有限要素法による
一次元定常熱伝導解析プログラム
C言語編

中島 研吾

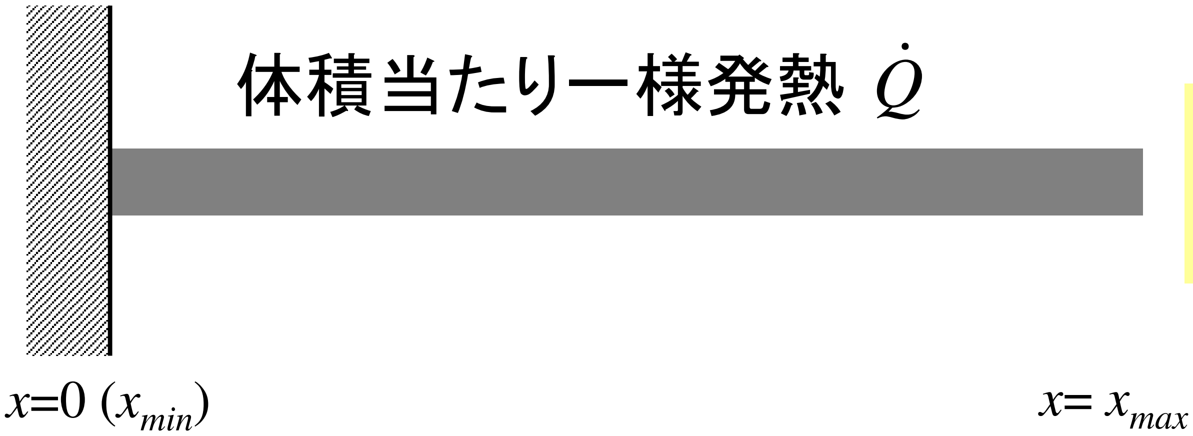
東京大学情報基盤センター

- ガラーキン法による一次元熱伝導問題の解法
- 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

キーワード

- 一次元熱伝導問題
- ガラーキン法
- 線形一次要素
- 前処理付共役勾配法

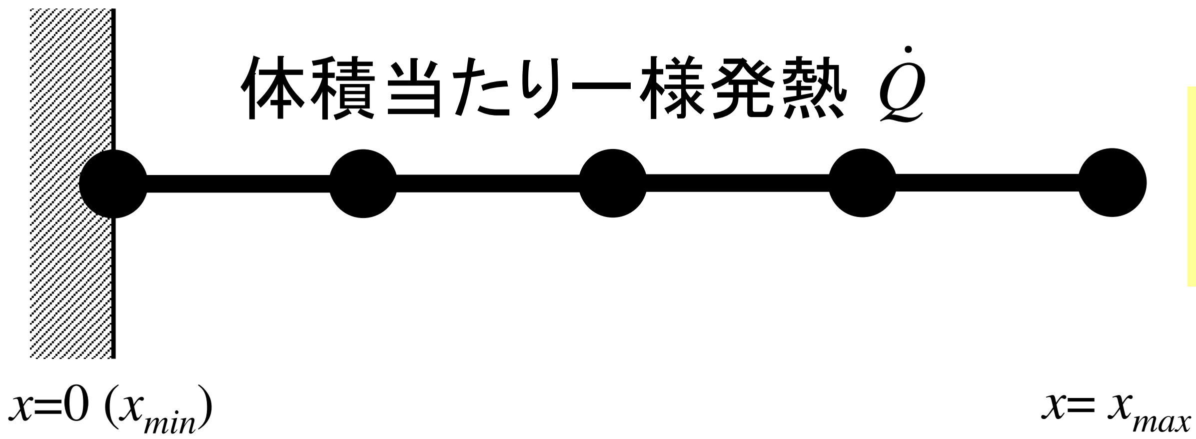
対象とする問題：一次元熱伝導問題



$$\frac{\partial}{\partial x}\left(\lambda \frac{\partial T}{\partial x}\right)+\dot{Q}=0$$

- 一様な：断面積A，熱伝導率λ
- 体積当たり一様発熱（時間当たり）〔QL^{-3}T^{-1}〕$\dot{Q}$
- 境界条件
  - x=0　：T= 0　（固定）
  - x=x_{max} ：$\dfrac{\partial T}{\partial x}=0$（断熱）

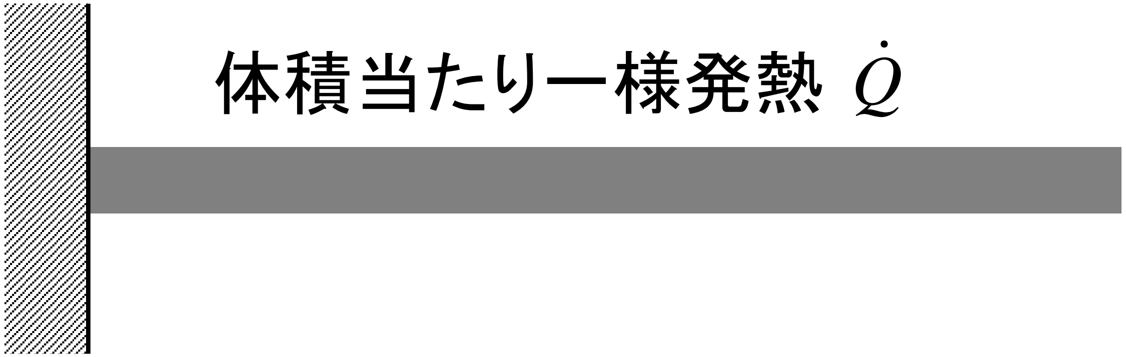
対象とする問題：一次元熱伝導問題



$$\frac{\partial}{\partial x}\left(\lambda \frac{\partial T}{\partial x}\right)+\dot{Q}=0$$

- 一様な：断面積A，熱伝導率λ
- 体積当たり一様発熱（時間当たり）〔QL^{-3}T^{-1}〕$\dot{Q}$
- 境界条件
  - x=0　：T= 0　（固定）
  - x=x_{max}：$\dfrac{\partial T}{\partial x}=0$（断熱）

解析解



$$\frac{\partial}{\partial x}\left(\lambda \frac{\partial T}{\partial x}\right) + \dot{Q} = 0$$

x=0 (x_{min})　　x= x_{max}

$$T = 0 \;@\; x = 0$$

$$\frac{\partial T}{\partial x} = 0 \;@\; x = x_{\max}$$

$$\lambda T'' = -\dot{Q}$$

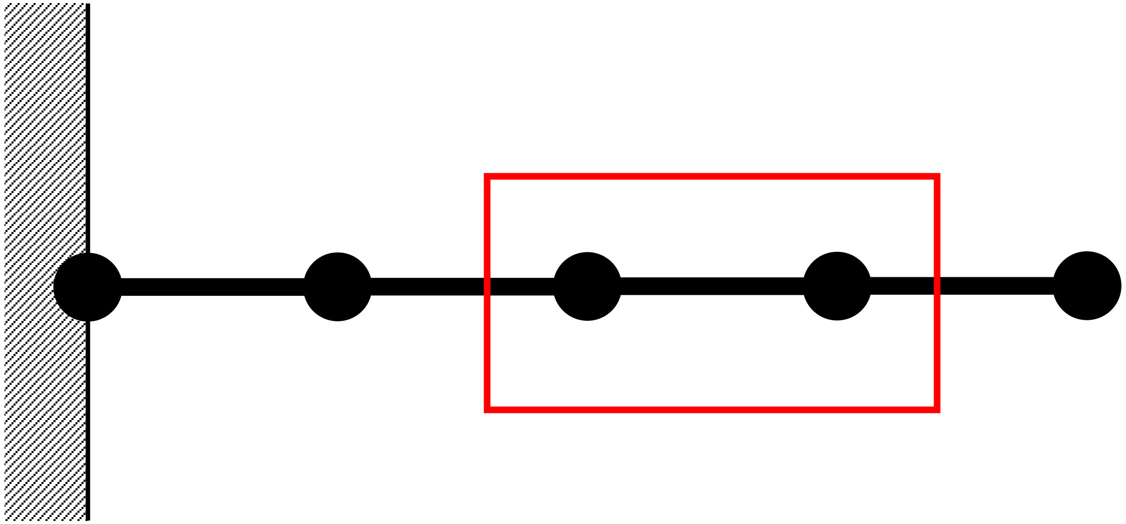
$$\lambda T' = -\dot{Q}x + C_1 \Rightarrow C_1 = \dot{Q}x_{\max}, \quad T' = 0 \;@\; x = x_{\max}$$

$$\lambda T = -\frac{1}{2}\dot{Q}x^2 + C_1 x + C_2 \Rightarrow C_2 = 0, \quad T = 0 \;@\; x = 0$$

$$\therefore \quad T = -\frac{1}{2\lambda}\dot{Q}x^2 + \frac{\dot{Q}x_{\max}}{\lambda}x$$

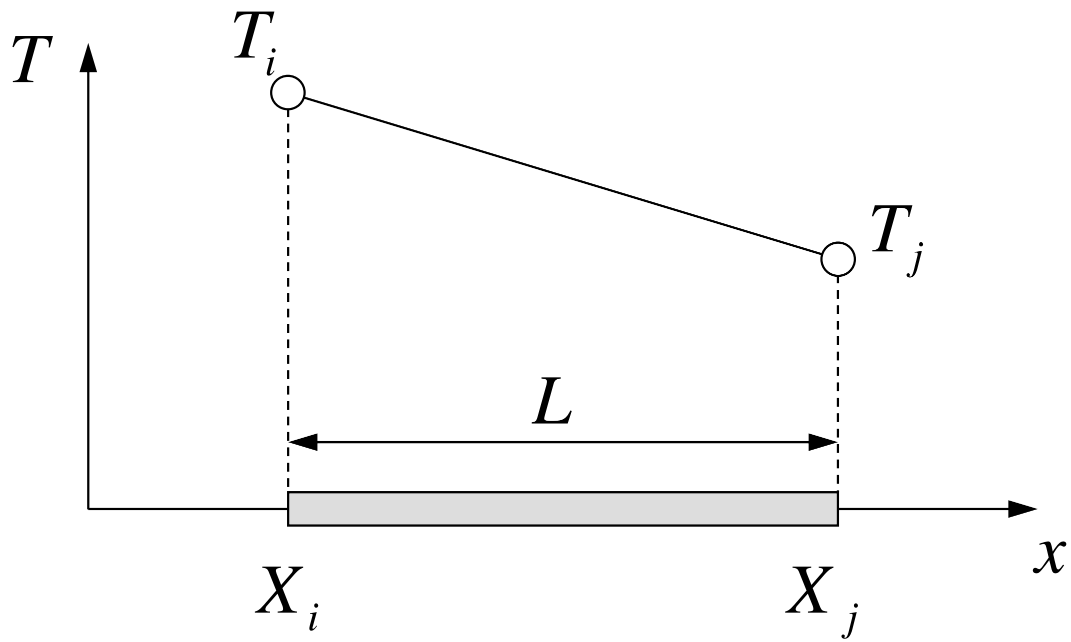
一次元線形要素（1/4）

- 一次元線形要素
  - 長さLの両端に節点（node）を持つ線分
    - 節点：node
    - 要素：element



  - 節点 i,j における温度を T_i, T_j
  - 要素内での温度Tは以下のように表される（座標xの一次関数, Piecewise Linear）：

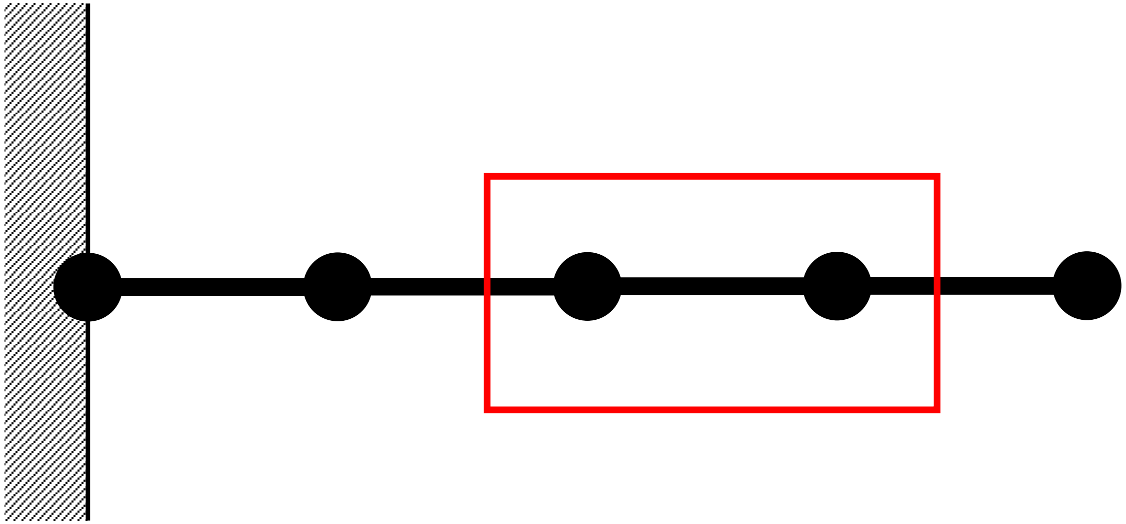
$$T = \alpha_1 + \alpha_2 x$$

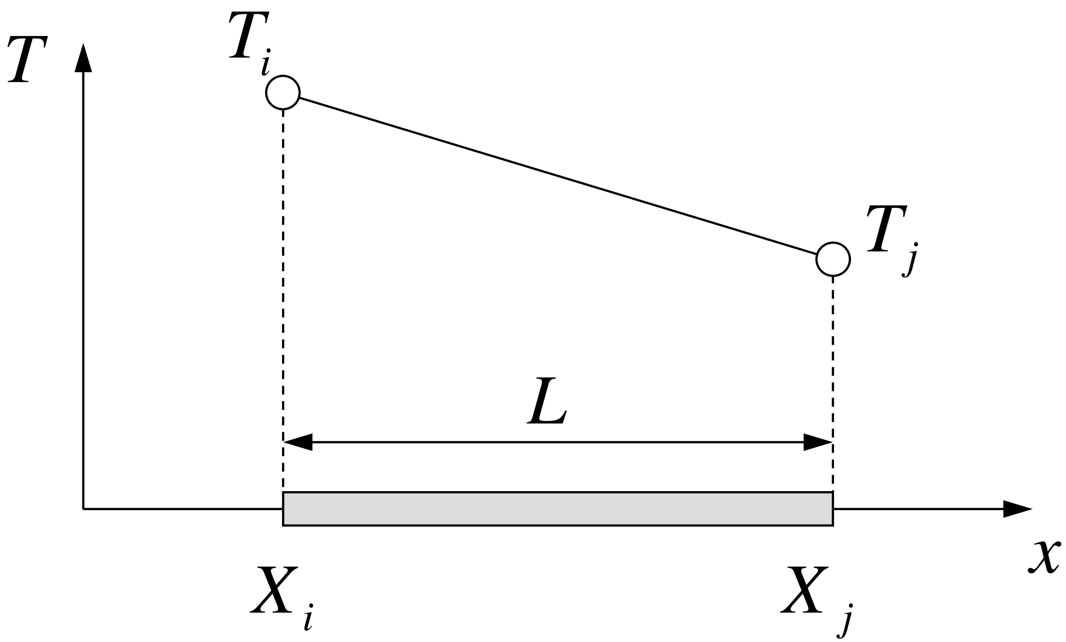


一次元線形要素（1/4）

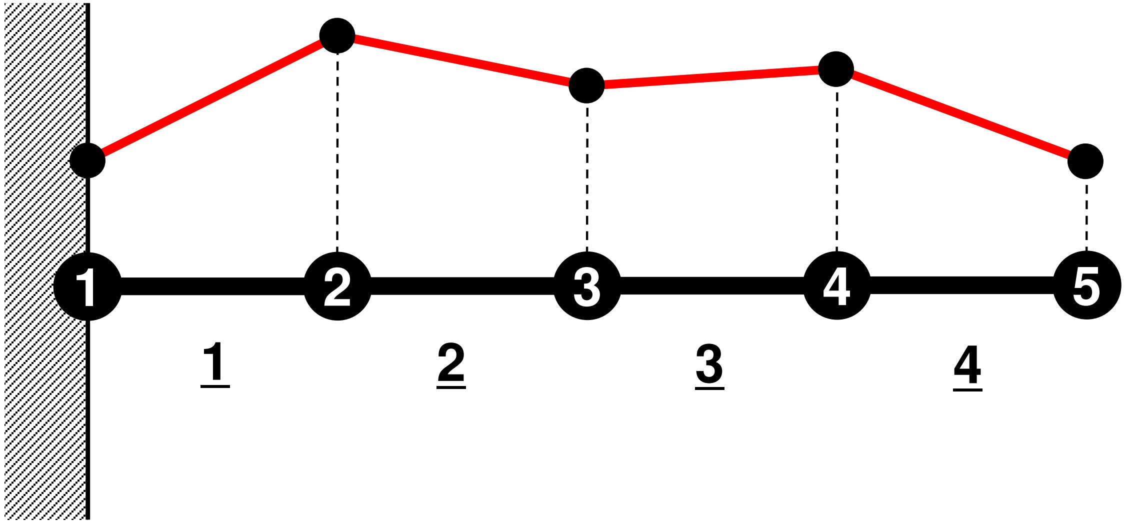
- 一次元線形要素
  - 長さLの両端に節点（node）を持つ線分
    - 節点：node
    - 要素：element
  - 節点 i,j における温度を T_i, T_j
  - 要素内での温度Tは以下のように表される（座標xの一次関数, Piecewise Linear）：

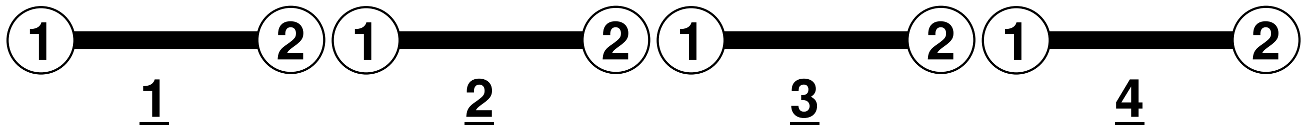
$$T = \alpha_1 + \alpha_2 x$$





Piecewise Linear
各要素内で「温度Tの分布」が線形





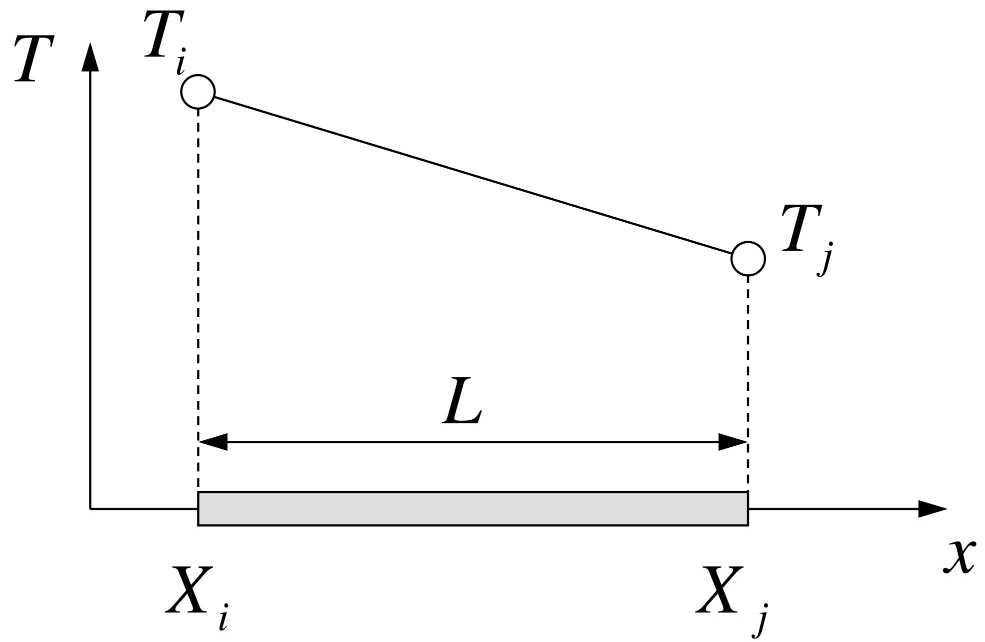
温度勾配は要素内で一定
(節点で不連続となる可能性あり)

