equation in the principle of virtual work.

Let's write down the total potential energy for our twospring/bar system, equal to the potential energy of the two springs, and subtract the external work due to loads:

$$\Pi = U - W = \frac{1}{2}k_1 u_1^2 + \frac{1}{2}k_2 \left(u_2 - u_1\right)^2 - F_1 u_1 - F_2 u_2$$

At equilibrium, Π is a minimum. To find this minimum condition, we take the first variation (kind of like a slope) and set this to zero.

$$\delta\Pi = \frac{\partial \Pi}{\partial u_1} \cdot \delta u_1 + \frac{\partial \Pi}{\partial u_2} \delta u_2 = 0$$

Since this holds for any nonzero displacement variations δu_1 and δu_2, we have two equilibrium equations associated with nodes 1 and 2:

$$\frac{\partial \Pi}{\partial u_1} = k_1 u_1 - k_2 \left(u_2 - u_1\right) - F_1 = 0$$
$$\frac{\partial \Pi}{\partial u_2} = k_2 \left(u_2 - u_1\right) - F_2 = 0$$

which are the same two equations as before. The first variation is equal to minus the total virtual work.

The first variation is equal to minus the total virtual work.

$$\delta\Pi = -\delta V = \delta V_{\text{internal}} - \delta V_{\text{external}} = 0$$
$$\delta\Pi = -\delta V = \underbrace{\left[k_1 u_1 - k_2 \left(u_2 - u_1\right) - F_1\right]}_{0} \cdot \delta u_1 + \underbrace{\left[k_2 \left(u_2 - u_1\right) - F_2\right]}_{0} \cdot \delta u_2 = 0$$

We recognize that the displacement variations are equivalent to virtual displacements!

We can also express this stationary condition as equivalent to the principle of virtual work balancing internal with external virtual work due to F_1 and F_2 :

$$\begin{aligned}\delta\Pi = [\ &\underbrace{k_1 u_1 - k_2\left(u_2 - u_1\right)}_{\text{Sum of internal forces at node 1}}\ ]\cdot\delta u_1 \\ +[\ &\underbrace{k_2\left(u_2 - u_1\right)}_{\text{Sum of internal forces at node 2}}\ ]\cdot\delta u_2 = F_1\cdot\delta u_1 + F_2\cdot\delta u_2\end{aligned}$$

To give this a matrix structure, the sum of products of derivatives in the first variation is expressed as the vector inner product,

$$\delta\Pi = [\delta u_1, \delta u_2]\left\{\begin{array}{c}\frac{\partial\Pi}{\partial u_1}\\ \frac{\partial\Pi}{\partial u_2}\end{array}\right\}$$

$$\Pi = \frac{1}{2}[u_1, u_2]\left[\begin{array}{cc}k_1+k_2 & -k_2\\ -k_2 & k_2\end{array}\right]\left\{\begin{array}{c}u_1\\ u_2\end{array}\right\} - [u_1, u_2]\left\{\begin{array}{c}F_1\\ F_2\end{array}\right\}$$

Define column vectors of nodal displacements and virtual displacements (variations):

$$\boldsymbol{u} = \left\{\begin{array}{c}u_1\\ u_2\end{array}\right\}, \quad \delta\boldsymbol{u} = \left\{\begin{array}{c}\delta u_1\\ \delta u_2\end{array}\right\}$$

with row vectors defined by the matrix transpose,

$$\boldsymbol{u}^T = [u_1, u_2], \quad \delta\boldsymbol{u}^T = [\delta u_1, \delta u_2]$$

$$\Pi = \frac{1}{2}\boldsymbol{u}^T\boldsymbol{K}\boldsymbol{u} - \boldsymbol{u}^T\boldsymbol{F}$$

The vector of partial derivatives can then express the vector form of equilibrium equations:

$$\frac{\partial\Pi}{\partial\boldsymbol{u}} = \left\{\begin{array}{c}\frac{\partial}{\partial u_1}\\ \frac{\partial}{\partial u_2}\end{array}\right\}\Pi = \left\{\begin{array}{c}\frac{\partial\Pi}{\partial u_1}\\ \frac{\partial\Pi}{\partial u_2}\end{array}\right\} = \boldsymbol{K}\boldsymbol{u} - \boldsymbol{F} = \left[\begin{array}{c}0\\ 0\end{array}\right]$$

Rearranging, we have the stiffness equations

$$Ku = \boldsymbol{F}$$

Remember, the principle of virtual work for structures and solids is the same as the weak form used as the basis for the finite element method.

10 References

- Energy Methods in Applied Mechanics, H.L. Langhaar, 1962.
- Advanced Mechanics of Materials, Robert D. Cook and Warren C. Young. 1999.
- Structural Optimization: Size, Shape, and Topology, G. K. Ananthasuresh 10

1MAE701 - CSM: THE RAYLEIGH RITZ METHOD FOR BARS UNDER AXIAL LOADS

Prof. Joseph Morlier

October 9, 2024

LEARNING OUTCOMES

Describe the steps required to find an approximate solution for a bar system using the Rayleigh Ritz (RR) method:

- Step1: Assume a displacement function, apply the BC.
- Step 2: Write the expression for the PE of the system.
- Step 3: Find the minimizers of the PE of the system.

Employ the RR method to compute an approximate solution for the displacement in a beam under axial load.
Differentiate between the requirement for an approximate solution and an exact solution.

1 RR?

The Rayleigh Ritz method is a classical approximate method to find the displacement function of an object such that the it is in equilibrium with the externally applied loads. It is regarded as an ancestor of the widely used Finite Element Method (FEM). The Rayleigh Ritz method relies on the principle of minimum potential energy for conservative systems. The method involves assuming a form or a shape for the unknown displacement functions, and thus, the displacement functions would have a few unknown parameters. These assumed shape functions are termed Trial Functions. Afterwards, the potential energy function of the system is written in terms of those few parameters, and the values of those parameters that would minimize the potential energy function of the system are calculated. Pythonlly speaking, we assume that the unknown displacement function u is a member of a certain space of functions; for example, we can assume that $u \in V$ where V is the set of all the possible vector valued linear functions defined on the body represented by the set $\Omega_0 \subset \mathbf{R}^3$: $V = \left\{u : \Omega_0 \to \mathbf{R}^3 \mid \forall x \in \Omega_0 : u(x) = Bx, B \in \mathbf{M}^3\right\}$. With this assumption, we restrict the possible solutions to those in the form $u = Bx$, and thus, the unknown parameters are restricted to the matrix B, which in this particular example, has nine components. Finally, the nine components that would minimize the potential-energy function are obtained. The method will be illustrated using various one-dimensional examples.

2 BARS UNDER AXIAL LOADS?

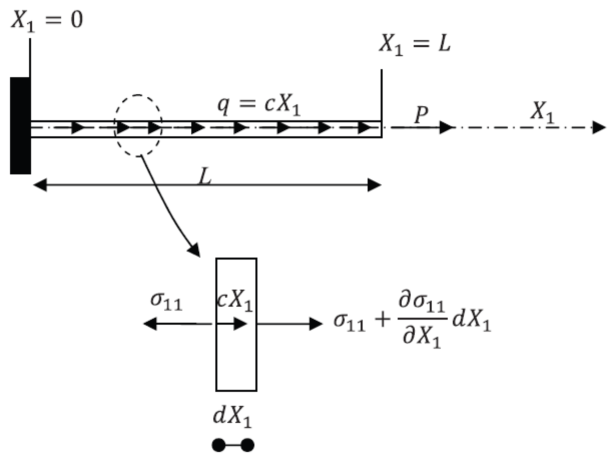
The Rayleigh Ritz method seeks to find an approximate solution to minimize the potential energy of the system. For plane bars under axial loading, the unknown displacement function is u_1. In this case, the potential energy of the system has the following form (See beams under axial loads and energy expressions):

$$PE = \text{ Total Strain Energy } - W = \int \frac{EA}{2}\left(\frac{du_1}{dX_1}\right)^2 dX_1 - W$$

where E is Young's modulus, A is the cross sectional area, and W is the work done by all the external forces acting on the bar (concentrated forces and distributed loads). To obtain an approximate solution for u_1, a trial function of a finite number of unknowns is first selected. Then, this trial function has to satisfy the "essential" boundary conditions, which are those boundary conditions that would satisfy the physical stability of the system. In the case of bars under axial loads, the trial function u_1 has to satisfy the displacement boundary conditions. The following examples show the method for different external loads, and bar configurations.

2.1 Let's take an example

Let a bar with a length L and a constant cross sectional area A be aligned with the coordinate axis X_1. Assume that the bar is subjected to a horizontal body force (in the direction of the axis X_1) $p = CX_1$ units of force/unit length where C is a known constant. Also, assume that the bar has a constant force of value P applied at the end $X_1 = L$. If the bar is fixed at the end $X_1 = 0$, find the displacement of the bar by directly solving the differential equation of equilibrium. Also, find the displacement function using the Rayleigh Ritz method assuming a polynomial function for the displacement of the degrees $(0, 1, 2, 3,$ and $4)$. Assume that the bar is linear elastic with Young's modulus E and that the small strain tensor is the appropriate measure of strain. Ignore the effect of Poisson's ratio.



2.2 Exact solution?

First, the exact solution that would satisfy the equilibrium equations can be obtained. The equilibrium equation as shown in the beams under axial loading section when E and A are constant is:

$$\frac{d^2u_1}{dX_1^2} = -\frac{CX_1}{EA}$$

Therefore, the exact solution has the form:

$$u_1 = -\frac{CX_1^3}{6EA} + C_1X_1 + C_2$$

where C_1 and C_2 can be obtained from the boundary conditions:

$$@X_1 = 0 : u_1 = 0 \Rightarrow C_2 = 0$$

$$@X_1 = L : \sigma_{11} = E\frac{du_1}{dX_1} = \frac{P}{A} \Rightarrow C_1 = \frac{P}{AE} + \frac{CL^2}{2EA}$$

Therefore, the exact solution for the displacement u_1 and the stress σ_{11} are:

$$u_1 = -\frac{CX_1^3}{6FA} + \left(\frac{P}{AF} + \frac{CL^2}{2FA}\right) X_1$$
$$\sigma_{11} = E\frac{du_1}{dX_1} = -\frac{CX_1^2}{2A} + \left(\frac{P}{A} + \frac{CL^2}{2A}\right)$$

2.3 Check with Python?

```python
from sympy import *
import sympy as sp
sp.init_printing(use_latex = "mathjax")
x, A, E , C, L, P = symbols("x A E C L P")
u = Function("u")
u1 = u(x).subs(x,0)
u2 = u(x).diff(x).subs(x,L)
a = dsolve(u(x).diff(x,2) + C*x/(E*A), u(x), ics = {u1:0, u2:P/(E*A)})
u = a.rhs
sigma = E * u.diff(x)
display("displacement and stress: ", u, simplify(sigma))
```

Listing 1: Python example

3 But How do you do RR in practice?

The first step in the Rayleigh Ritz method finds the minimizer of the potential energy of the system which can be written as:

$$PE = \int \bar{U} dX - W = \int_0^L \frac{\sigma_{11}\varepsilon_{11}}{2} A dX_1 - Pu_1|_{X_1=L} - \int_0^L pu_1 dX_1$$

Notice that the potential energy lost by the action of the end force P is equal to the product of P and the displacement u_1 evaluated at $X_1 = L$ while the potential energy lost by the action of the distributed body forces is an integral since it acts on each point along the beam length. Further, we will use the constitutive equation to rewrite the potential energy function in terms of the function u_1 :

$$PE = \int_0^L \frac{EA}{2}\left(\frac{du_1}{dX_1}\right)^2 dX_1 - Pu_1|_{X_1=L} - \int_0^L CX_1u_1 dX_1$$

To find an approximate solution, an assumption for the shape or the form of u_1 has to be introduced.

Polynomial of the Zero Degree: First, we can assume that u_1 is a constant function (a polynomial of the zero degree):

$$u_1 = a_0$$

However, to satisfy the "essential" boundary condition that $u_1 = 0$ when $X_1 = 0$, the constant a_0 has to equal to zero; this leads to the trivial solution $u_1 = 0$, which will automatically be rejected.
Polynomial of the First Degree: As a more refined approximation, we can assume that u_1 is a linear function (a polynomial of the first degree):

$$u_1 = a_0 + a_1 X_1$$

To satisfy the essential boundary condition $@X_1 = 0 : u_1 = 0$ the coefficient a_0 is equal to zero. Therefore:

$$u_1 = a_1 X_1$$

The strain component ε_{11} can be computed as follows:

$$\varepsilon_{11} = \frac{du_1}{dX_1} = a_1$$

Substituting into the equation for the potential energy of the system:

$$PE = \int_0^L \frac{EA}{2} a_1^2 dX_1 - Pa_1 L - \int_0^L CX_1 a_1 X_1 dX_1 = \frac{EAL}{2} a_1^2 - Pa_1 L - \frac{Ca_1 L^3}{3}$$

The only variable that can be controlled to minimize the potential energy of the system is a_1. Therefore:

$$\frac{dPE}{da_1} = 0 \Rightarrow EAa_1 L - PL - \frac{CL^3}{3} = 0 \Rightarrow a_1 = \frac{P + \frac{CL^2}{3}}{EA}$$

Therefore, the best linear function according to the Rayleigh Ritz method is:

$$u_1 = \left(\frac{P + \frac{CL^2}{3}}{EA}\right) X_1$$

A linear displacement would produce a constant stress:

$$\sigma_{11} = \left(\frac{P + \frac{CL^2}{3}}{A}\right)$$

Unfortunately, this solution is far from being accurate, since it does not satisfy the differential equation of equilibrium (Why?). However, within the only possible or allowed linear shapes, the obtained solution is the minimizer of the potential energy.

