

Step 10: calculating stiffness matrix

Example of virtual function:

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

```
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;
};
```

```
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;
}
```

Step 11: Assembly from local to global system

All the elements have the same form of assembly to the global system

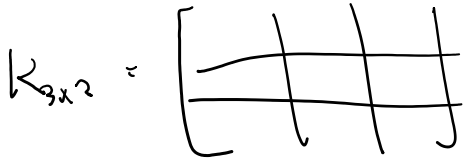
Truss example: Assembly of global system

The diagram shows three truss elements:
 

- e1**: Vertical element,  $\theta = 90^\circ$ ,  $c=0, s=1$ . Local stiffness matrix  $k^{e1} = \begin{bmatrix} 3.8 & 0 \\ 0 & 0 \end{bmatrix}$ .
- e2**: Diagonal element,  $\theta = \tan^{-1}(1/2)$ ,  $c=0.5, s=0.6$ . Local stiffness matrix  $k^{e2} = \begin{bmatrix} 2.3 & 0 \\ 0 & 2.3 \end{bmatrix}$ .
- e3**: Diagonal element,  $\theta = -45^\circ$ ,  $c=1/\sqrt{2}, s=-1/\sqrt{2}$ . Local stiffness matrix  $k^{e3} = \begin{bmatrix} 2.2 & 0 \\ 0 & 2.2 \end{bmatrix}$ .

 The global stiffness matrix  $K$  is assembled as  $K = k^{e1} + k^{e2} + k^{e3}$ , resulting in  $K = \begin{bmatrix} 3.8 & 0 & 0 \\ 0 & 2.3 & 0 \\ 0 & 0 & 2.2 \end{bmatrix}$ .

step 10 calculates  $k_{e \times e}$  &  $f_e$



Assembly of one element for local  $i = 1$ : nodal

for  $e_s$  +  $\dots$

for local index  $i = 1: \text{ndof}$   
 global index  $I = \text{dofMap}(i)$

for  $i = 1: \text{ndof}$   
 $\text{dofMap} = [2 \ 3 \ 1 \ 2]$

if  $I < 0$  // I prescribed

Continue;

end

~~$f_I = f_{\text{ext}}(i)$~~

recall  $f = f_0^e - f_D^e$

for  $j = 1: \text{ndof}$

$J = \text{dofMap}(j)$

if  $J > 0$

$K(I, J) = K(I, J) + k_e(i, j);$

else

$f_D(i) = f_D(i) + k_e(i, j) * a_e(j);$

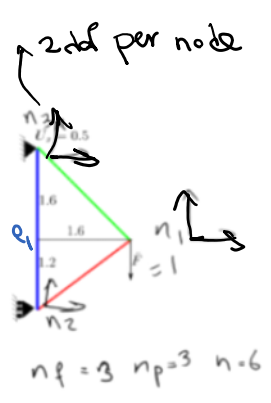
$$\begin{bmatrix} p^e \\ f_D^e \end{bmatrix} = K^e a^e$$

end

$f_{\text{ext}}(i) = f_{\text{ext}}(i) - f_D(i);$

$F(I) = F(I) + f_{\text{ext}}(i);$

end



$\text{ndof} = 3 \times 2 = 6$        $n_p = 3 \rightarrow n_f = \text{ndof} - n_p = 3$

node	P	pos	v	f
1	0	1		0
	0	2		-1
2	1	1	.0	0
	0	3		0
3	1	2	0.5	0
	1	3	0.6	0

step 6

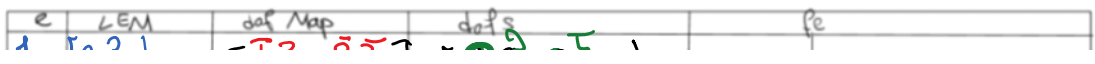
$$U_p = \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix}$$

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

$K_{n_f \times n_f}$

$F = F_n = \begin{bmatrix} 0 \\ -1 \\ 0 \end{bmatrix}$

step 7



$n_f = 3$  r.p. ...

step 6

e	LEM	dof Map	dofs	fe
1	[2 3]	[1 3 2 3]	[0 0 0 0]	
2	[2 1]	[1 3 1 2]	[0 0 0 0]	
3	[3 1]	[2 3 1 2]	[0 0 0 0]	

Handwritten annotations: Red brackets under the 'dof Map' column group the first two rows as 'step 8' and the last two rows as 'step 9'. Green circles highlight the '0' values in the 'dofs' column.

