一次元弾性解析コード（2/2）

2012年夏学期

中島 研吾

科学技術計算Ⅰ（4820-1027）・コンピュータ科学特別講義Ⅰ（4810-1204）

- ガラーキン法による一次元弾性問題の解法

- 連立一次方程式の解法
  - 共役勾配法
  - 前処理手法
- 疎行列格納法

- プログラムの内容

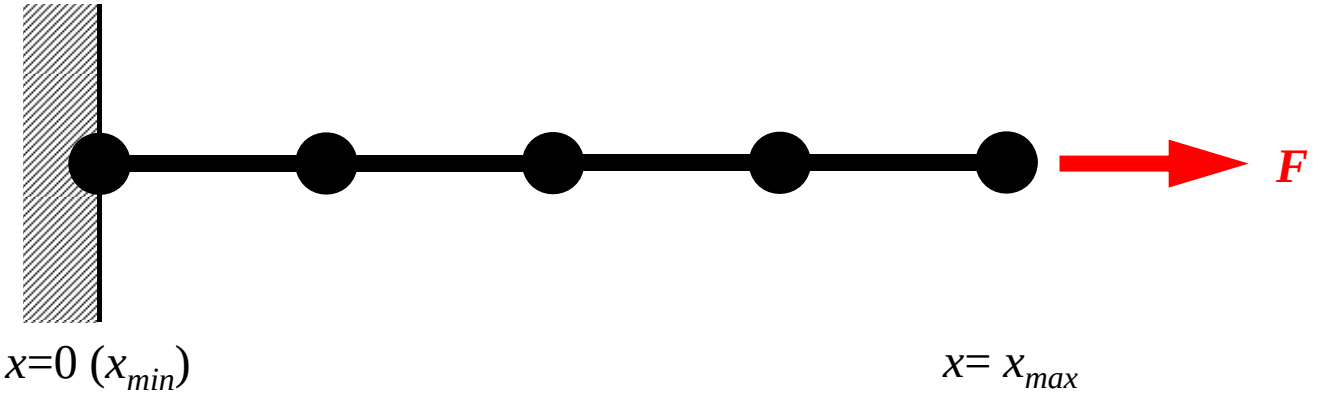
- 高次要素
- 数値積分法，アイソパラメトリック要素
- レポート課題1

- 断面形状が変化する場合
  - <$fem1>/1darea/a1.c
- 二次要素
  - <$fem1>/1d/1d2.c

対象とする問題：一次元弾性体（トラス）

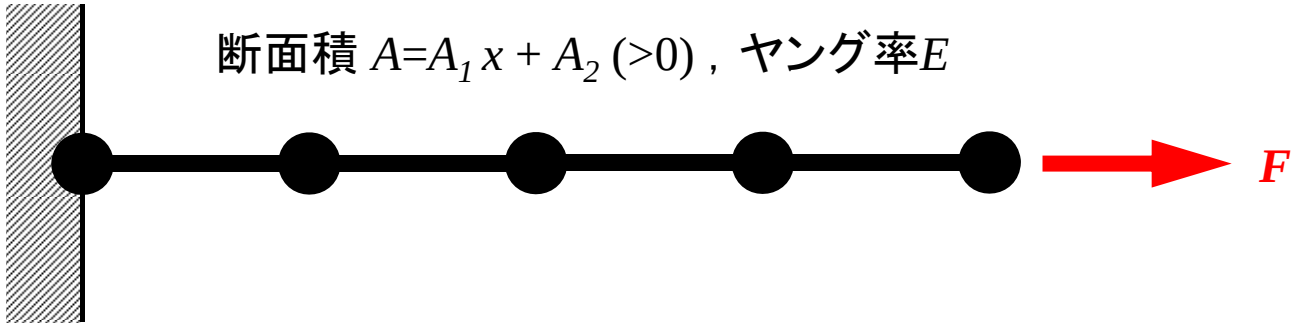
- x方向にのみ自由度（変位u）
  - 一様な：ヤング率E
  - 断面積：$A=A_1 x + A_2$ (>0)
  - 境界条件
    - $x=0$　：$u=0$（固定）
    - $x=x_{max}$：大きさFの力（軸力）
- 自重によるたわみ等はナシ：バネと同じ

対象とする問題：一次元弾性体（トラス）



- x方向にのみ自由度（変位u）
  - 一様な：ヤング率E
  - 断面積：$A=A_1 x + A_2$ (>0)
  - 境界条件
    - $x=0$　：$u=0$（固定）
    - $x=x_{max}$：大きさFの力（軸力）
- 自重によるたわみ等はナシ：バネと同じ

対象とする問題：一次元弾性体（トラス）



つりあい式 $$\frac{\partial \sigma_x}{\partial x} + X = 0$$

ひずみ～変位関係式 $$\varepsilon_x = \frac{\partial u}{\partial x}$$

ひずみ～応力関係式 $$\sigma_x = E\varepsilon_x$$

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

変位uを自由度とする支配方程式

計算の手順

- まず変位を求め $\frac{\partial}{\partial x}\left(E\frac{\partial u}{\partial x}\right)+X=0$
- 「ひずみ」を計算し $\varepsilon_x=\frac{\partial u}{\partial x}$
- 「応力」を計算する $\sigma_x=E\varepsilon_x$

解析解

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

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

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

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

ファイルコピー

一次元弾性解析コード

```
>$ cd <$fem1>
>$ cp /home03/skengon/Documents/class/fem1/1d2.tar .
>$ tar xvf 1d2.tar
>$ cd 1darea
```

実行

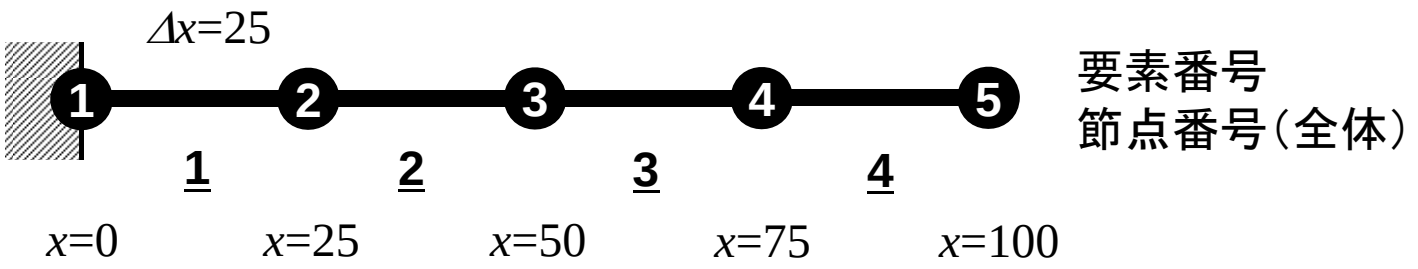
```
>$ cd <$fem1>/1darea
>$ cc –O a1.c      (or g95 –O a1.f)
>$ ./a.out
```

制御ファイル　input.dat

```
4                                          NE（要素数）
25.0  5.e4  -0.105 12  5.e6                Δx（要素長さL）, F, A1, A2, E
100                                        反復回数（CG法後述）
1.e-8                                      CG法の反復打切誤差
```

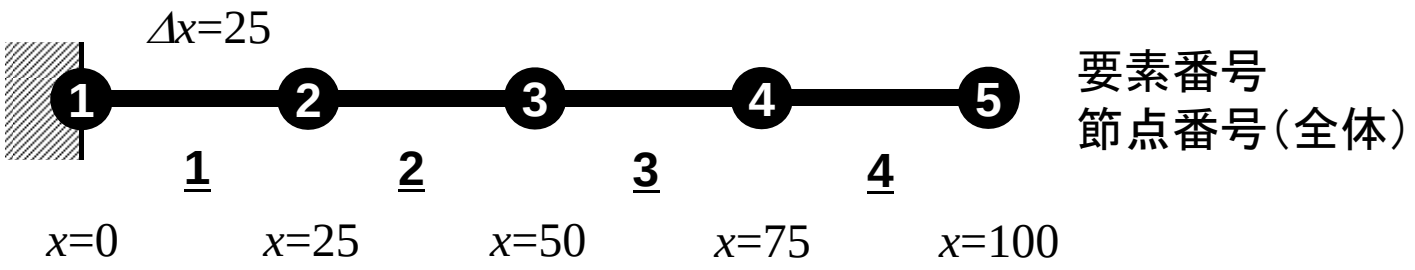
$$x = 0 \quad A_1 x + A_2 = 12.$$

$$x = 100 \quad A_1 x + A_2 = 1.5$$



結果：結構違う・・・

```
>$ ./a.out

  4 iters, RESID=    2.079723E-16 U(N)=     1.892655E-01

### DISPLACEMENT 結果(変位)
       1    0.000000E+00   -0.000000E+00
       2    2.339181E-02    2.351048E-02
       3    5.439956E-02    5.479659E-02
       4    1.003766E-01    1.016991E-01
       5    1.892655E-01    1.980421E-01

                 計算結果          解析解
```



プログラム:a1.c(1/2)
諸変数

```
/*
//   1D Solid Mechanics for Truss Elements solved by
//   CG (Conjugate Gradient) Method
//
//   d/dx(EdU/dx) + F = 0
//   U=0@x=0
*/

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

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

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

        int *Index, *Item, *IcelNod;
        double Kmat[2][2], Emat[2][2];

        int i, j, in1, in2, k, icel, k1, k2, jS;
        int iter;
        FILE *fp;
        double BNorm2, Rho, Rho1=0.0, C1, Alpha, DNorm2;
        int ierr = 1;
```

変数表(1/2)

変数名	種別	サイズ	I/O	内　容
NE	I		I	要素数
N	I		O	節点数
NPLU	I		O	非零非対角成分数
IterMax	I		I	最大反復回数
errno	I		O	エラー戻り値
R, Z, Q, P, DD	I		O	CG法ベクトル名
dX	R		I	要素長さ
Resid	R		O	CG法残差
Eps	R		I	CG法反復打ち切り残差
Area, A1, A2	R		I	要素断面積（Area= A1*x+A2）
F	R		I	軸力F（境界条件）
Young	R		I	ヤング率
XX	R		O	要素中心の座標
X1,X2,X3,U1,U2,U3	R		O	局所節点1,2,3の座標，変位

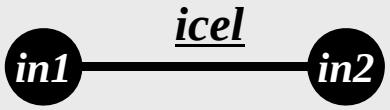
変数表(2/2)

変数名	種別	サイズ	I/O	内　容
DL, Ck	R		O	要素マトリクス計算用定数
Disp	R		O	節点変位
Strain, Stress	R		O	要素ひずみ，要素応力
X	R	N	O	節点座標
U	R	N	O	節点変位
Rhs	R	N	O	右辺ベクトル
Diag	R	N	O	全体マトリクス：対角成分
W	R	[4][N]	O	CG法のwork配列
Amat	R	NPLU	O	全体マトリクス：非零非対角成分
Index	I	N+1	O	全体マトリクス：各行の非零非対角成分数
Item	I	NPLU	O	全体マトリクス：列番号
Icelnod	I	2*NE	O	各要素節点番号
Kmat	R	[2][2]	O	要素マトリクス[k]
Emat	R	[2][2]	O	要素マトリクス

プログラム:a1.c(2/2)

全体マトリクス生成:要素マトリクス⇒全体マトリクス

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

          XX = 0.5 * (X1+X2);
          Area= A1*XX + A2;

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

          Ck= Area*Young/DL;
          Emat[0][0]= Ck*Kmat[0][0];
          Emat[0][1]= Ck*Kmat[0][1];
          Emat[1][0]= Ck*Kmat[1][0];
          Emat[1][1]= Ck*Kmat[1][1];
```



要素単位での積分 : $[k]$

$$\int_V E\left(\frac{d[N]^T}{dx}\frac{d[N]}{dx}\right)dV$$

$$N_i = \left(\frac{X_j - x}{L}\right), \quad N_j = \left(\frac{x - X_i}{L}\right)$$

$$= E\int_{X_i}^{X_j}\begin{bmatrix} -1/L \\ 1/L \end{bmatrix}\begin{bmatrix} -1/L, 1/L \end{bmatrix} A\, dx$$

$$\frac{dN_i}{dx} = \left(\frac{-1}{L}\right), \quad \frac{dN_j}{dx} = \left(\frac{1}{L}\right)$$

$$= \frac{E}{L^2}\int_{X_i}^{X_j}\begin{bmatrix} +1 & -1 \\ -1 & +1 \end{bmatrix} A dx = \frac{E}{L^2}\begin{bmatrix} +1 & -1 \\ -1 & +1 \end{bmatrix}\int_{X_i}^{X_j}\left(A_1 x + A_2\right) dx$$

$$= \frac{E}{L^2}\begin{bmatrix} +1 & -1 \\ -1 & +1 \end{bmatrix}\left[\frac{1}{2}A_1 x^2 + A_2 x\right]_{X_i}^{X_j}$$

$$= \frac{E\left(\frac{1}{2}A_1\left(X_j{}^2 - X_i{}^2\right) + A_2\left(X_j - X_i\right)\right)}{L^2}\begin{bmatrix} +1 & -1 \\ -1 & +1 \end{bmatrix}$$

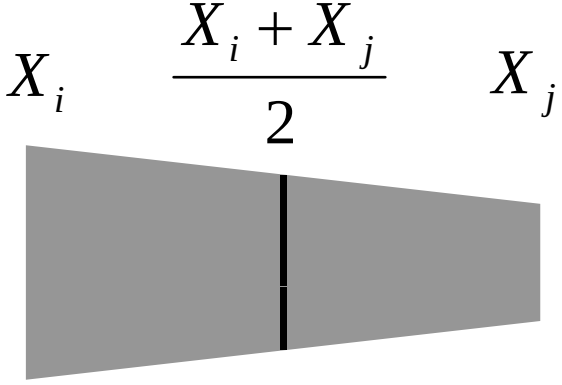
X=L/2での面積をかけた値

$$\frac{E\left(\frac{1}{2}A_1\left(X_j{}^2-X_i{}^2\right)+A_2\left(X_j-X_i\right)\right)}{L^2}\begin{bmatrix}+1 & -1\\ -1 & +1\end{bmatrix}$$

$$=\frac{E\left(\frac{1}{2}A_1\left(X_i+X_j\right)\left(X_j-X_i\right)+A_2\left(X_j-X_i\right)\right)}{L^2}\begin{bmatrix}+1 & -1\\ -1 & +1\end{bmatrix}$$

$$X_j-X_i=L$$

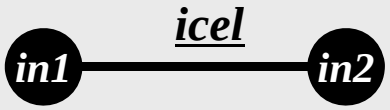
$$=\frac{E\left(A_1\frac{\left(X_i+X_j\right)}{2}+A_2\right)}{L}\begin{bmatrix}+1 & -1\\ -1 & +1\end{bmatrix}$$



プログラム:a1.c(2/2)

全体マトリクス生成:要素マトリクス⇒全体マトリクス

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

          XX = 0.5 * (X1+X2);
          Area= A1*XX + A2;

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

          Ck= Area*Young/DL;
          Emat[0][0]= Ck*Kmat[0][0];
          Emat[0][1]= Ck*Kmat[0][1];
          Emat[1][0]= Ck*Kmat[1][0];
          Emat[1][1]= Ck*Kmat[1][1];
```



$$[Emat]=[k]^{(e)}=\frac{E\cdot A(XX)}{L}\begin{bmatrix}+1 & -1\\ -1 & +1\end{bmatrix}$$

$$=\frac{E\cdot A(XX)}{L}[Kmat]$$

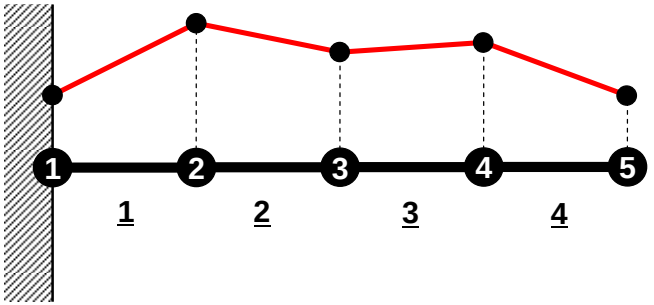
- 断面形状が変化する場合
  - <$fem1>/1darea/a1.c
- 二次要素
  - <$fem1>/1d/1d2.c