Polynomial of the Second Degree:
As a more refined approximation, we can assume that u_1 is a polynomial of the second degree:

$$u_1 = a_0 + a_1 X_1 + a_2 X_1^2$$

To satisfy the essential boundary condition $@X_1 = 0 : u_1 = 0$ the coefficient a_0 is equal to zero. Therefore:

$$u_1 = a_1 X_1 + a_2 X_1^2$$

The strain component ε_{11} can be computed as follows:

$$\varepsilon_{11} = \frac{\mathrm{d}u_1}{\mathrm{d}X_1} = a_1 + 2a_2 X_1$$

Substituting into the equation for the potential energy of the system:

$$\begin{aligned} PE &= \int_0^L \frac{EA}{2}\left(a_1 + 2a_2X_1\right)^2 \,\mathrm{d}X_1 - P\left(a_1L + a_2L^2\right) - \int_0^L C\left(a_1X_1^2 + a_2X_1^3\right)\mathrm{d}X_1 \\ &= \frac{2EAL^3}{3}a_2^2 + EAL^2a_2a_1 + \frac{EAL}{2}a_1^2 - PL^2a_2 - PLa_1 - \frac{CL^4}{4}a_2 - \frac{CL^3}{3}a_1 \end{aligned}$$

a_1 and a_2 are the variables that can be controlled to minimize the potential energy of the system. Therefore:

$$\frac{\partial PE}{\partial a_1} = 0 \Rightarrow EAL^2a_2 + EALa_1 - PL - \frac{CL^3}{3} = 0$$

and

$$\frac{\partial PE}{\partial a_2} = 0 \Rightarrow \frac{4EAL^3}{3}a_2 + EAL^2a_1 - PL^2 - \frac{CL^4}{4} = 0$$

Solving the above two equations yields:

$$a_1 = \frac{7CL^2 + 12P}{12EA} \qquad a_2 = -\frac{CL}{4EA}$$

Therefore, the best second degree polynomial according to the Rayleigh Ritz method is:

$$u_1 = -\frac{cL}{4EA}X_1^2 + \frac{\left(7CL^2 + 12P\right)}{12EA}X_1$$

A parabolic displacement would produce a linear stress:

$$\sigma_{11} = -\frac{cL}{2A}X_1 + \frac{\left(7CL^2 + 12P\right)}{12A}$$

3.1 Check with Python?

```python
from sympy import *
import sympy as sp
sp.init_printing(use_latex = "mathjax")
x, a1, a2, E, A, P, c = symbols("x a_1 a_2 E A P c")
u = a2*x**2+a1*x
uL = u.subs(x,L)
PE = integrate((1/2)*E*A*(u.diff(x)**2), (x,0,L)) - P*uL - integrate(c*x*u, (x,0,L))
display("Potential Energy: ", PE)
Eq1 = PE.diff(a2)
Eq2 = PE.diff(a1)
display("Minimize PE: ", Eq1, Eq2)
s = solve((Eq1, Eq2), (a2, a1))
display("Solve: ", s)
u = u.subs({a1:s[a1], a2:s[a2]})
```

```
display("Best second degree Polynomial(Rayleigh Ritz method): ", u)
stress = E * u.diff(x)
display("stress: ", simplify(stress))
```

Listing 2: Python example

3.2 Why choosing a 3rd order?

Polynomial of the Third Degree:
Similarly, choosing a polynomial of the third degree as an approximate displacement function means:

$$u_1 = a_0 + a_1 X_1 + a_2 X_1^2 + a_3 X_1^3$$

with the following approximation for the axial strain

$$\varepsilon_{11} = a_1 + 2a_2 X_1 + 3a_3 X_1^2$$

The essential boundary condition lead to $a_0 = 0$. The potential energy of the system has the form:

$$PE = \int_0^L \frac{EA}{2} \left(a_1 + 2a_2 X_1 + 3a_3 X_1^2\right)^2 \, \mathrm{d}X_1 - P\left(a_1 L + a_2 L^2 + a_3 L^3\right) - \int_0^L C\left(a_1 X_1^2 + a_2 X_1^3 + a_3 X_1^4\right) \mathrm{d}X_1$$

Clearly, the method generates a large set of equations that are difficult to solve by hand. Using Python we can find the values of the constants a_1, a_2, and a_3 that would minimize the potential energy of the system:

3.3 Check with Python?

```
from sympy import *
import sympy as sp
sp.init_printing(use_latex = "mathjax")
x, a1, a2, a3, E, A, P, c = symbols("x a_1 a_2 a_3 E A P c")
u = a3*x**3+a2*x**2+a1*x
uL = u.subs(x,L)
PE = integrate((1/2)*E*A*(u.diff(x)**2), (x,0,L)) - P*uL - integrate(c*x*u, (x,0,L))
display("Potential Energy: ", PE)
Eq1 = PE.diff(a2)
Eq2 = PE.diff(a1)
Eq3 = PE.diff(a3)
display("Minimize PE: ", Eq1, Eq2, Eq3)
s = solve((Eq1, Eq2, Eq3), (a2, a1, a3))
display("Solve: ", s)
u = u.subs({a1:s[a1], a2:s[a2], a3:s[a3]})
display("Best third degree Polynomial(Rayleigh Ritz method): ", u)
stress = E * u.diff(x)
display("stress: ", simplify(stress))
```

Listing 3: Python example

Since the assumed trial function is a polynomial of the third degree, which is similar to the exact solution, the Rayleigh Ritz method is able to produce the exact solution for the displacement and the stresses:

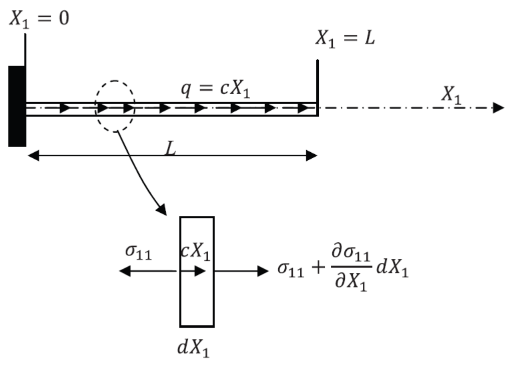
$$u_1 = -\frac{CX_1^3}{6EA} + \left(\frac{P}{AE} + \frac{CL^2}{2EA}\right) X_1$$

$$\sigma_{11} = E\frac{\mathrm{d}u_1}{\mathrm{d}X_1} = -\frac{CX_1^2}{2A} + \left(\frac{P}{A} + \frac{CL^2}{2A}\right)$$

4 It's up to you to solve this simple exercise

Using a polynomial trial function of the third degree, find the displacement function of the shown bar using the RR method. Assume that the bar is linear elastic with Young's modulus E and cross-sectional area A and that the small strain tensor is the appropriate measure of strain. Ignore the effect of Poisson's ratio.

NB: AT THE END ONLY, please Check your new results with previous results with $P = 0$ in the previous exercise !



$X_1 = 0$
$X_1 = L$
$q = cX_1$
X_1
L
σ_{11}
cX_1
$\sigma_{11} + \frac{\partial \sigma_{11}}{\partial X_1} dX_1$
dX_1

4.1 EXACT SOLUTION?

The exact solution can be obtained by directly solving the differential equation of equilibrium utilizing $\sigma_{11} = E\frac{\mathrm{d}u_1}{\mathrm{d}X_1}$:

$$\frac{\mathrm{d}^2 u_1}{\mathrm{d}X_1^2} = -\frac{cX_1}{EA}$$

with the boundary conditions: $@X_1 = 0 : u_1 = 0$ and $X_1 = L : \frac{\mathrm{d}u_1}{\mathrm{d}X_1} = 0$. Therefore:

$$u_1 = \frac{cL^2}{2EA}X_1 - \frac{c}{6EA}X_1^3$$

4.2 Check with Python?

```python
from sympy import *
import sympy as sp
sp.init_printing(use_latex = "mathjax")
u, c, EA, x, L = symbols("u c EA x L")
u = Function("u")
u1 = u(x).subs(x,0)
u2 = u(x).diff(x).subs(x,L)
sol = dsolve(u(x).diff(x,2)+c*x/EA, u(x), ics = {u1:0, u2:0})
display(sol)
```

Listing 4: Python example

4.3 Again the 3rd order?

Polynomial of the Third Degree:
Similarly, choosing a polynomial of the third degree as an approximate displacement function means:

$$u_1 = a_0 + a_1 X_1 + a_2 X_1^2 + a_3 X_1^3$$

with the following approximation for the axial strain

$$\varepsilon_{11} = a_1 + 2a_2 X_1 + 3a_3 X_1^2$$

The essential boundary condition lead to $a_0 = 0$. The potential energy of the system has the form:

$$PE = \int_0^L \frac{EA}{2}\left(a_1 + 2a_2 X_1 + 3a_3 X_1^2\right)^2 \,\mathrm{d}X_1 - \int_0^L C\left(a_1 X_1^2 + a_2 X_1^3 + a_3 X_1^4\right)\mathrm{d}X_1$$

Clearly, the method generates a large set of equations that are difficult to solve by hand. Using Python we can find the values of the constants a_1, a_2, and a_3 that would minimize the potential energy of the system:

4.4 Modify Previous Python code?

```python
sp.init_printing(use_latex = "mathjax")
x, a1, a2, a3, E, A, c = symbols("x a_1 a_2 a_3 E A c")
u = a3*x**3+a2*x**2+a1*x
uL = u.subs(x,L)
PE = integrate((1/2)*E*A*(u.diff(x)**2), (x,0,L)) - integrate(c*x*u, (x,0,L))
display("Potential Energy: ", PE)
Eq1 = PE.diff(a2)
Eq2 = PE.diff(a1)
Eq3 = PE.diff(a3)
display("Minimize PE: ", Eq1, Eq2, Eq3)
s = solve((Eq1, Eq2, Eq3), (a2, a1, a3))
display("Solve: ", s)
u = u.subs({a1:s[a1], a2:s[a2], a3:s[a3]})
display("Best third degree Polynomial(Rayleigh Ritz method): ", u)
stress = E * u.diff(x)
display("stress: ", simplify(stress))
```

Listing 5: Python example

To find the unknowns a_1, a_2, and a_3, three equations are formed with $W_1 = \phi_1(X_1) = X_1, W_2 = \phi_2(X_1) = X_1^2$, and $W_3 = \phi_3(X_1) = X_1^3$. Solving those three equations the following approximate solution is obtained:

$$u_{\text{approx}} = \frac{cL^2}{2EA}X_1 - \frac{c}{6EA}X_1^3$$

Notice that since the exact solution (a polynomial of the third degree) is an element of the space of possible approximate solutions, the RR method yields the exact solution!

5 References

- Adeeb, S. (2011). Introduction to solid mechanics and finite element analysis using Mathematica. Kendall Hunt.

1MAE701 - CSM: FEA IN ONE DIMENSION FOR BARS UNDER AXIAL LOADS

Prof. Joseph Morlier

October 9, 2024

LEARNING OUTCOMES

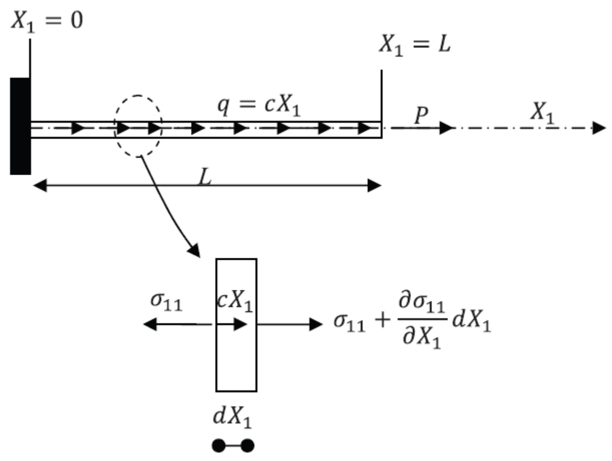
Describe the steps required to find an approximate solution for a bar system using the FEA method:

- Galerkin is Virtual Work
- By expressing VW you can build the stiffness matrix K
- Knowing BCs and loads f you can inverse to find the displacement u

Employ the FEA method to compute an approximate solution for the displacement in a beam under axial load.
Differentiate between the requirement for an approximate solution and an exact solution.

1 FEA?

To illustrate the finite element method, we will start by solving the same example that was solved before using the Galerkin method but employing a finite element approximation. Using a four-piecewise linear trial function, find the approximate displacement function of the shown bar. Assume that the bar is linear elastic with Young's modulus E and cross-sectional area A and that the small strain tensor is the appropriate measure of strain. Ignore the effect of Poisson's ratio.



The different aspects of the finite element analysis method will be presented through solving this example as follows.

2 SHAPE FUNCTIONS?

In this example, a piecewise C_0 linear trial function for the displacement that satisfies the essential boundary condition $@X_1 = 0 : u = 0$ is imposed. The displacement is interpolated between the four nodes and thus has the following form:

$$u = u_1 N_1 + u_2 N_2 + u_3 N_3 + u_4 N_4$$

where $\forall i \leq 4 : u_i$ are multipliers that happen to be the displacements at the nodes $1, 2, 3,$ and 4 respectively, while N_i are the interpolation functions that have a value of 1 at node i and decrease linearly to zero at the neighbouring nodes (Figure 1). These functions are traditionally termed the "hat" functions, for obvious reasons. They are also referred to as the shape functions since they define the shape of the deformation of the model. Except when they vanish, the shape functions have the form:

$$N_1 = \begin{cases} \frac{4X_1}{L} & 0 \leq X_1 \leq \frac{L}{4} \\ \frac{4}{L}\left(\frac{L}{2} - X_1\right) & \frac{L}{4} \leq X_1 \leq \frac{L}{2} \end{cases}$$

$$N_2 = \begin{cases} \frac{4}{L}\left(X_1 - \frac{L}{4}\right) & \frac{L}{4} \leq X_1 \leq \frac{L}{2} \\ \frac{4}{L}\left(\frac{3L}{4} - X_1\right) & \frac{L}{2} \leq X_1 \leq \frac{3L}{4} \end{cases}$$

$$N_3 = \begin{cases} \frac{4}{L}\left(X_1 - \frac{L}{2}\right) & \frac{L}{2} \leq X_1 \leq \frac{3L}{4} \\ \frac{4}{L}\left(L - X_1\right) & \frac{3L}{4} \leq X_1 \leq L \end{cases}$$

$$N_4 = \frac{4}{L}\left(X_1 - \frac{3L}{4}\right)$$

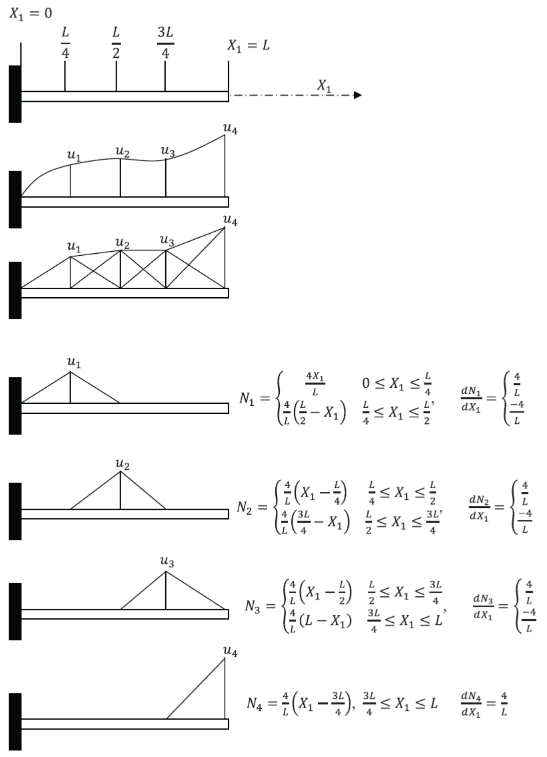

$X_1 = 0$
$\frac{L}{4}$
$\frac{L}{2}$
$\frac{3L}{4}$
$X_1 = L$
X_1
u_1
u_2
u_3
u_4
$N_1 = \begin{cases} \frac{4X_1}{L} & 0 \leq X_1 \leq \frac{L}{4} \\ \frac{4}{L}\left(\frac{L}{2} - X_1\right) & \frac{L}{4} \leq X_1 \leq \frac{L}{2} \end{cases}, \quad \frac{dN_1}{dX_1} = \begin{cases} \frac{4}{L} \\ \frac{-4}{L} \end{cases}$
$N_2 = \begin{cases} \frac{4}{L}\left(X_1 - \frac{L}{4}\right) & \frac{L}{4} \leq X_1 \leq \frac{L}{2} \\ \frac{4}{L}\left(\frac{3L}{4} - X_1\right) & \frac{L}{2} \leq X_1 \leq \frac{3L}{4} \end{cases}, \quad \frac{dN_2}{dX_1} = \begin{cases} \frac{4}{L} \\ \frac{-4}{L} \end{cases}$
$N_3 = \begin{cases} \frac{4}{L}\left(X_1 - \frac{L}{2}\right) & \frac{L}{2} \leq X_1 \leq \frac{3L}{4} \\ \frac{4}{L}\left(L - X_1\right) & \frac{3L}{4} \leq X_1 \leq L \end{cases}, \quad \frac{dN_3}{dX_1} = \begin{cases} \frac{4}{L} \\ \frac{-4}{L} \end{cases}$
$N_4 = \frac{4}{L}\left(X_1 - \frac{3L}{4}\right), \ \frac{3L}{4} \leq X_1 \leq L \quad \frac{dN_4}{dX_1} = \frac{4}{L}$

Figure 1: One dimensional piecewise linear interpolation functions or "hat" functions

