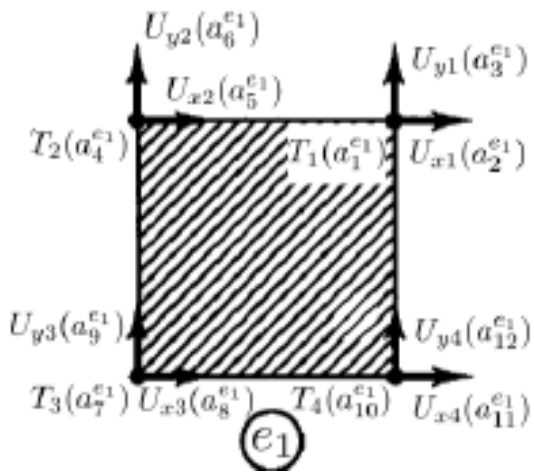
2017/11/02

Thursday, November 02, 2017 10:01 AM

Continuing from the last time (element members and functions)



solves these U_x, U_y, T

element vector of physics

1. Solid Mechanics

Fields U (displacement)

components U_x

U_y

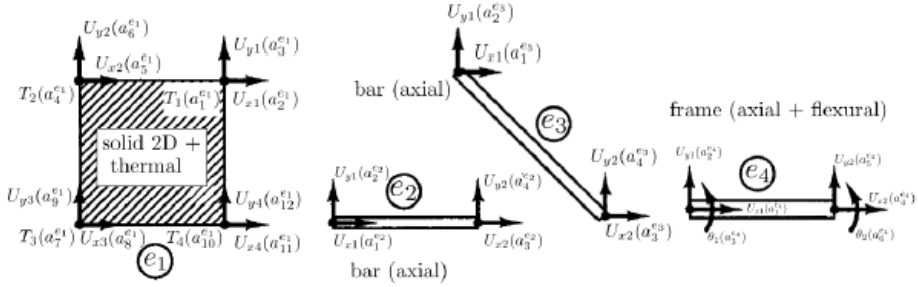
2. Thermal

T (temperature)

scalar T

The breakdown between what is called physics or field is a bit arbitrary. For example, I call e frame element a multiphysics element with bar + beam elements as its two separate physics.

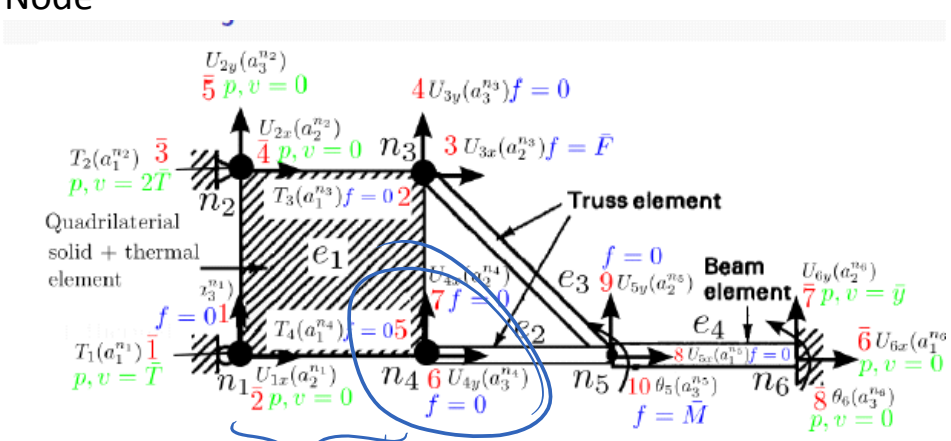
FEM Solver Objects: 2. Physics



- Physics refers to physics that an element represents. For example for e_1 the element includes two physics of plane solid and thermal.
- Elements e_2 and e_3 have one physics of solid truss.
- Depending on interpretation, e_4 can be considered as one physics of solid frame or two physics of truss + beam.
- Physics includes data that specify how physics is integrated into the element.
- Some of physics data are similar to element data in terms of dofs, dof maps, *etc.*.
- As an example remember frame element where the integration of two physics of axial (bar) and flexural(beam) into the element resembled assembly of local element to global system.
- Due to complexity and flexibility of description of physics object (*cf.* refer to the choice of interpretation above), as well as the fact that our focus will be on single physics problems we skip details of physics class.
- One important member of physics is {field}: that is a physics includes a set of tensor (or parts of tensor) fields. Examples are (U_x, U_y) for truss, and (U_x, U_y) and θ for frame.

