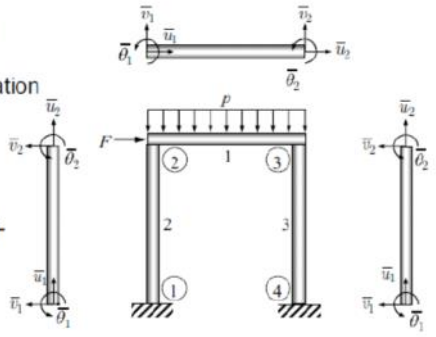
Frames: 2D frame elements

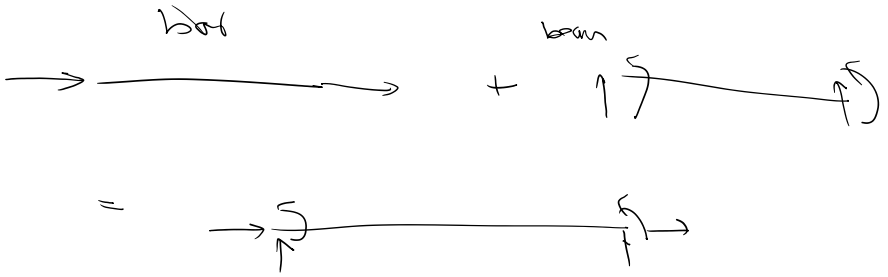
- Beam
  - Vertical deflection and slope. No axial deformation
- Frame structure
  - Can carry axial force, transverse shear force, and bending moment (Beam + Truss)
- Assumption
  - Axial and bending effects are uncoupled
  - Reasonable when deformation is small
- 3 DOFs per node

  $\{u_i, v_i, \theta_i\}$
- Need coordinate transformation like plane truss



source: Nam-Ho Kim, Raphael T. Haftka; http://www2.mae.ufl.edu/nkim/eml5526/ sec. 5

381 / 456



2D frame element: axial and bending stiffness matrices

- Axial deformation (in local coord.)

$$\frac{EA}{L}\begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix}\begin{Bmatrix} \overline{u}_1 \\ \overline{u}_2 \end{Bmatrix} = \begin{Bmatrix} f_{\overline{x}1} \\ f_{\overline{x}2} \end{Bmatrix}$$

- Beam bending

$$\frac{EI}{L^3}\begin{bmatrix} 12 & 6L & -12 & 6L \\ 6L & 4L^2 & -6L & 2L^2 \\ -12 & -6L & 12 & -6L \\ 6L & 2L^2 & -6L & 4L^2 \end{bmatrix}\begin{Bmatrix} \overline{v}_1 \\ \overline{\theta}_1 \\ \overline{v}_2 \\ \overline{\theta}_2 \end{Bmatrix} = \begin{Bmatrix} f_{\overline{y}1} \\ \overline{c}_1 \\ f_{\overline{y}2} \\ \overline{c}_2 \end{Bmatrix}$$

- Basically, it is equivalent to overlapping a beam with a bar
- A frame element has 6 DOFs

source: Nam-Ho Kim, Raphael T. Haftka; http://www2.mae.ufl.edu/nkim/eml5526/ sec. 5

383 / 456

2D frame: local & global coordinate stiffness matrices

- Element matrix equation (local coord.)

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

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

- Element matrix equation (global coord.)

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

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

- Same procedure for assembly and applying BC

source: Nam-Ho Kim, Raphael T. Haftka; http://www2.mae.ufl.edu/nkim/eml5526/ sec. 5

384 / 456

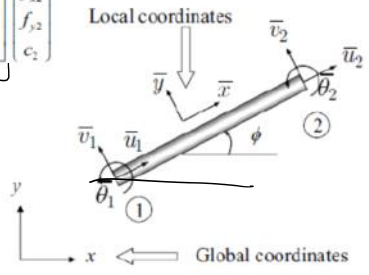
2D frame element: coordinate transformation

- Element-fixed local coordinates $\bar{x}-\bar{y}$
- Local DOFs $\{\bar{u},\bar{v},\bar{\theta}\}$ Local forces $\{\bar{f}_{\bar{x}},\bar{f}_{\bar{y}},\bar{c}\}$
- Transformation between local and global coord.

