Report #1 Example
Parallel FEM Class in Winter Semester

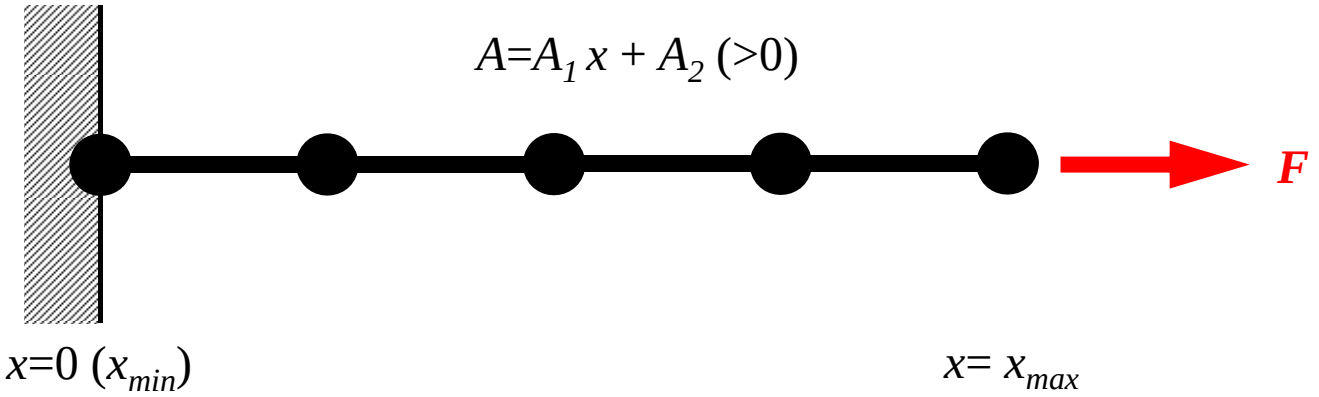
Kengo Nakajima
Information Technology Center

Technical & Scientific Computing I (4820-1027)
Seminar on Computer Science I (4810-1204)

1D Static Linear Elastic Problem

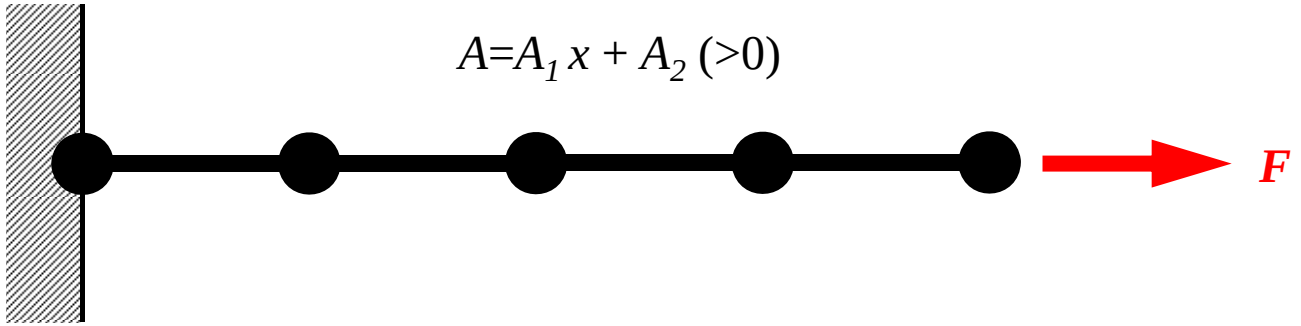
- Only deforms in x-direction (displacement: u)
  - Sectional Area $A=A_1 x + A_2$ (>0)
  - Uniform Young's Modulus E
  - Boundary Conditions (B.C.)
    - $x=0$: $u=0$ (fixed)
    - $x=x_{max}$: F (axial force)
- Truss: NO bending deformation by G-force

1D Static Linear Elastic Problem



- Only deforms in x-direction (displacement: u)
  - Sectional Area $A=A_1 x + A_2$ (>0)
  - Uniform Young's Modulus E
  - Boundary Conditions (B.C.)
    - x=0 : u=0 (fixed)
    - x=x_{max} : F (axial force)
- Truss: NO bending deformation by G-force

1D Static Linear Elastic Problem



Equilibrium Equation $\frac{\partial \sigma_x}{\partial x} + X = 0$

Strain~Displacement $\varepsilon_x = \frac{\partial u}{\partial x}$

Stress~Strain $\sigma_x = E\varepsilon_x$

$$\frac{\partial}{\partial x}\left(E\frac{\partial u}{\partial x}\right) + X = 0$$

Governing Equation for u

Procedures for Computation

- solve equations for displacement u

$$\frac{\partial}{\partial x}\left(E\frac{\partial u}{\partial x}\right)+X=0$$

- calc. strain

$$\varepsilon_x=\frac{\partial u}{\partial x}$$

- calc. stress

$$\sigma_x=E\varepsilon_x$$

Analytical Solution

$$\sigma_x = E\varepsilon_x = \frac{F}{A}$$

$$E\frac{du}{dx} = \frac{F}{A_1 x + A_2}$$

$$Eu = \frac{F}{A_1}\log\left(A_1 x + A_2\right) + C \qquad C = -\frac{F}{A_1}\log\left(A_2\right) \quad \because u = 0 @ x = 0$$

$$\therefore \quad u = \frac{F}{EA_1}\left[\log\left(A_1 x + A_2\right) - \log\left(A_2\right)\right]$$

Report #1

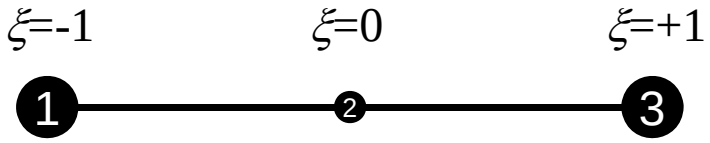
- Implement quadratic interpolation（二次形状関数）on “b1.c/b1.f”. Name of the developed code: “b2.c/b2.f”
- Evaluate accuracy of “b1” and “b2” according to effect of number of meshes.
- Confirm the following equation is correct, if sectional area is constant:

$$\int_V E\left(\frac{d[N]^T}{dx}\frac{d[N]}{dx}\right)dV = \frac{EA}{6L}\begin{bmatrix} +14 & -16 & +2 \\ -16 & +32 & -16 \\ +2 & -16 & +14 \end{bmatrix}$$

- Due on August 18th (M), 2014 at 17:00
- Documents
  - Report (Outline, Results, Discussions) (less than 5 pages)
  - List of Source Code

Report #1: Tips (1)

- Derivative of Shape Functions



$$N_1(\xi)=\frac{1}{2}\xi(-1+\xi)$$
$$N_2(\xi)=(1+\xi)(1-\xi)$$
$$N_3(\xi)=\frac{1}{2}\xi(1+\xi)$$

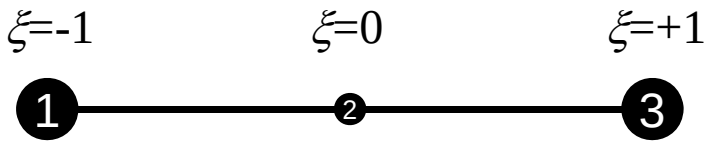
$$\frac{dN_1}{d\xi}=-\frac{1}{2}+\xi$$
$$\frac{dN_2}{d\xi}=-2\xi$$
$$\frac{dN_3}{d\xi}=\frac{1}{2}+\xi$$

$$[Emat]=E\sum_{k=1}^{m}\left[w_k\cdot\frac{1}{|J|}\Big|_{\xi=\xi_k}\begin{bmatrix}\frac{\partial N_1}{\partial\xi}\frac{\partial N_1}{\partial\xi} & \frac{\partial N_1}{\partial\xi}\frac{\partial N_2}{\partial\xi} & \frac{\partial N_1}{\partial\xi}\frac{\partial N_3}{\partial\xi}\\ \frac{\partial N_2}{\partial\xi}\frac{\partial N_1}{\partial\xi} & \frac{\partial N_2}{\partial\xi}\frac{\partial N_2}{\partial\xi} & \frac{\partial N_2}{\partial\xi}\frac{\partial N_3}{\partial\xi}\\ \frac{\partial N_3}{\partial\xi}\frac{\partial N_1}{\partial\xi} & \frac{\partial N_3}{\partial\xi}\frac{\partial N_2}{\partial\xi} & \frac{\partial N_3}{\partial\xi}\frac{\partial N_3}{\partial\xi}\end{bmatrix}\Big|_{\xi=\xi_k}A(\xi_k)\right]$$