Next main class

Node



5

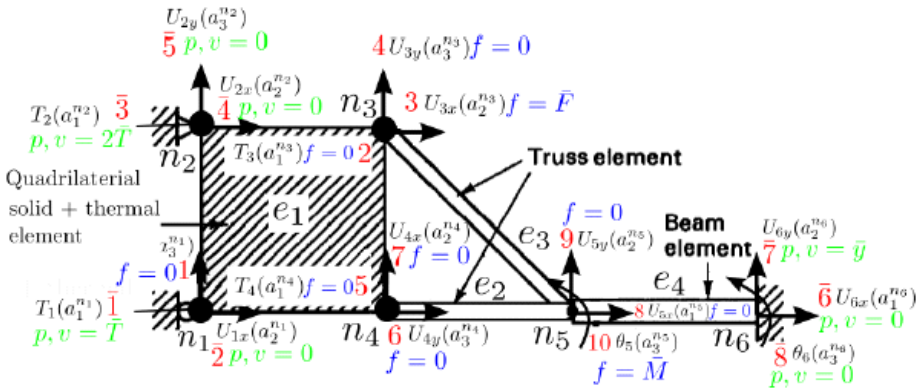
id = 4

Coord = (5,0)

Coord = (5,0)

set, vector, of dofs $\begin{pmatrix} \text{Temperature} \\ U_x \\ U_y \end{pmatrix}$

FEM Solver Objects: 4. Node: Data



- id: Clearly, id of n_1 is 1.
- coordinate: *e.g.*, for n_1 the coordinate in the figure can be:

$$\mathrm{crd}_{XY}(n_1) = [0 \quad 0]$$

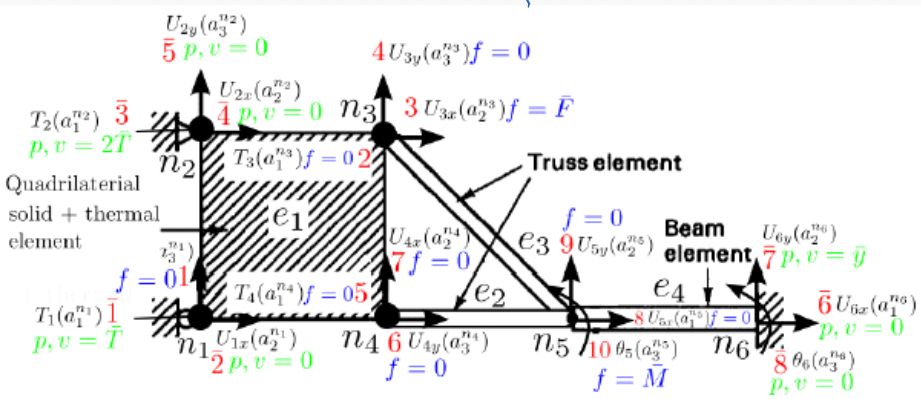
  coordinate components are alway represented with respect to a coordinate system (another geometry object we will not further discuss herein).
- {ndof}: *i.e.*, a "set" of dofs. Dof is a class being described next. It includes data such as being free or prescribed, position in global free or prescribed dofs, value (*e.g.*, displacement), and force.
- nndof: Number of dof for the given node.

```
class PhyNode
{
    ID id;
    VECTOR coordinate;
    vector <PhyDof> ndof;
    int nndof;      // number of dofs
    }
```

Next class
Dof

FEM Solver Objects: 5. Dof: Data

FEM Solver Objects: 5. Dof: Data



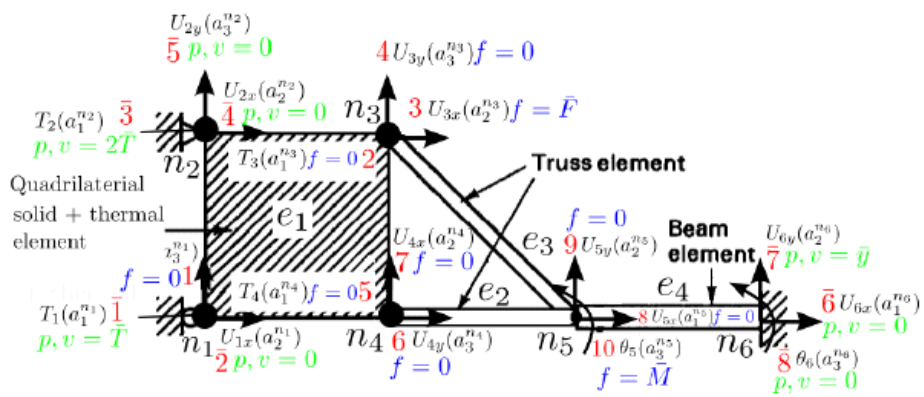
boolean	prescribed	true	false
number	value	$\bar{T}$	?
number	forces	?	0
Name	field	Temperature	U (displacement)
numbers	component	—	2 (U_y)
number	position	$\bar{T}$	1

```
class PhyDof
{
public:

    PhyDof();

    bool p;   // boolean: whether the dof is prescribed
    int pos;  // position in the global system (for free and prescribed)
    double v;      // value of dof
```

```
    double f;       // force corresponding to dof

//    F can be stress i can be (0, 1) sigma_{01}
//    Field       F;
//    INDEX i;
};
```