$$\begin{Bmatrix} f_{\bar{x}1} \\ f_{\bar{y}1} \\ \bar{c}_1 \\ f_{\bar{x}2} \\ f_{\bar{y}2} \\ \bar{c}_2 \end{Bmatrix} = \begin{bmatrix} \cos\phi & \sin\phi & 0 & 0 & 0 & 0 \\ -\sin\phi & \cos\phi & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & \cos\phi & \sin\phi & 0 \\ 0 & 0 & 0 & -\sin\phi & \cos\phi & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{Bmatrix} f_{x1} \\ f_{y1} \\ c_1 \\ f_{x2} \\ f_{y2} \\ c_2 \end{Bmatrix}$$

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

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



source: Nam-Ho Kim, Raphael T. Haftka; http://www2.mae.ufl.edu/nkim/eml5526/ sec. 5

382 / 456

For the term project

1. Calculate the stiffness 6x6 from

- Element matrix equation (local coord.)

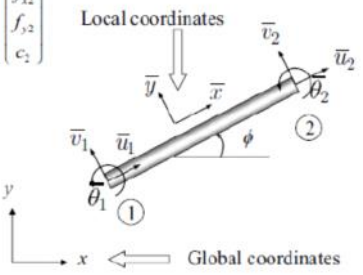
$$\begin{bmatrix} a_1 & 0 & 0 & -a_1 & 0 & 0 \\ 0 & 12a_2 & 6La_2 & 0 & -12a_2 & 6La_2 \\ 0 & 6La_2 & 4L^2a_2 & 0 & -6La_2 & 2L^2a_2 \\ -a_1 & 0 & 0 & a_1 & 0 & 0 \\ 0 & -12a_2 & -6La_2 & 0 & 12a_2 & -6La_2 \\ 0 & 6La_2 & 2L^2a_2 & 0 & -6La_2 & 4L^2a_2 \end{bmatrix} \begin{Bmatrix} \overline{u}_1 \\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

2. Calculate T from

$$\begin{Bmatrix} f_{\overline{x}1} \\ f_{\overline{y}1} \\ \overline{c}_1 \\ f_{\overline{x}2} \\ f_{\overline{y}2} \\ \overline{c}_2 \end{Bmatrix} = \begin{bmatrix} \cos\phi & \sin\phi & 0 & 0 & 0 & 0 \\ -\sin\phi & \cos\phi & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & \cos\phi & \sin\phi & 0 \\ 0 & 0 & 0 & -\sin\phi & \cos\phi & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{Bmatrix} f_{x1} \\ f_{y1} \\ c_1 \\ f_{x2} \\ f_{y2} \\ c_2 \end{Bmatrix}$$

$\{\overline{\mathbf{f}}\} = [\mathbf{T}]\{\mathbf{f}\}$

$\{\overline{\mathbf{q}}\} = [\mathbf{T}]\{\mathbf{q}\}$



3. Rotate K from 1 using T from 2 to global coordinate system using
   - Element matrix equation (global coord.)

$$[\overline{\mathbf{k}}][\mathbf{T}]\{\mathbf{q}\} = [\mathbf{T}]\{\mathbf{f}\} \implies [\mathbf{T}]^T[\overline{\mathbf{k}}][\mathbf{T}]\{\mathbf{q}\} = \{\mathbf{f}\} \implies [\mathbf{k}]\{\mathbf{q}\} = \{\mathbf{f}\}$$

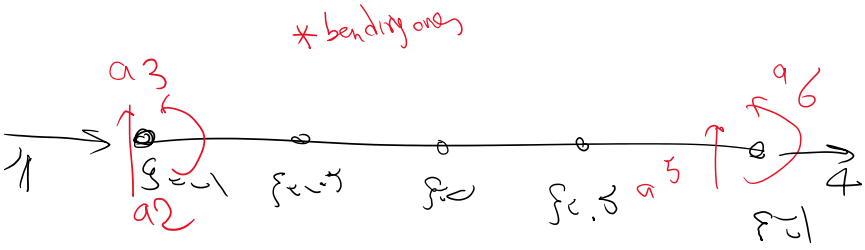
$$[\mathbf{k}] = [\mathbf{T}]^T[\overline{\mathbf{k}}][\mathbf{T}]$$

   - Same procedure for assembly and applying BC

source: Nam-Ho Kim, Raphael T. Haftka; http://www2.mae.ufl.edu/nkim/eml5526/ sec. 5

384 / 456

For the term project, I have asked to report, y, theta, M, V on 5 locations:



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 $\frac{dx}{d\xi} = \frac{L^e}{2}$:

$$\theta(\xi) = \frac{dy}{dx}(\xi) = \frac{\frac{dy}{d\xi}(\xi)}{\frac{dx}{d\xi}(\xi)} = \frac{2}{L^e}\left\{\frac{dN_1^e}{d\xi}(\xi)a_1^e + \frac{dN_2^e}{d\xi}(\xi)a_2^e + \frac{dN_3^e}{d\xi}(\xi)a_3^e + \frac{dN_4^e}{d\xi}(\xi)a_4^e\right\}$$

- Moment is directly obtained by differentiating the above equation:

$$M(\xi) = E(\xi)I(\xi)\frac{d^2y}{dx^2}(\xi) = E(\xi)I(\xi)\mathbf{B}^e(\xi)$$
$$= E(\xi)I(\xi)\{B_1^e(\xi)a_1^e + B_2^e(\xi)a_2^e + B_3^e(\xi)a_3^e + B_4^e(\xi)a_4^e\} \quad \text{cf. (424) for } \mathbf{B}^e$$

- 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:

$$\frac{dM}{dx}(\xi) = \frac{\frac{dM}{d\xi}(\xi)}{\frac{dx}{d\xi}(\xi)} = \frac{2EI}{L^e}\left\{\frac{dB_1^e}{d\xi}(\xi)a_1^e + \frac{dB_2^e}{d\xi}(\xi)a_2^e + \frac{dB_3^e}{d\xi}(\xi)a_3^e + \frac{dB_4^e}{d\xi}(\xi)a_4^e\right\}$$

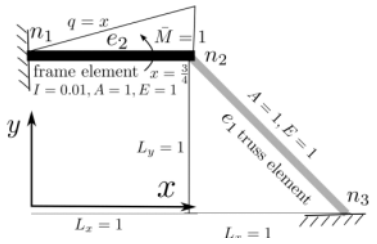
from y with the difference that if EI are not constant we need to take it into account.
For constant EI we have:

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

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

374 / 456

Use formulas on slide 374 for the last problem of HW4



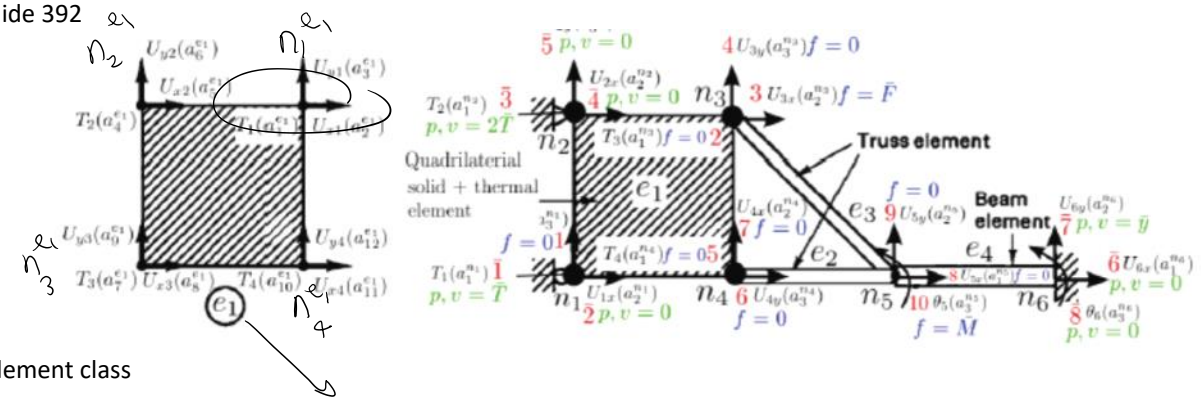
(d) Obtain displacement (y), rotation ($\theta = \frac{\mathrm{d}y}{\mathrm{d}x}$), and moment ($M = EI\frac{\mathrm{d}^2y}{\mathrm{d}x^2}$) for the frame element at $x = 0.5$. Note that $y(\xi) = \sum_{i=1}^{4} N_i^e(\xi)a_i^e$. Also, since $\mathbf{B}^e = \frac{\mathrm{d}^2\mathbf{N}^e}{\mathrm{d}x^2} \Rightarrow M = EI\sum_{i=1}^{4} B_i^e(\xi)a_i^e$. **(30 Points)**

Coding a finite element method

Object-oriented programing:
We have objects, and objects have:
- Data
- Functions

associated with them

Slide 392



Element class

Data
- id 1
- nNodes 4
- eNodes (LEM) [3, 2, 1, 4]
- edof (a^e) vector of unknowns $[T^1\ U_x^1\ U_y^1 \mid T^2\ U_x^2\ U_y^2 \mid \ldots \mid T^4\ U_x^4\ U_y^4]$
- dofMap (M) $[2\ 3\ 4 \mid \bar{3}\ \bar{4}\ \bar{5} \mid 1\ \bar{2} \mid 1 \mid 5\ 6\ 7]$ $n \times 1$
- Stiffness matrix k^e

