Lecture no. 7

Finite Element Method

- Define the approximating functions locally over “finite elements”

Advantages

- It’s much easier to satisfy the b.c.’s with local functions over local parts of the boundary than it is with global functions over the entire boundary.
- Splitting the domain into intervals and using lower order approximations within each element will cause the integral error to assure better accuracy on a pointwise basis. (Courant, 1920).

  An integral norm attempts to minimize total error over the entire domain. A low integral norm does not always mean that we have good pointwise error norm.
- The properties of the matrices will generally be much better than when using globally defined trial functions.

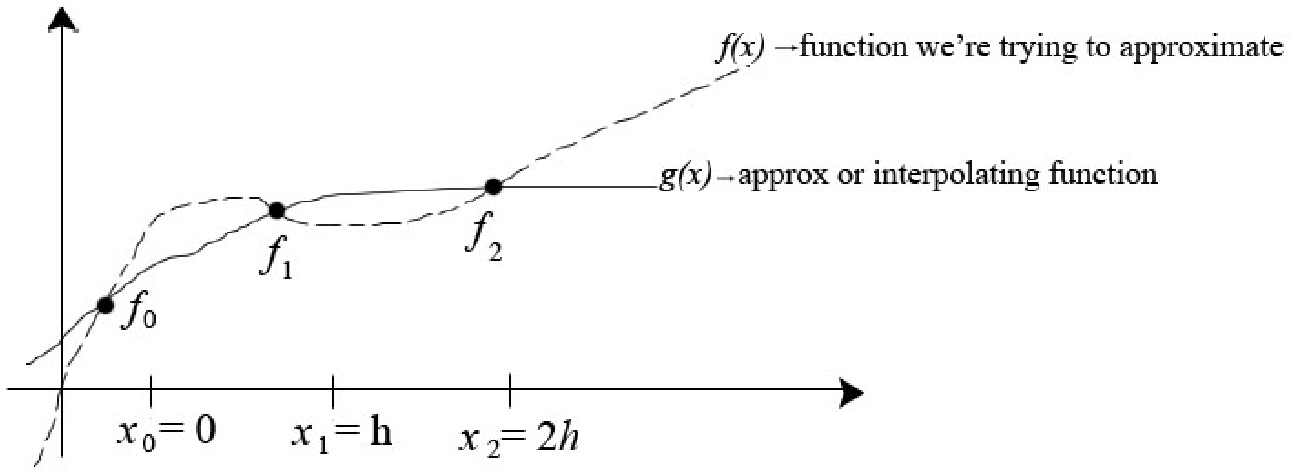
General Steps to FEM

- Divide the domain into N sub-intervals or finite elements.
- Develop interpolating functions valid over each element. We will make use of localized coordinate systems to assure functional universality (i.e. they will be applicable to any length element at any location).

  We will tailor these functions such that the required degree of functional continuity can be readily enforced.
- Enforce functional continuity – 2 options
  - Through definition of Cardinal basis
  - Through "global" matrix assembly
- Note that we use these interpolating functions in conjunction with the implementation of the desired weighted residual form (i.e. integrations etc.) as before.

<u>Lagrange Interpolation</u>

<u>Lagrange Interpolation</u> : pass an approximating function, $g(x)$, exactly through the functional values at a set of interpolation points or nodes.


$f(x)$ →function we're trying to approximate
$g(x)$→approx or interpolating function
f_0
f_1
f_2
$x_0 = 0$
$x_1 = h$
$x_2 = 2h$

- Method 1 to deriving $g(x)$ – Power series:
  - We are given f_0, f_1, f_2 and corresponding points x_0, x_1, x_2
  - Constraints we can apply

    $g(x_0 = 0) = f_0$

    $g(x_1 = h) = f_1$

    $g(x_2 = 2h) = f_2$

    3 Constraints $\Rightarrow$ 3 d.o.f/3 nodes = 1 d.o.f/node

    $\Rightarrow$ Polynomial form $g(x)$ can have 3 d.o.f $\Rightarrow$ quadratic
  - General form of $g(x)$

    $g(x) = a + bx + cx^2$
  - Apply constraints

    $g(x_0 = 0) = f_0 \quad \Rightarrow \quad a = f_0$

    $g(x_1 = h) = f_1 \quad \Rightarrow \quad a + bh + ch^2 = f_1$

    $g(x_2 = 2h) = f_2 \quad \Rightarrow \quad a + 2hb + 4h^2c = f_2$

- Solve a system of simultaneous equations.

$$\begin{bmatrix} 1 & 0 & 0 \\ 1 & 1 & 1 \\ 1 & 2 & 4 \end{bmatrix} \begin{bmatrix} a \\ bh \\ ch^2 \end{bmatrix} = \begin{bmatrix} f_0 \\ f_1 \\ f_2 \end{bmatrix} \qquad \Rightarrow$$

$$\begin{bmatrix} a \\ bh \\ ch^2 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ -1.5 & 2 & -0.5 \\ 0.5 & -1 & 0.5 \end{bmatrix} \begin{bmatrix} f_0 \\ f_1 \\ f_2 \end{bmatrix} \qquad \Rightarrow$$

$$a = f_0$$

$$b = \frac{1}{h}\left(-\frac{3}{2} f_0 + 2f_1 - \frac{1}{2} f_2\right)$$

$$c = \frac{1}{h^2}\left(\frac{1}{2} f_0 - f_1 + \frac{1}{2} f_2\right)$$

$$\therefore g(x) = f_0 + \frac{1}{2h}(-3f_0 + 4f_1 - f_2)x + \frac{1}{2h^2}(f_0 - 2f_1 + f_2)x^2$$

Interpolating function throughout interval of interest $[0,2h]$

- Let's rewrite $g(x)$

  Factor out of f_0, f_1 and f_2

$$g(x) = f_0 \underbrace{\left(1 - \frac{3x}{2h}x + \frac{1}{2h^2}x^2\right)}_{\equiv \phi_0(x)} + f_1 \underbrace{\left(\frac{4}{2h}x - \frac{1}{h^2}x^2\right)}_{\equiv \phi_1(x)} + f_2 \underbrace{\left(-\frac{1}{2h}x + \frac{1}{2h^2}x^2\right)}_{\equiv \phi_2(x)}$$

$\Rightarrow$

- $g(x) = \sum_{i=0}^{2} f_i \underbrace{\phi_i(x)}$

$\phi_i(x)$ are the Interpolating basis functions!!