Values at Gaussian Quad Points (ξ_k)

Report #1: Tips (2)

- Jacobian



$$\frac{\partial x}{\partial \xi} = \frac{\partial}{\partial \xi}\sum_{i=1}^{3}\left(N_i x_i\right) = \sum_{i=1}^{3}\left(\frac{\partial N_i}{\partial \xi}x_i\right) = \frac{\partial N_1}{\partial \xi}x_1 + \frac{\partial N_2}{\partial \xi}x_2 + \frac{\partial N_3}{\partial \xi}x_3$$

- Jacobian at Gaussian Quad Points (ξ_k)

$$\left.\frac{\partial x}{\partial \xi}\right|_{\xi=\xi_k} = \left.\frac{\partial N_1}{\partial \xi}\right|_{\xi=\xi_k} x_1 + \left.\frac{\partial N_2}{\partial \xi}\right|_{\xi=\xi_k} x_2 + \left.\frac{\partial N_3}{\partial \xi}\right|_{\xi=\xi_k} x_3$$

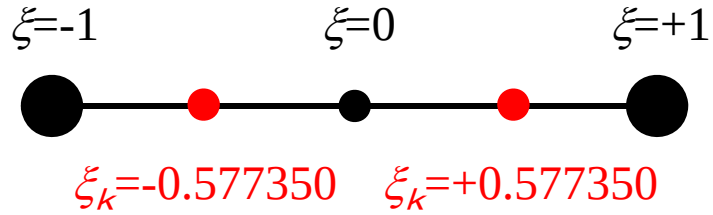
Strategy

- Uniform Sect. Area, Linear Elem. (Analytical Integration)
  - <$fem1>/1d/1d.c
- Uniform Sect. Area, Higher-Order Elem.
  - <$fem1>/1d/1d2.c
- Non-uniform Sect. Area, Linear Elem. (Analytical)
  - <$fem1>/1darea/a1.c
- Non-uniform Sect. Area, Isoparametric Linear Elem. (Numerical Integration)
  - <$fem1>/1darea/b1.c
- Non-uniform Sect. Area, Isoparametric Higher-Order Elem. (Numerical Integration)
  - Starting from “1d2.c” is easy.

Gaussian Quadrature

- On normalized “natural (or local)” coordinate system [-1,+1]
- Can approximate up to (*2m-1*)-th order of functions by *m* quadrature points (*m*=2 is enough for quadratic shape functions).

$$\int_{-1}^{+1} f(\xi)d\xi = \sum_{k=1}^{m}[w_k \cdot f(\xi_k)]$$



$$m=1 \quad \xi_k = 0.00,\ w_k = 2.00$$

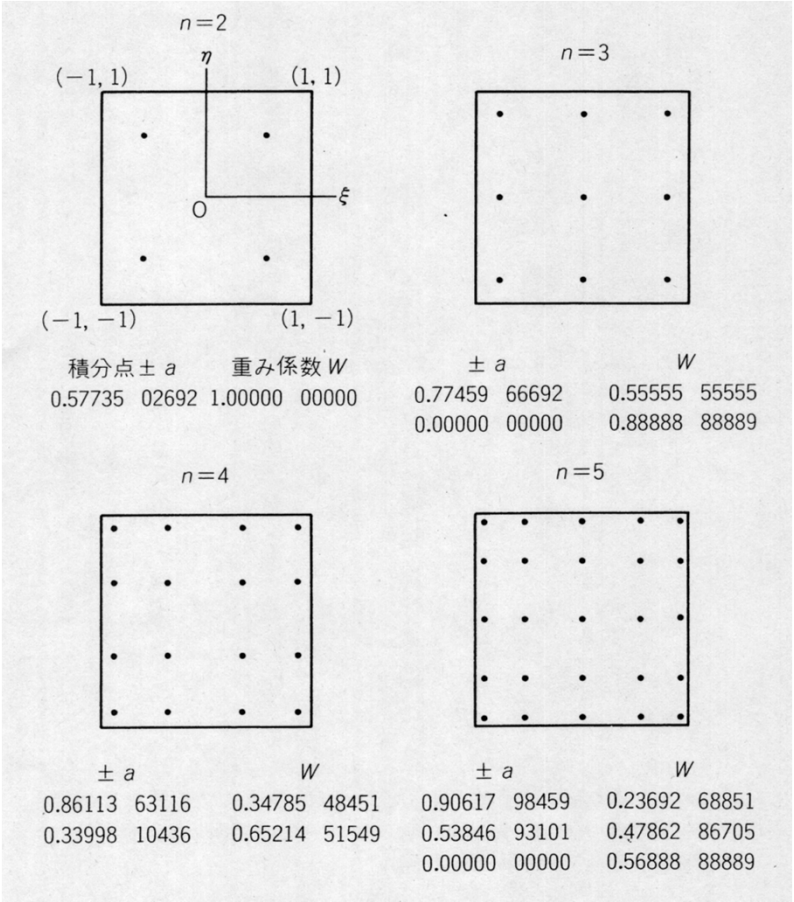
$$m=2 \quad \xi_k = \pm 0.577350,\ w_k = 1.00$$

$$m=3 \quad \xi_k = 0.00,\ w_k = 8/9$$

$$\xi_k = \pm 0.774597,\ w_k = 5/9$$

Gaussian Quadrature

can be easily extended to 2D & 3D



$$I = \int_{-1}^{+1} \int_{-1}^{+1} f(\xi,\eta)\, d\xi d\eta$$

$$= \sum_{i=1}^{m} \sum_{j=1}^{n} \left[W_i \cdot W_j \cdot f(\xi_i,\eta_j) \right]$$

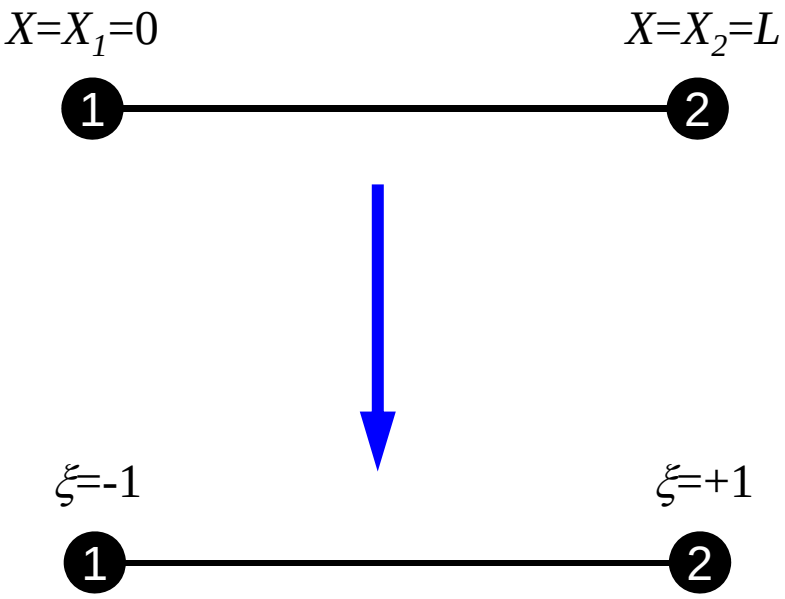
m,n: number of quadrature points in ξ,η-direction

(ξ_i,η_j): Coordinates of Quad's

W_i, W_j: Weighting Factor

How to use Gaussian Quadrature

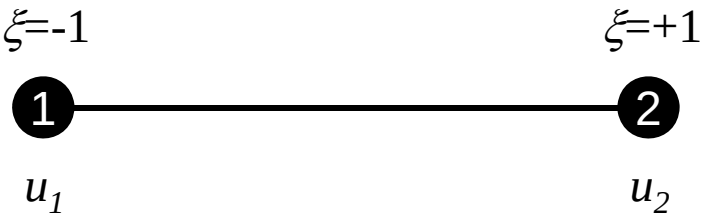
- Coordinate transformation from $[0,L]$ (or $[X_1,X_2]$) to [-1,+1] is needed.
- Shape/Interpolation functions must be handled on natural/local coordinate system.



Local node ID's (1,2,3…) are defined as subscripts of X, u, N and etc.