一次元線形要素：形状関数（2/4）

- 節点での条件から，係数は以下のように求められる：

$$T=T_i \ @\ x=X_i,\quad T=T_j \ @\ x=X_j$$

$$T_i=\alpha_1+\alpha_2 X_i,\quad T_j=\alpha_1+\alpha_2 X_j$$



- 従って：

$$\alpha_1=\frac{T_i X_j - T_j X_i}{L},\ \alpha_2=\frac{T_j-T_i}{L}$$

- 元の式に代入して，書き直すと以下のようになる

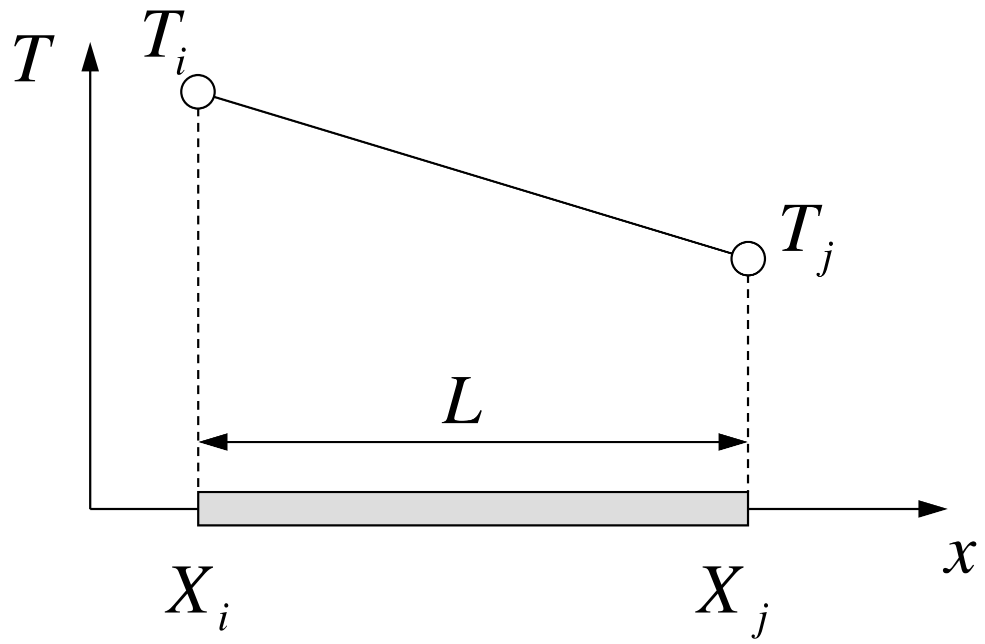
$$T=\underbrace{\boxed{\left(\frac{X_j-x}{L}\right)}}_{N_i}T_i+\underbrace{\boxed{\left(\frac{x-X_i}{L}\right)}}_{N_j}T_j$$

これらのxに関する一次式を形状関数（shape function）または内挿関数（interpolation function）と呼ぶ（N_i, N_jと表す）

一次元線形要素：形状関数（3/4）

- 形状関数$\mathrm{N_k}$は要素を構成する節点数と同じ数だけ存在する：
  - 位置座標のみの関数である
  - 「試行関数」の一種

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



- 形状関数の一次結合により要素内の温度を表す
  - 係数（＝未知数）が節点における温度

$$T = N_i T_i + N_j T_j \longleftrightarrow T_M = \sum_{i=1}^{M} a_i \Psi_i$$

Ψ_i　領域，境界において定義される，位置座標のみ既知関数，互いに独立である：試行関数（trial/test function）と呼ばれる。線形代数における基底（basis）に相当する

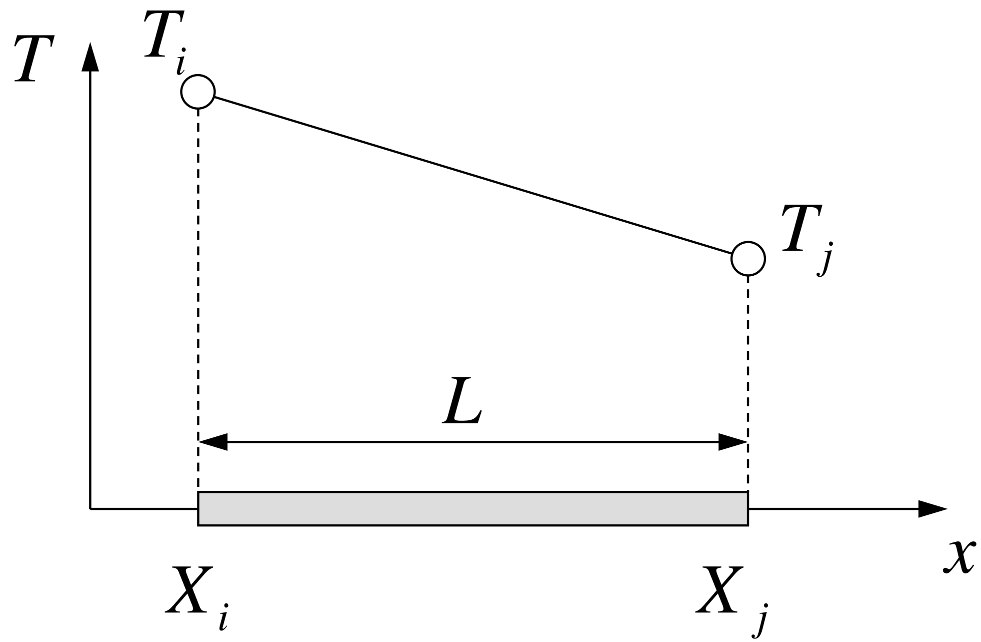
a_i　係数（未知数）

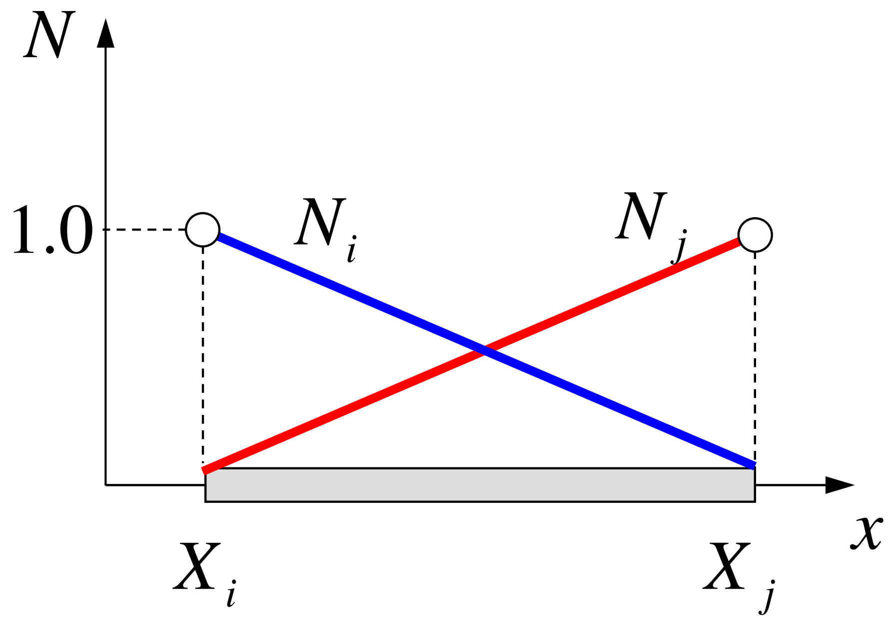
一次元線形要素：形状関数（4/4）

- 形状関数はある節点で1の値をとり，他の節点では必ず0の値をとる：

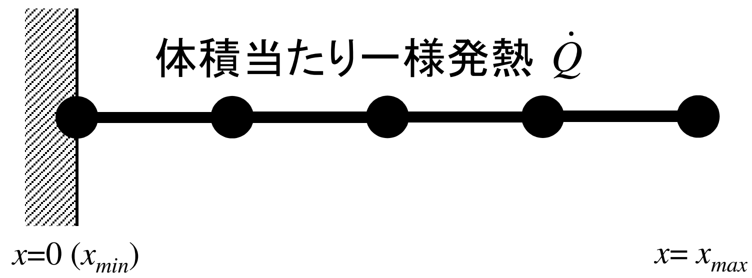
$$N_i = \left(\frac{X_j - x}{L} \right), \quad N_j = \left(\frac{x - X_i}{L} \right)$$

確認してみよう





ガラーキン法の適用（1/4）



- 以下のような一次元熱伝導方程式を考慮する（熱伝導率一定）：

$$\lambda\left(\frac{d^2T}{dx^2}\right)+\dot{Q}=0$$

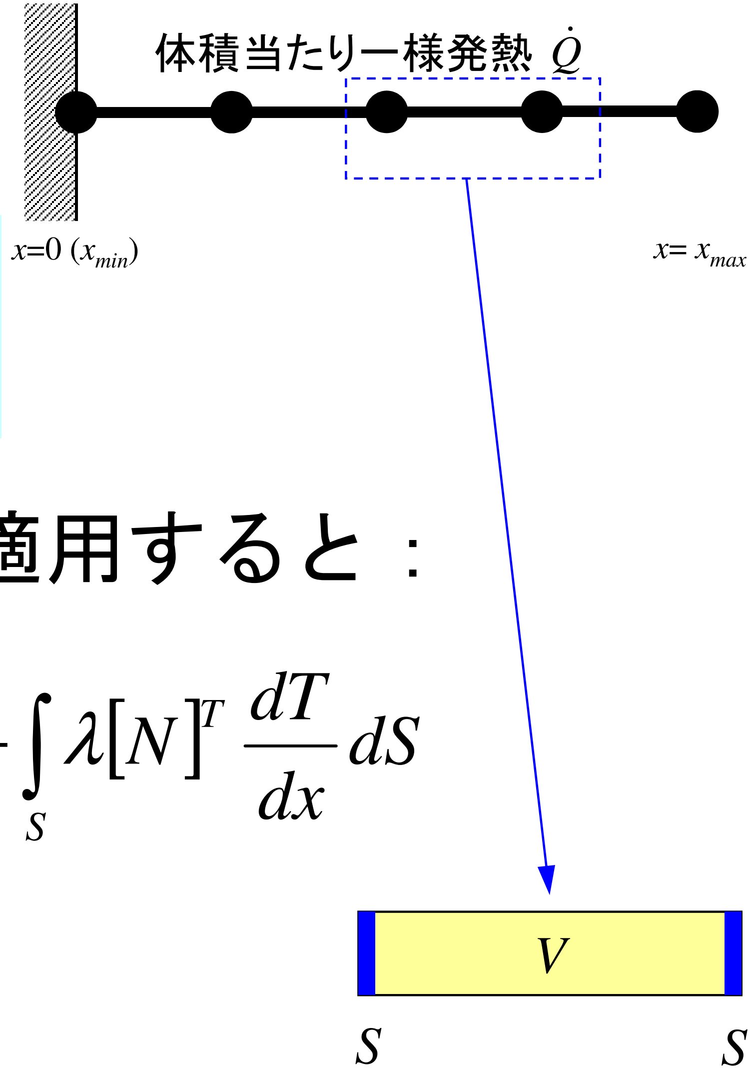
$$T=[N]\{\phi\}$$

要素内の温度分布
（マトリクス形式），節点における温度を 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：

$$\int_V [N]^T\left\{\lambda\left(\frac{d^2T}{dx^2}\right)+\dot{Q}\right\}dV=0$$

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：

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

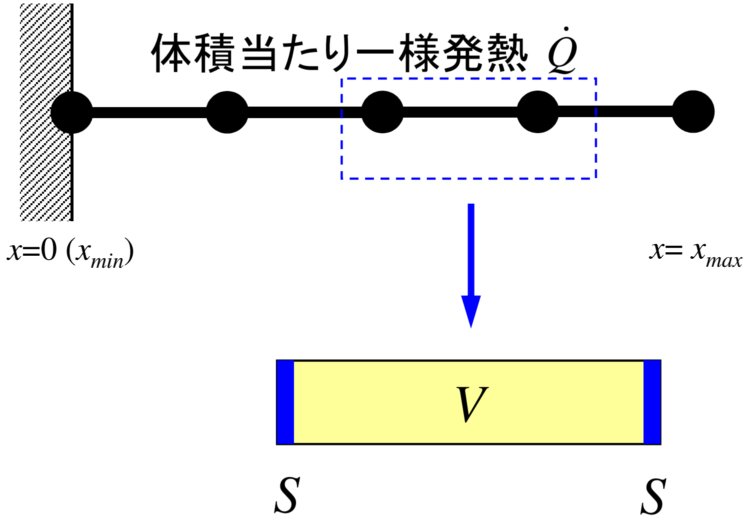
- これに以下を代入する：

$$T = [N]\{\phi\},\quad \frac{dT}{dx} = \frac{d[N]}{dx}\{\phi\} \qquad \bar{q} = -\lambda\frac{dT}{dx}$$

:要素表面熱流量〔QL^{-2}T^{-1}〕

ガラーキン法の適用（3/4）

- 更に体積あたり発熱量の項 $\dot{Q}$ を加えて次式が得られる：



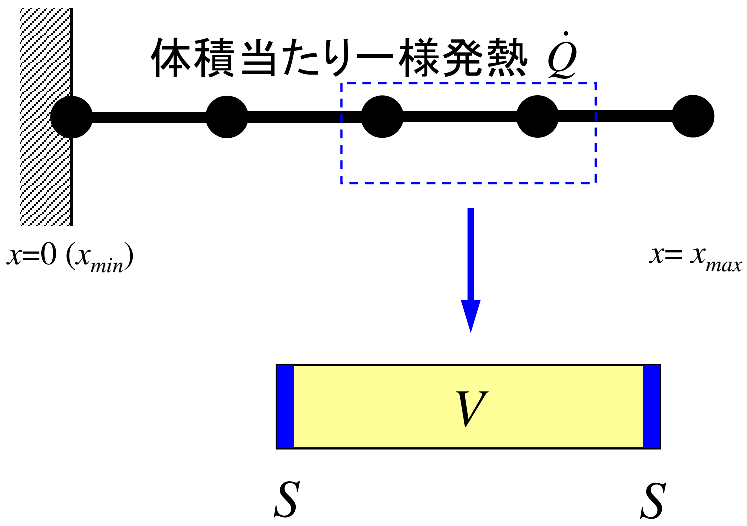
$$
-\int_V \lambda \left(\frac{d[N]^T}{dx} \frac{d[N]}{dx} \right) dV \cdot \{\phi\}
$$

$$
-\int_S \bar{q}[N]^T dS + \int_V \dot{Q}[N]^T dV = 0
$$

- 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）

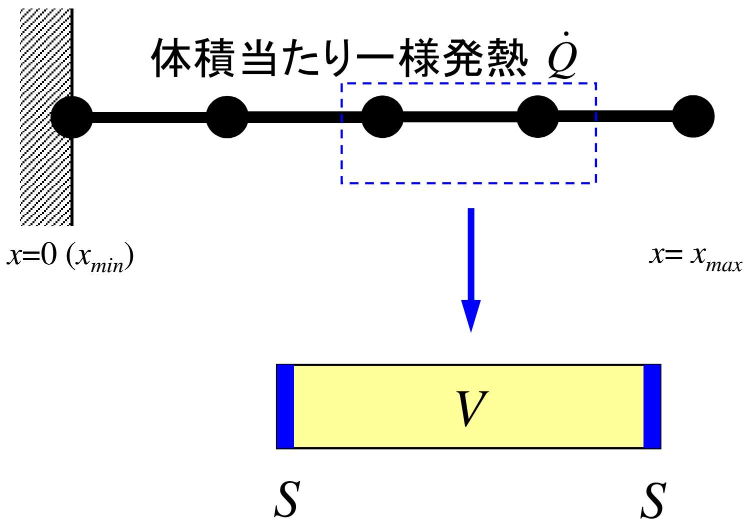
$$-\int_V \lambda\left(\frac{d[N]^T}{dx}\frac{d[N]}{dx}\right)dV\cdot\{\phi\}$$
$$\boxed{-\int_S \bar{q}[N]^T dS}+\int_V \dot{Q}[N]^T dV=0$$



- 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
- （Robin）
  - DirichletとNeumannの線形結合
  - 第三種境界条件
  - 電磁気学：インピーダンス



$$-\int_V \lambda \left(\frac{d[N]^T}{dx} \frac{d[N]}{dx} \right) dV \cdot \{\phi\}$$

$$-\int_S \bar{q}[N]^T dS + \int_V \dot{Q}[N]^T dV = 0$$

$\bar{q} = -\lambda \dfrac{dT}{dx}$ から得られる

境界条件を考慮した弱形式：各要素

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

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

$$[f]^{(e)}=\int_V \dot{Q}[N]^T dV-\int_S \overline{q}[N]^T dS$$

要素単位での積分：$[k]$

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

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

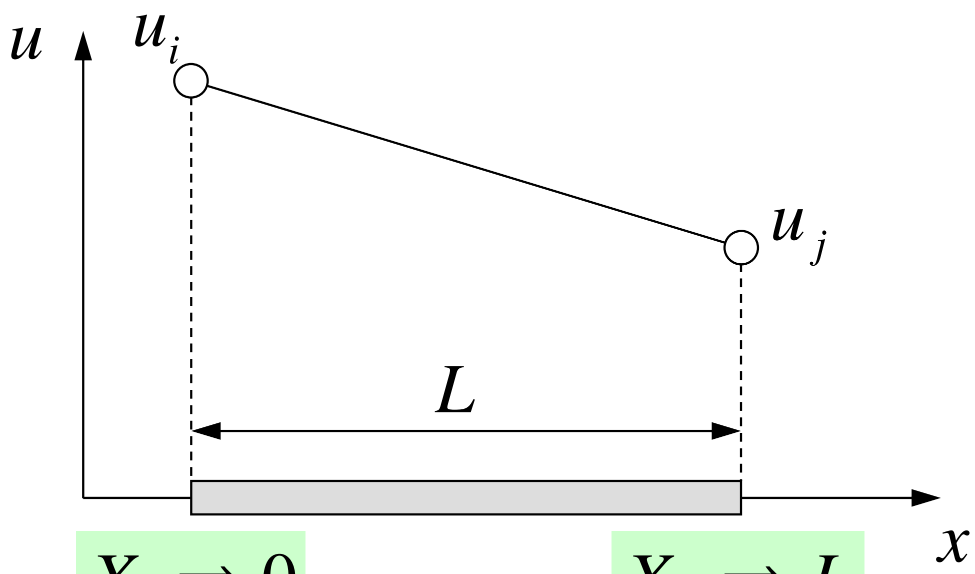
$$\int_V \lambda \left(\frac{d[N]^T}{dx} \frac{d[N]}{dx} \right) dV$$

$$= \lambda \int_0^L \begin{bmatrix} -1/L \\ 1/L \end{bmatrix} \left[-1/L, 1/L\right] A\, dx$$

2x1 matrix　　1x2 matrix

$$= \frac{\lambda A}{L^2} \int_0^L \begin{bmatrix} +1 & -1 \\ -1 & +1 \end{bmatrix} dx = \frac{\lambda A}{L} \begin{bmatrix} +1 & -1 \\ -1 & +1 \end{bmatrix}$$

A: 断面積, L: 要素長さ



$$X_i \Rightarrow 0 \qquad X_j \Rightarrow L$$

$$N_i = \left(1 - \frac{x}{L}\right), \quad N_j = \left(\frac{x}{L}\right)$$

要素単位での積分： $\{f\}$（1/2）

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

$$N_i = \left(1 - \frac{x}{L}\right), \quad N_j = \left(\frac{x}{L}\right)$$

$$\int_V \dot{Q}[N]^T dV = \dot{Q}A\int_0^L \begin{bmatrix} 1 - x/L \\ x/L \end{bmatrix} dx = \frac{\dot{Q}AL}{2}\begin{Bmatrix} 1 \\ 1 \end{Bmatrix}$$

