2.7 1D Isoparametric Formulation

In the isoparametric formulation, we work our way in a so-called **natural coordinate** (in 1D) as we have done in Gauss quadrature. The natural coordinate is useful in two ways: (1) it makes the construction of the shape functions convenient and (2) it is required by Gauss quadrature which we will use to compute element stiffness matrix , and element nodal force vector if we cannot evaluate them explicitly. **This kind of coordinate mapping technique is one of the most often used techniques in the FEM.** It is extremely powerful when used for developing high order finite elements.

Recall for a two-node linear finite element, the axial displacements can be interpolated as:

(2.8)

We now wish to turn our finite element formulation in terms of natural coordinate

(2.58) $$N_1^{L2}(\xi)=\frac{1}{2}(1-\xi);\quad N_2^{L2}(\xi)=\frac{1}{2}(1+\xi)$$

Or in a matrix form:

(2.59) $$u^e=\begin{bmatrix}N_1^{L2}(\xi) & N_2^{L2}(\xi)\end{bmatrix}\begin{bmatrix}u_1^e\\ u_2^e\end{bmatrix}=\mathbf{N}^{L2}\ ^e$$

The most important aspect behind the isoparametric formulation to **use the same shape functions** to interpolate both the displacement field (or some other field variable) and **element geometry**. Thus, we have

(2.60) $$x=[N_1^{L2}(\xi)\quad N_2^{L2}(\xi)]\begin{bmatrix}x_1^e\\ x_2^e\end{bmatrix}=\mathbf{N}^{L2}\mathbf{x}^e$$

Axial strain in the element is:

(2.61) $$\varepsilon_x=\frac{du^e}{dx}=\left(\frac{d}{dx}\mathbf{N}^{L2}\right)\begin{bmatrix}u_1^e\\ u_2^e\end{bmatrix}=\mathbf{B}^e\mathbf{u}^e$$

The strain-displacement $\mathbf{B}^e$ matrix that relates the strain with the nodal displacement is:

(2.62) $$\mathbf{B}^e=\left(\frac{d}{dx}\mathbf{N}^{L2}\right)=\left(\frac{d\mathbf{N}^{L2}}{d\xi}\frac{d\xi}{dx}\right)=\left(\frac{d\mathbf{N}^{L2}}{d\xi}\frac{1}{J}\right)$$

where J is the **Jacobian** and can be obtained from (2.60):

(2.63a) $$J=\frac{dx}{d\xi}=\left(\frac{d}{d\xi}\mathbf{N}^{L2}\right)\begin{bmatrix}x_1^e\\ x_2^e\end{bmatrix}=\begin{bmatrix}-\frac{1}{2} & \frac{1}{2}\end{bmatrix}\begin{bmatrix}x_1^e\\ x_2^e\end{bmatrix}$$

Or,

(2.63b) $$J=-\frac{1}{2}x_1^e+\frac{1}{2}x_2^e=\frac{1}{2}L^e$$

The Jacobian J is a scalar and can be regarded as a scale factor that describes the physical length dx associated with a reference length $d\xi$. We see the same conclusion drawn from Section 2.6 Gauss Quadrature.

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

$$\begin{aligned}\mathbf{B}^e &= \frac{2}{L^e}\left[\frac{dN_1(\xi)}{d\xi} \quad \frac{dN_2(\xi)}{d\xi}\right] \\ &= \frac{2}{L^e}\left[-\frac{1}{2} \quad \frac{1}{2}\right] \\ &= \frac{1}{L^e}\left[-1 \quad 1\right]\end{aligned} \tag{2.64}$$