Piecewise Linear Element

$$u = \alpha_1 + \alpha_2 \xi$$
$$u_1 = \alpha_1 + \alpha_2 (-1)$$
$$u_2 = \alpha_1 + \alpha_2 (+1)$$



$$\alpha_1 = \frac{u_1 + u_2}{2}, \alpha_2 = \frac{-u_1 + u_2}{2}$$

$$u = \alpha_1 + \alpha_2 \xi = \frac{u_1 + u_2}{2} + \frac{-u_1 + u_2}{2} \xi = \underbrace{\frac{1}{2}(1-\xi)}_{N_1(\xi)} u_1 + \underbrace{\frac{1}{2}(1+\xi)}_{N_2(\xi)} u_2$$

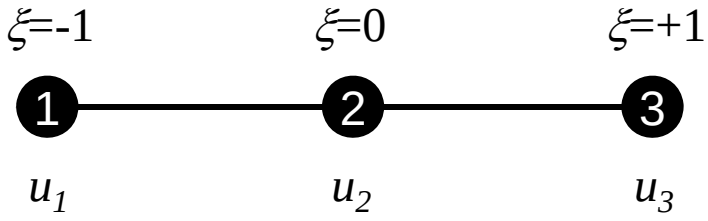
$$N_1(\xi) = \frac{1}{2}(1-\xi), \quad N_2(\xi) = \frac{1}{2}(1+\xi)$$

2nd-order/Quadratic Element

$$u = \alpha_1 + \alpha_2 \xi + \alpha_3 \xi^2$$

$$u_1 = \alpha_1 - \alpha_2 + \alpha_3, \quad u_2 = \alpha_1$$

$$u_3 = \alpha_1 + \alpha_2 + \alpha_3$$



$$\alpha_1 = u_2, \alpha_2 = \frac{-u_1 + u_3}{2}, \alpha_3 = \frac{u_1 - 2u_2 + u_3}{2}$$

$$u = \alpha_1 + \alpha_2 \xi + \alpha_3 \xi^2 = u_2 + \frac{-u_1 + u_3}{2}\xi + \frac{u_1 - 2u_2 + u_3}{2}\xi^2$$

$$= \frac{-\xi + \xi^2}{2} u_1 + \left(1 - \xi^2\right) u_2 + \frac{\xi + \xi^2}{2} u_3$$

$$= \frac{1}{2}\xi(-1+\xi) u_1 + (1+\xi)(1-\xi) u_2 + \frac{1}{2}\xi(1+\xi) u_3$$

$N_1(\xi)$ $N_2(\xi)$ $N_3(\xi)$

$$N_1(\xi) = \frac{1}{2}\xi(-1+\xi), \; N_2(\xi) = (1+\xi)(1-\xi), \; N_3(\xi) = \frac{1}{2}\xi(1+\xi)$$

Isoparametric Element

- Definitions of “Isoparametric” Elements
  - Each element is defined on natural/local coordinate for [-1,+1]
  - And the components of global coordinate system of each node (e.g. x,y,z) for certain kinds of elements are defined by shape functions $[N]$ on natural/local coordinate system, where shape functions $[N]$ are also used for interpolation of dependent variables.

$$u = \sum_{i=1}^{n_N} N_i(\xi) \cdot u_i, \quad x = \sum_{i=1}^{n_N} N_i(\xi) \cdot x_i$$

$$n_N : 2,3...$$

$$u = \frac{1}{2}(1-\xi)u_1 + \frac{1}{2}(1+\xi)u_2$$

$$x = \frac{1}{2}(1-\xi)x_1 + \frac{1}{2}(1+\xi)x_2$$

$$\left(= \frac{1}{2}(x_2 - x_1)\xi + \frac{1}{2}(x_1 + x_2) \right)$$

Integration over Each Element: $[k]$

$$\int_V E\left(\frac{\partial[N]^T}{\partial x}\frac{\partial[N]}{\partial x}\right)dV = \int_V E\left(\frac{\partial[N]^T}{\partial x}\frac{\partial[N]}{\partial x}\right)Adx$$

Partial Differentiation on Natural/Local Coordinate System (1/2)

- According to formulae:

$$\frac{\partial N_i(\xi)}{\partial \xi} = \frac{\partial N_i}{\partial x}\frac{\partial x}{\partial \xi}$$

$\left[\frac{\partial N_i}{\partial \xi}\right]$ can be easily derived according to definitions.

$\left[\frac{\partial N_i}{\partial x}\right]$ are required for computations.

$\left[\frac{\partial x}{\partial \xi}\right]$ is Jacobian (=J)

$$\frac{\partial x}{\partial \xi} = \frac{\partial}{\partial \xi}\left(\sum_{i=1}^{n_N} N_i x_i\right) = \sum_{i=1}^{n_N} \frac{\partial N_i}{\partial \xi} x_i = J$$

Partial Differentiation on Natural/Local Coordinate System (2/2)

- Target terms are derived as follows:

$$\frac{\partial N_i}{\partial x} = \frac{\dfrac{\partial N_i(\xi)}{\partial \xi}}{\dfrac{\partial x}{\partial \xi}} = \frac{\partial N_i(\xi)}{\partial \xi} \cdot \frac{1}{J}$$

- Integration on natural/local coordinate system:

$$\int_0^L f(x)dx = \int_{-1}^{+1} f(\xi)|J|d\xi \quad \because dx = |J|d\xi = \left|\frac{\partial x}{\partial \xi}\right| d\xi$$

Integration over Each Element: $[k]$ (1/2)

$$\int_V E\left(\frac{\partial[N]^T}{\partial x}\frac{\partial[N]}{\partial x}\right)dV = E\int_V\left(\frac{\partial[N]^T}{\partial x}\frac{\partial[N]}{\partial x}\right)A(x)dx$$

$$= E\int_{-1}^{+1}\left(\frac{\partial[N]^T}{\partial x}\frac{\partial[N]}{\partial x}\right)A(\xi)|J|d\xi = E\int_{-1}^{+1}\left(\left[\frac{\partial[N]^T}{\partial \xi}\frac{1}{J}\right]\left[\frac{\partial[N]}{\partial \xi}\frac{1}{J}\right]\right)A(\xi)|J|d\xi$$

$$= E\int_{-1}^{+1}\left(\frac{1}{|J|}\frac{\partial[N]^T}{\partial \xi}\frac{\partial[N]}{\partial \xi}A(\xi)\right)d\xi$$

$$= E\sum_{k=1}^{m}\left[w_k \cdot \frac{1}{|J|}\Big|_{\xi=\xi_k}\frac{\partial[N]^T}{\partial \xi}\Big|_{\xi=\xi_k}\frac{\partial[N]}{\partial \xi}\Big|_{\xi=\xi_k}A(\xi_k)\right]$$

Integration over Each Element: $[k]$ (2/2)

$$E\sum_{k=1}^{m}\left[w_k \cdot \frac{1}{|J|}\bigg|_{\xi=\xi_k} \frac{\partial[N]^T}{\partial\xi}\bigg|_{\xi=\xi_k} \frac{\partial[N]}{\partial\xi}\bigg|_{\xi=\xi_k} A(\xi_k) \right]$$

$$= E\sum_{k=1}^{m}\left[w_k \cdot \frac{1}{|J|}\bigg|_{\xi=\xi_k} \begin{bmatrix} \partial N_1/\partial\xi \\ \partial N_2/\partial\xi \end{bmatrix} \begin{bmatrix} \frac{\partial N_1}{\partial\xi}, \frac{\partial N_2}{\partial\xi} \end{bmatrix}\bigg|_{\xi=\xi_k} A(\xi_k) \right]$$

$$= E\sum_{k=1}^{m}\left[w_k \cdot \frac{1}{|J|}\bigg|_{\xi=\xi_k} \begin{bmatrix} \frac{\partial N_1}{\partial\xi}\frac{\partial N_1}{\partial\xi} & \frac{\partial N_1}{\partial\xi}\frac{\partial N_2}{\partial\xi} \\ \frac{\partial N_2}{\partial\xi}\frac{\partial N_1}{\partial\xi} & \frac{\partial N_2}{\partial\xi}\frac{\partial N_2}{\partial\xi} \end{bmatrix}\bigg|_{\xi=\xi_k} A(\xi_k) \right]$$

Example (1/7): Var's/Array's

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <assert.h>

int main(){
        int NE, N, NPLU, IterMax, errno, NPLU0;
        int R, Z, Q, P, DD, ip;

        double dX, Resid, Eps, Area, F, Young, Jacobi;
        double X1, X2, X3, U1, U2, U3, DL, Strain, Sigma, Ck, XX, X0, A1, A2, DISP;
        double *U, *Rhs, *X;
        double *Diag, *AMat;
        double **W;

        int *Index, *Item, *IcelNod;
        double POI[2], WEI[2], dNdQ[3], Emat[3][3];

        int i, j, in1, in2, in3, k, icel, k12, k13, k21, k23, k31, k32, jS;
        int iter;
        FILE *fp;
        double BNorm2, Rho, Rho1=0.0, C1, Alpha, DNorm2;
        int ierr = 1;
```

Example (2/7)

Initialization, Array Allocation

