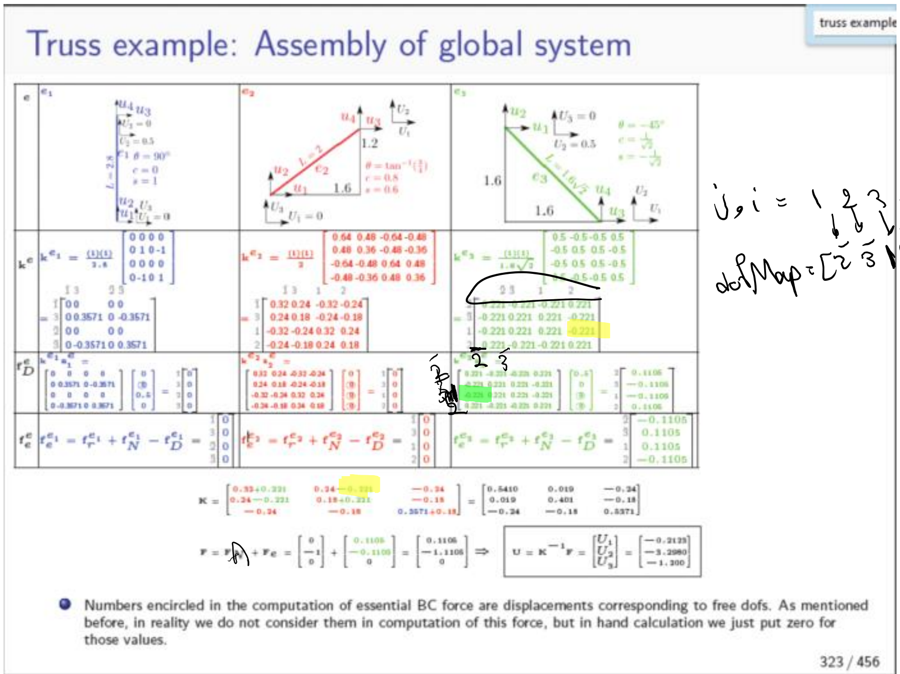
2022/11/10b

Thursday, November 10, 2022 12:56 PM



j, i = 1 2 3 4

dofMap = [$\bar{2}$ $\bar{3}$ 1 2]

for i = 1: ndof (e.g. 4 here)

I = dofMap(i)

if I < 0

Continue;

end

free row I>0

for j=1:ndof

J = dofMap(j);

if J>0 / free row & colum

K(I,J)=K(I,J)+ke(i,j);

else // J<0

fD(i)=K(i,j)*ae(j)

end

fee(i) = fee(i) − fD(i);

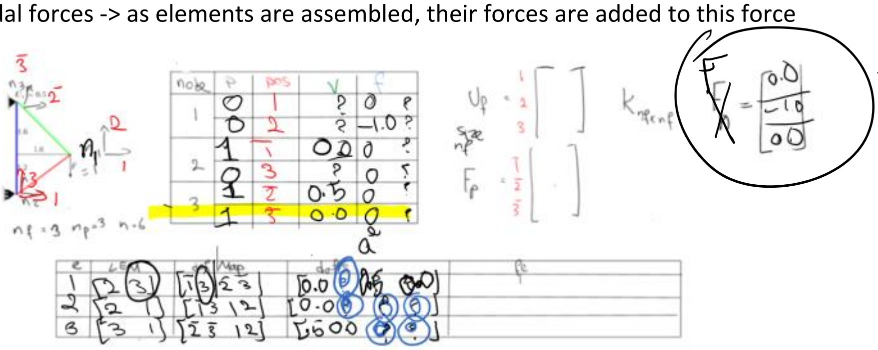
F(I) = F(I)+fee(i);

end

```
for e = 1:ne loop over elements
  fee = feo element total force = element all forces except essential force
  for i = 1:nedof loop over rows of ke; nedof = element # dof
    I = dofMap(i) local to global dof map M_t^e
    if (I > 0) I corresponds to a free dof, we skip prescribed dofs
      for j = 1:nedof loop over columns of ke
        J = dofMap(j) global dof corresponding to j
        if (J > 0) now both I and J are free and can add ke(i,j) to global K
          K(I, J) = K(I, J) + ke(i, j)
        else J < 0, prescribed dof j; add contributions of f_D^e = k^e a^e to f_e^e
          fee(i) = fee(i) - ke(i, j) * edofs(j) edofs: element dofs = a^e
        end
      end
      F(I) = F(I) + fee(i)element's total force fee component i'th is computed→added to F(I)
    end
  end
end
```

