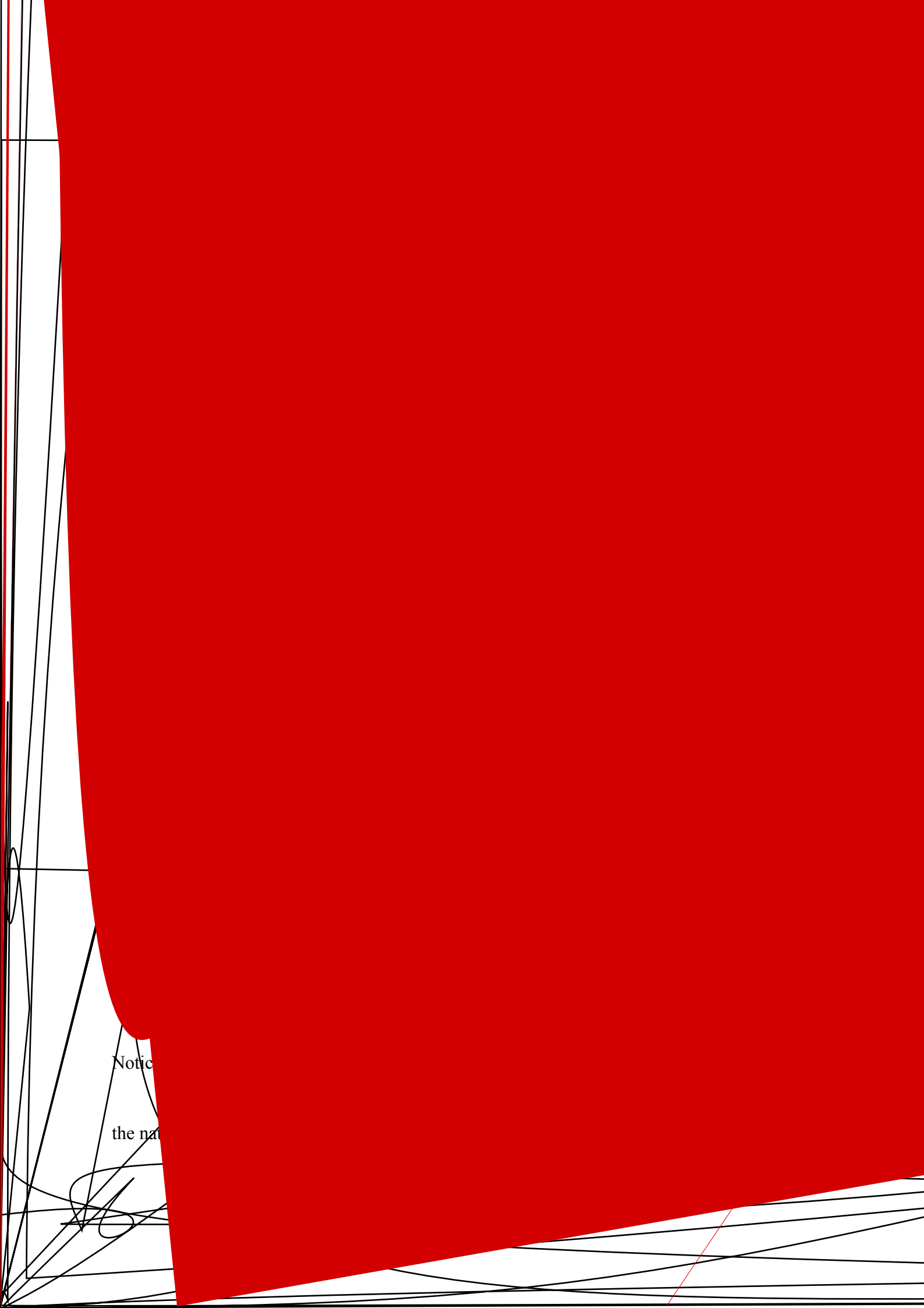

Notic
the na

Substituting (2.63b) into (2.62), the resulting $\mathbf{B}^e$ matrix becomes:

$$\mathbf{B}^e = \frac{2}{L^e}\begin{bmatrix}\frac{dN_1(\xi)}{d\xi} & \frac{dN_2(\xi)}{d\xi}\end{bmatrix} = \frac{2}{L^e}\begin{bmatrix}-\frac{1}{2} & \frac{1}{2}\end{bmatrix} = \frac{1}{L^e}\begin{bmatrix}-1 & 1\end{bmatrix} \tag{2.64}$$