より精度をあげるには？

- メッシュを細かくする

```
NE=8, dX=12.5

8 iters, RESID=    2.822910E-16 U(N)=     1.953586E-01

### DISPLACEMENT
       1    0.000000E+00   -0.000000E+00
       2    1.101928E-02    1.103160E-02
       3    2.348034E-02    2.351048E-02
       4    3.781726E-02    3.787457E-02
       5    5.469490E-02    5.479659E-02
       6    7.520772E-02    7.538926E-02
       7    1.013515E-01    1.016991E-01
       8    1.373875E-01    1.381746E-01
       9    1.953586E-01    1.980421E-01
```

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



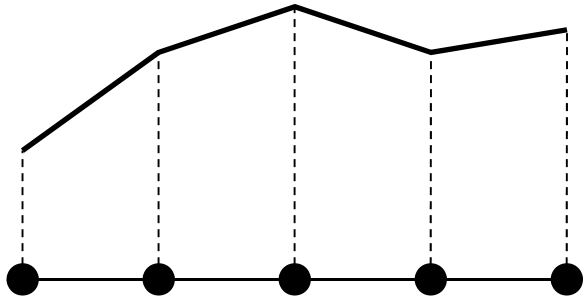

```
NE=20, dX=5

 20 iters, RESID=    5.707508E-15 U(N)=     1.975734E-01

### DISPLACEMENT
       1    0.000000E+00   -0.000000E+00
       2    4.259851E-03    4.260561E-03
       3    8.719160E-03    8.720685E-03
       4    1.339752E-02    1.339999E-02
……
      17    1.145876E-01    1.146641E-01
      18    1.295689E-01    1.296764E-01
      19    1.473466E-01    1.475060E-01
      20    1.692046E-01    1.694607E-01
      21    1.975734E-01    1.980421E-01
```

より精度をあげるには？

- メッシュを細かくする
- 高次の補間関数（形状関数）を使用する
  - 高次要素
  - 線形要素，一次要素は低次要素と呼ばれる
- n次微分係数の連続性を保証する定式化を適用する
  - C^n連続性

より精度をあげるには？

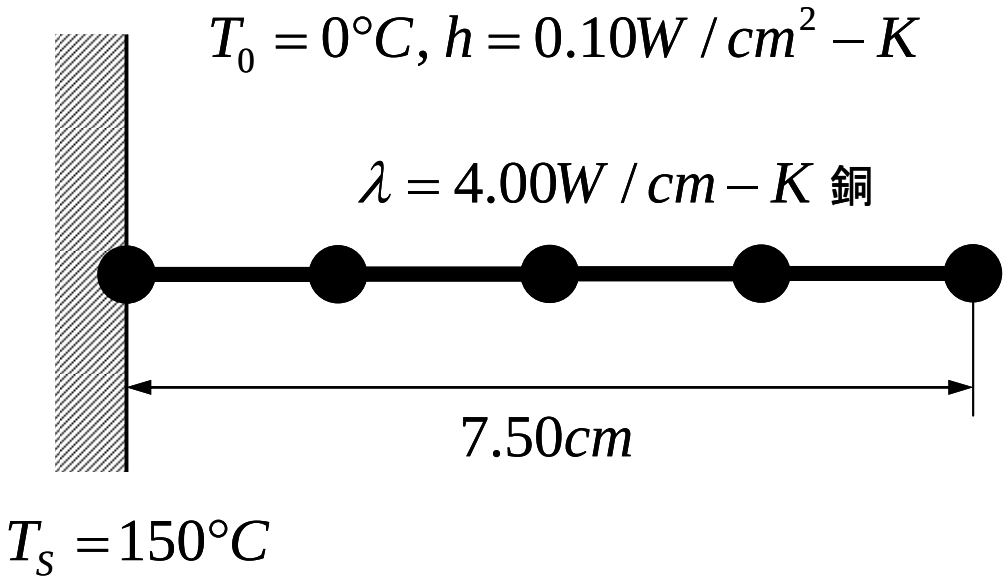
- メッシュを細かくする
- 高次の補間関数（形状関数）を使用する
- n次微分係数の連続性を保証する定式化を適用する
  - C^n連続性
- これまで紹介してきたのは：
  - 一次要素（線形要素）
    - 区分的一次近似（Piecewise Linear）
  - C^0連続
    - 従属変数（のみ）が要素境界で連続

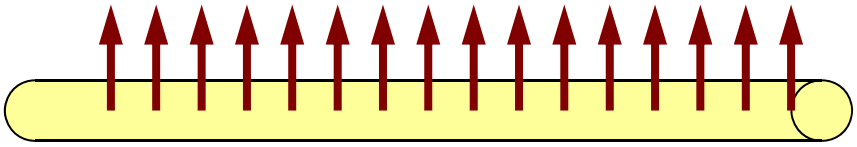


- これから話すのは：
  - 二次要素：曲線の近似により適している
    - 要素内で二次関数的な分布
  - C^0連続

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

例：熱伝導問題（1/2）





熱が円筒周囲表面から逃げていく

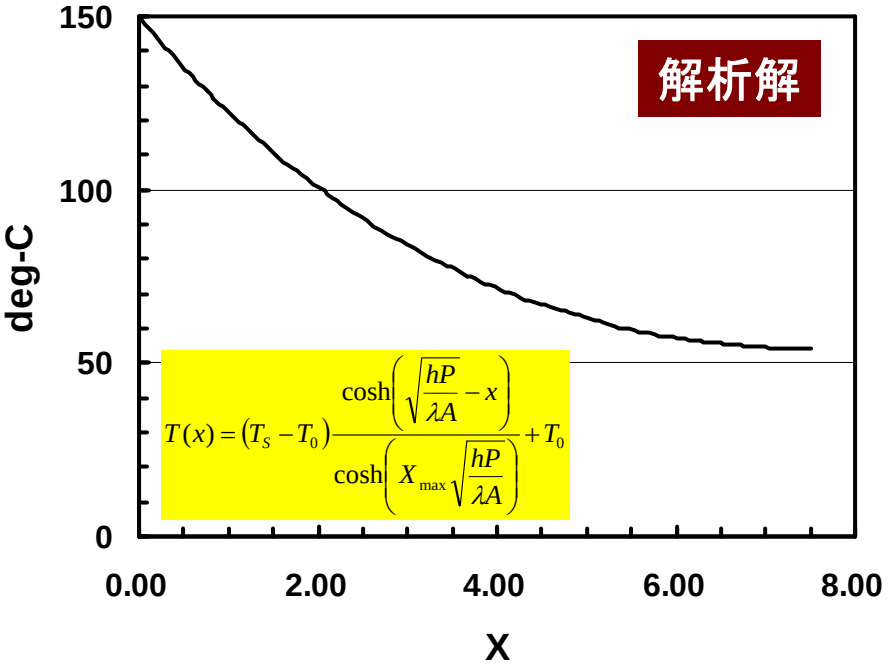
- フィンの温度分布
- 半径1cmの円形断面
- 端面境界条件
  - x=0　：温度固定
  - x=7.5：断熱
- 円筒周囲での対流熱伝達
  - $q = h(T-T_0)$
  - q：熱流束（Heat Flux）
    - 単位面積・単位時間当たり熱流量

例：熱伝導問題（2/2）

```
### RESULTS (linear interpolation)
ID        X          FEM.       ANALYTICAL
 1      0.00000    150.00000    150.00000    ERR(%):  0.00000
 2      1.87500    102.62226    103.00165    ERR(%):  0.25292
 3      3.75000     73.82803     74.37583    ERR(%):  0.36520
 4      5.62500     58.40306     59.01653    ERR(%):  0.40898
 5      7.50000     53.55410     54.18409    ERR(%):  0.41999
```

```
### RESULTS (quadratic interpolation)
ID        X          FEM.       ANALYTICAL
 1      0.00000    150.00000    150.00000    ERR(%):  0.00000
 2      1.87500    102.98743    103.00165    ERR(%):  0.00948
 3      3.75000     74.40203     74.37583    ERR(%):  0.01747
 4      5.62500     59.02737     59.01653    ERR(%):  0.00722
 5      7.50000     54.21426     54.18409    ERR(%):  0.02011
```

```
### RESULTS (linear interpolation)
ID        X          FEM.       ANALYTICAL
 1      0.00000    150.00000    150.00000    ERR(%):  0.00000
 2      0.93750    123.71561    123.77127    ERR(%):  0.03711
 3      1.87500    102.90805    103.00165    ERR(%):  0.06240
 4      2.81250     86.65618     86.77507    ERR(%):  0.07926
 5      3.75000     74.24055     74.37583    ERR(%):  0.09019
 6      4.68750     65.11151     65.25705    ERR(%):  0.09703
 7      5.62500     58.86492     59.01653    ERR(%):  0.10107
 8      6.56250     55.22426     55.37903    ERR(%):  0.10317
 9      7.50000     54.02836     54.18409    ERR(%):  0.10382
```



線形近似と比較すると，二次関数による内挿関数の方が高精度，解析解をよく近似している

特に x＝7.50cmに近づいた場合

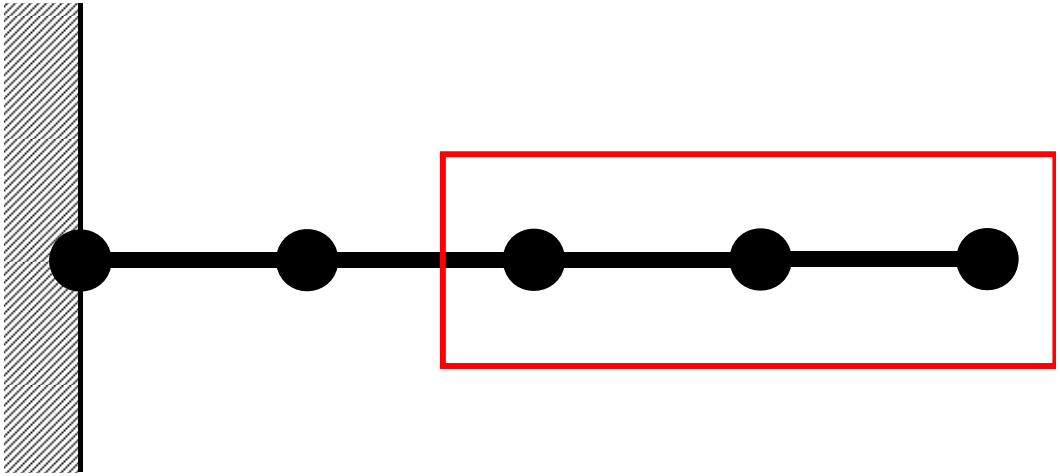
とりあえず最初の問題（断面積一定）に戻ってみよう

- x方向にのみ自由度（変位u）
  - 一様な：断面積A，ヤング率E
  - 境界条件
    - $x=0$　：$u=0$（固定）
    - $x=x_{max}$：大きさFの力（軸力）
- 自重によるたわみ等はナシ：バネと同じ

一次元二次要素の定式化（1/2）

one-dimensional quadratic element

- 長さL
- 線分の両端（i,k）とその中点（j）（中間節点）を節点とする要素



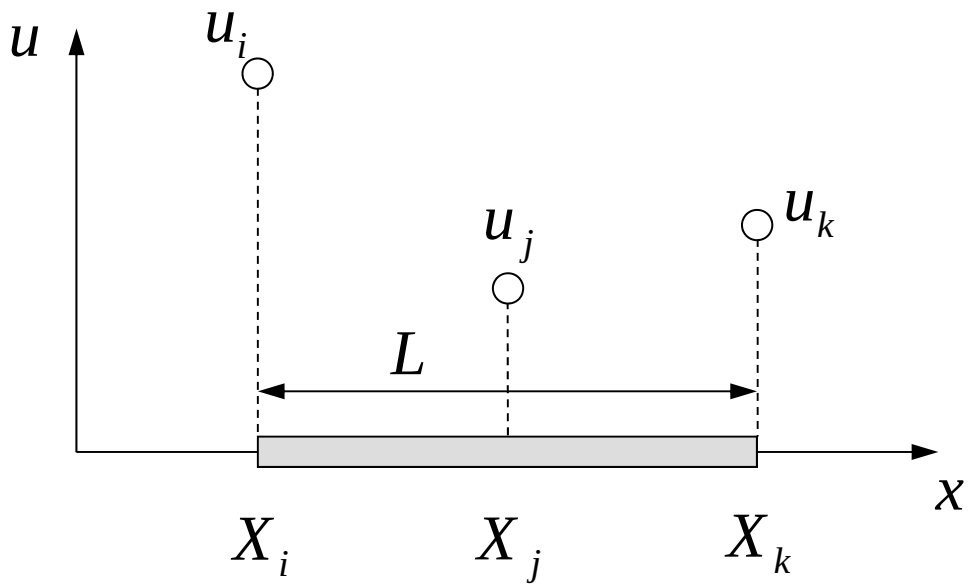
- 要素内の変位uは以下のように定義される：

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

$$u_i = \alpha_1 + X_i \alpha_2 + X_i^{\,2} \alpha_3$$

$$u_j = \alpha_1 + X_j \alpha_2 + X_j^{\,2} \alpha_3$$

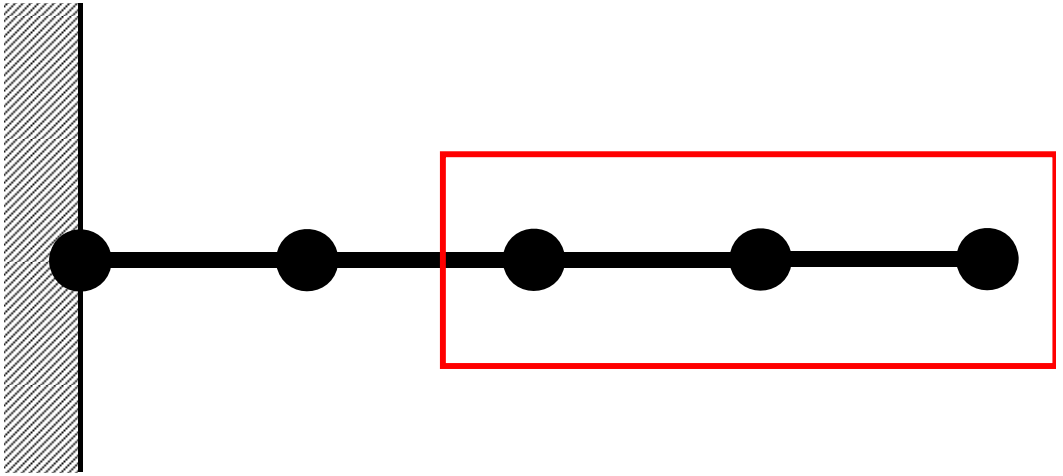
$$u_k = \alpha_1 + X_k \alpha_2 + X_k^{\,2} \alpha_3$$