431 / 456

In programming to reduce memory usage, We'll have only One F, it's initialized to nodal forces -> as elements are assembled, their forces are added to this force

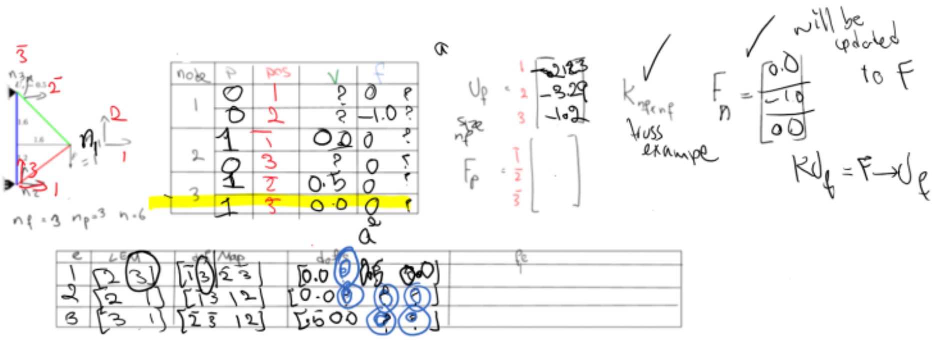


Next step is calculating a:

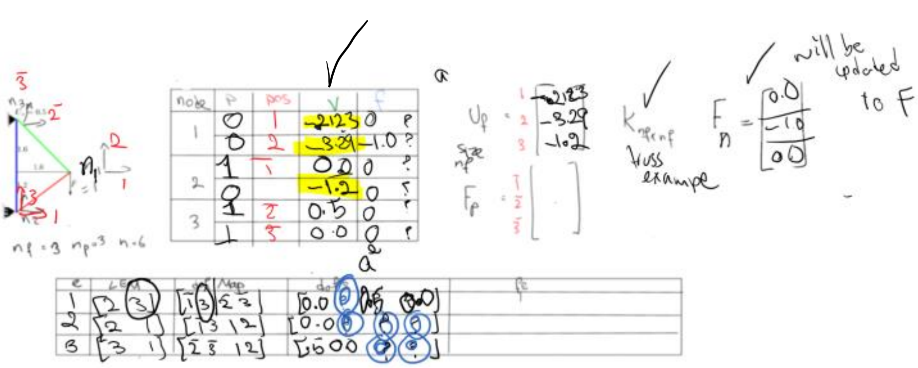
Step 12: Solve global (free) dof $\mathbf{a}$ from $\mathbf{Ka} = \mathbf{F}$

- Two major computational costs during FEM solve are:
  1. Assembly: Refers to: all node, element, and dof set up; computation of local ke and fee; assembly of those to global system. This step scales linearly versus n_e (ne)
  2. Linear algebra solution: $\mathbf{Ka} = \mathbf{F}$: We solve for unknown $\mathbf{a}$. Although conceptually simple, this step is a major source of computational cost. It scales higher than linear versus n_e $\Rightarrow$ As the problem size increases this term becomes more dominant.
- Solution of $\mathbf{Ka} = \mathbf{F}$:
  - WE DO NOT OBTAIN $\mathbf{a}$ from $\mathbf{a} = \mathbf{K}^{-1}\mathbf{F}$: We do not invert $\mathbf{K}$.
  - We only solve the problem for the specific RHS of $\mathbf{F}$.
  - In Comparison $\mathbf{K}^{-1}$ corresponds to the solution of $\mathbf{Ka} = \mathbf{F}$ for n_f RHS of $\mathbf{F} = e_i,\ i = 1, \ldots, n_f$ where n_f is the number of rows (and columns) of $\mathbf{K}$.
  - We employ methods such as LU factorization that computationally only solve the problem for the given RHS $\mathbf{F}$.
  - We take advantage of the structure of stiffness matrix: symmetry, bandedness, sparsity in choosing the right solution technique.
  - order of free dofs affects band of the matrix $\to$ various algorithms reorder free dofs such that the matrix band get smaller and the solution cost is optimized.
  - In your term projects you can simply employ simply compute