```
/*
// +-------+
// | INIT. |
// +-------+
*/
        fp = fopen("input2.dat", "r");
        assert(fp != NULL);
        fscanf(fp, "%d", &NE);
        fscanf(fp, "%lf %lf %lf %lf %lf", &dX, &F, &A1, &A2, &Young);
        fscanf(fp, "%d", &IterMax);
        fscanf(fp, "%lf", &Eps);
        fclose(fp);

        N    = 2*NE + 1;
        NPLU0= 2*2 + NE*2 + (NE-1)*4;

        U    = calloc(N, sizeof(double));
        X    = calloc(N, sizeof(double));
        Diag = calloc(N, sizeof(double));
        AMat = calloc(NPLU0, sizeof(double));
        Rhs = calloc(N, sizeof(double));
        Index= calloc(N+1, sizeof(int));
        Item = calloc(NPLU0, sizeof(int));
        Icelnod= calloc(3*NE, sizeof(int));
```

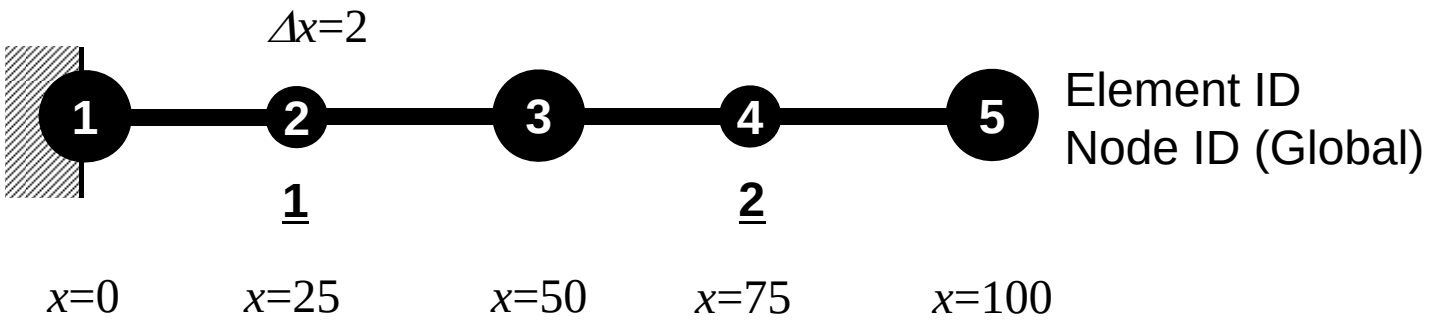
Input Data

input.dat

```
2
50.0   5.e4   -0.105 12   5.e6
100
1.e-8
```

NE (Number of Elements)
Δx, **F**, **A_1**, **A_2**, **E**
Number of MAX. Iterations for CG Solver
Convergence Criteria for CG Solver

$$x = 0 \quad A_1 x + A_2 = 12.$$
$$x = 100 \quad A_1 x + A_2 = 1.5$$

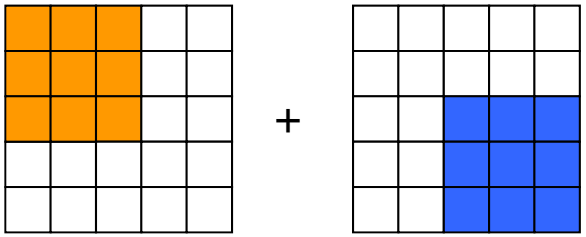


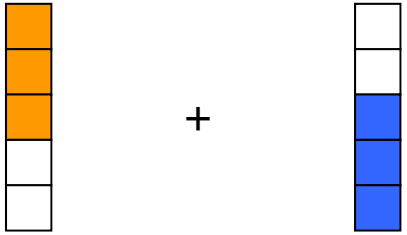
Example (2/7)
Initialization, Array Allocation

```
/*
// +-------+
// | INIT. |
// +-------+
*/
        fp = fopen("input2.dat", "r");
        assert(fp != NULL);
        fscanf(fp, "%d", &NE);
        fscanf(fp, "%lf %lf %lf %lf %lf", &dX, &F, &A1, &A2, &Young);
        fscanf(fp, "%d", &IterMax);
        fscanf(fp, "%lf", &Eps);
        fclose(fp);

        N    = 2*NE + 1;
        NPLU0= 2*2 + NE*2 + (NE-1)*4;

        U    = calloc(N, sizeof(double));
        X    = calloc(N, sizeof(double));
        Diag = calloc(N, sizeof(double));
        AMat = calloc(NPLU0, sizeof(double));
        Rhs = calloc(N, sizeof(double));
        Index= calloc(N+1, sizeof(int));
        Item = calloc(NPLU0, sizeof(int));
        Icelnod= calloc(3*NE, sizeof(int));
```

Amat: Non-Zero Off-Diag. Comp.
Item: Corresponding Column ID

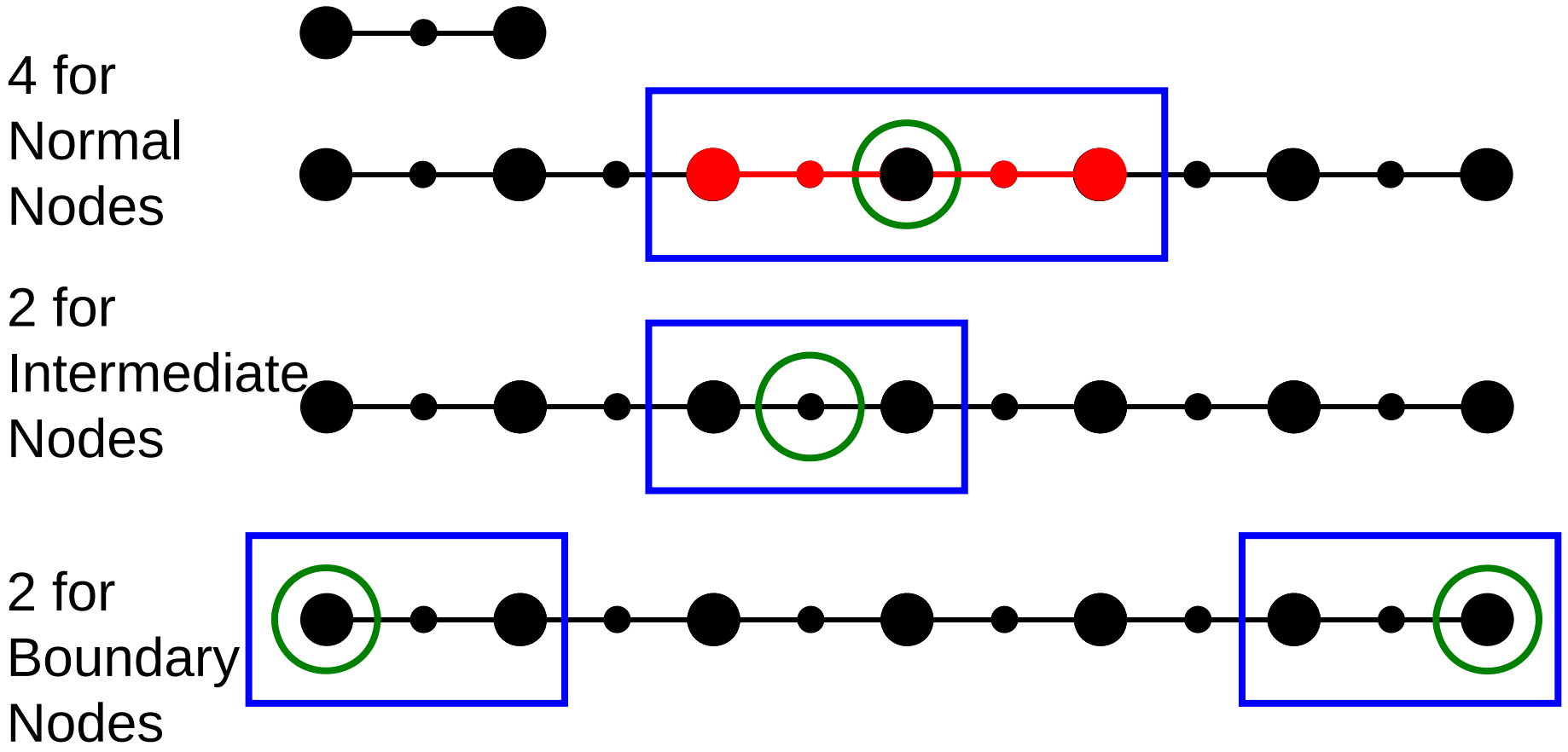
Non-Zero Off-Diagonal Components Nodes are connected to node on each element each other

$$[K]=\sum_{i=1}^{2}\left[k^{(i)}\right]=$$  +

$$\{F\}=\sum_{i=1}^{4}\left\{f^{(i)}\right\}=$$  +

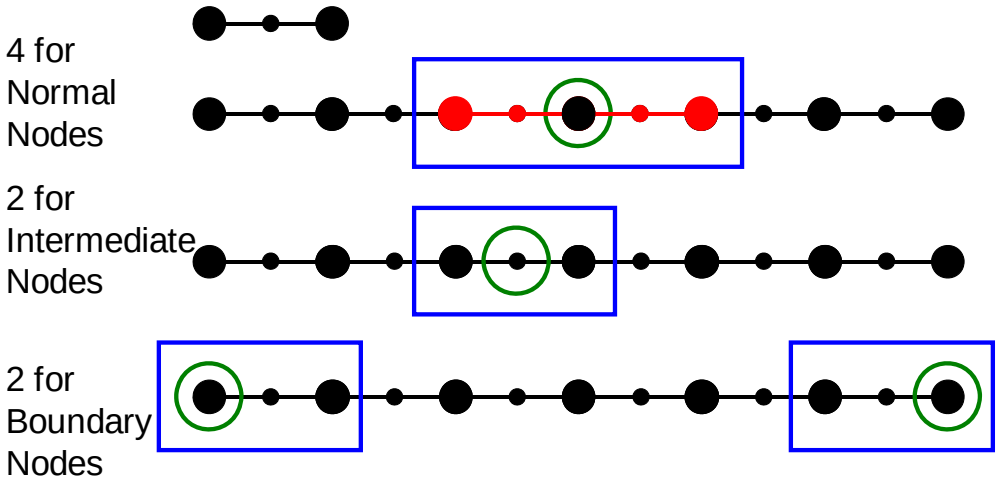
Number of Non-Zero Off-Diagonals

Different number of connections according to location of each node



Number of Non-Zero Off-Diagonals

Different number of connections according to location of each node



- Boundary Node# 2
- Intermediate Node# NE
- Normal Node# NE+1-2= NE-1
  - except Boundary Nodes
- Total Node#: N= 2+NE+NE-1= 2*NE + 1
- Non-Zero Off-Diag.#: NPLU= 2*2 + 2*NE + 4*(NE-1)

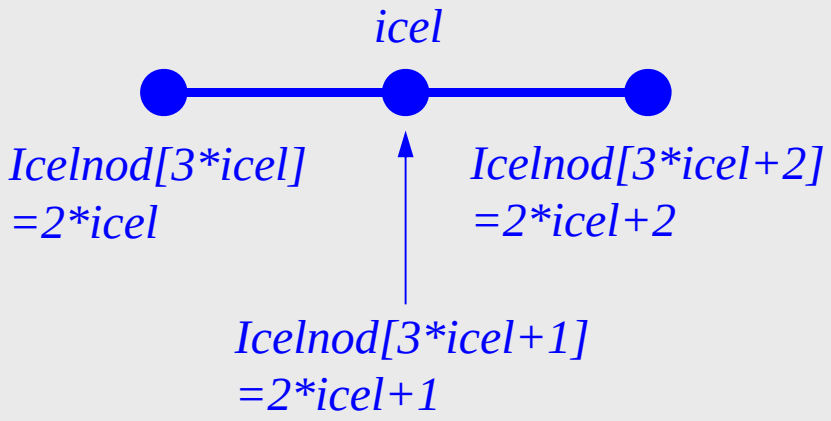
Example (3/7)
Initialization, Array Allocation (cont.)