一次元二次要素の定式化（1/2）

one-dimensional quadratic element

- 長さL
- 線分の両端（i,k）とその中点（j）（中間節点）を節点とする要素
  - $Xi=0$とする局所座標

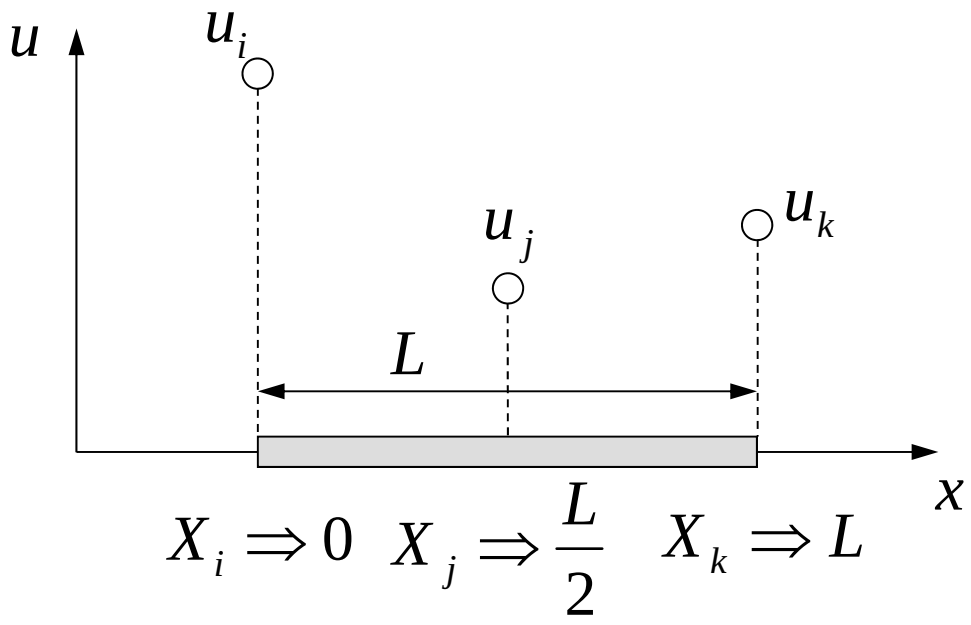


- 要素内の変位uは以下のように定義される：

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

$$u_i = \alpha_1, \quad u_j = \alpha_1 + \frac{L}{2}\alpha_2 + \frac{L^2}{4}\alpha_3$$

$$u_k = \alpha_1 + L\alpha_2 + L^2\alpha_3$$



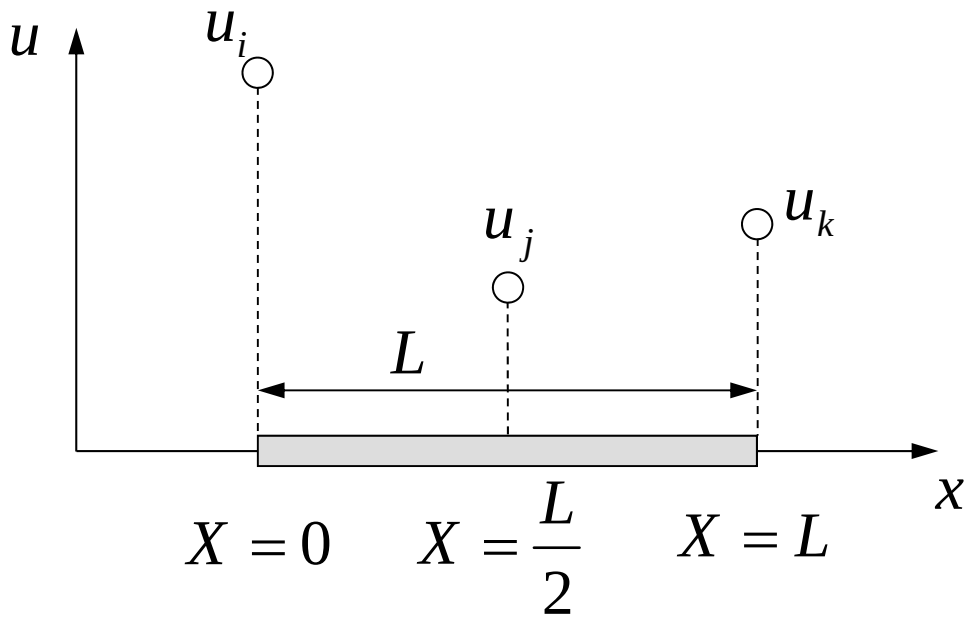
一次元二次要素の定式化（2/2）

one-dimensional quadratic element

- 節点での条件から，係数は以下のように求められる：

$$\alpha_1 = u_i,\ \alpha_2 = \frac{4u_i - 3u_j - u_k}{L},$$

$$\alpha_3 = \frac{2}{L^2}\left(u_i - 2u_j + u_k\right)$$



- 形状関数は以下のようになる

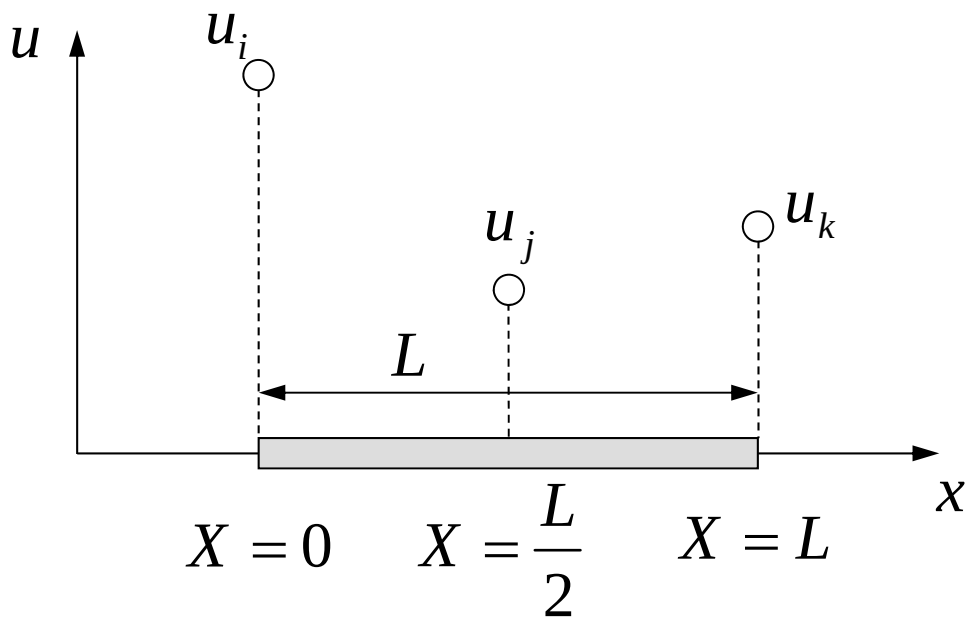
$$u = N_i u_i + N_j u_j + N_k u_k$$

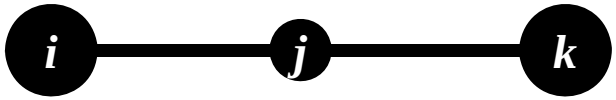
$$= \left(1 - \frac{2x}{L}\right)\left(1 - \frac{x}{L}\right)u_i + \left(\frac{4x}{L}\right)\left(1 - \frac{x}{L}\right)u_j + \left(-\frac{x}{L}\right)\left(1 - \frac{2x}{L}\right)u_k$$

X=0, X=L/2，X=Lでの値を確認してみよう

一次元二次要素

one-dimensional quadratic element



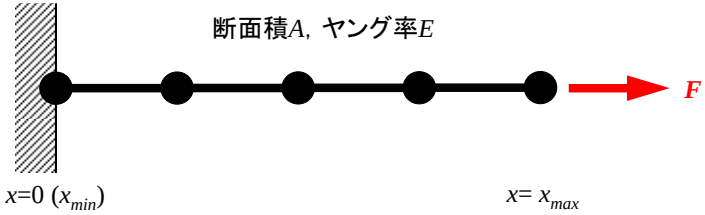


中間節点

ガラーキン法の適用（1/4）

- 以下のような一次元弾性力学方程式を考慮する：

$$E\left(\frac{d^2u}{dx^2}\right)+X=0$$



$u=[N]\{\phi\}$ 要素内の変位分布（マトリクス形式），節点における変位をϕとしてある。

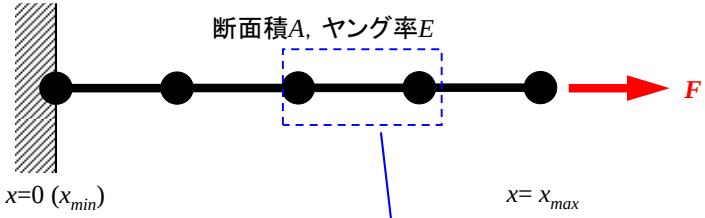
- ガラーキン法に従い，重み関数を[N]とすると，各要素において以下の積分方程式が得られる：

$$\int_V [N]^T\left\{E\left(\frac{d^2u}{dx^2}\right)+X\right\}dV=0$$

ガラーキン法の適用（2/4）

- 一次元のグリーンの定理

$$\int_V A\left(\frac{d^2B}{dx^2}\right)dV = \int_S A\frac{dB}{dx}\,dS - \int_V \left(\frac{dA}{dx}\frac{dB}{dx}\right)dV$$

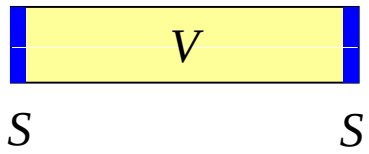


- これを前式の2階微分の部分に適用すると：

$$\int_V E[N]^T\left(\frac{d^2u}{dx^2}\right)dV = -\int_V E\left(\frac{d[N]^T}{dx}\frac{du}{dx}\right)dV + \int_S E[N]^T\frac{du}{dx}dS$$

- これに以下を代入する：

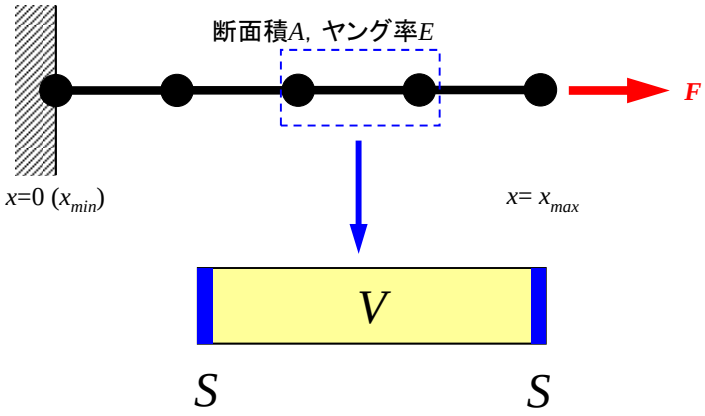
$$u = [N]\{\phi\} \qquad \bar{\sigma} = E\frac{du}{dx}$$ ：要素表面応力



ガラーキン法の適用（3/4）

- 更に体積力（物体力）の項Xを加えて次式が得られる：

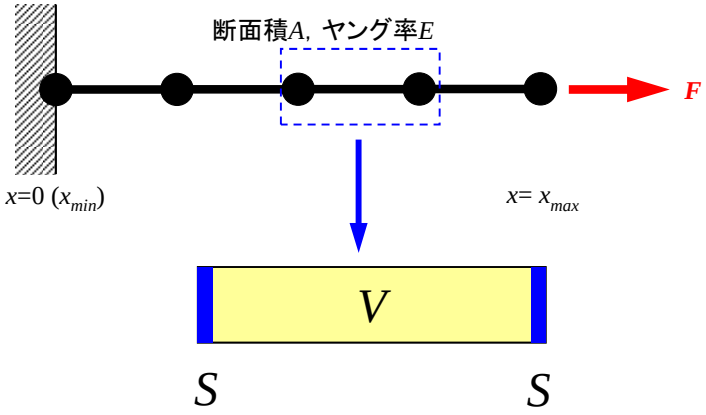
$$-\int_V E\left(\frac{d[N]^T}{dx}\frac{d[N]}{dx}\right)dV\cdot\{\phi\}$$
$$+\int_S \bar{\sigma}[N]^T dS+\int_V X[N]^T dV=0$$



- この式を弱形式（weak form）と呼ぶ。元の微分方程式では2階の微分が含まれていたが，上式では，グリーンの定理によって1階微分に低減されている。
  - 弱形式によって近似関数（形状関数，内挿関数）に対する要求が弱くなっている：すなわち線形関数で2階微分の効果を記述できる。

ガラーキン法の適用（4/4）

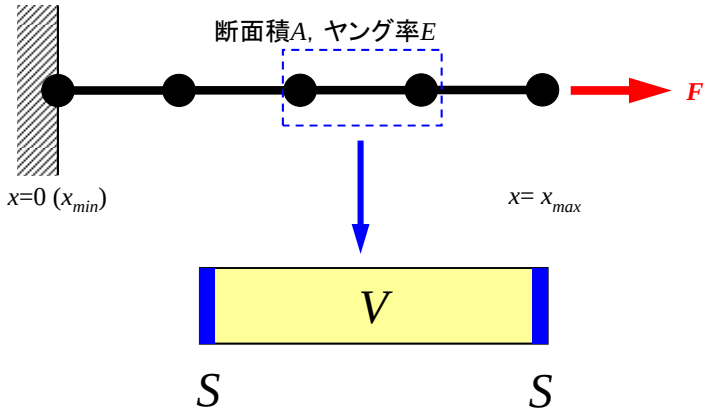
$$-\int_V E\left(\frac{d[N]^T}{dx}\frac{d[N]}{dx}\right)dV\cdot\{\phi\}$$
$$+\int_S \bar{\sigma}[N]^T dS+\int_V X[N]^T dV=0$$



- この項は要素境界で相殺するため，領域境界における項のみが残る。

弱形式と境界条件

- 未知数の値が直接与えられる（Dirichlet）
  - 重み関数＝0となる
  - 第一種境界条件
  - 基本境界条件
    - essential boundary condition
- 未知数の導関数が与えられる（Neumann）
  - 弱形式中で自然に考慮される
  - 第二種境界条件
  - 自然境界条件
    - natural boundary condition



$$-\int_V E\left(\frac{d[N]^T}{dx}\frac{d[N]}{dx}\right)dV\cdot\{\phi\}+\int_S \bar{\sigma}[N]^T dS+\int_V X[N]^T dV=0$$

$\bar{\sigma}=E\dfrac{du}{dx}$ から得られる

境界条件を考慮した弱形式：各要素

$$[k]^{(e)}\{\phi\}^{(e)} = \{f\}^{(e)}$$

$$[k]^{(e)} = \int_V E\left(\frac{d[N]^T}{dx}\frac{d[N]}{dx}\right)dV$$

$$[f]^{(e)} = \int_V X[N]^T dV + \int_S \bar{\sigma}[N]^T dS$$

ここまでは一次要素と全く同じ

要素単位での積分：[k]（1/2）

$$N_i = \left(1-\frac{2x}{L}\right)\left(1-\frac{x}{L}\right) \qquad \frac{dN_i}{dx} = \left(\frac{4x}{L^2}-\frac{3}{L}\right)$$

$$N_j = \left(\frac{4x}{L}\right)\left(1-\frac{x}{L}\right) \qquad \frac{dN_j}{dx} = \left(\frac{4}{L}-\frac{8x}{L^2}\right)$$

$$N_k = \left(-\frac{x}{L}\right)\left(1-\frac{2x}{L}\right) \qquad \frac{dN_k}{dx} = \left(\frac{4x}{L^2}-\frac{1}{L}\right)$$

微分係数は要素内で
単調な一次関数となる

要素単位での積分：[k]（2/2）

$$\int_V E\left(\frac{d[N]^T}{dx}\frac{d[N]}{dx}\right)dV = \int_0^L \begin{bmatrix} dN_i/dx \\ dN_j/dx \\ dN_k/dx \end{bmatrix} E\left[\frac{dN_i}{dx}, \frac{dN_j}{dx}, \frac{dN_k}{dx}\right] A\,dx$$

$$= EA\int_0^L \begin{bmatrix} \frac{dN_i}{dx}\frac{dN_i}{dx} & \frac{dN_i}{dx}\frac{dN_j}{dx} & \frac{dN_i}{dx}\frac{dN_k}{dx} \\ \frac{dN_j}{dx}\frac{dN_i}{dx} & \frac{dN_j}{dx}\frac{dN_j}{dx} & \frac{dN_j}{dx}\frac{dN_k}{dx} \\ \frac{dN_k}{dx}\frac{dN_i}{dx} & \frac{dN_k}{dx}\frac{dN_j}{dx} & \frac{dN_k}{dx}\frac{dN_k}{dx} \end{bmatrix} dx = \frac{EA}{6L}\begin{bmatrix} +14 & -16 & +2 \\ -16 & +32 & -16 \\ +2 & -16 & +14 \end{bmatrix}$$

