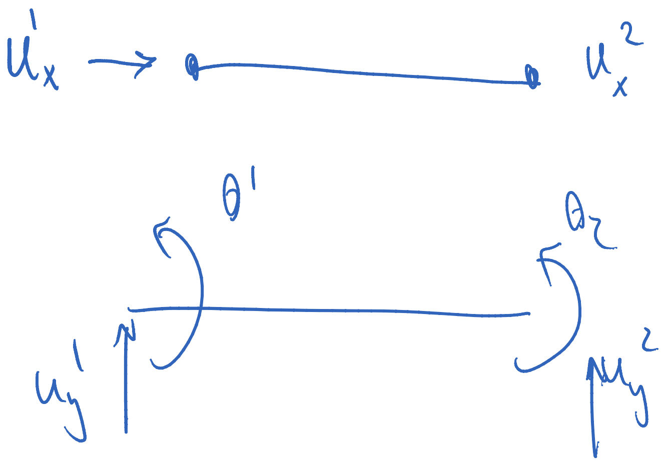
2016/11/08

Tuesday, November 08, 2016
10:10 AM

Frames: 2D frame elements

Frame is an element that has both axial and bending modes active

fram element

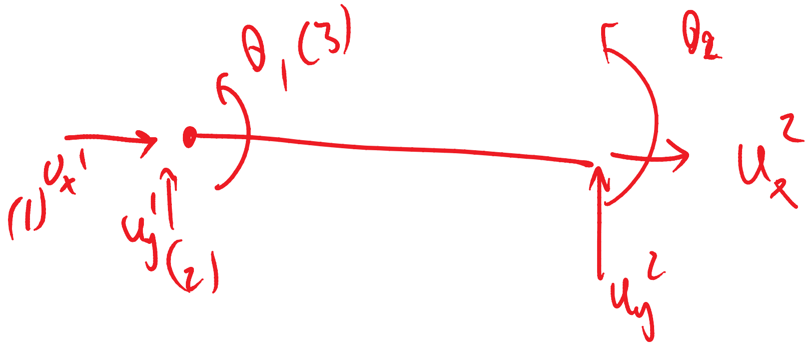


bar element

+

beam element

Frame element



6 dofs

stiffness for axial part (bar element)

$$\begin{bmatrix} f_x^1 \\ f^2 \end{bmatrix} = \frac{AE}{L} \begin{bmatrix} 1 & -1 \\ & \end{bmatrix} \begin{bmatrix} u_x^1 \\ u^2 \end{bmatrix}$$

$$\begin{bmatrix} \cdots \\ f_x^2 \end{bmatrix} = \frac{AE}{L} \begin{bmatrix} \cdots \\ -1 & 1 \end{bmatrix} \begin{bmatrix} \cdots \\ u_x^2 \end{bmatrix}$$

bending mods come from a beam element:

$k_b^e = \dfrac{EI}{L^{e3}} \begin{bmatrix} 12 & 6L^e & -12 & 6L^e \\ & 4L^{e2} & -6L^e & 2L^{e2} \\ & & 12 & -6L^e \\ \text{sym.} & & & 4L^{e2} \end{bmatrix}$	for constant E and I

$$\begin{bmatrix} f_y^1 \\ M_1 \\ f_y^2 \\ M_2 \end{bmatrix} = k_b^e \begin{bmatrix} u_y^1 \\ \theta^1 \\ u_y^2 \\ \theta^2 \end{bmatrix}$$

By Combine the stiffness matrices we obtain:

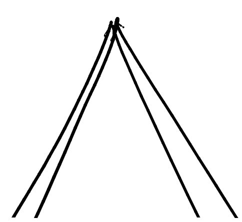
step 1

- Element matrix equation (local coord.)

$$\begin{bmatrix} a_1 & 0 & 0 & -a_1 & 0 & 0 \\ 0 & 12a_2 & 6La_2 & 0 & -12a_2 & 6La_2 \\ 0 & 6La_2 & 4L^2a_2 & 0 & -6La_2 & 2L^2a_2 \\ -a_1 & 0 & 0 & a_1 & 0 & 0 \\ 0 & -12a_2 & -6La_2 & 0 & 12a_2 & -6La_2 \\ 0 & 6La_2 & 2L^2a_2 & 0 & -6La_2 & 4L^2a_2 \end{bmatrix} \begin{Bmatrix} \bar{u}_1 \\ \bar{v}_1 \\ \bar{\theta}_1 \\ \bar{u}_2 \\ \bar{v}_2 \\ \bar{\theta}_2 \end{Bmatrix} = \begin{Bmatrix} \bar{f}_{x1} \\ \bar{f}_{y1} \\ \bar{c}_1 \\ \bar{f}_{x2} \\ \bar{f}_{y2} \\ \bar{c}_2 \end{Bmatrix} \qquad a_1 = \frac{EA}{L} \qquad a_2 = \frac{EI}{L^3}$$