```
W = (double **)malloc(sizeof(double *)*4);
if(W == NULL) {
        fprintf(stderr, "Error: %s¥n", strerror(errno));
        return -1;
}
for(i=0; i<4; i++) {
        W[i] = (double *)malloc(sizeof(double)*N);
        if(W[i] == NULL) {
                fprintf(stderr, "Error: %s¥n", strerror(errno));
                return -1;
        }
}

for(i=0;i<N;i++)    U[i] = 0.0;
for(i=0;i<N;i++) Diag[i] = 0.0;
for(i=0;i<N;i++)  Rhs[i] = 0.0;
for(k=0;k<NPLU0;k++)  AMat[k] = 0.0;
for(i=0;i<N;i++) X[i]= i*dX*0.5;
for(icel=0;icel<NE;icel++){
        Icelnod[3*icel  ]= 2*icel;
        Icelnod[3*icel+1]= 2*icel+1;
        Icelnod[3*icel+2]= 2*icel+2;
}

WEI[0]= +1.0;
WEI[1]= +1.0;
POI[0]= -0.577350;
POI[1]= +0.577350;
```

X: X-coordinate component of each node

Example (3/7)
Initialization, Array Allocation (cont.)

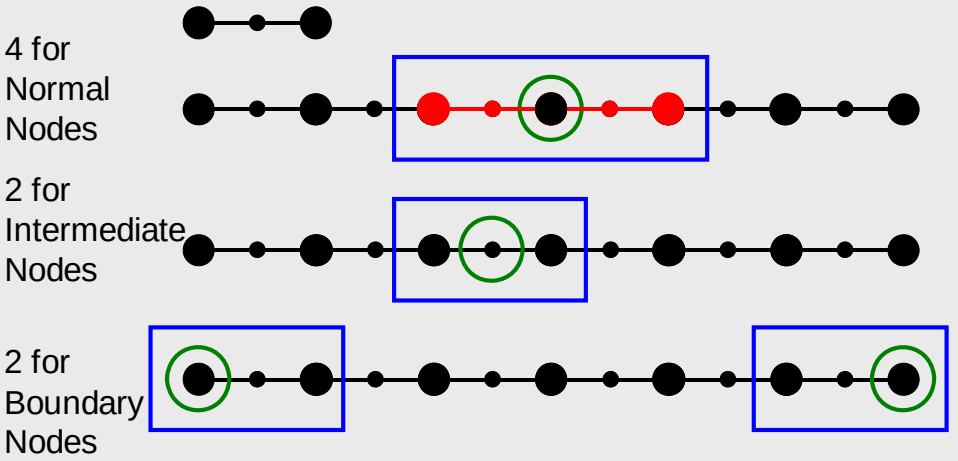
```
W = (double **)malloc(sizeof(double *)*4);
if(W == NULL) {
        fprintf(stderr, "Error: %s¥n", strerror(errno));
        return -1;
}
for(i=0; i<4; i++) {
        W[i] = (double *)malloc(sizeof(double)*N);
        if(W[i] == NULL) {
                fprintf(stderr, "Error: %s¥n", strerror(errno));
                return -1;
        }
}

for(i=0;i<N;i++)    U[i] = 0.0;
for(i=0;i<N;i++) Diag[i] = 0.0;
for(i=0;i<N;i++)  Rhs[i] = 0.0;
for(k=0;k<NPLU0;k++)  AMat[k] = 0.0;
for(i=0;i<N;i++) X[i]= i*dX*0.5;
for(icel=0;icel<NE;icel++){
        Icelnod[3*icel  ]= 2*icel;
        Icelnod[3*icel+1]= 2*icel+1;
        Icelnod[3*icel+2]= 2*icel+2;
}

WEI[0]= +1.0;
WEI[1]= +1.0;
POI[0]= -0.577350;
POI[1]= +0.577350;
```



Example (4/7)

Global Matrix: Column ID for Non-Zero Off-Diag's

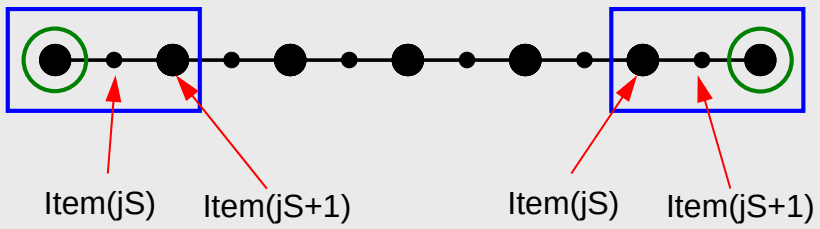
```
/*
// +--------------+
// | CONNECTIVITY |
// +--------------+
*/
        Index[0]= 0;
        for(i=1;i<N;i++) {
          if (i%2==1) {Index[i]=4;
          }else       {Index[i]=2;}}
        Index[1]= 2;
        Index[N]= 2;
        for(i=0;i<N;i++){
                Index[i+1]= Index[i+1] + Index[i];}

