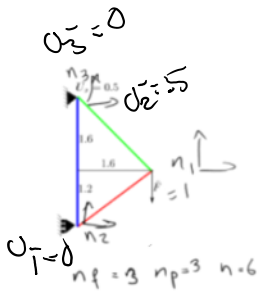


Step 4: Set global prescribed nodal dof



node	P	pos	v	f
1				
2				
3				

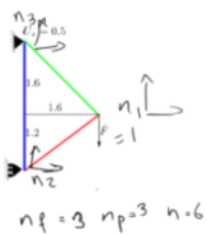
$$K_{n \times n} = \begin{bmatrix} 1 & & \\ & 2 & \\ & & 3 \end{bmatrix}$$

$$F_p = \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix}$$

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

$n_{p=3}$
 $n_{p=3}$

e	LEM	dof Map	dofs	fe



node	P	pos	v	f
1	0		2	2
2	1		0.0	2
3	0		0.5 0.0	2

$$K_{n \times n} = \begin{bmatrix} 1 & & \\ & 2 & \\ & & 3 \end{bmatrix}$$

$$F_p = \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix}$$

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

e	LEM	dof Map	dofs	fe

np 3
node node_dof_index value
2 1 0.0
3 1 0.5
3 2 0.0

Step 5: Set global free nodal dof

All dofs in FEM are by default FREE with ZERO "force". Since from step 4, we already know which dofs are prescribed (-> we'd know which ones are free), here we only need to provide free dofs with nonzero force



node	p	pos	v	f
1	0		?	0
2	1		0	0
3	0		0	0

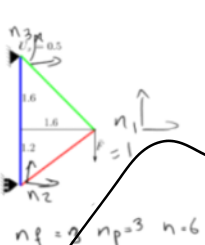
$$U_p = \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix} \quad K_{ref} \quad F_n = \begin{bmatrix} \\ \\ \end{bmatrix}$$

$$F_p = \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix}$$

e	LEM	dof Map	dofs	fe

FreeDofs
 nNonZeroForceFDOFs 1
 node node_dof_index value
 1 2 -1.0

Step 6: dof positions; Step 7: Set F(F_f)



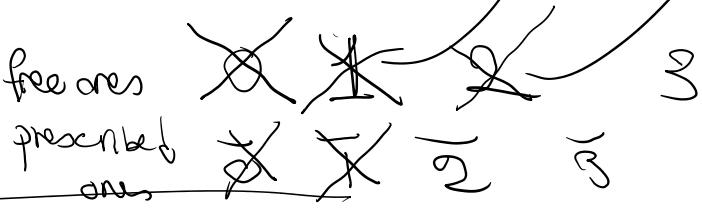
node	p	pos	v	f
1	0		?	0
2	1		0	0
3	1		0	0

$$U_p = \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix} \quad K_{ref} \quad F_n = \begin{bmatrix} \\ \\ \end{bmatrix}$$

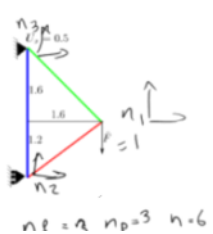
$$F_p = \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix}$$