要素単位での積分：{f}

$$\int_V X\left[N\right]^T dV = XA\int_0^L \begin{bmatrix} N_i \\ N_j \\ N_k \end{bmatrix} dx = XA\int_0^L \begin{bmatrix} 1-\frac{3x}{L}+\frac{2x^2}{L^2} \\ \frac{4x}{L}-\frac{4x^2}{L^2} \\ -\frac{x}{L}+\frac{2x^2}{L^2} \end{bmatrix} dx = \frac{XAL}{6}\begin{Bmatrix} 1 \\ 4 \\ 1 \end{Bmatrix}$$

1 : 4 : 1

物体力

線形要素のときは１：１だった

$$N_i = \left(\frac{X_j - x}{L} \right), \quad N_j = \left(\frac{x - X_i}{L} \right) \qquad \frac{dN_i}{dx} = \left(\frac{-1}{L} \right), \quad \frac{dN_j}{dx} = \left(\frac{1}{L} \right)$$

$$\int_V X \left[N \right]^T dV = XA \int_0^L \begin{bmatrix} 1 - x/L \\ x/L \end{bmatrix} dx = \frac{XAL}{2} \begin{Bmatrix} 1 \\ 1 \end{Bmatrix}$$

物体力

A：断面積，L：要素長さ

要素単位での積分： $\{f\}$

$$\int_V X[N]^T dV = XA\int_0^L \begin{bmatrix} N_i \\ N_j \\ N_k \end{bmatrix} dx = XA\int_0^L \begin{bmatrix} 1-\frac{3x}{L}+\frac{2x^2}{L^2} \\ \frac{4x}{L}-\frac{4x^2}{L^2} \\ -\frac{x}{L}+\frac{2x^2}{L^2} \end{bmatrix} dx = \frac{XAL}{6}\begin{Bmatrix} 1 \\ 4 \\ 1 \end{Bmatrix}$$

1 : 4 : 1

物体力

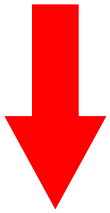
$$\int_V \bar{\sigma}[N]^T dS = \bar{\sigma}A\big|_{x=L} = \bar{\sigma}A\begin{Bmatrix} 0 \\ 0 \\ 1 \end{Bmatrix}$$

表面力

全体方程式

- 要素単位の方程式を全体で足し合わせ，

$$[k]^{(e)}\{\phi\}^{(e)} = \{f\}^{(e)}$$ 要素マトリクス，要素方程式



$$[K]\bullet\{\Phi\} = \{F\}$$ 全体マトリクス，全体方程式

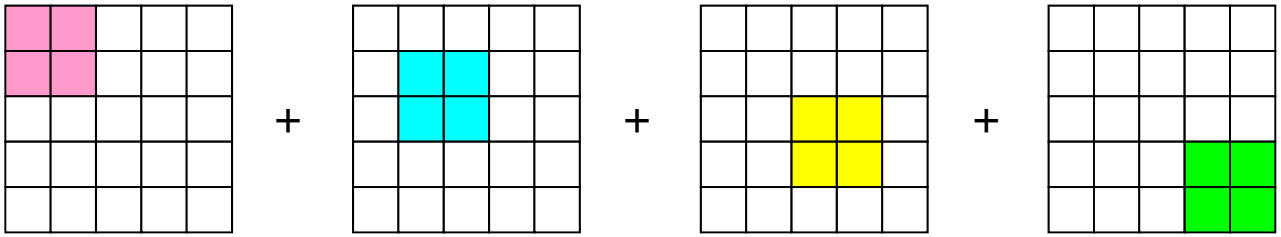
$$[K] = \sum [k], \quad \{F\} = \sum \{f\}$$

$$\{\Phi\}: global\ vector\ of\ \{\phi\}$$

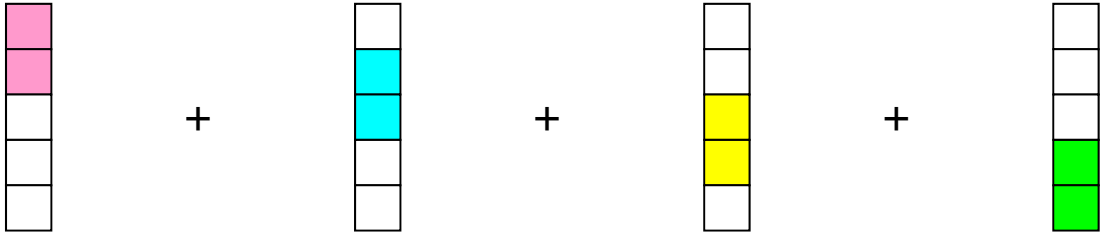
この連立一次方程式（全体方程式）
を解いてやればよい

要素方程式とその重ね合わせ
一次元線形要素

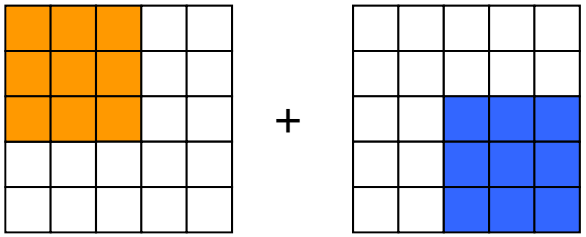
$$[K]=\sum_{i=1}^{4}\left[k^{(i)}\right]=$$

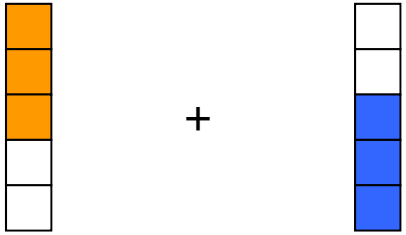


$$\{F\}=\sum_{i=1}^{4}\left\{f^{(i)}\right\}=$$



要素方程式とその重ね合わせ
一次元二次要素：要素数は2

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

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

実行

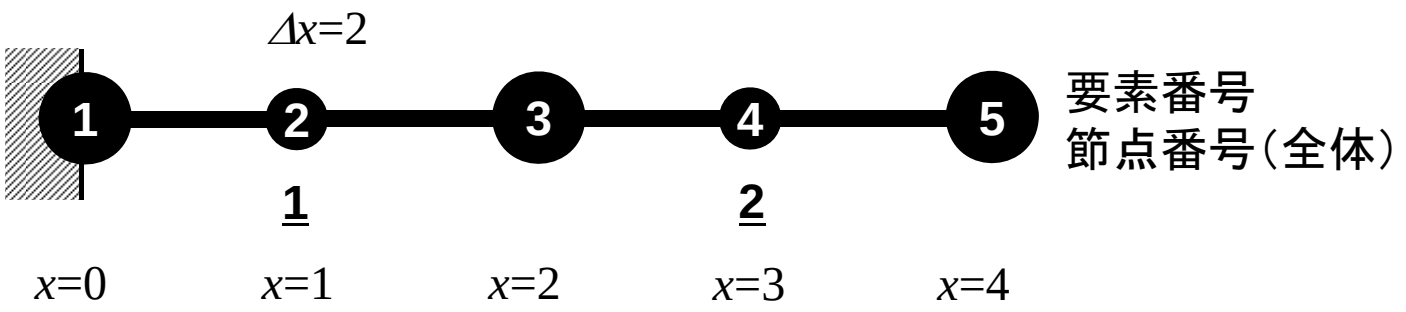
```
>$ cd <$fem1>/1d
>$ cc –O 1d2.c       (or g95 –O 1d2.f)
>$ ./a.out
```

制御ファイル　input2.dat

```
2                                   NE（要素数）
2.0  1.0  1.0  1.0                  Δx（要素長さL），F, A, E
100                                 反復回数（CG法後述）
1.e-8                               CG法の反復打切誤差
```

$$\sigma = \frac{F}{A} = \frac{1}{1} = 1$$

$$\frac{du}{dx} = \varepsilon = \frac{\sigma}{E} = \frac{1}{1} = 1$$



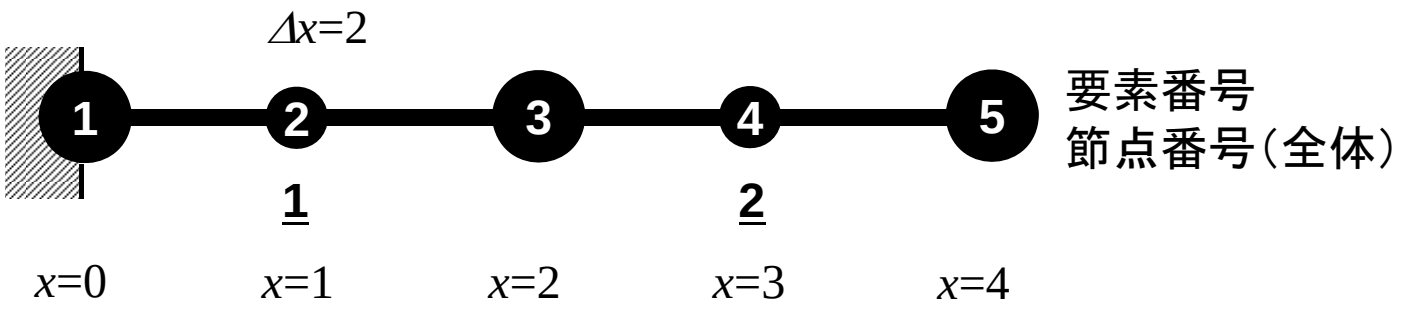
結果

```
>$ ./a.out

  4 iters, RESID=    1.914442e-15

### DISPLACEMENT   結果(変位)
        1    0.000000E+00    0.000000E+00
        2    1.000000E+00    1.000000E+00
        3    2.000000E+00    2.000000E+00
        4    3.000000E+00    3.000000E+00
        5    4.000000E+00    4.000000E+00
```

計算結果　　解析解

$$\sigma = \frac{F}{A} = \frac{1}{1} = 1$$

$$\frac{du}{dx} = \varepsilon = \frac{\sigma}{E} = \frac{1}{1} = 1$$



プログラム：1d2.c（1/7）

諸変数

```
/*
//   1D Solid Mechanics for Truss Elements solved by
//   CG (Conjugate Gradient) Method using 2nd-order Elements
//
//   d/dx(EdU/dx) + F = 0
//   U=0@x=0
*/
#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;

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

        int *Index, *Item, *IcelNod;
        double Kmat[3][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;
```

変数表(1/2)

変数名	種別	サイズ	I/O	内　容
NE	I		I	要素数
N	I		O	節点数
NPLU, NPLU0	I		O	非零非対角成分数
IterMax	I		I	最大反復回数
errno	I		O	エラー戻り値
R, Z, Q, P, DD	I		O	CG法ベクトル名
dX	R		I	要素長さ
Resid	R		O	CG法残差
Eps	R		I	CG法反復打ち切り残差
Area	R		I	要素断面積
F	R		I	軸力F（境界条件）
Young	R		I	ヤング率
X1,X2,U1,U2	R		O	局所節点1,2の座標，変位

変数表(2/2)

変数名	種別	サイズ	I/O	内　容
DL, Ck	R		O	要素マトリクス計算用定数
Strain, Stress	R		O	要素ひずみ，要素応力
X	R	N	O	節点座標
U	R	N	O	節点変位
Rhs	R	N	O	右辺ベクトル
Diag	R	N	O	全体マトリクス：対角成分
W	R	[4][N]	O	CG法のwork配列
Amat	R	NPLU	O	全体マトリクス：非零非対角成分
Index	I	N+1	O	全体マトリクス：各行の非零非対角成分数
Item	I	NPLU	O	全体マトリクス：列番号
Icelnod	I	3*NE	O	各要素節点番号
Kmat	R	[3][3]	O	要素マトリクス[k]
Emat	R	[3][3]	O	要素マトリクス

プログラム：1d2.c（2/7）

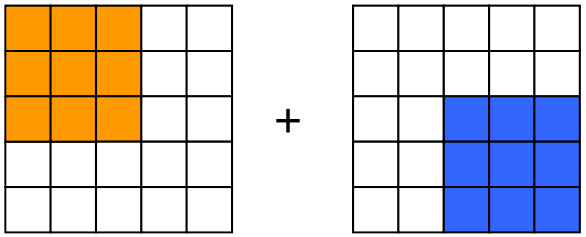
初期設定，配列宣言

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

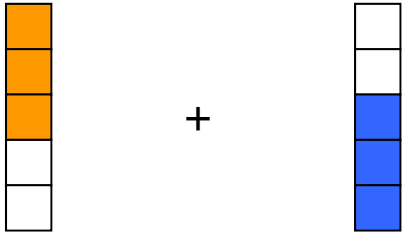
Amat： 非零非対角成分
Item： 対応する列番号

非零非対角成分
同じ要素に属する節点と関係を持つ

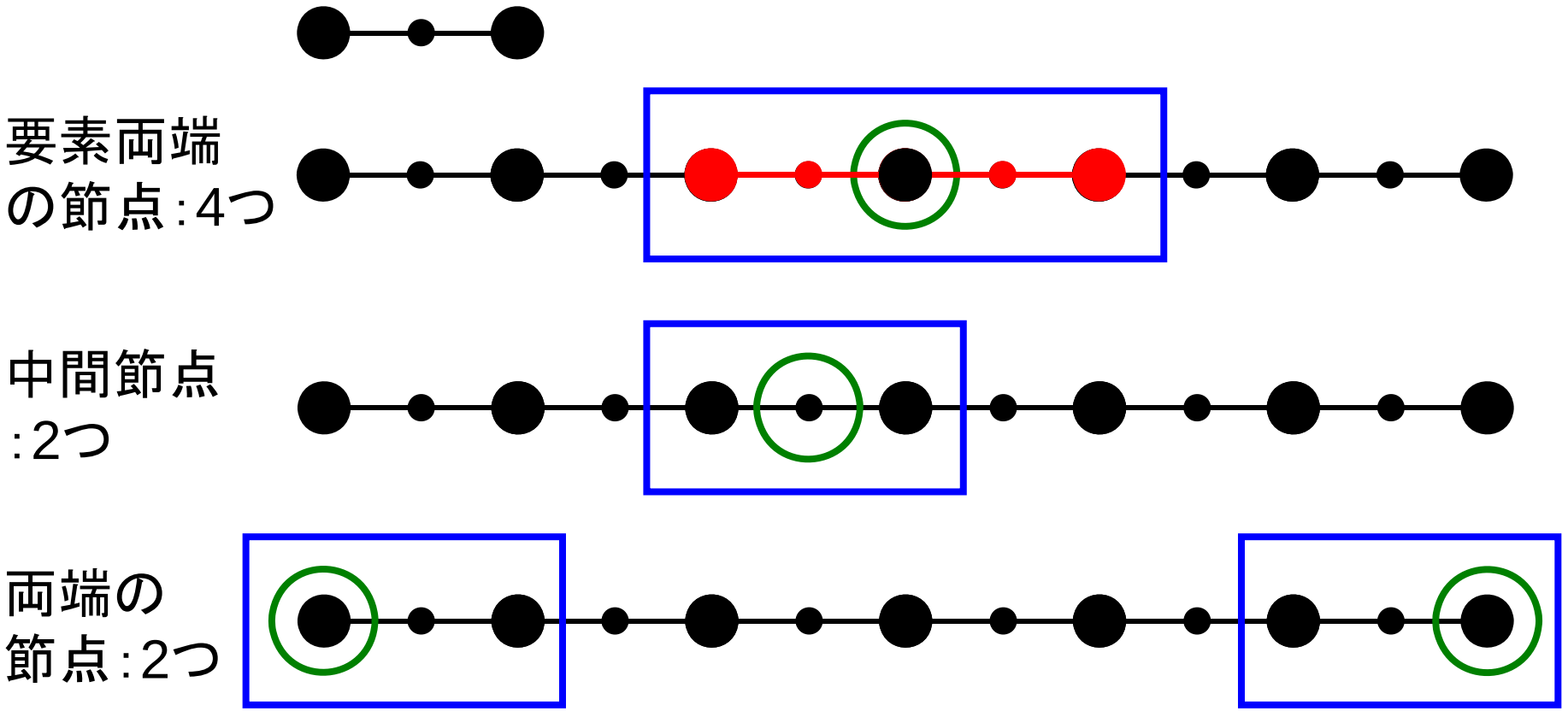
$$[K]=\sum_{i=1}^{2}\left[k^{(i)}\right]=$$