Each is associated with one node.

Looks very much like expansions we used for w.r. methods!!!

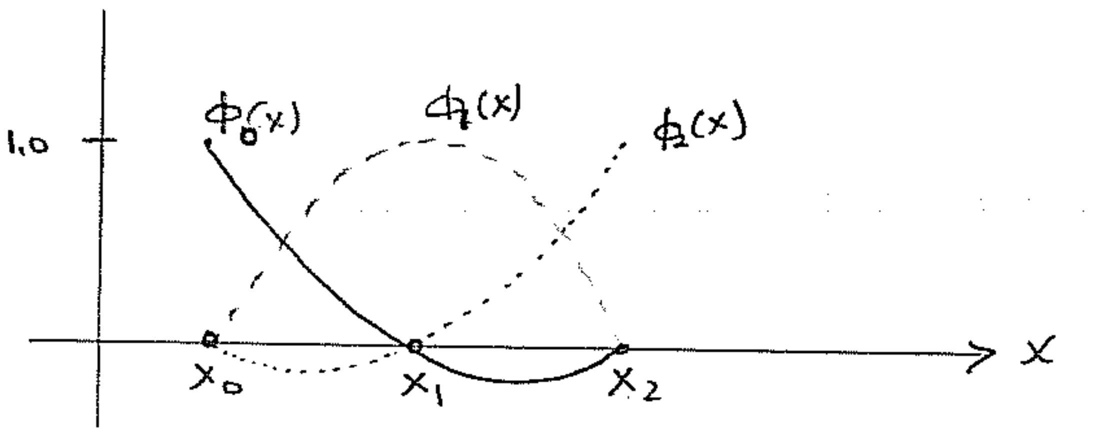
- Let's plot these functions

$\phi_0(x_0 = 0) = 1$	$\phi_1(x_0 = 0) = 0$	$\phi_2(x_0 = 0) = 0$
$\phi_0(x_1 = h) = 0$	$\phi_1(x_1 = h) = 1$	$\phi_2(x_1 = h) = 0$
$\phi_0(x_2 = 2h) = 0$	$\phi_1(x_2 = 2h) = 0$	$\phi_2(x_2 = 2h) = 1$

- Thus $\phi_i(x_j) = \delta_{ij} = \begin{matrix} 1 & i = j \\ 0 & i \neq j \end{matrix}$

  We can in fact use these constraints to derive ϕ_0, ϕ_1 and ϕ_2

- Thus interpolating functions are 1 at nodes they are associated with and 0 at all other nodes. At *x* values <u>other</u> than nodal values these functions vary and do not equal zero



- A very important consequence of using Lagrange interpolation is that

$$g(x_i) = f_i$$

This property and the fact that we define nodes on inter-element boundaries will enable us to easily enforce functional continuity on inter-element boundaries

<u>Applying Lagrange Interpolation to develop u_{app}</u>

<u>Option 1</u> – Develop a higher order approximation which is ***global*** and based on Lagrange basis function defined over the entire domain.

$$u_{app} = \sum_{i=1}^{N} u_i \phi_i$$

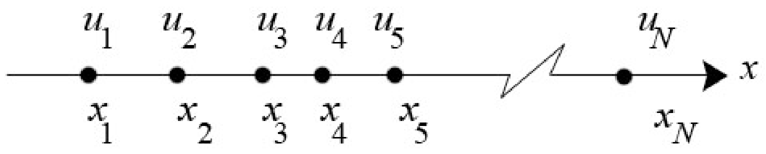
where

ϕ_i = globally defined Lagrange basis functions valid over the entire domain

u_i = the expansion coefficients and by definition these equal the function values at the nodes!!

$i = 1, N$ are the N "nodes" defined throughout the global domain

These nodes can be equispaced or nonequispaced



- Boundary conditions can be readily incorporated into the expansion <u>if</u> they are function specified (essential type boundary conditions)
  - For example $u(x_L) = u_L$

    $u(x_R) = u_R$
  - The expansion can now be written as

$$u_{app} = u_L\phi_1 + u_R\phi_N + \sum_{i=2}^{N-1} u_i\phi_i$$

(since $u_1 = u_L$ and $u_N = u_R$)

  - Thus

$$u_B = u_L\phi_1 + u_R\phi_N$$

  - We note that u_B satisfies admissibility conditions

$$u_B(x_L) = u_L\,\underbrace{\phi_1(x_L)}_{=1} + u_R\,\underbrace{\phi_N(x_L)}_{=0} = u_L$$

$$u_B(u_R) = u_L\,\underbrace{\phi_1(x_R)}_{=0} + u_R\,\underbrace{\phi_N(x_R)}_{=1} = u_R$$

- Also we note that the remaining $\phi_i \quad i = 2, N-1$ satisfy

$$\left.\begin{array}{l}\phi_i(x_L) = 0 \\ \\ \phi_i(x_R) = 0\end{array}\right. \quad \rightarrow \; i = 2, N-1$$

- Thus satisfying function specified b.c.'s (essential) for 1-D is very easy!! For natural b.c.'s it is much more difficult.
- The sequence of functions ϕ_i are linearly independent
- The coefficients in the expansion are now actually equal to the values of the function at the nodes!!
- The drawback of *option 1* is that you obtain poor pointwise convergence as *N* becomes large for most problems.
- Another drawback is that the matrix will also be almost fully populated and will be poorly conditioned
- Typically the technique does work well for slowly varying smooth solutions.