Examples of dof in the structure we are dealing with

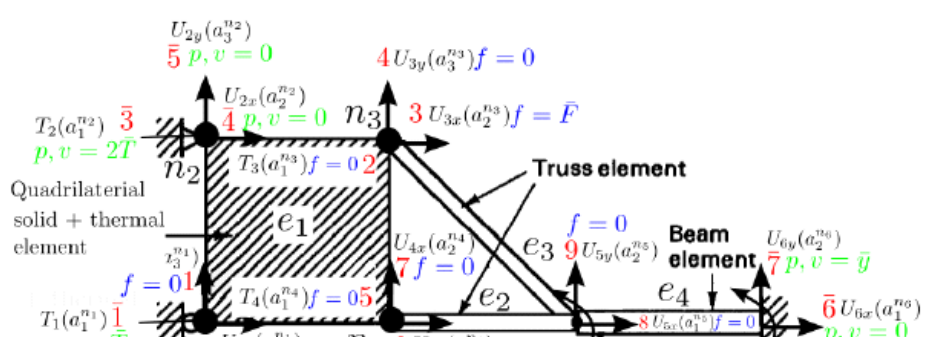
FEM Solver Objects: 5. Dof

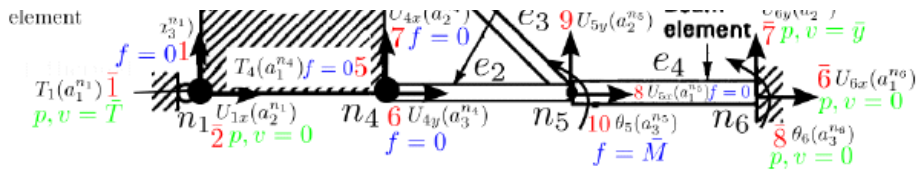


Examples of dof for the structure shown are:

dof	p	pos	v	f	field	index
1 of n_1	true	$\bar{1}$	$\bar{T}$	unknown	T	-
3 of n_1	false	1	unknown	0	U	2
3 of n_5	false	10	unknown	$\bar{M}$	θ	- (a vector in 3D)
2 of n_6	true	$\bar{7}$	$\bar{y}$	unknown	U	2

FEM Solver Objects: 6. Other solver related objects



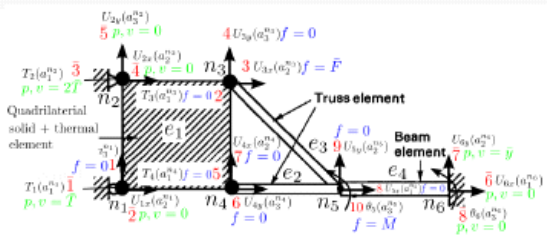


Some other objects in solver stage are:

- Material Property: This object stores material properties, such as elastic modulus E, conductivity κ. Some other properties such as A, and I may be included in this object or in a separate object.
- Objects for I/O: Several objects or temporary storage members are used for input (*e.g.*, p and f dofs) and output (nodal and element solutions).

You can have a class that stores everything needed for computations

FEM Solver Objects: 7. FEM solver



There is not a clear object for FEM solver. However, we can think of FEM solver as an object that is responsible for 1) reading in discretization data; 2) storing all node and element data; 3) solution of global system. Basically FEM solver is the driver for FEM solution.

- dim, n_{dim} spatial dimension for the problem (1D, 2D, and 3D)
- number of nodes (nNodes, n_{n}) in the domain; *e.g.*, $n_{\text{n}} = 6$.
- nodes {node}: vector of nodes in the domain; *e.g.*, $n_1, n_2, n_3, n_4, n_5, n_6$.
- number of elements (ne, n_{e}) in the the domain; *e.g.*, $n_{\text{e}} = 4$.
- elements {element}: vector of elements in the domain; *e.g.*, e_1, e_2, e_3, e_4.
- free dofs (dofs: $\mathbf{a}$): vector of global free dofs.
- number of free dof (nf, n_{f}); *e.g.*, $n_{\text{f}} = 10$.
- number of prescribed dof (np, n_{p}); *e.g.*, $n_{\text{p}} = 8$.
- number of dof (ndof, $n_{\text{dof}} = n_{\text{f}} + n_{\text{p}}$); *e.g.*, $n_{\text{dof}} = 18$.
- stiffness matrix ($\mathbf{K} = \mathbf{K}_{ff}$); $n_{\text{f}} \times n_{\text{f}}$ matrix.
- force vector ($\mathbf{F} = \mathbf{F}_f$); $n_{\text{f}}(\times 1)$ vector.
- prescribed force vector ($\mathbf{F}_p$) (Optional); $n_{\text{p}}(\times 1)$ vector. May be used for more streamlined computation of prescribed dof forces.
- number of materials: nmats.
- material database {mats}: Parameters and Values for all material in the model.