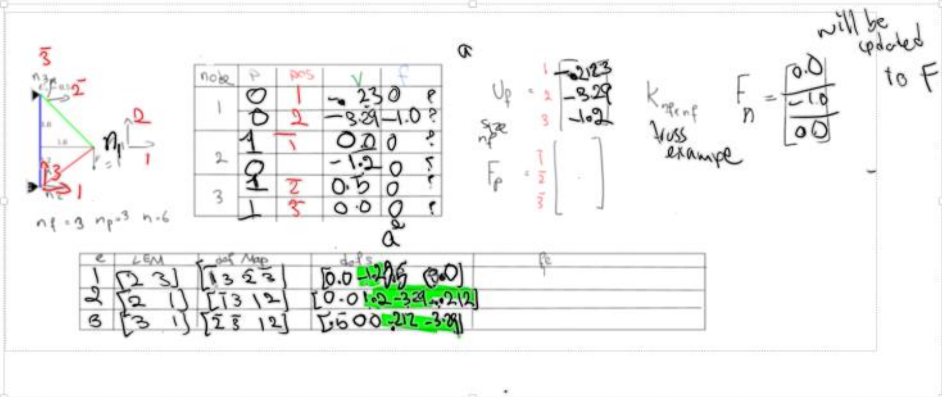
$a = K\backslash F;$



Step 13: Assign $\mathbf{a}$ to nodes and elements



Step 13: Assign $\mathbf{a}$ to nodes and elements



13.a nodal values

- Loop over nodes (outer loop), loop over nodes' dofs (inner loop), for free dofs set their values equal to their corresponding values in dofs (a).

```
for n = 1:nNodes
  for dofi = 1: node(n).nndof num dof for node (n)
    if node(n).ndof(dofi).p == false free dof
      posn = node(n).ndof(dofi).pos position of dof in global free F
      node(n).ndof(dofi).v = dofs(posn) set free dof val to corresponding val in global dofs (a)
    end
  end
end
```

435 / 456

```
for e = 1:ne loop over elements
  for i = element(e).nedof loop over element dofs; nedof = # dof (n^e_dof)
    posn = element(e).dofMap(i) corresponding global position using dofMat (M^e_t)
    if (posn > 0) free dof
      element(e).edofs(i) = dofs(posn)
        set free element dof a^e to corresponding val in global dofs (a)
    end
  end
end
```

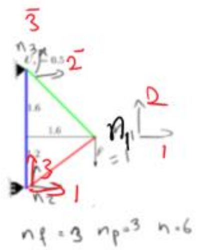
436 / 456

Very important:
Whether indices start from zero or 1: Starting from 1 All Good

Starting with 0:
In the element dofMap, we would end up with (0, 1, 2, …) for global free dofs (0, 1, 2) and with (-0, -1, -2, -3, ..) for global prescribed dofs (0, 1, ….) so 0 is double used.

My trick is in eDOF I subtract 1 from the prescribed dofs global prescribed dofs (0, 1, 2) will become (-1, -2, -3)

Step 14: Compute prescribed dof forces



node	p	pos	v	f
1	0	1	-.2123	?
	0	2	-3.29	-1.0 ?
2	1	$\bar{1}$	0.0	0 ?
	0		-1.2	0 ?
3	1	$\bar{2}$	0.5	0 ?
	1	$\bar{3}$	0.0	0 ?