Example

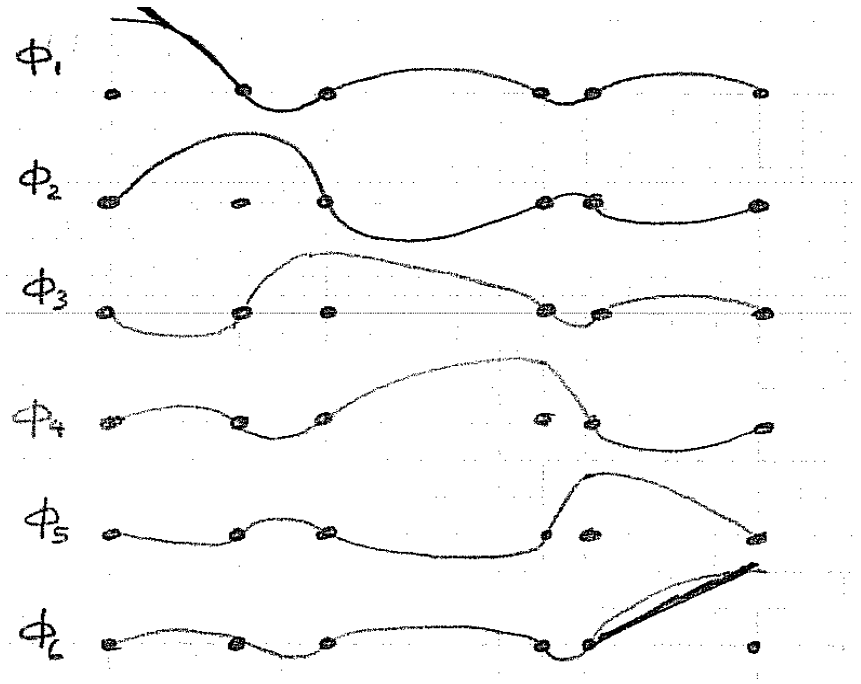
Solve $L(u) = p(x) \qquad x \in [x_L, x_R]$

b.c.'s $u(x_L) = u_L$

$u(x_R) = u_R$

Assume that we will apply a six term expansion

$$u_{app} = u_L\phi_1 + u_R\phi_6 + \sum_{i=2}^{5} u_i\phi_i$$



- Note that $u_B = u_L\phi_1 + u_R\phi_6$
  - u_B satisfies b.c.'s as specified
  - Note that ϕ_2, ϕ_3, ϕ_4 and ϕ_5 all satisfy the homogeneous form of the essential function specified b.c.'s

Option 2 – Develop an approximation u_{app} which is the sum of localized approximations

$$u_{app} = \sum_{j=1}^{M} \sum_{i=1}^{N_j} u_i^j \phi_i^j$$

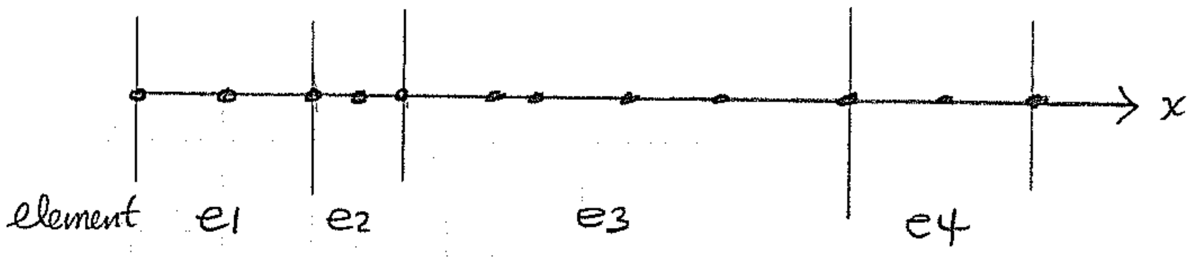
Where u_i^j = expansion coefficient for element *j* and node *i* within element *j*

$\phi_i^j =$ Lagrange basis function for element *j* and node *i*

Note that $\phi_i^j \equiv 0$ outside of element j

$j = 1, M =$ total number of localized domains or “finite elements”

$i = 1, N_j =$ total number of nodes within element *j*



Local unknowns at nodes

$$u_1^1 \quad u_3^1 \quad u_3^1 \qquad u_1^3 \quad u_2^3 \; u_3^3 \; u_4^3 \qquad u_5^3 \quad u_6^3$$

$$u_1^2 \; u_2^2 \; u_3^2 \qquad\qquad u_1^4 \quad u_2^4 \quad u_3^4 \begin{array}{l} \leftarrow \textit{element index} \\ \leftarrow \textit{local node index} \end{array}$$

- The boundary conditions can again be readily built into 1-D type problems for function specified conditions (essential). Even in multiple dimensions, function specified b.c.'s (essential) can be incorporated in a straight forward manner
- The sequence of all functions will be linearly independent.
- Again note that those coefficients of expansion are equal to the actual function values at the nodes

- To ensure C_0 inter-element functional continuity we must have at all inter-element boundaries:

  u_{app}(left of an inter-element boundary) $=$ u_{app} (right of an inter-element boundary)

  Thus in the example given

  <u>Anywhere in element 1</u>

$$u^1(\xi) = u_1^1\phi_1^1(\xi) + u_2^1\phi_2^1(\xi) + u_3^1\phi_3^1(\xi)$$

Note that all other Lagrange basis function from other elements are defined as zero.

