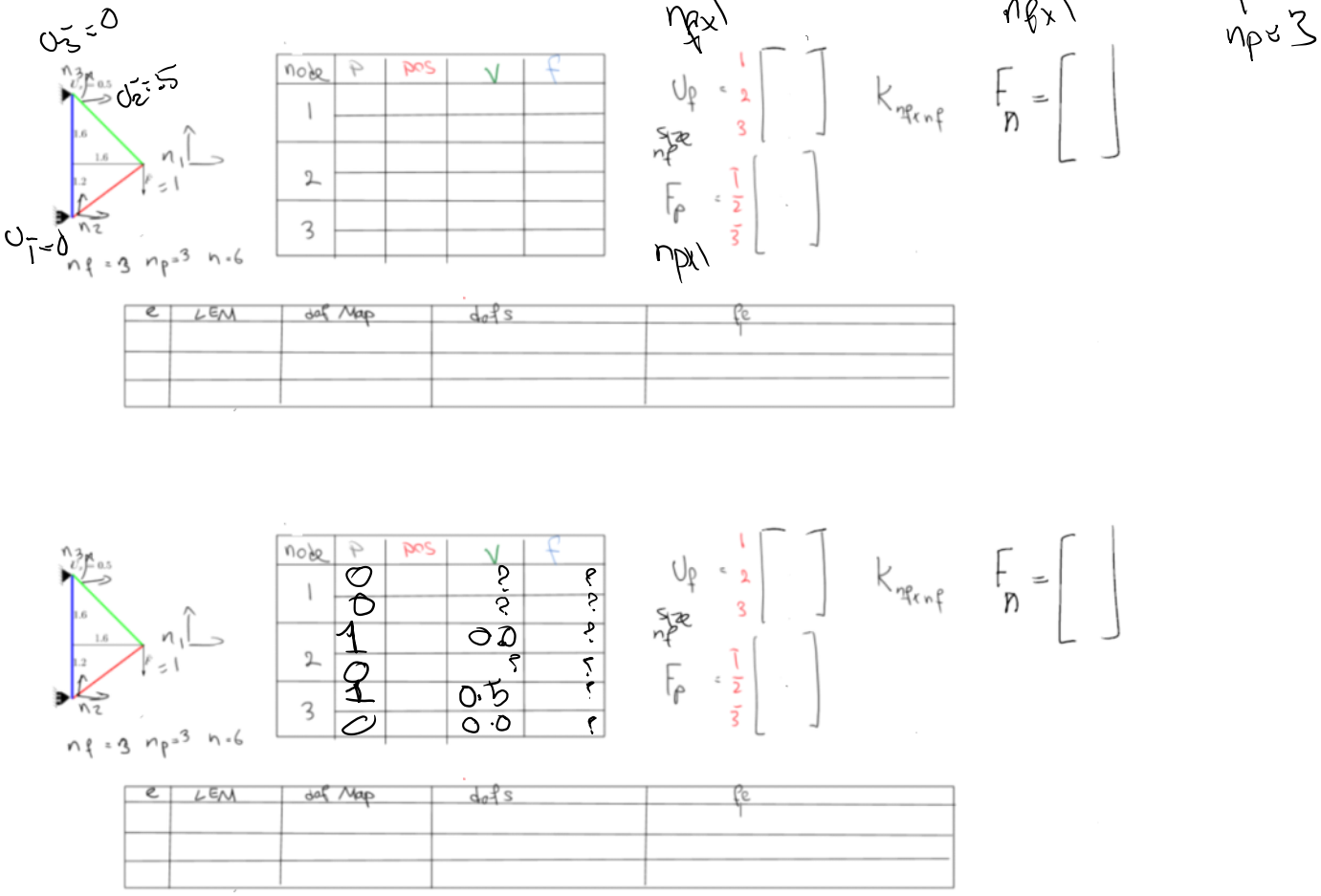
2022/11/10

Thursday, November 10, 2022 11:25 AM

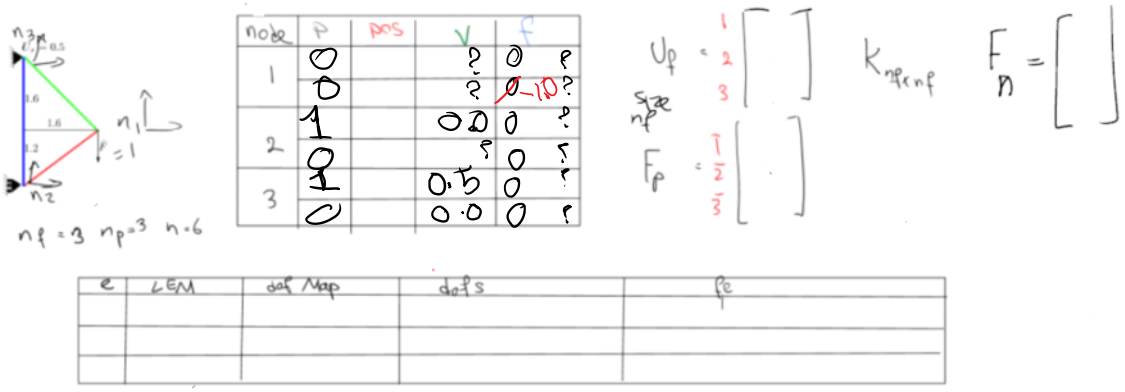
Step 4: Set global prescribed nodal dof



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

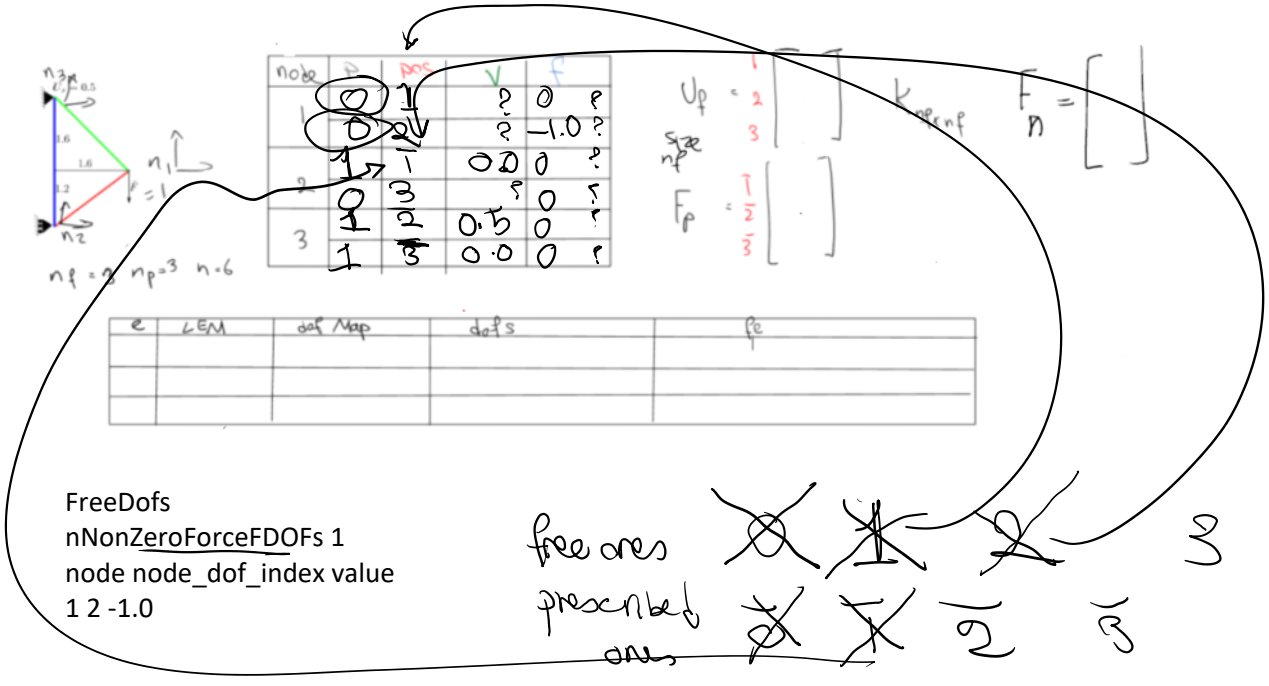