In local coordinate system

Frames like this

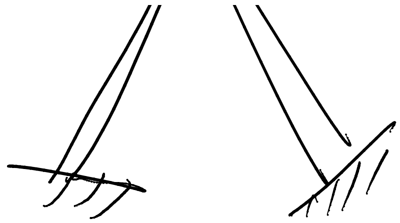


flanges like

we need to transfer

k^e, f^e from

local coordinate to global coordinate

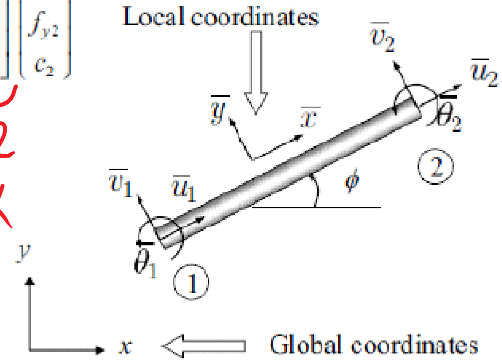


$$\begin{Bmatrix} \bar{f}_{x1} \\ \bar{f}_{y1} \\ \bar{c}_1 \\ \bar{f}_{x2} \\ \bar{f}_{y2} \\ \bar{c}_2 \end{Bmatrix} = \underbrace{\begin{bmatrix} \cos\phi & \sin\phi & 0 & 0 & 0 & 0 \\ -\sin\phi & \cos\phi & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & \cos\phi & \sin\phi & 0 \\ 0 & 0 & 0 & -\sin\phi & \cos\phi & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}} \begin{Bmatrix} f_{x1} \\ f_{y1} \\ c_1 \\ f_{x2} \\ f_{y2} \\ c_2 \end{Bmatrix}$$

$$\boxed{\{\bar{\mathbf{f}}\} = [\mathbf{T}]\{\mathbf{f}\}}$$

$$\boxed{\{\bar{\mathbf{q}}\} = [\mathbf{T}]\{\mathbf{q}\}}$$

step 2
K



Step 3 K in global coordinate system

- Element matrix equation (global coord.)

$$[\bar{\mathbf{k}}][\mathbf{T}]\{\mathbf{q}\} = [\mathbf{T}]\{\mathbf{f}\} \Longrightarrow [\mathbf{T}]^T[\bar{\mathbf{k}}][\mathbf{T}]\{\mathbf{q}\} = \{\mathbf{f}\} \Longrightarrow [\mathbf{k}]\{\mathbf{q}\} = \{\mathbf{f}\}$$