- stiffness matrix k^e

- forces (vectors) $f^e = \underbrace{(f^e_\gamma + f^e_N + \dots)}_{f^e_o \ (f^e_{other})} - f^e_D$ → Dirichle

for the term project we only have f^e_D

- e Type square

- { physics } 2D elasticity + thermal

Examples of functions

← Calculate stiffness

polymorphism Different element to element

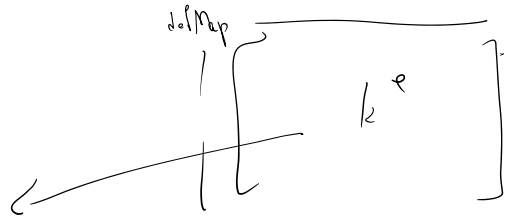
bar $\frac{AE}{L}\begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix}$ beam $\frac{EI}{l^3}\Big[\quad\Big]_{4\times4}$, - truss - -

Matlab, Fortran, ... (procedural)

```
if (elementT == bar)
    K = AE/L [1 -1; -1 1]
else if (elementT == beam)
```

Assemble stiffness

Same for all elements



K

FEM Solver Objects: 1. Element: Function

Some sample element functions are:

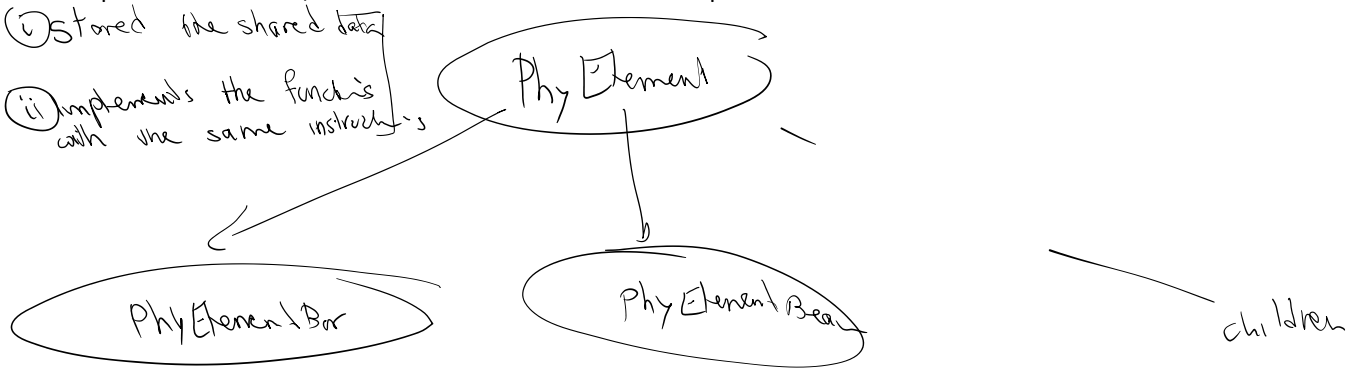
- Calculate $\mathbf{k}^e$ (virtual)
- Calculate $\mathbf{f}_o^e$: sum of all forces, but $\mathbf{f}_D^e$ (virtual)
- Compute_Output_Element: computes and outputs element; *e.g.*, axial forces for truss: (virtual)
- Calculate $\mathbf{f}_D^e = \mathbf{k}^e\mathbf{a}^e$
- Assemble $\mathbf{k}^e$ and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

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 $\mathbf{f}_o^e$ 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 $\mathbf{f}_D^e = \mathbf{k}^e\mathbf{a}^e$) where the same exact routine is performed for all types of objects.