体積当たり発熱

A: 断面積, L: 要素長さ

要素単位での積分：$\{f\}$（2/2）

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

$$\int_V \dot{Q} [N]^T dV = \dot{Q} A \int_0^L \begin{bmatrix} 1 - x/L \\ x/L \end{bmatrix} dx = \frac{\dot{Q} A L}{2} \begin{Bmatrix} 1 \\ 1 \end{Bmatrix}$$

体積当たり発熱

$$\int_S \overline{q} [N]^T dS = \overline{q} A \big|_{x=L} = \overline{q} A \begin{Bmatrix} 0 \\ 1 \end{Bmatrix}, \quad \overline{q} = -\lambda \frac{dT}{dx}$$

表面熱流束

表面熱流束がこの断面のみに
作用しているとすると

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

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

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

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

この連立一次方程式（全体方程式）を解いてやればよい

ファイル準備 on PC

コピー，展開

http://nkl.cc.u-tokyo.ac.jp/files/fem-c.tar
Windows上であれば，¥Cygwin¥home¥YourName へコピー

```
>$ cd
>$ tar xvf fem-c.tar

>$ cd fem-c
```

以下のディレクトリが出来ていることを確認
    1D   fem3D

これらを以降 <$P-TOP>/1d，<$P-TOP>/fem3D

Your PC

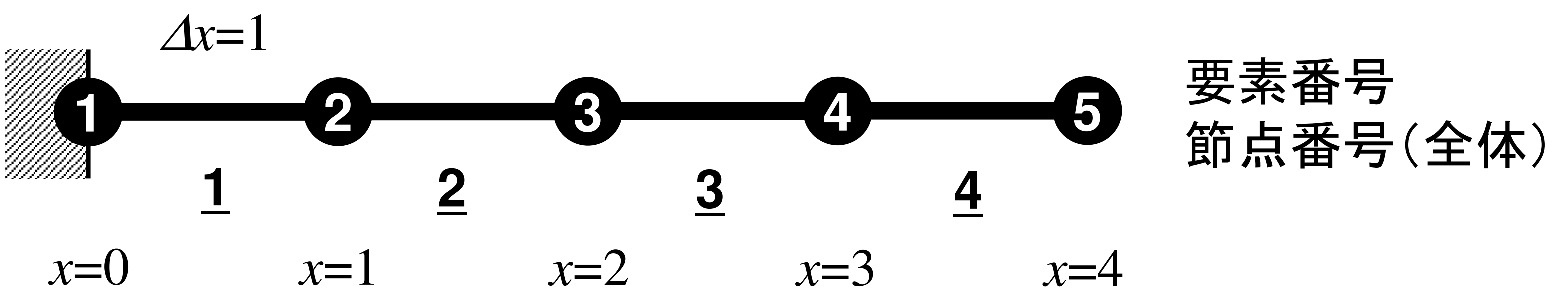
Odyssey

実行（Cygwinではa.exe）

```
>$ cd <$P-TOP>/1d
>$ cc -O 1d.c -lm
>$ ./a.out
```

制御ファイル　**input.dat**

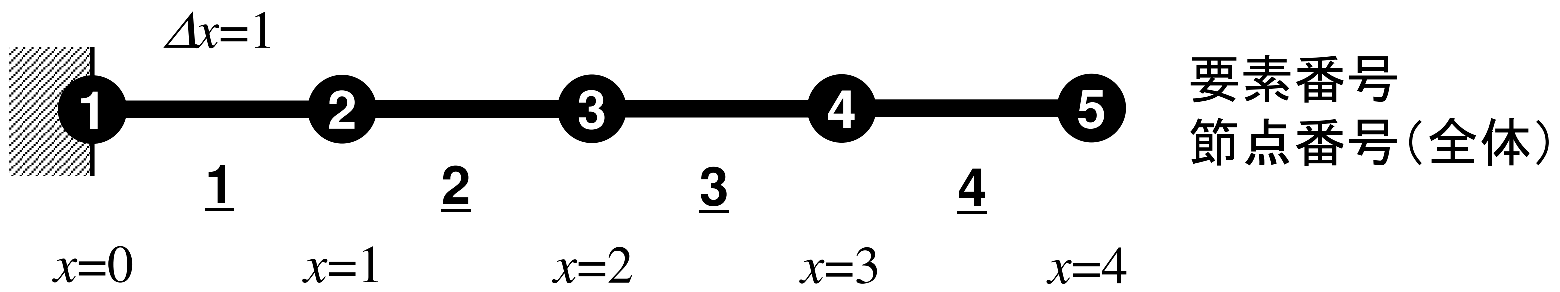
4	**NE**（要素数）
1.0　1.0　1.0　1.0	**Δx**（要素長さ**L**），**Q**, **A**, λ
100	反復回数（CG法後述）
1.e-8	CG法の反復打切誤差



結果

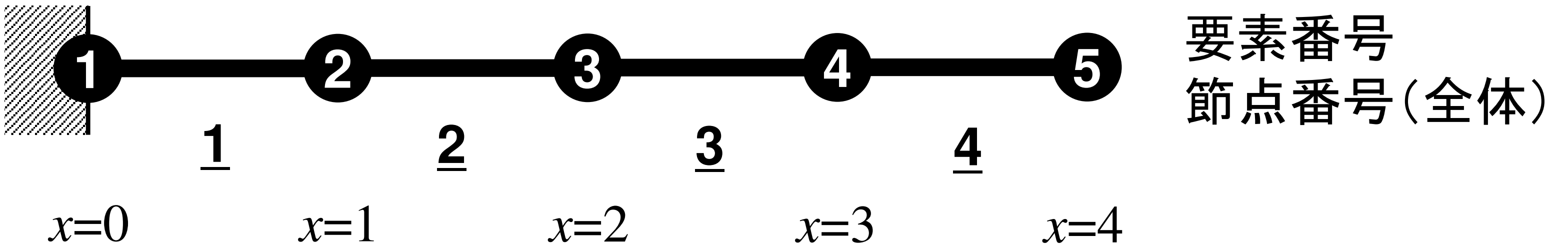
```
>$ ./a.out

  4 iters, RESID=    4.154074e-17

### TEMPERATURE
        1     0.000000E+00     0.000000E+00
        2     3.500000E+00     3.500000E+00
        3     6.000000E+00     6.000000E+00
        4     7.500000E+00     7.500000E+00
        5     8.000000E+00     8.000000E+00
```

計算結果　　解析解



要素方程式とその重ね合わせ（1/3）

- 4要素，5節点の例題



- 要素1の[*k*],{*f*} は以下のようになる：

$$[k]^{(1)} = \frac{\lambda A}{L}\begin{bmatrix} +1 & -1 \\ -1 & +1 \end{bmatrix} \quad \{f\}^{(1)} = \frac{\dot{Q}AL}{2}\begin{Bmatrix} 1 \\ 1 \end{Bmatrix}$$

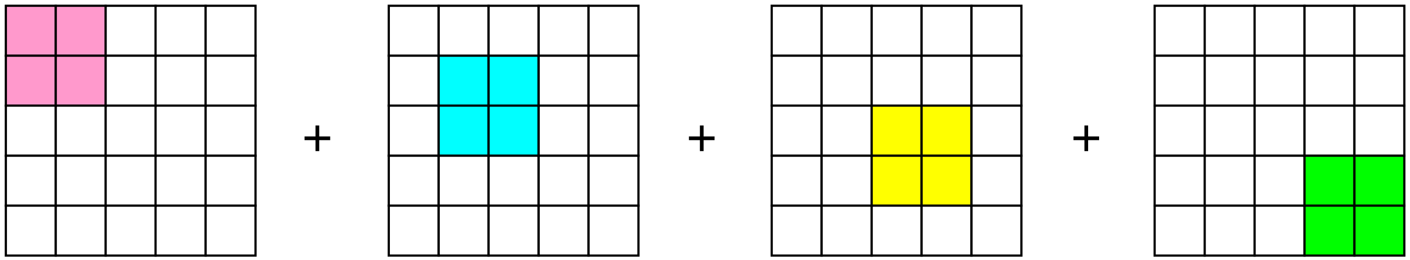
- 要素4については：

$$[k]^{(4)} = \frac{\lambda A}{L}\begin{bmatrix} +1 & -1 \\ -1 & +1 \end{bmatrix} \quad \{f\}^{(4)} = \frac{\dot{Q}AL}{2}\begin{Bmatrix} 1 \\ 1 \end{Bmatrix}$$

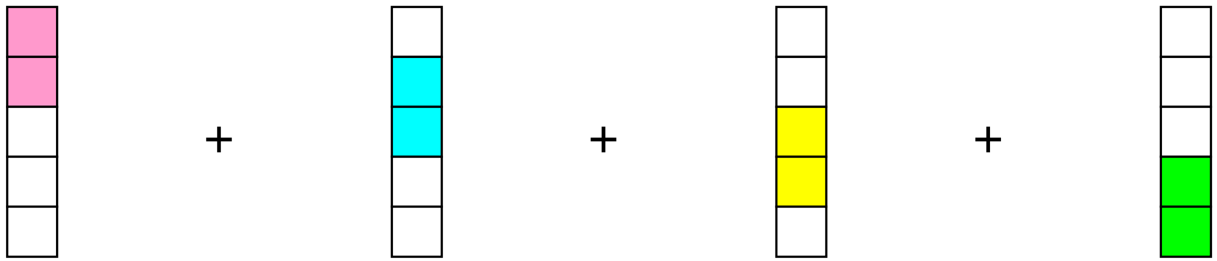
要素方程式とその重ね合わせ（2/3）

- これを順番に足していけばよい

$$[K]=\sum_{e=1}^{4}[k]^{(e)}=$$



$$\{F\}=\sum_{e=1}^{4}\{f\}^{(e)}=$$



要素方程式とその重ね合わせ（3/3）

- 差分との関係

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

$$[K]=\sum_{e=1}^{4}[k]^{(e)}=\left[\begin{bmatrix}+1&-1&&&\\-1&+1&&&\\&&&&\\&&&&\\&&&&\end{bmatrix}+\begin{bmatrix}&&&&\\&+1&-1&&\\&-1&+1&&\\&&&&\\&&&&\end{bmatrix}+\begin{bmatrix}&&&&\\&&&&\\&&+1&-1&\\&&-1&+1&\\&&&&\end{bmatrix}+\begin{bmatrix}&&&&\\&&&&\\&&&&\\&&&+1&-1\\&&&-1&+1\end{bmatrix}\right]\times\frac{\lambda A}{L}$$

$$=\begin{bmatrix}+1&-1&&&\\-1&+2&-1&&\\&-1&+2&-1&\\&&-1&+2&-1\\&&&-1&+1\end{bmatrix}\times\frac{\lambda A}{L}$$

$$-\int_V\left(\frac{d^2T}{dx^2}\right)dV=-\int_V\left(\frac{T_{i+1}-2T_i+T_{i-1}}{L^2}\right)dV$$

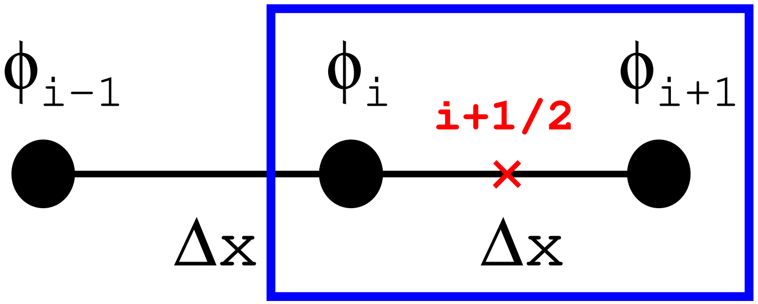
$$=-\left(\frac{T_{i+1}-2T_i+T_{i-1}}{L^2}\right)\cdot AL=-\left(T_{i+1}-2T_i+T_{i-1}\right)\cdot\frac{A}{L}$$

見覚えのある式が出てくる

有限要素法：一般に0の多い「疎」な係数行列

差分法における二階微分係数

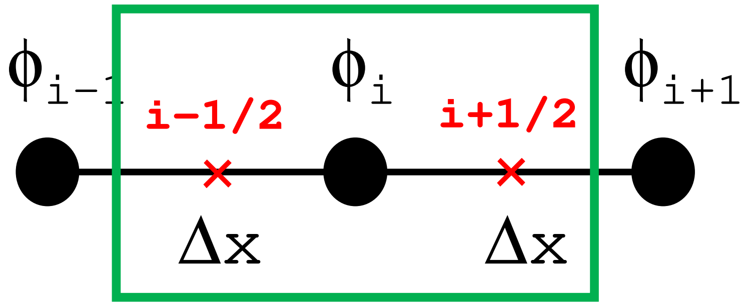
- ×（iとi+1の中点）における微分係数



$$\left(\frac{d\phi}{dx}\right)_{i+1/2} \approx \frac{\phi_{i+1}-\phi_i}{\Delta x}$$

Δx→0となると微分係数の定義そのもの

- iにおける二階微分係数



$$\left(\frac{d^2\phi}{dx^2}\right)_i \approx \frac{\left(\frac{d\phi}{dx}\right)_{i+1/2}-\left(\frac{d\phi}{dx}\right)_{i-1/2}}{\Delta x} = \frac{\frac{\phi_{i+1}-\phi_i}{\Delta x}-\frac{\phi_i-\phi_{i-1}}{\Delta x}}{\Delta x} = \frac{\phi_{i+1}-2\phi_i+\phi_{i-1}}{\Delta x^2}$$

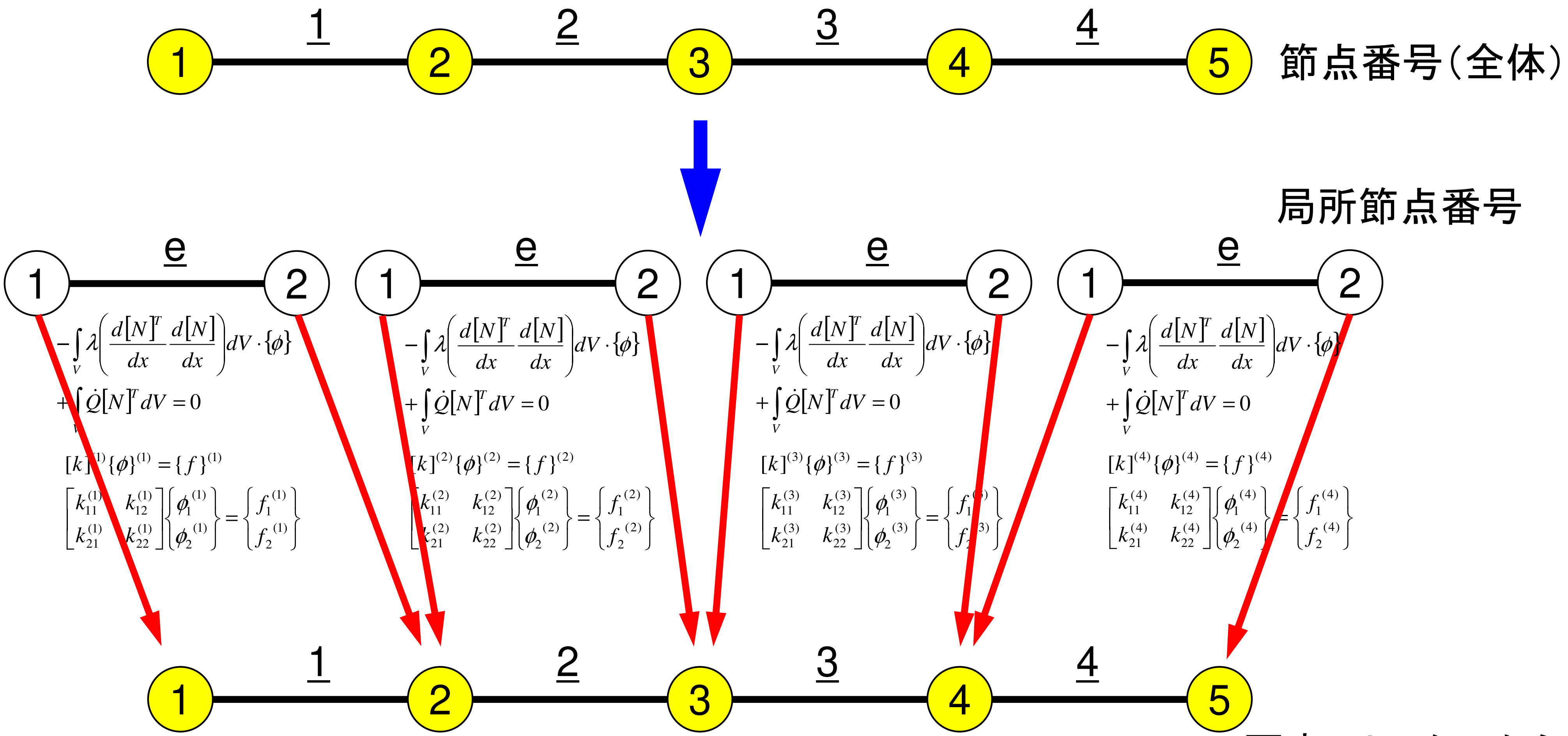
要素ごとに物性，寸法が異なっても簡単に対応が可能

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

$$[K] = \sum_{e=1}^{4} \left[k^{(e)}\right] = \begin{bmatrix} +1 & -1 & & & \\ -1 & +1 & & & \\ & & & & \\ & & & & \\ & & & & \end{bmatrix} \times \frac{\lambda^{(1)} A^{(1)}}{L^{(1)}} + \begin{bmatrix} & & & & \\ & +1 & -1 & & \\ & -1 & +1 & & \\ & & & & \\ & & & & \end{bmatrix} \times \frac{\lambda^{(2)} A^{(2)}}{L^{(2)}}$$

$$\begin{bmatrix} & & & & \\ & & & & \\ & & +1 & -1 & \\ & & -1 & +1 & \\ & & & & \end{bmatrix} \times \frac{\lambda^{(3)} A^{(3)}}{L^{(3)}} + \begin{bmatrix} & & & & \\ & & & & \\ & & & & \\ & & & +1 & -1 \\ & & & -1 & +1 \end{bmatrix} \times \frac{\lambda^{(4)} A^{(4)}}{L^{(4)}}$$