$$\boxed{[\mathbf{k}] = [\mathbf{T}]^T[\bar{\mathbf{k}}][\mathbf{T}]}$$

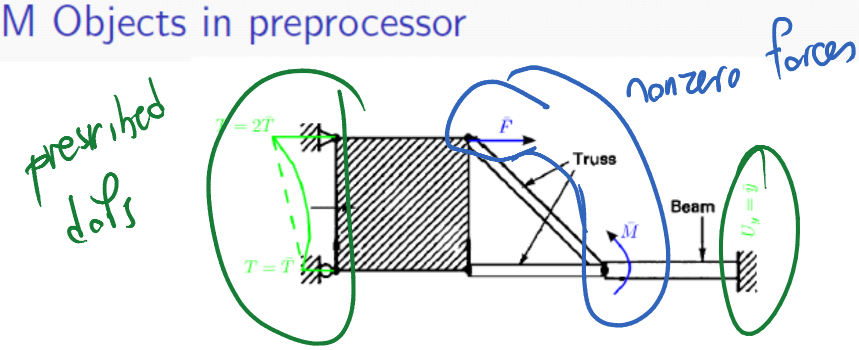
global coord. sys. K

transfer matrix

local coordinate system stiffness

NEW SECTION: FEM CODING

FEM Objects in preprocessor

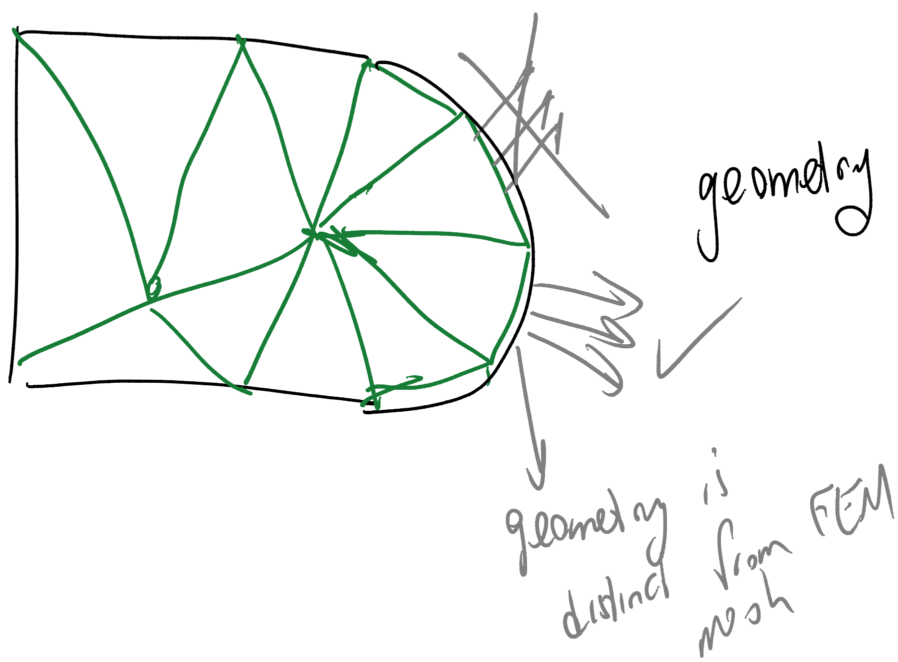


This is a problem description

Inputs are:

- Geometry
- Which dofs are prescribed
- Which dofs have nonzero foces

In general geometry is a separate entity from the mesh

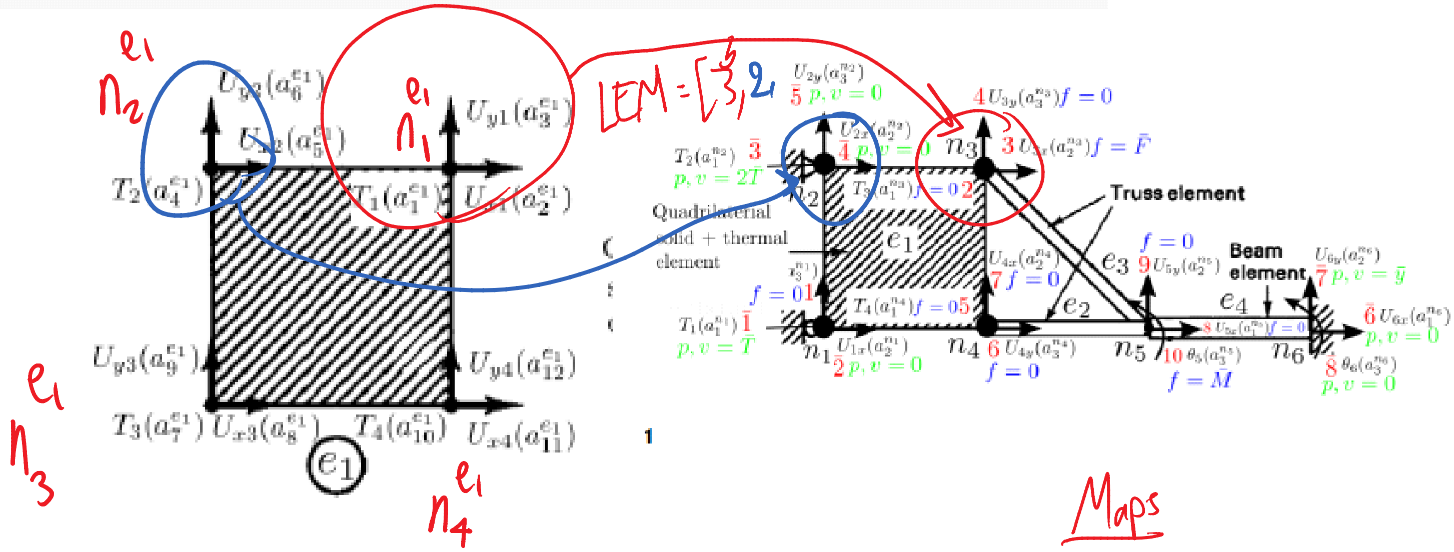


All loads, BC, material properties →

entered for geometry NOT FEM objects (nodes, elements)