401 / 456

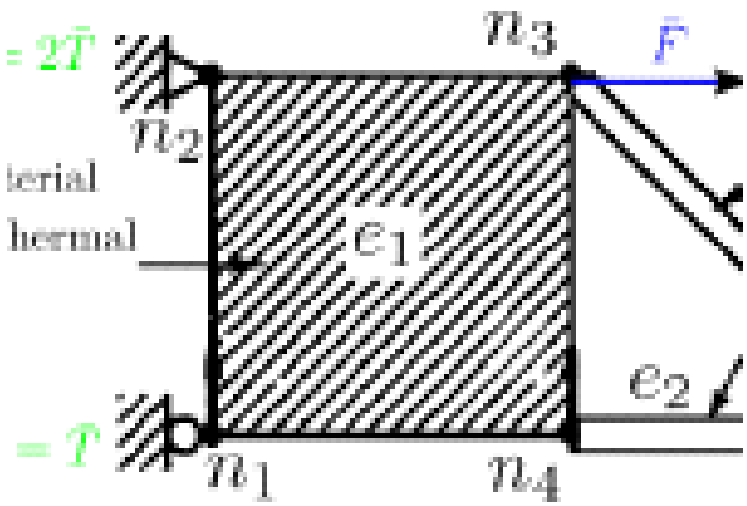
Ka = F

class FEMSolver

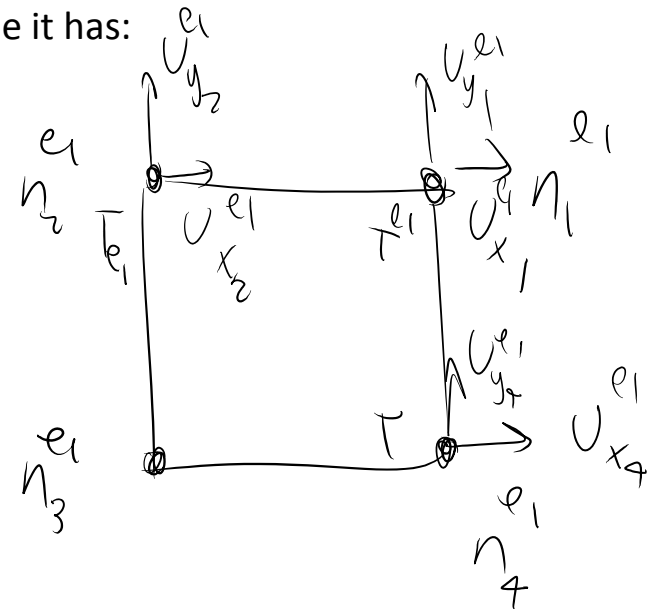
In the sample code sent to you.

Solution steps:
Start from the input from say a text file to writing your results to a terminal

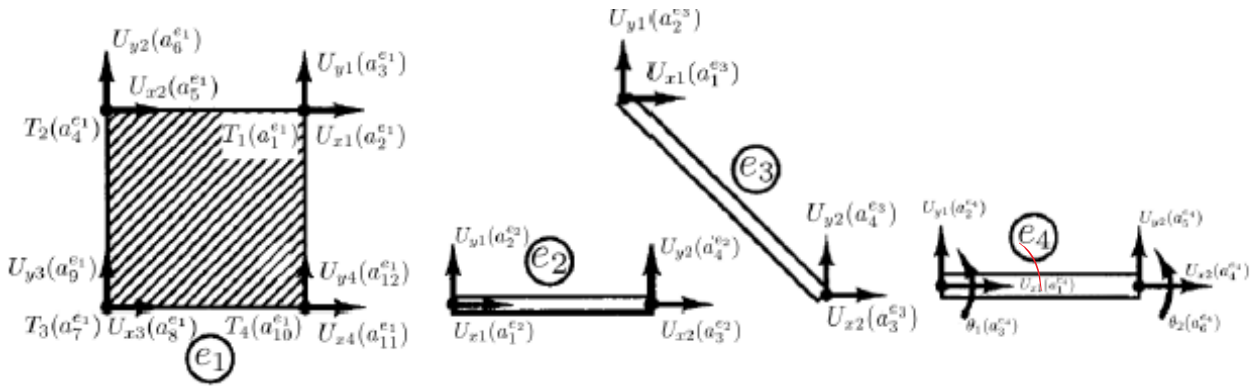
Step 1: Set element nodal dofs



By telling the element that it solves solid (with Ux and Uy components) and thermal (with temperature) it knows that at each node it has:



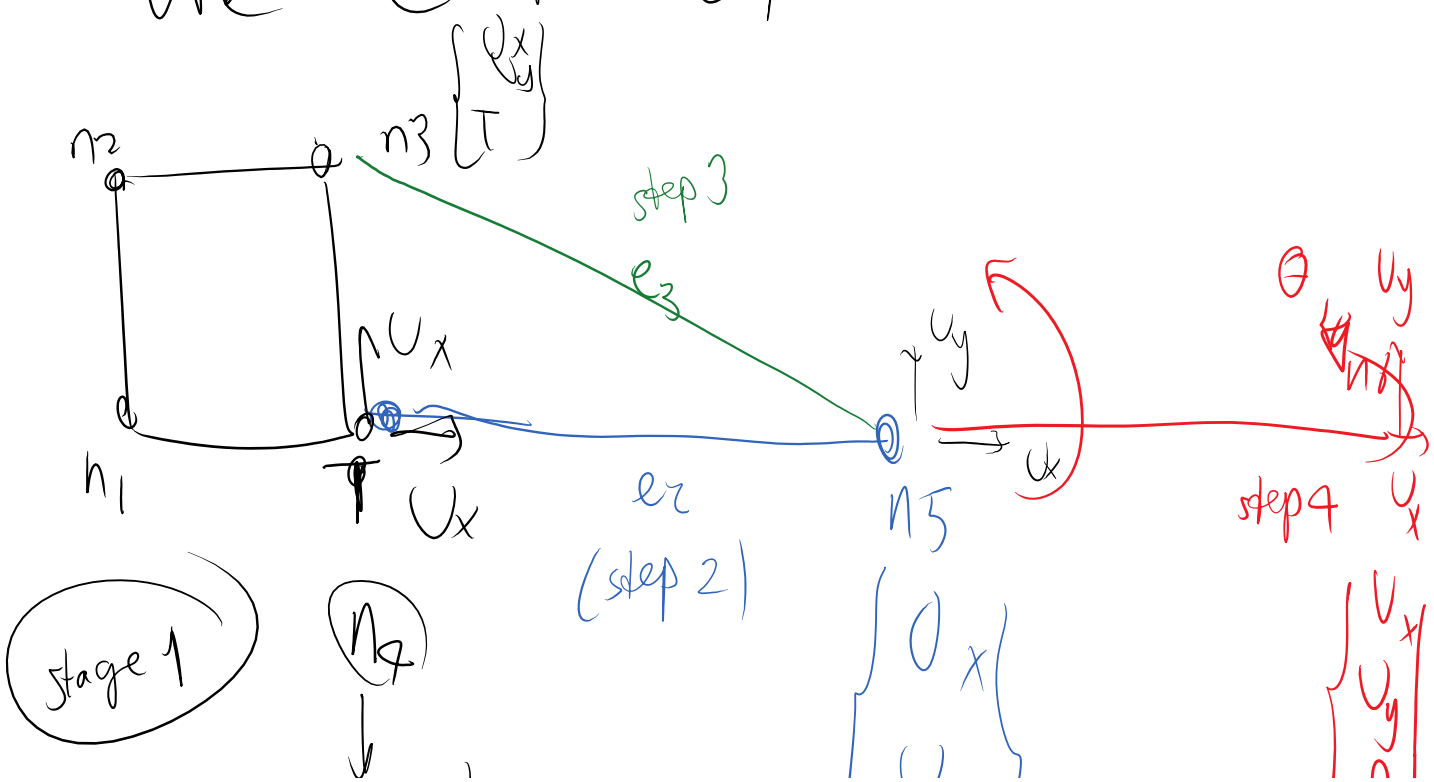
We do the same process for all the elements:

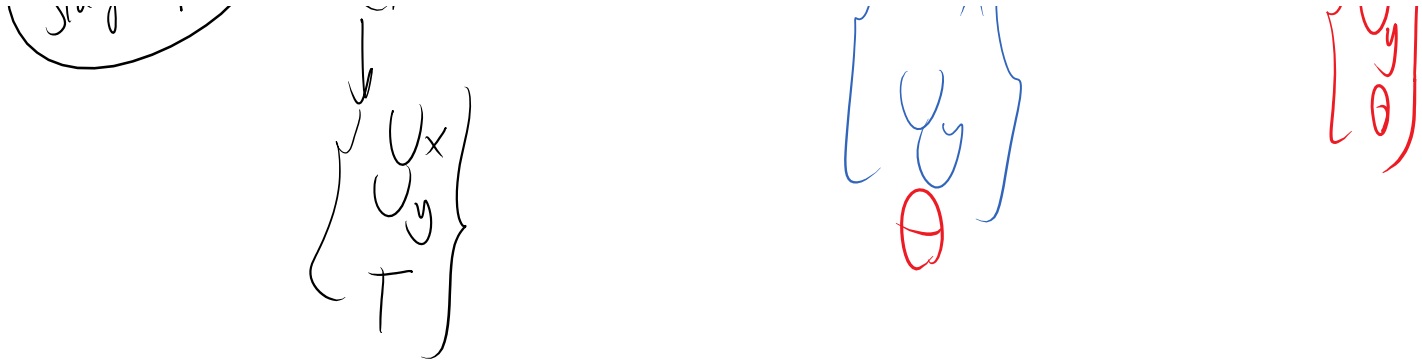


Step 2:
We want to know what dofs EACH NODE in the global system has

we cannot loop over nodes & see what elements are attached to them!