要素処理と全体処理



$$-\int_V \lambda \left(\frac{d[N]^T}{dx}\frac{d[N]}{dx}\right) dV \cdot \{\phi\} + \int_V \dot{Q}[N]^T dV = 0$$

$$[k]^{(1)}\{\phi\}^{(1)} = \{f\}^{(1)} \qquad \begin{bmatrix} k_{11}^{(1)} & k_{12}^{(1)} \\ k_{21}^{(1)} & k_{22}^{(1)} \end{bmatrix} \begin{Bmatrix} \phi_1^{(1)} \\ \phi_2^{(1)} \end{Bmatrix} = \begin{Bmatrix} f_1^{(1)} \\ f_2^{(1)} \end{Bmatrix}$$

$$-\int_V \lambda \left(\frac{d[N]^T}{dx}\frac{d[N]}{dx}\right) dV \cdot \{\phi\} + \int_V \dot{Q}[N]^T dV = 0$$

$$[k]^{(2)}\{\phi\}^{(2)} = \{f\}^{(2)} \qquad \begin{bmatrix} k_{11}^{(2)} & k_{12}^{(2)} \\ k_{21}^{(2)} & k_{22}^{(2)} \end{bmatrix} \begin{Bmatrix} \phi_1^{(2)} \\ \phi_2^{(2)} \end{Bmatrix} = \begin{Bmatrix} f_1^{(2)} \\ f_2^{(2)} \end{Bmatrix}$$

$$-\int_V \lambda \left(\frac{d[N]^T}{dx}\frac{d[N]}{dx}\right) dV \cdot \{\phi\} + \int_V \dot{Q}[N]^T dV = 0$$

$$[k]^{(3)}\{\phi\}^{(3)} = \{f\}^{(3)} \qquad \begin{bmatrix} k_{11}^{(3)} & k_{12}^{(3)} \\ k_{21}^{(3)} & k_{22}^{(3)} \end{bmatrix} \begin{Bmatrix} \phi_1^{(3)} \\ \phi_2^{(3)} \end{Bmatrix} = \begin{Bmatrix} f_1^{(3)} \\ f_2^{(3)} \end{Bmatrix}$$

$$-\int_V \lambda \left(\frac{d[N]^T}{dx}\frac{d[N]}{dx}\right) dV \cdot \{\phi\} + \int_V \dot{Q}[N]^T dV = 0$$

$$[k]^{(4)}\{\phi\}^{(4)} = \{f\}^{(4)} \qquad \begin{bmatrix} k_{11}^{(4)} & k_{12}^{(4)} \\ k_{21}^{(4)} & k_{22}^{(4)} \end{bmatrix} \begin{Bmatrix} \phi_1^{(4)} \\ \phi_2^{(4)} \end{Bmatrix} = \begin{Bmatrix} f_1^{(4)} \\ f_2^{(4)} \end{Bmatrix}$$

$$[K]\{\Phi\} = \{F\}$$

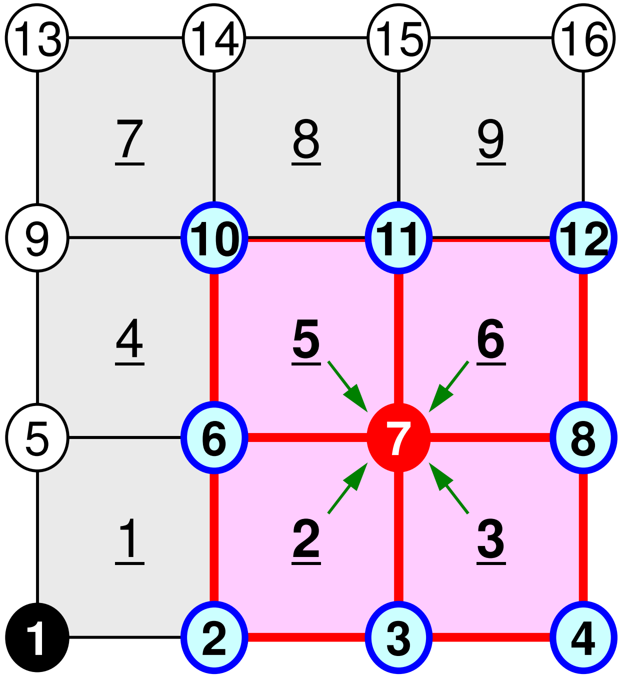
$$\begin{bmatrix} D_1 & AU_{11} & & & \\ AL_{21} & D_2 & AU_{21} & & \\ & AL_{31} & D_3 & AU_{31} & \\ & & AL_{41} & D_4 & AU_{41} \\ & & & AL_{51} & D_5 \end{bmatrix} \begin{Bmatrix} \Phi_1 \\ \Phi_2 \\ \Phi_3 \\ \Phi_4 \\ \Phi_5 \end{Bmatrix} = \begin{Bmatrix} B_1 \\ B_2 \\ B_3 \\ B_4 \\ B_5 \end{Bmatrix}$$

要素マトリクスから全体マトリクスへのマッピングを考える必要がある。

3分でわかる有限要素法

Global/overall Matrix：全体マトリクス

各要素マトリクスを全体マトリクスに足しこむ



$$[K]\{\Phi\}=\{F\}$$

$$
\begin{bmatrix}
D & X & & & X & X & & & & & & & & & & \\
X & D & X & & X & X & X & & & & & & & & & \\
 & X & D & X & & X & X & X & & & & & & & & \\
 & & X & D & & & X & X & & & & & & & & \\
X & X & & & D & X & & & X & X & & & & & & \\
X & X & X & & X & D & X & & X & X & X & & & & & \\
 & X & X & X & & X & D & X & & X & X & X & & & & \\
 & & X & X & & & X & D & & & X & X & & & & \\
 & & & & X & X & & & D & X & & & X & X & & \\
 & & & & X & X & X & & X & D & X & & X & X & X & \\
 & & & & & X & X & X & & X & D & X & & X & X & X \\
 & & & & & & X & X & & & X & D & & & X & X \\
 & & & & & & & & X & X & & & D & X & & \\
 & & & & & & & & X & X & X & & X & D & X & \\
 & & & & & & & & & X & X & X & & X & D & X \\
 & & & & & & & & & & X & X & & & X & D
\end{bmatrix}
\begin{Bmatrix}
\Phi_1 \\ \Phi_2 \\ \Phi_3 \\ \Phi_4 \\ \Phi_5 \\ \Phi_6 \\ \Phi_7 \\ \Phi_8 \\ \Phi_9 \\ \Phi_{10} \\ \Phi_{11} \\ \Phi_{12} \\ \Phi_{13} \\ \Phi_{14} \\ \Phi_{15} \\ \Phi_{16}
\end{Bmatrix}
=
\begin{Bmatrix}
F_1 \\ F_2 \\ F_3 \\ F_4 \\ F_5 \\ F_6 \\ F_7 \\ F_8 \\ F_9 \\ F_{10} \\ F_{11} \\ F_{12} \\ F_{13} \\ F_{14} \\ F_{15} \\ F_{16}
\end{Bmatrix}
$$

- 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

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

- 多くの科学技術計算は，最終的に大規模線形方程式Ax=bを解くことに帰着される。
- 様々な手法が提案されている
  - 疎行列（sparse），密行列（dense）
  - 直接法（direct），反復法（iterative）
- 密行列（dense）
  - 境界要素法，スペクトル法など
- 疎行列（sparse）：0の部分が多い
  - FEM，FDMなど

直接法（Direct Method）

- Gaussの消去法，完全LU分解
  - 逆行列A^{-1}を直接求める（またはそれと同等の計算をする）

- 利点
  - 安定，幅広いアプリケーションに適用可能
    - Partial Pivoting
  - 疎行列，密行列いずれにも適用可能
- 欠点
  - 反復法よりもメモリ，計算時間を必要とする
    - 密行列の場合，$O(N^3)$の計算量
  - 大規模な計算向けではない
    - $O(N^2)$の記憶容量，$O(N^3)$の計算量

反復法(Iterative Method)とは?

Linear Equations
連立一次方程式

$$\begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{pmatrix}$$

$$\mathbf{A} \qquad \mathbf{x} \qquad \mathbf{b}$$

Initial Solution
初期解

$$\mathbf{x}^{(0)} = \begin{pmatrix} x_1^{(0)} \\ x_2^{(0)} \\ \vdots \\ x_n^{(0)} \end{pmatrix}$$

Starting from a initial vector $\mathbf{x}^{(0)}$, iterative method obtains the final converged solutions by iterations

$$\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \cdots$$

反復法（Iterative Method）

- 定常（stationary）法
  - 反復計算中，解ベクトル以外の変数は変化せず
  - SOR，Gauss-Seidel，Jacobiなど
  - 概して遅い

$$\mathbf{Ax} = \mathbf{b} \Rightarrow$$
$$\mathbf{x}^{(k+1)} = \mathbf{Mx}^{(k)} + \mathbf{Nb}$$

- 非定常（nonstationary）法
  - 拘束，最適化条件が加わる
  - Krylov部分空間（subspace）への写像を基底として使用するため，Krylov部分空間法とも呼ばれる
  - CG（Conjugate Gradient：共役勾配法）
  - BiCGSTAB（Bi-Conjugate Gradient Stabilized）
  - GMRES（Generalized Minimal Residual）

反復法（Iterative Method）（続き）

- 利点
  - 直接法と比較して，メモリ使用量，計算量が少ない。
  - 並列計算には適している。

- 欠点
  - 収束性が，アプリケーション，境界条件の影響を受けやすい。
  - 前処理（preconditioning）が重要。

非定常反復法：クリロフ部分空間法（1/2）
Krylov Subspace Method

$$\mathbf{Ax} = \mathbf{b} \Rightarrow \mathbf{x} = \mathbf{b} + (\mathbf{I} - \mathbf{A})\mathbf{x}$$

以下の反復式を導入し$\mathbf{x_0}$, $\mathbf{x_1}$, $\mathbf{x_2}$, ..., $\mathbf{x_k}$を求める：

$$\begin{aligned}\mathbf{x}_k &= \mathbf{b} + (\mathbf{I} - \mathbf{A})\mathbf{x}_{k-1} \\ &= (\mathbf{b} - \mathbf{Ax}_{k-1}) + \mathbf{x}_{k-1} \\ &= \mathbf{r}_{k-1} + \mathbf{x}_{k-1}\end{aligned}$$

$where\ \mathbf{r}_k = \mathbf{b} - \mathbf{Ax}_k$ ：残差ベクトル（residual）

$$\mathbf{x}_k = \mathbf{x}_0 + \sum_{i=0}^{k-1} \mathbf{r}_i$$

$$\begin{aligned}\mathbf{r}_k &= \mathbf{b} - \mathbf{Ax}_k = \mathbf{b} - \mathbf{A}(\mathbf{r}_{k-1} + \mathbf{x}_{k-1}) \\ &= (\mathbf{b} - \mathbf{Ax}_{k-1}) - \mathbf{Ar}_{k-1} = \mathbf{r}_{k-1} - \mathbf{Ar}_{k-1} = (\mathbf{I} - \mathbf{A})\mathbf{r}_{k-1}\end{aligned}$$

非定常反復法：クリロフ部分空間法（2/2）
Krylov Subspace Method

$$\mathbf{x}_k = \mathbf{x}_0 + \sum_{i=0}^{k-1} \mathbf{r}_i = \mathbf{x}_0 + \mathbf{r}_0 + \sum_{i=0}^{k-2} (\mathbf{I} - \mathbf{A}) \mathbf{r}_i = \mathbf{x}_0 + \mathbf{r}_0 + \sum_{i=1}^{k-1} (\mathbf{I} - \mathbf{A})^i \mathbf{r}_0$$

$$\mathbf{z}_k = \mathbf{r}_0 + \sum_{i=1}^{k-1} (\mathbf{I} - \mathbf{A})^i \mathbf{r}_0 = \left[\mathbf{I} + \sum_{i=1}^{k-1} (\mathbf{I} - \mathbf{A})^i \right] \mathbf{r}_0$$

$\mathbf{z}_\mathbf{k}$はk次のクリロフ部分空間（Krylov Subspace）に属するベクトル，問題はクリロフ部分空間からどのようにして解の近似ベクトル$\mathbf{x}_\mathbf{k}$を求めるかにある：

$$\left[\mathbf{r}_0, \mathbf{A}\mathbf{r}_0, \mathbf{A}^2\mathbf{r}_0, \ldots, \mathbf{A}^{k-1}\mathbf{r}_0 \right]$$

代表的な反復法：共役勾配法

- Conjugate Gradient法，略して「CG」法
  - 最も代表的な「非定常」反復法
- 対称正定値行列（Symmetric Positive Definite：SPD）
  - 任意のベクトル$\{x\}$に対して$\{x\}^T[A]\{x\}>0$
  - 全対角成分>0，全固有値>0，全部分行列式>0と同値
  - （ガラーキン法）熱伝導，弾性，ねじり：本コードの場合もSPD
- アルゴリズム
  - 最急降下法（Steepest Descent Method）の変種
  - $\boldsymbol{x}^{(i)} = \boldsymbol{x}^{(i-1)} + \alpha_i \boldsymbol{p}^{(i)}$
    - $x^{(i)}$：反復解，$p^{(i)}$：探索方向，α_i：定数）
  - 厳密解をyとするとき $\{x-y\}^T[A]\{x-y\}$を最小とするような$\{x\}$を求める。
  - 詳細は参考文献参照
    - 例えば：森正武「数値解析（第2版）」（共立出版）

$$
\det \begin{bmatrix}
a_{11} & a_{12} & a_{13} & a_{14} & \cdots & a_{1n} \\
a_{21} & a_{22} & a_{23} & a_{24} & \cdots & a_{2n} \\
a_{31} & a_{32} & a_{33} & a_{34} & \cdots & a_{3n} \\
a_{41} & a_{42} & a_{43} & a_{44} & \cdots & a_{4n} \\
\vdots & \vdots & \vdots & \vdots & & \vdots \\
a_{n1} & a_{n2} & a_{n3} & a_{n4} & \cdots & a_{nn}
\end{bmatrix}
$$

共役勾配法のアルゴリズム

```
Compute r(0)= b-[A]x(0)
for i= 1, 2, …
    z(i-1)= r(i-1)
    ρi-1= r(i-1) z(i-1)
    if i=1
      p(1)= z(0)
     else
      βi-1= ρi-1/ρi-2
      p(i)= z(i-1) + βi-1 p(i-1)
    endif
    q(i)= [A]p(i)
    αi = ρi-1/p(i)q(i)
    x(i)= x(i-1) + αip(i)
    r(i)= r(i-1) - αiq(i)
    check convergence |r|
end
```

- 行列ベクトル積
- ベクトル内積
- ベクトル定数倍の加減

$x^{(i)}$:ベクトル

α_i :スカラー

共役勾配法のアルゴリズム

```
Compute r(0)= b-[A]x(0)
for i= 1, 2, …
    z(i-1)= r(i-1)
    ρi-1= r(i-1) z(i-1)
    if i=1
      p(1)= z(0)
     else
      βi-1= ρi-1/ρi-2
      p(i)= z(i-1) + βi-1 p(i-1)
    endif
    q(i)= [A]p(i)
    αi  = ρi-1/p(i)q(i)
    x(i)= x(i-1) + αip(i)
    r(i)= r(i-1) - αiq(i)
    check convergence |r|
end
```

- 行列ベクトル積
- ベクトル内積
- ベクトル定数倍の加減

$x^{(i)}$:ベクトル

α_i :スカラー

共役勾配法のアルゴリズム

```
Compute r(0)= b-[A]x(0)
for i= 1, 2, …
    z(i-1)= r(i-1)
    ρi-1= r(i-1) z(i-1)
    if i=1
      p(1)= z(0)
     else
      βi-1= ρi-1/ρi-2
      p(i)= z(i-1) + βi-1 p(i-1)
    endif
    q(i)= [A]p(i)
    αi = ρi-1/p(i)q(i)
    x(i)= x(i-1) + αip(i)
    r(i)= r(i-1) - αiq(i)
    check convergence |r|
end
```

- 行列ベクトル積
- ベクトル内積
- ベクトル定数倍の加減

$x^{(i)}$:ベクトル

α_i :スカラー

共役勾配法のアルゴリズム

```
Compute r(0)= b-[A]x(0)
for i= 1, 2, …
    z(i-1)= r(i-1)
    ρi-1= r(i-1) z(i-1)
    if i=1
      p(1)= z(0)
     else
      βi-1= ρi-1/ρi-2
      p(i)= z(i-1) + βi-1 p(i-1)
    endif
    q(i)= [A]p(i)
    αi = ρi-1/p(i)q(i)
    x(i)= x(i-1) + αip(i)
    r(i)= r(i-1) - αiq(i)
    check convergence |r|
end
```

- 行列ベクトル積
- ベクトル内積
- ベクトル定数倍の加減

$x^{(i)}$ ：ベクトル

α_i ：スカラー

共役勾配法のアルゴリズム

```
Compute r(0)= b-[A]x(0)
for i= 1, 2, …
    z(i-1)= r(i-1)
    ρi-1= r(i-1) z(i-1)
    if i=1
      p(1)= z(0)
     else
      βi-1= ρi-1/ρi-2
      p(i)= z(i-1) + βi-1 p(i-1)
    endif
    q(i)= [A]p(i)
    αi = ρi-1/p(i)q(i)
    x(i)= x(i-1) + αip(i)
    r(i)= r(i-1) - αiq(i)
    check convergence |r|
end
```

$x^{(i)}$:ベクトル

α_i :スカラー

CG法アルゴリズムの導出(1/5)

yを厳密解($Ay=b$)とするとき，下式を最小にするxを求める：

$$(x-y)^T[A](x-y)$$

$$\begin{aligned}(x-y)^T[A](x-y) &= (x, Ax)-(y, Ax)-(x, Ay)+(y, Ay)\\ &= (x, Ax)-2(x, Ay)+(y, Ay) = (x, Ax)-2(x,b)+\underline{(y,b)}\end{aligned}$$

定数

従って，下記 $f(x)$ を最小にするxを求めればよい：

$$f(x)=\frac{1}{2}(x, Ax)-(x,b)$$

$$f(x+h)=f(x)+(h, Ax-b)+\frac{1}{2}(h, Ah)$$

任意のベクトル h

$$f(x) = \frac{1}{2}(x, Ax) - (x, b)$$

$$f(x+h) = f(x) + (h, Ax - b) + \frac{1}{2}(h, Ah)$$

•任意のベクトルh

$$f(x+h) = \frac{1}{2}(x+h, A(x+h)) - (x+h, b)$$

$$= \frac{1}{2}(x+h, Ax) + \frac{1}{2}(x+h, Ah) - (x, b) - (h, b)$$

$$= \frac{1}{2}(x, Ax) + \underline{\frac{1}{2}(h, Ax)} + \underline{\frac{1}{2}(x, Ah)} + \frac{1}{2}(h, Ah) - (x, b) - (h, b)$$

$$= \frac{1}{2}(x, Ax) - (x, b) + (h, Ax) - (h, b) + \frac{1}{2}(h, Ah)$$

$$= f(x) + (h, Ax - b) + \frac{1}{2}(h, Ah)$$

CG法アルゴリズムの導出(2/5)

CG法は任意の $x^{(0)}$ から始めて, $f(x)$の最小値を逐次探索する。
今, k番目の近似値 $x^{(k)}$と探索方向$p^{(k)}$が決まったとすると:

$$x^{(k+1)} = x^{(k)} + \alpha_k p^{(k)}$$

$f(x^{(k+1)})$ を最小にするためには:

$$f\left(x^{(k)} + \alpha_k p^{(k)}\right) = \frac{1}{2}\alpha_k{}^2\left(p^{(k)}, Ap^{(k)}\right) - \alpha_k\left(p^{(k)}, b - Ax^{(k)}\right) + f\left(x^{(k)}\right)$$

$$\frac{\partial f\left(x^{(k)} + \alpha_k p^{(k)}\right)}{\partial \alpha_k} = 0 \Rightarrow \alpha_k = \frac{\left(p^{(k)}, b - Ax^{(k)}\right)}{\left(p^{(k)}, Ap^{(k)}\right)} = \frac{\left(p^{(k)}, r^{(k)}\right)}{\left(p^{(k)}, Ap^{(k)}\right)} \quad \textbf{(1)}$$

$r^{(k)} = b - Ax^{(k)}$ は第k近似に対する残差

CG法アルゴリズムの導出(3/5)

残差 $r^{(k)}$も以下の式によって計算できる:

$$r^{(k+1)} = r^{(k)} - \alpha_k A p^{(k)}$$ (2)

$$r^{(k+1)} = b - Ax^{(k+1)},\ r^{(k)} = b - Ax^{(k)}$$

$$r^{(k+1)} - r^{(k)} = -Ax^{(k+1)} + Ax^{(k)} = -\alpha_k A p^{(k)}$$

探索方向を以下の漸化式によって求める:

$$p^{(k+1)} = r^{(k+1)} + \beta_k p^{(k)},\ r^{(0)} = p^{(0)}$$ (3)

本当のところは下記のように(k+1)回目に厳密解 y が求まれば良いのであるが，解がわかっていない場合は困難・・・

$$y = x^{(k+1)} + \alpha_{k+1} p^{(k+1)}$$

CG法アルゴリズムの導出(4/5)

ところで，下式のような都合の良い直交関係がある:

$$\left(Ap^{(k)}, y-x^{(k+1)}\right)=0$$

$$\left(Ap^{(k)}, y-x^{(k+1)}\right)=\left(p^{(k)}, Ay-Ax^{(k+1)}\right)=\left(p^{(k)}, b-Ax^{(k+1)}\right)$$
$$=\left(p^{(k)}, b-A\left[x^{(k)}+\alpha_k p^{(k)}\right]\right)=\left(p^{(k)}, b-Ax^{(k)}-\alpha_k Ap^{(k)}\right)$$
$$=\left(p^{(k)}, r^{(k)}-\alpha_k Ap^{(k)}\right)=\left(p^{(k)}, r^{(k)}\right)-\alpha_k\left(p^{(k)}, Ap^{(k)}\right)=0$$

$$\because \alpha_k=\frac{\left(p^{(k)}, r^{(k)}\right)}{\left(p^{(k)}, Ap^{(k)}\right)}$$