a

$$U_f = \begin{bmatrix} -.2123 \\ -3.29 \\ -1.02 \end{bmatrix} \quad \text{size } n_f$$

$$F_p = \begin{bmatrix} \\ \\ \\ \end{bmatrix}$$

$K_{n_f \times n_f}$ truss example

$$F_n = \begin{bmatrix} 0.0 \\ -1.0 \\ 0.0 \end{bmatrix}$$ will be updated to F

$-f^e_D$

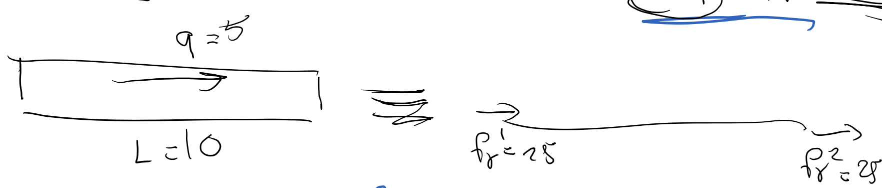
e	LEM	dof Map	dofs	$-f^e = f^e_r - f^e_N + ka$
1	[2 3]	[$\bar{1}$ 3 $\bar{2}$ $\bar{3}$]	[0.0 -1.2 .5 0.0]	[0, -.4285, 0, .4285]
2	[2 1]	[$\bar{1}$ 3 1 2]	[0.0 -1.2 -3.29 -.2123]	[.5715 .4286 -.5715 -.4286]
3	[3 1]	[$\bar{2}$ $\bar{3}$ 1 2]	[.5 0.0 -.2123 -3.29]	[-.5715 .5715 .5715 -.5715]

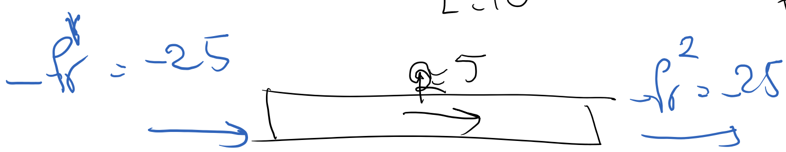
$f = ka$ good for the project

general

$$-f^e = -f^e_r - f^e_N \cdots + f^e_D$$

$$= -f^e_r - f^e_N + ka$$

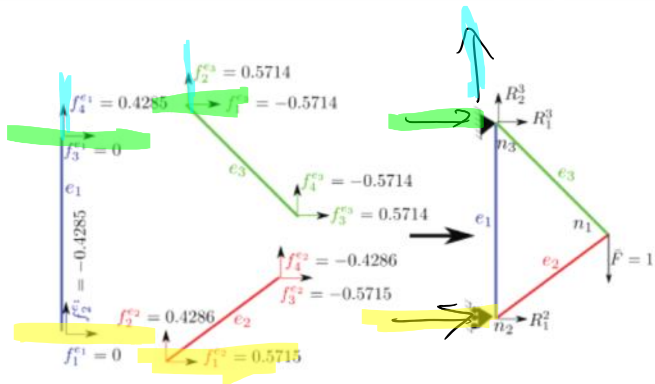




Next, are the support forces OR the forces of the prescribed dofs

Truss Example: Reaction Forces

truss examp



- First, we compute reaction forces by adding up forces from individual elements that contribute to reaction forces:

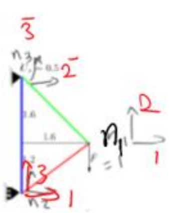
$$R_1^2 = f_1^{e_1} + f_1^{e_2} = 0 + 0.5715 = 0.5715 \tag{397a}$$

$$R_1^3 = f_3^{e_1} + f_1^{e_3} = 0 + -0.5714 = -0.5714 \tag{397b}$$

$$R_2^3 = f_4^{e_1} + f_2^{e_3} = 0.4285 + 0.5714 = 0.9999 \tag{397c}$$

What we need to do, for each element

Loop over its dofs, those that are prescribed, add their forces to the global Fp



nf = 3 np = 3 n = 6

node	P	pos	V	F	
1	0	1	-.2.3	0	?
	0	2	-3.29	-1.0	?
2	1	$\bar{1}$	0.0	0	?
	0		-1.2	0	?
3	1	$\bar{2}$	0.5	0	?
	1	$\bar{3}$	0.0	0	?