The derivatives of the shape functions are very simple and have the form:

$$\frac{dN_1}{dX_1} = \begin{cases} \frac{4}{L} & 0 \leq X_1 \leq \frac{L}{4} \\ \frac{-4}{L} & \frac{L}{4} \leq X_1 \leq \frac{L}{2} \end{cases}$$
$$\frac{dN_2}{dX_1} = \begin{cases} \frac{4}{L} & \frac{L}{4} \leq X_1 \leq \frac{L}{2} \\ \frac{-4}{L} & \frac{L}{2} \leq X_1 \leq \frac{3L}{4} \end{cases}$$
$$\frac{dN_3}{dX_1} = \begin{cases} \frac{4}{L} & \frac{L}{2} \leq X_1 \leq \frac{3L}{4} \\ \frac{-4}{L} & \frac{3L}{4} \leq X_1 \leq L \end{cases}$$
$$\frac{dN_4}{dX_1} = \frac{4}{L}$$

3 Exact solution?

The exact solution as shown previously is known and is equal to:

$$u_{\text{exact}} = \frac{cL^2}{2EA}X_1 - \frac{c}{6EA}X_1^3$$

The exact displacement at the points 1, 2, 3, and 4 are given by:

$$u_{\text{exact }1} = \frac{47cL^3}{384F_iA}$$
$$u_{\text{exact }2} = \frac{11cL^3}{48EA}$$
$$u_{\text{exact }3} = \frac{39cL^3}{128EA}$$
$$u_{\text{exact }4} = \frac{cL^3}{3F/A}$$

3.1 Check with Python?

```python
import sympy as sp
import numpy as np
from sympy import lambdify
from matplotlib import pyplot as plt
x, c, L, EA = sp.symbols("x c L EA")
xL = np.arange(0,1,0.01)
#c,L,EA = 1,1,1
# Exact Calculations
u_exact = c*L**2/2/EA*x - c/sp.Rational("6")/EA*x**3
s_exact = u_exact.diff(x)
display("u_exact(x) =",u_exact,"s_exact(x) =",s_exact)
F = lambdify(x,u_exact)
dF = lambdify(x,s_exact)
y_exact = F(xL)
sy_exact = dF(xL)

u1 = c*L**3/EA*(47/384)
u2 = c*L**3/EA*(11/48)
u3 = c*L**3/EA*(39/128)
u4 = c*L**3/EA*(1/3)
```

Listing 1: Python example

4 But How do you do FEA in practice?

As stated previously, the Galerkin method and the virtual work method are equivalent. The principle of virtual work states that, from a state of equilibrium, when a virtual admissible (compatible) displacement is applied, the increment in the work done by the external and body forces during the application of the virtual displacement is equal to the increment in the deformation energy stored. We will adopt the traditional matrix multiplication convention with differentiation between row and column vectors. The displacement function has the form:

$$u = u_1N_1 + u_2N_2 + u_3N_3 + u_4N_4 =< \quad N_1 \quad N_2 \quad N_3 \quad N_4 \quad > \begin{pmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \end{pmatrix}$$

The stress component σ_{11} due to the assumed trial displacement function has the following form:

$$\sigma_{11} = E\varepsilon_1 = E\frac{du}{dX_1} = E\sum_{i=1}^{4} u_i \frac{dN_i}{dX_1}$$
$$= E < \frac{dN_1}{dX_1}\frac{dN_2}{dX_1}\frac{dN_3}{dX_1}\frac{dN_4}{dX_1} > \begin{pmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \end{pmatrix}$$

The virtual displacement can be set by choosing virtual parameters u_i^* :

$$u^* = u_1^*N_1 + u_2^*N_2 + u_3^*N_3 + u_4^*N_4$$

Setting:

$$N = \langle \quad N_1 \quad N_2 \quad N_3 \quad N_4 \quad >$$
$$B =< \frac{dN_1}{dX_1}\frac{dN_2}{dX_1}\frac{dN_3}{dX_1}\frac{dN_4}{dX_1} >$$
$$u_e = \begin{pmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \end{pmatrix}$$
$$u_e^* =< u_1^*u_2^*u_3^*u_4^* >$$

then the equations can be rewritten as follows:

$$u = Nu_e$$
$$u^* = u_e^*N^T$$
$$\varepsilon_{11} = Bu_e$$
$$\varepsilon_{11}^* = u_e^*B^T$$
$$\sigma_{11} = EBu_e$$

The internal virtual work has the form:

$$IVW = \int_0^L A\varepsilon_{11}^*\sigma_{11}dX_1 = u_e^*\int_0^L EAB^TBdX_1u_e$$

The external virtual work has the form:

$$EVW = u_e^*\int_0^L cX_1N^TdX_1$$

Equating the internal (1) and external virtual work (2), the following is obtained:

$$< u_1^* u_2^* u_3^* u_4^* > \begin{pmatrix} \int_0^{\frac{L}{2}} \frac{dN_1}{dX_1}\frac{dN_1}{dX_1}dX_1 & \int_{\frac{L}{2}}^{\frac{L}{2}} \frac{dN_1}{dX_1}\frac{dN_2}{dX_1}dX_1 & 0 & 0 \\ \int_{\frac{L}{4}}^{\frac{3L}{4}} \frac{dN_2}{dX_1}\frac{dN_1}{dX_1}dX_1 & \int_{\frac{4}{4}}^{\frac{3L}{4}} \frac{dN_2}{dX_1}\frac{dN_2}{dX_1}dX_1 & \int_{\frac{L}{4}}^{\frac{3L}{4}} \frac{dN_2}{dX_1}\frac{dN_3}{dX_1}dX_1 & 0 \\ 0 & \int_{\frac{L}{2}}^{\frac{3L}{4}} \frac{dN_3}{dX_1}\frac{dN_2}{dX_1}dX_1 & \int_{\frac{L}{2}}^{L} \frac{dN_3}{dX_1}\frac{dN_3}{dX_1}dX_1 & \int_{\frac{3L}{4}}^{L} \frac{dN_3}{dX_1}\frac{dN_4}{dX_1}dX_1 \\ 0 & 0 & \int_{\frac{3L}{4}}^{L} \frac{dN_4}{dX_1}\frac{dN_3}{dX_1}dX_1 & \int_{\frac{3L}{L}}^{\frac{dN_4}{dX_1}\frac{dN_4}{dX_1}dX_1} \end{pmatrix} \begin{pmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \end{pmatrix} \tag{1}$$

$$= \frac{1}{EA} < u_1^* u_2^* u_3^* u_4^* > \begin{pmatrix} f_1 \\ f_2 \\ f_3 \\ f_4 \end{pmatrix} \tag{2}$$

where $\forall i \leq 4 : f_i = \int_0^L N_i c X_1 \, \mathrm{d}X_1$. Note that the factor EA is constant in this problem and was pulled out as a common factor for simplicity. Since the coefficients u_i^* are arbitrary, the coefficients of each u_i^* on both sides of the equality are equal. Therefore, the same equations of the Galerkin method are retrieved.

4.1 ELEMENTS, NODES AND DEGREES OF FREEDOM

In the finite element method, the domain is discretized into points which are called nodes and the regions between the nodes are called elements. The approximate solution involves assuming a trial function for the displacements interpolated between the values of the displacements the nodes. The displacement function is formed by multiplying the displacements of the nodes by shape functions. The displacements of the nodes are termed the degrees of freedom of the system. The right-hand side of the matrix equation formed is equivalent to a nodal force vector and is formed by lumping the distributed load and forming equivalent concentrated loads at the nodes.

For this particular example the solution is exact at the selected nodes. However, the solution within the elements will be linearly interpolated between the two values on each side and thus is not exact. The value of the gradient of the displacement at each node does not exist, as the displacement function is not differentiable at the nodes. In the finite element analysis method, the stresses that are related to the strains (the gradients of the displacements) are usually averaged at the nodes, and the degree of variation of the stresses between the different elements could be an indication of the accuracy of the solution. The matrix multiplied by the degrees of freedom is, in conservative systems, symmetric and is termed the stiffness matrix.

When the virtual work method was applied, the integration was done on the whole domain. However, it is possible to perform the integration locally on the elements. The equations of the virtual work can be written as follows:

$$\int_0^{\frac{L}{4}} A\varepsilon_{11}^*\sigma_{11}dX_1 + \int_{\frac{L}{4}}^{\frac{L}{2}} A\varepsilon_{11}^*\sigma_{11}dX_1 + \int_{\frac{L}{2}}^{\frac{3L}{4}} A\varepsilon_{11}^*\sigma_{11}dX_1 + \int_{\frac{3L}{4}}^{L} A\varepsilon_{11}^*\sigma_{11}dX_1$$
$$= \int_0^{\frac{L}{4}} u^* c X_1 dX_1 + \int_{\frac{L}{4}}^{\frac{L}{2}} u^* c X_1 dX_1 + \int_{\frac{L}{2}}^{\frac{3L}{4}} u^* c X_1 dX_1 + \int_{\frac{3L}{4}}^{L} u^* c X_1 dX_1$$

Because the integration can be done locally on the elements, it is possible to isolate the elements and perform the integration on each element separately (Figure 4). On each element, the shape functions or interpolation functions are associated with the end nodes. For example, element e 2 connects nodes 1 and 2 with two shape functions, N_1 and N_2. What is remarkable about the method is the similarity between all the elements shown in (Figure 3). Thus, a local stiffness matrix for each element can be developed, and then, the global stiffness matrix can be easily assembled by combining all the local stiffness matrices.

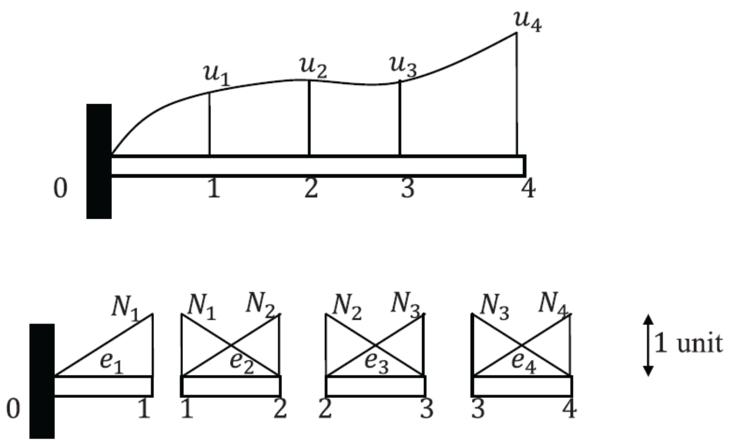


Figure 2: Discretizing the domain into elements and nodes

4.2 Element1

On element 1, the displacement is given by: $u = u_1N_1$ and the virtual displacement is given by $u^* = u_1^*N_1$. By considering the first element, the first term (denoted IVW_1) in the internal virtual work equation has the following form:

$$IVW_1 = u_1^* \int_0^{\frac{L}{4}} EA\frac{dN_1}{dX_1}\frac{dN_1}{dX_1}dX_1u_1 = u_1^*k_{11}^{e1}u_1$$

K_{11}^{e1} is the local stiffness entry for element 1 corresponding to the degree of freedom u_1. The first term in the external virtual work has the form:

$$EVW_1 = u_1^* \int_0^{\frac{L}{4}} N_1cX_1dX_1 = u_1^*f_1^{e1}$$

where f_1^{e1} is the nodal force to be applied at node 1 corresponding to lumping the distributed load acting on element 1.

4.3 Element2

On element 2, the displacement is given by: $u = u_1N_1 + u_2N_2$ and the virtual displacement is given by $u^* = u_1^*N_1 + u_2^*N_2$. By considering the second element, the second term in the internal virtual work equation has the following form:

$$\begin{aligned} IVW_2 &=< u_1^*u_2^* > \begin{pmatrix} \int_{\frac{L}{4}}^{\frac{L}{2}} EA\frac{dN_1}{dX_1}\frac{dN_1}{dX_1}dX_1 & \int_{\frac{L}{4}}^{\frac{L}{2}} EA\frac{dN_1}{dX_1}\frac{dN_2}{dX_1}dX_1 \\ \int_{\frac{L}{4}}^{\frac{L}{2}} EA\frac{dN_2}{dX_1}\frac{dN_1}{dX_1}dX_1 & \int_{\frac{L}{4}}^{\frac{L}{2}} EA\frac{dN_2}{dX_1}\frac{dN_2}{dX_1}dX_1 \end{pmatrix} \begin{pmatrix} u_1 \\ u_2 \end{pmatrix} \\ &=< u_1^*u_2^* > \begin{pmatrix} K_{11}^{e2} & K_{12}^{e2} \\ K_{21}^{e2} & K_{22}^{e2} \end{pmatrix} \begin{pmatrix} u_1 \\ u_2 \end{pmatrix} \end{aligned}$$

where K_{ij}^{e2} are the local stiffness matrix entries for element 2 corresponding to the degrees of freedom u_1 and u_2. The second term in the external virtual work has the form:

$$\begin{aligned} EVW_2 &= < u_1^*u_2^* > \begin{pmatrix} \int_{\frac{L}{4}}^{\frac{L}{2}} N_1cX_1dX_1 \\ \int_{\frac{L}{4}}^{\frac{L}{2}} N_2cX_1dX_1 \end{pmatrix} \\ &=< u_1^*u_2^* > \begin{pmatrix} f_1^{e2} \\ f_2^{e2} \end{pmatrix} \end{aligned}$$

where f_1^{e2} and f_2^{e2} are the nodal forces to be applied at nodes 1 and 2 corresponding to lumping the distributed load acting on element 2.

4.4 Element3

On element 3, the displacement is given by: $u = u_2N_2 + u_3N_3$ and the virtual displacement is given by $u^* = u_2^*N_2 + u_3^*N_3$. By considering the third element, the third term in the internal virtual work equation has the following form:

$$\begin{aligned} IVW_3 =& < u_2^*u_3^* > \begin{pmatrix} \int_{\frac{L}{2}}^{\frac{3L}{4}} EA\frac{dN_2}{dX_1}\frac{dN_2}{dX_1}dX_1 & \int_{\frac{L}{2}}^{\frac{3L}{4}} EA\frac{dN_2}{dX_1}\frac{dN_3}{dX_1}dX_1 \\ \int_{\frac{L}{2}}^{\frac{3L}{4}} EA\frac{dN_3}{dX_1}\frac{dN_2}{dX_1}dX_1 & \int_{\frac{L}{2}}^{\frac{3L}{4}} EA\frac{dN_3}{dX_1}\frac{dN_3}{dX_1}dX_1 \end{pmatrix} \begin{pmatrix} u_2 \\ u_3 \end{pmatrix} \\ =& < u_2^*u_3^* > \begin{pmatrix} K_{11}^{e3} & K_{12}^{e3} \\ K_{21}^{e3} & K_{22}^{e3} \end{pmatrix} \begin{pmatrix} u_2 \\ u_3 \end{pmatrix} \end{aligned}$$

where K_{ij}^{e3} are the local stiffness matrix entries for element 3 corresponding to the degrees of freedom u_2 and u_3. The third term in the external virtual work has the form:

$$\begin{aligned} EVW_3 =& < u_2^*u_3^* > \begin{pmatrix} \int_{\frac{L}{2}}^{\frac{3L}{4}} N_2cX_1dX_1 \\ \int_{\frac{L}{2}}^{\frac{3L}{4}} N_3cX_1dX_1 \end{pmatrix} \\ =& < u_2^*u_3^* > \begin{pmatrix} f_1^{e3} \\ f_2^{e3} \end{pmatrix} \end{aligned}$$

where f_1^{e3} and f_2^{e3} are the nodal forces to be applied at nodes 2 and 3 corresponding to lumping the distributed load acting on element 3.

4.5 Element4

On element 4, the displacement is given by: $u = u_3N_3 + u_4N_4$ and the virtual displacement is given by $u^* = u_3^*N_3 + u_4^*N_4$. By considering the fourth element, the fourth term in the internal virtual work equation has the following form:

$$=< u_3^*u_4^* > \begin{pmatrix} K_{11}^{e4} & K_{12}^{e4} \\ K_{21}^{e4} & K_{22}^{e4} \end{pmatrix} \begin{pmatrix} u_3 \\ u_4 \end{pmatrix}$$

where K_{ij}^{e4} are the local stiffness matrix entries for element 4 corresponding to the degrees of freedom u_3 and u_4. The fourth term in the external virtual work has the form:

$$\begin{aligned} EVW_4 =& < u_3^*u_4^* > \begin{pmatrix} \int_{\frac{3L}{4}}^{L} N_3cX_1dX_1 \\ \int_{\frac{3L}{4}}^{L} N_4cX_1dX_1 \end{pmatrix} \\ =& < u_3^*u_4^* > \begin{pmatrix} f_1^{e4} \\ f_2^{e4} \end{pmatrix} \end{aligned}$$

where f_1^{e4} and f_2^{e4} are the nodal forces to be applied at nodes 3 and 4 corresponding to lumping the distributed load acting on element 4.