<u>Anywhere in element 2</u>

$$u^2(\xi) = u_1^2\phi_1^2(\xi) + u_2^2\phi_2^2(\xi) + u_3^1\phi_3^2(\xi)$$

Element 1 evaluated at r.h.s. of the element

$$u^1(\xi_3^1) = u_3^1$$

Element 2 evaluated at l.h.s. of the element

$$u^2(\xi_1^2) = u_1^2$$

In order to enforce c_0 functional continuity we must have

$$u^1(\xi_3^1) = u^2(\xi_1^2)$$

$$\therefore \quad u_3^1 = u_1^2$$

- In general we must have the expansion coefficients equal at inter-element boundaries in order to satisfy c_0 functional continuity requirements
- In our example, the expansion coefficients are related as

$$u_3^1 = u_1^2$$

$$u_3^2 = u_1^3$$

$$u_6^3 = u_1^4$$

  ∴ We must have expansion coef.'s at inter-element boundaries be equal

- There are several approaches to implement the inter-element functional continuity (i.e. setting equal the adjacent inter-element expansion coef.'s)
  - Develop "Cardinal" basis functions which are formed by patching together the localized Lagrange functions and defining them globally. This also implies that you redefine the expansion coef.'s globally (so that there will now be only one coef. per global node)
  - Actually implement all expansions locally. Then take care of inter-element functional continuity by assembling the "global" matrix correctly

Notes

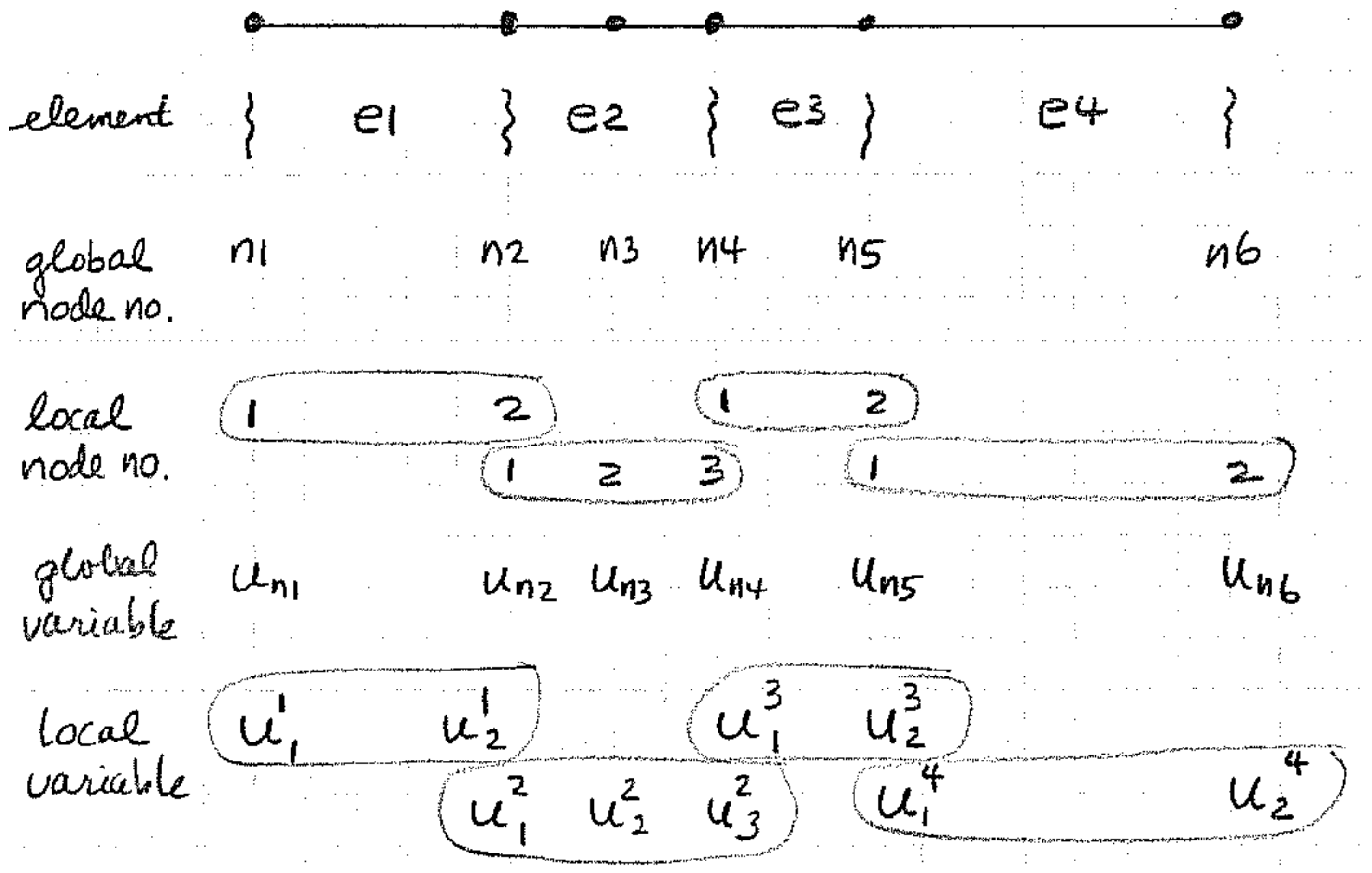
- Boundary condition implementation/satisfaction as well as inter-element functional continuity enforcement are made simple and possible by the use of Lagrange basis functions. However we must place nodes at the ends of the domain as well as at the ends of each element for this to work.
- The advantages of defining local functions are:
  - Essential function specified b.c.’s are very easy to implement in 2-D and 3-D as well as 1-D
  - Excellent pointwise as well as integral norm convergence
  - Matrices are sparse with a limited number of non-zero entries per row and the conditioning of the matrix will be better