Step 5: Set global free nodal dof

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



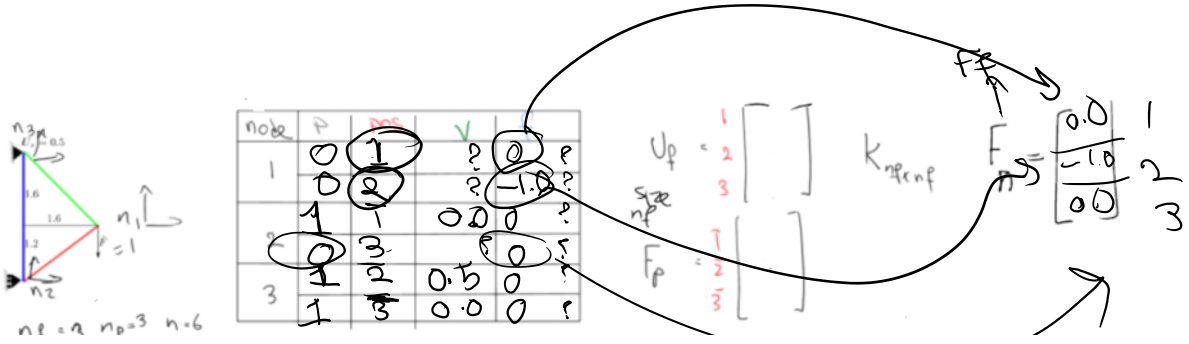
FreeDofs
nNonZeroForceFDOFs 1
node node_dof_index value
1 2 -1.0

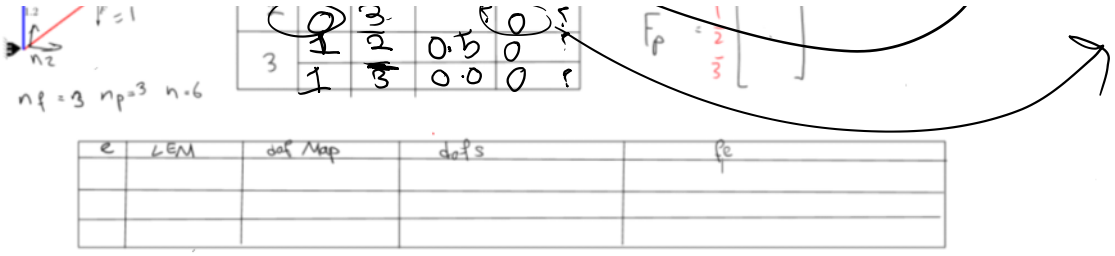
Step 6: dof positions; Step 7: Set $\mathbf{F}(\mathbf{F}_f)$