5 Global Stiffness Matrix

The global stiffness matrix can be assembled by the equation:

$$IVW_1 + IVW_2 + IVW_3 + IVW_4 = EVW_1 + EVW_2 + EVW_3 + EVW_4$$

The resulting equations (global stiffness matrix and global nodal forces vector) considering the arbitrariness of the multipliers u_i^* :

$$\begin{pmatrix} K_{11}^{e1} + K_{11}^{e2} & K_{12}^{e2} & 0 & 0 \\ K_{21}^{e2} & K_{22}^{e2} + K_{11}^{e3} & K_{12}^{e3} & 0 \\ 0 & K_{21}^{e3} & K_{22}^{e3} + K_{11}^{e4} & K_{12}^{e4} \\ 0 & 0 & K_{21}^{e4} & K_{22}^{e4} \end{pmatrix} \begin{pmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \end{pmatrix} = \begin{pmatrix} f_1^{e1} + f_1^{e2} \\ f_2^{e2} + f_1^{e3} \\ f_2^{e3} + f_1^{e4} \end{pmatrix}$$

It is important to note that because the assumed displacement function satisfies the essential boundary conditions, the structure is stable, and the equations can be solved directly. In general, the structure under consideration would have five degrees of freedom and would have a 5×5 global stiffness matrix and a 5×1 global nodal forces vector. The nodal forces vector corresponding to the displacement of the first node where the displacement is prescribed would have an unknown reaction. The stiffness matrix can then be reduced into a 4×4 matrix which can be solved.

The above four equations can be written in matrix form as follows:

$$\begin{pmatrix} \int_0^{\frac{L}{2}} EA\frac{dN_1}{dX_1}\frac{dN_1}{dX_1}dX_1 & \int_{\frac{L}{4}}^{\frac{L}{2}} EA\frac{dN_1}{dX_1}\frac{dN_2}{dX_1}dX_1 & 0 & 0 \\ \int_{\frac{L}{4}}^{\frac{3L}{4}} EA\frac{dN_2}{dX_1}\frac{dN_1}{dX_1}dX_1 & \int_{\frac{L}{4}}^{\frac{3L}{4}} EA\frac{dN_2}{dX_1}\frac{dN_2}{dX_1}dX_1 & \int_{\frac{L}{4}}^{\frac{3L}{4}} EA\frac{dN_2}{dX_1}\frac{dN_3}{dX_1}dX_1 & 0 \\ 0 & \int_{\frac{L}{2}}^{\frac{3L}{4}} EA\frac{dN_3}{dX_1}\frac{dN_2}{dX_1}dX_1 & \int_{\frac{L}{2}}^{L} EA\frac{dN_3}{dX_1}\frac{dN_3}{dX_1}dX_1 & \int_{\frac{3L}{4}}^{L} EA\frac{dN_3}{dX_1}\frac{dN_4}{dX_1}dX_1 \\ 0 & 0 & \int_{\frac{3L}{4}}^{L} EA\frac{dN_4}{dX_1}\frac{dN_3}{dX_1}dX_1 & \int_{\frac{3L}{4}}^{L} EA\frac{dN_4}{dX_1}\frac{dN_4}{dX_1}dX_1 \end{pmatrix} \begin{pmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \end{pmatrix}$$
$$= \begin{pmatrix} f_1 \\ f_2 \\ f_3 \\ f_4 \end{pmatrix}$$

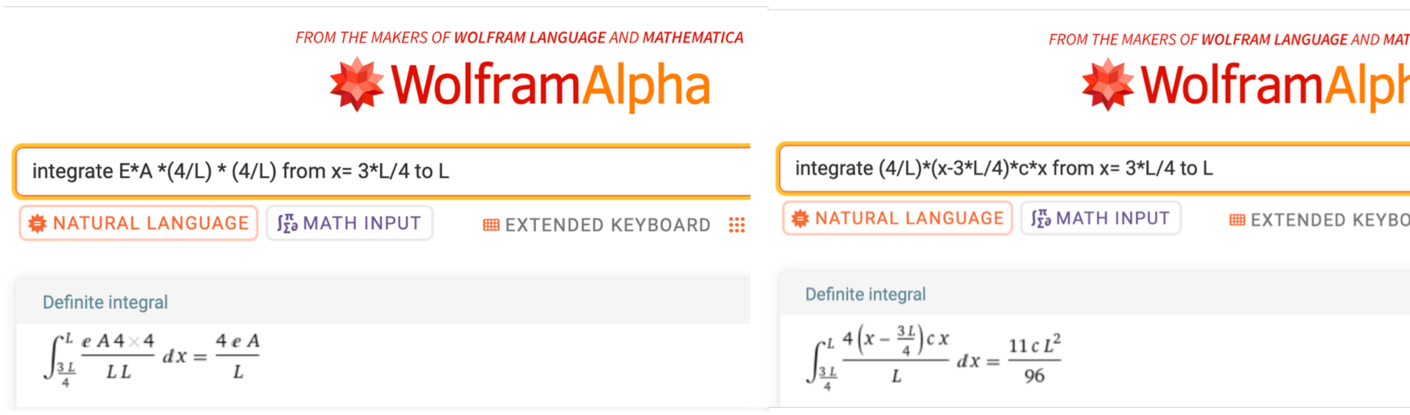
where $\forall i \leq 4 : f_i = \int_0^L N_i c X_1 \; \mathrm{d}X_1$. The integration on both sides is relatively simple to calculate. The final equations have the following form:

$$\begin{pmatrix} \frac{8}{L} & \frac{-4}{L} & 0 & 0 \\ \frac{-4}{L} & \frac{8}{L} & \frac{-4}{L} & 0 \\ 0 & \frac{-4}{L} & \frac{8}{L} & \frac{-4}{L} \\ 0 & 0 & \frac{-4}{L} & \frac{4}{L} \end{pmatrix} \begin{pmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \end{pmatrix} = \frac{cL^2}{EA} \begin{pmatrix} \frac{1}{16} \\ \frac{1}{8} \\ \frac{3}{16} \\ \frac{11}{96} \end{pmatrix}$$

which when solved gives the solution:

$$\begin{pmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \end{pmatrix} = \frac{cL^3}{EA} \begin{pmatrix} \frac{47}{384} \\ \frac{11}{48} \\ \frac{39}{128} \\ \frac{1}{3} \end{pmatrix}$$

To check this easy integration let's compute K_{22}^{e4} and f_2^{e4}:
$K_{22}^{e4} = \int_{\frac{3L}{4}}^{L} EA\frac{dN_4}{dX_1}\frac{dN_4}{dX_1}dX_1$ with $\frac{dN_4}{dX_1} = \frac{4}{L}$
$f_2^{e4} = f_i = \int_0^L N_4 c X_1 \; \mathrm{d}X_1 = f_i = \int_{3L/4}^{L} N_4 c X_1 \; \mathrm{d}X_1$ with $N_4 = \frac{4}{L}\left(X_1 - \frac{3L}{4}\right)$
NB: you could also integrate with sympy!


FROM THE MAKERS OF WOLFRAM LANGUAGE AND MATHEMATICA

WolframAlpha

integrate E*A *(4/L) * (4/L) from x= 3*L/4 to L

NATURAL LANGUAGE | MATH INPUT | EXTENDED KEYBOARD

Definite integral

$$\int_{\frac{3L}{4}}^{L} \frac{e\,A\,4\times 4}{L\,L}\,dx = \frac{4\,e\,A}{L}$$

FROM THE MAKERS OF WOLFRAM LANGUAGE AND MAT

WolframAlph

integrate (4/L)*(x-3*L/4)*c*x from x= 3*L/4 to L

NATURAL LANGUAGE | MATH INPUT | EXTENDED KEYBO

Definite integral

$$\int_{\frac{3L}{4}}^{L} \frac{4\left(x - \frac{3L}{4}\right)c\,x}{L}\,dx = \frac{11\,c\,L^2}{96}$$

Figure 3: Solution of the two easy integrals

```python
from sympy import symbols
from sympy.matrices import Matrix
L, c, E, A = symbols('L, c, E, A')
K = Matrix([[8/L, -4/L, 0, 0],
[-4/L, 8/L, -4/L, 0],
[0, -4/L, 8/L,-4/L],
[0, 0, -4/L, 4/L]])
b = (c*L**2)/(E*A) * Matrix([1/16., 1/8., 3/16., 11/96.])
print (K.solve(b))
```

Listing 2: Python example

6 It's up to you to code this in python

```python
import sympy as sp
import numpy as np
from sympy import lambdify
from matplotlib import pyplot as plt
x = sp.symbols("x")
xL = np.arange(0,1,0.01)
c,L,EA = 1,1,1
# Exact Calculations
u_exact = c*L**2/2/EA*x - c/sp.Rational("6")/EA*x**3
s_exact = u_exact.diff(x)
display("u_exact(x) =",u_exact,"s_exact(x) =",s_exact)
F = lambdify(x,u_exact)
dF = lambdify(x,s_exact)
y_exact = F(xL)
sy_exact = dF(xL)
def PN1(x):
    conds = [(x>=0)&(x<=L/4),(x>=L/4)&(x<L/2)]
    functions = [lambda x:4*x/L,lambda x:4/L*(L/2-x)]
    Dfunctions = [lambda x:4/L,lambda x:-4]
    return np.piecewise(x,conds,functions), np.piecewise(x,conds,Dfunctions)
def PN2(x):
    conds = [(x>=L/4)&(x<=L/2),(x>=L/2)&(x<3*L/4)]
    functions = [lambda x:4/L*(-L/4+x),lambda x:4/L*(3*L/4-x)]
    Dfunctions = [lambda x:4/L,lambda x:-4]
    return np.piecewise(x,conds,functions), np.piecewise(x,conds,Dfunctions)
def PN3(x):
    conds = [(x>=L/2)&(x<=3*L/4),(x>=3*L/4)&(x<=L)]
    functions = [lambda x:4/L*(-L/2+x),lambda x:4/L*(L-x)]
    Dfunctions = [lambda x:4/L,lambda x:-4]
    return np.piecewise(x,conds,functions), np.piecewise(x,conds,Dfunctions)
def PN4(x):
    conds = [(x>=3*L/4)&(x<=L)]
    functions = [lambda x:4/L*(-3*L/4+x)]
    Dfunctions = [lambda x:4/L]
    return np.piecewise(x,conds,functions), np.piecewise(x,conds,Dfunctions)
N1, DN1 = PN1(xL)
N2, DN2 = PN2(xL)
N3, DN3 = PN3(xL)
N4, DN4 = PN4(xL)
u1 = c*L**3/EA*(47/384)
u2 = c*L**3/EA*(11/48)
u3 = c*L**3/EA*(39/128)
u4 = c*L**3/EA*(1/3)
u = u1*N1+u2*N2+u3*N3+u4*N4
up = u1*DN1+u2*DN2+u3*DN3+u4*DN4
fig, ax = plt.subplots(2, figsize=(6,8))
ax[0].set_title("")
ax[0].set_xlabel("x")
ax[0].set_ylabel("displacement")
ax[0].plot(xL,y_exact,label="(c*L^2*x)/(2*EA) - (c*x^3)/(6*EA)")
ax[0].plot(xL,u,label="u_FEA")
ax[1].plot(xL,sy_exact,label="sigma_exact")
ax[1].plot(xL,up,label="sigma_FEA")
ax[1].set_ylabel("sigma11")
for i in ax:
    i.grid(True, which='both')
    i.axhline(y = 0, color = 'k')
    i.axvline(x = 0, color = 'k')
    i.set_xlabel("x")
    i.legend()
```

Listing 3: Python example

The finite element analysis approximation gave exact solution for the displacements of the points 1, 2, 3, and 4.

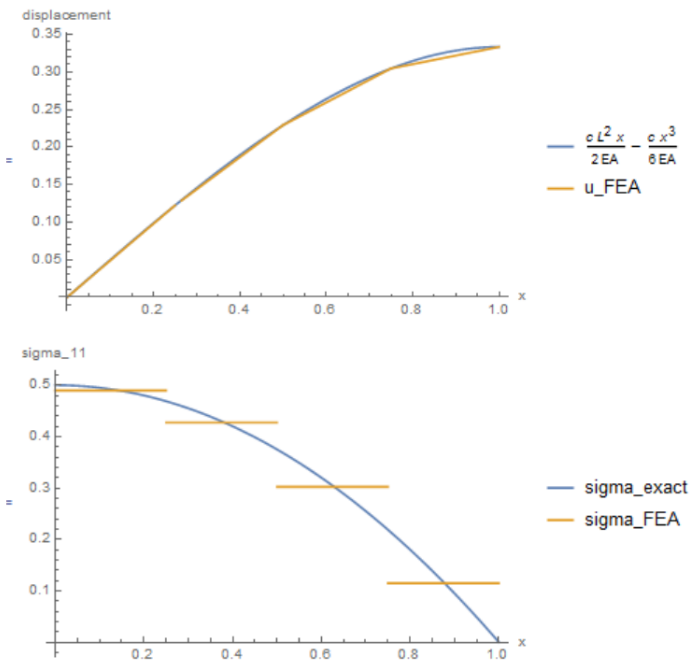


Figure 4: FEA in orange versus exact in blue

However, in between those points, the finite element analysis approximation following linear interpolation. As shown in the figure below when setting $c = 1$, $E = 1, A = 1$, and $L = 1$ units, the displacement values are almost accurate. However, the linear interpolation of the displacements means that the strain is constant in the area between points. Therefore, the stresses are constant between points and in fact, the stress distribution is discontinuous!

6.1 Properties

In general, the global stiffness matrix K has the following properties:

- Sparsity: The stiffness matrix formed in the traditional finite element analysis is sparse, i.e., contains many zero entries. This property is due to the choice of piecewise-linear interpolation functions. The shape functions (or the interpolation functions) are chosen to be "Hat functions" that take values of 1 at each node and decrease linearly or in a parabolic fashion across the neighboring elements connected to that node. Nodes that are not connected together with any elements have a corresponding zero entry in the global stiffness matrix.

- Invertibility: (See the section on Invertible linear maps) The stiffness matrix of a structure is a linear map between $u \in \mathbf{R}^n$ and $F \in \mathbf{R}^n$. Such map is invertible if and only if the map is injective (one to one), which means that the only element in the kernel of the map is the zero vector. Otherwise, if the map is not injective, then there are two nonzero distinct displacement vectors v and $w \in \mathbf{R}^n$ that can be produced by the same force vector $F \in \mathbf{R}^n$. l.e., $K(v-w) = F - F = 0$, then the nonzero vector displacement $v - w$ can be applied to the structure without any external forces! If no boundary conditions are applied to a structure, then naturally rigid body motions and rigid body rotations can be applied without any external forces. Therefore, a stiffness matrix of a structure that is not restrained is not invertible and its determinant is equal to zero. Once enough boundary conditions are applied to the structure, then the only element of the kernel of the linear map is the zero element, the stiffness matrix is invertible, and its determinant is not equal to zero. Notice, however, that when a structure is not well supported and does not have the correct boundary conditions, the stiffness matrix is termed "ill conditioned". Some finite element analysis software will still be able to find a possible solution, depending on the solution method. A user should be cautious in such cases since the solution could involve very large numbers for the displacement that would render the solution inaccurate.