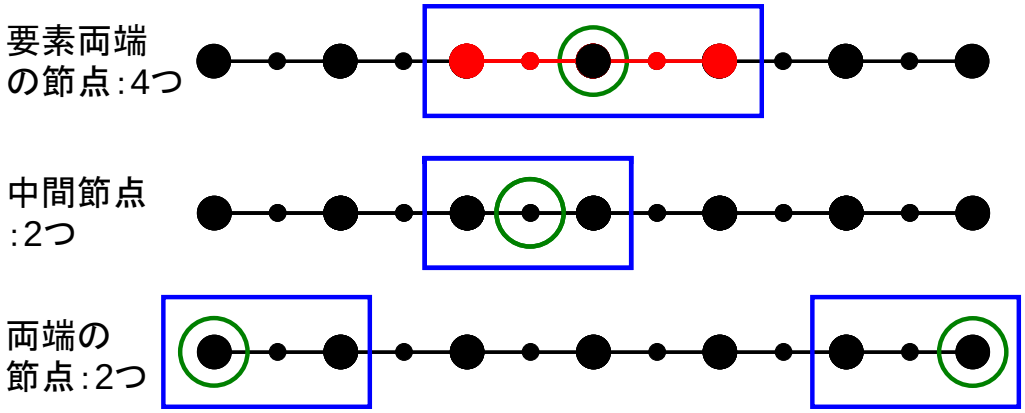
$$\{F\}=\sum_{i=1}^{4}\left\{f^{(i)}\right\}=$$



非対角成分数：節点によって異なる
同じ要素に属する節点と関係を持つ



非対角成分数：節点によって異なる
同じ要素に属する節点と関係を持つ



- 「両端」節点数　　2
- 中間節点数　　NE
- 要素両端節点数　　NE+1-2= NE-1
  - （「両端」を除く）
- 節点数：N= 2+NE+NE-1= 2*NE + 1
- 非零非対角成分数：NPLU= 2*2 + 2*NE + 4*(NE-1)

プログラム：1d2.c（3/7）

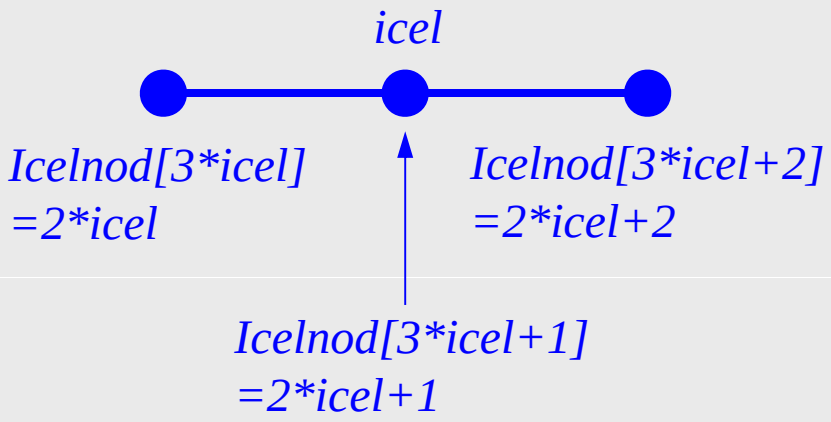
配列宣言（続き），初期化

```
 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;
}
Kmat[0][0]= +14.0/6.;
Kmat[0][1]= -16.0/6.;
Kmat[0][2]=  +2.0/6.;
Kmat[1][0]= -16.0/6.;
Kmat[1][1]= +32.0/6.;
Kmat[1][2]= -16.0/6.;
Kmat[2][0]=  +2.0/6.;
Kmat[2][1]= -16.0/6.;
Kmat[2][2]= +14.0/6.;
```

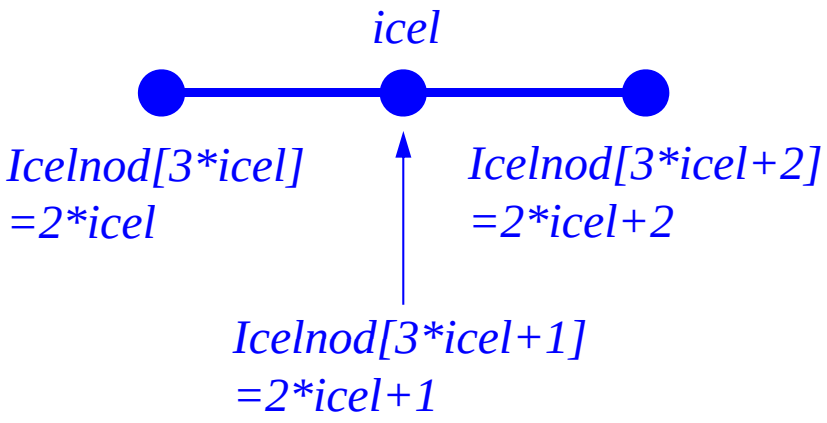
X：　　各節点の座標

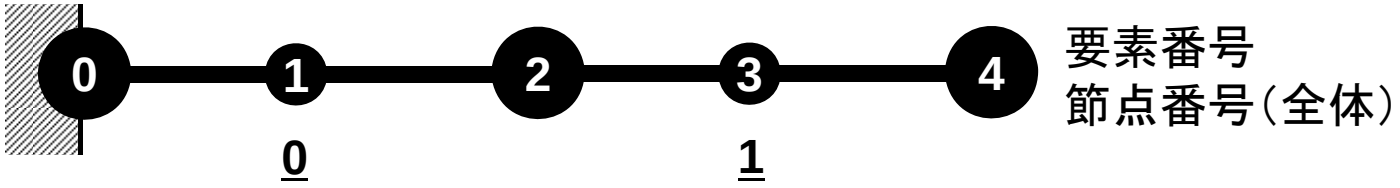
プログラム：1d2.c（3/7）
配列宣言（続き），初期化

```
 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;
}
Kmat[0][0]= +14.0/6.;
Kmat[0][1]= -16.0/6.;
Kmat[0][2]=  +2.0/6.;
Kmat[1][0]= -16.0/6.;
Kmat[1][1]= +32.0/6.;
Kmat[1][2]= -16.0/6.;
Kmat[2][0]=  +2.0/6.;
Kmat[2][1]= -16.0/6.;
Kmat[2][2]= +14.0/6.;
```



要素番号と節点番号（内部）





プログラム：1d2.c（3/7）

配列宣言（続き），初期化

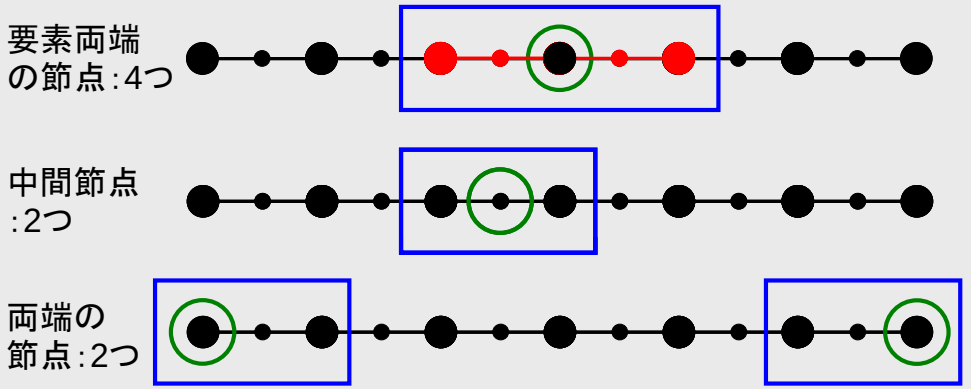
```
 W = (double **)malloc(sizeof(double *)*4);
if(W == NULL) {
        fprintf(stderr, "Error: %s¥n", strerror(errno));
        return -1;
}
for(i=0; i<4; i++) {
        W[i] = (double *)malloc(sizeof(double)*N);
        if(W[i] == NULL) {
                fprintf(stderr, "Error: %s¥n", strerror(errno));
                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;
}
Kmat[0][0]= +14.0/6.;
Kmat[0][1]= -16.0/6.;
Kmat[0][2]=  +2.0/6.;
Kmat[1][0]= -16.0/6.;
Kmat[1][1]= +32.0/6.;
Kmat[1][2]= -16.0/6.;
Kmat[2][0]=  +2.0/6.;
Kmat[2][1]= -16.0/6.;
Kmat[2][2]= +14.0/6.;
```

$$[k]^{(e)} = \frac{EA}{L}\frac{1}{6}\begin{bmatrix} +14 & -16 & +2 \\ -16 & +32 & -16 \\ +2 & -16 & +14 \end{bmatrix}$$

プログラム：1d2.c（4/7）

全体マトリクス：非零非対角成分に対応する列番号

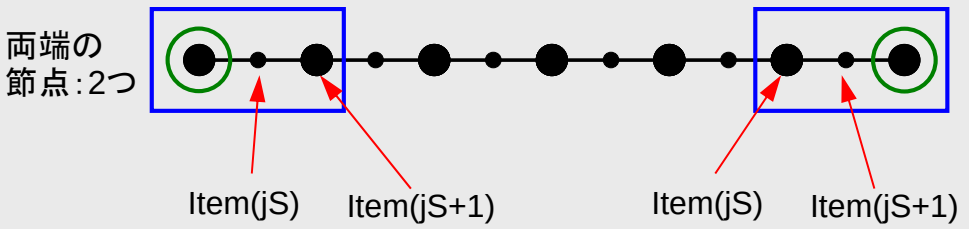
```
/*
// +--------------+
// | CONNECTIVITY |
// +--------------+
*/
      Index[0]= 0;
      for(i=1;i<N;i++) {
        if (i%2==1) {Index[i]=4;
        }else       {Index[i]=2;}}
      Index[1]= 2;
      Index[N]= 2;
      for(i=0;i<N;i++){
            Index[i+1]= Index[i+1] + Index[i];}

      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;}}}
```



プログラム：1d2.c（4/7）

全体マトリクス：非零非対角成分に対応する列番号

```
/*
// +--------------+
// | 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;}}}
```

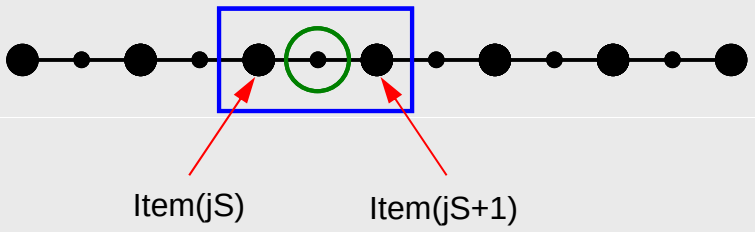


プログラム：1d2.c（4/7）

全体マトリクス：非零非対角成分に対応する列番号

```
/*
// +--------------+
// | 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;}}}
```

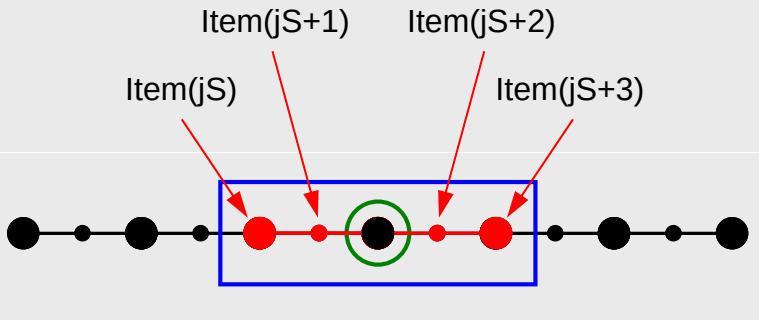
中間節点
:2つ



プログラム：1d2.c（4/7）

全体マトリクス：非零非対角成分に対応する列番号

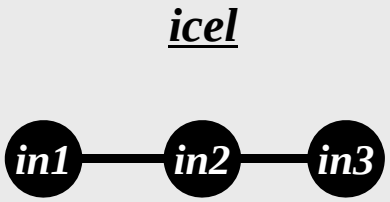
```
/*
// +--------------+
// | CONNECTIVITY |
// +--------------+
*/
        Index[0]= 0;
        for(i=1;i<N;i++) {
          if (i%2==1) {Index[i]=4;
          }else       {Index[i]=2;}}
        Index[1]= 2;
        Index[N]= 2;
        for(i=0;i<N;i++){
                Index[i+1]= Index[i+1] + Index[i];}

        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;}}}
```



プログラム：1d2.c（5/7）

全体マトリクス生成：要素マトリクス⇒全体マトリクス

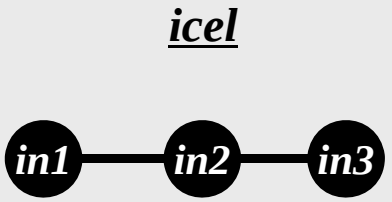
```
/*
// +-----------------+
// | MATRIX assemble |
// +-----------------+
*/
 for(icel=0;icel<NE;icel++){
          in1= Icelnod[3*icel];
          in2= Icelnod[3*icel+1];
          in3= Icelnod[3*icel+2];
          X1 = X[in1];
          X2 = X[in2];
          X3 = X[in3];
          DL = fabs(X3-X1);

          Ck= Area*Young/DL;
          Emat[0][0]= Ck*Kmat[0][0];
          Emat[0][1]= Ck*Kmat[0][1];
          Emat[0][2]= Ck*Kmat[0][2];
          Emat[1][0]= Ck*Kmat[1][0];
          Emat[1][1]= Ck*Kmat[1][1];
          Emat[1][2]= Ck*Kmat[1][2];
          Emat[2][0]= Ck*Kmat[2][0];
          Emat[2][1]= Ck*Kmat[2][1];
          Emat[2][2]= Ck*Kmat[2][2];

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



プログラム：1d2.c（5/7）

全体マトリクス生成：要素マトリクス⇒全体マトリクス

```
/*
// +-----------------+
// | 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);

          Ck= Area*Young/DL;
          Emat[0][0]= Ck*Kmat[0][0];
          Emat[0][1]= Ck*Kmat[0][1];
          Emat[0][2]= Ck*Kmat[0][2];
          Emat[1][0]= Ck*Kmat[1][0];
          Emat[1][1]= Ck*Kmat[1][1];
          Emat[1][2]= Ck*Kmat[1][2];
          Emat[2][0]= Ck*Kmat[2][0];
          Emat[2][1]= Ck*Kmat[2][1];
          Emat[2][2]= Ck*Kmat[2][2];

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

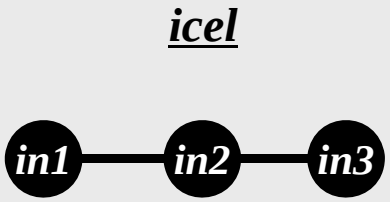


$$[Emat]=\frac{EA}{L}\frac{1}{6}\begin{bmatrix}+14 & -16 & +2\\ -16 & +32 & -16\\ +2 & -16 & +14\end{bmatrix}=\frac{EA}{L}[Kmat]$$

プログラム：1d2.c（5/7）

全体マトリクス生成：要素マトリクス⇒全体マトリクス

```
/*
// +-----------------+
// | 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);

          Ck= Area*Young/DL;
          Emat[0][0]= Ck*Kmat[0][0];
          Emat[0][1]= Ck*Kmat[0][1];
          Emat[0][2]= Ck*Kmat[0][2];
          Emat[1][0]= Ck*Kmat[1][0];
          Emat[1][1]= Ck*Kmat[1][1];
          Emat[1][2]= Ck*Kmat[1][2];
          Emat[2][0]= Ck*Kmat[2][0];
          Emat[2][1]= Ck*Kmat[2][1];
          Emat[2][2]= Ck*Kmat[2][2];

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



$$[Emat] = \frac{EA}{6L}\begin{bmatrix} +14 & -16 & +2 \\ -16 & +32 & -16 \\ +2 & -16 & +14 \end{bmatrix}$$

プログラム：1d2.c（6/7）

全体マトリクス生成：要素マトリクス⇒全体マトリクス