a

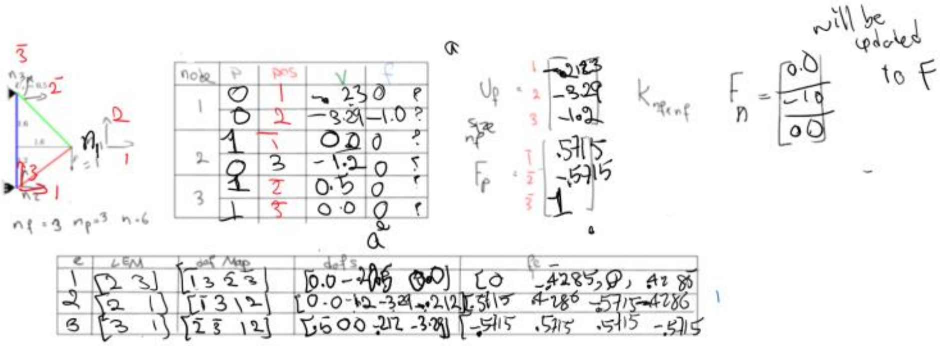
$U_f = \begin{bmatrix} -.2123 \\ -3.29 \\ -1.02 \end{bmatrix}$ $K_{nf \times nf}$

size nf

$F_p = \begin{bmatrix} 0 + .5715 \\ 0 - .5715 \\ .4285 + .5715 \end{bmatrix}$

$F_n = \begin{bmatrix} 0.0 \\ -1.0 \\ 0.0 \end{bmatrix}$ will be updated to F

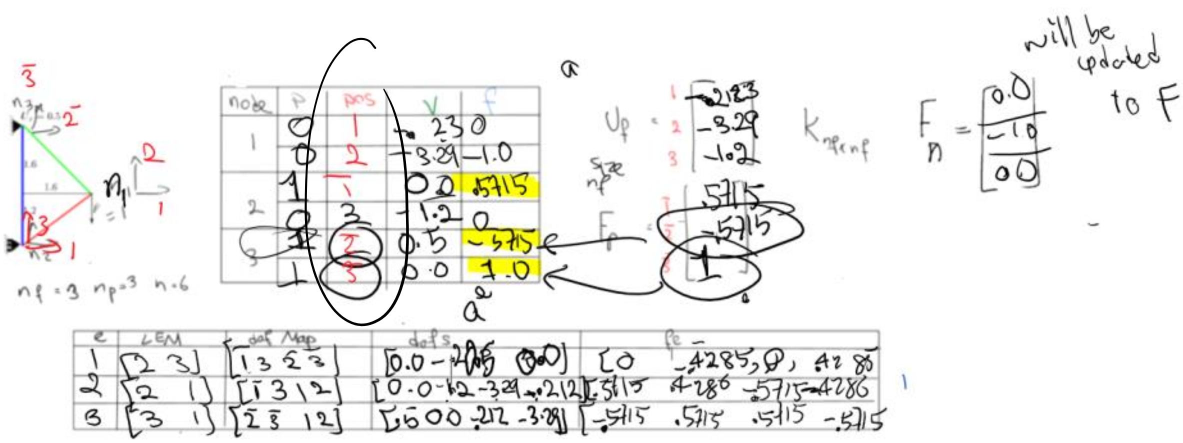
e	LEM	dof Map	dofs	fe
1	[2 3]	[$\bar{1}$ 3 $\bar{2}$ $\bar{3}$]	[0.0 -.2125 0.0]	[0, -.4285, 0, .4285]
2	[2 1]	[$\bar{1}$ 3 1 2]	[0.0 -1.2 -3.29 -.2123]	[.5715 .4286 -.5715 -.4286]
3	[3 1]	[$\bar{2}$ $\bar{3}$ 1 2]	[.5 0.0 -.212 -3.29]	[-.5715 .5715 .5715 -.5715]



for e = 1:ne loop over elements
- fee = feo element total force = element all forces except essential force
- for i = 1:nedof loop over rows of ke; nedof = element # dof
  - I = dofMap(i) local to global dof map M_t^e
  - if (I < 0) I corresponds to a prescribed dof, we skip free dofs
    - for j = 1:nedof loop over columns of ke. ALL columns (dofs) of p and f used
      - fee(i) = fee(i) - ke(i, j) * edofs(j) edofs: element dofs = a^e
    - end
    - Fp(-I) = Fp(-I) - fee(i)
      1. element's total force fee component i'th is computed→added to Fp(-I)
      2. -I used because I < 0: *prescribed* dof
      3. fee is **subtracted**
  - end
- end