We CAN loop over elements...



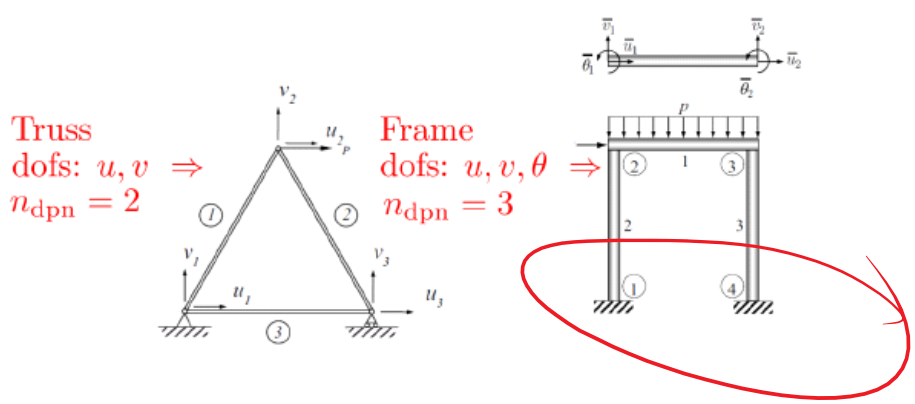


Design a general FEM program is difficult

A simplifying assumption is that we solve a structure with same type of element.

ALL the NODEs that exactly the same set of DOFs.

Steps 1 & 2: Simplified limited case



- FEM implementation become considerably simpler for problems where all elements are of the same type (regardless of number of physics per element).
- In this case, we define:

$$\boxed{n_{\text{dpn}} := \text{Number of dof. per node denoted by } \underline{\text{ndofpn}}} \tag{448}$$

Step 3: Set global number of dofs, stiffness, and force.

How many total, free, prescribed dofs we have.
Size Global F's and K's

$n = ndofpn * nnodes$

Prescribed dofs are given in the input text file,
Example

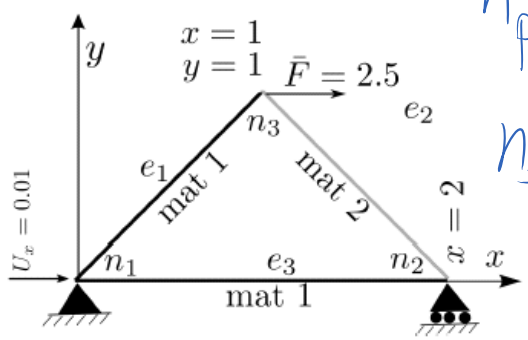
Input file format

```
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
- ---
```

$n = 3 \times 2 = 6$

$n_p = 3$

$n_f = n - n_p = 3$



K.resize(n_f, n_f)

F. " (n_f)

a. " (n_f)

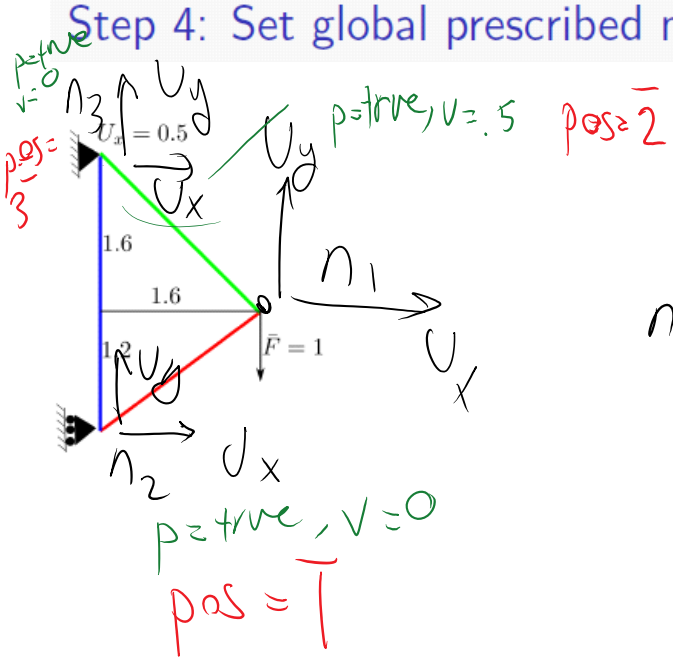
& set them to zero.