- Symmetry: The stiffness matrix of an elastic structure is symmetric. The symmetry is a direct consequence of elasticity. Elasticity implies that there is an energy function such that $F_i = \frac{\partial W}{\partial u_i}$. Therefore, $K_{ij} = \frac{\partial F_i}{\partial u_j} = \frac{\partial^2 W}{\partial u_i \partial u_j}$ implying that K is symmetric.

- Positive and Semi-Positive Definiteness: (See the corresponding section in linear vector maps) A stable structure will deform in the direction of increasing external forces. Mathematically speaking, the dot product between the displacement and the force vectors has to be positive. If the structure is well restrained, i.e., K is invertible, then K is positive definite and any displacement field in the structure will produce positive energy. In mathematical terms, this imples: $\forall u \in \mathbf{R}^n : F.u = Ku.u > 0$. However, if the structure is not well restrained, then K is not invertible and $\exists u \in \mathbf{R}^n$ such that $Ku = O$. If the material of the structure is elastic, then the global stiffness matrix of the structure is semipositive definite. Most finite element analysis software will check the invertibility of the stiffness matrix by finding the smallest eigenvalue of the stiffness matrix. If one of the eigenvalues of the stiffness matrix is zero, then the stiffness matrix is not invertible. If one of the eigenvalues is negative, then the stiffness matrix is not positive definite, since positive definiteness ensures positive eigenvalues (See the corresponding section in linear vector maps). A direct consequence of the positive definiteness is that the diagonal entries of the stiffness matrix have to be positive; since a diagonal element represents the force applied at the corresponding degree of freedom to produce unit displacement at this particular degree of freedom, then the requirement that the energy is positive ensures that the force and the displacement are in the same direction.

- Any vertical column of the stiffness matrix is equal to the vector of nodal forces required to move the corresponding degree of freedom by one unit displacement while keeping the remaining degrees of freedom equal to zero.

7 References

- Adeeb, S. (2011). Introduction to solid mechanics and finite element analysis using Mathematica. Kendall Hunt.

1MAE701 - CSM: FINITE ELEMENT ANALYSIS: FEA IN TWO DIMENSIONS

Prof. Joseph Morlier

October 9, 2024

LEARNING OUTCOMES

Describe the steps required to find an approximate solution for a 2D elasticity probleme (plane stress or strain hypothesis)

- Write K and fe, solve u
- Compare CST versus QUAD
- Examples in plane stress

1 STIFFNESS MATRIX AND NODAL FORCES VECTOR FOR A GENERAL 2D LINEAR ELASTIC ELEMENT

The equations in the previous section are repeated after reducing them to two dimensions. In this case, the displacement vector u_{2D} of an element has two components designated u and v such that:

$$u_{2D} = \begin{pmatrix} u \\ v \end{pmatrix}$$

Assuming that the element has n nodes, then, each node i has 2 nodal degrees of freedom designated u_i and v_i. So, the nodal degrees of freedom vector u_e can have the form:

$$u_e = \begin{pmatrix} u_1 \\ v_1 \\ u_2 \\ v_2 \\ \vdots \\ u_n \\ v_n \end{pmatrix}$$

If $\forall i \leq n : N_i$ is the shape function associated with node i, then the displacement vector function has the form:

$$u_{2D} = \begin{pmatrix} u \\ v \end{pmatrix} = \begin{pmatrix} u_1 N_1 + u_2 N_2 + \cdots + u_n N_n \\ v_1 N_1 + v_2 N_n + \cdots + v_n N_n \end{pmatrix}$$

$$= \begin{pmatrix} N_1 & 0 & N_2 & 0 & \cdots & N_n & 0 \\ 0 & N_1 & 0 & N_2 & \cdots & 0 & N_n \end{pmatrix} \begin{pmatrix} u_1 \\ v_1 \\ u_2 \\ v_2 \\ \vdots \\ u_n \\ v_n \end{pmatrix} = N u_e$$

For simplicity, the vector representation of the small strain matrix will be used. In this case, the small strain matrix can be written in vector form as a function of the nodal degrees of freedom vector as follows:

$$\begin{aligned}
\varepsilon = \begin{pmatrix} \varepsilon_{11} \\ \varepsilon_{22} \\ 2\varepsilon_{12} \end{pmatrix} &= \begin{pmatrix} \frac{\partial u}{\partial X_1} \\ \frac{\partial v}{\partial X_2} \\ \frac{\partial u}{\partial X_2} + \frac{\partial v}{\partial X_1} \end{pmatrix} \\
&= \begin{pmatrix} \frac{\partial N_1}{\partial X_1} & 0 & \frac{\partial N_2}{\partial X_1} & 0 & \cdots & \frac{\partial N_n}{\partial X_1} & 0 \\ 0 & \frac{\partial N_1}{\partial X_2} & 0 & \frac{\partial N_2}{\partial X_2} & \cdots & 0 & \frac{\partial N_n}{\partial X_2} \\ \frac{\partial N_1}{\partial X_2} & \frac{\partial N_1}{\partial X_1} & \frac{\partial N_2}{\partial X_2} & \frac{\partial N_2}{\partial X_1} & \cdots & \frac{\partial N_n}{\partial X_2} & \frac{\partial N_n}{\partial X_1} \end{pmatrix} \begin{pmatrix} u_1 \\ v_1 \\ u_2 \\ v_2 \\ \vdots \\ u_n \\ v_n \end{pmatrix} \\
&= B u_e
\end{aligned}$$

For a general plane strain or plain stress linear elastic material, the vector representation of the stress matrix is related to the vector representation of the small strain matrix by a 3×3 symmetric material coefficients matrix C :

$$\sigma = \begin{pmatrix} \sigma_{11} \\ \sigma_{22} \\ \sigma_{12} \end{pmatrix} = C \begin{pmatrix} \varepsilon_{11} \\ \varepsilon_{22} \\ 2\varepsilon_{12} \end{pmatrix}$$

1.1 VW?

Utilizing the principle of virtual work, the virtual displacement field u^*_{2D} and the associated virtual strain field ε^* can have the following forms with u^*_e being an arbitrary "virtual" vector of degrees of freedom of the system:

$$u^*_{2D} = \begin{pmatrix} u^* \\ v^* \end{pmatrix} = u^*_e N^T$$
$$\varepsilon^{*T} = u^*_e B^T$$

where

$$u^*_e = \langle u^*_1 v^*_1 u^*_2 v^*_2 \cdots u^*_n v^*_n >$$
$$\varepsilon^{*T} =< \varepsilon^*_{11} \varepsilon^*_{22} 2\varepsilon^*_{12} >$$

The internal virtual work integral term for this particular element will have the form:

$$\begin{aligned}
IVW_e &= \int_e \sum_{i,j=1}^{3} \varepsilon^*_{ij} \sigma_{ij} dx = \int_e \varepsilon^* \cdot \sigma dx \\
&= u^*_e \left(\int_e B^T C B dx \right) u_e
\end{aligned}$$

where the integration is done on each term in the matrix $B^T C B$ and is performed over the volume (which in the 2D case would be an integration over the area multiplied by the thickness) of the element. The local stiffness matrix has dimensions $2n \times 2n$ and has the form:

$$K^e = \int_e B^T C B dx$$

The external virtual work integral term for this particular element will have the form:

$$EVW_e = u^*_e \int_{\partial e} N^T t_n ds + u^*_e \int_e N^T \rho b dx$$

Where $t_n \in \mathbf{R}^2$ is the traction vectors on the surface of the element, ρ is the mass density, and $b \in \mathbf{R}^2$ is the body forces vector on the element. The integration in the first term is done over the element surface while in the second term is done over the element volume. The local nodal forces vector will $2n$ components and will have the form:

$$f^e = \int_{\partial e} N^T t_n ds + \int_e N^T \rho b dx$$

2 CST?

The linear triangular element is obtained by assuming that the displacement within a triangular shape is a linear function of the displacements at the three corner nodes. This linear triangular element is also called the constant strain triangle, since as will be shown in the derivation below, the strain across the whole element is always constant (the matrix B has constant values). This is a major disadvantage of such element since it tends to be extremely stiff. Approximating the domain with triangular elements can only be considered within good accuracy if the difference in the strains between neighboring elements is relatively small.

Consider the plane triangle shown in Figure 1. For simplicity, two of the triangle sides are assumed to have unit lengths. The assumed (approximate) linear-displacement function across the element has the form:

$$u_{2D} = \begin{pmatrix} u \\ v \end{pmatrix} = \begin{pmatrix} a_0 + a_1 X_1 + a_2 X_2 \\ b_0 + b_1 X_1 + b_2 X_2 \end{pmatrix}$$

where a_0, a_1, a_2, b_0, b_1, and b_2 are six generalized degrees of freedom of the element. The approximate displacement function in terms of the nodal degrees of freedom has the form:

$$u_{2D} = \begin{pmatrix} u \\ v \end{pmatrix} = \begin{pmatrix} N_1 u_1 + N_2 u_2 + N_3 u_3 \\ N_1 v_1 + N_2 v_2 + N_3 v_3 \end{pmatrix}$$

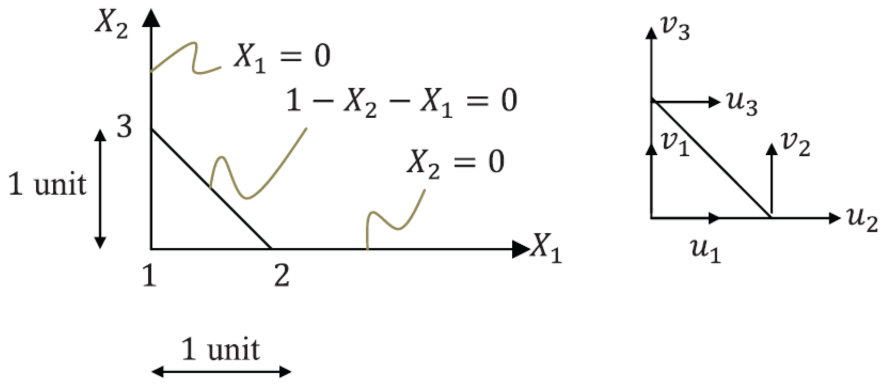


Figure 1: Linear Triangular Element (Constant Strain Triangle)

2.1 K matrix?

There are two ways to find the shape functions N_i. The first way is to replace the generalized degrees of freedom with the nodal degrees of freedom in the first equation. The multipliers of the nodal degrees of freedom would then be the shape function. The following mathematica code does that for you:

An alternate way is directly by realizing that N_1 has to satisfy that the condition that at node 1, $N_1 = 1$, it varies linearly and is equal to zero on the line $1 - X_1 - X_2 = 0$. Similarly, N_2 is equal to 1 at node 2 and is equal to zero on the line $X_1 = 0$. Also, N_3 is equal to 1 at node 3 and is equal to zero on the line $X_2 = 0$. These result in the following expressions for the shape functions:

$$N_1 = 1 - X_1 - X_2$$
$$N_2 = X_1$$
$$N_3 = X_2$$

The distribution of the shape functions on the element are illustrated in Figure.

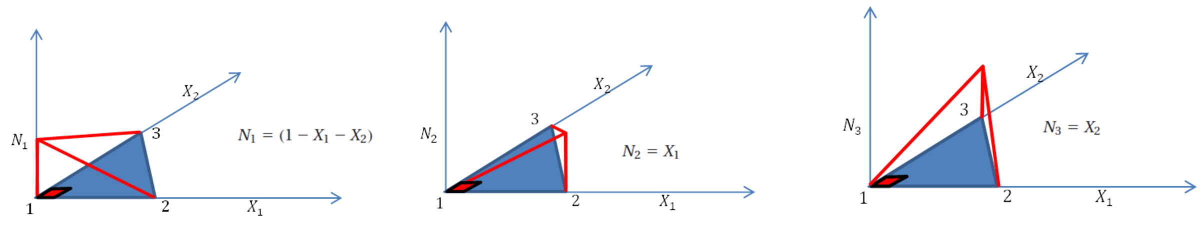


Figure 2: Shape functions in linear triangular elements

The strain associated with the assumed displacement field has the following form:

$$\begin{aligned}
\varepsilon = \begin{pmatrix} \varepsilon_{11} \\ \varepsilon_{22} \\ 2\varepsilon_{12} \end{pmatrix} &= \begin{pmatrix} \frac{\partial u}{\partial X_1} \\ \frac{\partial v}{\partial X_2} \\ \frac{\partial u}{\partial X_2} + \frac{\partial v}{\partial X_1} \end{pmatrix} \\
&= \begin{pmatrix} \frac{\partial N_1}{\partial X_1} & 0 & \frac{\partial N_2}{\partial X_1} & 0 & \frac{\partial N_3}{\partial X_1} & 0 \\ 0 & \frac{\partial N_1}{\partial X_2} & 0 & \frac{\partial N_2}{\partial X_2} & 0 & \frac{\partial N_3}{\partial X_2} \\ \frac{\partial N_1}{\partial X_2} & \frac{\partial N_1}{\partial X_1} & \frac{\partial N_2}{\partial X_2} & \frac{\partial N_2}{\partial X_1} & \frac{\partial N_3}{\partial X_2} & \frac{\partial N_3}{\partial X_1} \end{pmatrix} \begin{pmatrix} u_1 \\ v_1 \\ u_2 \\ v_2 \\ u_3 \\ v_3 \end{pmatrix} \\
&= \begin{pmatrix} -1 & 0 & 1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 & 0 & 1 \\ -1 & -1 & 0 & 1 & 1 & 0 \end{pmatrix} \begin{pmatrix} u_1 \\ v_1 \\ u_2 \\ v_2 \\ u_3 \\ v_3 \end{pmatrix} \\
&= Bu_e
\end{aligned}$$

Notice that as mentioned above, the values of the strain components are independent of the location inside the triangle; rather the strain components are constant across the element.

```python
import sympy as sp
from sympy import solve
a0,a1,a2,x1,x2 = sp.symbols("a_0 a_1 a_2 x_1 x_2")
b0,b1,b2= sp.symbols("b_0 b_1 b_2")
u1,u2,u3,v1,v2,v3 = sp.symbols("u_1 u_2 u_3 v_1 v_2 v_3")
u = a0+a1*x1+a2*x2
v = b0+b1*x1+b2*x2
display("u =",u,"v =",v)
Coordinates = [{x1:0,x2:0},{x1:1,x2:0},{x1:0,x2:1}]
display("Coordinates: ",Coordinates)
Eq1 = u.subs(Coordinates[0])
Eq2 = u.subs(Coordinates[1])
Eq3 = u.subs(Coordinates[2])
Eq4 = v.subs(Coordinates[0])
Eq5 = v.subs(Coordinates[1])
Eq6 = v.subs(Coordinates[2])
s = solve((Eq1-u1,Eq2-u2,Eq3-u3,Eq4-v1,Eq5-v2,Eq6-v3),
          (a0,a1,a2,b0,b1,b2))
display("Solve: ",s)
u = u.subs(s).expand()
v = v.subs(s).expand()
display("u =",u,"v =",v)
N1_u = u.coeff(u1)
N1_v = v.coeff(v1)
N2_u = u.coeff(u2)
N2_v = v.coeff(v2)
N3_u = u.coeff(u3)
```

```python
N3_v = v.coeff(v3)
display("N1u1 =",N1_u,"N1v1 =",N1_v,"N2u2 =",N2_u,
        "N2v2 =",N2_v,"N3u3 =",N3_u,"N3v3 =",N3_v)
```

Listing 1: Python example

The constitutive relationship of plane linear elastic materials is defined using a 3×3 matrix C that depends on whether the material is in a plane strain or a plane stress state. For the sake of illustration, the plane strain state is chosen here, the material constitutive relationship matrix in that case is:

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

The stiffness matrix of the constant strain triangle can be evaluated by the following integral:

$$K^e = \int_e B^T C B dX = \int_e \begin{pmatrix} -1 & 0 & -1 \\ 0 & -1 & -1 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix} C \begin{pmatrix} -1 & 0 & 1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 & 0 & 1 \\ -1 & -1 & 0 & 1 & 1 & 0 \end{pmatrix} dX$$