e	LEM	dof Map	dofs	fe

FreeDofs
 nNonZeroForceFDOFs 1
 node node_dof_index value
 1 2 -1.0



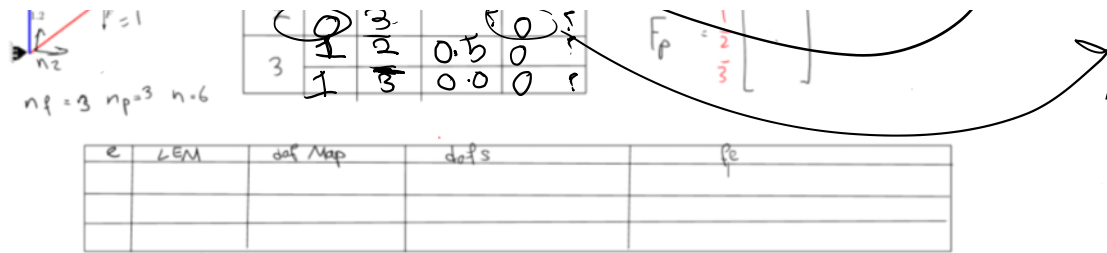
Step 7: Set F



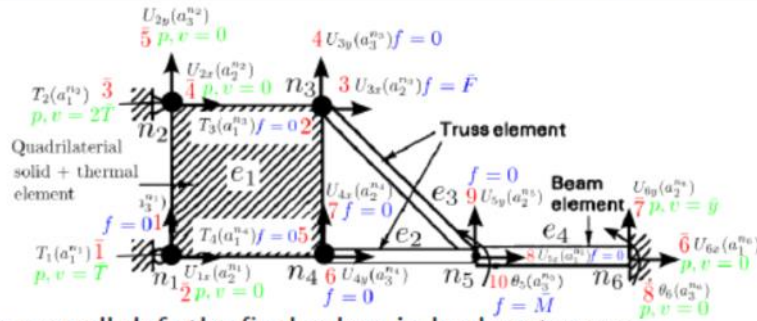
node	p	pos	v	f
1	0		?	0
2	1		0	0
3	1		0	0

$$U_p = \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix} \quad K_{ref} \quad F_n = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

$$F_p = \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix}$$



Step 6: dof positions; Step 7: Set $F(F_f)$



- After looping over all dofs the final values in load vector are:

$$F = [0 \quad 0 \quad \bar{F} \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad \bar{M}]$$

posf = 0, posp = 0

for n = 1:nNodes

for dofi = 1: node(n).ndof num dof for node (n)

if node(n).ndof(dofi).p == true prescribed dof

posp = posp - 1;

node(n).ndof(dofi).pos = posp;

step 6

else free dof

posf = posf + 1;

node(n).ndof(dofi).pos = posf;

step 6

F(posf) = node(n).ndof(dofi).f

step 7

end

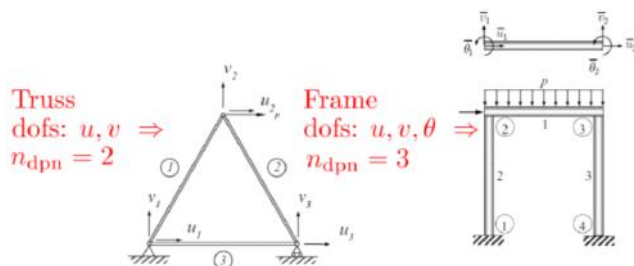
end

end

415 / 456

Step 8: Element dof maps M_t^e

Step 8: Element dof maps M_t^e : Simplified limited case

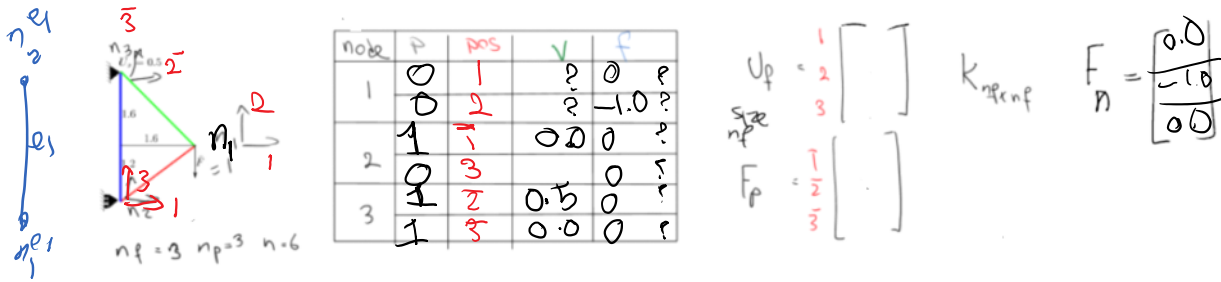


n_e 3



1 2

n_e



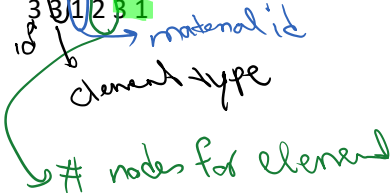
e	LEM	dof Map	dofs	fe
1	[1 2 3]	[1 3 2 3]		
2	[2 3 1]	[1 3 1 2]		
3	[3 1 1]	[2 3 1 2]		

For step 8, to form dofMap, we first need to know the map between element nodes and the global domain nodes -> NodalMap or LEM.
How do we get this?