        NPLU= Index[N];
        for(i=0;i<N;i++){
                int jS = Index[i];
                if(i == 0){
                        Item[jS  ] = i+1;
                        Item[jS+1] = i+2;
                }else if(i == N-1){
                        Item[jS  ] = i-2;
                        Item[jS+1] = i-1;
                }else{
                  if (i%2==1){
                        Item[jS  ] = i-1;
                        Item[jS+1] = i+1;
                  } else       {
                        Item[jS  ] = i-2;
                        Item[jS+1] = i-1;
                        Item[jS+2] = i+1;
                        Item[jS+3] = i+2;}}}
```



Example (4/7)

Global Matrix: Column ID for Non-Zero Off-Diag's

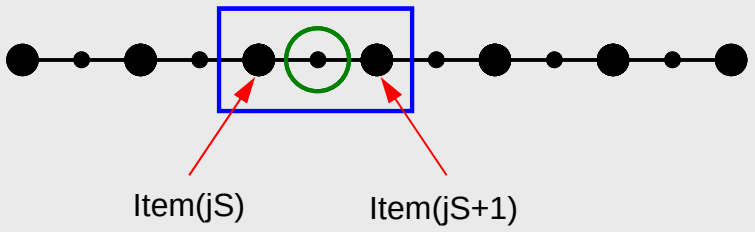
```
/*
// +--------------+
// | CONNECTIVITY |
// +--------------+
*/
        Index[0]= 0;
        for(i=1;i<N;i++) {
          if (i%2==1) {Index[i]=4;
          }else       {Index[i]=2;}}
        Index[1]= 2;
        Index[N]= 2;
        for(i=0;i<N;i++){
                Index[i+1]= Index[i+1] + Index[i];}

        NPLU= Index[N];
        for(i=0;i<N;i++){
                int jS = Index[i];
                if(i == 0){
                        Item[jS  ] = i+1;
                        Item[jS+1] = i+2;
                }else if(i == N-1){
                        Item[jS  ] = i-2;
                        Item[jS+1] = i-1;
                }else{
                  if (i%2==1){
                        Item[jS  ] = i-1;
                        Item[jS+1] = i+1;
                  } else       {
                        Item[jS  ] = i-2;
                        Item[jS+1] = i-1;
                        Item[jS+2] = i+1;
                        Item[jS+3] = i+2;}}}
```



Example (4/7)

Global Matrix: Column ID for Non-Zero Off-Diag's

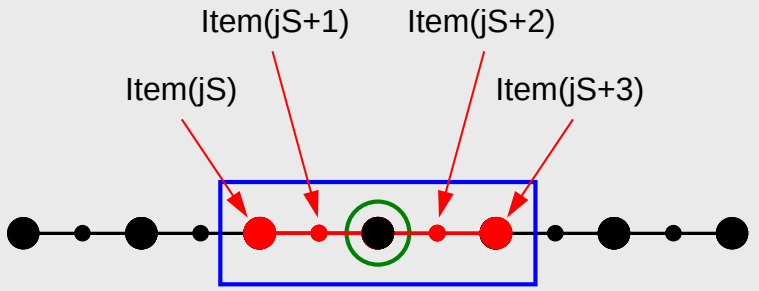
```
/*
// +--------------+
// | CONNECTIVITY |
// +--------------+
*/
        Index[0]= 0;
        for(i=1;i<N;i++) {
          if (i%2==1) {Index[i]=4;
          }else       {Index[i]=2;}}
        Index[1]= 2;
        Index[N]= 2;
        for(i=0;i<N;i++){
                Index[i+1]= Index[i+1] + Index[i];}

        NPLU= Index[N];
        for(i=0;i<N;i++){
                int jS = Index[i];
                if(i == 0){
                        Item[jS  ] = i+1;
                        Item[jS+1] = i+2;
                }else if(i == N-1){
                        Item[jS  ] = i-2;
                        Item[jS+1] = i-1;
                }else{
                  if (i%2==1){
                        Item[jS  ] = i-1;
                        Item[jS+1] = i+1;
                  } else       {
                        Item[jS  ] = i-2;
                        Item[jS+1] = i-1;
                        Item[jS+2] = i+1;
                        Item[jS+3] = i+2;}}}
```



Example (4/7)

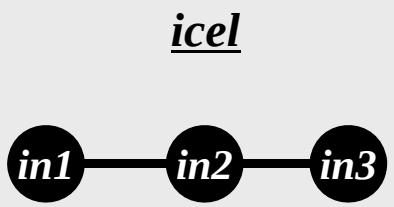
Global Matrix: Column ID for Non-Zero Off-Diag's

```
/*
// +--------------+
// | CONNECTIVITY |
// +--------------+
*/
        Index[0]= 0;
        for(i=1;i<N;i++) {
          if (i%2==1) {Index[i]=4;
          }else       {Index[i]=2;}}
        Index[1]= 2;
        Index[N]= 2;
        for(i=0;i<N;i++){
                Index[i+1]= Index[i+1] + Index[i];}

        NPLU= Index[N];
        for(i=0;i<N;i++){
                int jS = Index[i];
                if(i == 0){
                        Item[jS  ] = i+1;
                        Item[jS+1] = i+2;
                }else if(i == N-1){
                        Item[jS  ] = i-2;
                        Item[jS+1] = i-1;
                }else{
                  if (i%2==1){
                        Item[jS  ] = i-1;
                        Item[jS+1] = i+1;
                  } else       {
                        Item[jS  ] = i-2;
                        Item[jS+1] = i-1;
                        Item[jS+2] = i+1;
                        Item[jS+3] = i+2;}}}
```



Example (5/7): Matrix (1/3)

```
/*
// +-----------------+
// | MATRIX assemble |
// +-----------------+
*/
        for(icel=0;icel<NE;icel++){
          in1= Icelnod[3*icel];
          in2= Icelnod[3*icel+1];
          in3= Icelnod[3*icel+2];
          X1 = X[in1];
          X2 = X[in2];
          X3 = X[in3];

          DL = fabs(X3-X1);
          X0 = 0.5 * (X1+X3);

          Emat[0][0]= 0.0;
          Emat[0][1]= 0.0;
          Emat[0][2]= 0.0;
          Emat[1][0]= 0.0;
          Emat[1][1]= 0.0;
          Emat[1][2]= 0.0;
          Emat[2][0]= 0.0;
          Emat[2][1]= 0.0;
          Emat[2][2]= 0.0;
```

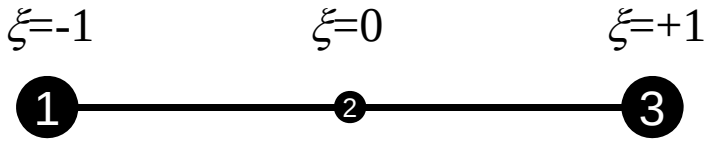


Example (6/7): Matrix (2/3)

```
for(ip=0;ip<2;ip++){
  dNdQ[0]= -0.5 + POI[ip];
  dNdQ[1]= -2.0 * POI[ip];
  dNdQ[2]=  0.5 + POI[ip];

  XX= X0 + POI[ip]*0.50*DL;
  Area= A1*XX + A2;

 if(Area<= 0.) {
       fprintf(stderr, "ERROR: Area<0: ¥n");
       return -1;
 }