Assuming that the triangle has a constant thickness t, then the differential volume $\mathrm{d}X = t\,\mathrm{d}X_1\,\mathrm{d}X_2$. In addition, all the components of the matrix B^TCB are constant; therefore, the integral can be evaluated by simply multiplying the matrix B^TCB by $\frac{t}{2}$, which represents the volume of the triangle:

$$K^e = \frac{tE}{2(1+\nu)} \begin{pmatrix} \frac{3-4\nu}{2-4\nu} & \frac{1}{2-4\nu} & \frac{1-\nu}{-1+2\nu} & \frac{-1}{2} & \frac{-1}{2} & \frac{\nu}{-1+2\nu} \\ \frac{1}{2-4\nu} & \frac{3-4\nu}{2-4\nu} & \frac{\nu}{-1+2\nu} & \frac{-1}{2} & \frac{-1}{2} & \frac{1-\nu}{-1+2\nu} \\ \frac{1-\nu}{-1+2\nu} & \frac{\nu}{-1+2\nu} & \frac{1-\nu}{-1+2\nu} & 0 & 0 & \frac{\nu}{-1+2\nu} \\ \frac{-1}{2} & \frac{-1}{2} & 0 & \frac{1}{2} & \frac{1}{2} & 0 \\ \frac{-1}{2} & \frac{-1}{2} & 0 & \frac{1}{2} & \frac{1}{2} & 0 \\ \frac{\nu}{-1+2\nu} & \frac{1-\nu}{-1+2\nu} & \frac{\nu}{-1+2\nu} & 0 & 0 & \frac{1-\nu}{1-2\nu} \end{pmatrix}$$

```python
import sympy as sp
from sympy import Matrix,simplify
Ee,nu,t = sp.symbols("E nu t")
B = Matrix([[-1,0,1,0,0,0],[0,-1,0,0,0,1],[-1,-1,0,1,1,0]])

Cc = Ee*t/2/(1 + nu)*Matrix([[(1-nu)/(1-2*nu),nu/(1-2*nu),0],
                             [nu/(1-2*nu),(1-nu)/(1-2*nu), 0],
                             [0, 0, 1/2]])
K = simplify(B.T*Cc*B)
display("K^e =",K)
```

Listing 2: Python example

2.2 NODAL FORCES DUE TO BODY FORCES

Assuming that the distributed body forces vector per unit mass is constant and is given by $b = b_1, b_2$, and that the mass density is ρ, then the nodal forces due to the distributed body forces can be obtained using the integral:

$$f^e = \int_e N^T \rho b dX = \rho t \int_0^1 \int_0^{1-X_2} \begin{pmatrix} N_1 & 0 \\ 0 & N_1 \\ N_2 & 0 \\ 0 & N_2 \\ N_3 & 0 \\ 0 & N_3 \end{pmatrix} \begin{pmatrix} b_1 \\ b_2 \end{pmatrix} dX_1 dX_2 = \frac{\rho t}{6} \begin{pmatrix} b_1 \\ b_2 \\ b_1 \\ b_2 \\ b_1 \\ b_2 \end{pmatrix}$$

i.e., the constant body forces vector can be represented by lumped equal concentrated loads at the nodes (Figure 3a).

```python
import sympy as sp
from sympy import Matrix,integrate
x1,x2,rb1,rb2 = sp.symbols("x_1 x_2 rho_1 rho_2")
Shapefun = Matrix([[1-x1-x2],[x1],[x2]])
Nn = Matrix([[0 for x in range(6)] for y in range(2)])
for i in range(3):
    Nn[0,2*i] = Nn[1,2*i+1] = Shapefun[i]
rb = Matrix([[rb1],[rb2]])
f = Matrix([[integrate(i,(x1,0,1-x2),(x2,0,1))]for i in Nn.T*rb])
display("Nodal Force due to Body Force:")
display("f^e =",f)
```

Listing 3: Python example

2.3 NODAL FORCES DUE TO TRACTION VECTOR ON ONE SIDE

Assuming that a constant distributed pressure per unit area of $t_n = t_1, t_2$ is acting on the surface joining nodes 1 and 3 , then the nodal forces due to the distributed traction vector can be obtained using the following integral evaluated on the left side $X_1 = 0$:

$$f^e = \int_{\partial e} N^T t_n dS = t\int_0^1 \begin{pmatrix} N_1 & 0 \\ 0 & N_1 \\ N_2 & 0 \\ 0 & N_2 \\ N_3 & 0 \\ 0 & N_3 \end{pmatrix} \begin{pmatrix} t_1 \\ t_2 \end{pmatrix} dX_2 = \frac{t}{2} \begin{pmatrix} t_1 \\ t_2 \\ 0 \\ 0 \\ t_1 \\ t_2 \end{pmatrix}$$

Thus, the distributed constant traction on one side of the triangle can be lumped into equal nodal loads applied on the nodes of that side (Figure 3b).

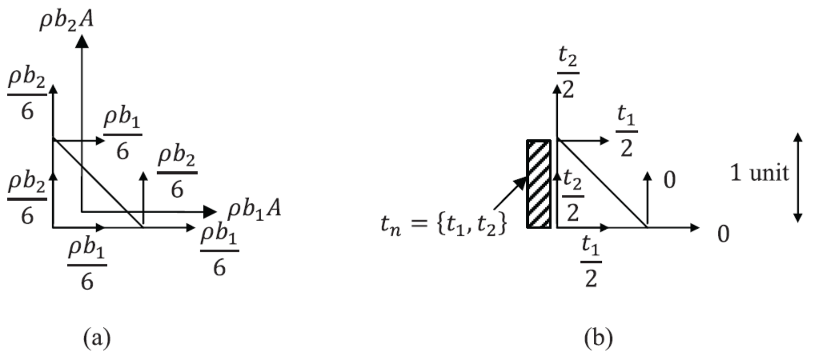


Figure 3: Nodal forces in a linear triangular element with a constant unit thickness due to (a) constant body forces vectors, (b) constant traction vector on one side.

```python
import sympy as sp
from sympy import Matrix,integrate
x1,x2,t1,t2 = sp.symbols("x_1 x_2 t_1 t_2")
Shapefun = Matrix([[1-x1-x2],[x1],[x2]])
Nn = Matrix([[0 for x in range(6)] for y in range(2)])
for i in range(3):
    Nn[0,2*i] = Nn[1,2*i+1] = Shapefun[i]
tn = Matrix([[t1],[t2]])
f = Matrix([[(integrate(i,(x2,0,1))).subs({x1:0})]for i in Nn.T*tn])
display("Nodal Force due to Traction Vector on One Side:")
display("f^e =",f)
```

Listing 4: Python example

3 QUAD?

It is usually easier to discretize a domain by using triangular elements; however, when the domain to be discretized has a regular shape, quadrilateral elements can be used to offer a more regular displacement discretization, which, in some cases, might offer a better approximation to the displacement shape. The bilinear quadrilateral element offers a bilinear displacement approximation where the displacement within the element is assumed to vary bilinearly (linear in two directions within the element). This element, however, behaves in a stiff manner when it is used to model linear strains (bending strains) since, as will be shown in the derivation, it is not possible to model pure bending (pure linear normal strain components in one direction) without introducing additional shear strains (usually termed parasitic shear strains).

Consider the plane quadrilateral shown in Figure 7. The bilinear displacement function for this element is assumed to have the following form:

$$u_{2D} = \begin{pmatrix} u \\ v \end{pmatrix} = \begin{pmatrix} a_0 + a_1X_1 + a_2X_2 + a_3X_1X_2 \\ b_0 + b_1X_1 + b_2X_2 + b_3X_1X_2 \end{pmatrix}$$

where $a_0, a_1, a_2, a_3, b_0, b_1, b_2,$ and b_3 are eight generalized degrees of freedom of the element. The displacement function in terms of the nodal degrees of freedom has the form:

$$u_{2D} = \begin{pmatrix} u \\ v \end{pmatrix} = \begin{pmatrix} N_1u_1 + N_2u_2 + N_3u_3 + N_4u_4 \\ N_1v_1 + N_2v_2 + N_3v_3 + N_4v_4 \end{pmatrix}$$

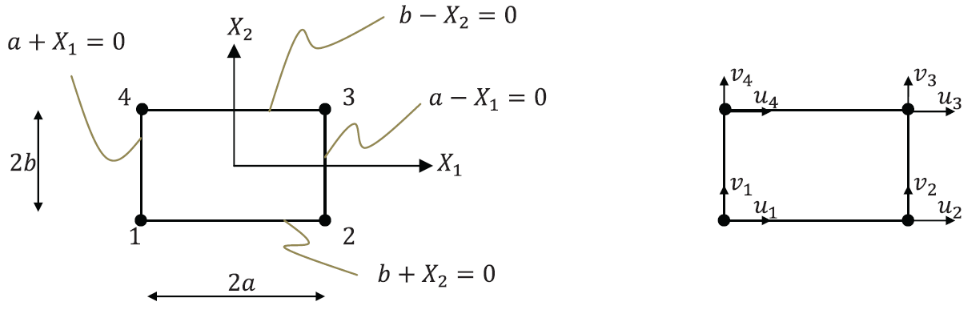


Figure 4: Bilinear quadrilateral

There are two ways to find the shape functions N_i. The first way is to replace the generalized degrees of freedom with the nodal degrees of freedom in the first equation. The multipliers of the nodal degrees of freedom would then be the shape function. The following mathematica code does that for you:

An alternate way is directly by realizing that N_1 has to satisfy that the condition that at node $1, N_1 = 1$, and is equal to zero on the lines $a - X_1 = 0$ and $b - X_2 = 0$. The same can be applied for N_2 and N_3. These result in the following expressions for the shape functions:

$$N_1 = \frac{(b - X_2)(a - X_1)}{4ab}$$
$$N_2 = \frac{(b - X_2)(a + X_1)}{4ab}$$
$$N_3 = \frac{(b + X_2)(a + X_1)}{4ab}$$
$$N_4 = \frac{(b + X_2)(a - X_1)}{4ab}$$

The distribution of the shape functions on the element are illustrated in Figure 8.

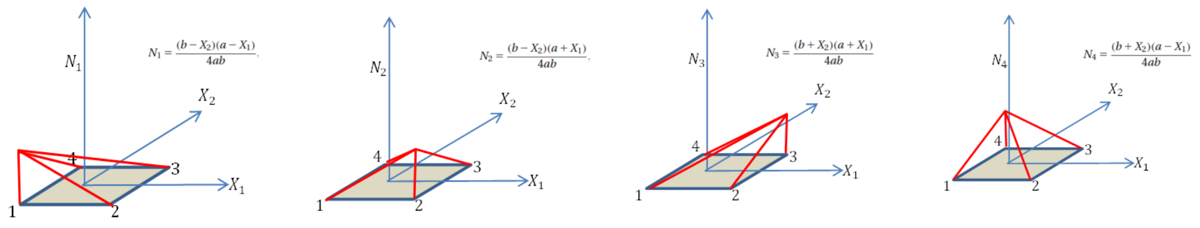

N_1
$N_1 = \frac{(b-X_2)(a-X_1)}{4ab}$
X_2
X_1
1
2
3
4
N_2
$N_2 = \frac{(b-X_2)(a+X_1)}{4ab}$
N_3
$N_3 = \frac{(b+X_2)(a+X_1)}{4ab}$
N_4
$N_4 = \frac{(b+X_2)(a-X_1)}{4ab}$

Figure 5: Shape functions distribution in the bilinear quadrilateral element

The strain associated with the assumed displacement field has the following form:

$$\varepsilon = \begin{pmatrix} \varepsilon_{11} \\ \varepsilon_{22} \\ 2\varepsilon_{12} \end{pmatrix} = \begin{pmatrix} \frac{\partial u}{\partial X_1} \\ \frac{\partial v}{\partial X_2} \\ \frac{\partial u}{\partial X_2} + \frac{\partial v}{\partial X_1} \end{pmatrix}$$

$$= \begin{pmatrix} \frac{\partial N_1}{\partial X_1} & 0 & \frac{\partial N_2}{\partial X_1} & 0 & \frac{\partial N_3}{\partial X_1} & 0 & \frac{\partial N_4}{\partial X_1} & 0 \\ 0 & \frac{\partial N_1}{\partial X_2} & 0 & \frac{\partial N_2}{\partial X_2} & 0 & \frac{\partial N_3}{\partial X_2} & 0 & \frac{\partial N_4}{\partial X_2} \\ \frac{\partial N_1}{\partial X_2} & \frac{\partial N_1}{\partial X_1} & \frac{\partial N_2}{\partial X_2} & \frac{\partial N_2}{\partial X_1} & \frac{\partial N_3}{\partial X_2} & \frac{\partial N_3}{\partial X_1} & \frac{\partial N_4}{\partial X_2} & \frac{\partial N_4}{\partial X_1} \end{pmatrix} \begin{pmatrix} u_1 \\ v_1 \\ u_2 \\ v_2 \\ u_3 \\ v_3 \\ u_4 \\ v_4 \end{pmatrix}$$

$$= Bu_e$$

Where, the explicit representation of the matrix B is:

$$B = \begin{pmatrix} -\frac{b-X_2}{4ab} & 0 & \frac{b-X_2}{4ab} & 0 & \frac{b+X_2}{4ab} & 0 & -\frac{b+X_2}{4ab} & 0 \\ 0 & -\frac{a-X_1}{4ab} & 0 & -\frac{a+X_1}{4ab} & 0 & \frac{a+X_1}{4ab} & 0 & \frac{a-X_1}{4ab} \\ -\frac{a-X_1}{4ab} & -\frac{b-X_2}{4ab} & -\frac{a+X_1}{4ab} & \frac{b+X_2}{4ab} & \frac{a+X_1}{4ab} & \frac{b+X_2}{4ab} & \frac{a-X_1}{4ab} & -\frac{b+X_2}{4ab} \end{pmatrix}$$

```
    import sympy as sp
from sympy import *
a,a0,a1,a2,a3,b,b0,b1,b2,b3,X1,X2,u1,u2,u3,u4,v1,v2,v3,v4 = symbols("a a_0 a_1 a_2 a_3
     b b_0 b_1 b_2 b_3 X_1 X_2 u_1 u_2 u_3 u_4 v_1 v_2 v_3 v_4")
u = a0+a1*X1+a2*X2+a3*X1*X2
v = b0+b1*X1+b2*X2+b3*X1*X2
Coordinates = [{X1:-a,X2:-b},{X1:a,X2:-b},{X1:a,X2:b}, {X1:-a, X2:b}]
display("Coordinates: ",Coordinates)
Eq1 = u.subs(Coordinates[0])
Eq2 = u.subs(Coordinates[1])
Eq3 = u.subs(Coordinates[2])
Eq4 = u.subs(Coordinates[3])
Eq5 = v.subs(Coordinates[0])
Eq6 = v.subs(Coordinates[1])
Eq7 = v.subs(Coordinates[2])
Eq8 = v.subs(Coordinates[3])
s = solve((Eq1-u1,Eq2-u2,Eq3-u3,Eq4-u4,Eq5-v1,Eq6-v2, Eq7-v3, Eq8-v4),
          (a0,a1,a2,a3,b0,b1,b2,b3))
display("Solve: ",s)
u = u.subs(s).expand()
v = v.subs(s).expand()
display("u =",u,"v =",v)
N1_u = u.coeff(u1)
N1_v = v.coeff(v1)
N2_u = u.coeff(u2)
N2_v = v.coeff(v2)
N3_u = u.coeff(u3)
N3_v = v.coeff(v3)
N4_u = u.coeff(u4)
```

```
N4_v = v.coeff(v4)
display("N1u1 =",N1_u,"N1v1 =",N1_v,"N2u2 =",N2_u,
        "N2v2 =",N2_v,"N3u3 =",N3_u,"N3v3 =",N3_v,
        "N4u4=",N4_u,"N4v4 =",N4_v)
```

Listing 5: Python example

A quick glance at the matrix B shows that the bilinear quadrilateral can model bending (linear strains) since the strains ε_{11} and ε_{22} contain linear expressions in X_1 and X_2. However, it will be shown here that to model linear strains, the element predicts an associated shear strain γ_{12} as well (termed parasitic shear strains). For simplicity, we will calculate the strains based on Equation 1. The strains have the form:

$$\begin{aligned}\varepsilon_{11} &= a_1 + a_3X_2\\ \varepsilon_{22} &= b_2 + b_3X_1\\ \gamma_{12} &= a_2 + a_3X_1 + b_1 + b_3X_2\end{aligned}$$

If we now assume pure bending state of strain with $a_3 \neq 0$. Then, the shear strain cannot be equal to zero since a_3 appears in the expression for γ_{12} ! This behaviour is termed shear locking and causes the element to be too stiff under pure bending.