Elements
ne 3

id elementType matID neNodes eNodes

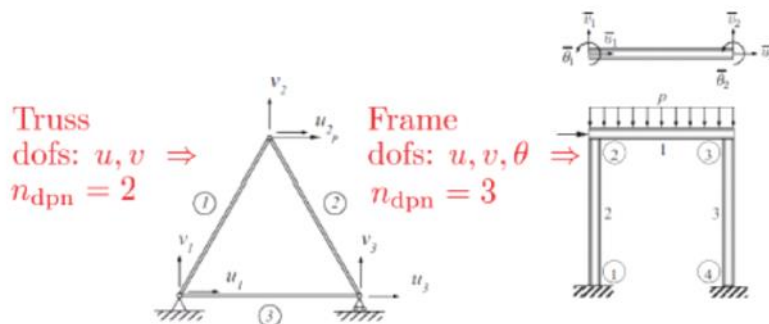
1	3	1	2	3
2	3	1	2	1
3	3	1	2	3



1. bar
2. beam
3. truss
4. frame

Slide 420:

Step 8: Element dof maps M_t^e : Simplified limited case

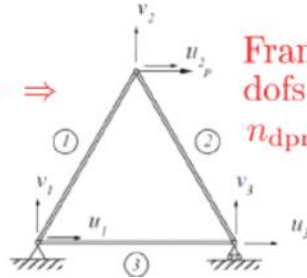


A simplified pseudo code looks like: $ecdof = 1$ dof counter for element
for $en = 1: neNodes$ number of element nodes
 $gn = LEM(en)$ global node number for element node en
 for $endof = 1: ndofpn$ This number is fixed now, e.g., 2 for 2D trusses
 $dofMap(ecdof) = node(gn).dof(endof).pos$
 $gndof = endof$, we bypass some steps here
 $ecdof = ecdof + 1$ increment counter
 end
end

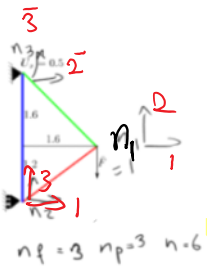
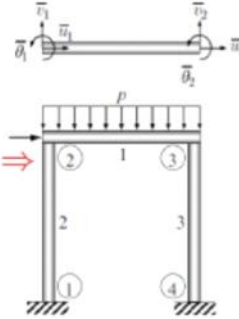
end
end

Step 9: Set element dofs a^e: Simplified limited case

Truss
dofs: $u, v \Rightarrow$
 $n_{dofn} = 2$



Frame
dofs: $u, v, \theta \Rightarrow$
 $n_{dofn} = 3$



node	p	pos	v	f
1	0	1	?	?
2	1	2	0.0	?
3	1	3	0.5	?

$$U_p = \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix}$$

$$F_p = \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix}$$

$$K_{ref} \quad F_n = \begin{bmatrix} 0.0 \\ 1.0 \\ 0.0 \end{bmatrix}$$

e	LEM	Map	dof	fe
1	[1 3]	[1 3 2 2]	[0.0 0.0 0.0]	
2	[2 1]	[1 3 1 2]	[0.0 0.0 0.0]	
3	[3 1]	[2 3 1 2]	[0.5 0.0 0.0]	

we need this

$$f^e = k^e \underline{a}^e$$

steps 8 & 9

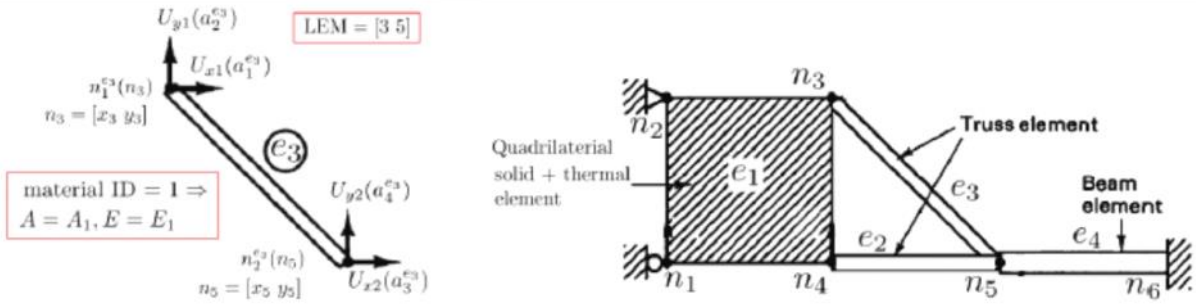
```

dofs = zeros(nedof) element dofs (edof) resized to number of element dofs and zeroed
ecdof = 1 dof counter for element
for en = 1: neNodes number of element nodes
    gn = LEM(en) global node number for element node en
    for endof = 1: ndofpn This number is fixed now, e.g., 2 for 2D trusses
        if (node(gn).dof(endof).p == true) gn dof = endof, we bypass some steps here
            dofs(ecdof) = node(gn).dof(endof).value; e dof val = corresponding global val
        end
        dofMap(ecdof) = node(gn).dof(endof).pos
        ecdof = ecdof + 1 increment counter
    end
end
end

```

added for step 9 a^e (dofs) is updated

Step 10: Compute element stiffness/force



Every element has its own implementation of stiffness calculation.