The element stiffness matrix, $\mathbf{K}^e$, in the system:

$$\mathbf{K}^e = \int_{x_1^e}^{x_2^e} \mathbf{B}^{eT} EA\, \mathbf{B}^e\, dx = \int_{-1}^{1} \mathbf{B}^{eT}\, EA\, \mathbf{B}^e\, Jd\xi = \sum_{i=1}^{n_{gp}} \mathbf{B}^{eT} EA\, \mathbf{B}^e\, JW_i \tag{2.65}$$

MATLAB Codes

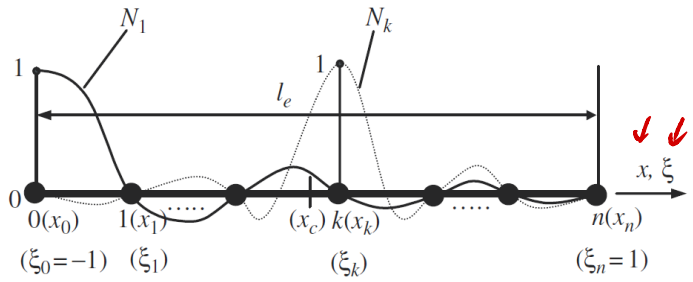
Similarly, the element nodal force vector is given by,

$$\mathbf{f}^e = \mathbf{f}_\Omega^e + \mathbf{f}_\Gamma^e = \int_{x_1^e}^{x_2^e} \mathbf{N}^{eT} b\, dx + (\mathbf{N}^{eT} A^e \bar{t})_{\Gamma_t^e}$$

$$\mathbf{f}_\Omega^e = \int_{-1}^{1} b\left(\mathbf{N}^{L2}\right)^T Jd\xi = \sum_{i=1}^{n_{gp}} b\left(\mathbf{N}^{L2}\right)^T JW_i \tag{2.66}$$