FEM Objects in solver

FEM Solver Objects: 1. Element: Data



- id: Clearly, id of e_1 is 1.
- neNodes (n_n^e): Number of element nodes (*e.g.*, for element 1 $n_n^e = 4$).
- eNodes (LEM): Indices of element nodes in global system; *e.g.*, for e_1:

$$\mathrm{LEM}^{e_1} = [3 \quad 2 \quad 1 \quad 4]$$

- number of element dof (nedof: n_f^e): can be different than n_n^e; for $e_1 : n_f^e = 12$. $(\overbrace{T + U_x + U_y}^{3}) \times 4 = 12$
- edofs (a^e): n_f^e vector of dofs; *e.g.*, for e_1:

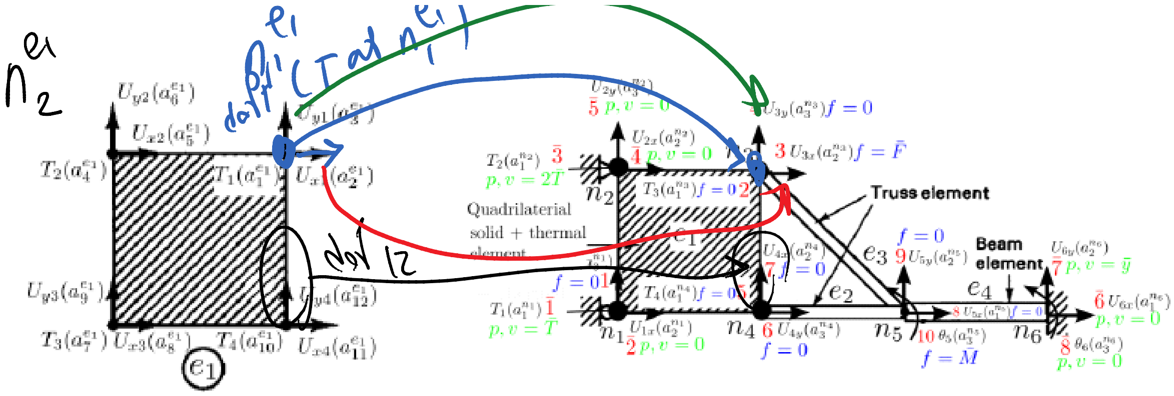
$$a_1^e = \begin{bmatrix} a_1^{e_1} & a_2^{e_1} & \cdots & a_{n_f^e}^{e_1} \end{bmatrix}$$
$$= [\, T_1 \quad U_{x1} \quad U_{y1} \quad T_2 \quad U_{x2} \quad U_{y2} \quad T_3 \quad U_{x3} \quad U_{y3} \quad T_4 \quad U_{x4} \quad U_{y4} \,]$$

- $\boxed{\text{dofMap}}$ (M_t^e): map from element to global dofs; *e.g.*, for e_i we will observe:

$$M^{e_1}{}_t = [\, 2 \quad 3 \quad 4 \quad \bar{3} \quad \bar{4} \quad \bar{5} \quad \bar{1} \quad \bar{2} \quad 1 \quad 5 \quad 6 \quad 7 \,]$$

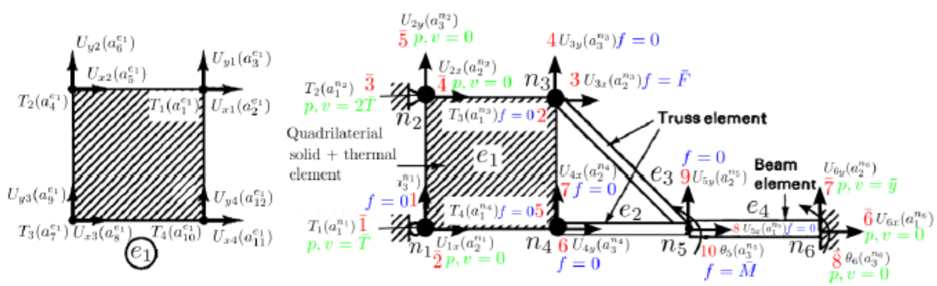
392 / 45

Map between local & global system



$$M^{e_1} = [2\ \ 3\ \ 4\ \ \bar{3}\ \ \bar{4}\ \ \bar{5}\ \ \ldots\ \ 7]$$

FEM Solver Objects: 1. Element: Data



- stiffness matrix (ke: $\mathbf{k}^e$): $n_f^e \times n_f^e$ local stiffness (conductivity, *etc.*) matrix.
- force vectors (fde, foe, fee: $\mathbf{f}_D^e, \mathbf{f}_o^e, \mathbf{f}_e^e$): A variety of element n_f^e load vectors such as essential BC ($\mathbf{f}_D^e$), sum of other forces ($\mathbf{f}_o^e = \mathbf{f}_r^e + \mathbf{f}_N^e + \cdots$, *etc.*), and element total $\mathbf{f}_e^e = \mathbf{f}_D^e + \mathbf{f}_o^e$.
- {physics}: Physics represented by the element; *e.g.*, for e_1 physics are solid and thermal.
- eType: one or more than one object that identify the element type. Examples are shape and order of element (linear triangle, *etc.*).