```
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];
 }
```

$$[Emat] = \frac{EA}{6L}\begin{bmatrix} +14 & -16 & +2 \\ -16 & +32 & -16 \\ +2 & -16 & +14 \end{bmatrix}$$

1行目：*in1*の非対角成分 $\begin{bmatrix} - & k12 & k13 \\ k21 & - & k23 \\ k31 & k32 & - \end{bmatrix}$

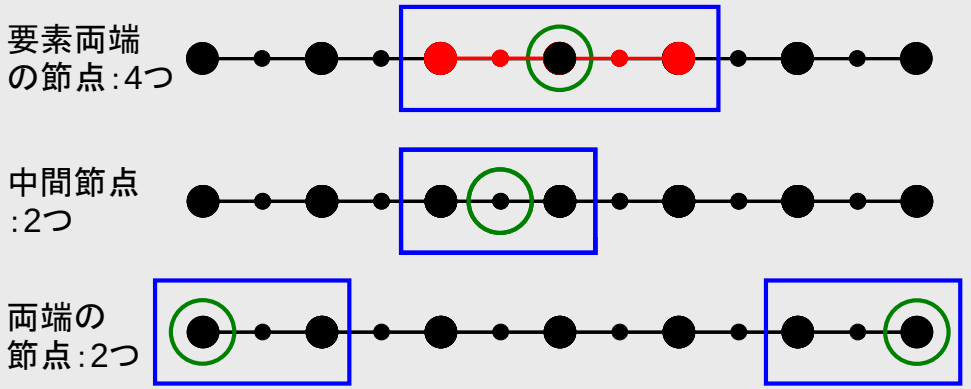
2行目：*in2*の非対角成分

3行目：*in3*の非対角成分

補足

全体マトリクス:非零非対角成分に対応する列番号

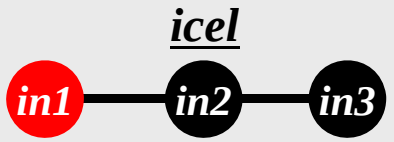
```
/*
// +--------------+
// | CONNECTIVITY |
// +--------------+
*/
      Index[0]= 0;
      for(i=1;i<N;i++) {
        if (i%2==1) {Index[i]=4;
        }else       {Index[i]=2;}}
      Index[1]= 2;
      Index[N]= 2;
      for(i=0;i<N;i++){
            Index[i+1]= Index[i+1] + Index[i];}

      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;}}}
```



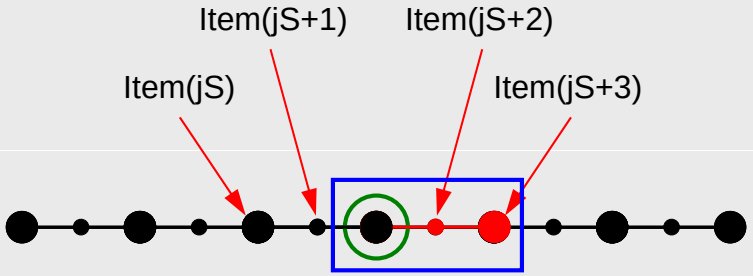
補足

全体マトリクス：非零非対角成分に対応する列番号

```
/*
// +--------------+
// | 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;}}}
```




```
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;}
```



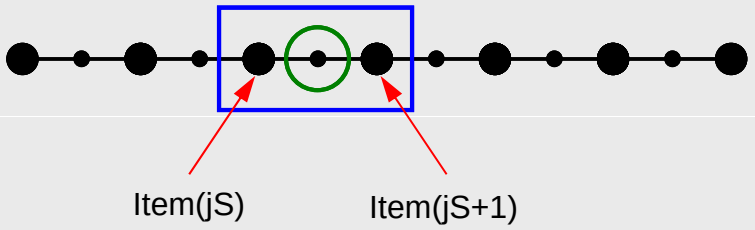
補足

全体マトリクス：非零非対角成分に対応する列番号

```
/*
// +--------------+
// | 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;}}}
```


```
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;}
```

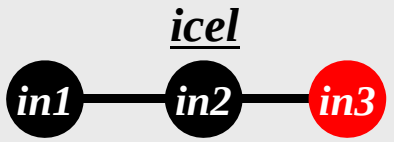
中間節点
:2つ



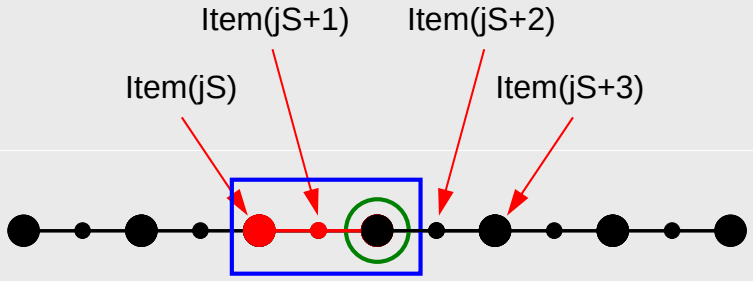
補足

全体マトリクス：非零非対角成分に対応する列番号

```
/*
// +--------------+
// | 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;}}}
```



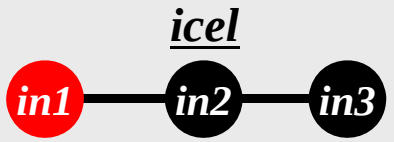

```
if (icel==0){k12=Index[in1];
             k13=Index[in1]+1;
      }else {k12=Index[in1]+2;
             k13=Index[in1]+3;}

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

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



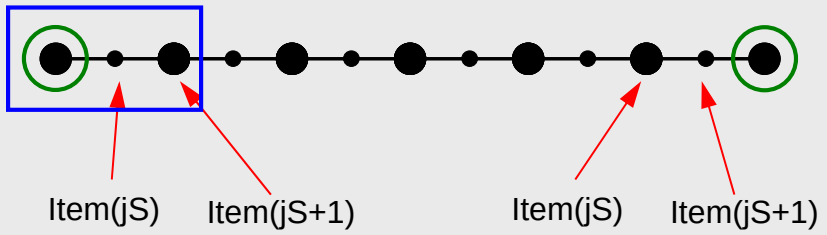
補足

全体マトリクス：非零非対角成分に対応する列番号

```
/*
// +--------------+
// | 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;}}}
```



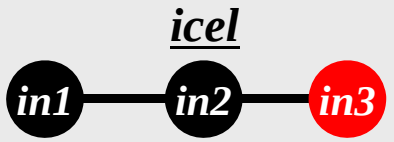

```
if (icel==0){k12=Index[in1];
             k13=Index[in1]+1;
      }else {k12=Index[in1]+2;
             k13=Index[in1]+3;}

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

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



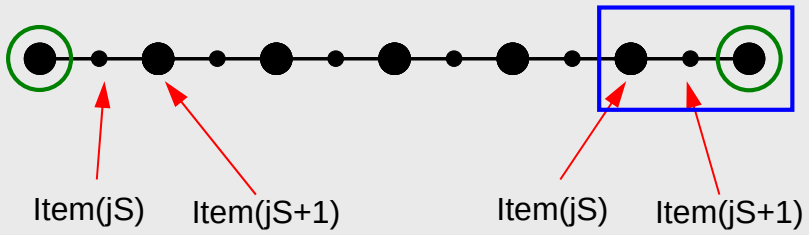
補足

全体マトリクス：非零非対角成分に対応する列番号

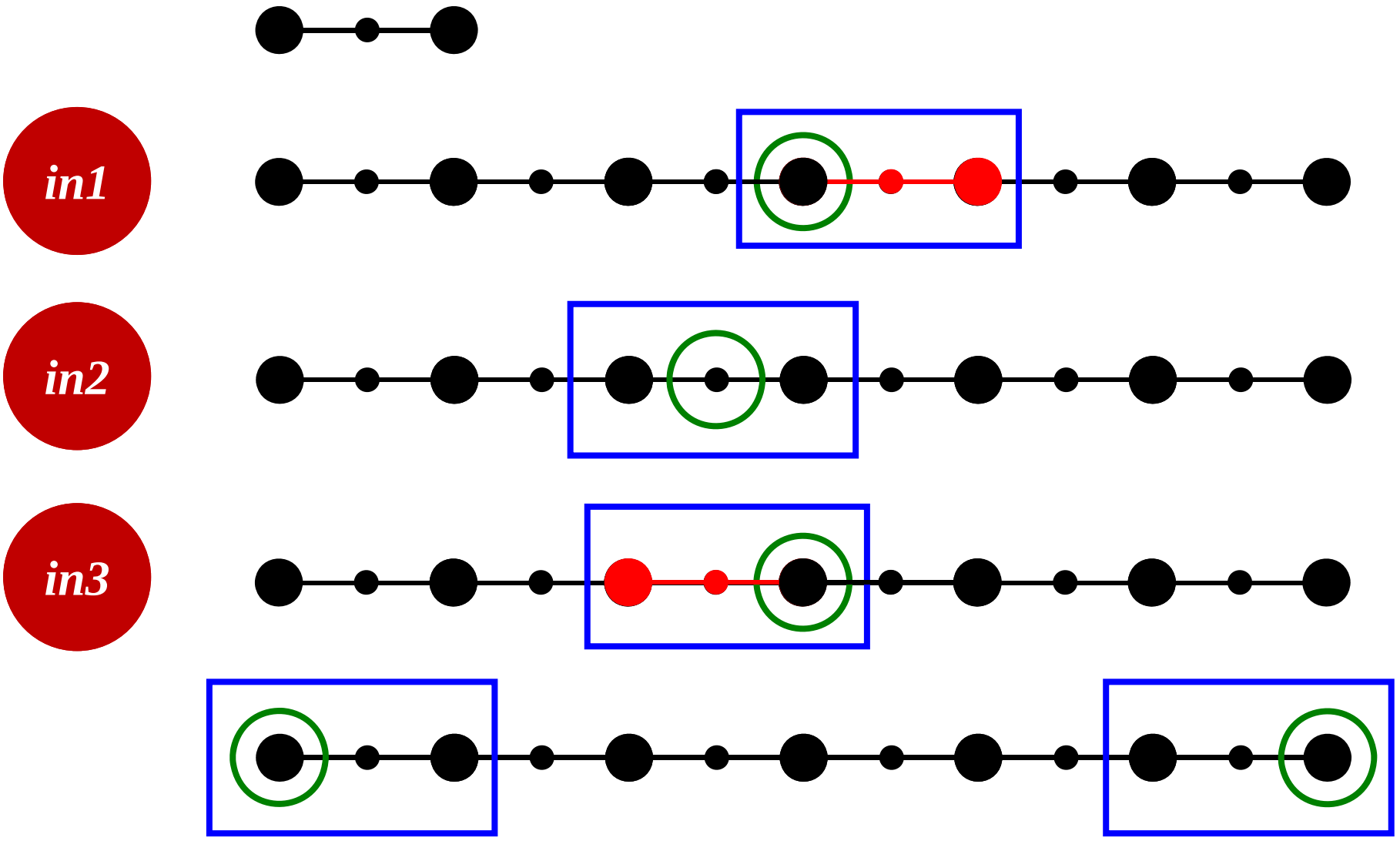
```
/*
// +--------------+
// | CONNECTIVITY |
// +--------------+
*/
        Index[0]= 0;
        for(i=1;i<N;i++) {
          if (i%2==1) {Index[i]=4;
          }else       {Index[i]=2;}}
        Index[1]= 2;
        Index[N]= 2;
        for(i=0;i<N;i++){
                Index[i+1]= Index[i+1] + Index[i];}

        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;}}}
```




```
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;}
```

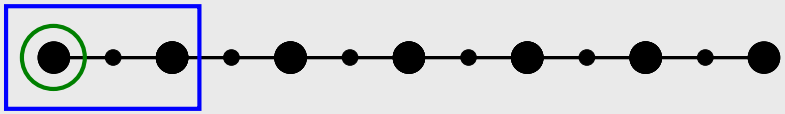


非対角成分数：節点によって異なる
同じ要素に属する節点と関係を持つ



プログラム：1d2.c（7/7）
境界条件

```
/*
// +--------------------+
// | BOUNDARY conditions |
// +--------------------+
*/

/* X=Xmin */
        i=0;
        jS= Index[i];
        AMat[jS]  = 0.0;
        AMat[jS+1]= 0.0;
        Diag[i ]= 1.0;
        Rhs [i ]= 0.0;