394 / 456

In the provided C++ code, this is how we can have different implementations of stiffness matrix



i)

PhyElement.h

```
class PhyElement
{
Shared data:
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
```

(ii)

Functions with the same procedure

Example assembly

```
// Step 11: Assembly from local to global system
void AssembleStiffnessForce(MATRIX& globalK, VECTOR& globalF);
....
void PhyElement::AssembleStiffnessForce(MATRIX& globalK, VECTOR& globalF)
{
    fee.resize(nedof);
    if (foe.size() == nedof)
        fee = foe;
    else
        fee = 0.0;

    int I, J;
    for (int i = 0; i < nedof; ++i)
    {
        I = dofMap[i];
        if (I < 0) // prescribed dof
            continue;
```

Examples of class specific data and functions:
PhyElementBar

```
class PhyElementBar : public PhyElement
{
public:
    virtual void setGeometry();
    virtual void setInternalMaterialProperties(PhyMaterial* pMat);
    virtual void Calculate_ElementStiffness_Force();
    virtual void SpecificOutput(ostream& out) const;
    double L;
    double A;
    double E;
};
```

bar Element specific funct's
" derived from the parent class"

→ bar specific data E, A, L

```
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;
}
```

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

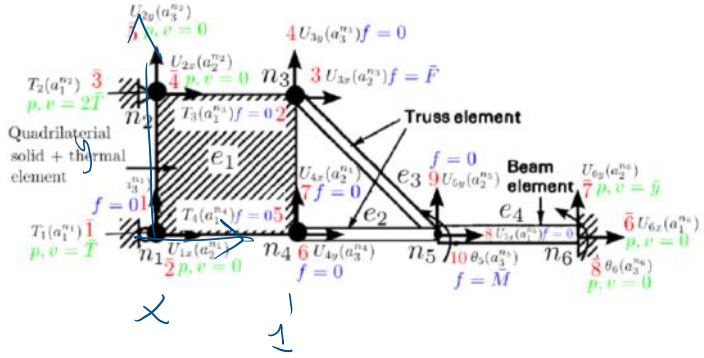
Node class

```
class PhyNode
{
public:
    void set_nndof(int nndofIn);
    void UpdateNodePrescribedDofForces(VECTOR& Fp);
    ID id;
    VECTOR coordinate;
    vector <PhyDof> ndof;
    int nndof;// number of dofs
};
```

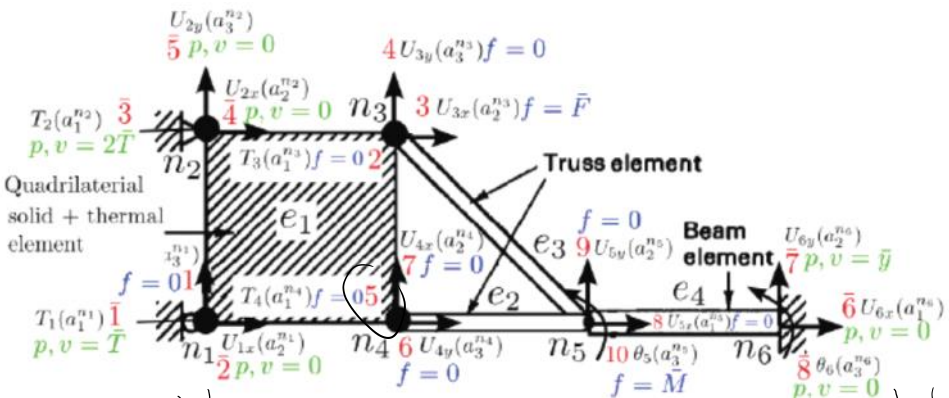
id 4
Coordinate (1, 0)
{ , , }

```
    vector <PhyDof> ndof;
    int nndof;// number of dofs
};
```

FEM Solver Objects: 4. Node: Data



FEM Solver Objects: 5. Dof: Data



	prescribed boolean	value	force	pos	Field	needed for a more general component
dof #1 of node 4	0	?	0	5	T	—
dof #3 of node 5	0	?	M	10	M	—
dof #2 of node 2	1	0	?	4	U	1 (x)

- 4
& boolean is redundant
just put 4

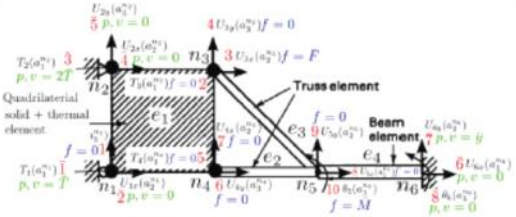
```
class PhyDof
{
public:

    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}
//    FieldF;
//    INDEX i;
};
```

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

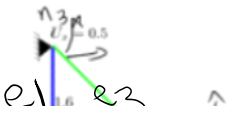
Solution steps

The steps for FEM solution are:

1. Set Element nodal dofs.
2. Set global dofs using element dofs.
3. Compute n_{f} from n_{dof} and n_{p} and resize and zero stiffness matrix and force vector.
4. Set global prescribed dofs.
5. Set global free dofs.
6. Set dof (free + prescribed) positions.
7. Set $\mathbf{F}(\mathbf{F}_f)$.
8. Set element dof maps $\mathbf{M}_t^e$.
9. Set element (prescribed) dofs.
10. Compute element stiffness matrix and force vectors.
11. Assemble element stiffnesses and forces to global system.
12. Solve for (free) dofs $\mathbf{a}$ from $\mathbf{Ka} = \mathbf{F}$.
13. Assign $\mathbf{a}$ to nodes and elements.
14. Compute prescribed dof forces: $\mathbf{F}_p$ (if needed).
15. Compute (if needed) output nodes and elements.