従って以下が成立する:

$$\left(Ap^{(k)}, y-x^{(k+1)}\right)=\left(Ap^{(k)}, \alpha_{k+1}p^{(k+1)}\right)=0 \Rightarrow \left(p^{(k+1)}, Ap^{(k)}\right)=0$$

CG法アルゴリズムの導出(5/5)

$$\left(p^{(k+1)}, Ap^{(k)}\right) = \left(r^{(k+1)} + \beta_k p^{(k)}, Ap^{(k)}\right) = \left(r^{(k+1)}, Ap^{(k)}\right) + \beta_k \left(p^{(k)}, Ap^{(k)}\right) = 0$$

$$\Rightarrow \beta_k = \frac{-\left(r^{(k+1)}, Ap^{(k)}\right)}{\left(p^{(k)}, Ap^{(k)}\right)} \quad \textbf{(4)}$$

$\left(p^{(k+1)}, Ap^{(k)}\right) = 0$ $p^{(k)}$ と $p^{(k+1)}$ が行列Aに関して共役（conjugate）

```
Compute p(0)=r(0)= b-[A]x(0)
for i= 1, 2, …
    calc. αi-1
    x(i)= x(i-1) + αi-1p(i-1)
    r(i)= r(i-1) - αi-1[A]p(i-1)

    check convergence |r|
    (if not converged)
    calc. βi-1
    p(i)= r(i) + βi-1 p(i-1)
end
```

$$\alpha_{i-1} = \frac{\left(p^{(i-1)}, r^{(i-1)}\right)}{\left(p^{(i-1)}, Ap^{(i-1)}\right)}$$

$$\beta_{i-1} = \frac{-\left(r^{(i)}, Ap^{(i-1)}\right)}{\left(p^{(i-1)}, Ap^{(i-1)}\right)}$$

CG法アルゴリズム

任意の(i,j)に対して以下の共役関係が得られる:

$$\left(p^{(i)}, Ap^{(j)}\right)=0\ (i \neq j)$$

探索方向$p^{(k)}$，残差ベクトル$r^{(k)}$についても以下の関係が成立する:

$$\left(r^{(i)}, r^{(j)}\right)=0\ (i \neq j),\quad \left(p^{(k)}, r^{(k)}\right)=\left(r^{(k)}, r^{(k)}\right)$$

N次元空間で互いに直交で一次独立な残差ベクトル $r^{(k)}$ はN個しか存在しない，従って共役勾配法は未知数がN個のときにN回以内に収束する ⇒ 実際は丸め誤差の影響がある（条件数が大きい場合）

Top 10 Algorithms in the 20th Century (SIAM)

http://www.siam.org/news/news.php?id=637

モンテカルロ法，シンプレックス法，クリロフ部分空間法，行列分解法，最適化Fortranコンパイラ，QR法，クイックソート，FFT，整数関係アルゴリズム，FMM（高速多重極法）

Proof (1/3)

Mathematical Induction
数学的帰納法

$$\left(r^{(i)}, r^{(j)}\right) = 0 \; (i \neq j)$$ 直交性

$$\left(p^{(i)}, Ap^{(j)}\right) = 0 \; (i \neq j)$$ 共役性

(1) $$\alpha_k = \frac{\left(p^{(k)}, r^{(k)}\right)}{\left(p^{(k)}, Ap^{(k)}\right)}$$

(2) $$r^{(k+1)} = r^{(k)} - \alpha_k Ap^{(k)}$$

(3) $$p^{(k+1)} = r^{(k+1)} + \beta_k p^{(k)}, \; r^{(0)} = p^{(0)}$$

(4) $$\beta_k = \frac{-\left(r^{(k+1)}, Ap^{(k)}\right)}{\left(p^{(k)}, Ap^{(k)}\right)}$$

Proof (2/3)

Mathematical Induction
数学的帰納法

$$\left(r^{(i)}, r^{(j)}\right) = 0 \ (i \neq j)$$
$$\left(p^{(i)}, Ap^{(j)}\right) = 0 \ (i \neq j) \qquad (*)$$

$(*)$ is satisfied for $i \leq k,\ j \leq k$ where $i \neq j$

if $i < k$

$$\left(r^{(k+1)}, r^{(i)}\right) = \left(r^{(i)}, r^{(k+1)}\right) \overset{(2)}{=} \left(r^{(i)}, r^{(k)} - \alpha_k A p^{(k)}\right)$$
$$\overset{(*)}{=} -\alpha_k \left(r^{(i)}, Ap^{(k)}\right) \overset{(3)}{=} -\alpha_k \left(p^{(i)} - \beta_{i-1} p^{(i-1)}, Ap^{(k)}\right)$$
$$= -\alpha_k \left(p^{(i)}, Ap^{(k)}\right) + \alpha_k \beta_{i-1} \left(p^{(i-1)}, Ap^{(k)}\right) \overset{(*)}{=} 0$$

if $i = k$

$$\left(r^{(k+1)}, r^{(k)}\right) \overset{(2)}{=} \left(r^{(k)}, r^{(k)}\right) - \left(r^{(k)}, \alpha_k Ap^{(k)}\right)$$
$$\overset{(3)}{=} \left(r^{(k)}, r^{(k)}\right) - \left(p^{(k)} - \beta_{k-1} p^{(k-1)}, \alpha_k Ap^{(k)}\right)$$
$$\overset{(*)}{=} \left(r^{(k)}, r^{(k)}\right) - \alpha_k \left(p^{(k)}, Ap^{(k)}\right) \overset{(1)}{=} \left(r^{(k)}, r^{(k)}\right) - \left(p^{(k)}, r^{(k)}\right)$$
$$\overset{(3)}{=} \left(r^{(k)}, r^{(k)}\right) - \left(\beta_{k-1} p^{(k-1)} + r^{(k)}, r^{(k)}\right)$$
$$= -\beta_{k-1} \left(p^{(k-1)}, r^{(k)}\right) \overset{(2)}{=} -\beta_{k-1} \left(p^{(k-1)}, r^{(k-1)} - \alpha_{k-1} Ap^{(k-1)}\right)$$
$$= -\beta_{k-1} \left\{\left(p^{(k-1)}, r^{(k-1)}\right) - \alpha_{k-1} \left(p^{(k-1)}, Ap^{(k-1)}\right)\right\} \overset{(1)}{=} 0$$

(1) $\alpha_k = \dfrac{\left(p^{(k)}, r^{(k)}\right)}{\left(p^{(k)}, Ap^{(k)}\right)}$

(2) $r^{(k+1)} = r^{(k)} - \alpha_k A p^{(k)}$

(3) $p^{(k+1)} = r^{(k+1)} + \beta_k p^{(k)}$

(4) $\beta_k = \dfrac{-\left(r^{(k+1)}, Ap^{(k)}\right)}{\left(p^{(k)}, Ap^{(k)}\right)}$

Proof (3/3)
Mathematical Induction
数学的帰納法

$$\left(r^{(i)}, r^{(j)}\right) = 0 \; (i \neq j)$$
$$\left(p^{(i)}, Ap^{(j)}\right) = 0 \; (i \neq j) \quad (*)$$

(*) is satisfied for $i \leq k,\ j \leq k$ where $i \neq j$

<u>if $i < k$</u>

$$\left(p^{(k+1)}, Ap^{(i)}\right) \overset{(3)}{=} \left(r^{(k+1)} + \beta_k p^{(k)}, Ap^{(i)}\right)$$
$$\overset{(*)}{=} \left(r^{(k+1)}, Ap^{(i)}\right)$$
$$\overset{(2)}{=} \frac{1}{\alpha_i}\left(r^{(k+1)}, r^{(i)} - r^{(i+1)}\right) = 0$$

<u>if $i = k$</u>

$$\left(p^{(k+1)}, Ap^{(k)}\right) \overset{(3)}{=} \left(r^{(k+1)}, Ap^{(k)}\right) + \beta_k \left(p^{(k)}, Ap^{(k)}\right)$$
$$\overset{(4)}{=} 0$$

(1) $\alpha_k = \dfrac{\left(p^{(k)}, r^{(k)}\right)}{\left(p^{(k)}, Ap^{(k)}\right)}$

(2) $r^{(k+1)} = r^{(k)} - \alpha_k Ap^{(k)}$

(3) $p^{(k+1)} = r^{(k+1)} + \beta_k p^{(k)}$

(4) $\beta_k = \dfrac{-\left(r^{(k+1)}, Ap^{(k)}\right)}{\left(p^{(k)}, Ap^{(k)}\right)}$

$$\left(r^{(k+1)}, r^{(k)}\right) = 0$$

$$\begin{aligned}
\left(r^{(k+1)}, r^{(k)}\right) &\overset{(2)}{=} \left(r^{(k)}, r^{(k)}\right) - \left(r^{(k)}, \alpha_k A p^{(k)}\right) \\
&\overset{(3)}{=} \left(r^{(k)}, r^{(k)}\right) - \left(p^{(k)} - \beta_{k-1} p^{(k-1)}, \alpha_k A p^{(k)}\right) \\
&\overset{(*)}{=} \left(r^{(k)}, r^{(k)}\right) - \alpha_k \left(p^{(k)}, A p^{(k)}\right) \overset{(1)}{=} \left(r^{(k)}, r^{(k)}\right) - \left(p^{(k)}, r^{(k)}\right) = 0
\end{aligned}$$

$$\therefore \left(r^{(k)}, r^{(k)}\right) = \left(p^{(k)}, r^{(k)}\right)$$

(1) $\alpha_k = \dfrac{\left(p^{(k)}, r^{(k)}\right)}{\left(p^{(k)}, A p^{(k)}\right)}$

(2) $r^{(k+1)} = r^{(k)} - \alpha_k A p^{(k)}$

(3) $p^{(k+1)} = r^{(k+1)} + \beta_k p^{(k)}$

(4) $\beta_k = \dfrac{-\left(r^{(k+1)}, A p^{(k)}\right)}{\left(p^{(k)}, A p^{(k)}\right)}$

α_k, β_k

実際はα_k, β_kはもうちょっと簡単な形に変形できる:

$$\alpha_k = \frac{\left(p^{(k)}, b - Ax^{(k)}\right)}{\left(p^{(k)}, Ap^{(k)}\right)} = \frac{\left(p^{(k)}, r^{(k)}\right)}{\left(p^{(k)}, Ap^{(k)}\right)} = \frac{\left(r^{(k)}, r^{(k)}\right)}{\left(p^{(k)}, Ap^{(k)}\right)}$$

$$\because \left(p^{(k)}, r^{(k)}\right) = \left(r^{(k)}, r^{(k)}\right)$$

$$\beta_k = \frac{-\left(r^{(k+1)}, Ap^{(k)}\right)}{\left(p^{(k)}, Ap^{(k)}\right)} = \frac{\left(r^{(k+1)}, r^{(k+1)}\right)}{\left(r^{(k)}, r^{(k)}\right)}$$

$$\because \left(r^{(k+1)}, Ap^{(k)}\right) = \frac{\left(r^{(k+1)}, r^{(k)} - r^{(k+1)}\right)}{\alpha_k} = -\frac{\left(r^{(k+1)}, r^{(k+1)}\right)}{\alpha_k}$$

共役勾配法（CG法）のアルゴリズム

```
Compute r(0)= b-[A]x(0)
for i= 1, 2, …
    ρi-1= r(i-1) r(i-1)
    if i=1
      p(1)= r(0)
     else
      βi-1= ρi-1/ρi-2
      p(i)= r(i-1) + βi-1 p(i-1)
    endif
    q(i)= [A]p(i)
    αi  = ρi-1/p(i)q(i)
    x(i)= x(i-1) + αip(i)
    r(i)= r(i-1) - αiq(i)
    check convergence |r|
end
```

$x^{(i)}$: Vector

α_i : Scalar

$$\beta_{i-1} = \frac{\left(r^{(i-1)}, r^{(i-1)}\right)}{\left(r^{(i-2)}, r^{(i-2)}\right)} \quad \begin{matrix} (= \rho_{i-1}) \\ (= \rho_{i-2}) \end{matrix}$$

$$\alpha_i = \frac{\left(r^{(i-1)}, r^{(i-1)}\right)}{\left(p^{(i)}, Ap^{(i)}\right)} \quad (= \rho_{i-1})$$

前処理（preconditioning）とは?

- 反復法の収束は係数行列の固有値分布に依存
  - 固有値分布が少なく，かつ1に近いほど収束が早い（単位行列）
  - 条件数（condition number）（対称正定）＝最大最小固有値比
    - 条件数が1に近いほど収束しやすい
- もとの係数行列 **[A]** に良く似た前処理行列 **[M]** を適用することによって固有値分布を改善する。
  - 前処理行列 **[M]** によって元の方程式 [A]{x}={b}を [A']{x}={b'}へと変換する。ここで $[A']=[M]^{-1}[A]$，$\{b'\}=[M]^{-1}\{b\}$ である。
  - $[A']=[M]^{-1}[A]$ が単位行列に近ければ良い
  - より一般的には [A']{x'}={b'}（$[A']=[M_L]^{-1}[A][M_R]^{-1}$, $\{b'\}=[M_L]^{-1}\{b\}$, $\{x'\}=[M_R]\{x\}$
  - $[M_L]$／$[M_R]$：左／右前処理（left/right preconditioning）

前処理付き共役勾配法のアルゴリズム

```
Compute r(0)= b-[A]x(0)
for i= 1, 2, …
    solve [M]z(i-1)= r(i-1)
    ρi-1= r(i-1) z(i-1)
    if i=1
      p(1)= z(0)
     else
      βi-1= ρi-1/ρi-2
      p(i)= z(i-1) + βi-1 p(i-1)
    endif
    q(i)= [A]p(i)
    αi  = ρi-1/p(i)q(i)
    x(i)= x(i-1) + αip(i)
    r(i)= r(i-1) - αiq(i)
    check convergence |r|
end
```

$$[\mathbf{M}] = [\mathbf{M}_1][\mathbf{M}_2]$$

$$[\mathbf{A}']\mathbf{x}' = \mathbf{b}'$$

$$[\mathbf{A}'] = [\mathbf{M}_1]^{-1}[\mathbf{A}][\mathbf{M}_2]^{-1}$$

$$\mathbf{x}' = [\mathbf{M}_2]\mathbf{x},\quad \mathbf{b}' = [\mathbf{M}_1]^{-1}\mathbf{b}$$

$$\mathbf{p}' \Rightarrow [\mathbf{M}_2]\mathbf{p},\quad \mathbf{r}' \Rightarrow [\mathbf{M}_1]^{-1}\mathbf{r}$$

$$\mathbf{p}'^{(i)} = \mathbf{r}'^{(i-1)} + \beta'_{i-1}\,\mathbf{p}'^{(i-1)}$$

$$[\mathbf{M}_2]\mathbf{p}^{(i)} = [\mathbf{M}_1]^{-1}\mathbf{r}^{(i-1)} + \beta'_{i-1}\,[\mathbf{M}_2]\mathbf{p}^{(i-1)}$$

$$\mathbf{p}^{(i)} = [\mathbf{M}_2]^{-1}[\mathbf{M}_1]^{-1}\mathbf{r}^{(i-1)} + \beta'_{i-1}\,\mathbf{p}^{(i-1)}$$

$$\mathbf{p}^{(i)} = [\mathbf{M}]^{-1}\mathbf{r}^{(i-1)} + \beta'_{i-1}\,\mathbf{p}^{(i-1)}$$

$$\beta'_{i-1} = \left([\mathbf{M}]^{-1}\mathbf{r}^{(i-1)}, \mathbf{r}^{(i-1)}\right) / \left([\mathbf{M}]^{-1}\mathbf{r}^{(i-2)}, \mathbf{r}^{(i-2)}\right)$$

$$\alpha'_{i-1} = \left([\mathbf{M}]^{-1}\mathbf{r}^{(i-1)}, \mathbf{r}^{(i-1)}\right) / \left(\mathbf{p}^{(i-1)}, [\mathbf{A}]\mathbf{p}^{(i-1)}\right)$$

CG法では通常，$[M_2]=[M_1]^T$ である（例：不完全コレスキー分解）従って $[M_1]$ と $[M_2]$ を以下のように定義する：

$$[M_1]=[X]^T,[M_2]=[X],[M]=[M_1][M_2]$$

$$[A']x'=b'$$

$$[A']=[M_1]^{-1}[A][M_2]^{-1}=[[X]^T]^{-1}[A][X]^{-1}=[X]^{-T}[A][X]^{-1}$$

$$x'=[X]x,\quad b'=[X]^{-T}b,\quad r'=[X]^{-T}r$$

$$\alpha'_{i-1}=\frac{\left(r'^{(i-1)},r'^{(i-1)}\right)}{\left(p'^{(i-1)},A'p'^{(i-1)}\right)}=\frac{\left([X]^{-T}r^{(i-1)},[X]^{-T}r^{(i-1)}\right)}{\left([X]p^{(i-1)},[X]^{-T}[A][X]^{-1}[X]p^{(i-1)}\right)}$$

$$=\frac{\left(\left([X]^{-T}r^{(i-1)}\right)^T,[X]^{-T}r^{(i-1)}\right)}{\left(\left([X]p^{(i-1)}\right)^T,[X]^{-T}[A]p^{(i-1)}\right)}=\frac{\left(\left(r^{(i-1)}\right)^T[X]^{-1},\left[X^T\right]^{-1}r^{(i-1)}\right)}{\left(\left(p^{(i-1)}\right)^T[X]^T,[X]^{-T}[A]p^{(i-1)}\right)}$$

$$=\frac{\left(r^{(i-1)},\left[\left[X^T\right][X]\right]^{-1}r^{(i-1)}\right)}{\left(p^{(i-1)},[A]p^{(i-1)}\right)}=\frac{\left(r^{(i-1)},[M]^{-1}r^{(i-1)}\right)}{\left(p^{(i-1)},[A]p^{(i-1)}\right)}=\frac{\left(r^{(i-1)},z^{(i-1)}\right)}{\left(p^{(i-1)},[A]p^{(i-1)}\right)}$$

$$\beta'_{i-1} = \frac{\left(r'^{(i-1)}, r'^{(i-1)}\right)}{\left(r'^{(i-2)}, r'^{(i-2)}\right)} == \frac{\left([\mathrm{X}]^{-T} r^{(i-1)}, [\mathrm{X}]^{-T} r^{(i-1)}\right)}{\left([\mathrm{X}]^{-T} r^{(i-2)}, [\mathrm{X}]^{-T} r^{(i-2)}\right)}$$

$$= \frac{\left(\left([\mathrm{X}]^{-T} r^{(i-1)}\right)^{T}, [\mathrm{X}]^{-T} r^{(i-1)}\right)}{\left(\left([\mathrm{X}]^{-T} r^{(i-2)}\right)^{T}, [\mathrm{X}]^{-T} r^{(i-2)}\right)} = \frac{\left(\left(r^{(i-1)}\right)^{T} [\mathrm{X}]^{-1}, \left[\mathrm{X}^{T}\right]^{-1} r^{(i-1)}\right)}{\left(\left(r^{(i-2)}\right)^{T} [\mathrm{X}]^{-1}, \left[\mathrm{X}^{T}\right]^{-1} r^{(i-2)}\right)}$$

$$= \frac{\left(r^{(i-1)}, \left[\left[\mathrm{X}^{T}\right][\mathrm{X}]\right]^{-1} r^{(i-1)}\right)}{\left(r^{(i-2)}, \left[\left[\mathrm{X}^{T}\right][\mathrm{X}]\right]^{-1} r^{(i-2)}\right)} = \frac{\left(r^{(i-1)}, [\mathrm{M}]^{-1} r^{(i-1)}\right)}{\left(r^{(i-2)}, [\mathrm{M}]^{-1} r^{(i-2)}\right)} = \frac{\left(r^{(i-1)}, z^{(i-1)}\right)}{\left(r^{(i-2)}, z^{(i-2)}\right)}$$