          for(k=0;k<NPLU;k++){
            if(Item[k]==0){AMat[k]=0.0;
            }}

/* X=Xmax */
        i=N-1;
        Rhs[i]= F;
```



あとは出てきた全体方程式（連立一次方程式）を解けばよい

もし断面積が x の関数だったら？

$$\int_V E\left(\frac{d[N]^T}{dx}\frac{d[N]}{dx}\right)dV = \int_0^L \begin{bmatrix} dN_i/dx \\ dN_j/dx \\ dN_k/dx \end{bmatrix} E\left[\frac{dN_i}{dx}, \frac{dN_j}{dx}, \frac{dN_k}{dx}\right] A(x)\,dx$$

$$= E\int_0^L \begin{bmatrix} A(x)\frac{dN_i}{dx}\frac{dN_i}{dx} & A(x)\frac{dN_i}{dx}\frac{dN_j}{dx} & A(x)\frac{dN_i}{dx}\frac{dN_k}{dx} \\ A(x)\frac{dN_j}{dx}\frac{dN_i}{dx} & A(x)\frac{dN_j}{dx}\frac{dN_j}{dx} & A(x)\frac{dN_j}{dx}\frac{dN_k}{dx} \\ A(x)\frac{dN_k}{dx}\frac{dN_i}{dx} & A(x)\frac{dN_k}{dx}\frac{dN_j}{dx} & A(x)\frac{dN_k}{dx}\frac{dN_k}{dx} \end{bmatrix} dx$$

$$N_i = \left(1-\frac{2x}{L}\right)\left(1-\frac{x}{L}\right) \qquad \frac{dN_i}{dx} = \left(\frac{4x}{L^2}-\frac{3}{L}\right)$$

$$N_j = \left(\frac{4x}{L}\right)\left(1-\frac{x}{L}\right) \qquad \frac{dN_j}{dx} = \left(\frac{4}{L}-\frac{8x}{L^2}\right)$$

$$N_k = \left(-\frac{x}{L}\right)\left(1-\frac{2x}{L}\right) \qquad \frac{dN_k}{dx} = \left(\frac{4x}{L^2}-\frac{1}{L}\right)$$

積分がかなり面倒になる
（できないことはないが）
⇒数値的にできると便利

- ガラーキン法による一次元弾性問題の解法

- 連立一次方程式の解法
  - 共役勾配法
  - 前処理手法
- 疎行列格納法

- プログラムの内容

- 高次要素
- 数値積分法，アイソパラメトリック要素
- レポート課題1

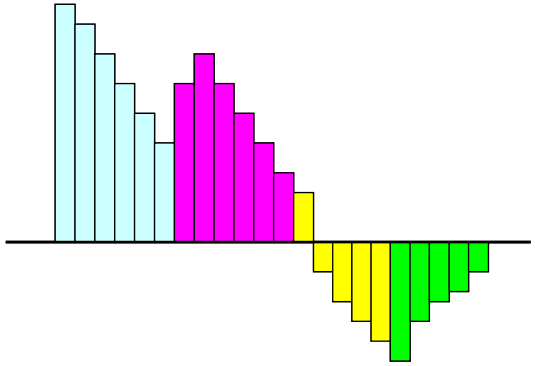
- ガウス積分公式
- アイソパラメトリック要素
- 実装例
- レポート課題1

数値的に積分を実施する方法

- 台形公式
- シンプソンの公式
- ガウスの積分公式（ガウス＝ルジャンドル（Gauss-Legendre）とも呼ばれる，精度良い）

- 不定積分を解析的に求めるのではなく，有限な数のサンプル点における値を利用する

$$\int_{X_1}^{X_2} f(x)dx \Rightarrow \sum_{k=1}^{m} [w_k \cdot f(x_k)]$$



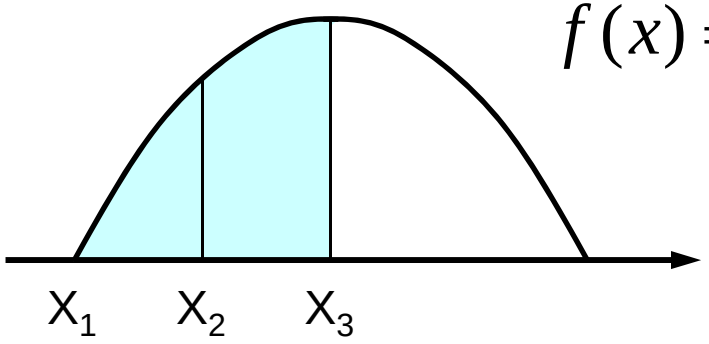
ガウス積分公式：一次元の例

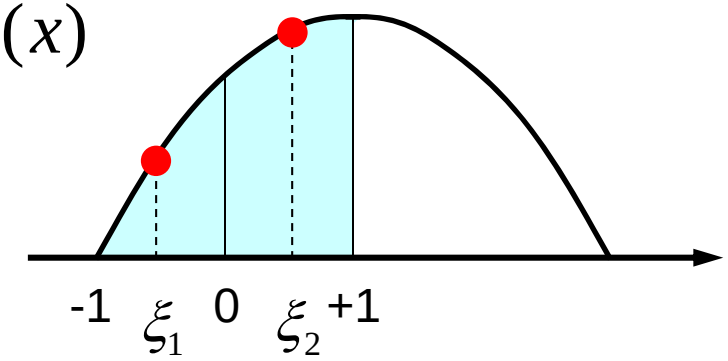
シンプソンの公式より精度良い

Simpson's

Gauss

$$f(x) = \sin(x)$$





$$X_1 = 0, \quad X_2 = \frac{\pi}{4}, \quad X_3 = \frac{\pi}{2}$$

$$h = X_2 - X_1 = X_3 - X_1 = \frac{\pi}{4}$$

$$S = \frac{h}{3}\left[f(X_1) + 4f(X_2) + f(X_3)\right] = 1.0023$$

$$\xi_1, \xi_2 = \pm 0.5773502692$$

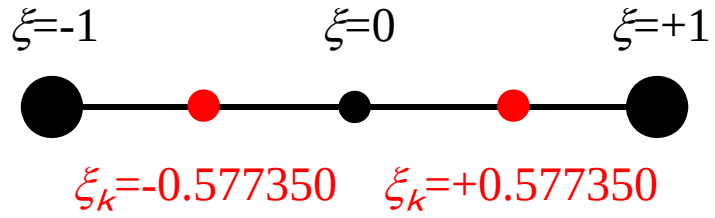
$$S = \int_0^{\pi/2} f(x)dx = \int_{-1}^{+1} f(\xi)\, h\, d\xi$$

$$\cong h\sum_{k=1}^{2} W_k \cdot f(\xi_k) = 0.99847$$

ガウスの積分公式

- 無次元化された自然座標系 [-1,+1] を対象とする。
- m個の積分点を使用すると（2m-1）次の関数までは近似可能（従って1次，2次の内挿関数（形状関数）を使用するときは，m=2で十分）

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



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

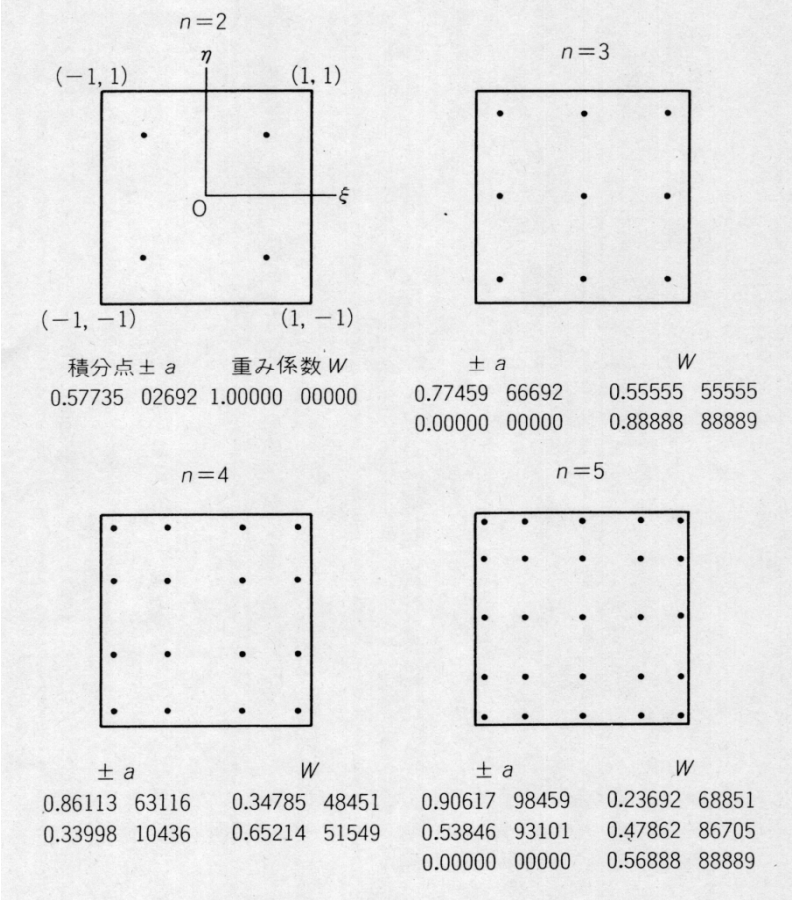
$m=2 \quad \xi_k = \pm 0.577350,\ w_k = 1.00$

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

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

ガウスの積分公式

二次元，三次元にも容易に拡張可能



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

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

m,n: ξ,η方向の積分点数

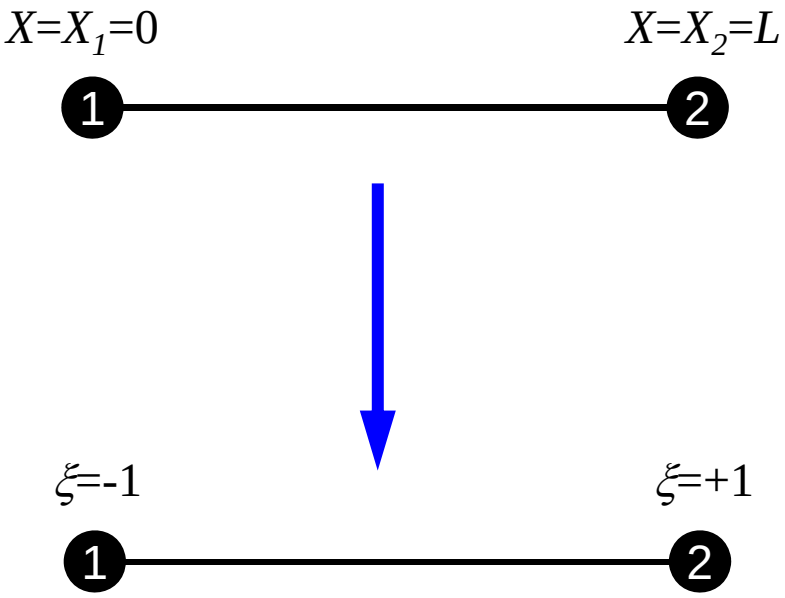
(ξ_i,η_j): 積分点の座標値

W_i, W_j: 積分点での重み係数

- ガウス積分公式
- アイソパラメトリック要素
- 実装例
- レポート課題1

ガウスの積分公式

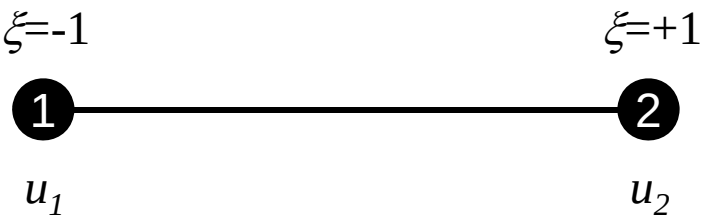
- 使用するためには，$[0,L]$（または$[X_1,X_2]$）から[-1,+1]への座標変換が必要
- 内挿関数（形状関数）等も自然座標系上で扱う必要がある



以後，X,u,N などの添字を局所節点番号（1,2,3…）とする

1次(線形)要素

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



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

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

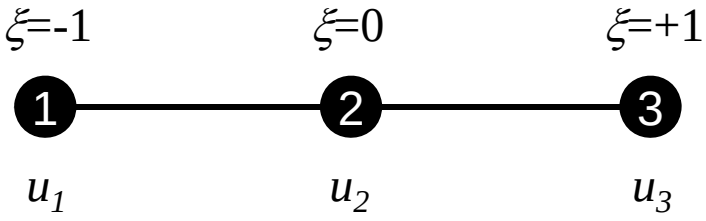
$$N_1(\xi) = \frac{1}{2}(1-\xi), \quad N_2(\xi) = \frac{1}{2}(1+\xi)$$

2次要素

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

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

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



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

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

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

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

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

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

アイソパラメトリック要素

- 各要素を，自然座標系（local coordinate） [-1,+1] に変換する。
- 各要素の全体座標系（global coordinate）（x）における座標成分を，自然座標系における形状関数[N]を使用して変換する場合，このような要素をアイソパラメトリック要素（isoparametric element）という（従属変数の内挿と同じ[N]を使用）。

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

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

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

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

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

- ガウス積分公式
- アイソパラメトリック要素
- 実装例
- レポート課題1

要素単位での積分:$[k]$

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

自然座標系における偏微分（1/2）

- 偏微分の公式より以下のようになる：

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

$\left[\frac{\partial N_i}{\partial \xi}\right]$ は定義より簡単に求められるが

$\left[\frac{\partial N_i}{\partial x}\right]$ を実際の計算で使用する

$\left[\frac{\partial x}{\partial \xi}\right]$ ヤコビアン（Jacobian）（$=J$）

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

自然座標系における偏微分（2/2）

- 以下のように求められる

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

- 自然座標系における積分

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

要素単位での積分:[k](1/2)

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

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

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

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

要素単位での積分:[k](2/2)

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

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

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

ファイルコピー

b1.c：a1.cを自然座標系にしただけ一次元線形要素を使用

一次元弾性解析コード

```
>$ cd <$fem1>
>$ cp /home03/skengon/Documents/class/fem1/1d3.tar .
>$ tar xvf 1d3.tar
>$ cd 1darea
```

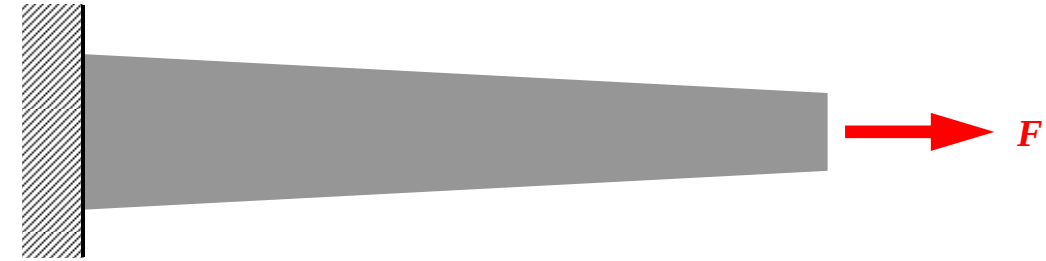
実行

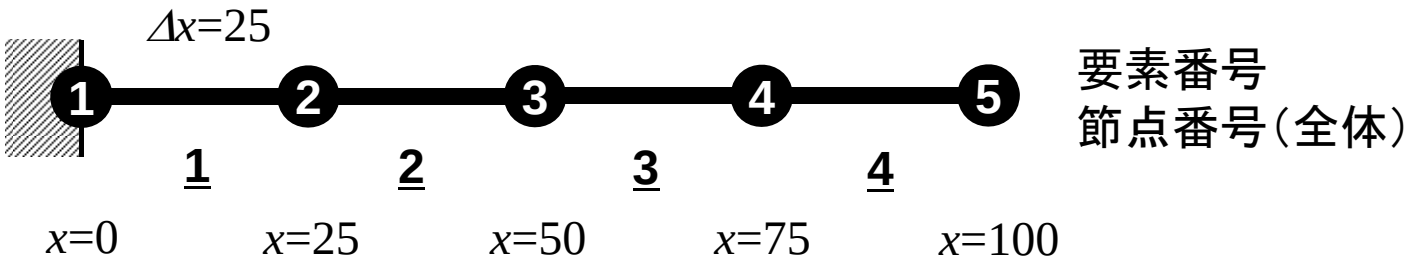
```
>$ cd <$fem1>/1darea
>$ cc -O b1.c      (or g95 -O b1.f)
>$ ./a.out
```

制御ファイル　input.dat

```
4                                   NE（要素数）
25.0  5.e4  -0.105 12  5.e6         Δx（要素長さL）, F, A1, A2, E
100                                 反復回数（CG法後述）
1.e-8                               CG法の反復打切誤差
```

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





プログラム:b1.c(1/6)
諸変数

```
/*
//   1D Solid Mechanics for Truss Elements solved by
//   CG (Conjugate Gradient) Method
//
//   d/dx(EdU/dx) + F = 0
//   U=0@x=0
*/

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

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

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

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

        int i, j, in1, in2, k, icel, k1, k2, jS, ip;
        int iter;
        FILE *fp;
        double BNorm2, Rho, Rho1=0.0, C1, Alpha, DNorm2;
        int ierr = 1;
```

変数表(1/2)

変数名	種別	サイズ	I/O	内　容
NE	I		I	要素数
N	I		O	節点数
NPLU	I		O	非零非対角成分数
IterMax	I		I	最大反復回数
errno	I		O	エラー戻り値
R, Z, Q, P, DD	I		O	CG法ベクトル名
dX	R		I	要素長さ
Resid	R		O	CG法残差
Eps	R		I	CG法反復打ち切り残差
Area, A1, A2	R		I	要素断面積（Area= A1*x+A2）
F	R		I	軸力F（境界条件）
Young	R		I	ヤング率
Jacobi	R		O	ガウス積分点におけるヤコビアン
X0	R		O	要素中心における全体座標
XX	R		O	ガウス積分点における全体座標
X1,X2,U1,U2	R		O	局所節点1,2の座標，変位

変数表(2/2)

変数名	種別	サイズ	I/O	内　容
DL, Ck	R		O	要素マトリクス計算用定数
Disp	R		O	節点変位
Strain, Stress	R		O	要素ひずみ，要素応力
X	R	N	O	節点座標
U	R	N	O	節点変位
Rhs	R	N	O	右辺ベクトル
Diag	R	N	O	全体マトリクス：対角成分
W	R	[4][N]	O	CG法のwork配列
Amat	R	NPLU	O	全体マトリクス：非零非対角成分
Index	I	N+1	O	全体マトリクス：各行非零非対角成分数
Item	I	NPLU	O	全体マトリクス：列番号
Icelnod	I	2*NE	O	各要素節点番号
POI, WEI	R	[2]	O	ガウス積分点座標，重み係数
dNdQ	R	[2]	O	形状関数微分係数（$dN/d\xi$）
Emat	R	[2][2]	O	要素マトリクス

プログラム:b1.c(2/6)
初期設定，配列宣言:a1.cと同じ

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

          N= NE + 1;

          U    = calloc(N, sizeof(double));
          X    = calloc(N, sizeof(double));
          Diag = calloc(N, sizeof(double));

          AMat = calloc(2*N-2, sizeof(double));

          Rhs = calloc(N, sizeof(double));
          Index= calloc(N+1, sizeof(int));
          Item = calloc(2*N-2, sizeof(int));

          IcelNod= calloc(2*NE, sizeof(int));
```