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

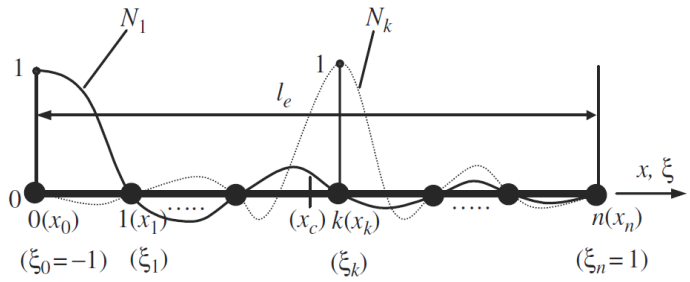
$$\mathbf{K}^e \quad \int_{x_1^e}^{x_2^e} \mathbf{B}^{eT} EA\, \mathbf{B}^e \, dx \quad \int_{-1}^{1} \mathbf{B}^{eT} \, EA\, \mathbf{B}^e \, Jd\xi \quad \sum_{i\ 1}^{n_{gp}} \mathbf{B}^{eT} EA\, \mathbf{B}^e \, JW_i \tag{2.65}$$

Similarly, the element nodal force vector is given by,

$$\begin{aligned}\mathbf{f}^e &= \mathbf{f}_\Omega^e + \mathbf{f}_\Gamma^e = \mathbf{K}^e = \int_{x_1^e}^{x_2^e} \mathbf{N}^{eT} b\, dx + (\mathbf{N}^{eT} A^e \bar{t})_{\Gamma_t^e} \\ \mathbf{f}_\Omega^e &= \int_{-1}^{1} b\left(\mathbf{N}^{L2}\right)^T \, Jd\xi = \sum_{i=1}^{n_{gp}} b\left(\mathbf{N}^{L2}\right)^T JW_i\end{aligned} \tag{2.66}$$

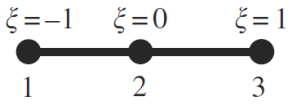
2.7.2 Higher Order Shape Functions for Natural Coordinates

We can immediately **extend** the shape functions in natural coordinates to higher polynomial order. Consider a general 1D element of **n^{th} order** with **(n + 1) nodes** 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 **Lagrange interpolants**:

(2.67) $$N_k(\xi) = \frac{(\xi-\xi_0)(\xi-\xi_1)\cdots(\xi-\xi_{k-1})(\xi-\xi_{k+1})\cdots(\xi-\xi_n)}{(\xi_k-\xi_0)(\xi_k-\xi_1)\cdots(\xi_k-\xi_{k-1})(\xi_k-\xi_{k+1})\cdots(\xi_k-\xi_n)}$$



Q: Using Eq. (2.67), write down the shape functions for the quadratic one-dimensional element with three nodes shown above:

A:

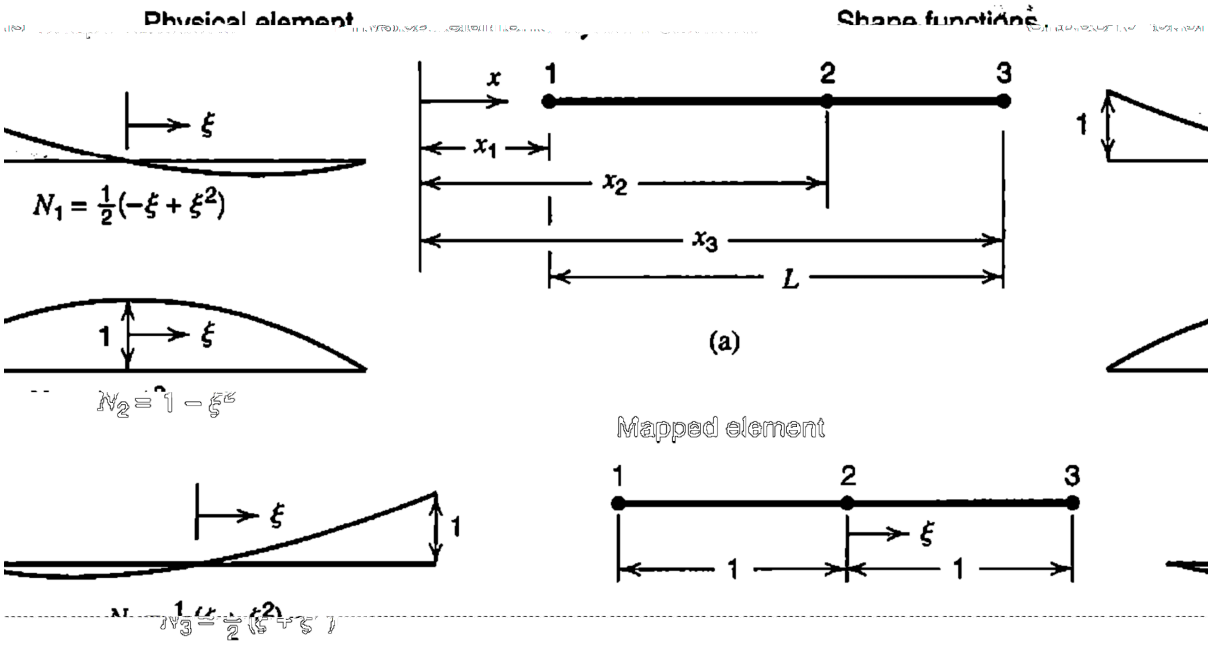
Q: Write down the derivatives of shape functions $\frac{dN_i^{L3}}{d\xi}$ for the quadratic one-dimensional element with three nodes shown above:

A:

2.7.2 Isoparametric Formulation: Quadratic Element

We now recap and highlight the isoparametric formulation and work through the formula for a 1D three-node quadratic element. Although the 1D element does not show the practical value of the isoparametric formulation, but it illustrates its manipulations in a simple context. We will consider the isoparametric formulation in depth in Chapter 4 for 2D elements.

The isoparametric formulation **uses the same shape functions** to interpolate both the displacement field (or some other field variable) and element geometry.



(2.68) $$u^e = [N_1^{L3}(\xi) \quad N_2^{L3}(\xi) \quad N_3^{L3}(\xi)] \begin{bmatrix} u_1^e \\ u_2^e \\ u_3^e \end{bmatrix} = \mathbf{N}^{L3}\mathbf{u}^e$$

(2.69) $$x = [N_1^{L3}(\xi) \quad N_2^{L3}(\xi) \quad N_3^{L3}(\xi)] \begin{bmatrix} x_1^e \\ x_2^e \\ x_3^e \end{bmatrix} = \mathbf{N}^{L3}\mathbf{x}^e$$

That is, displacement of a point within an element can be expressed in terms of nodal d.o.f. and shape functions $\mathbf{N}^{L3}$, which are functions of natural coordinates. Similarly, the global position (coordinates) of a point within the element can be expressed in terms of global nodal positions and shape functions $\mathbf{N}^{L3}$, which are also functions of natural coordinates.

Axial strain in the element is:

(2.70) $$\varepsilon_x = \frac{du^e}{dx} = \left(\frac{d}{dx}\mathbf{N}^{L3}\right) \begin{bmatrix} u_1^e \\ u_2^e \\ u_3^e \end{bmatrix} = \mathbf{B}^e\mathbf{u}^e$$

The strain-displacement $\mathbf{B}^e$ matrix that relates the strain with the nodal displacement is (cf: Eq. (2.62)):

(2.71) $$\mathbf{B}^e = \left(\frac{d}{dx}\mathbf{N}^{L3}\right) = \left(\frac{d\mathbf{N}^{L3}}{d\xi}\frac{d\xi}{dx}\right) = \left(\frac{d\mathbf{N}^{L3}}{d\xi}\frac{1}{J}\right)$$

where *J* is the **Jacobian** and can be obtained from (2.69):