end

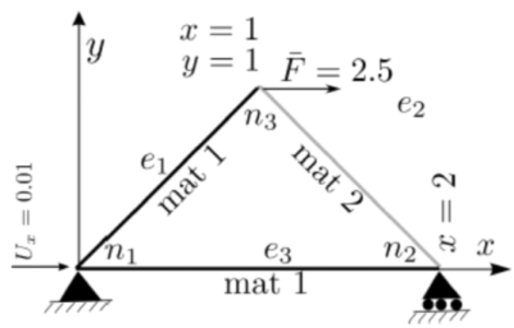
Prescribed nodeal forces in node output



```
for n = 1:nNodes
  for dofi = 1: node(n).nndof    num dof for node (n)
    if node(n).ndof(dofi).p == true    prescribed dof
      posn = node(n).ndof(dofi).pos    position of dof in global prescribed force Fp
      node(n).ndof(dofi).f = Fp(-posn)
        1. set prescribed dof force to corresponding force in global Fp (Fp)
        2. posn < 0; prescribed dof
    end
  end
end
```


Input file format

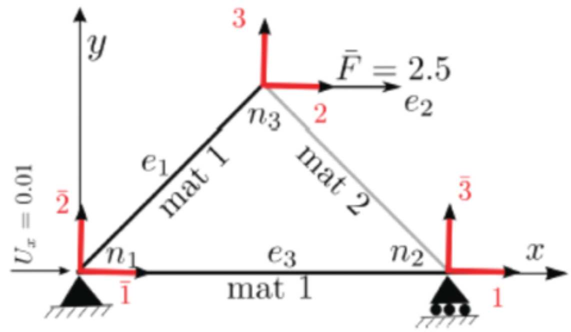
```
dim 2
ndofpn 2
Nodes
nNodes 3
id crd
1 0 0
2 2 0
3 1 1
Elements
ne 3
id elementType matID neNodes eNodes
1 3 1 2 1 3
2 3 2 2 3 2
3 3 1 2 1 2
PrescribedDOF
np 3
node node_dof_index value
1 1 0.01
1 2 0
2 2 0
FreeDOFs
nNonZeroForceFDOFs 1
node node_dof_index value
3 1 2.5
Materials
nMat 2
id numPara Paras
1 2 100 1
2 2 200 2
```



$E_1 = 100, A_1 = 1$
$E_2 = 200, A_2 = 2$

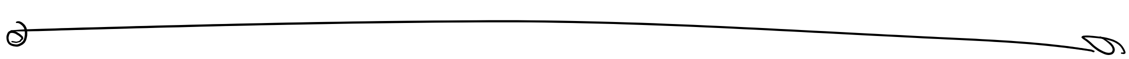
Output file format

```
Nodes
nNodes 3
id crd
values
forces
position(verbose)
prescribed_boolean(verbose)
1 0 0
a1_1 a1_2
F1_1 F1_2
-1 -2 (verbose)
1 1 (verbose)
2 2 0
a2_1 a2_2
F2_1 F2_2
1 -3 (verbose)
0 1 (verbose)
a3_1 a3_2
F3_1 F3_2
2 3 (verbose)
0 0 (verbose)
Elements
ne 3
id elementType
forces(verbose)
specific output
1 3
fee1_1 fee1_2 fee1_3 fee1_4 (verbose)
Te1
2 3
fee2_1 fee2_2 fee2_3 fee2_4 (verbose)
Te2
3 3
fee3_1 fee3_2 fee3_3 fee3_4 (verbose)
Te3
```