FreeDofs
nNonZeroForceFDOFs 1
node node_dof_index value
1 2 -1.0

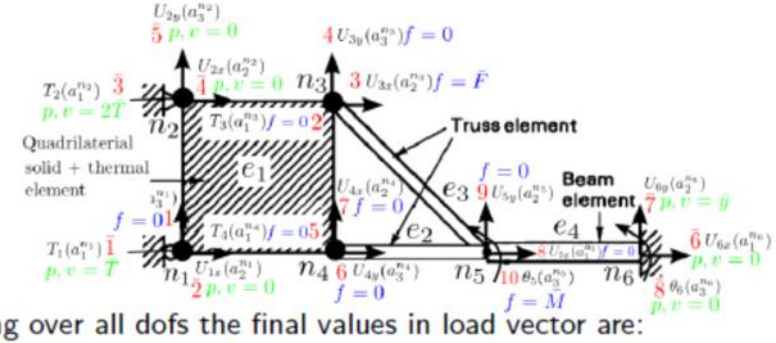
Step 7: Set Ff





e	LEM	dof Map	dofs	fe

Step 6: dof positions; Step 7: Set $\mathbf{F}(\mathbf{F}_f)$



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

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

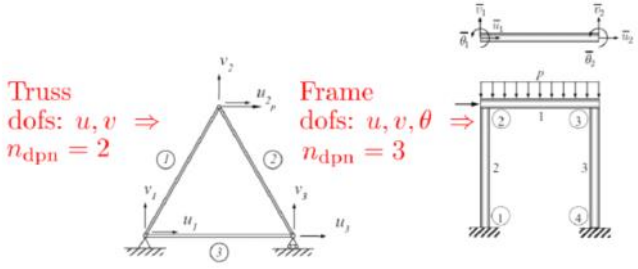
```
posf = 0, posp = 0
for n = 1:nNodes
  for dofi = 1: node(n).nndof num dof for node (n)
    if node(n).ndof(dofi).p == true prescribed dof
      posp = posp - 1;
      node(n).ndof(dofi).pos = posp;
    else free dof
      posf = posf + 1;
      node(n).ndof(dofi).pos = posf;
      F(posf) = node(n).ndof(dofi).f
    end
  end
end
```

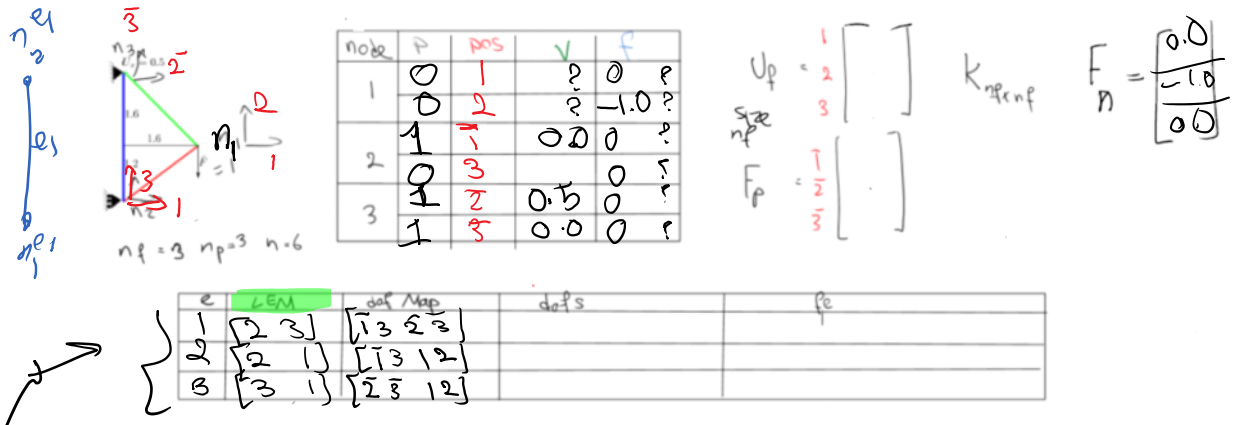
step 6

Step 6
step 7

Step 8: Element dof maps $\mathbf{M}_t^e$

Step 8: Element dof maps $\mathbf{M}_t^e$: Simplified limited case





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?

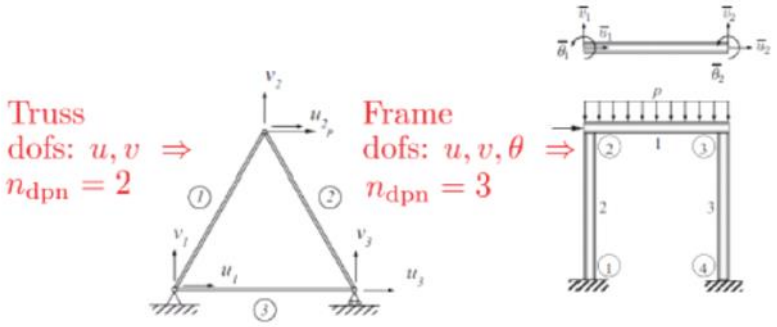
Elements
ne 3
id elementType matID neNodes eNodes
1 3 1 2 2 3
2 3 1 2 2 1
3 3 1 2 3 1

material id
element type
nodes for element

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

Slide 420:

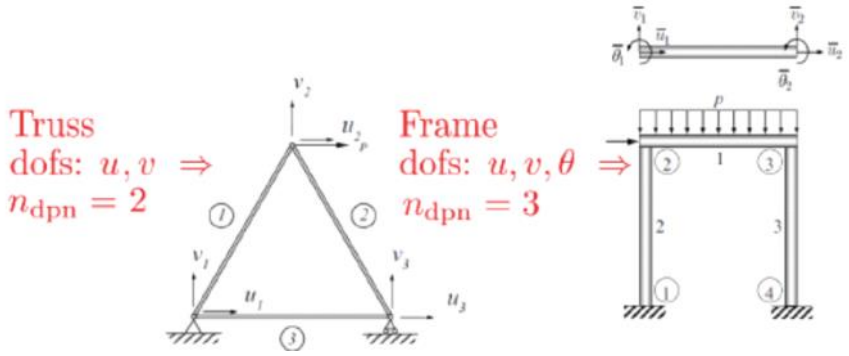
Step 8: Element dof maps $\mathbf{M}_t^e$: Simplified limited case

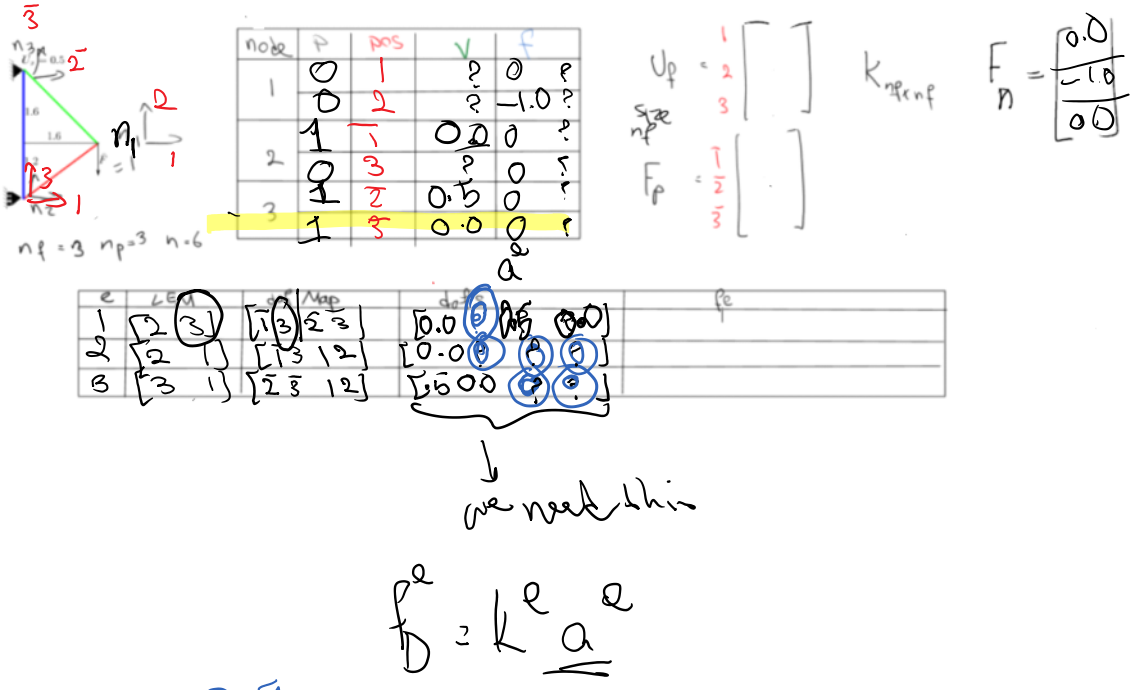


A simplified pseudo code looks like: ecdof = 1 dof counter for element
for en = 1: neNodes number of element nodes
  gn = LEM(en) global node number for element node en
  for endof = 1: ndofpn This number is fixed now, *e.g.*, 2 for 2D trusses
    dofMap(ecdof) = node(gn).dof(endof).pos
      gndof =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

```
    end
end
```