In C++

```
class PhyElement
{
....
int id;
int neNodes; // # element nodes
vector<int> eNodes; // element node vector
vector <PhyNode*> eNodePtrs;
int nedof; // # element dof
VECTOR edofs; // element dofs
vector<int> dofMap;
ElementType eType;
int matID;
MATRIX ke; // element stiffness matrix
```

VECTOR foe; // element force vector from all sources other than essential BC
VECTOR fde; // element essential BC force
VECTOR fee; // all element forces
}

Second important part of a class is the set of functions

FEM Solver Objects: 1. Element: Function

Some sample element functions are:

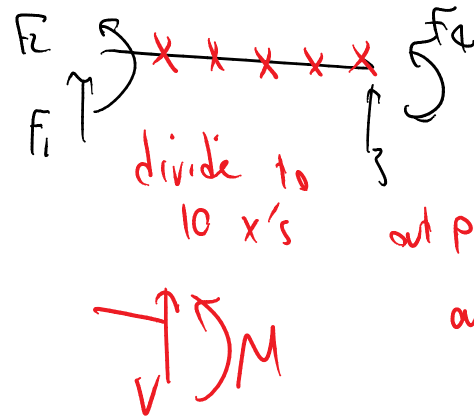
- Calculate $\mathbf{k}^e$ (virtual)
- Calculate $\mathbf{f}^e_o$: sum of all forces, but $\mathbf{f}^e_D$ (virtual)
- Compute_Output_Element: computes and outputs element; *e.g.*, axial forces for truss: (virtual)
- Calculate $\mathbf{f}^e_D = \mathbf{k}^e\mathbf{a}^e$
- Assemble $\mathbf{k}^e$ and $\mathbf{f}^e_e$ into global $\mathbf{K}$ and $\mathbf{F}$

k^e

$f^e_o = f^e_r + f^e_N + \ldots$

(F_1, F_2)

Non virtual functions
Same implementation for all element types



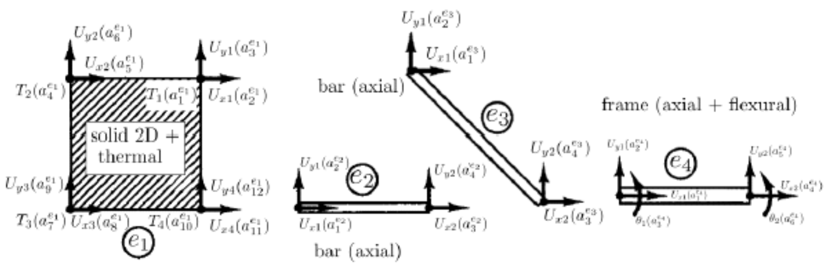
Virtual

Virtual is an attribute of some functions in object oriented programming. Without going to details, virtual functions are black box functions where any different type of object (*e.g.*, solid physics element, thermal physics element, *etc.*) performs its independent routine to achieve the objective of the function (*e.g.*, compute $\mathbf{k}^e$ and $\mathbf{f}^e_o$ i the first two functions; compute area and surface for shapes, *etc.*).
This is opposed to the last two functions in this example (assembly, and $\mathbf{f}^e_D = \mathbf{k}^e\mathbf{a}^e$) where the same exact routine is performed for all types of objects.

C++
Examples of virtual and nonvirtual functions

```
Class PhyElement
{
…
// Step 10: Compute element sti
ness/force (ke, foe (fre: source term; fNe: Neumann BC))
virtual void Calculate_ElementStiffness_Force() = 0;


// nonvirtual examples
// Step 11: Assembly from local to global system
void AssembleStiffnessForce(MATRIX& globalK, VECTOR& globalF);
```