- Solve $L(u) = p(x) \quad [x_L, x_R]$

  $u(x_L) = u_L$

  $u(x_R) = u_R$

- Consider the following 6 global nodes defined over 4 elements



- We will have the following elemental based expansion

$$\overset{\text{part of } u_B}{} \quad \downarrow \; \downarrow \text{ elemental index}$$

$$u_{app} = u_L \phi_1^1 + u_2^1 \phi_2^1 \qquad \text{Element 1}$$

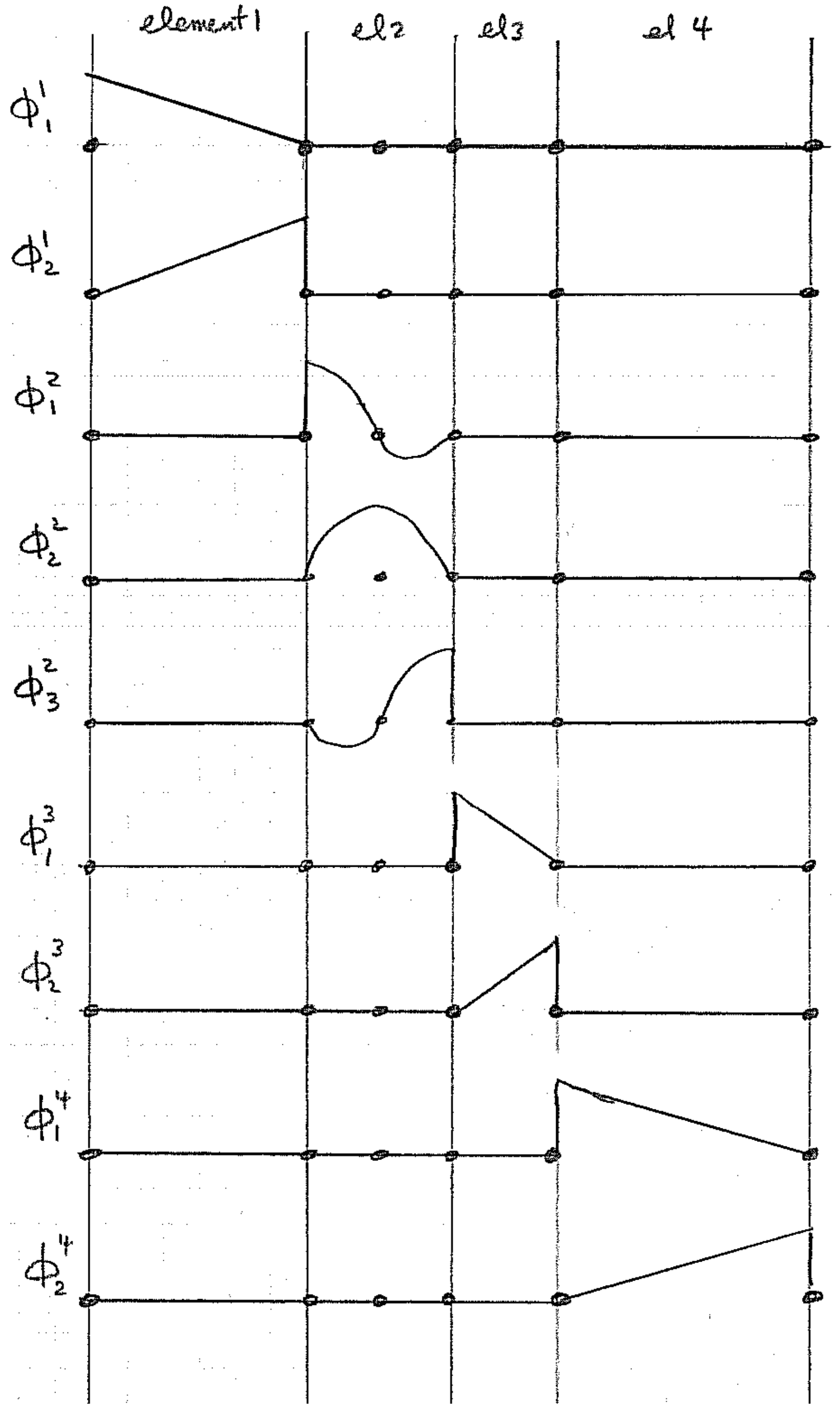
$$\uparrow \; \uparrow \text{ local node index}$$

$$+ u_1^2 \phi_1^2 + u_2^2 \phi_2^2 + u_3^2 \phi_3^2 \qquad \text{Element 2}$$

$$+ u_1^3 \phi_1^3 + u_2^3 \phi_2^3 \qquad \text{Element 3}$$

$$+ u_1^4 \phi_1^4 + u_R \phi_2^4 \quad \leftarrow \text{part of } u_B \qquad \text{Element 4}$$

- This is a local expansion!!
- It has 7 elemental unknowns
- However 3 functional continuity constraints will be applied


element 1
el 2
el 3
el 4
ϕ_1^1
ϕ_2^1
ϕ_1^2
ϕ_2^2
ϕ_3^2
ϕ_1^3
ϕ_2^3
ϕ_1^4
ϕ_2^4

- We can also patch together functions and form truly "global" functions. These are called "Cardinal" bases

$$u_{app} = \overbrace{u_L \Phi_{n1}}^{\text{part of } u_B} + u_{n2}\Phi_{n2} + u_{n3}\Phi_{n3} + u_{n4}\Phi_{n4}$$