前処理付き共役勾配法

```
Compute r(0)= b-[A]x(0)
for i= 1, 2, …
    solve [M]z(i-1)= r(i-1)
    ρi-1= r(i-1) z(i-1)
    if i=1
      p(1)= z(0)
     else
      βi-1= ρi-1/ρi-2
      p(i)= z(i-1) + βi-1 p(i-1)
    endif
    q(i)= [A]p(i)
    αi  = ρi-1/p(i)q(i)
    x(i)= x(i-1) + αip(i)
    r(i)= r(i-1) - αiq(i)
    check convergence |r|
end
```

下記の方程式を解く:

$$\{z\} = [M]^{-1}\{r\}$$

近似逆行列

$$[M]^{-1} \approx [A]^{-1}, \quad [M] \approx [A]$$

究極の前処理:本当の逆行列

$$[M]^{-1} = [A]^{-1}, \quad [M] = [A]$$

対角スケーリング:簡単だが弱い

$$[M]^{-1} = [D]^{-1}, \quad [M] = [D]$$

ILU(0), IC(0)

- 最もよく使用されている前処理（疎行列用）
  - 不完全LU分解
    - Incomplete LU Factorization
  - 不完全コレスキー分解
    - Incomplete Cholesky Factorization（対称行列）

- 不完全な直接法
  - もとの行列が疎でも，逆行列は疎とは限らない。
  - fill-in
  - もとの行列と同じ非ゼロパターン（fill-in無し）を持っているのがILU（0），IC（0）

対角スケーリング，点ヤコビ前処理

- 前処理行列として，もとの行列の対角成分のみを取り出した行列を前処理行列 [M] とする。
  - 対角スケーリング，点ヤコビ（point-Jacobi）前処理

$$
[M]=\begin{bmatrix} D_1 & 0 & \dots & 0 & 0 \\ 0 & D_2 & & 0 & 0 \\ \dots & & \dots & & \dots \\ 0 & 0 & & D_{N-1} & 0 \\ 0 & 0 & \dots & 0 & D_N \end{bmatrix}
$$

- **solve [M]z$^{(i-1)}$= r$^{(i-1)}$**という場合に逆行列を簡単に求めることができる。
- 簡単な問題では収束する。
- 1d.f, 1d.cはこの手法を使用している

- 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

有限要素法で得られるマトリクス

- 疎行列
  - 0が多い
- $A(i,j)$のように正方行列の全成分を記憶することは疎行列では非効率的
  - 「密」行列向け

$$
\begin{bmatrix}
D & X & & & X & X & & & & & & & & & & \\
X & D & X & & X & X & X & & & & & & & & & \\
& X & D & X & & X & X & X & & & & & & & & \\
& & X & D & & & X & X & & & & & & & & \\
X & X & & & D & X & & & X & X & & & & & & \\
X & X & X & & X & D & X & & X & X & X & & & & & \\
& X & X & X & & X & D & X & & X & X & X & & & & \\
& & X & X & & & X & D & & & X & X & & & & \\
& & & & X & X & & & D & X & & & X & X & & \\
& & & & X & X & X & & X & D & X & & X & X & X & \\
& & & & & X & X & X & & X & D & X & & X & X & X \\
& & & & & & X & X & & & X & D & & & X & X \\
& & & & & & & & X & X & & & D & X & & \\
& & & & & & & & X & X & X & & X & D & X & \\
& & & & & & & & & X & X & X & & X & D & X \\
& & & & & & & & & & X & X & & & X & D
\end{bmatrix}
\begin{Bmatrix}
\Phi_1 \\ \Phi_2 \\ \Phi_3 \\ \Phi_4 \\ \Phi_5 \\ \Phi_6 \\ \Phi_7 \\ \Phi_8 \\ \Phi_9 \\ \Phi_{10} \\ \Phi_{11} \\ \Phi_{12} \\ \Phi_{13} \\ \Phi_{14} \\ \Phi_{15} \\ \Phi_{16}
\end{Bmatrix}
=
\begin{Bmatrix}
F_1 \\ F_2 \\ F_3 \\ F_4 \\ F_5 \\ F_6 \\ F_7 \\ F_8 \\ F_9 \\ F_{10} \\ F_{11} \\ F_{12} \\ F_{13} \\ F_{14} \\ F_{15} \\ F_{16}
\end{Bmatrix}
$$

- 有限要素法：非零非対角成分の数は高々「数百」規模
  - 例えば未知数が10^8個あるとすると記憶容量（ワード数）は
    - 正方行列：$O(10^{16}) \sim O(10^{17})$ bytes for DP: 100PB (100x K computer)
    - 非零非対角成分数：$O(10^{10})$：100GB
- **非零成分のみ記憶するのが効率的**

1d.f, 1d.cにおけるマトリクス関連変数

変数名	型	サイズ	内容
N	I	-	未知数総数
NPLU	I	-	連立一次方程式係数マトリクス非対角成分総数
Diag(:)	R	N	連立一次方程式係数マトリクス対角成分
PHI(:)	R	N	連立一次方程式未知数ベクトル
Rhs(:)	R	N	連立一次方程式右辺ベクトル
Index(:)	I	0:N N+1	係数マトリクス非対角成分要素番号用一次元圧縮配列(非対角成分数)
Item(:)	I	NPLU	係数マトリクス非対角成分要素番号用一次元圧縮配列(非対角成分要素(列)番号)
AMat(:)	R	NPLU	係数マトリクス非対角成分要素番号用一次元圧縮配列(非対角成分)

非零非対角成分のみを格納する
Compressed Row Storage法を使用している。

行列ベクトル積への適用

(非零)非対角成分のみを格納，疎行列向け方法
Compressed Row Storage (CRS)

Diag [i]	対角成分(実数，[N])
Index[i]	非対角成分数に関する一次元配列 (通し番号)(整数，[N+1])
Item[k]	非対角成分の要素(列)番号 (整数，[Index[N]])
AMat[k]	非対角成分 (実数，[Index[N]])

```
{Y}=[A]{X}

for(i=0;i<N;i++){
        Y[i] = Diag[i] * X[i];
        for(k=Index[i];k<Index[i+1];k++){
                Y[i] += AMat[k]*X[Item[k]];
        }
}
```

$$
\begin{bmatrix}
D & X & & & X & X & & & & & & & & & & \\
X & D & X & & X & X & X & & & & & & & & & \\
& X & D & X & & X & X & X & & & & & & & & \\
& & X & D & & & X & X & & & & & & & & \\
X & X & & & D & X & & & X & X & & & & & & \\
X & X & X & & X & D & X & & X & X & X & & & & & \\
& X & X & X & & X & D & X & & X & X & X & & & & \\
& & X & X & & & X & D & & & X & X & & & & \\
& & & & X & X & & & D & X & & & X & X & & \\
& & & & X & X & X & & X & D & X & & X & X & X & \\
& & & & & X & X & X & & X & D & X & & X & X & X \\
& & & & & & X & X & & & X & D & & & X & X \\
& & & & & & & & X & X & & & D & X & & \\
& & & & & & & & X & X & X & & X & D & X & \\
& & & & & & & & & X & X & X & & X & D & X \\
& & & & & & & & & & X & X & & & X & D
\end{bmatrix}
\begin{Bmatrix}
\Phi_1 \\ \Phi_2 \\ \Phi_3 \\ \Phi_4 \\ \Phi_5 \\ \Phi_6 \\ \Phi_7 \\ \Phi_8 \\ \Phi_9 \\ \Phi_{10} \\ \Phi_{11} \\ \Phi_{12} \\ \Phi_{13} \\ \Phi_{14} \\ \Phi_{15} \\ \Phi_{16}
\end{Bmatrix}
=
\begin{Bmatrix}
F_1 \\ F_2 \\ F_3 \\ F_4 \\ F_5 \\ F_6 \\ F_7 \\ F_8 \\ F_9 \\ F_{10} \\ F_{11} \\ F_{12} \\ F_{13} \\ F_{14} \\ F_{15} \\ F_{16}
\end{Bmatrix}
$$

行列ベクトル積：密行列⇒とても簡単

$$
\begin{bmatrix}
a_{11} & a_{12} & \cdots & a_{1,N-1} & a_{1,N} \\
a_{21} & a_{22} & & a_{2,N-1} & a_{2,N} \\
\cdots & & \cdots & & \cdots \\
a_{N-1,1} & a_{N-1,2} & & a_{N-1,N-1} & a_{N-1,N} \\
a_{N,1} & a_{N,2} & \cdots & a_{N,N-1} & a_{N,N}
\end{bmatrix}
\begin{Bmatrix} x_1 \\ x_2 \\ \vdots \\ x_{N-1} \\ x_N \end{Bmatrix}
=
\begin{Bmatrix} y_1 \\ y_2 \\ \vdots \\ y_{N-1} \\ y_N \end{Bmatrix}
$$

```
{Y}= [A]{X}

for (i=0;j<N;j++){
  Y[i]= 0.d0;
  for (j=0;j<N;j++){
    Y[i]= Y[i] + A[i][j]*X[j]
  }
}
```

Compressed Row Storage (CRS)

$$
\begin{array}{c@{}c}
 & \begin{array}{cccccccc} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \end{array} \\
\begin{array}{c} 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \\ 8 \end{array} &
\begin{bmatrix}
1.1 & 2.4 & 0 & 0 & 3.2 & 0 & 0 & 0 \\
4.3 & 3.6 & 0 & 2.5 & 0 & 3.7 & 0 & 9.1 \\
0 & 0 & 5.7 & 0 & 1.5 & 0 & 3.1 & 0 \\
0 & 4.1 & 0 & 9.8 & 2.5 & 2.7 & 0 & 0 \\
3.1 & 9.5 & 10.4 & 0 & 11.5 & 0 & 4.3 & 0 \\
0 & 0 & 6.5 & 0 & 0 & 12.4 & 9.5 & 0 \\
0 & 6.4 & 2.5 & 0 & 0 & 1.4 & 23.1 & 13.1 \\
0 & 9.5 & 1.3 & 9.6 & 0 & 3.1 & 0 & 51.3
\end{bmatrix}
\end{array}
$$

Compressed Row Storage (CRS)
Cでは0番から番号付け

	0	1	2	3	4	5	6	7
0	**1.1** ◎	2.4 ①			3.2 ④			
1	4.3 ◎	**3.6** ①		2.5 ③		3.7 ⑤		9.1 ⑦
2			**5.7** ②		1.5 ④		3.1 ⑥	
3		4.1 ①		**9.8** ③	2.5 ④	2.7 ⑤		
4	3.1 ◎	9.5 ①	10.4 ②		**11.5** ④		4.3 ⑥	
5			6.5 ②			**12.4** ⑤	9.5 ⑥	
6		6.4 ①	2.5 ②			1.4 ⑤	**23.1** ⑥	13.1 ⑦
7		9.5 ①	1.3 ②	9.6 ③		3.1 ⑤		**51.3** ⑦

N= 8

対角成分

```
Diag[0]=  1.1
Diag[1]=  3.6
Diag[2]=  5.7
Diag[3]=  9.8
Diag[4]= 11.5
Diag[5]= 12.4
Diag[6]= 23.1
Diag[7]= 51.3
```

Compressed Row Storage (CRS)

		0	1	2	3	4	5	6	7
0	1.1 ◎		2.4 ①			3.2 ④			
1	3.6 ①	4.3 ◎			2.5 ③		3.7 ⑤		9.1 ⑦
2	5.7 ②					1.5 ④		3.1 ⑥	
3	9.8 ③		4.1 ①			2.5 ④	2.7 ⑤		
4	11.5 ④	3.1 ◎	9.5 ①	10.4 ②				4.3 ⑥	
5	12.4 ⑤			6.5 ②				9.5 ⑥	
6	23.1 ⑥		6.4 ①	2.5 ②			1.4 ⑤		13.1 ⑦
7	51.3 ⑦		9.5 ①	1.3 ②	9.6 ③		3.1 ⑤		

Compressed Row Storage (CRS)

						非対角成分数
0	1.1 ◎	2.4 ①	3.2 ④			2
1	3.6 ①	4.3 ◎	2.5 ③	3.7 ⑤	9.1 ⑦	4
2	5.7 ②	1.5 ④	3.1 ⑥			2
3	9.8 ③	4.1 ①	2.5 ④	2.7 ⑤		3
4	11.5 ④	3.1 ◎	9.5 ①	10.4 ②	4.3 ⑥	4
5	12.4 ⑤	6.5 ②	9.5 ⑥			2
6	23.1 ⑥	6.4 ①	2.5 ②	1.4 ⑤	13.1 ⑦	4
7	51.3 ⑦	9.5 ①	1.3 ②	9.6 ③	3.1 ⑤	4

```
Index[0]=  0
Index[1]=  2
Index[2]=  6
Index[3]=  8
Index[4]= 11
Index[5]= 15
Index[6]= 17
Index[7]= 21
Index[8]= 25
```

NPLU= 25
(=Index[N])

Index[i]~Index[i+1]-1番目がi行目の非対角成分

Compressed Row Storage (CRS)

Index[0]= 0

						非対角成分数	
0	1.1 ◎	2.4 ①,0	3.2 ④,1			2	Index[1]= 2
1	3.6 ①	4.3 ◎,2	2.5 ③,3	3.7 ⑤,4	9.1 ⑦,5	4	Index[2]= 6
2	5.7 ②	1.5 ④,6	3.1 ⑥,7			2	**Index[3]= 8**
3	9.8 ③	**4.1 ①,8**	**2.5 ④,9**	**2.7 ⑤,10**		**3**	**Index[4]= 11**
4	11.5 ④	3.1 ◎,11	9.5 ①,12	10.4 ②,13	4.3 ⑥,14	4	Index[5]= 15
5	12.4 ⑤	6.5 ②,15	9.5 ⑥,16			2	Index[6]= 17
6	23.1 ⑥	6.4 ①,17	2.5 ②,18	1.4 ⑤,19	13.1 ⑦,20	4	Index[7]= 21
7	51.3 ⑦	9.5 ①,21	1.3 ②,22	9.6 ③,23	3.1 ⑤,24	4	Index[8]= 25

NPLU= 25
(=Index[N])

Index[i]~Index[i+1]-1番目がi行目の非対角成分

Compressed Row Storage (CRS)

0	1.1 ◎	2.4 ①,0	3.2 ④,1		
1	3.6 ①	4.3 ◎,2	2.5 ③,3	3.7 ⑤,4	9.1 ⑦,5
2	5.7 ②	**1.5 ④,6**	3.1 ⑥,7		
3	9.8 ③	4.1 ①,8	2.5 ④,9	2.7 ⑤,10	
4	11.5 ④	3.1 ◎,11	9.5 ①,12	10.4 ②,13	4.3 ⑥,14
5	12.4 ⑤	6.5 ②,15	9.5 ⑥,16		
6	23.1 ⑥	6.4 ①,17	**2.5 ②,18**	1.4 ⑤,19	13.1 ⑦,20
7	51.3 ⑦	9.5 ①,21	1.3 ②,22	9.6 ③,23	3.1 ⑤,24

例:

```
Item[ 6]= 4, AMat[ 6]= 1.5
Item[18]= 2, AMat[18]= 2.5
```

Compressed Row Storage (CRS)

0	1.1 ◎	2.4 ①,0	3.2 ④,1		
1	3.6 ①	4.3 ◎,2	2.5 ③,3	3.7 ⑤,4	9.1 ⑦,5
2	5.7 ②	1.5 ④,6	3.1 ⑥,7		
3	9.8 ③	4.1 ①,8	2.5 ④,9	2.7 ⑤,10	
4	11.5 ④	3.1 ◎,11	9.5 ①,12	10.4 ②,13	4.3 ⑥,14
5	12.4 ⑤	6.5 ②,15	9.5 ⑥,16		
6	23.1 ⑥	6.4 ①,17	2.5 ②,18	1.4 ⑤,19	13.1 ⑦,20
7	51.3 ⑦	9.5 ①,21	1.3 ②,22	9.6 ③,23	3.1 ⑤,24

`Diag [i]` 対角成分（実数，[N]）
`Index[i]` 非対角成分数に関する一次元配列（通し番号）（整数，[N+1]）
`Item[k]` 非対角成分の要素（列）番号（整数，[Index[N]]）
`AMat[k]` 非対角成分（実数，[Index[N]]）

```
{Y}=[A]{X}

for(i=0;i<N;i++){
    Y[i] = Diag[i] * X[i];
    for(k=Index[i];k<Index[i+1];k++){
        Y[i] += AMat[k]*X[Item[k]];
      }
}
```

- 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

有限要素法の処理:プログラム

- 初期化
  - 制御変数読み込み
  - 座標読み込み⇒要素生成(N:節点数, NE:要素数)
  - 配列初期化(全体マトリクス, 要素マトリクス)
  - 要素⇒全体マトリクスマッピング(Index, Item)
- マトリクス生成
  - 要素単位の処理(do icel= 1, NE)
    - 要素マトリクス計算
    - 全体マトリクスへの重ね合わせ
  - 境界条件の処理
- 連立一次方程式
  - 共役勾配法(CG)

プログラム：1d.c（1/6）

諸変数

```
/*
//   1D Steady-State Heat Transfer
//   FEM with Piece-wise Linear Elements
//   CG (Conjugate Gradient) Method
//
//   d/dx(CdT/dx) + Q = 0
//   T=0@x=0
*/
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <assert.h>

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

        double dX, Resid, Eps, Area, QV, COND;
        double X1, X2, U1, U2, DL, Strain, Sigma, Ck;
        double QN, XL, C2, Xi, PHIa;
        double *PHI, *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;
        int errno = 0;
```

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	R		I	要素断面積
QV	R		I	体積当たり発熱量 $\dot{Q}$
COND	R		I	熱伝導率

変数表(2/2)

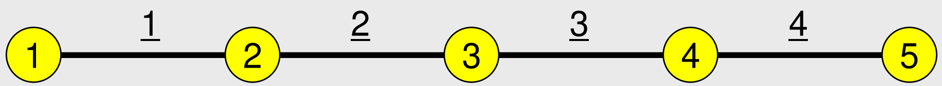
変数名	種別	サイズ	I/O	内　容
X	R	N	O	節点座標
PHI	R	N	O	節点温度
Rhs	R	N	O	右辺ベクトル
Diag	R	N	O	全体マトリクス：対角成分
W	R	[4][N]	O	CG法のwork配列
Amat	R	NPLU	O	全体マトリクス：非零非対角成分
Index	I	N+1	O	全体マトリクス：各行の非零非対角成分数
Item	I	NPLU	O	全体マトリクス：列番号
Icelnod	I	2*NE	O	各要素節点番号
Kmat	R	[2][2]	O	要素マトリクス$[k]$
Emat	R	[2][2]	O	要素マトリクス

プログラム：1d.c（2/6）
初期設定，配列宣言

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

        N= NE + 1;

        PHI  = 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));
```

制御ファイル input.dat