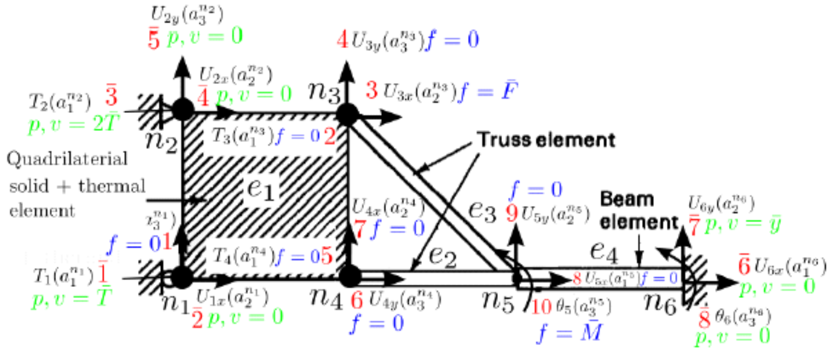
FEM Solver Objects: 3. Field



- Field is one complete or partial tensor field represented by the element
- For example, if we consider a 2D frame as one physics (as opposed to two physics of axial bar + flexural bending) this physics has the two fields: 1. Displacement: (U_x, U_y) and 2. rotation: (θ).
- In this example, (U_x, U_y) is partial since it misses U_z and $\theta := \theta_z$ is only one of the three rotations.
- Data: One important data of field is component. U_x is a component of the field, while the set (U_x, U_y) is the actual field.
- In summary:

Element data: {Physics}
Physics data: {Field}
Field data: {Component}

FEM Solver Objects: 4. Node: Data



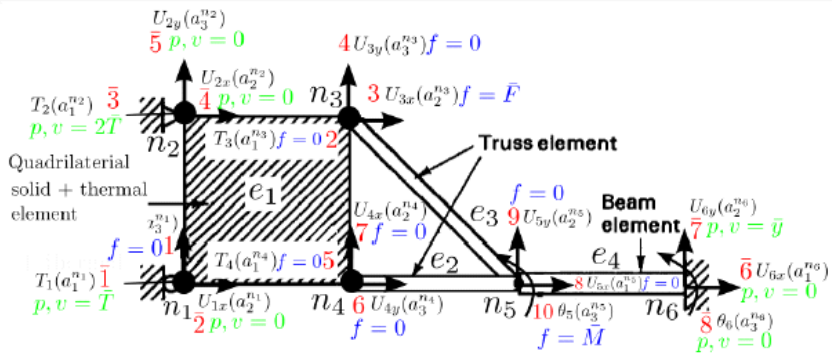
- id: Clearly, id of n_1 is 1.
- coordinate: *e.g.*, for n_1 the coordinate in the figure can be:

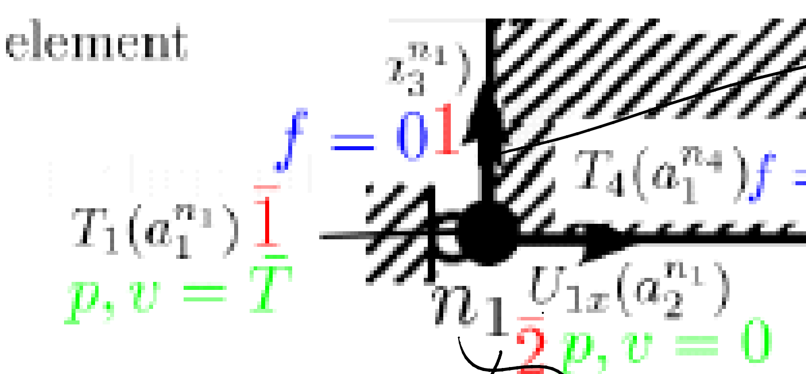
$$\mathrm{crd}_{XY}(n_1) = [0 \quad 0]$$

  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.

We will not discuss functions for the node object.

FEM Solver Objects: 5. Dof: Data





pos : 1
prescribed = false
value = ?
force = 0
field U
component 2

pos : 2

prescribed = true

value = 0

force = ?

option

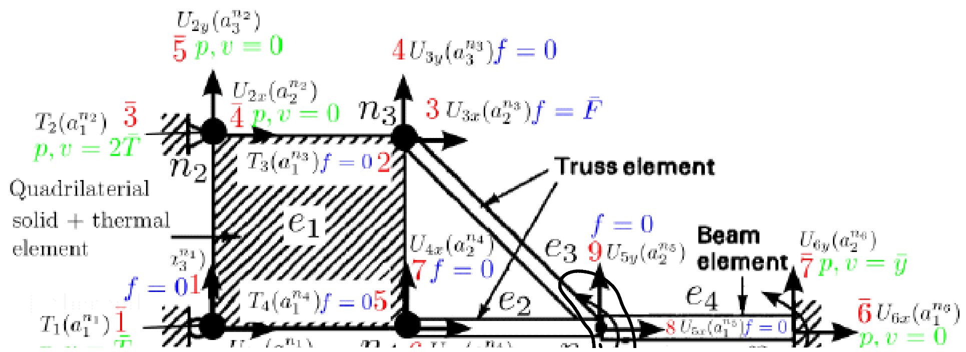
Field : U (displacement)