## Step 11: Assembly from local to global system

- $K$  and  $F$  (global stiffness and force) are already sized and set to zero.
- Element level (local) stiffness and force is calculated (Step 10).
- Element local to global dof  $M_l^e$  is already set (Step 8).
- Using dof map, we assemble local values to global values.
- Clearly, only free dofs are added to stiffness matrix and force vector.
- Element dof values (dofs:  $a^e$  is also set (Step 9).
- $f_D^e = k^e a^e$  may not need to be formed and can be directly added to  $f_e^e$ .

```

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
    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 ke
        J = dofMap(j) global dof corresponding to j
        if (J > 0) now both l and J are free and can add ke(i,j) to global K
          K(l, J) = K(l, 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(l) = F(l) + fee(i) element's total force fee component i'th is computed → added to F(l)
  end
end
end
end
  
```

431 / 456

## Step 12: Solve global (free) dof a from $Ka = F$

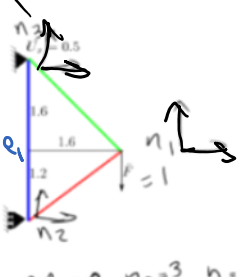
- Two major computational costs during FEM solve are:
  - 1 Assembly: Refers to: all node, element, and dof set up; computation of local ke and fee; assembly of those to global system. This step scales linearly versus  $n_e$  ( $ne$ )
  - 2 Linear algebra solution:  $Ka = F$ : We solve for unknown a. Although conceptually simple, this step is a major source of computational cost. It scales higher than linear versus  $n_e \Rightarrow$  As the problem size increases this term becomes more dominant.
- Solution of  $Ka = F$ :
  - WE DO NOT OBTAIN a from  $a = K^{-1}F$ . We do not invert  $K$ .
  - We only solve the problem for the specific RHS of  $F$ .
  - In Comparison  $K^{-1}$  corresponds to the solution of  $Ka = F$  for  $n_f$  RHS of  $F = e_i, i = 1, \dots, n_f$  where  $n_f$  is the number of rows (and columns) of  $K$ .
  - We employ methods such as LU factorization that computationally only solve the problem for the given RHS  $F$ .
  - We take advantage of the structure of stiffness matrix: symmetry, bandedness, sparsity in choosing the right solution technique.
  - order of free dofs affects band of the matrix  $\rightarrow$  various algorithms reorder free dofs such that the matrix band get smaller and the solution cost is optimized.
  - In your term projects you can simply employ simply compute

433 / 456

$$a = K^{-1}F$$

## Step 13: Assign a to nodes and elements

2 dof per node



$n_f = 3$   $n_p = 3$   $n = 6$

$n_{dof} = 3 \times 2 = 6$

$n_p = 3 \rightarrow n_f = n_{dof} - n_p = 3$

node	P	pos	v	f
1	0	1	-2.225	0
	0	2	-3.29	-1
2	1	1	0.0	0?
	0	3	1.2	0
3	1	2	0.5	0?
	1	3	0.6	0?

step 6

$$K_{free} = \begin{bmatrix} 1 & -2.223 & -3.29 \\ -2.223 & 2 & -1 \\ -3.29 & -1 & 3 \end{bmatrix}$$

$$F = F_{free} = \begin{bmatrix} 0 \\ -1 \\ 0 \end{bmatrix}$$

step 7

e	LEM	dof Map	dofs	fe
1	[2 3]	[1 3 2 3]	[0.5 0.0 0.0]	
2	[2 1]	[1 3 1 2]	[0 0 0]	
3	[3 1]	[2 3 1 2]	[0.5 0 0]	

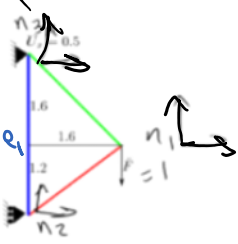
step 8

step 9

```

for n = 1:nNodes
    for dofi = 1:node(n).ndof
        num dof for node (n)
        if node(n).ndof(dofi).p == false
            free dof
            posn = node(n).ndof(dofi).pos
            position of dof in global free F
            node(n).ndof(dofi).v = dofs(posn)
            set free dof val to corresponding val in global dofs (a)
        end
    end
end
end
    
```