The constitutive relationship of plane linear elastic materials is defined using a 3×3 matrix C that depends on whether the material is in a plane strain or a plane stress state. For the sake of illustration, the plane strain state is chosen here, the material constitutive relationship matrix in that case is:

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

3.1 K?

The stiffness matrix of the linear strain triangle can be evaluated using the following integral (assuming a constant thickness t :

$$K^e = t\int_{-b}^{b}\int_{-a}^{a} B^TCBdX_1dX_2$$

The stiffness matrix has the dimensions of 8×8, and the following Mathematica code can be utilized to view its components:

```
import sympy as sp
from sympy import *
a, b, x1, x2, nu, E, t = symbols("a b x_1 x_2 nu E t")
Shapefun = Matrix([(b-x2)*(a-x1)/4/a/b,(b-x2)*(a+x1)/4/a/b,(b+x2)*(a+x1)/4/a/b,(b+x2)
    *(a-x1)/4/a/b])
B = zeros(3,8)
for i in range(4):
    B[0,2*i] = B[2,2*i+1] = Shapefun[i].diff(x1)
    B[1,2*i+1] = B[2,2*i] = Shapefun[i].diff(x2)
display("B:", B)
Cc = E/(1 + nu)*Matrix([[(1-nu)/(1-2*nu),nu/(1-2*nu),0],
                         [nu/(1-2*nu),(1-nu)/(1-2*nu), 0],
                         [0, 0, 1/2]])
KBeforeintegration = t * B.transpose()*Cc*B
K = Matrix([[simplify(integrate(KBeforeintegration[i,j],(x1,-a,a),(x2,-b,b)))for i in
    range (8)] for j in range (8)])
display("K: ", K)
```

Listing 6: Python example

3.2 NODAL LOADS

The nodal loads due to distributed body forces and traction forces on the boundaries are equally distributed among the nodes. The same procedure followed for the triangular elements can be repeated to obtain the distribution shown in Figure 9. The following Mathematica code can be utilized for the calculations producing the distributions in Figure 9.

```python
import sympy as sp
from sympy import *
a, b, x1, x2, t1, t2, rb1, rb2 = symbols("a b x_1 x_2 t_1 t_2 rb_1 rb_2")
Shapefun = Matrix([(b-x2)*(a-x1)/4/a/b,(b-x2)*(a+x1)/4/a/b,(b+x2)*(a+x1)/4/a/b,(b+x2)
    *(a-x1)/4/a/b])
Nn= zeros(2,8)
for i in range(4):
    Nn[0,2*i] = Nn[1,2*i+1] = Shapefun[i]
display("Nn:", Nn)
tn = Matrix([t1,t2])
rb = Matrix([rb1,rb2])
integrand1 = (Nn.transpose()*tn).subs(x1,-a)
integrand2 = (Nn.transpose()*rb)
fetraction = Matrix([simplify(integrate(integrand1[i],(x2,-b,b)))for i in range (8)])
febodyforces = Matrix([simplify(integrate(integrand2[i],(x1,-a,a),(x2,-b,b)))for i in
    range (8)])
display("fetraction: ", fetraction)
display("febodyforces: ", febodyforces)
```

Listing 7: Python example

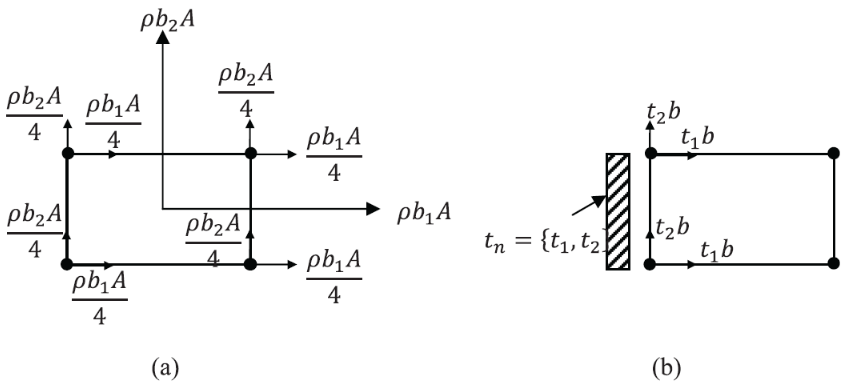


Figure 6: Nodal forces in a bilinear quadrilateral element with a constant unit thickness due to (a) constant body forces vectors, (b) constant traction vector on one side.

4 References

- Adeeb, S. (2011). Introduction to solid mechanics and finite element analysis using Mathematica. Kendall Hunt.

1MAE701 - CSM: ONE AND TWO DIMENSIONAL ISOPARAMETRIC ELEMENTS AND GAUSS INTEGRATION: ISOPARAMETRIC ELEMENTS

Prof. Joseph Morlier

October 9, 2024

LEARNING OUTCOMES

Describe the steps required to find an approximate solution for a 2D elasticity problems (plane stress or strain hypothesis)

- One and Two Dimensional Isoparametric Elements and Gauss Integration and Jacobian
- Compute K for a triangle element (CST) of thickness t
- Examples in plane stress

1 ONE DIMENSIONAL 3-NODE QUADRATIC ISOPARAMETRIC ELEMENT

the one dimensional quadratic element described previously was defined with the middle node equidistant from the end nodes. However, the isoparametric formulation allows defining a general one dimensional quadratic element with nodal coordinates x_1, x_2, and x_3 such that the only requirement is that $x_1 < x_2 < x_3$. Another coordinate system ξ can be defined for the element such that the coordinates of the nodes 1,2 , and 3 are $\xi_1 = -1, \xi_2 = 0$, and $\xi_3 = 1$, respectively. A mapping between the general coordinate system x and the coordinate system ξ can be defined (Figure 1) using the same shape functions that define the displacement in the coordinate system ξ as follows:

$$x = N_1 x_1 + N_2 x_2 + N_3 x_3$$

where the shape functions have the form:

$$N_1 = \frac{-\xi(1-\xi)}{2}$$
$$N_2 = (1-\xi)(1+\xi)$$
$$N_3 = \frac{\xi(1+\xi)}{2}$$

The gradient of this mapping has the following form:

$$\frac{dx}{d\xi} = \sum_{i=1}^{3} \frac{dN_1}{d\xi} x_i$$

Since $x_1 < x_2 < x_3$, and since $\xi \in [-1, 1]$, then

$$\frac{dx}{d\xi} > 0$$

Therefore, the inverse of the gradient is well defined as:

$$\frac{d\xi}{dx} = \frac{1}{\frac{dx}{d\xi}} = \frac{1}{\sum_{i=1}^{3} \frac{dN_i}{d\xi} x_i}$$

If u_1, u_2, and u_3 are the horizontal displacements of the nodes 1,2 , and 3 , respectively, then, the horizontal displacement function is defined in the ξ coordinate system as follows:

$$u = N_1 u_1 + N_2 u_2 + N_3 u_3$$

The normal strain ε_{11} is defined in the x coordinate system as follows:

$$\varepsilon_{11} = \frac{du}{dx} = \frac{du}{d\xi}\frac{d\xi}{dx} = \frac{d\xi}{dx} < \frac{dN_1}{d\xi} \frac{dN_2}{d\xi} \frac{dN_3}{d\xi} > \begin{pmatrix} u_1 \\ u_2 \\ u_3 \end{pmatrix}$$

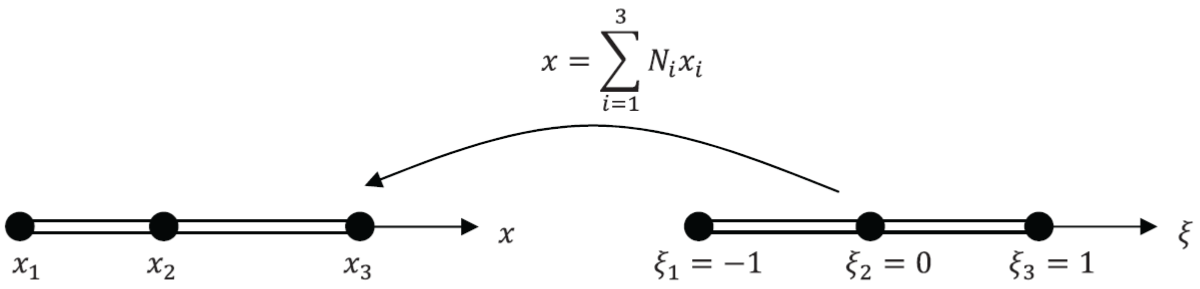


Figure 1: One Dimensional Quadratic Isoparametric Mapping

1.1 K matrix?

The virtual displacement u^* and the associated virtual strain ε_{11}^* can be chosen as follows:

$$u^* = N_1 u_1^* + N_2 u_2^* + N_3 u_3^*$$

$$\varepsilon_{11}^* = \frac{d\xi}{dx} < u_1^* u_2^* u_3^* > \begin{pmatrix} \frac{dN_1}{d\xi} \\ \frac{dN_2}{d\xi} \\ \frac{dN_3}{d\xi} \end{pmatrix}$$

Assuming p as a distributed load per unit length of the one dimensional element, then, the contribution to the virtual work equation of this element are as follows:

$$IVW_e = A \int_{x_1}^{x_3} \sigma_{11} \varepsilon_{11}^* dx$$

$$EVW_e = \int_{x_1}^{x_3} p u^* dx$$

To utilize the numerical Gauss integration scheme (as described later) Integration is performed in the ξ coordinate system. In general, for a function $g(x)$ we have:

$$\int_{x_1}^{x_3} g(x) dx = \int_{-1}^{1} g(x(\xi)) \frac{dx}{d\xi} d\xi$$

Using the constitutive law $\sigma_{11} = E\varepsilon_{11}$, the virtual work contributions of the element have the forms:

$$IVW_e =< u_1^* u_2^* u_3^* > EA \int_{-1}^{1} \left(\frac{d\xi}{dx}\right)^2 \frac{dx}{d\xi} \begin{pmatrix} \frac{dN_1}{d\xi} \\ \frac{dN_2}{d\xi} \\ \frac{dN_3}{d\xi} \end{pmatrix} < \frac{dN_1}{d\xi} \frac{dN_2}{d\xi} \frac{dN_3}{d\xi} > d\xi \begin{pmatrix} u_1 \\ u_2 \\ u_3 \end{pmatrix}$$

Therefore, the stiffness matrix and the nodal forces vector for this element have the form:

$$K^e = EA \int_{-1}^{1} \left(\frac{d\xi}{dx}\right)^2 \frac{dx}{d\xi} \begin{pmatrix} \frac{dN_1}{d\xi} \\ \frac{dN_2}{d\xi} \\ \frac{dN_3}{d\xi} \end{pmatrix} < \frac{dN_1}{d\xi} \frac{dN_2}{d\xi} \frac{dN_3}{d\xi} > d\xi$$

$$f^e = \int_{-1}^{1} p \begin{pmatrix} N_1 \\ N_2 \\ N_3 \end{pmatrix} \frac{dx}{d\xi} d\xi$$

1.2 Example

Let a nonlinear isoparametric one dimensional element be placed such that the nodal coordinates x_1, x_2, and x_3 are equal to $0, 4.5$, and 10 units, respectively. Assume that a constant load per unit length p is applied in the direction of x. Find the stiffness matrix and the nodal forces for that element.

SOLUTION The mapping function between the two coordinate systems has the form:

$$x = N_1 x_1 + N_2 x_2 + N_3 x_3 = 4.5 + (5 + 0.5\xi)\xi$$

Applying the above equations, the stiffness matrix and the nodal forces for the element have the form:

$$K^e = EA \begin{pmatrix} 0.2653 & -0.3006 & 0.0353 \\ -0.3006 & 0.5465 & -0.2459 \\ 0.0353 & -0.2459 & 0.21067 \end{pmatrix}$$

$$f^e = p \begin{pmatrix} 1.333 \\ 6.667 \\ 2.000 \end{pmatrix}$$

```python
from sympy import *
import sympy as sp
xi, x1, x2, x3, EA, p = sp.symbols("xi x_1 x_2 x_3 EA p")
N1 = -xi*(1-xi)/sp.sympify('2')
N2 = (1-xi)*(1+xi)
N3 = xi*(1+xi)/sp.sympify('2')
display("Shape Functions:", N1,N2,N3)
x = N1*x1+N2*x2+N3*x3
dxxi = x.diff(xi)
dxix = 1/dxxi
x = x.subs({x1:0,x2:4.5,x3:10})
display("Mapping Function: ", x)
B = Matrix([N1.diff(xi), N2.diff(xi), N3.diff(xi)])
K1 = EA * dxix * B.transpose()*B
Kb = integrate(K1, (xi,-1,1))
a = integrate(p*N1*dxxi, (xi,-1,1))
b = integrate(p*N2*dxxi, (xi,-1,1))
c = integrate(p*N3*dxxi, (xi,-1,1))
ff = Matrix([a,b,c]).subs({x1:0,x2:9/sp.sympify('2'),x3:10})
display("Nodal forces: ",ff)
```

Listing 1: Python example

The constitutive relationship of plane linear elastic materials is defined using a 3×3 matrix C that depends on whether the material is in a plane strain or a plane stress state. For the sake of illustration, the plane strain state is chosen here, the material constitutive relationship matrix in that case is:

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

The stiffness matrix of the constant strain triangle can be evaluated by the following integral:

$$K^e = \int_e B^T C B dX = \int_e \begin{pmatrix} -1 & 0 & -1 \\ 0 & -1 & -1 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix} C \begin{pmatrix} -1 & 0 & 1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 & 0 & 1 \\ -1 & -1 & 0 & 1 & 1 & 0 \end{pmatrix} dX$$

Assuming that the triangle has a constant thickness t, then the differential volume $\mathrm{d}X = t\ \mathrm{d}X_1\ \mathrm{d}X_2$. In addition, all the components of the matrix B^TCB are constant; therefore, the integral can be evaluated by simply multiplying the matrix B^TCB by $\frac{t}{2}$, which represents the volume of the triangle:

$$K^e = \frac{tE}{2(1+\nu)} \begin{pmatrix} \frac{3-4\nu}{2-4\nu} & \frac{1}{2-4\nu} & \frac{1-\nu}{-1+2\nu} & \frac{-1}{2} & \frac{-1}{2} & \frac{\nu}{-1+2\nu} \\ \frac{1}{2-4\nu} & \frac{3-4\nu}{2-4\nu} & \frac{\nu}{-1+2\nu} & \frac{-1}{2} & \frac{-1}{2} & \frac{1-\nu}{-1+2\nu} \\ \frac{1-\nu}{-1+2\nu} & \frac{\nu}{-1+2\nu} & \frac{1-\nu}{-1+2\nu} & 0 & 0 & \frac{\nu}{-1+2\nu} \\ \frac{-1}{2} & \frac{-1}{2} & 0 & \frac{1}{2} & \frac{1}{2} & 0 \\ \frac{-1}{2} & \frac{-1}{2} & 0 & \frac{1}{2} & \frac{1}{2} & 0 \\ \frac{\nu}{-1+2\nu} & \frac{1-\nu}{-1+2\nu} & \frac{\nu}{-1+2\nu} & 0 & 0 & \frac{1-\nu}{1-2\nu} \end{pmatrix}$$

```python
import sympy as sp
from sympy import Matrix,simplify
Ee,nu,t = sp.symbols("E nu t")
B = Matrix([[-1,0,1,0,0,0],[0,-1,0,0,0,1],[-1,-1,0,1,1,0]])

Cc = Ee*t/2/(1 + nu)*Matrix([[(1-nu)/(1-2*nu),nu/(1-2*nu),0],
                            [nu/(1-2*nu),(1-nu)/(1-2*nu), 0],
                            [0, 0, 1/2]])
K = simplify(B.T*Cc*B)
display("K^e =",K)
```