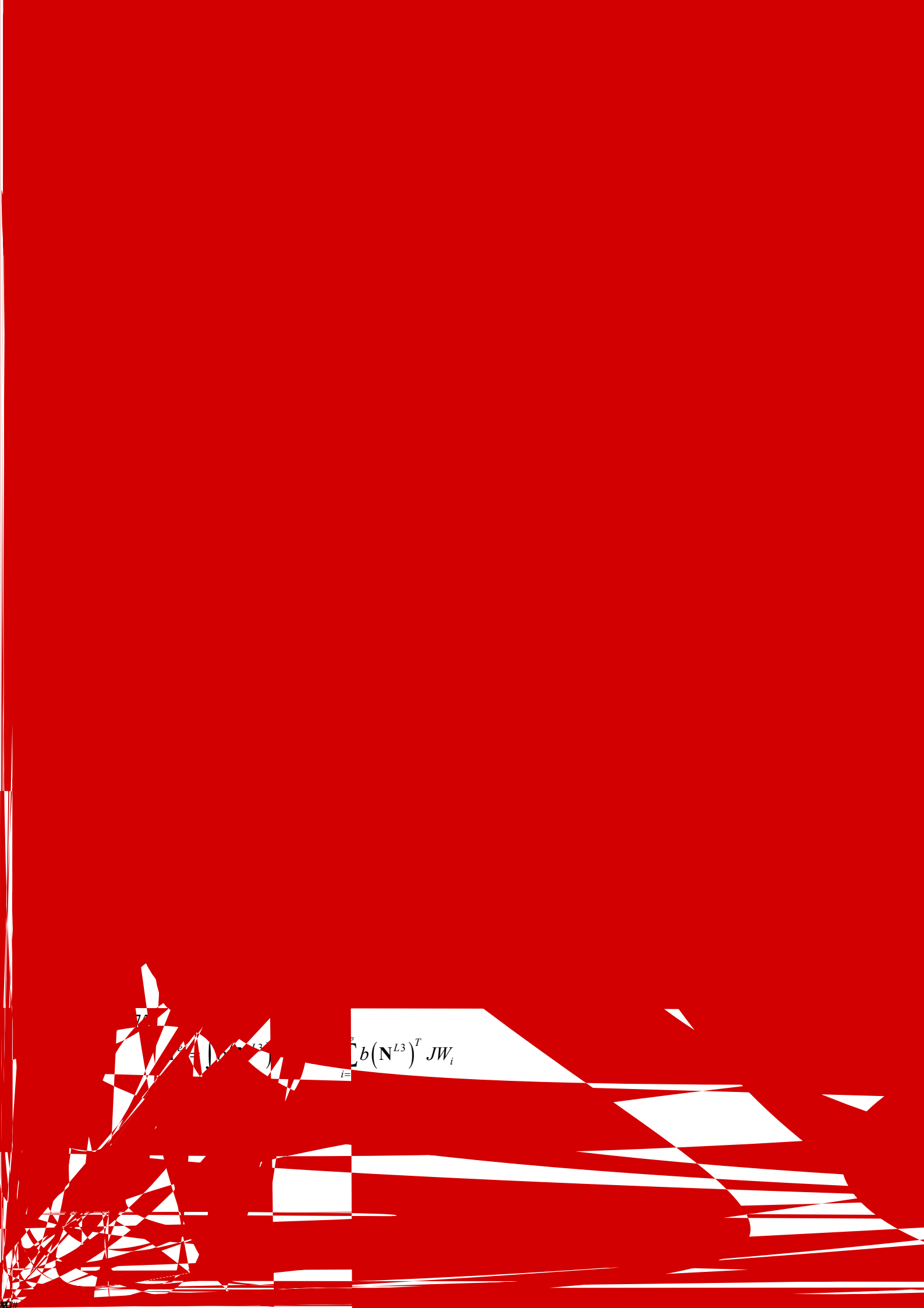
2.7.2

We can immediately the shape functions in natural coordinates to higher polynomial order. Consider a general 1D element of with **(+ 1)** defined in the range of $-1 \le \xi \le 1$ shown below:



The shape function of the element in the natural coordinates can be written in the following form using :

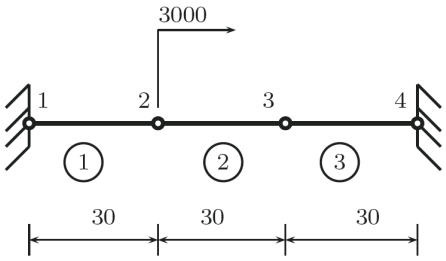
(2.68) $u^e = [N_1^L$



2. **1**

Example 1

Let us again consider the same clamped bar subjected to a point load:



The only step that differs from the previous implementation is **2**:

2: Derivation of element equations for all typical elements in the mesh. For each element, compute stiffness matrix $\mathbf{K}^e$, and force vector $\mathbf{f}^e$.

3: Assembly of element equations to obtain equations of the whole problem **Kd f**.

```
% f       c   e:
   % di  laceme   : di  laceme    ec
   % f  ce : f  ce  ec
   %   iff e  :   iff e   ma  i
f  ce= e   (  mbe N de ,1);
  iff e  = e   (  mbe N de ,  mbe N de );
% a  lied l ad a    de 2
f  ce(2)=3000.0;

% c m   a i    f  he     em   iff e   ma  i
f   e=1:  mbe Eleme   ;
  % eleme  D f: eleme   deg ee   f f eed m (D f)
 eleme  D f=eleme  N de (e,:) ;
  detJacobian=L(e)/2;
  invJacobian=1/detJacobian;
  % one Gauss point (xi=0, weight W=2)
  [shape,naturalDerivatives]=shapeFunctionL2(0.0);
  B=naturalDerivatives*invJacobian;
  stiffness(elementDof,elementDof)=...
      stiffness(elementDof,elementDof)+ B'*B*2*detJacobian*E*A;
e d
```