```
4                        NE（要素数）
1.0  1.0  1.0  1.0       Δx（要素長さL）Q,A,COND
100                      反復回数
1.e-8                    CG法の反復打切誤差
```



NE：　　要素数
N：　　節点数（=NE+1）

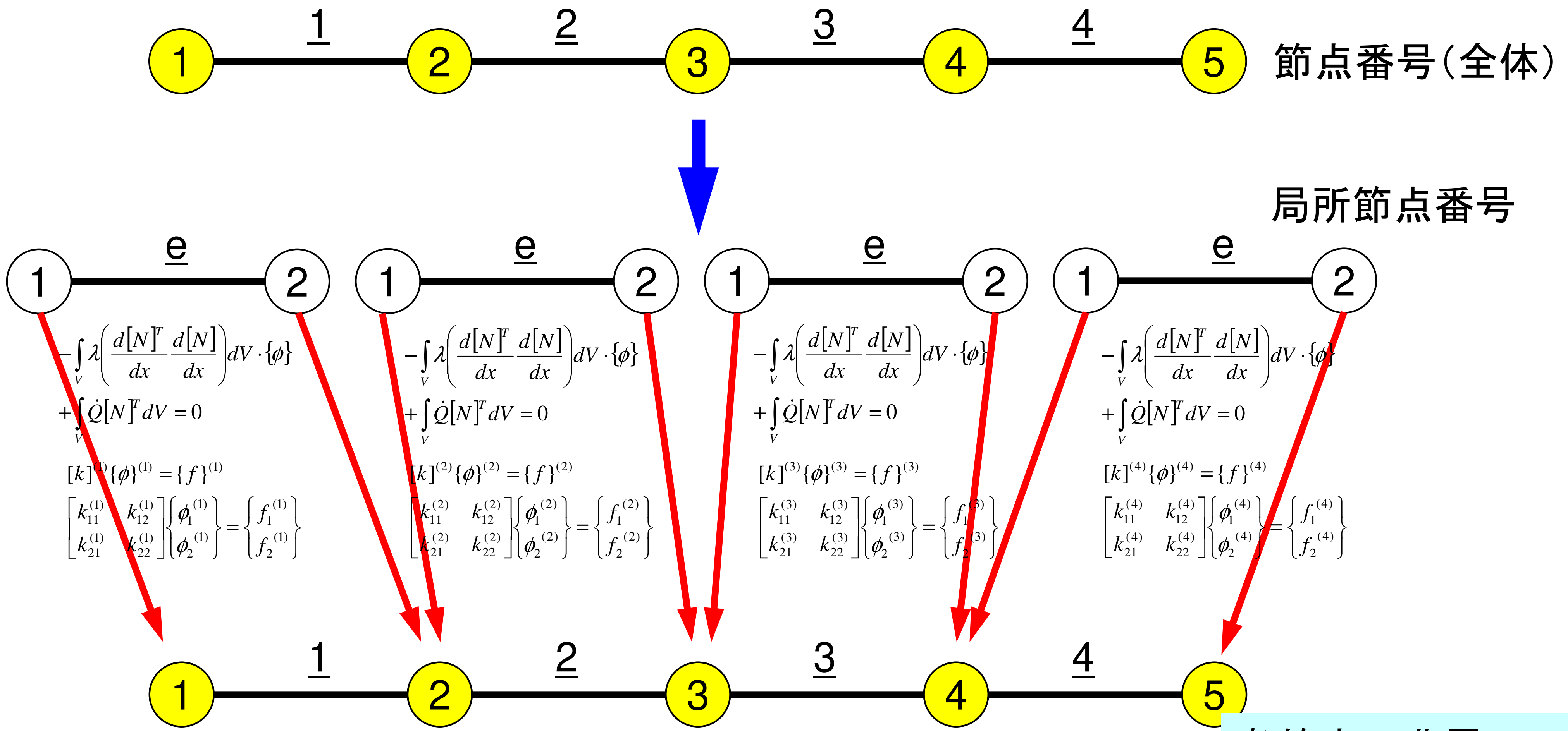
プログラム：1d.c（2/6）

初期設定，配列宣言

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

        N= NE + 1;

        PHI  = 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));
```

Amat： 非零非対角成分
Item： 対応する列番号

要素処理と全体処理

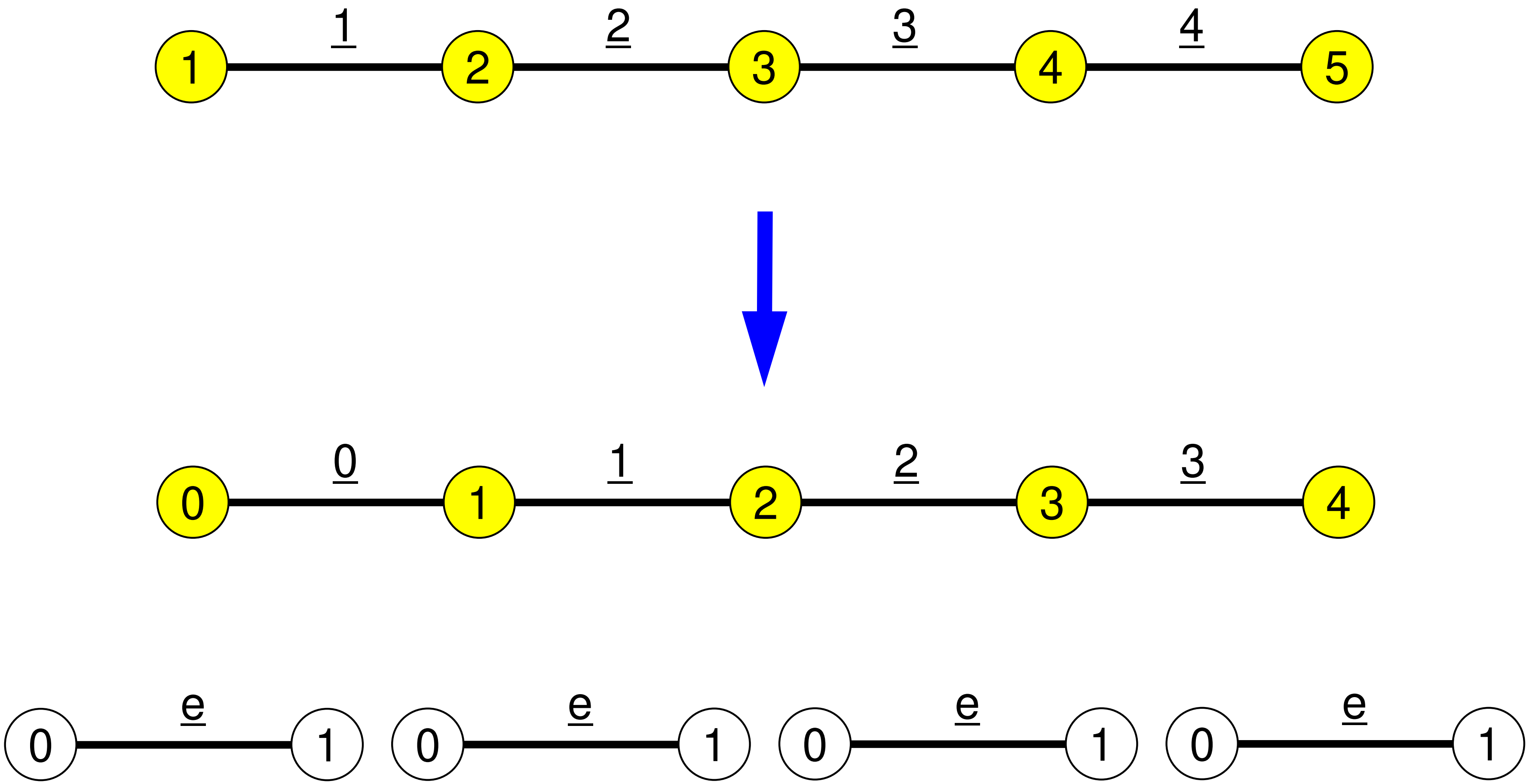


$$[K]\{\Phi\} = \{F\}$$

$$\begin{bmatrix} D_1 & AU_{11} & & & \\ AL_{21} & D_2 & AU_{21} & & \\ & AL_{31} & D_3 & AU_{31} & \\ & & AL_{41} & D_4 & AU_{41} \\ & & & AL_{51} & D_5 \end{bmatrix} \begin{Bmatrix} \Phi_1 \\ \Phi_2 \\ \Phi_3 \\ \Phi_4 \\ \Phi_5 \end{Bmatrix} = \begin{Bmatrix} B_1 \\ B_2 \\ B_3 \\ B_4 \\ B_5 \end{Bmatrix}$$

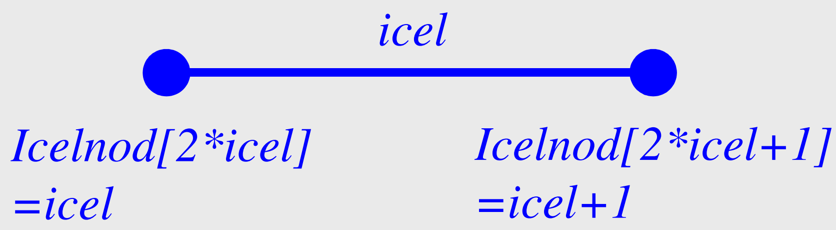
各節点の非零
非対角成分数
は「2」(ただし両端
では「1」)

注意：プログラムの中では節点・要素番号は0からふられている（C言語）



プログラム：1d.c（2/6）
初期設定，配列宣言

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

        N= NE + 1;

        PHI  = 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));
```



Amat： 非零非対角成分
Item： 対応する列番号

各節点の非零非対角成分数は「2」（ただし両端では「1」）

総数：2*(N-2)+1+1 = 2*N-2

プログラム：1d.c（3/6）

配列宣言（続き），初期化

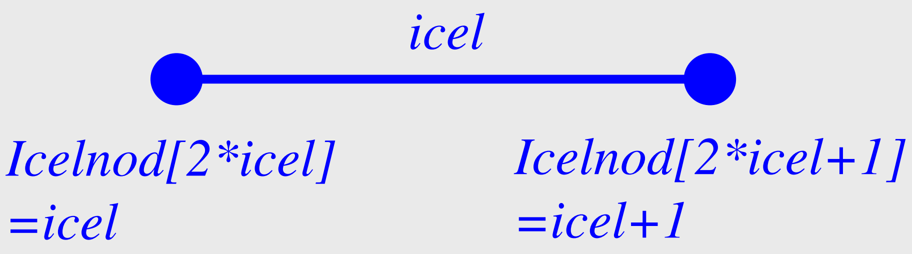
```
W = (double **)malloc(sizeof(double *)*4);
if(W == NULL) {
  fprintf(stderr, "Error: %s¥n", strerror(errno));
  return -1;
}
for(i=0; i<4; i++) {
  W[i] = (double *)malloc(sizeof(double)*N);
  if(W[i] == NULL) {
    fprintf(stderr, "Error: %s¥n", strerror(errno));
    return -1;
  }
}

for(i=0;i<N;i++)  PHI[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;
}

Kmat[0][0]= +1.0;
Kmat[0][1]= -1.0;
Kmat[1][0]= -1.0;
Kmat[1][1]= +1.0;
```

X：　　　　各節点の座標

プログラム：1d.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++)  PHI[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;
       }

       Kmat[0][0]= +1.0;
       Kmat[0][1]= -1.0;
       Kmat[1][0]= -1.0;
       Kmat[1][1]= +1.0;
```



プログラム：1d.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++)  PHI[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;
}

Kmat[0][0]= +1.0;
Kmat[0][1]= -1.0;
Kmat[1][0]= -1.0;
Kmat[1][1]= +1.0;
```

$$[k]^{(e)} = \int_V \lambda \left(\frac{d[N]^T}{dx} \frac{d[N]}{dx} \right) dV = \frac{\lambda A}{L} \begin{bmatrix} +1 & -1 \\ -1 & +1 \end{bmatrix}$$

[Kmat]

プログラム：1d.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++){
          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;
          }
  }
```

各節点の非零非対角成分数は「2」（ただし両端では「1」）

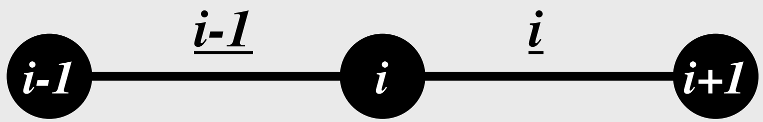
総数：2*(N-2)+1+1 = 2*N-2
Index[N]= 2*N-2 = NPLU

						非対角成分数	Index[0]= 0
0	1.1 ◎	2.4 ①,0	3.2 ④,1			2	Index[1]= 2
1	3.6 ①	4.3 ◎,2	2.5 ③,3	3.7 ⑤,4	9.1 ⑦,5	4	Index[2]= 6
2	5.7 ②	1.5 ④,6	3.1 ⑥,7			2	**Index[3]= 8**
3	9.8 ③	**4.1 ①,8**	**2.5 ④,9**	**2.7 ⑤,10**		3	**Index[4]= 11**
4	11.5 ④	3.1 ◎,11	9.5 ①,12	10.4 ②,13	4.3 ⑥,14	4	Index[5]= 15
5	12.4 ⑤	6.5 ②,15	9.5 ⑥,16			2	Index[6]= 17
6	23.1 ⑥	6.4 ①,17	2.5 ②,18	1.4 ⑤,19	13.1 ⑦,20	4	Index[7]= 21
7	51.3 ⑦	9.5 ①,21	1.3 ②,22	9.6 ③,23	3.1 ⑤,24	4	Index[8]= 25

Index[i]~Index[i+1]-1番目がi行目の非対角成分

プログラム：1d.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++){
          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;
          }
  }
```



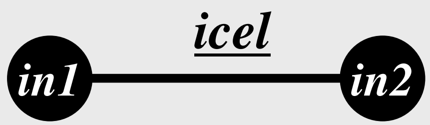
						非対角成分数	Index[0]= 0
0	1.1 ◎	2.4 ①,0	3.2 ④,1			2	Index[1]= 2
1	3.6 ①	4.3 ◎,2	2.5 ③,3	3.7 ⑤,4	9.1 ⑦,5	4	Index[2]= 6
2	5.7 ②	1.5 ④,6	3.1 ⑥,7			2	**Index[3]= 8**
3	9.8 ③	**4.1 ①,8**	**2.5 ④,9**	**2.7 ⑤,10**		**3**	**Index[4]= 11**
4	11.5 ④	3.1 ◎,11	9.5 ①,12	10.4 ②,13	4.3 ⑥,14	4	Index[5]= 15
5	12.4 ⑤	6.5 ②,15	9.5 ⑥,16			2	Index[6]= 17
6	23.1 ⑥	6.4 ①,17	2.5 ②,18	1.4 ⑤,19	13.1 ⑦,20	4	Index[7]= 21
7	51.3 ⑦	9.5 ①,21	1.3 ②,22	9.6 ③,23	3.1 ⑤,24	4	Index[8]= 25

Index[i]~Index[i+1]-1番目がi行目の非対角成分

プログラム：1d.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);

          Ck= Area*COND/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];

          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];

          QN= 0.5*QV*Area*dX;
          Rhs[in1]= Rhs[in1] + QN;
          Rhs[in2]= Rhs[in2] + QN;
        }
```



プログラム：1d.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);

          Ck= Area*COND/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];

          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];

          QN= 0.5*QV*Area*dX;
          Rhs[in1]= Rhs[in1] + QN;
          Rhs[in2]= Rhs[in2] + QN;
        }
```

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

プログラム：1d.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);

        Ck= Area*COND/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];

        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];

        QN= 0.5*QV*Area*dX;
        Rhs[in1]= Rhs[in1] + QN;
        Rhs[in2]= Rhs[in2] + QN;
      }
```

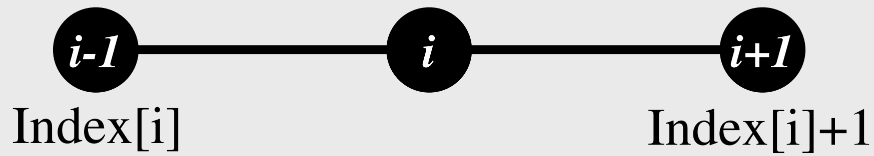
$$[Emat]=[k]^{(e)}=\frac{EA}{L}\begin{bmatrix}+1 & -1\\ -1 & +1\end{bmatrix}$$

プログラム：1d.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);

      Ck= Area*COND/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];

      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];

      QN= 0.5*QV*Area*dX;
      Rhs[in1]= Rhs[in1] + QN;
      Rhs[in2]= Rhs[in2] + QN;
    }
```

「i」行の非対角成分：
Index[i], Index[i]+1



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

k1

k2

通常の要素：k1

in1の非対角成分としてのin2

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

        Ck= Area*COND/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];

        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];

        QN= 0.5*QV*Area*dX;
        Rhs[in1]= Rhs[in1] + QN;
        Rhs[in2]= Rhs[in2] + QN;
      }
```

「i」行の非対角成分：
Index[i], Index[i]+1

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

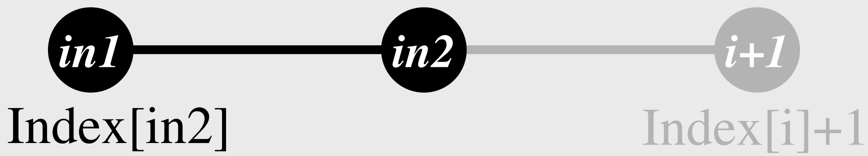
k1

通常の要素：k2

in2の非対角成分としてのin1

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

        Ck= Area*COND/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];

        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];

        QN= 0.5*QV*Area*dX;
        Rhs[in1]= Rhs[in1] + QN;
        Rhs[in2]= Rhs[in2] + QN;
      }
```

「i」行の非対角成分：
Index[i]，Index[i]+1



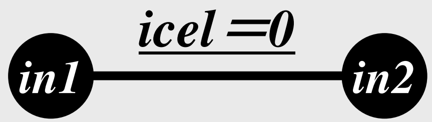
$$[Emat]=[k]^{(e)}=\frac{\lambda A}{L}\begin{bmatrix}+1 & -1\\ -1 & +1\end{bmatrix}$$

k2

0番要素(左端):k1

in1の非対角成分としてのin2

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

        Ck= Area*COND/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];

        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];

        QN= 0.5*QV*Area*dX;
        Rhs[in1]= Rhs[in1] + QN;
        Rhs[in2]= Rhs[in2] + QN;
      }
```



「i」行の非対角成分：Index[i]のみ

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

k1

プログラム：1d.c（5/6）

体積発熱項，右辺

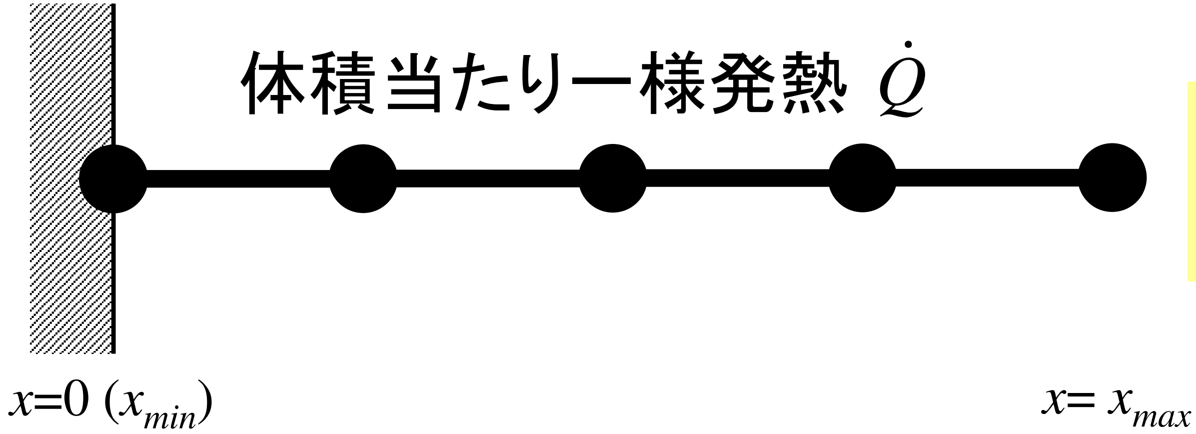
```
/*
// +-------------------+
// | MATRIX assemble |
// +-------------------+
*/
      for(icel=0;icel<NE;icel++){
        in1= Icelnod[2*icel];
        in2= Icelnod[2*icel+1];
        X1 = X[in1];
        X2 = X[in2];
        DL = fabs(X2-X1);

        Ck= Area*COND/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];

        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];

        QN= 0.5*QV*Area*dX;
        Rhs[in1]= Rhs[in1] + QN;
        Rhs[in2]= Rhs[in2] + QN;
      }
```

$$\int_V \dot{Q}[N]^T dV = \dot{Q}A\int_0^L \begin{bmatrix} 1-x/L \\ x/L \end{bmatrix} dx = \frac{\dot{Q}AL}{2}\begin{Bmatrix} 1 \\ 1 \end{Bmatrix}$$