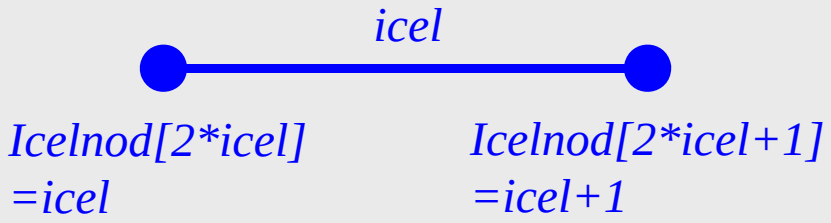
プログラム:b1.c(3/6)
配列宣言(続き), 初期化

```
        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<2*N-2;k++)  AMat[k] = 0.0;
        for(i=0;i<N;i++) X[i]= i*dX;
        for(icel=0;icel<NE;icel++){
                Icelnod[2*icel  ]= icel;
                Icelnod[2*icel+1]= icel+1;
        }

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

X : 各節点の座標

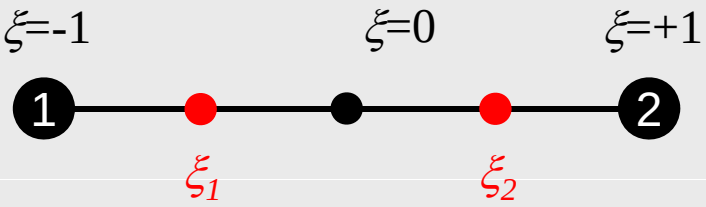
プログラム:b1.c(3/6)
配列宣言(続き)，初期化

```
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<2*N-2;k++)  AMat[k] = 0.0;
for(i=0;i<N;i++) X[i]= i*dX;
for(icel=0;icel<NE;icel++){
        Icelnod[2*icel  ]= icel;
        Icelnod[2*icel+1]= icel+1;
}

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



プログラム：b1.c(3/6)
配列宣言(続き)，初期化

```
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<2*N-2;k++)  AMat[k] = 0.0;
for(i=0;i<N;i++) X[i]= i*dX;
for(icel=0;icel<NE;icel++){
        Icelnod[2*icel  ]= icel;
        Icelnod[2*icel+1]= icel+1;
}

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



プログラム:b1.c(4/6)

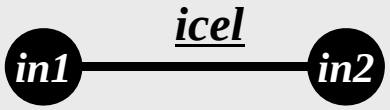
全体マトリクス:非零非対角成分に対応する列番号

```
/*
// +--------------+
// | CONNECTIVITY |
// +--------------+
*/
        for(i=0;i<N+1;i++)  Index[i] = 2;
        Index[0]= 0;
        Index[1]= 1;
        Index[N]= 1;

        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;
                }else if(i == N-1){
                        Item[jS] = i-1;
                }else{
                        Item[jS] = i-1;
                        Item[jS+1] = i+1;
                }
        }
```

プログラム:b1.c(5/6)

全体マトリクス生成:要素マトリクス⇒全体マトリクス

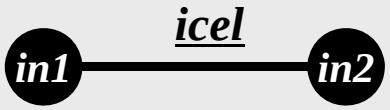
```
/*
// +-----------------+
// | MATRIX assemble |
// +-----------------+
*/
      for(icel=0;icel<NE;icel++){
          in1= Icelnod[2*icel];
          in2= Icelnod[2*icel+1];
          X1 = X[in1];
          X2 = X[in2];
          DL = fabs(X2-X1);
          X0 = 0.5 * (X1+X2);

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

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



プログラム：b1.c（5/6）

全体マトリクス生成：要素マトリクス⇒全体マトリクス

```
/*
// +-----------------+
// | MATRIX assemble |
// +-----------------+
*/
      for(icel=0;icel<NE;icel++){
          in1= Icelnod[2*icel];
          in2= Icelnod[2*icel+1];
          X1 = X[in1];
          X2 = X[in2];
          DL = fabs(X2-X1);
          X0 = 0.5 * (X1+X2);

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

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



各形状関数の微分係数（$dN/d\xi$）

1次(線形)要素

形状関数の微分係数は定数

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

$$u = N_1(\xi)\cdot u_1 + N_2(\xi)\cdot u_2$$

$$\frac{dN_1}{d\xi}=\frac{d}{d\xi}\left[\frac{1}{2}(1-\xi)\right]=-\frac{1}{2}$$

$$\frac{dN_2}{d\xi}=\frac{d}{d\xi}\left[\frac{1}{2}(1+\xi)\right]=+\frac{1}{2}$$

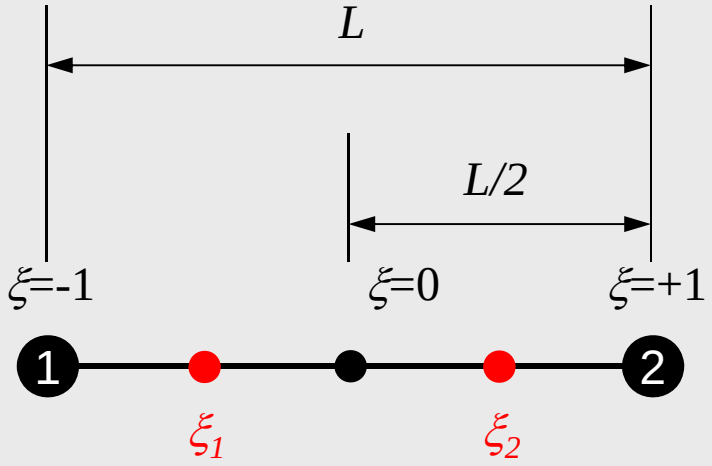
dNdQ[0]
（FORTRANでは**dNdQ(1)**）

dNdQ[1]
（FORTRANでは**dNdQ(2)**）

プログラム：b1.c（5/6）

全体マトリクス生成：要素マトリクス⇒全体マトリクス

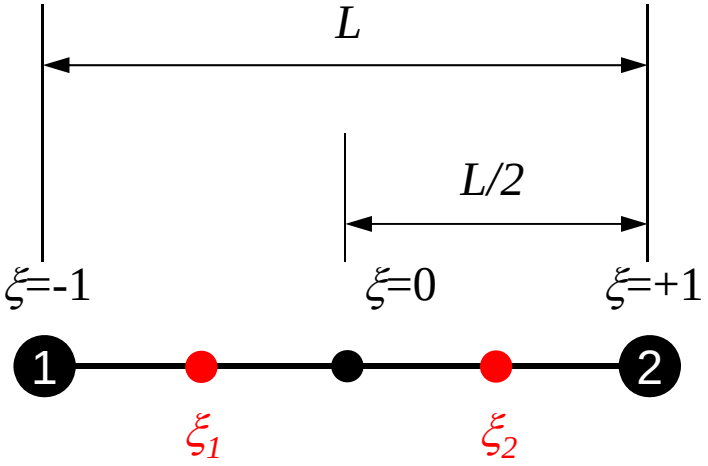
```
/*
// +-----------------+
// | MATRIX assemble |
// +-----------------+
*/
      for(icel=0;icel<NE;icel++){
          in1= Icelnod[2*icel];
          in2= Icelnod[2*icel+1];
          X1 = X[in1];
          X2 = X[in2];
          DL = fabs(X2-X1);
          X0 = 0.5 * (X1+X2);

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

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



XX　：ガウス積分点における座標

ガウス積分点でのX座標：一次要素

- アイソパラメトリック要素の定義より



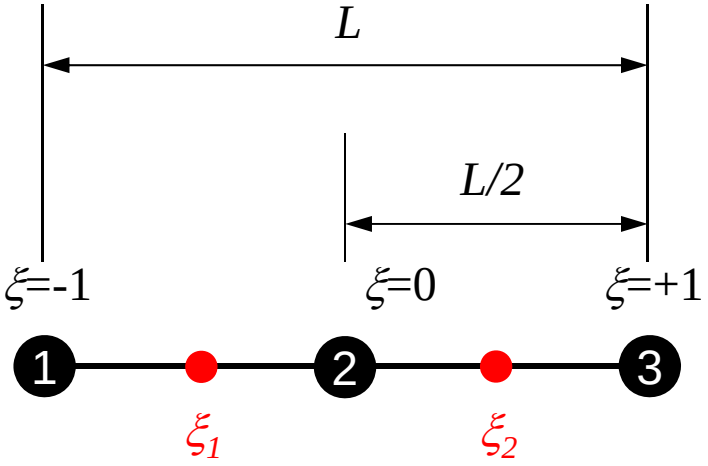
$$X(\xi)=\sum_{k=1}^{2}N_k(\xi)X_k$$
$$=N_1(\xi)X_1+N_2(\xi)X_2$$
$$=\frac{1}{2}(1-\xi)X_1+\frac{1}{2}(1+\xi)X_2$$
$$=\frac{X_1+X_2}{2}+\frac{X_2-X_1}{2}\xi$$

X0　　L/2（プログラム中ではDL/2）

∴ XX= X0 + POI[ip]*0.50*DL;

ガウス積分点でのX座標：二次要素

- アイソパラメトリック要素の定義より



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

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

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

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

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

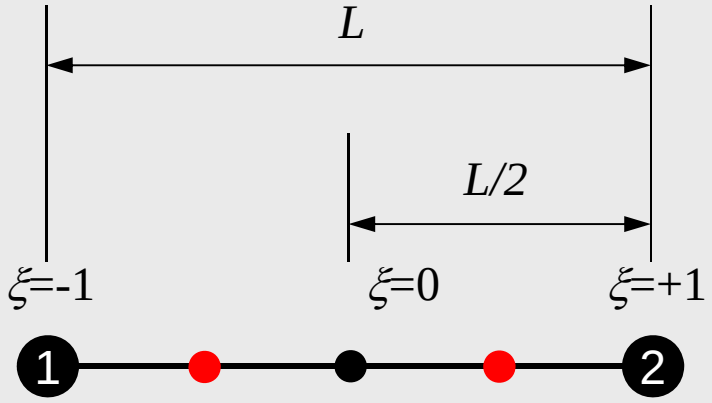
一次要素のときと同じ

∴ XX= XO + POI[ip]*0.50*DL;

プログラム：b1.c（5/6）

全体マトリクス生成：要素マトリクス⇒全体マトリクス

```
/*
// +-----------------+
// | MATRIX assemble |
// +-----------------+
*/
      for(icel=0;icel<NE;icel++){
          in1= Icelnod[2*icel];
          in2= Icelnod[2*icel+1];
          X1 = X[in1];
          X2 = X[in2];
          DL = fabs(X2-X1);
          X0 = 0.5 * (X1+X2);

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

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

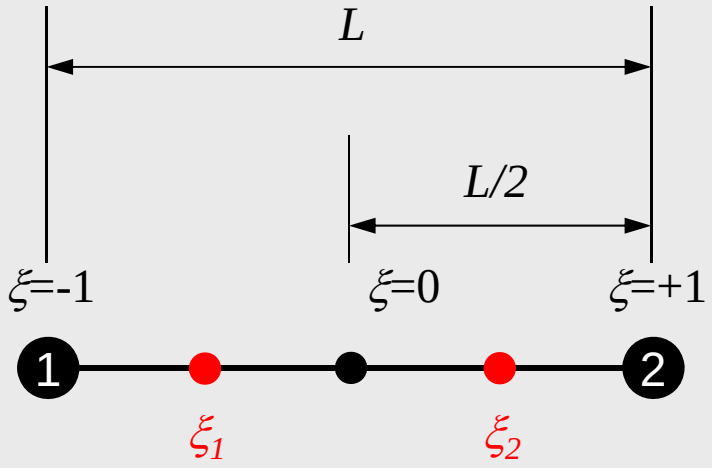


XX　:ガウス積分点における座標
Area: ガウス積分点における断面積

プログラム：b1.c（5/6）

全体マトリクス生成：要素マトリクス⇒全体マトリクス

```
/*
// +-----------------+
// | MATRIX assemble |
// +-----------------+
*/
      for(icel=0;icel<NE;icel++){
           in1= Icelnod[2*icel];
           in2= Icelnod[2*icel+1];
           X1 = X[in1];
           X2 = X[in2];
           DL = fabs(X2-X1);
           X0 = 0.5 * (X1+X2);

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

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



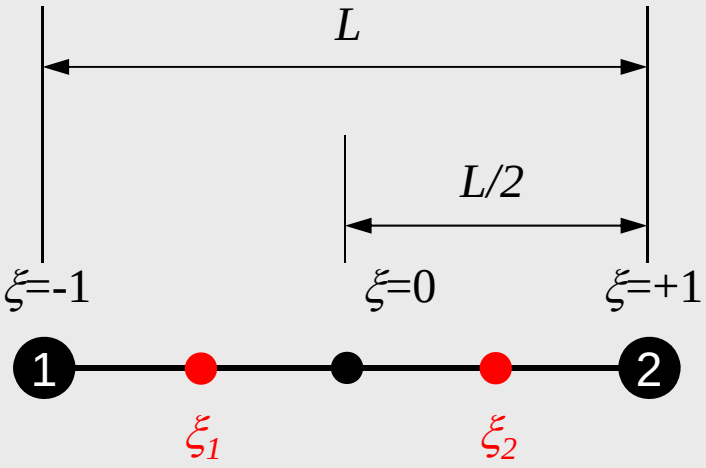
$$|J|_{\xi=\xi_k} = \left|\frac{\partial x}{\partial \xi}\right|_{\xi=\xi_k} = \left|\frac{\partial}{\partial \xi}\left(N_1 x_1 + N_2 x_2\right)\right|_{\xi=\xi_k}$$

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

プログラム：b1.c（5/6）

全体マトリクス生成：要素マトリクス⇒全体マトリクス

```
/*
// +-----------------+
// | MATRIX assemble |
// +-----------------+
*/
      for(icel=0;icel<NE;icel++){
          in1= Icelnod[2*icel];
          in2= Icelnod[2*icel+1];
          X1 = X[in1];
          X2 = X[in2];
          DL = fabs(X2-X1);
          X0 = 0.5 * (X1+X2);

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

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



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

プログラム:b1.c(6/6)
境界条件

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

            if (icel==0){k1=Index[in1];
                  }else {k1=Index[in1]+1;}
            k2=Index[in2];

            AMat[k1]= AMat[k1] + Emat[0][1];
            AMat[k2]= AMat[k2] + Emat[1][0];
        }
/*
// +--------------------+
// | BOUNDARY conditions |
// +--------------------+
*/

/* X=Xmin */
        i=0;
        jS= Index[i];
        AMat[jS]  = 0.0;
        Diag[i ]= 1.0;
        Rhs [i ]= 0.0;

          for(k=0;k<NPLU;k++){
            if(Item[k]==0){AMat[k]=0.0;
            }}

/* X=Xmax */
        i=N-1;
        Rhs[i]= F;
```

- ガウス積分公式
- アイソパラメトリック要素
- 実装例
- レポート課題1

レポート課題1

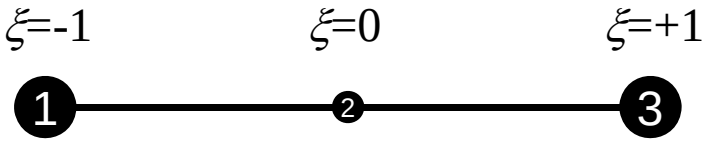
- b1.c，b1.fを高次要素（二次の補間関数使用）の使用により改良せよ（b2.c，b2.f）
- b1とb2についてメッシュ数を変更したケースを実施し，精度について考察せよ
- 下記の成立を確認せよ（断面積一定の場合）

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

- 提出期限
  - 2012年8月24日（金）17:00（9月末卒業の人は要相談）
- 提出物
  - レポート（概要，計算結果，考察）（図表含A4 5枚以内）
  - プログラムソースリスト

レポート課題1：ヒント①

- 形状関数の微分係数



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

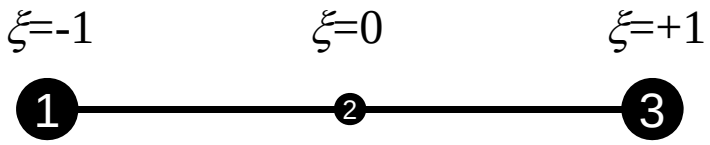
$$\frac{dN_1}{d\xi}=-\frac{1}{2}+\xi$$
$$\frac{dN_2}{d\xi}=-2\xi$$
$$\frac{dN_3}{d\xi}=\frac{1}{2}+\xi$$

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

積分点（ξ_k）における値を代入

レポート課題1：ヒント②

- ヤコビアン



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

- ガウス積分点（ξ_k）における値

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