  Jacobi= fabs(dNdQ[0]*X1 + dNdQ[1]*X2 + dNdQ[2]*X3);
  Ck= Area*Young/Jacobi;
  Emat[0][0]= Emat[0][0] + Ck * WEI[ip] * dNdQ[0] * dNdQ[0];
  Emat[0][1]= Emat[0][1] + Ck * WEI[ip] * dNdQ[0] * dNdQ[1];
  Emat[0][2]= Emat[0][2] + Ck * WEI[ip] * dNdQ[0] * dNdQ[2];
  Emat[1][0]= Emat[1][0] + Ck * WEI[ip] * dNdQ[1] * dNdQ[0];
  Emat[1][1]= Emat[1][1] + Ck * WEI[ip] * dNdQ[1] * dNdQ[1];
  Emat[1][2]= Emat[1][2] + Ck * WEI[ip] * dNdQ[1] * dNdQ[2];
  Emat[2][0]= Emat[2][0] + Ck * WEI[ip] * dNdQ[2] * dNdQ[0];
  Emat[2][1]= Emat[2][1] + Ck * WEI[ip] * dNdQ[2] * dNdQ[1];
  Emat[2][2]= Emat[2][2] + Ck * WEI[ip] * dNdQ[2] * dNdQ[2];
}
```

Derivatives at Gaussian Quad. Points

- Derivative of Shape Functions



$$N_1(\xi)=\frac{1}{2}\xi(-1+\xi)$$
$$N_2(\xi)=(1+\xi)(1-\xi)$$
$$N_3(\xi)=\frac{1}{2}\xi(1+\xi)$$

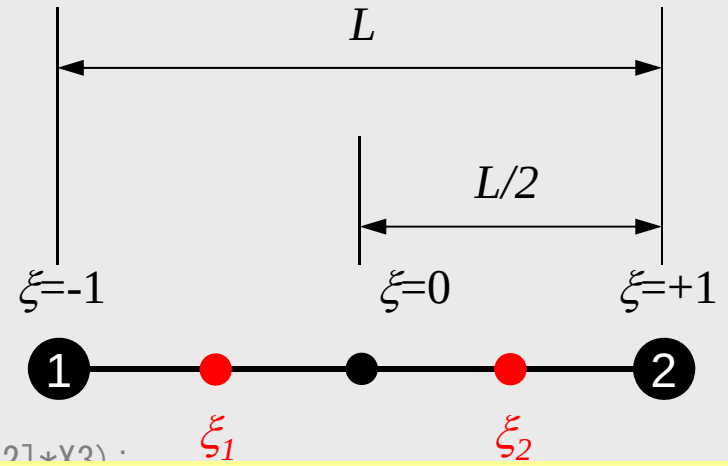
$$\frac{dN_1}{d\xi}=-\frac{1}{2}+\xi$$
$$\frac{dN_2}{d\xi}=-2\xi$$
$$\frac{dN_3}{d\xi}=\frac{1}{2}+\xi$$

$$[Emat]=E\sum_{k=1}^{m}\left[w_k\cdot\frac{1}{|J|}\Big|_{\xi=\xi_k}\begin{bmatrix}\frac{\partial N_1}{\partial\xi}\frac{\partial N_1}{\partial\xi} & \frac{\partial N_1}{\partial\xi}\frac{\partial N_2}{\partial\xi} & \frac{\partial N_1}{\partial\xi}\frac{\partial N_3}{\partial\xi}\\ \frac{\partial N_2}{\partial\xi}\frac{\partial N_1}{\partial\xi} & \frac{\partial N_2}{\partial\xi}\frac{\partial N_2}{\partial\xi} & \frac{\partial N_2}{\partial\xi}\frac{\partial N_3}{\partial\xi}\\ \frac{\partial N_3}{\partial\xi}\frac{\partial N_1}{\partial\xi} & \frac{\partial N_3}{\partial\xi}\frac{\partial N_2}{\partial\xi} & \frac{\partial N_3}{\partial\xi}\frac{\partial N_3}{\partial\xi}\end{bmatrix}_{\xi=\xi_k}A(\xi_k)\right]$$

Values at Gaussian Quad Points (ξ_k)

Example (6/7): Matrix (2/3)

```
for(ip=0;ip<2;ip++){
  dNdQ[0]= -0.5 + POI[ip];
  dNdQ[1]= -2.0 * POI[ip];
  dNdQ[2]=  0.5 + POI[ip];

  XX= X0 + POI[ip]*0.50*DL;
  Area= A1*XX + A2;

 if(Area<= 0.) {
       fprintf(stderr, "ERROR: Area<0: ¥n");
       return -1;
 }

  Jacobi= fabs(dNdQ[0]*X1 + dNdQ[1]*X2 + dNdQ[2]*X3);
  Ck= Area*Young/Jacobi;
  Emat[0][0]= Emat[0][0] + C
  Emat[0][1]= Emat[0][1] + C
  Emat[0][2]= Emat[0][2] + Ck * WEI[ip] * dNdQ[0] * dNdQ[2];
  Emat[1][0]= Emat[1][0] + Ck * WEI[ip] * dNdQ[1] * dNdQ[0];
  Emat[1][1]= Emat[1][1] + Ck * WEI[ip] * dNdQ[1] * dNdQ[1];
  Emat[1][2]= Emat[1][2] + Ck * WEI[ip] * dNdQ[1] * dNdQ[2];
  Emat[2][0]= Emat[2][0] + Ck * WEI[ip] * dNdQ[2] * dNdQ[0];
  Emat[2][1]= Emat[2][1] + Ck * WEI[ip] * dNdQ[2] * dNdQ[1];
  Emat[2][2]= Emat[2][2] + Ck * WEI[ip] * dNdQ[2] * dNdQ[2];
}
```

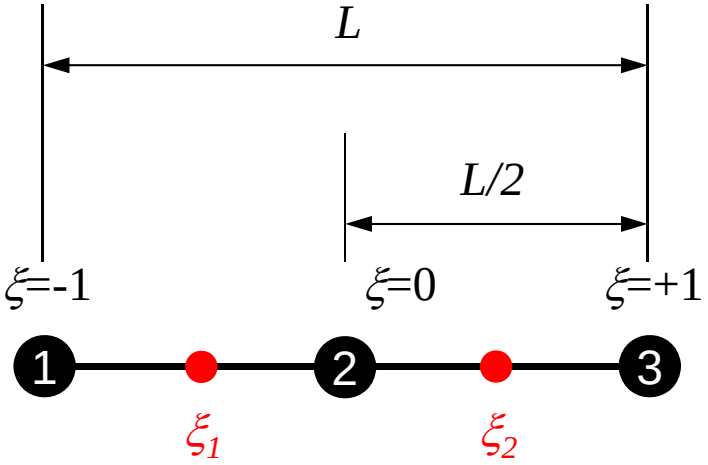


XX: Global Coordinate for Gaussian Quad. Point
Area: Sectional Area at XX

X-Coord. or Gaussian Quad. Points

2nd-Order/Quadratic Element

- According to definitions of "isoparametric" elements:



$$X(\xi)=\sum_{k=1}^{3} N_k(\xi)X_k$$

$$= N_1(\xi)X_1 + N_2(\xi)X_2 + N_3(\xi)X_3$$

$$= \frac{1}{2}\xi(-1+\xi)X_1 + (1-\xi^2)X_2 + \frac{1}{2}\xi(1+\xi)X_3$$

$$= X_2 + \frac{X_3 - X_1}{2}\xi + \frac{X_1 - 2X_2 + X_3}{2}\xi^2$$

$$= \frac{X_1 + X_3}{2} + \frac{X_3 - X_1}{2}\xi \quad \because X_2 = \frac{X_1 + X_3}{2}$$

∴ XX= XO + POI[ip]*0.50*DL;

Example (6/7): Matrix (2/3)

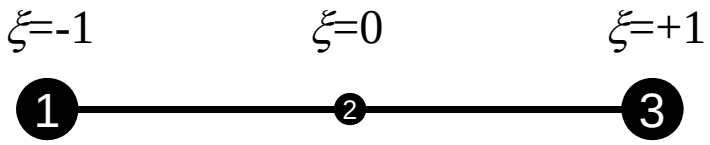
```
for(ip=0;ip<2;ip++){
  dNdQ[0]= -0.5 + POI[ip];
  dNdQ[1]= -2.0 * POI[ip];
  dNdQ[2]=  0.5 + POI[ip];

  XX= X0 + POI[ip]*0.50*DL;
  Area= A1*XX + A2;