2 dof per node



$n_f = 3$   $n_p = 3$   $n = 6$

$n_{dof} = 3 \times 2 = 6$

$n_p = 3 \rightarrow n_f = n_{dof} - n_p = 3$

node	P	pos	v	f
1	0	1	-2.225	0
	0	2	-3.29	-1
2	1	1	0.0	0?
	0	3	1.2	0
3	1	2	0.5	0?
	1	3	0.6	0?

step 6

$$K_{free} = \begin{bmatrix} 1 & -2.223 & -3.29 \\ -2.223 & 2 & -1 \\ -3.29 & -1 & 3 \end{bmatrix}$$

$$F = F_{free} = \begin{bmatrix} 0 \\ -1 \\ 0 \end{bmatrix}$$

step 7

e	LEM	dof Map	dofs	fe
1	[2 3]	[1 3 2 3]	[0.5 -1.2 0.5]	
2	[2 1]	[1 3 1 2]	[0 -1.2 -2.23 -3.29]	
3	[3 1]	[2 3 1 2]	[0.5 0 -2.3 -3.29]	

step 8

step 9

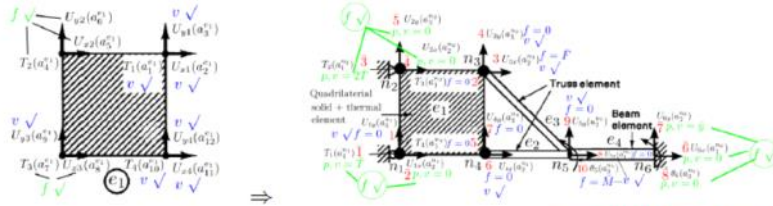
```

for e = 1:ne loop over elements
  for i = element(e).nedof loop over element dofs; nedof = # dof (n_dof^e)
    posn = element(e).dofMap(i) corresponding global position using dofMat (M_i^e)
    if (posn > 0) free dof
      element(e).edofs(i) = dofs(posn)
      set free element dof a^e to corresponding val in global dofs (a)
    end
  end
end
end
end

```

Remaining things: Element forces and prescribed dof forces in global structure

### Step 14: Compute prescribed dof forces



- At global nodes, the only unknowns at this stage are **prescribed forces (reaction forces for solid mechanics)**.
- To set global nodal prescribed forces we need to:
  - Compute all element forces:**  $f^e$  (all forces but  $f_D^e$  is calculated), we add this to contributions from nodal dof values (previously known prescribed and newly solved free dofs).
  - Add element nodal forces to get global nodal forces:** The contribution (sum) of all element forces at a given **prescribed dof** is equal to the **prescribed force (reaction force)** at that dof. For convenient we store these values in  $F_p$  ( $n_p$  vector).
  - Assign prescribed global nodal dof forces** from their corresponding values in  $F_p$ .
- loop over elements → steps 1 & 2; loop over nodes → step 3.

Handwritten notes and diagrams illustrating the calculation of prescribed forces:

2 dof per node

$n_{dof} = 3 \times 2 = 6$

$n_p = 3 \rightarrow n_p = n_{dof} - n_f = 3$

Diagram of a 3-node structure with nodes 1, 2, and 3. Node 1 is at the bottom left, node 2 is at the top left, and node 3 is at the bottom right. A force  $f=1$  is applied at node 2. The structure consists of a vertical member (1-2), a diagonal member (1-3), and a horizontal member (2-3).

node	P	pos	V	F
1	0	1	0.0	0
	0	2	-3.29	-1
2	1	1	0.0	0?
	0	3	1.2	0
3	1	2	0.5	0?
	1	3	0.6	0?