Parent class: PhyElement (this is a generic element)

```
class PhyElement
```

```
{
// Step 10: Compute element stiffness/force (ke, foe (fre: source term; fNe: Neumann BC))
virtual void Calculate_ElementStiffness_Force() = 0;
}
```

parent does not have any implementation

Children (bar, beam, ...) may have different implementations

Child (subclass)

PhyElementBar.h

```
class PhyElementBar : public PhyElement
```

```
{
....
virtual void Calculate_ElementStiffness_Force();
}
```

```
void PhyElementBar::Calculate_ElementStiffness_Force()
```

```
{
// compute stiffness matrix:
ke.resize(2, 2);
double factor = A * E / L;
ke(0, 0) = ke(1, 1) = factor;
ke(1, 0) = ke(0, 1) = -factor;
}
```

$$K_e = \frac{AE}{L} \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix}$$

```
void PhyElementTruss::Calculate_ElementStiffness_Force()
```

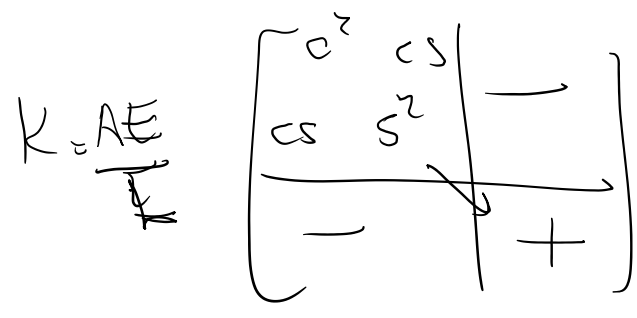
```
{
// compute stiffness matrix:
ke.resize(4, 4);
double factor = A * E / L;
for (int l = 0; l < 2; ++l)
for (int i = 0; i < 2; ++i)
```

~ 2 1

```

double factor = A * E / L;
for (int I = 0; I < 2; ++I)
  for (int J = 0; J < 2; ++J)
  {
    double f2 = factor;
    if ((I + J) % 2 != 0)
      f2 = -factor;
    ke(2 * I, 2 * J) = c * c * f2;
    ke(2 * I + 1, 2 * J) = ke(2 * I, 2 * J + 1) = c * s * f2;
    ke(2 * I + 1, 2 * J + 1) = s * s * f2;
  }
cout << "ke\n" << ke << endl;
}

```



Subroutine based

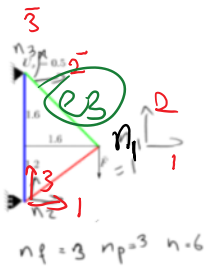
Calculate Stiffness ()

```

}
if (eType == 1) // bar
  K = AE * [ 1 -1 ]
            [ -1 1 ]
else if (eType == 2)
  }
}

```

How do you get L, c, s for a truss element w/o a figure?



node	P	POS	V	F
1	0	1	?	0
2	1	2	0	-1.0
3	2	3	0.5	0

$$U_p = \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix}$$

$$F_p = \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix}$$

$$K_{ref} = \frac{F}{n} = \begin{bmatrix} 0.0 \\ -1.0 \\ 0.0 \end{bmatrix}$$

e	LEM	Map	def	fe
1	2 3	[1 3 2]	[0.0 0.0 0.0]	
2	2 1	[1 3 2]	[0.0 0.0 0.0]	
3	3 1	[2 3 1]	[1.6 0.0 0.0]	

we need this

Input file

...
 nNodes 3
 id crd
 1 1.6 1.2
 2 0 0
 3 0 2.8

$$D_{e3} = \text{Coord}(node 1) - \text{Coord}(node 3)$$

$$= [1.6, 1.2] - [0, 2.8]$$

$$= [1.6, -1.6]$$

$$L_{e3} = \sqrt{(1.6)^2 + (-1.6)^2}$$

$$\cos \alpha_{e3} = \frac{(D_{e3})(1)}{L} = \frac{1.6}{2} = \frac{\sqrt{2}}{2}$$