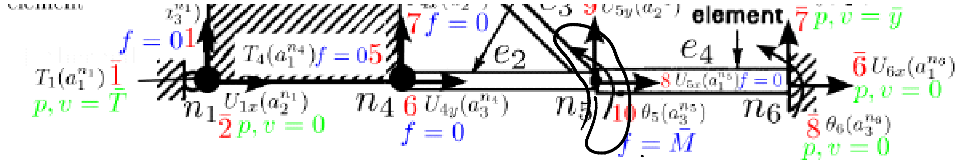
component = 1

Example from C++

```
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;
};
```

FEM Solver Objects: 5. Dof

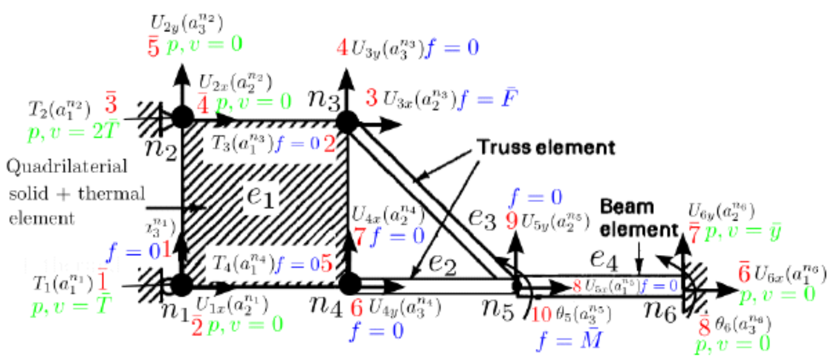




Examples of dof for the structure shown are:

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

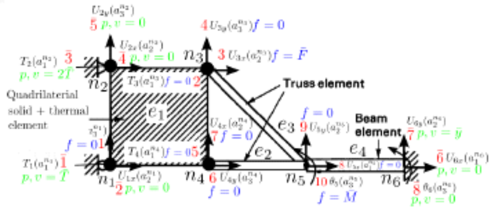
FEM Solver Objects: 6. Other solver related objects



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).

FEM Solver Objects: 7. FEM solver



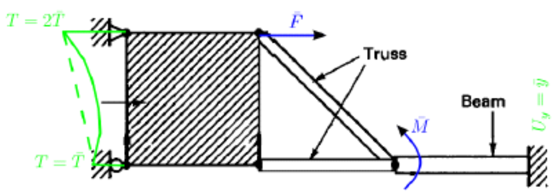
There is not a clear object for FEM solver. However, we can think of FEM solver as an object that is responsible for 1) reading in discretization data; 2) storing all node and element data; 3) solution of global system. Basically FEM solver is the driver for FEM solution.

- dim, n_{dim} spatial dimension for the problem (1D, 2D, and 3D)
- number of nodes (nNodes, n_{n}) in the domain; *e.g.*, $n_{\mathrm{n}} = 6$.
- 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.*, $n_{\mathrm{e}} = 4$.
- 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.*, $n_{\mathrm{f}} = 10$.
- number of prescribed dof (np, n_{p}); *e.g.*, $n_{\mathrm{p}} = 8$.
- number of dof (ndof, $n_{\mathrm{dof}} = n_{\mathrm{f}} + n_{\mathrm{p}}$); *e.g.*, $n_{\mathrm{dof}} = 18$.
- stiffness matrix ($\mathbf{K} = \mathbf{K}_{ff}$); $n_{\mathrm{f}} \times n_{\mathrm{f}}$ matrix.
- force vector ($\mathbf{F} = \mathbf{F}_f$); $n_{\mathrm{f}}(\times 1)$ vector.
- prescribed force vector ($\mathbf{F}_p$) (Optional); $n_{\mathrm{p}}(\times 1)$ 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

Now we are going to cover steps needed to solve and FEM problem and put results back in nodes and elements

STARTING POINT



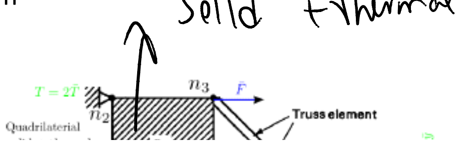
END RESULT:
- ALL NODAL VALUES & FORCES
- ELEMENT VALUES AND FORCES
- SUPPORT FORCES