Matrix  $K_{ref}$  (Step 10):

$$K_{ref} = \begin{bmatrix} 1 & 2 & 3 \\ 2 & 2.23 & -3.29 \\ 3 & -3.29 & 1.2 \end{bmatrix}$$

Force vector  $F_p$  (Step 7):

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

Global force vector  $F$  (Step 7):

$$F = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}$$

Element matrices (Step 8):

e	LEM	dof Map	dofs	$f^e = -(f_D^e - f_D^e) = f_D^e - f_D^e = k \cdot f^e$
1	[2 3]	[1 3 2 3]	[0 1.2 0.5 0]	[0 -4.285 0 4.285]
2	[2 1]	[1 3 1 2]	[0 1.2 2.3 -3.29]	[5.715 4.286 7.5715 -4.286]
3	[3 1]	[2 3 1 2]	[0.5 0 2.3 -3.29]	[.5714 .5714 .5714 -.5714]

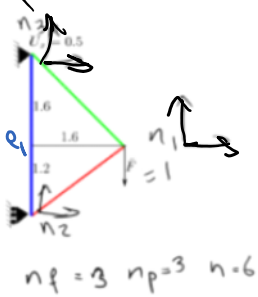
Step 14:  $f^e = -(f_D^e - f_D^e) = f_D^e - f_D^e = k \cdot f^e$

$k^1_{a_1 c} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0.357 & 0 & -0.357 \\ 0 & 0 & 0 & 0 \\ 0 & -0.357 & 0 & 0.357 \end{bmatrix} \begin{bmatrix} 0 \\ -1.2 \\ 0.5 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ -0.4285 \\ 0 \\ 0.4285 \end{bmatrix}$	$k^2_{a_2 c} = \begin{bmatrix} 0.32 & 0.24 & -0.32 & -0.24 \\ 0.24 & 0.18 & -0.24 & -0.18 \\ -0.32 & -0.24 & 0.32 & 0.24 \\ -0.24 & -0.18 & 0.24 & 0.18 \end{bmatrix} \begin{bmatrix} 0 \\ -1.2 \\ -0.2125 \\ -3.2980 \end{bmatrix} = \begin{bmatrix} 0.5715 \\ 0.4286 \\ -0.5715 \\ -0.4286 \end{bmatrix}$	$k^3_{a_3 c} = \begin{bmatrix} 0.221 & -0.221 & -0.221 & 0.221 \\ -0.221 & 0.221 & 0.221 & -0.221 \\ -0.221 & 0.221 & 0.221 & -0.221 \\ 0.221 & -0.221 & -0.221 & 0.221 \end{bmatrix} \begin{bmatrix} 0.5 \\ 0 \\ -0.2125 \\ -3.2980 \end{bmatrix} = \begin{bmatrix} -0.5714 \\ 0.5714 \\ 0.5714 \\ -0.5714 \end{bmatrix}$
--	---	--

2 dof per node

$n_{dof} = 3 \times 2 = 6$

$n_p = 3 \rightarrow n_f = n_{dof} - n_p = 3$



node	P	POS	V	F
1	0	1	0.2125	0
	0	2	-3.29	-1
2	1	3	0.0	0?
	0	3	1.2	0
3	1	2	0.5	0?
	1	3	0.6	0?

$$U_p = \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix} \begin{bmatrix} -2.23 \\ -3.29 \\ -1.2 \end{bmatrix}$$

$$F_p = \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix} \begin{bmatrix} 0.5715 \\ 0.5714 \\ 4.285 \end{bmatrix}$$

$$K_{p \times p} = \begin{bmatrix} 0.5714 & & \\ & 0.5714 & \\ & & 1 \end{bmatrix}$$

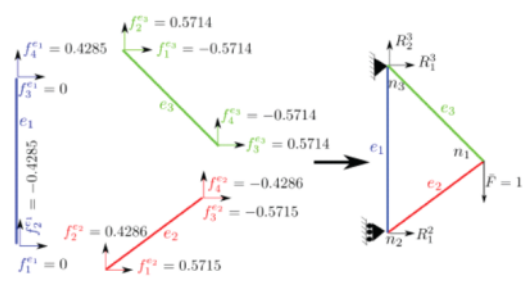
$$F = F_p = \begin{bmatrix} 0 \\ -1 \\ 0 \end{bmatrix}$$

Step 7

e	LEM	dof Map	dofs	$-f^e = f_0^e - f_1^e = f_0^e - f_1^e = k_{21}^e f_0^e$
1	[2 3]	[1 3   2 3]	[0 -1.2 0.5]	[0 -4.285 0 4.285] (Step 14)
2	[2 1]	[1 3   2 1]	[0 -1.2 2.23 -3.29]	[0.5715 4.286 0.5715 -4.286]
3	[3 1]	[2 3   1 2 3]	[0.5 0 2.3 -3.29]	[0.5714 0.5714 0.5714 -0.5714]