We use a Gauss quadrature with one Gauss point ξ 0 and W_1 2. The evaluation of the stiffness matrix involves the integral in Eq. (2.65) by

```
    iff e  (eleme  D f,eleme  D f)=...
        iff e  (eleme  D f,eleme  D f)+ B'*B*2*detJacobian*E*A;
```

where $\mathbf{B}^e$ is a matrix with the derivatives of the shape functions (Eq. (2.62)):

```
  B= a   alDe i a i e *i  Jac bia ;
```

And the shape function and derivatives with respect to a value ξ_i in

are computed in the function `ha eF  c i  L2`. The values ξ_i are the integration locations from the Gauss quadrature. For this case, we only have one integration point and its location $\xi_i = 0$.

Is one Gauss point enough for evaluating element stiffness matrix for the linear element?

$$\int B^T E A B \, d\Omega \qquad n \geq \frac{0+1}{2}$$

$p=0$ $p=0$

```
  f  c i   [ ha e, a   alDe i a i e ]= ha eF  c i  L2( i)

  %  ha e f  c i   a d de i a i e  f   L2 eleme
  %  ha e : Sha e f  c i
  %  a   alDe i a i e : de i a i e   . . .  i
  %  i:  a   al c   di a e  (-1 ... +1)

   ha e=([1- i,1+ i]/2);

   a   alDe i a i e =[-1,1]/2;

  e d % e d f  c i   ha eF  c i  L2
```

And we obtain the same results as in the direct stiffness method.

```
Di  laceme
  de    di  laceme
 1:       0.0000e+000
 2:       2.0000e-003
 3:       1.0000e-003
 4:       0.0000e+000
Reac i
```

```
  de      eac i
 1:       -2.0000e+003
 4:       -1.0000e+003
```

Example 2

We can easily extend the MATLAB codes to higher order Gauss quadrature using the following `ga  1d` MATLAB code.

```
f  c i   [ eigh  ,l ca i   ] = gauss1d( g )
%GAUSS1D   Ga      ad a   e i    e dime  i  .
%       Ge  ga      i   l ca i    i   he  a e   eleme   d mai  [-
1, 1] a d  he c   e    di g  eigh
%       [WEIGHTS, LOCATIONS] = GAUSS1D(NGP)
%
  i ch  g
   ca e 1
      l ca i    = 0;
       eigh    = 2;
   ca e 2
      l ca i    = [-0.57735027, 0.57735027];
       eigh    = [1,          1];
   ca e 3
      l ca i    = [-0.7745966692,  0.0,          0.7745966692];
       eigh   = [0.5555555556,    0.8888888889,  0.5555555556];
   ca e 4
      l ca i    = [-0.861136311594052, -0.339981043584856,
0.339981043584856, 0.861136311594052];
       eigh   = [0.347854845137454, 0.652145154862546,
0.652145154862546, 0.347854845137454];
   ca e 5
      l ca i    = [-0.906179845938664, -0.538469310105683, 0.0,
0.538469310105683, 0.906179845938664];
       eigh   = [0.236926885056189, 0.478628670499366,
0.568888888888889, 0.478628670499366, 0.236926885056189];
     he  i e
      di  ('d             m  e  ha  fi e Ga      i   ')
e d
```

We can then modify the MATLAB codes from the previous example with three Gauss points:

```
% c m   a i    f  he     em   iff e   ma  i
```

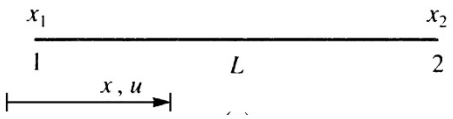
$$\mathbf{f}^e = \mathbf{f}^e_\Omega + \mathbf{f}^e_\Gamma = \int_{x_1^e}^{x_3^e} \mathbf{N}^{eT} b \, dx + (\mathbf{N}^{eT} A^e \bar{t})_{\Gamma_t^e}$$

$$\mathbf{f}^e_\Omega = \int_{-1}^{1} b\left(\mathbf{N}^{L2}\right)^T J d\xi = \sum_{i=1}^{n_{gp}} b\left(\mathbf{N}^{L2}\right)^T J W_i$$

in which

$$b(\xi) = -80000x$$
$$= -80000(x_c + J\xi)$$

after a transformation of coordinates from x to ξ with $x_c = \frac{1}{2}(x_1 + x_2)$.