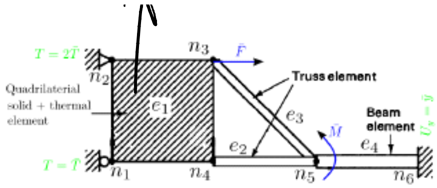
16 steps:

Step 1: Set element nodal dofs

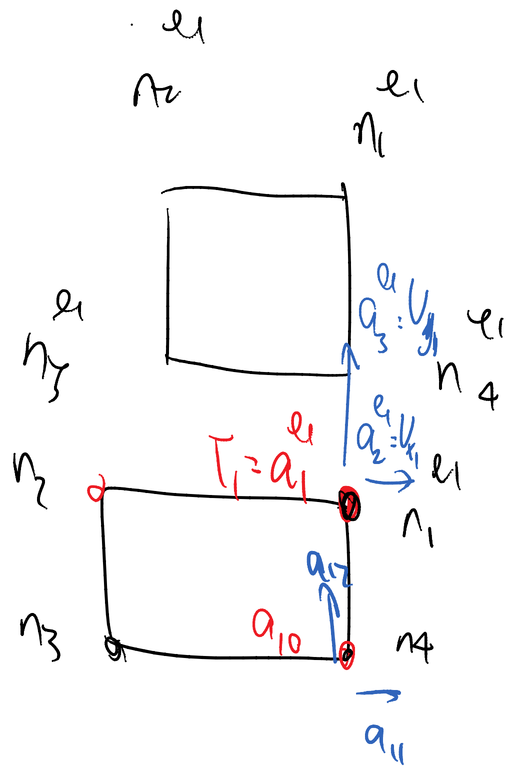
From

Solid +thermal

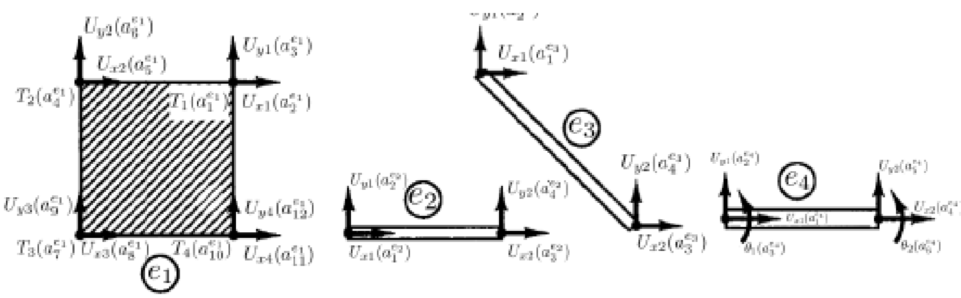




Know each elements nodal dof

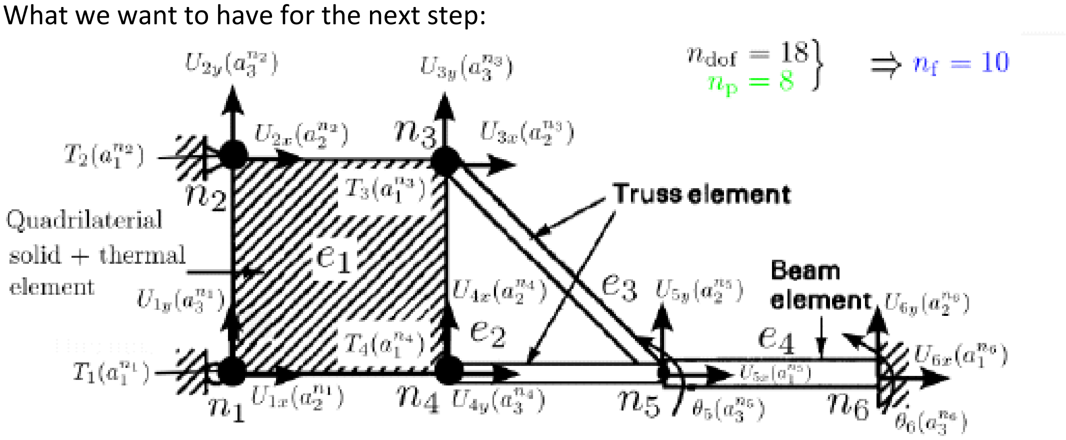


After this step we have the following:



Step 2: Set global dofs using element dofs.

What we want to have for the next step:



Global node DOES NOT know what elements are attached to it
BUT
Elements know what global nodes are attached to them (based on LEM - element nodal map)