F_p. resize(n_p)

tp . resize (np)

prescribed forces

Step 4: Set global prescribed nodal dof



np 3

nodeID	dofID	value
2	1	0
3	1	.5
3	2	0

node

dof

At these stage we loop over the prescribed nodes

In the loop

Get right node

Access the dof given

class FEMSolver
vector<PhyNode> nodes;

class PhyNode
vector <PhyDof> ndof;

class PhyDof

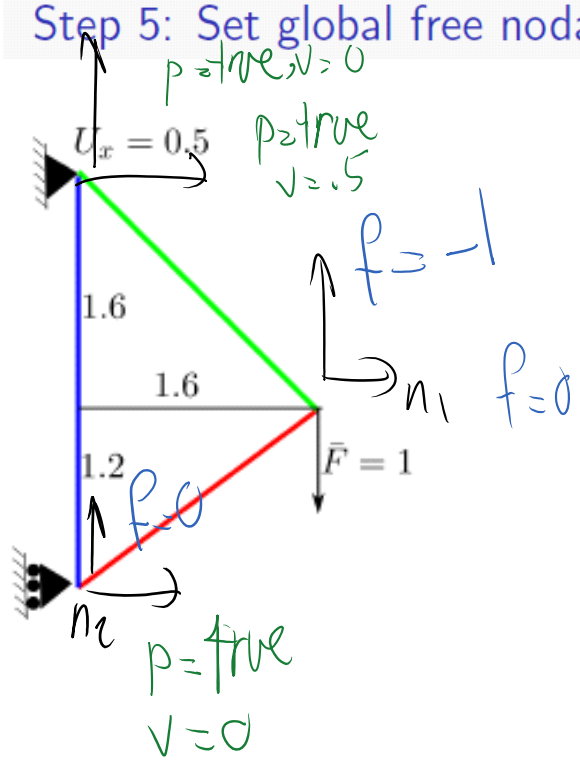
```
{
public:
    PhyDof();
    bool p;  // boolean: whether the dof is
    prescribed
    int pos;  // position in the global system (for
    free and prescribed)
    double v;      // value of dof
    double f;      // force corresponding to dof
```

by default is false

turned to true

set

Step 5: Set global free nodal dof

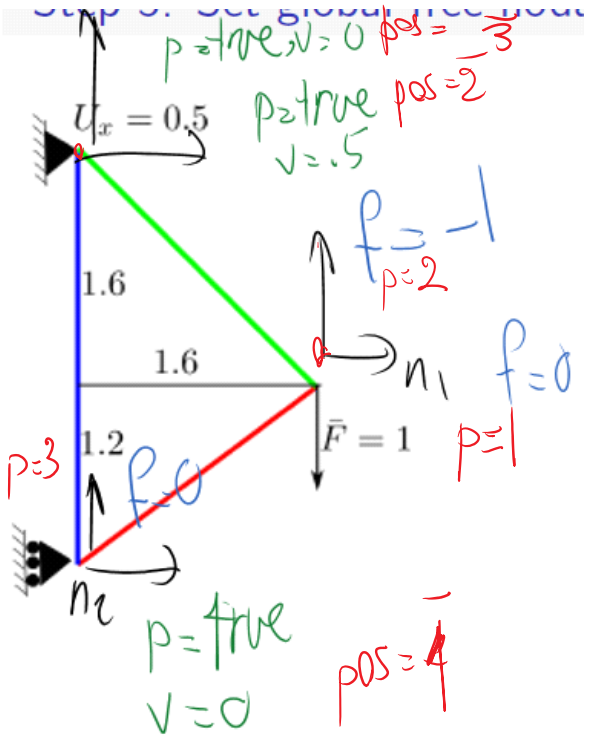


Only list free dof's with a force

n free w/ force

node	dof	force
1	2	-1

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



$$F_n = \begin{bmatrix} 0 \\ -1 \\ 0 \end{bmatrix} \begin{matrix} \text{pos } 1 \\ \text{pos } 2 \\ \text{pos } 3 \end{matrix}$$

nf

Below is a loop that does 3 things:

1,2 Number positions of free and prescribed dofs

3. setting F (global force)