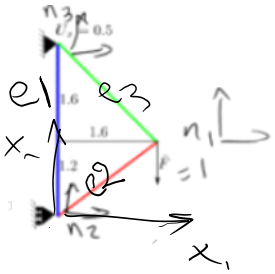
402 / 456

We'll solve this next following these steps:



dim 2 → 2D
ndofpn 2
Nodes
nNodes 3

num dof/node = 2



ndofpn 2 num dof/node = 2
Nodes
nNodes 3
id crd
1 1.6 1.2
2 0 0
3 0 2.8

3elements

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

#nodes for element

LEM

vector<int> eNodes; // element node vector

material #1 for all the element (Some A&E)

elementType
1 bar
2 beam
3 truss
4 frame

$n = ndofs = nnodes * ndof/node = 3*2 = 6$

In FEM every dof is by default free and it has zero force. Anything other than that should be specified.

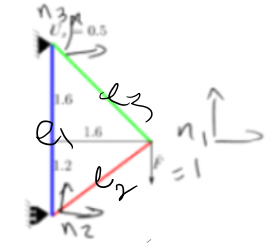
PrescribedDOF
np 3

$np = 3$

$nf = n - np = 6 - 3 = 3$

nf: #free dofs; n: #dofs; np: # prescribed dofs

Now that we have nf and np, we can form all the element and node storages



nf = 3 np = 3 n = 6

node	P	pos	V	f
1				
2				
3				

U_f (size nf) = 1, 2, 3 $\quad K_{nf \times nf} \quad F_n = [\ ]$

F_p = 1, 2, 3

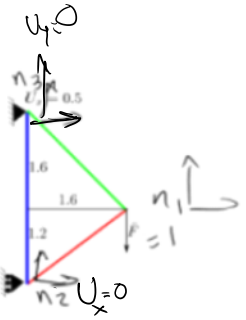
e	LEM	dof Map	dofs	fe
1	[2,3]			
2	[2,1]			
3	[3,1]			

We did steps 1, 2, 3 above

1. Set Element nodal dofs.
2. Set global dofs using element dofs.
3. Compute n_f from n_{dof} and n_p and resize and zero stiffness matrix and force vector.

Step 4: Set global prescribed nodal dof

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



node	P	pos	V	f
1	0		0?	0
	0		0?	0
2	1		0.0	0?
	0		0?	0
3	1		0.5	0?
	1		0.0	0?

$U_p = \begin{matrix}1\\2\\3\end{matrix}\begin{bmatrix}\\\\\end{bmatrix}$ size n_p $\quad K_{n_p \times n_f}$ $\quad F_n = \begin{bmatrix}\\\\\end{bmatrix}$

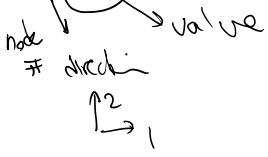
$F_p = \begin{matrix}1\\2\\3\end{matrix}\begin{bmatrix}\\\\\end{bmatrix}$

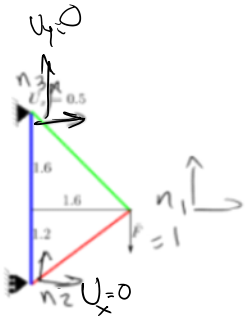
e	LEM	dof Map	dofs	fe

Step 5: Set global free nodal dof

We specify only the nonzero forces

FreeDofs
nNonZeroForceFDOFs 1
node node_dof_index value
1 2 -1.0





$n_f = 3 \quad n_p = 3 \quad n = 6$

node	P	BOS	V	f
1	0		0?	0
	0		0?	0
2	1		0.0	0?
	0		0?	-1
3	1		0.5	0?
	1		0.0	0?

U_f = size n_f = $\begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix}$ $\quad K_{n_f \times n_f}$ $\quad F_n = \begin{bmatrix} \\ \\ \end{bmatrix}$

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

e	LEM	dof Map	dofs	f_e