Step 9: Set element dofs $\mathbf{a}^e$: Simplified limited case



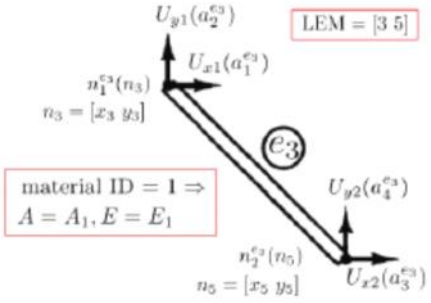


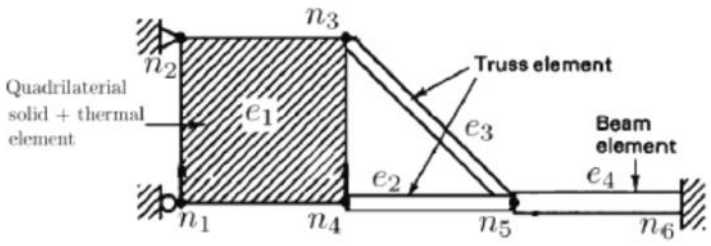
Steps 8&9

```
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)   gndof = 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
```

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 sti
ness/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 implementation

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 I = 0; I < 2; ++I)
        for (int J = 0; J < 2; ++J)
```