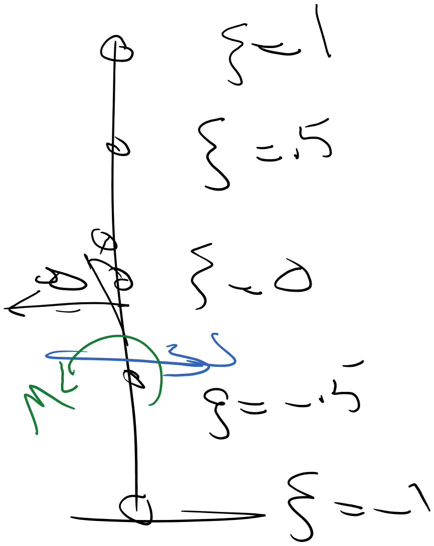


- lines with (verbose) are only output for verboseOutput == 1. Obviously (verbose) is not printed in either case and is only printed for clarity here.
- ai_j: is value (solution) for node i dof number j;*e.g.*, a3_1 is x displacement at node 3 ($x = 1, y = 1$).
- Fi_j: is force for node i dof number j;*e.g.*, F3_1 is x force at node 3 (which should be equal to 2.5, why?)
- feei_j: is total force (foe + fDe) for element i dof number j;*e.g.*, fee3_1 is the x force at its left node (global n_1).
- Last item of element output is specific to its type.
- For 2 node bar and truss elements Tei is the axial force in the element.

frame



calculate y, θ, V, M



Beam Example: Calculation of y, θ, M, V within element

- After the solution of global free dofs, they are transferred to elements.
- Once element dofs are know, we have the Displacement in the entire elements:

$$y(\xi) = N_1^e(\xi)a_1^e + N_2^e(\xi)a_2^e + N_3^e(\xi)a_3^e + N_4^e(\xi)a_4^e.$$

element shape functions are given in (421).

- Rotation: Obtained by differentiating previous equation w.r.t. x & noting that $\frac{\mathrm{d}x}{\mathrm{d}\xi} = \frac{L^e}{2}$:

$$\theta(\xi) = \frac{\mathrm{d}y}{\mathrm{d}x}(\xi) = \frac{\frac{\mathrm{d}y}{\mathrm{d}\xi}(\xi)}{\frac{\mathrm{d}x}{\mathrm{d}\xi}(\xi)} = \frac{2}{L^e}\left\{\frac{\mathrm{d}N_1^e}{\mathrm{d}\xi}(\xi)a_1^e + \frac{\mathrm{d}N_2^e}{\mathrm{d}\xi}(\xi)a_2^e + \frac{\mathrm{d}N_3^e}{\mathrm{d}\xi}(\xi)a_3^e + \frac{\mathrm{d}N_4^e}{\mathrm{d}\xi}(\xi)a_4^e\right\}$$

- Moment is directly obtained by differentiating the above equation:

$$\begin{aligned} M(\xi) &= E(\xi)I(\xi)\frac{\mathrm{d}^2 y}{\mathrm{d}x^2}(\xi) = E(\xi)I(\xi)\mathbf{B}^e(\xi) \\ &= E(\xi)I(\xi)\left\{B_1^e(\xi)a_1^e + B_2^e(\xi)a_2^e + B_3^e(\xi)a_3^e + B_4^e(\xi)a_4^e\right\} \quad \textit{cf. } (424) \text{ for } \mathbf{B}^e \end{aligned}$$

- Shear force is obtained by differentiating M w.r.t. x. It's a similar process to deriving θ from y with the difference that if EI are not constant we need to take it into account. For constant EI we have:

$$V(\xi) = \frac{\mathrm{d}M}{\mathrm{d}x}(\xi) = \frac{\frac{\mathrm{d}M}{\mathrm{d}\xi}(\xi)}{\frac{\mathrm{d}x}{\mathrm{d}\xi}(\xi)} = \frac{2EI}{L^e}\left\{\frac{\mathrm{d}B_1^e}{\mathrm{d}\xi}(\xi)a_1^e + \frac{\mathrm{d}B_2^e}{\mathrm{d}\xi}(\xi)a_2^e + \frac{\mathrm{d}B_3^e}{\mathrm{d}\xi}(\xi)a_3^e + \frac{\mathrm{d}B_4^e}{\mathrm{d}\xi}(\xi)a_4^e\right\}$$

- To obtain these fields for the entire beam we evaluate these equations for all elements.