Step 8, Step 9

### Truss Example: Reaction Forces



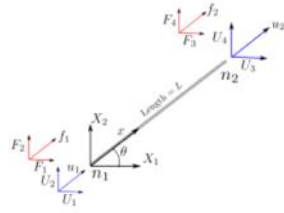
- First, we compute reaction forces by adding up forces from individual elements that contribute to reaction forces:

$$R_1^2 = f_1^{e1} + f_1^{e2} = 0 + 0.5715 = 0.5715 \quad (397a)$$

$$R_1^3 = f_3^{e1} + f_3^{e3} = 0 + -0.5714 = -0.5714 \quad (397b)$$

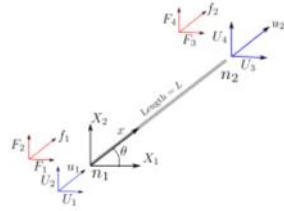
$$R_2^3 = f_4^{e1} + f_2^{e3} = 0.4285 + 0.5714 = 0.9999 \quad (397c)$$

## Step 15: Compute/output nodes & elements: b) elements



- The computation and output of an element is again a **black box** function.

## Step 15: Compute/output nodes & elements: b) elements

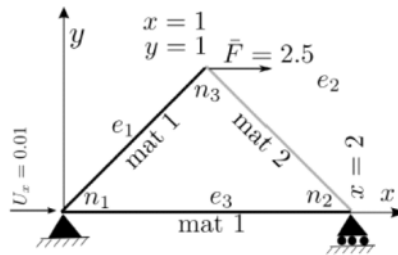


- The computation and output of an element is again a **black box** function.

## 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 nNodes 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
FreeDOFs
nNonZeroForceFDOFs 1
node node_dof_index value
3 1 2.5
Materials
nMat 2
id numPara Paras
1 2 100 1
2 2 200 2
    
```



$$E_1 = 100, A_1 = 1$$

$$E_2 = 200, A_2 = 2$$

## Output file format

- Main FEMSolver function would be:

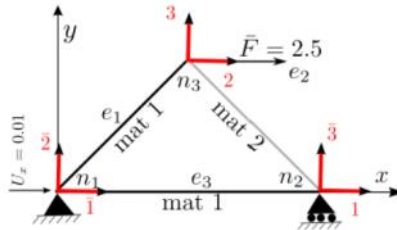
```
function FEMSolver(runName, verboseOutput)
```

- `runName` is the name of the run (e.g., TrussA).
- FEMSolver opens a file with name `runName.txt` to read the problem (with the format conforming to the sample on slide 455).
- This function, computes and outputs results in file `runName.out`.
- `verboseOutput`: if it is true more data will be output (explained on the next slide).
- Format of output file is
  - 1 Nodes: Node dof values and forces (plus dof position and prescribed boolean with verbose option).
  - 2 Elements: All element nodal forces (plus element specific output; e.g., axial force for bars and trusses, etc.).

454 / 456

## Output file format

```
Nodes
nNodes 3
id crd
values
forces
position(verbose)
prescribed_boolean(verbose)
1 0 0
a1_1 a1_2
F1_1 F1_2
-1 -2 (verbose)
1 1 (verbose)
2 2 0
a2_1 a2_2
F2_1 F2_2
1 -3 (verbose)
0 4 (verbose)
a3_1 a3_2
F3_1 F3_2
2 3 (verbose)
0 0 (verbose)
Elements
ne 3
id elementType
forces(verbose)
specific output
1 3
fee1_1 fee1_2 fee1_3 fee1_4 (verbose)
Te1
2 3
fee2_1 fee2_2 fee2_3 fee2_4 (verbose)
Te2
3 3
fee3_1 fee3_2 fee3_3 fee3_4 (verbose)
Ta3
```



- lines with `(verbose)` are only output for `verboseOutput == 1`. Obviously `(verbose)` is not printed in either case and is only printed for clarity here.
- `ai,j`: is value (solution) for node `i` dof number `j`; e.g., `a3_1` is `x` displacement at node 3 (`x = 1, y = 1`).
- `Fi,j`: is force for node `i` dof number `j`; e.g., `F3_1` is `x` force at node 3 (which should be equal to 2.5, why?)
- `feei,j`: is total force (foe + fDe) for element `i` dof number `j`; e.g., `fee3_1` is the `x` force at its left node (global `n1`).
- Last item of element output is specific to its type.
- For 2 node bar and truss elements `Tei` is the axial force in the element.

455 / 456

For the truss example in the class  
TrussTestOutput.txt