(2.72a) $J = \frac{dx}{d\xi} = \left(\frac{d}{d\xi}\mathbf{N}^{L3}\right)\begin{bmatrix} x_1^e \\ x_2^e \\ x_3^e \end{bmatrix} = \begin{bmatrix} \frac{1}{2}(-1+2\xi) & -2\xi & \frac{1}{2}(1+2\xi) \end{bmatrix}\begin{bmatrix} x_1^e \\ x_2^e \\ x_3^e \end{bmatrix}$

Or,

(2.72b) $J = \frac{1}{2}(-1+2\xi)x_1^e - 2\xi x_2^e + \frac{1}{2}(1+2\xi)x_3^e$

If the mid-node happens in the middle of the physical coordinate $x_2^e = \frac{x_1^e + x_3^e}{2}$ (**Note**: in Section 2.9, we will play a trick to move the mid-node to the 1/4 position)

(2.72c) $J = \frac{1}{2}(-1+2\xi)x_1^e - 2\xi\frac{x_1^e+x_3^e}{2} + \frac{1}{2}(1+2\xi)x_3^e = \frac{1}{2}(x_3^e - x_1^e) = \frac{1}{2}L^e$

The Jacobian *J* is a scalar and can be regarded as a scale factor that describes the physical length dx associated with a reference length $d\xi$.

We can now rewrite the strain-displacement $\mathbf{B}^e$ matrix (2.71) as:

(2.73) $$\mathbf{B}^e = \frac{d\mathbf{N}^e}{dx} = \frac{d\mathbf{N}^{L3}}{d\xi}\frac{d\xi}{dx} = \frac{d\mathbf{N}^{L3}}{d\xi}\frac{1}{J} = \frac{1}{J}\left[\frac{1}{2}(-1+2\xi) \quad -2\xi \quad \frac{1}{2}(1+2\xi)\right]$$

The element stiffness matrix (cf: (2.65)):

(2.74) $$\mathbf{K}^e = \int_{x_1^e}^{x_3^e} \mathbf{B}^{eT} EA\, \mathbf{B}^e \, dx = \int_{-1}^{1} \mathbf{B}^{eT}\, EA\, \mathbf{B}^e \, Jd\xi = \sum_{i=1}^{n_{gp}} \mathbf{B}^{eT} EA\, \mathbf{B}^e \, JW_i$$

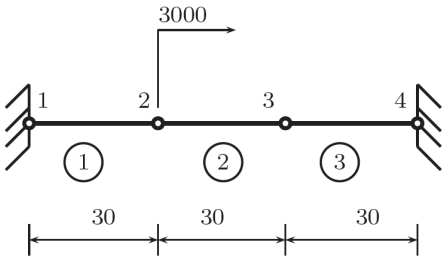
Similarly, the element nodal force vector (cf: (2.66)):

(2.75)
$$\mathbf{f}^e = \mathbf{f}_\Omega^e + \mathbf{f}_\Gamma^e = \int_{x_1^e}^{x_3^e} \mathbf{N}^{eT} b\, dx + (\mathbf{N}^{eT} A^e \bar{t})_{\Gamma_t^e}$$
$$\mathbf{f}_\Omega^e = \int_{-1}^{1} b\left(\mathbf{N}^{L3}\right)^T \, Jd\xi = \sum_{i=1}^{n_{gp}} b\left(\mathbf{N}^{L3}\right)^T \, JW_i$$

2.8 MATLAB Implementation for 1D Isoparametric Formulation

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 **step 2**:

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

Step 3: Assembly of element equations to obtain equations of the whole problem $\mathbf{Kd} = \mathbf{f}$.

```
% for structure:
   % displacements: displacement vector
   % force : force vector
   % stiffness: stiffness matrix
force=zeros(numberNodes,1);
stiffness=zeros(numberNodes,numberNodes);
% applied load at node 2
force(2)=3000.0;

% computation of the system stiffness matrix
for e=1:numberElements;
  % elementDof: element degrees of freedom (Dof)
  elementDof=elementNodes(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;
end
```

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

```
stiffness(elementDof,elementDof)=...
    stiffness(elementDof,elementDof)+ B'*B*2*detJacobian*E*A;
```

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

```
B=naturalDerivatives*invJacobian;
```

And the shape function and derivatives with respect to a value ξ_i in **natural coordinates** are computed in the function `shapeFunctionL2`. The values ξ_i are the integration locations from the Gauss quadrature. For this case, we only have one integration point and its location ξ_i 0.