```
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;
}
```

$K = \frac{AE}{L}$

$$\begin{bmatrix} c^2 & cs & -c^2 & -cs \\ cs & s^2 & -cs & -s^2 \\ -c^2 & -cs & c^2 & cs \\ -cs & -s^2 & cs & s^2 \end{bmatrix}$$

Subroutine based

CalculateStiffness( )

{

if (eType == 1) // bar

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

}

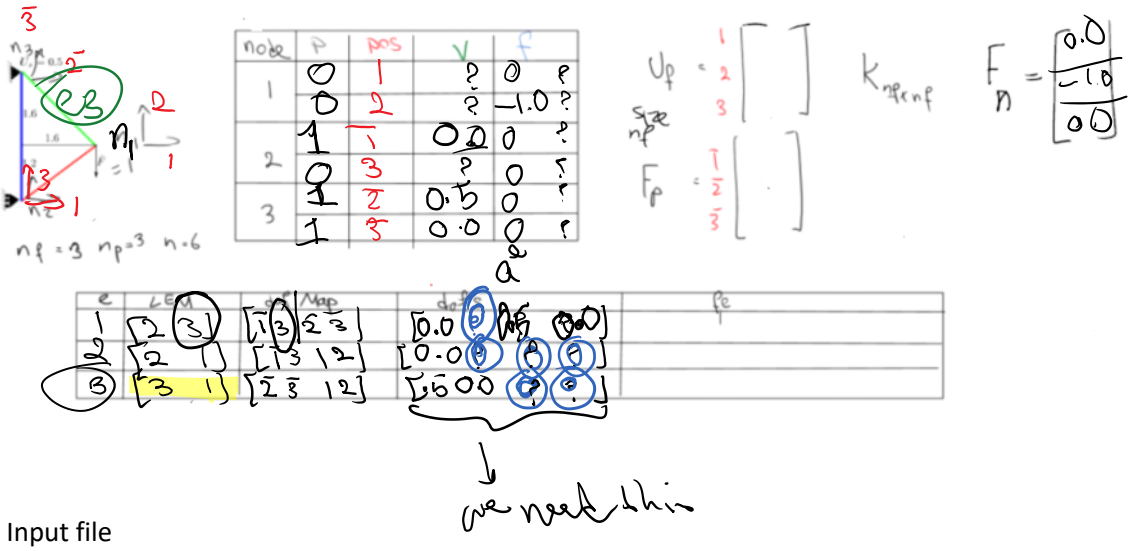
else if (eType == 2)

{

{ —

}

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



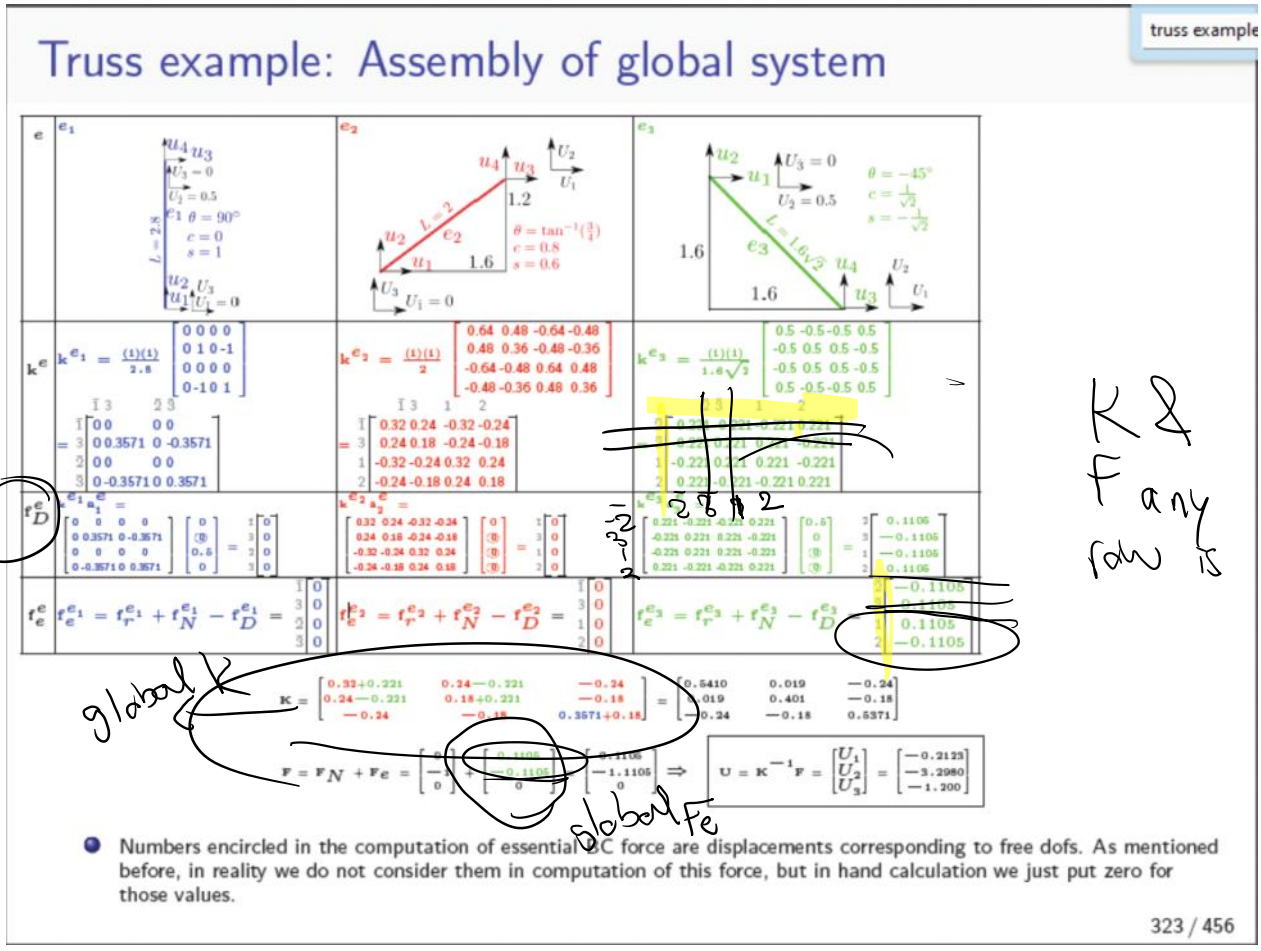
Input file
...
nNodes 3
id crd
1 1.6 1.2
2 0 0
3 0 2.8

$$\Delta_{e_3} = \text{coord(node1)} - \text{coord(node3)}$$
$$= [1.6, 1.2] - [0, 2.8]$$
$$= [1.6, -1.6]$$

$$L_{e_3} = \sqrt{(1.6)^2 + (1.6)^2}$$

$$\cos_{e_3} = \frac{(\Delta_{e_3})(1)}{L} = \frac{1.6}{2} = \frac{\sqrt{2}}{2}$$

Step 11: Assembly from local to global system



Summary:

Prescribed rows -> useless
=>

Only work with prescribed ROWS

Free column -> Stiffness is updated
else
Prescribed column -> $f_D = Ka$ is updated

```
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)
```

```
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
```