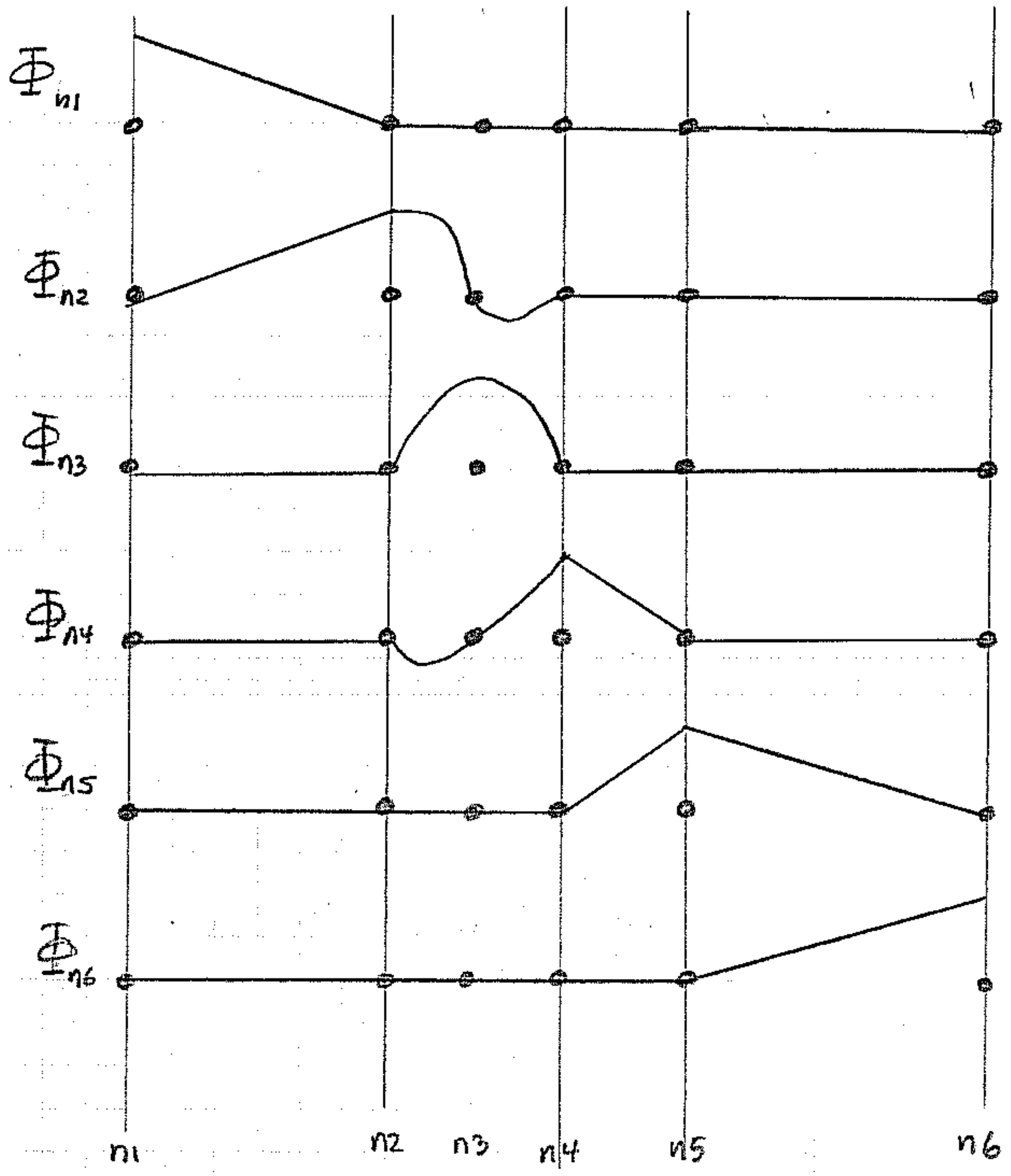
$$+u_{n5}\Phi_{n5} + u_R\Phi_{n6} \leftarrow \text{part of } u_B$$

$\Phi_{n1} = \phi_1^1$	$u_{n1} = u_1^1 = u_L$	essential b.c
$\Phi_{n2} = \phi_2^1 + \phi_1^2$	$u_{n2} = u_2^1 = u_1^2$	unknown
$\Phi_{n3} = \phi_2^2$	$u_{n3} = u_2^2$	unknown
$\Phi_{n4} = \phi_3^2 + \phi_1^3$	$u_{n4} = u_3^2 = u_1^3$	unknown
$\Phi_{n5} = \phi_2^3 + \phi_1^4$	$u_{n5} = u_2^3 = u_1^4$	unknown
$\Phi_{n6} = \underbrace{\phi_2^4}_{\text{local functions}}$	$u_{n6} = \underbrace{u_2^4 = u_R}_{\text{local coef.'s or unknown functions}}$	essential b.c.
↑ Cardinal or global functions	↑ Global coef.'s or unknown functions	

Notes

- The global or Cardinal basis functions and approximation automatically satisfy functional continuity. This is not true for local expansions for which we must still enforce the functional continuity constraints.
- However it will be very easy to handle the functional continuity constraints and it is much easier to work with the local functions in a finite element grid.