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

```
function [shape,naturalDerivatives]=shapeFunctionL2(xi)

% shape function and derivatives for L2 elements
% shape : Shape functions
% naturalDerivatives: derivatives w.r.t. xi
% xi: natural coordinates (-1 ... +1)

shape=([1-xi,1+xi]/2);

naturalDerivatives=[-1,1]/2;

end % end function shapeFunctionL2
```

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

```
Displacements
node    displacements
 1:       0.0000e+000
 2:       2.0000e-003
 3:       1.0000e-003
 4:       0.0000e+000
Reactions
```

```
node    reactions
 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 `gauss1d` MATLAB code.

```
function [weights,locations] = gauss1d(ngp)
%GAUSS1D   Gauss quadrature in one dimension.
%       Get gauss point locations in the parent element domain [-
1, 1] and the corresponding weights
%       [WEIGHTS, LOCATIONS] = GAUSS1D(NGP)
%
switch ngp
   case 1
      locations = 0;
      weights  = 2;
   case 2
      locations = [-0.57735027, 0.57735027];
      weights  = [1,         1];
   case 3
      locations = [-0.7745966692,  0.0,         0.7745966692];
      weights = [0.5555555556,    0.8888888889,  0.5555555556];
   case 4
      locations = [-0.861136311594052, -0.339981043584856,
0.339981043584856, 0.861136311594052];
      weights = [0.347854845137454, 0.652145154862546,
0.652145154862546, 0.347854845137454];
   case 5
      locations = [-0.906179845938664, -0.538469310105683, 0.0,
0.538469310105683, 0.906179845938664];
      weights = [0.236926885056189, 0.478628670499366,
0.568888888888889, 0.478628670499366, 0.236926885056189];
   otherwise
      disp('do not support more than five Gauss points')
end
```

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

```
% computation of the system stiffness matrix
for e=1:numberElements;
  % elementDof: element degrees of freedom (Dof)
  elementDof=elementNodes(e,:) ;
  detJacobian=L(e)/2;
  invJacobian=1/detJacobian;
  ngp = 3;
  [w,xi]=gauss1d(ngp);
  for ip=1:ngp;
     [shape,naturalDerivatives]=shapeFunctionL2(xi(ip));
     B=naturalDerivatives*invJacobian;
```

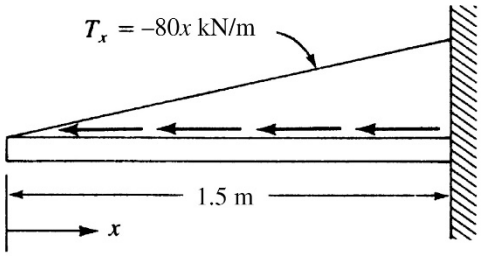
```
      stiffness(elementDof,elementDof)=...
      stiffness(elementDof,elementDof)+
B'*B*w(ip)*detJacobian*E*A;
  end
end
```

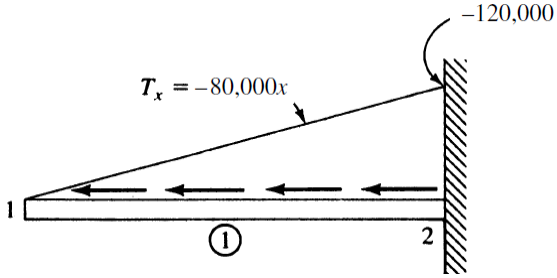
We will of course get the same results.