What is the number of Gauss points to exactly evaluate the element force vector in this case?

2

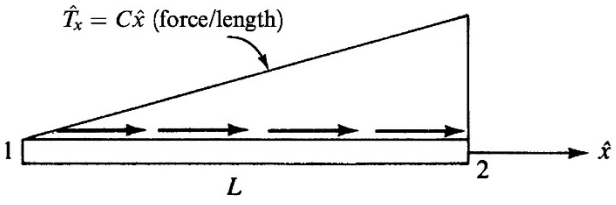
```
% I   a ame  ic F  m la i   Im leme  a i
% clea  mem
clea  all

% E: m d l    f ela  ici
% A: a ea  f c      ec i
% L: le g h  f ba
E=2e11; A=12.5e-4; L=[1.5];

%   mbe Eleme   :   mbe   f eleme
  mbe Eleme   =1;
%   mbe N de :   mbe   f   de
  mbe N de =2;
% ge e a i    f c   di a e  a d c   ec i i ie
eleme  N de =[1 2];
  deC   di a e =[0 1.5];
% f       c   e:
   % di  laceme   : di  laceme    ec
   % f  ce : f  ce  ec
   %   iff e  :   iff e   ma  i
f  ce= e   (  mbe N de ,1);
```

```
   iff e   = e    (   mbe N de ,   mbe N de );

% c m    a i     f   he      em    iff e    ma   i   a d f   ce   ec
f    e=1:   mbe Eleme    ;
  % eleme   D f: eleme    deg ee    f f eed m (D f)
 eleme   D f=eleme   N de (e,:) ;
 de Jac bia =L(e)/2;
 i   Jac bia =1/de Jac bia ;
  g   = 2;
  [ , i]=ga    1d( g );
   c=0.5*(  deC    di a e (eleme   D f(1))+   deC    di a e (eleme   D
 f(2)));
  f    i =1: g ;
      [ ha e, a    alDe i a i e ]= ha eF   c i   L2( i(i ));
      B= a    alDe i a i e *i   Jac bia ;
        iff e   (eleme   D f,eleme   D f)=...
        iff e   (eleme   D f,eleme   D f)+
B'*B* (i )*de Jac bia *E*A;
     f   ce(eleme   D f)=f   ce(eleme   D f)+...
        -80000*(xc+detJacobian*xi(ip))*shape'*detJacobian*w(ip);
  e d
e d

% b    da    c   di i     a d    l   i
%    e c ibed d f
  e c ibedD f=[2];
%    l   i
GD f=   mbe N de ;
di   laceme    =   l   i   (GD f,   e c ibedD f,   iff e   ,f   ce);
%         di   laceme    / eac i
      Di   laceme    Reac i    (di   laceme    ,   iff e   , ...
      mbe N de ,   e c ibedD f,f   ce)
```

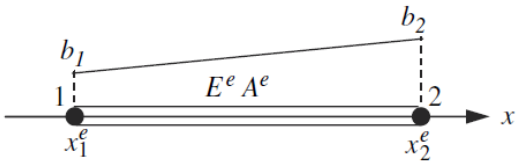
If we print out the force vector, the results are:

```
f   ce =
  1.0e+004 *
   -3.0000
   -6.0000
```

What the equivalent nodal forces for a bar subjected to a linearly distributed axial loading that varies from zero at node 1 and to a maximum at node 2:



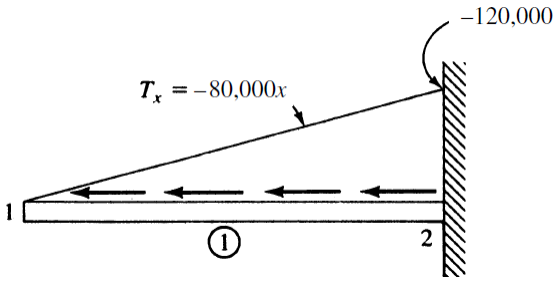
Recall what we have on the example on the linear element subjected to linear distributed body force



$$= \frac{l^e}{6}\begin{bmatrix} 2b_1 + b_2 \\ b_1 + 2b_2 \end{bmatrix}$$

The displacements and reactions for this 1D problem are:

```
Di   laceme
  de     di   laceme
 1:        -1.8000e-004
 2:        0.0000e+000
Reac i
  de      eac i
 2:        9.0000e+004
```



We can compute the reaction by hand and it is 120,000*1.5/2=90,000. Bingo!