```
Nodes
nNodes 3
id crd
values
forces
position(verbose)
prescribed_boolean(verbose)
```

```
1 1.6 1.2
-0.212127 -3.29812
0 -1
0 1
0 0
```



```

2 0 0
0 -1.2
0.5714290
0 2
1 0

```

```

3 0 2.8
0.5 0
-0.571429 1
1 2
1 1

```

Elements

ne 3

id ElementType

forces(verbose)

specific output

1 3

-0 -0.428571 -0 0.428571

0.428571

2 3

0.571429 0.428571 -0.571429 -0.428571

-0.714286

3 3

-0.571429 0.571429 0.571429 -0.571429

0.808122

global\_unknowns-dofs

size 3 -0.212127 -3.29812 -1.2

Support\_forces-Fp

size 3 0.571429 -0.571429 1

K

rows3 cols 3

0.540971 0.0190291 -0.24

0.0190291 0.400971 -0.18

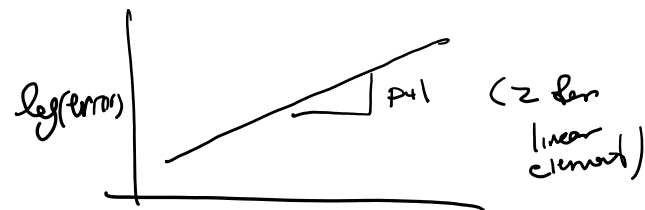