Step 11: Assembly from local to global system

Truss example: Assembly of global system

e_1 	e_2 	e_3
$k^{e_1} = \frac{(1)(1)}{2.8} \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & -1 \\ 0 & 0 & 0 & 0 \\ -0.1 & 0 & 1 & 0 \end{bmatrix}$	$k^{e_2} = \frac{(1)(1)}{2} \begin{bmatrix} 0.64 & 0.48 & -0.64 & -0.48 \\ 0.48 & 0.36 & -0.48 & -0.36 \\ -0.64 & -0.48 & 0.64 & 0.48 \\ -0.48 & -0.36 & 0.48 & 0.36 \end{bmatrix}$	$k^{e_3} = \frac{(1)(1)}{1.6\sqrt{2}} \begin{bmatrix} 0.5 & -0.5 & -0.5 & 0.5 \\ -0.5 & 0.5 & 0.5 & -0.5 \\ -0.5 & 0.5 & 0.5 & -0.5 \\ 0.5 & -0.5 & -0.5 & 0.5 \end{bmatrix}$
$r_D^{e_1} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0.3571 & 0 & -0.3571 \\ 0 & 0 & 0 & 0 \\ 0 & -0.3571 & 0 & 0.3571 \end{bmatrix}$	$r_D^{e_2} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0.32 & 0.24 & -0.32 & -0.24 \\ 0.24 & 0.18 & -0.24 & -0.18 \\ -0.32 & -0.24 & 0.32 & 0.24 \\ -0.24 & -0.18 & 0.24 & 0.18 \end{bmatrix}$	$r_D^{e_3} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0.221 & -0.221 & -0.221 & 0.221 \\ -0.221 & 0.221 & 0.221 & -0.221 \\ -0.221 & 0.221 & 0.221 & -0.221 \\ 0.221 & -0.221 & -0.221 & 0.221 \end{bmatrix}$
$r_N^{e_1} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$	$r_N^{e_2} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$	$r_N^{e_3} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$
$r_F^{e_1} = r_N^{e_1} + r_N^{e_1} - r_D^{e_1} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$	$r_F^{e_2} = r_N^{e_2} + r_N^{e_2} - r_D^{e_2} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$	$r_F^{e_3} = r_N^{e_3} + r_N^{e_3} - r_D^{e_3} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$

$K = \begin{bmatrix} 0.32+0.221 & 0.24-0.221 & -0.24 & 0.019 \\ 0.24-0.221 & 0.18+0.221 & -0.18 & -0.24 \\ -0.24 & -0.18 & 0.3571+0.18 & 0.019 \\ 0.019 & -0.24 & -0.18 & 0.6371 \end{bmatrix}$

$F = F_N + F_e = \begin{bmatrix} -1.1105 \\ -0.1105 \\ 0 \\ 0 \end{bmatrix}$

$U = K^{-1}F = \begin{bmatrix} U_1 \\ U_2 \\ U_3 \end{bmatrix} = \begin{bmatrix} -0.2123 \\ -3.2980 \\ -1.200 \end{bmatrix}$

Handwritten: K & F any row is prescribed useless

Handwritten: global K

Handwritten: global Fe

Numbers encircled in the computation of essential force are displacements corresponding to free dofs. As mentioned before, in reality we do not consider them in computation of this force, but in hand calculation we just put zero for those values.

Summary:

Prescribed rows -> useless
=>

Only work with prescribed ROWS

Free column -> Stiffness is updated
else
Prescribed column ->

Handwritten: K_{ij} is updated

for $e = 1:ne$ loop over elements

$f_{ee} = f_{eo}$ element total force = element all forces except essential force

for $i = 1:nedof$ loop over rows of k_e ; $nedof =$ element # dof

$l = dofMap(i)$ local to global dof map M_l^e

if $(l > 0)$ l corresponds to a free dof, we skip prescribed dofs

for $j = 1:nedof$ loop over columns of k_e

$J = dofMap(j)$ global dof corresponding to j

if $(J > 0)$ now both l and J are free and can add $k_e(i,j)$ to global K

$K(l, J) = K(l, J) + k_e(i, j)$

else $J < 0$, prescribed dof j ; add contributions of $f_D^e = k_e a^e$ to f_e^e

$f_{ee}(i) = f_{ee}(i) - k_e(i, j) * edofs(j)$ $edofs:$ element dofs = a^e

end

end

$F(l) = F(l) + f_{ee}(i)$ element's total force f_{ee} component i 'th is computed -> added to $F(l)$

```

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
    end
    F(I) = F(I) + fee(i) element's total force fee component i'th is computed → added to F(I)
  end
end
end
end

```

431 / 456