 if(Area<= 0.) {
       fprintf(stderr, "ERROR: Area<0: ¥n")
       return -1;
 }

  Jacobi= fabs(dNdQ[0]*X1 + dNdQ[1]*X2 + dNdQ[2]*X3);
  Ck= Area*Young/Jacobi;
  Emat[0][0]= Emat[0][0] + Ck * WEI[ip] * dNdQ[0] * dNdQ[0];
  Emat[0][1]= Emat[0][1] + Ck * WEI[ip] * dNdQ[0] * dNdQ[1];
  Emat[0][2]= Emat[0][2] + Ck * WEI[ip] * dNdQ[0] * dNdQ[2];
  Emat[1][0]= Emat[1][0] + Ck * WEI[ip] * dNdQ[1] * dNdQ[0];
  Emat[1][1]= Emat[1][1] + Ck * WEI[ip] * dNdQ[1] * dNdQ[1];
  Emat[1][2]= Emat[1][2] + Ck * WEI[ip] * dNdQ[1] * dNdQ[2];
  Emat[2][0]= Emat[2][0] + Ck * WEI[ip] * dNdQ[2] * dNdQ[0];
  Emat[2][1]= Emat[2][1] + Ck * WEI[ip] * dNdQ[2] * dNdQ[1];
  Emat[2][2]= Emat[2][2] + Ck * WEI[ip] * dNdQ[2] * dNdQ[2];
}
```

$$Jacobi = \left|\frac{\partial x}{\partial \xi}\right| = \left|\frac{\partial}{\partial \xi}\sum_{i=1}^{3}\left(N_i x_i\right)\right|$$

$$= \left|\sum_{i=1}^{3}\left(\frac{\partial N_i}{\partial \xi}x_i\right)\right| = \left|\frac{\partial N_1}{\partial \xi}x_1 + \frac{\partial N_2}{\partial \xi}x_2 + \frac{\partial N_3}{\partial \xi}x_3\right|$$

Jacobian at Gaussian Quad. Points

- Jacobian



$$\frac{\partial x}{\partial \xi} = \frac{\partial}{\partial \xi}\sum_{i=1}^{3}\left(N_i x_i\right) = \sum_{i=1}^{3}\left(\frac{\partial N_i}{\partial \xi}x_i\right) = \frac{\partial N_1}{\partial \xi}x_1 + \frac{\partial N_2}{\partial \xi}x_2 + \frac{\partial N_3}{\partial \xi}x_3$$

- Jacobian at Gaussian Quad Points (ξ_k)

$$\left.\frac{\partial x}{\partial \xi}\right|_{\xi=\xi_k} = \left.\frac{\partial N_1}{\partial \xi}\right|_{\xi=\xi_k} x_1 + \left.\frac{\partial N_2}{\partial \xi}\right|_{\xi=\xi_k} x_2 + \left.\frac{\partial N_3}{\partial \xi}\right|_{\xi=\xi_k} x_3$$

Example (6/7): Matrix (2/3)

```
for(ip=0;ip<2;ip++){
  dNdQ[0]= -0.5 + PO
  dNdQ[1]= -2.0 * PO
  dNdQ[2]=  0.5 + PO

  XX= X0 + POI[ip]*0.
  Area= A1*XX + A2;

 if(Area<= 0.) {
       fprintf(stderr
       return -1;
 }

  Jacobi= fabs(dNdQ[0]*X1 + dNdQ[1]*X2 + dNdQ[2]*X3);
  Ck= Area*Young/Jacobi;
  Emat[0][0]= Emat[0][0] + Ck * WEI[ip] * dNdQ[0] * dNdQ[0];
  Emat[0][1]= Emat[0][1] + Ck * WEI[ip] * dNdQ[0] * dNdQ[1];
  Emat[0][2]= Emat[0][2] + Ck * WEI[ip] * dNdQ[0] * dNdQ[2];
  Emat[1][0]= Emat[1][0] + Ck * WEI[ip] * dNdQ[1] * dNdQ[0];
  Emat[1][1]= Emat[1][1] + Ck * WEI[ip] * dNdQ[1] * dNdQ[1];
  Emat[1][2]= Emat[1][2] + Ck * WEI[ip] * dNdQ[1] * dNdQ[2];
  Emat[2][0]= Emat[2][0] + Ck * WEI[ip] * dNdQ[2] * dNdQ[0];
  Emat[2][1]= Emat[2][1] + Ck * WEI[ip] * dNdQ[2] * dNdQ[1];
  Emat[2][2]= Emat[2][2] + Ck * WEI[ip] * dNdQ[2] * dNdQ[2];
}
```

$$[Emat] = E\sum_{k=1}^{m}\left[w_k \cdot \frac{1}{|J|}\Big|_{\xi=\xi_k} \left[\begin{array}{ccc} \frac{\partial N_1}{\partial \xi}\frac{\partial N_1}{\partial \xi} & \frac{\partial N_1}{\partial \xi}\frac{\partial N_2}{\partial \xi} & \frac{\partial N_1}{\partial \xi}\frac{\partial N_3}{\partial \xi} \\ \frac{\partial N_2}{\partial \xi}\frac{\partial N_1}{\partial \xi} & \frac{\partial N_2}{\partial \xi}\frac{\partial N_2}{\partial \xi} & \frac{\partial N_2}{\partial \xi}\frac{\partial N_3}{\partial \xi} \\ \frac{\partial N_3}{\partial \xi}\frac{\partial N_1}{\partial \xi} & \frac{\partial N_3}{\partial \xi}\frac{\partial N_2}{\partial \xi} & \frac{\partial N_3}{\partial \xi}\frac{\partial N_3}{\partial \xi} \end{array} \right]\Big|_{\xi=\xi_k} A(\xi_k) \right]$$

Example (7/7): Matrix (3/3)

Same procedures in “1d2.c”

```
      Diag[in1]= Diag[in1] + Emat[0][0];
      Diag[in2]= Diag[in2] + Emat[1][1];
      Diag[in3]= Diag[in3] + Emat[2][2];

      if (icel==0){k12=Index[in1];
                   k13=Index[in1]+1;
           }else {k12=Index[in1]+2;
                   k13=Index[in1]+3;}

        k21=Index[in2];
        k23=Index[in2]+1;

        k31=Index[in3];
        k32=Index[in3]+1;

        AMat[k12]= AMat[k12] + Emat[0][1];
        AMat[k13]= AMat[k13] + Emat[0][2];
        AMat[k21]= AMat[k21] + Emat[1][0];
        AMat[k23]= AMat[k23] + Emat[1][2];
        AMat[k31]= AMat[k31] + Emat[2][0];
        AMat[k32]= AMat[k32] + Emat[2][1];
    }
```

1st row: corresponds to *in1*

2nd row: corresponds to *in2*

3rd row: corresponds to *in3*

$$\begin{bmatrix} - & k12 & k13 \\ k21 & - & k23 \\ k31 & k32 & - \end{bmatrix}$$

Example (7/7): Matrix (3/3)

Same procedures in “1d2.c”

```
      Diag[in1]= Diag[in1] + Emat[0][0];
      Diag[in2]= Diag[in2] + Emat[1][1];
      Diag[in3]= Diag[in3] + Emat[2][2];

      if (icel==0){k12=Index[in1];
                   k13=Index[in1]+1;
           }else {k12=Index[in1]+2;
                   k13=Index[in1]+3;}

        k21=Index[in2];
        k23=Index[in2]+1;

        k31=Index[in3];
        k32=Index[in3]+1;

        AMat[k12]= AMat[k12] + Emat[0][1];
        AMat[k13]= AMat[k13] + Emat[0][2];
        AMat[k21]= AMat[k21] + Emat[1][0];
        AMat[k23]= AMat[k23] + Emat[1][2];
        AMat[k31]= AMat[k31] + Emat[2][0];
        AMat[k32]= AMat[k32] + Emat[2][1];
    }
```

1st row: corresponds to *in1*

2nd row: corresponds to *in2*

3rd row: corresponds to *in3*

$$\begin{bmatrix} - & k12 & k13 \\ k21 & - & k23 \\ k31 & k32 & - \end{bmatrix}$$

Winter Semester: Parallel FEM

Technical & Scientific Computing II
Seminar on Computer Science II

- Contents
  - Parallel Programming using MPI
  - Data Structure for Parallel FEM
  - Implementation of Parallel FEM
  - Exercises using Fujitsu PRIMEHPC FX10 (Oakleaf-FX)

- “Parallelize” fem3d code for 3D static linear-elastic problems in this semester.

Technical & Scientific Computing I, II (Finite Element Method)

- Instructor: Kengo Nakajima
- Graduate Level
- Semester & Credits
  - I : Summer, 2-Credits
  - II: Winter, 2-Credits

- Overview
  - (I): Fundamental issues of finite-element method (FEM) on static linear-elastic problems, including linear equation solvers and programming.
  - (II): Data structure for parallel FEM, implementation of parallel FEM, framework for development of parallel codes, such as “HPC-MW” and “ppOpen-HPC” with Fujitsu PRIMEHPC FX10 (Oakleaf-FX)
  - **Strong collaborations between science & engineering, computer science and numerical algorithms are required towards success of large-scale scientific simulations using parallel computers. Goal of these classes is that students of graduate school of information science and technologies understand requirement of applications and try to develop new interdisciplinary research area.**
  - Grade based on reports

Summer (I)

1. Fundamental Theory’s for FEM, Static Linear-Elastic Problem
2. FEM by Galerkin Method
3. Sparse Linear Solvers, Preconditioners
4. FEM Programming
   - 1D, 3D
5. ECCS 2012 System of ITC

Winter (II)

1. Parallel Programming using MPI
2. Data Structure for Parallel FEM
3. Implementation of Parallel FEM
4. Framework for Development of Parallel Simulation Codes using Large-scale Systems
5. Fujitsu PRIMEHPC FX10 (Oakleaf-FX)