Listing 2: Python example

2 TWO DIMENSIONAL 4-NODE BILINEAR ISOPARAMETRIC ELEMENT

Similar to the one dimensional case, an irregular quadrilateral can be defined with four nodes such that the only requirement is that the node numbering is counterclockwise and that the element area is positive. Let $(x_1, y_1), (x_2, y_2), (x_3, y_3)$, and (x_4, y_4) be the two dimensional coordinates of nodes $1, 2, 3$, and 4 , respectively. The element coordinate system ξ and η is chosen such that the coordinates of nodes $1, 2, 3$, and 4 are: $(-1,-1), (1,-1), (1,1)$, and $(-1,1)$, respectively (Figure 2). The mapping between the two coordinate systems is performed using the same shape functions for the bilinear quadrilateral as follows:

$$x = N_1x_1 + N_2x_2 + N_3x_3 + N_4x_4$$
$$y = N_1y_1 + N_2y_2 + N_3y_3 + N_4y_4$$

where

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

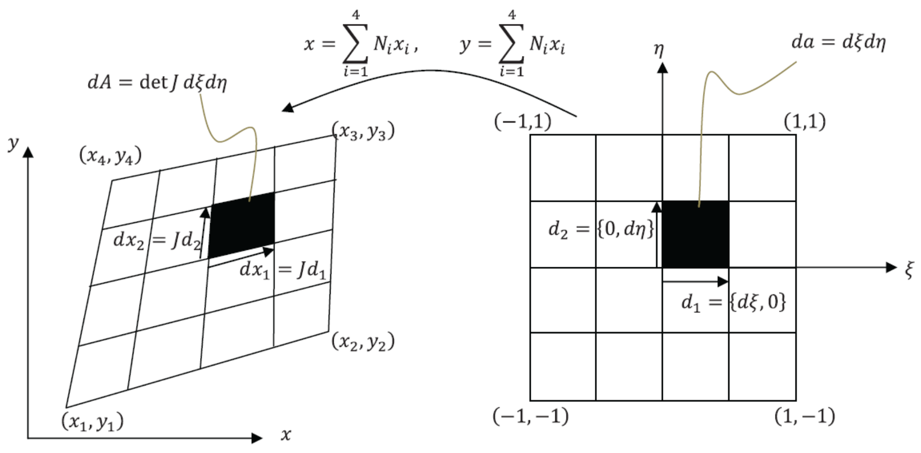


Figure 2: Isoparametric Mapping in the Two Dimensional Case of a 4-node Bilinear Element

The requirement that the element area is positive and that the node numbering is counterclockwise ensures that the Jacobian J of the coordinate transformation is invertible at every point within the element. J and its inverse J^{-1} have the forms:

$$J = \begin{pmatrix} \frac{\partial x}{\partial \xi} & \frac{\partial x}{\partial \eta} \\ \frac{\partial y}{\partial \xi} & \frac{\partial y}{\partial \eta} \end{pmatrix}$$

$$J^{-1} = \begin{pmatrix} \frac{\partial \xi}{\partial x} & \frac{\partial \xi}{\partial y} \\ \frac{\partial \eta}{\partial x} & \frac{\partial \eta}{\partial y} \end{pmatrix}$$

To demonstrate the role of these matrices, let $v_{\xi,\eta} = \{\Delta\xi, \Delta\eta\}$ represent an infinitesimal vector in the element coordinate system ξ, and η. The coordinates of this vector in the general coordinate system x, and y are given by $v_{x,y} = \{\Delta x, \Delta y\}$ such that:

$$\begin{pmatrix} \Delta x \\ \Delta y \end{pmatrix} = \begin{pmatrix} \frac{\partial x}{\partial \xi} & \frac{\partial x}{\partial \eta} \\ \frac{\partial y}{\partial \xi} & \frac{\partial y}{\partial \eta} \end{pmatrix} \begin{pmatrix} \Delta \xi \\ \Delta \eta \end{pmatrix}$$

i.e.,

$$v_{x,y} = J v_{\xi,\eta}$$

Similarly,

$$v_{\xi,\eta} = J^{-1} v_{x,y}$$

The determinant of J can be used to relate areas in the coordinate system defined by x and y and the coordinate system defined by ξ and η (See the determinant section). From Figure 2, let $d_1 = \{\mathrm{d}\xi, 0\}$ and $d_2 = \{0,\ \mathrm{d}\eta\}$ represent two vectors inscribing an area $\mathrm{d}a = \mathrm{d}\xi\mathrm{d}\eta$. Let $dx_1 = Jd_1$, and $dx_2 = Jd_2$ be the same two vectors in the coordinate system defined by x and y. Let the area measure in this coordinate system be $\mathrm{d}A$. The relationship between the two area measures is given by:

$$dA = \det J da = \det J d\xi d\eta$$

This formula can be used to transform integration over the x and y coordinate system to integration over the ξ, and η coordinate system. The displacement function in ξ and η coordinate system is given by:

$$\begin{pmatrix} u \\ v \end{pmatrix} = \begin{pmatrix} N_1 & 0 & N_2 & 0 & N_3 & 0 & N_4 & 0 \\ 0 & N_1 & 0 & N_2 & 0 & N_3 & 0 & N_4 \end{pmatrix} \begin{pmatrix} u_1 \\ v_1 \\ u_2 \\ v_2 \\ u_3 \\ v_3 \\ u_4 \\ v_4 \end{pmatrix}$$

$$= N u_e$$

To evaluate the strains, the gradients of the displacement functions in the two different coordinate systems need to be evaluated. The gradients of the displacement functions in the ξ and η coordinate system have the following form:

$$\begin{pmatrix} \frac{\partial u}{\partial \xi} \\ \frac{\partial u}{\partial \eta} \\ \frac{\partial v}{\partial \xi} \\ \frac{\partial v}{\partial \eta} \end{pmatrix} = \begin{pmatrix} \frac{\partial N_1}{\partial \xi} & 0 & \frac{\partial N_2}{\partial \xi} & 0 & \frac{\partial N_3}{\partial \xi} & 0 & \frac{\partial N_4}{\partial \xi} & 0 \\ \frac{\partial N_1}{\partial \eta} & 0 & \frac{\partial N_2}{\partial \eta} & 0 & \frac{\partial N_3}{\partial \eta} & 0 & \frac{\partial N_4}{\partial \eta} & 0 \\ 0 & \frac{\partial N_1}{\partial \xi} & 0 & \frac{\partial N_2}{\partial \xi} & 0 & \frac{\partial N_3}{\partial \xi} & 0 & \frac{\partial N_4}{\partial \xi} \\ 0 & \frac{\partial N_1}{\partial \eta} & 0 & \frac{\partial N_2}{\partial \eta} & 0 & \frac{\partial N_3}{\partial \eta} & 0 & \frac{\partial N_4}{\partial \eta} \end{pmatrix} \begin{pmatrix} u_1 \\ v_1 \\ u_2 \\ v_2 \\ u_3 \\ v_3 \\ u_4 \\ v_4 \end{pmatrix}$$

The gradients of the displacements functions in the x and y coordinate system can be evaluated as functions of the gradients of the displacements in the ξ and η coordinate system as follows:

$$\begin{pmatrix} \frac{\partial u}{\partial x} \\ \frac{\partial u}{\partial y} \\ \frac{\partial v}{\partial x} \\ \frac{\partial v}{\partial y} \end{pmatrix} = \begin{pmatrix} \frac{\partial \xi}{\partial x} & \frac{\partial \eta}{\partial x} & 0 & 0 \\ \frac{\partial \xi}{\partial y} & \frac{\partial \eta}{\partial y} & 0 & 0 \\ 0 & 0 & \frac{\partial \xi}{\partial x} & \frac{\partial \eta}{\partial x} \\ 0 & 0 & \frac{\partial \xi}{\partial y} & \frac{\partial \eta}{\partial y} \end{pmatrix} \begin{pmatrix} \frac{\partial u}{\partial \xi} \\ \frac{\partial u}{\partial \eta} \\ \frac{\partial v}{\partial \xi} \\ \frac{\partial v}{\partial \eta} \end{pmatrix}$$

$$= \begin{pmatrix} \frac{\partial \xi}{\partial x} & \frac{\partial \eta}{\partial x} & 0 & 0 \\ \frac{\partial \xi}{\partial y} & \frac{\partial \eta}{\partial y} & 0 & 0 \\ 0 & 0 & \frac{\partial \xi}{\partial x} & \frac{\partial \eta}{\partial x} \\ 0 & 0 & \frac{\partial \xi}{\partial y} & \frac{\partial \eta}{\partial y} \end{pmatrix} \begin{pmatrix} \frac{\partial N_1}{\partial \xi} & 0 & \frac{\partial N_2}{\partial \xi} & 0 & \frac{\partial N_3}{\partial \xi} & 0 & \frac{\partial N_4}{\partial \xi} & 0 \\ \frac{\partial N_1}{\partial \eta} & 0 & \frac{\partial N_2}{\partial \eta} & 0 & \frac{\partial N_3}{\partial \eta} & 0 & \frac{\partial N_4}{\partial \eta} & 0 \\ 0 & \frac{\partial N_1}{\partial \xi} & 0 & \frac{\partial N_2}{\partial \xi} & 0 & \frac{\partial N_3}{\partial \xi} & 0 & \frac{\partial N_4}{\partial \xi} \\ 0 & \frac{\partial N_1}{\partial \eta} & 0 & \frac{\partial N_2}{\partial \eta} & 0 & \frac{\partial N_3}{\partial \eta} & 0 & \frac{\partial N_4}{\partial \eta} \end{pmatrix} \begin{pmatrix} u_1 \\ v_1 \\ u_2 \\ v_2 \\ u_3 \\ v_3 \\ u_4 \\ v_4 \end{pmatrix}$$

Therefore, the strains can be written in terms of the degrees of freedom as follows:

$$\varepsilon = B u_e$$

Then, the stiffness matrix of the element can be evaluated as follows:

$$K^e = t \int_{-1}^{1} \int_{-1}^{1} B^T C B \det J d\xi d\eta$$

The element nodal forces can be evaluated as follows:

$$f^e = t \int_{-1}^{1} \int_{-1}^{1} N^t \rho b \det J d\xi d\eta + \int_{\text{element surface}} N^T t_n dS$$

where t is the thickness of the element, C is the plane strain or plane stress material constitutive relationship, ρb is the body forces vector per unit volume, and t_n is the traction vector per unit area on the surface dS. Notice that the surface element dS is given in the x and y coordinate system and it should be transformed accordingly to the ξ and η coordinate system.

Note that the results of this section can be extended in a straightforward manner to the three dimensional 8 node isoparametric brick element.

2.1 Example

Find the stiffness matrix and the nodal forces vector for a 4 node isoparametric plane element whose coordinates are $(0,0),(1,0.1),(1.2,1.2)$, and $(0.2,1)$ with a constant body forces vector $\rho b = \{\rho b_1, \rho b_2\}$ and a unit thickness. Assume plane stress conditions with $E = 1$ units and $\nu = 0$.

The following python code can be utilized to obtain the stiffness matrix and the nodal forces due to a constant body-force vector per unit volume for any 4 node isoparametric element as a function of the coordinates of the 4 nodes of the element. Study the code carefully. In particular, notice the following: If the nodes of the element are changed such that they are not defined in a counterclockwise fashion, then det J becomes negative and the stiffness matrix has negative diagonal entries (The stiffness matrix is not positive definite in that case!). Most finite element analysis software would give a warning message that such element is not adequately defined since the area of the element is negative. Try switching to the builtin exact integration function Integrate to notice the difference in the computational resources required. The numerical integration is a much faster technique.

```python
from sympy import *
from mpmath import *
import sympy as sp
import scipy as sc
import matplotlib.pyplot as plt
from scipy.integrate import dblquad
xi, eta, rb1, rb2 = sp.symbols("xi eta rb_1 rb_2")
coordinates = Matrix([[0,0],[1,0.1],[1.2,1.2],[0.2,1]])
t = 1
display("Plane Stress")
E = 1
nu = 0
Cc = E/(1+nu)/(1-nu)*Matrix([[1,nu,0],[nu,1,0],[0,0,(1-nu)/2]])
a = coordinates.row_insert(4, Matrix([[0,0]]))
fig = plt.figure()
ax = fig.add_subplot(111)
cp = ax.plot(a[:,0], a[:,1])
ax.grid(True, which='both')
plt.xlabel("x")
plt.ylabel("y")
plt.title("Shape")
ax.axhline(y = 0, color = 'k')
ax.axvline(x = 0, color = 'k')
Shapefun = Matrix([[(1-xi)*(1-eta)],[(1+xi)*(1-eta)],
                   [(1+xi)*(1+eta)],[(1-xi)*(1+eta)]])/4
Nn = Matrix([[0 for x in range(8)] for y in range(2)])
for i in range(4):
    Nn[0,2*i] = Nn[1,2*i+1] = Shapefun[i]
x = sum([Shapefun[i]*coordinates[i,0] for i in range(4)])
y = sum([Shapefun[i]*coordinates[i,1] for i in range(4)])
J = Matrix([[x.diff(xi), x.diff(eta)],[y.diff(xi),y.diff(eta)]])
JinvT = J.inv().transpose()
mat1 = Matrix([[1,0,0,0],[0,0,0,1],[0,1,1,0]])
mat2 = Matrix([[JinvT[0,0],JinvT[0,1],0,0],
               [JinvT[1,0],JinvT[1,1],0,0],
               [0,0,JinvT[0,0],JinvT[0,1]],
               [0,0,JinvT[1,0],JinvT[1,1]]])
mat3 = Matrix([[0 for x in range(8)] for y in range(4)])
for i in range(4):
    mat3[0,2*i] = mat3[2,2*i+1] = sp.diff(Shapefun[i],xi)
    mat3[1,2*i] = mat3[3,2*i+1] = sp.diff(Shapefun[i],eta)
B = mat1*mat2*mat3
k1 = B.transpose()*Cc*B
K = zeros(8)
for i in range(8):
    for j in range(8):
        integrand=k1[i,j]*J.det()
        intlam=lambdify((eta,xi),integrand)
        #mp.dps is for integral accuracy (# of decimal points)
        mp.dps = 3
        K[i,j] = quad(intlam,[-1,1],[-1,1])
# Scipy method, results in ZeroDivsionError
# K = sc.zeros(k1.shape, dtype = float)
# for (i,j), expr in sc.ndenumerate(k1):
#     tmp = sp.lambdify((xi, eta), expr*J.det(), 'math')
#     K[i,j] = dblquad(tmp, -1, 1, lambda xi: -1, lambda xi:1)[0]
display(K)
rb = Matrix([rb1,rb2])
fbeforeintegration = Nn.transpose()*rb*J.det()
f3 = Matrix([integrate(fbeforeintegration[i],(xi,-1,1),(eta,-1,1))for i in range(8)])
display("Nodal Forces: ", f3)
```

Listing 3: Python example

3 References

- Adeeb, S. (2011). Introduction to solid mechanics and finite element analysis using Mathematica. Kendall Hunt.