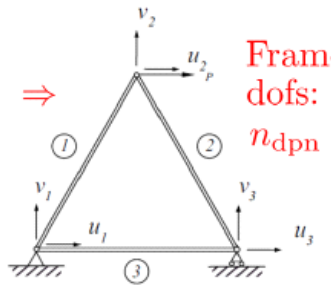
```
posf = 0, posp = 0
for n = 1:nNodes
   for dofi = 1: node(n).nndof num dof for node (n)
      if node(n).ndof(dofi).p == true prescribed dof
         posp = posp - 1;
         node(n).ndof(dofi).pos = posp;
      else free dof
         posf = posf + 1;
         node(n).ndof(dofi).pos = posf;
   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

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

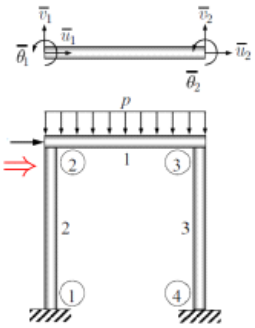
Very difficult in general

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

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



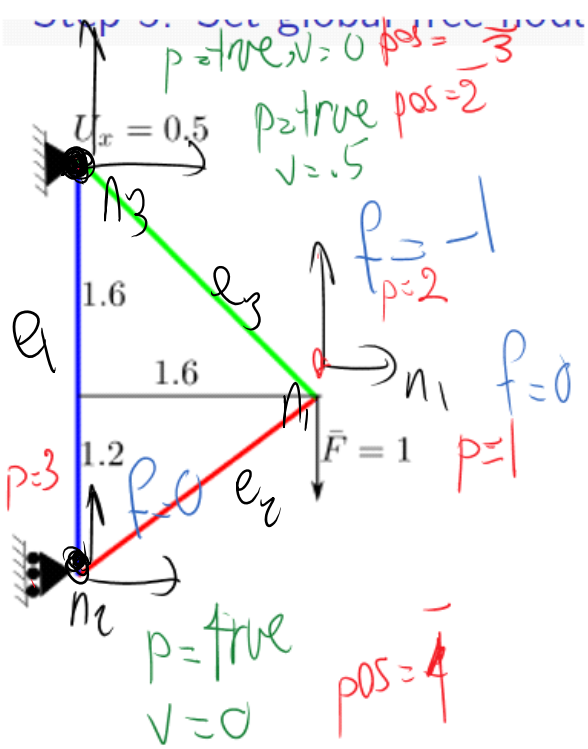
Frame
dofs: u, v, θ $\Rightarrow$
$n_{\text{dpn}} = 3$



- in Steps 1 and 2 we mentioned that FEM implementation becomes considerably simpler if we assume all nodes share exactly the same set of dofs.
- Examples are bars, beams, trusses, and frames.
- In (448) we defined n_{dpn} (ndofpn) as,

$$n_{\text{dpn}} := \text{Number of dof. per node}$$

- In this limited scenario ith dof of node in element is mapped to ith dof of its corresponding global node.



$(LEM)_{e_1} = [2, 3]$

$(LEM)_{e_2} = [2, 1]$

$(LEM)_{e_3} = [3, 1]$

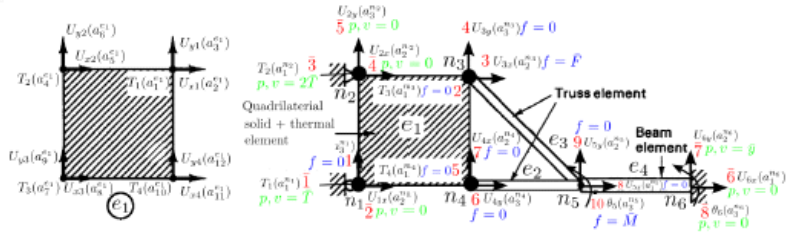
we provide this in our input files

$(dof)^{e_1}_{map} = [1\ 3, 2\ 3]$

$q^e = [0\ 0\ 0.5\ 0]$

dofMap for elements

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



- To calculate Essential BC force for element $\mathbf{f}_D^e = \mathbf{k}^e\mathbf{a}^e$, we need to set $\mathbf{a}^e$.
- This step as many other cases can be combined with other steps in one loop over elements, but for clarity it is described separately.
- We can merge this step with Step 8: Element dof maps $\mathbf{M}_t^e$ by adding lines in red:

```
edofs = zeros(nedof) element dofs (edof) resized to number of element dofs and zerod
ecdof = 1 dof counter for element
for en = 1: neNodes # element nodes
  gn = LEM(en) global node number for element node en, e.g., gn = 5 for en = 1 in e2
  for endof = 1:nndof(en) number of dofs of element node en e.g., nndof(en) = 2 for en
= 1 in e2
    gndof = e2globalDofMap(en, endof)
      gndof, corresponding dof in gn, e.g., gndof = 2 for endof = 1, en = 1 in e2
    if (node(gn).dof(gndof).p == true)
      dofs(ecdof) = node(gn).dof(gndof).value; e dof val = corresponding global val
    end
    dofMap(ecdof) = node(gn).dof(gndof).pos
      pos of dof #gndof of global node gn, e.g.,6 for endof = 1, en = 1 in e2
    ecdof = ecdof + 1    increment counter
  end
```

set element dofMap(M^e) dofs(a^e)