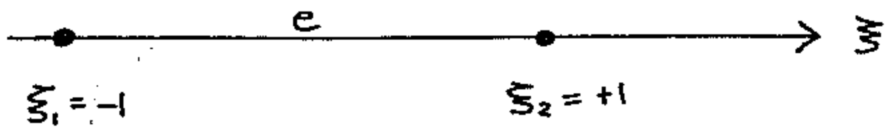
- Global basis functions



Local Lagrange basis functions

- Define a "unit" element with a local coordinate system

$$-1 \leq \xi \leq +1$$



- Map the element *j* which lies in the interval $x_j \leq x \leq x_{j+1}$ to the unit element. The transformation and its inverse are:

$$\xi = -1 + 2(x - x_j)/(x_{j+1} - x_j)$$

and

$$x = x_j + (x_{j+1} - x_j)(\xi + 1)/2$$

- Define the function associated with each node in a local coordinate system:

$$\phi_1(\xi) = a_1 + b_1\xi$$

$$\phi_2(\xi) = a_2 + b_2\xi$$

- Apply constraints to solve for the coefficients

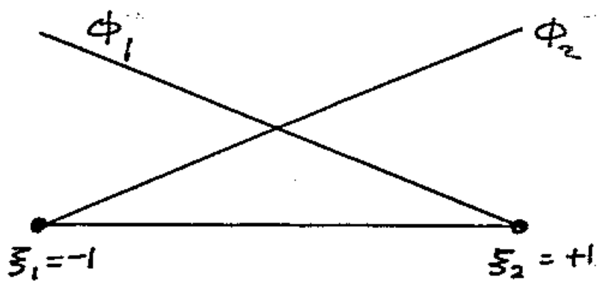
$$\phi_1(\xi_1) = 1 \quad \phi_1(\xi_2) = 0$$

$$\phi_2(\xi_1) = 0 \quad \phi_2(\xi_2) = 1$$

- This leads to:

$$\phi_1(\xi) = \frac{1}{2}(1 - \xi)$$

$$\phi_2(\xi) = \frac{1}{2}(1 + \xi)$$



- ***These functions represent the linear Lagrange interpolation functions. These allow C_o functional continuity and each local basis function equals unity at the associated node and zero elsewhere.***

- These local basis functions on the unit element are related to those defined over the j^{th} element by

$$\phi_1(\xi) = \phi_{2j-1}\big(x(\xi)\big) = \phi_{2j-1}(x)$$

$$\phi_2(\xi) = \phi_{2j}\big(x(\xi)\big) = \phi_{2j}(x)$$

where $\phi_{2j-1}(x)$ and $\phi_{2j}(x)$ are defined as nonzero for $x_j \leq x \leq x_{j+1}$ and zero everywhere else.

- We note that

$$\hat{u}_e = \alpha_1\phi_1(\xi) + \alpha_2\phi_2(\xi)$$

Therefore:

$$\hat{u}_e(\xi_1) = \alpha_1 \quad \hat{u}_e(\xi_2) = \alpha_2$$

Therefore the coefficients of the elemental expansion equals the actual value of the function at the nodes!

- Derivatives of $\phi_{2j-1}(x)$ w.r.t. x

$$\frac{d\phi_{2j-1}}{dx} = \frac{d\phi_1}{dx} = \frac{d\phi_1}{d\xi}\frac{d\xi}{dx} = \frac{2}{\left(x_{j+1} - x_j\right)}\frac{d\phi_1}{d\xi}$$

and

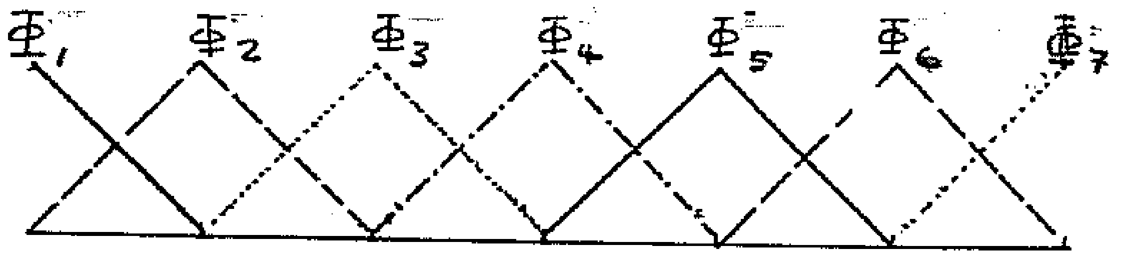
$$\frac{d\phi_{2j}}{dx} = \frac{2}{\left(x_{j+1} - x_j\right)}\frac{d\phi_2}{d\xi}$$

Note that these formula are valid whether or not elements of equal length are used. If $\left(x_{j+1} - x_j\right) = \Delta x$ is constant, then derivatives of the basis functions w.r.t., $d\phi_j/dx$, are related by the constant $2/\Delta x$ to the derivatives w.r.t. ξ of these functions.

<u>Cardinal Basis Functions</u>

- ***If we piece together the elemental functions such that we eliminate or satisfy the functional continuity requirements we form the cardinal basis functions:***

$\Phi^i(x) =$ chapeau functions (also called rooftop or tophat functions)



- Previously we had:

  $u_{app} = \sum_{el_j=1}^{N} u_1^{el_j}\,\phi_1^{el_j} + u_2^{el_j}\phi_2^{el_j}$ with $2N$ *unknowns* when N=the no. of elements

- Now we've enforced functional continuity by

  i. requiring $u_2^{el_1} = u_1^{el_2} = u_2^{\text{global}}$ etc.

  ii. defining $\Phi_2 = \phi_2^{el_1} + \phi_1^{el_2}$ etc.