```
Displacements
node     displacements
 1:        0.0000e+000
 2:        2.0000e-003
 3:        1.0000e-003
 4:        0.0000e+000
Reactions
node     reactions
 1:        -2.0000e+003
 4:        -1.0000e+003
```

Example 3

Let us now consider a bar subjected to a triangular load distribution shown below. Let $E = 2 \times 10^5$ Pa, $A = 12.5$ cm^2 and $L = 1.5$ m. Determine the axial displacements using one linear element.





We will pay special attention on how to incorporate element force vector from Eq. (2.66) into the MATLAB codes. Recall

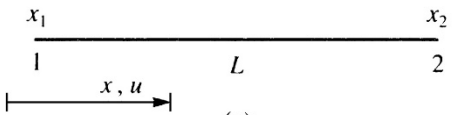
$$\mathbf{f}^e = \mathbf{f}^e_\Omega + \mathbf{f}^e_\Gamma = \int_{x_1^e}^{x_3^e} \mathbf{N}^{eT} b \, dx + (\mathbf{N}^{eT} A^e \bar{t})_{\Gamma_t^e}$$

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

in which

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

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



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

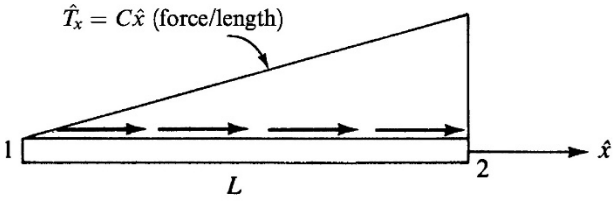
```
% Isoparametric Formulation Implementation
% clear memory
clear all

% E: modulus of elasticity
% A: area of cross section
% L: length of bar
E=2e11; A=12.5e-4; L=[1.5];

% numberElements: number of elements
numberElements=1;
% numberNodes: number of nodes
numberNodes=2;
% generation of coordinates and connectivities
elementNodes=[1 2];
nodeCoordinates=[0 1.5];
% for structure:
   % displacements: displacement vector
   % force : force vector
   % stiffness: stiffness matrix
force=zeros(numberNodes,1);
```

```
stiffness=zeros(numberNodes,numberNodes);

% computation of the system stiffness matrix and force vector
for e=1:numberElements;
  % elementDof: element degrees of freedom (Dof)
  elementDof=elementNodes(e,:) ;
  detJacobian=L(e)/2;
  invJacobian=1/detJacobian;
  ngp = 2;
  [w,xi]=gauss1d(ngp);
  xc=0.5*(nodeCoordinates(elementDof(1))+nodeCoordinates(elementD
of(2)));
  for ip=1:ngp;
      [shape,naturalDerivatives]=shapeFunctionL2(xi(ip));
      B=naturalDerivatives*invJacobian;
      stiffness(elementDof,elementDof)=...
      stiffness(elementDof,elementDof)+
B'*B*w(ip)*detJacobian*E*A;
      force(elementDof)=force(elementDof)+...
          -80000*(xc+detJacobian*xi(ip))*shape'*detJacobian*w(ip);
  end
end

% boundary conditions and solution
% prescribed dofs
prescribedDof=[2];
% solution
GDof=numberNodes;
displacements=solution(GDof,prescribedDof,stiffness,force);
% output displacements/reactions
outputDisplacementsReactions(displacements,stiffness, ...
    numberNodes,prescribedDof,force)
```

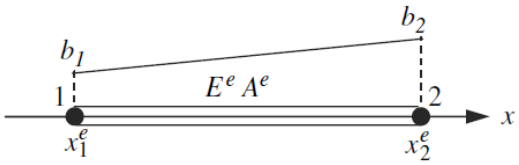
If we print out the force vector, the results are:

```
force =
  1.0e+004 *
   -3.0000
   -6.0000
```

Q: 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

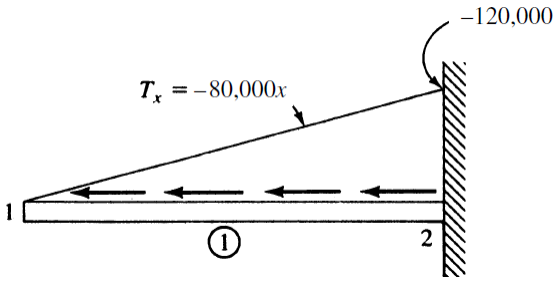


$$= \frac{l^e}{6}\begin{bmatrix} 2b_1 + b_2 \\ b_1 + 2b_2 \end{bmatrix}$$

A:

The displacements and reactions for this 1D problem are:

```
Displacements
node     displacements
 1:       -1.8000e-004
 2:       0.0000e+000
Reactions
node     reactions
 2:       9.0000e+004
```



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

Quadratic Element

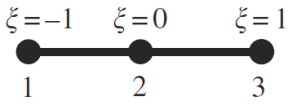
We can then build up our MATLAB element library for the quadratic element in the natural coordinates, similar to what we did for the linear element (`shapeFunctionL2.m`):

```
function [shape,naturalDerivatives]=shapeFunctionL3(xi)

% shape function and derivatives for L3 elements
% shape : Shape functions
% naturalDerivatives: derivatives w.r.t. xi
% xi: natural coordinates (-1 ... +1)

shape = [xi*(xi-1), 2*(1-xi*xi), xi*(1+xi)]/2;

naturalDerivatives = [2*xi-1, -4*xi, 2*xi+1]/2;

end % end function shapeFunctionL3
```

It can then directly replace `shapeFunctionL2.m` to produce desirable quadratic displacement field. That is very cool!

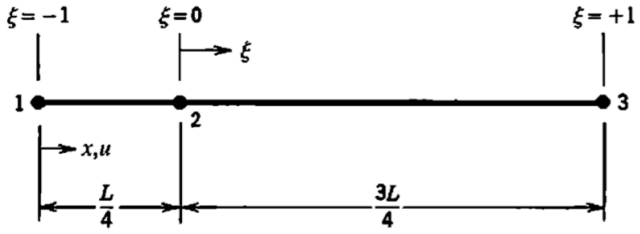
2.9 Mapping Trick: Quarter-Point Singular Elements

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 **quarter-point**, 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.



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

(Answer)

2.10 Chapter Summary

1. The finite element method treats a given domain as a collection of simple subdomains, called **finite elements**. Since the subdomain is relatively simple, it is possible to systematically construct the approximation functions needed in the weighted-residual or variational methods over **each element**.
2. Finite element method uses global interpolation functions constructed from element shape functions $\mathbf{N} = \sum_{e=1}^{n_{el}} \mathbf{N}^e(x)\mathbf{L}^e$ and unknown nodal values to construct the approximation solution. This differs from the classical way in which polynomial basis with unknown coefficients is used to construct the approximation solution.
3. These types of finite elements we develop in this course are known as C^0 elements, in which the superscript indicates that only derivative of zeroth order (i.e., $u^e(x)$) is continuous at the nodes connecting different elements.
4. We can construct shape functions for C^0 elements through Lagrange interpolation functions.
5. Finite element method uses Gauss quadrature for numerical integration. The number of integration points needed to integrate a polynomial of order *p* exactly is $n_{gp} \geq \frac{p+1}{2}$.
6. The isoparametric formulation **is one of the most often used techniques in the FEM.** It allows us to systematically develop high order finite elements in 2D and 3D.

It is VERY important for you to **link** the variables described in the MATLAB codes with the notations established in Section 2.3 and Section 2.7 (and other sections) to reinforce your understanding of finite element formulation and programming in general.