Quadratic Element

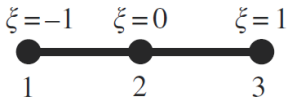
We can then build up our MATLAB element library for the quadratic element in the natural coordinates, similar to what we did for the linear element (`ha eF  c i  L2.m`):

```
f  c i   [ ha e, a   alDe i a i e ]= ha eF  c i  L3( i)

%  ha e f  c i   a d de i a i e  f   L3 eleme
%  ha e : Sha e f  c i
%  a   alDe i a i e : de i a i e   . . .  i
%  i:  a   al c   di a e  (-1 ... +1)

 ha e = [ i*( i-1), 2*(1- i* i),  i*(1+ i)]/2;

 a   alDe i a i e  = [2* i-1, -4* i, 2* i+1]/2;

e d % e d f  c i   ha eF  c i  L3
```

It can then directly replace `ha eF  c i  L2.m` to produce desirable quadratic displacement field. That is very cool!

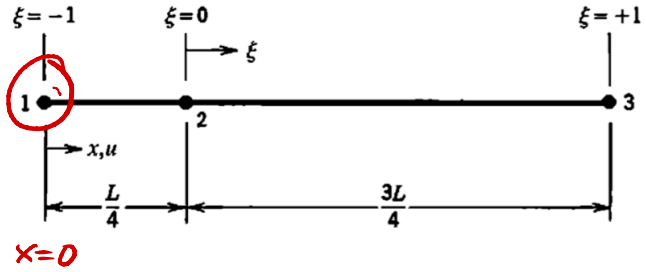
2. -

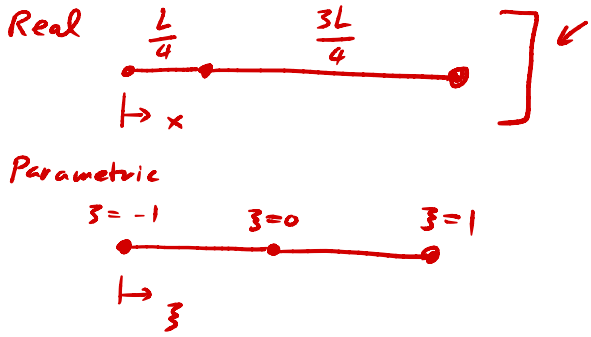
In fracture mechanics applications, one is interested in accurately capturing high stress concentration field near a crack tip. For normal applications it is best to place the side nodes at the middle points of the sides.



However, it turns out that, if the mid-side node is placed at the - , then the excessive distortion introduced in the element as a result of mapping produces a beneficial result of giving a singular stress field that is proportional to $1/\sqrt{r}$ with r measured from the corner node closest to the quarter-point node. This is precisely the behavior of stresses around a crack tip.

The singular element is less useful in 1D but to illustrate this behavior in the simplest way, we consider the three-node bar element discussed in the previous section.





: Consider a quarter-point element shown above. Work out the strain field for this element.

( )

$$\varepsilon_x = \left(\frac{d}{dx}\underline{N}^{L3}\right)\begin{bmatrix} u_1^e \\ u_2^e \\ u_3^e \end{bmatrix}$$ ↓ ltw

$$= \begin{bmatrix} \frac{2}{L} - \frac{3}{2\sqrt{Lx}} & -\frac{4}{L} + \frac{2}{\sqrt{Lx}} & \frac{2}{L} - \frac{1}{2\sqrt{Lx}} \end{bmatrix}\begin{bmatrix} u_1^e \\ u_2^e \\ u_3^e \end{bmatrix}$$

$x=0$ $\varepsilon_x \uparrow \infty$

$\sigma_x \uparrow \infty$

2.10

1.