  Thus we have N-1 constraints and we can now write

$$u_{app}(x) = \sum_{i=1}^{N+1} \left(u_i^{\text{global}}\Phi_i\right)$$

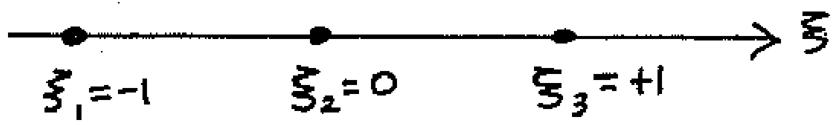
  We now have N+1 unknowns which equals the total number of nodes.

- Therefore the rooftop functions are the same as the first order polynomials defined over each element except that now the functional continuity requirement is automatically satisfied.

- In FE practice we don't really use Cardinal basis functions. We use the local elemental functions and account for functional continuity as matrix assembly proceeds.

Development of Lagrange Quadratic Basis

- C_o functional continuity using quadratic interpolation over an element (instead of the required minimum linear), requires 3 nodes per element.



- Consider the unit element and let the 3 nodes be defined such that $\xi_1 = -1$, $\xi_2 = 0$, and $\xi_3 = +1$.
- The general form of the Lagrange quadratic basis function is:

$$\phi_i(\xi) = a_i + b_i\xi + c_i\xi^2, \quad i = 1,3$$

and the elemental expansion is

$$\hat{u}_e = \sum_{i=1}^{3} u_i\phi_i$$

- We now require that $\phi_i\left(\xi_j\right) = \delta_{ij}, \quad i = 1,3; \;\; j = 1,3$ (this defines 9 constraints to solve for the 9 unknowns).

- Thus:

$$\phi_1^1(-1) = a_1 - b_1 + c_1 = 1$$
$$\phi_1^1(0) = a_1 = 0$$
$$\phi_1^1(+1) = a_1 + b_1 + c_1 = 0$$

Hence:

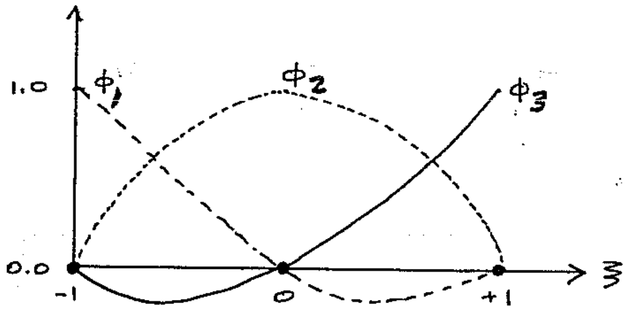
$$a_1 = 0, \quad b_1 = -\frac{1}{2}, \quad c_1 = \frac{1}{2}$$

which leads to:

$$\phi_1(\xi) = \frac{\xi(\xi - 1)}{2}$$

Similarly for $\phi_2(\xi)$ and $\phi_3(\xi)$ and we have:

$$\phi_2(\xi) = 1 - \xi^2$$
$$\phi_3(\xi) = \frac{\xi(1 + \xi)}{2}$$



Notes

- Again we note that the generic expansion is written over each element as

$$\hat{u}_{el} = u_1^{el}\phi_1(\xi) + u_2^{el}\phi_2(\xi) + u_3^{el}\phi_3(\xi)$$

  Thus the coefficients in the expansion for $\hat{u}_e$ equal the value of the function at the node ξ_i. This is only possible due to the form of the basis/expansion functions!

- We still only have C_o functional continuity between elements. We do have non-trivial 2nd derivatives *within* the element. However the 2nd derivatives are not defined at inter-element boundaries!

- Cardinal Basis

# of elemental unknowns	$3N$
# of functional continuity constraints	$N-1$
Total number of global unknowns	$2N+1$

We have

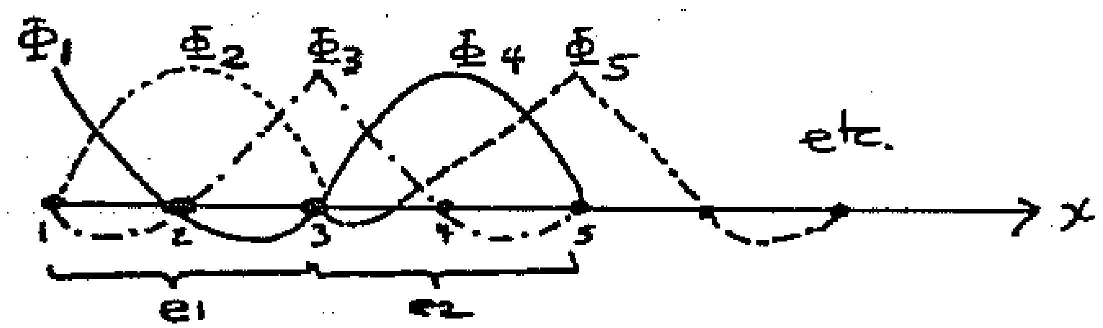
N+1 vertex nodes

N mid-side nodes

2N+1 total number of globally defined basis functions

Note that $2N + 1$ also equals the total number of nodes for N elements.

- Piecing together the element functions we arrive at the following set of $2N + 1$ Cardinal basis functions:



- Each function is associated with 1 global node and is defined such that inter-element continuity is assured.

$$u_{app}(x) = \sum_{i=1}^{2N+1} u_i \Phi_i(x)$$

<u>Higher Order Lagrange Interpolation can be treated in similar ways</u>

1. Add more nodes which allows us to define more interpolating functions of higher order.
2. Require each interpolating function to be equal to unity at the node it is associated with and zero at all other nodes within that element
3. The unknown coefficients will equal the functional value at the nodes!