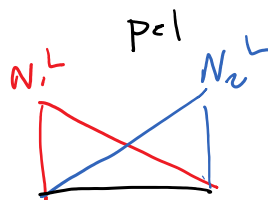
-0.24 -0.18 0.537143

F

size 3 0.110485 -1.11049 0

Higher order 1D, 2D, 3D elements

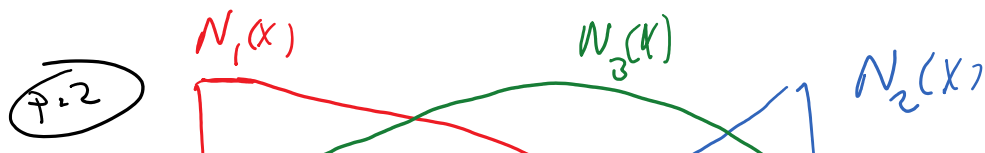
Higher order 1D elements

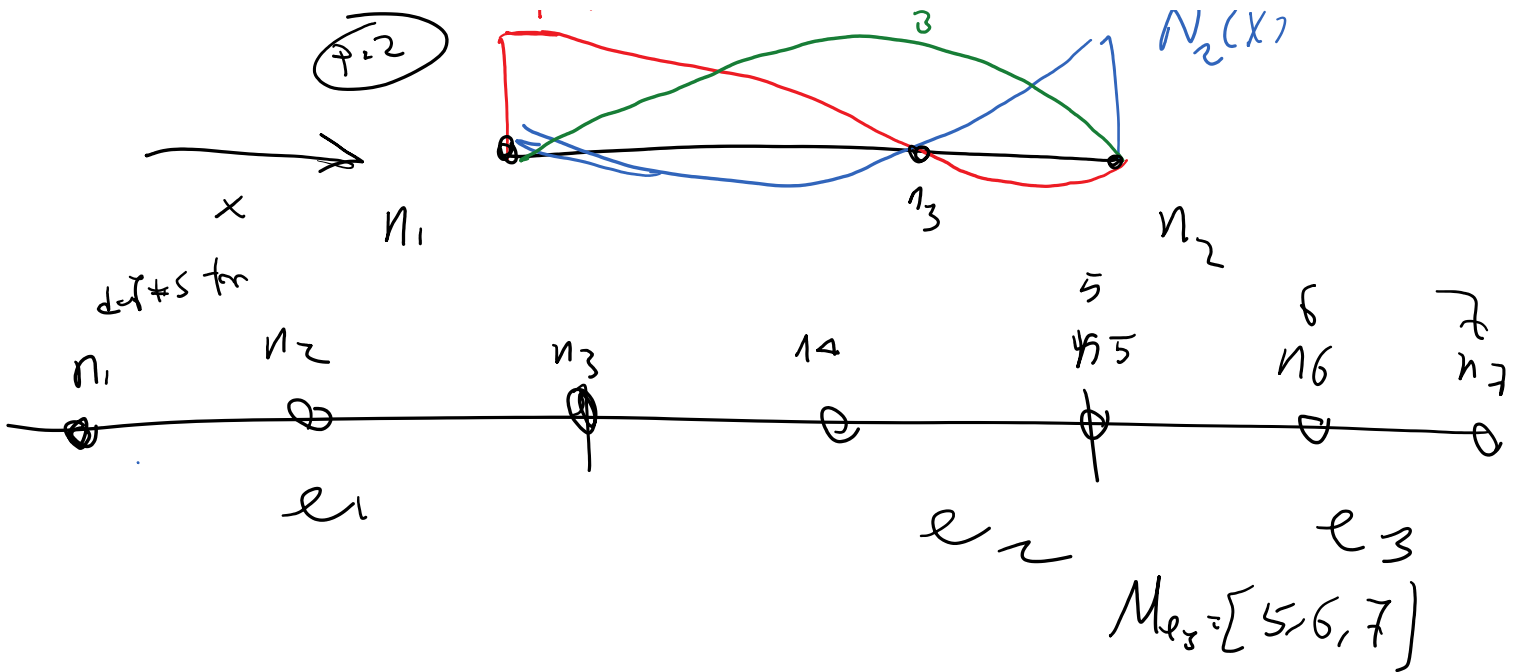


$\text{error} = Ch^{p+1}$

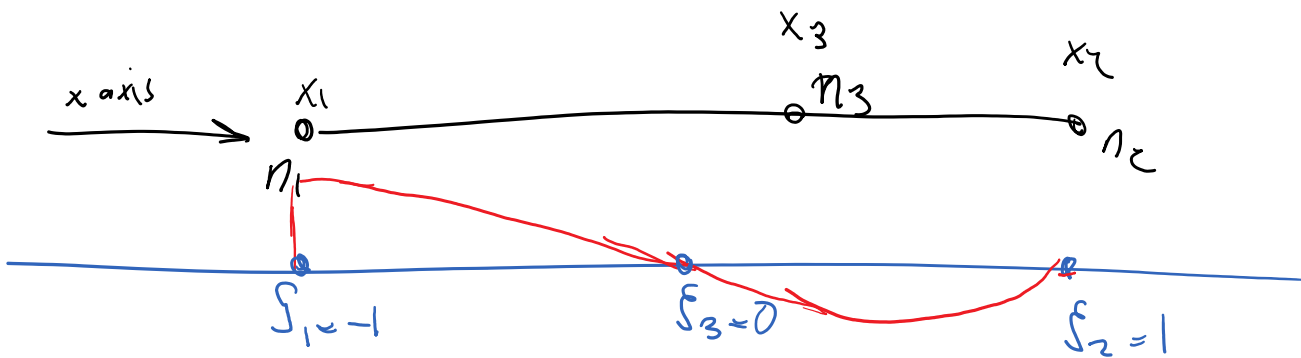
bubble shape function

1D  $p=2$  element





### Parent Geometry



$$N_1(\xi) = \frac{(\xi - \xi_2)(\xi - \xi_3)}{(\xi_1 - \xi_2)(\xi_1 - \xi_3)} \quad (L_1(\xi)) \quad \text{Lagrange polynomials}$$

$$N_1(\xi_1) = \frac{(\xi_1 - \xi_2)(\xi_1 - \xi_3)}{(\xi_1 - \xi_2)(\xi_1 - \xi_3)} = 1$$

$$N_1(\xi_2) = \frac{(\xi_2 - \xi_2)(\xi_2 - \xi_3)}{\text{denom}} = 0$$

$$N_1(\xi_3) = 0$$