プログラム：1d.c（6/6）
第一種境界条件@x=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;
          }}
```

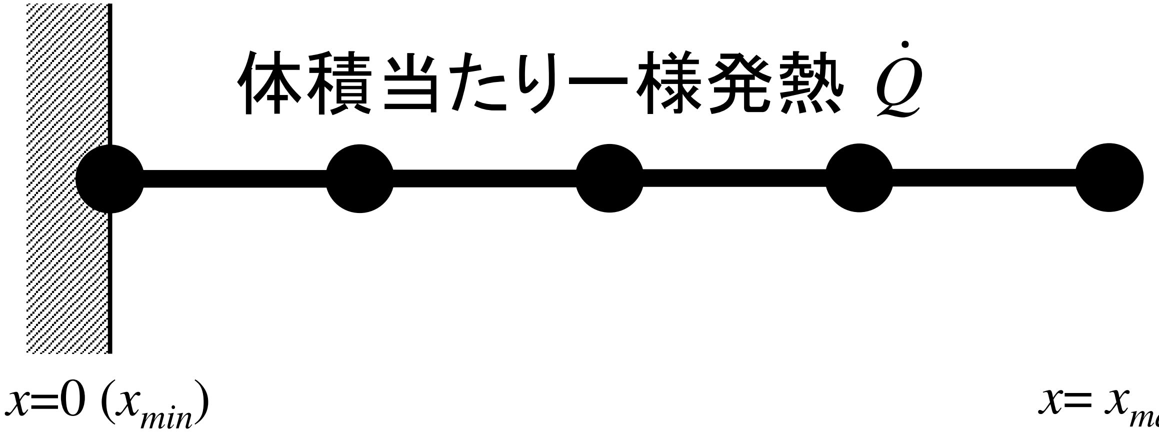
対象とする問題：一次元熱伝導問題



$$\frac{\partial}{\partial x}\left(\lambda\frac{\partial T}{\partial x}\right)+\dot{Q}=0$$

- 一様な：断面積A，熱伝導率λ
- 体積当たり一様発熱（時間当たり）〔QL^{-3}T^{-1}〕$\dot{Q}$
- 境界条件
  - x=0　：T= 0　（固定）
  - x=x_{max}：$\dfrac{\partial T}{\partial x}=0$（断熱）

x=0で成立する方程式 $T_0=0$



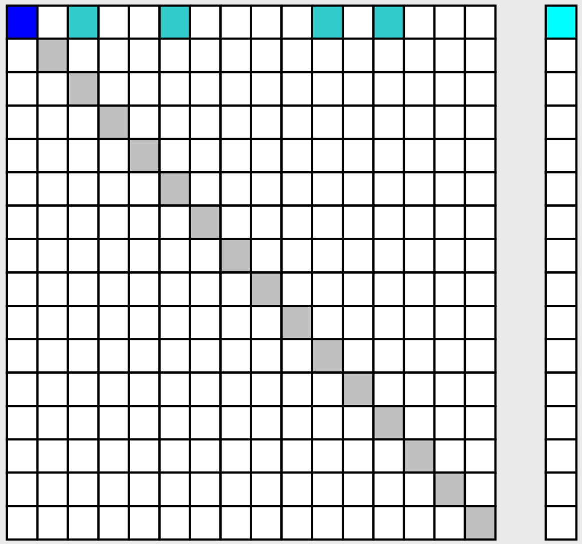
$$\frac{\partial}{\partial x}\left(\lambda \frac{\partial T}{\partial x}\right) + \dot{Q} = 0$$

- 一様な：断面積A，熱伝導率λ
- 体積当たり一様発熱（時間当たり）〔QL^{-3}T^{-1}〕$\dot{Q}$
- 境界条件
  - x=0　：T= 0　（固定）
  - x=x_{max} ：$\dfrac{\partial T}{\partial x} = 0$（断熱）

プログラム：1d.c（6/6）
第一種境界条件@x=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;
          }}
```

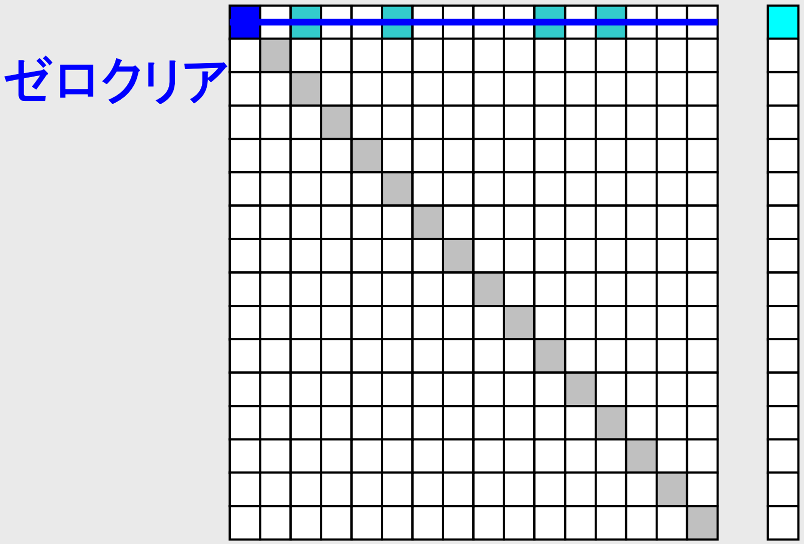
$T_0=0$
対角成分=1，右辺=0，非対角成分=0



プログラム：1d.c（6/6）
第一種境界条件@x=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;
          }}
```

T_0=0
対角成分=1，右辺=0，非対角成分=0

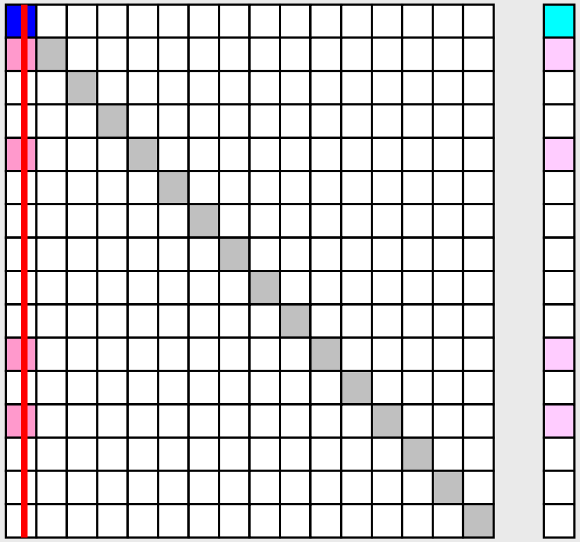


プログラム：1d.c（6/6）
第一種境界条件@x=0

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

$T_0=0$
対角成分=1，右辺=0，非対角成分=0

消去，ゼロクリア



行列の対称性を保つため，第一種境界条件を適用している節点に対応する「列」を，右辺に移項して消去する（今の場合は非対角成分を0にするだけで良い）

第一種境界条件がT≠0の場合

行列の対称性を保つため，第一種境界条件を適用している節点に対応する「列」を，右辺に移項して消去する

$$Diag_j \phi_j + \sum_{k=Index[j]}^{Index[j+1]-1} Amat_k \, \phi_{Item[k]} = Rhs_j$$

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

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

          for(j=1;i<N;i++){
            for (k=Index[j];k<Index[j+1];k++){
              if(Item[k]==0){
                Rhs [j]= Rhs[j] - Amat[k]*PHImin;
                AMat[k]= 0.0;
            }}
```

第一種境界条件がT≠0の場合

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

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

          for(j=1;i<N;i++){
            for (k=Index[j];k<Index[j+1];k++){
              if(Item[k]==0){
                Rhs [j]= Rhs[j] - Amat[k]*PHImin;
                AMat[k]= 0.0;
            }}
```

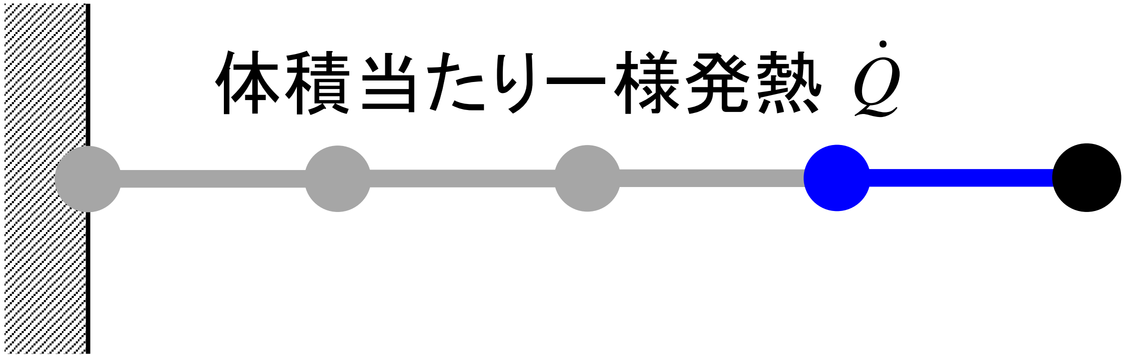
行列の対称性を保つため，第一種境界条件を適用している節点に対応する「列」を，右辺に移項して消去する

$$Diag_j \phi_j + \sum_{k=Index[j],k\neq k_s}^{Index[j+1]-1} Amat_k \, \phi_{Item[k]}$$

$$= Rhs_j - Amat_{k_s} \phi_{Item[k_s]}$$

$$= Rhs_j - Amat_{k_s} \phi_{\min} \qquad where \; Item[k_s] = 0$$

第二種境界条件（断熱）



$$\frac{\partial}{\partial x}\left(\lambda\frac{\partial T}{\partial x}\right)+\dot{Q}=0$$

x=0 (x_{min})　　x= x_{max}

$$T=0 @ x=0$$

$$\frac{\partial T}{\partial x}=0 @ x=x_{\max}$$

$$\int_S \overline{q}[N]^T dS=\overline{q}A\Big|_{x=L}=\overline{q}A\begin{Bmatrix}0\\1\end{Bmatrix},\quad \overline{q}=-\lambda\frac{dT}{dx}$$

表面熱流束

$$\frac{\partial T}{\partial x}=0 @ x=x_{\max}$$

断熱境界条件が成立するため，$\overline{q}=0$
従ってこの項の寄与は無い。
断熱境界条件は何もしなくても成立
→自然境界条件

前処理付き共役勾配法
Preconditioned Conjugate Gradient Method (CG)

```
Compute r(0)= b-[A]x(0)
for i= 1, 2, …
    solve [M]z(i-1)= r(i-1)
    ρi-1= r(i-1) z(i-1)
    if i=1
      p(1)= z(0)
     else
      βi-1= ρi-1/ρi-2
      p(i)= z(i-1) + βi-1 p(i-1)
    endif
    q(i)= [A]p(i)
    αi = ρi-1/p(i)q(i)
    x(i)= x(i-1) + αip(i)
    r(i)= r(i-1) - αiq(i)
    check convergence |r|
end
```

前処理：対角スケーリング

対角スケーリング，点ヤコビ前処理

- 前処理行列として，もとの行列の対角成分のみを取り出した行列を前処理行列 [M] とする。
  - 対角スケーリング，点ヤコビ（point-Jacobi）前処理

$$
[M]=\begin{bmatrix}
D_1 & 0 & \cdots & 0 & 0 \\
0 & D_2 & & 0 & 0 \\
\cdots & & \cdots & & \cdots \\
0 & 0 & & D_{N-1} & 0 \\
0 & 0 & \cdots & 0 & D_N
\end{bmatrix}
$$

- **solve [M]z$^{(i-1)}$= r$^{(i-1)}$**という場合に逆行列を簡単に求めることができる。

CGソルバー(1/6)

```
/*
// +----------------+
// | CG iterations |
// +----------------+
*/
        R = 0;
        Z = 1;
        Q = 1;
        P = 2;
        DD= 3;

        for(i=0;i<N;i++){
                W[DD][i]= 1.0 / Diag[i];
        }
```

W[0][i]= W[R][i] ⇒ {r}
W[1][i]= W[Z][i] ⇒ {z}
W[1][i]= W[Q][i] ⇒ {q}
W[2][i]= W[P][i] ⇒ {p}
W[3][i]= W[DD][i] ⇒ 1/{D}

Compute $r^{(0)}= b-[A]x^{(0)}$
<u>for</u> i= 1, 2, …
  solve $[M]z^{(i-1)}= r^{(i-1)}$
  $\rho_{i-1}= r^{(i-1)}\ z^{(i-1)}$
  <u>if</u> i=1
    $p^{(1)}= z^{(0)}$
  <u>else</u>
    $\beta_{i-1}= \rho_{i-1}/\rho_{i-2}$
    $p^{(i)}= z^{(i-1)} + \beta_{i-1}\ p^{(i-1)}$
  <u>endif</u>
  $q^{(i)}= [A]p^{(i)}$
  $\alpha_i = \rho_{i-1}/p^{(i)}q^{(i)}$
  $x^{(i)}= x^{(i-1)} + \alpha_i p^{(i)}$
  $r^{(i)}= r^{(i-1)} - \alpha_i q^{(i)}$
  check convergence $|r|$
<u>end</u>

CGソルバー(1/6)

```
/*
// +----------------+
// | CG iterations |
// +----------------+
*/
          R = 0;
          Z = 1;
          Q = 1;
          P = 2;
          DD= 3;

          for(i=0;i<N;i++){
                    W[DD][i]= 1.0 / Diag[i];
          }
```

対角成分の逆数(前処理用)
その都度，除算をすると効率が悪いため，予め配列に格納。
嘗ては除算と加減乗算は10:1と言われていたが最近はそれほどでもない。

```
W[0][i]= W[R][i]  ⇒ {r}
W[1][i]= W[Z][i]  ⇒ {z}
W[1][i]= W[Q][i]  ⇒ {q}
W[2][i]= W[P][i]  ⇒ {p}
W[3][i]= W[DD][i] ⇒ 1/{D}
```

CGソルバー(2/6)

```
/*
//-- {r0}= {b} - [A]{xini} |
*/
        for(i=0;i<N;i++){
            W[R][i] = Diag[i]*PHI[i];
            for(j=Index[i];j<Index[i+1];j++){
                W[R][i] += AMat[j]*PHI[Item[j]];
            }
        }

        BNorm2 = 0.0;
        for(i=0;i<N;i++){
            BNorm2 += Rhs[i] * Rhs[i];
            W[R][i] = Rhs[i] - W[R][i];
        }
```

BNRM2=|b|²
あとで収束判定に使用

Compute $r^{(0)}= b-[A]x^{(0)}$

<u>for</u> i= 1, 2, …
  solve $[M]z^{(i-1)}= r^{(i-1)}$
  $\rho_{i-1}= r^{(i-1)}\ z^{(i-1)}$
  <u>if</u> i=1
    $p^{(1)}= z^{(0)}$
   <u>else</u>
    $\beta_{i-1}= \rho_{i-1}/\rho_{i-2}$
    $p^{(i)}= z^{(i-1)} + \beta_{i-1}\ p^{(i-1)}$
  <u>endif</u>
  $q^{(i)}= [A]p^{(i)}$
  $\alpha_i = \rho_{i-1}/p^{(i)}q^{(i)}$
  $x^{(i)}= x^{(i-1)} + \alpha_i p^{(i)}$
  $r^{(i)}= r^{(i-1)} - \alpha_i q^{(i)}$
  check convergence |r|
<u>end</u>

CGソルバー（3/6）

```
 for(iter=1;iter<=IterMax;iter++){

/*
//-- {z}= [Minv]{r}
*/
    for(i=0;i<N;i++){
        W[Z][i] = W[DD][i] * W[R][i];
    }

/*
//-- RHO= {r}{z}
*/
    Rho= 0.0;
    for(i=0;i<N;i++){
        Rho += W[R][i] * W[Z][i];
    }
```

Compute $r^{(0)}= b-[A]x^{(0)}$
<u>for</u> i= 1, 2, …
 solve $[M]z^{(i-1)}= r^{(i-1)}$
 $\rho_{i-1}= r^{(i-1)}\ z^{(i-1)}$
 <u>if</u> i=1
  $p^{(1)}= z^{(0)}$
 <u>else</u>
  $\beta_{i-1}= \rho_{i-1}/\rho_{i-2}$
  $p^{(i)}= z^{(i-1)} + \beta_{i-1}\ p^{(i-1)}$
 <u>endif</u>
 $q^{(i)}= [A]p^{(i)}$
 $\alpha_i = \rho_{i-1}/p^{(i)}q^{(i)}$
 $x^{(i)}= x^{(i-1)} + \alpha_i p^{(i)}$
 $r^{(i)}= r^{(i-1)} - \alpha_i q^{(i)}$
 check convergence $|r|$
<u>end</u>

CGソルバー（4/6）

```
/*
//-- {p} = {z} if       ITER=1
//    BETA= RHO / RHO1   otherwise
*/
  if(iter == 1){
     for(i=0;i<N;i++){
         W[P][i] = W[Z][i];
     }
  }else{
         Beta = Rho / Rho1;
         for(i=0;i<N;i++){
             W[P][i] = W[Z][i] + Beta*W[P][i];
         }
  }

/*
//-- {q}= [A]{p}
*/
  for(i=0;i<N;i++){
      W[Q][i] = Diag[i] * W[P][i];
      for(j=Index[i];j<Index[i+1];j++){
          W[Q][i] += AMat[j]*W[P][Item[j]];
          }
      }
```

Compute $r^{(0)}= b-[A]x^{(0)}$
<u>for</u> i= 1, 2, …
  solve $[M]z^{(i-1)}= r^{(i-1)}$
  $\rho_{i-1}= r^{(i-1)}\ z^{(i-1)}$
  <u>if</u> i=1
    $p^{(1)}= z^{(0)}$
   <u>else</u>
    $\beta_{i-1}= \rho_{i-1}/\rho_{i-2}$
    $p^{(i)}= z^{(i-1)} + \beta_{i-1}\ p^{(i-1)}$
  <u>endif</u>
  $q^{(i)}= [A]p^{(i)}$
  $\alpha_i = \rho_{i-1}/p^{(i)}q^{(i)}$
  $x^{(i)}= x^{(i-1)} + \alpha_i p^{(i)}$
  $r^{(i)}= r^{(i-1)} - \alpha_i q^{(i)}$
  check convergence $|r|$
<u>end</u>

CGソルバー（5/6）

```
/*
//-- ALPHA= RHO / {p}{q}
*/
  C1 = 0.0;
  for(i=0;i<N;i++){
      C1 += W[P][i] * W[Q][i];
  }

  Alpha = Rho / C1;

/*
//-- {x}= {x} + ALPHA*{p}
//    {r}= {r} - ALPHA*{q}
*/
  for(i=0;i<N;i++){
      PHI [i] += Alpha * W[P][i];
      W[R][i] -= Alpha * W[Q][i];
  }
```

Compute $r^{(0)}= b-[A]x^{(0)}$
<u>for</u> i= 1, 2, …
  solve $[M]z^{(i-1)}= r^{(i-1)}$
  $\rho_{i-1}= r^{(i-1)}\ z^{(i-1)}$
  <u>if</u> i=1
    $p^{(1)}= z^{(0)}$
  <u>else</u>
    $\beta_{i-1}= \rho_{i-1}/\rho_{i-2}$
    $p^{(i)}= z^{(i-1)} + \beta_{i-1}\ p^{(i-1)}$
  <u>endif</u>
  $q^{(i)}= [A]p^{(i)}$
  $\boldsymbol{\alpha_i = \rho_{i-1}/p^{(i)}q^{(i)}}$
  $\boldsymbol{x^{(i)}= x^{(i-1)} + \alpha_i p^{(i)}}$
  $\boldsymbol{r^{(i)}= r^{(i-1)} - \alpha_i q^{(i)}}$
  check convergence |r|
<u>end</u>

CGソルバー（6/6）

```
   DNorm2 = 0.0;
   for(i=0;i<N;i++){
       DNorm2 += W[R][i] * W[R][i];
   }
   Resid = sqrt(DNorm2/BNorm2);

   if((iter)%1000 == 0){
       printf("%8d%s%16.6e¥n", iter, "
              iters, RESID=", Resid);
    }
    if(Resid <= Eps){ierr = 0; break;}
    Rho1 = Rho;   ρi-2
}
```

```
Compute r(0)= b-[A]x(0)
for i= 1, 2, …
      solve [M]z(i-1)= r(i-1)
      ρi-1= r(i-1) z(i-1)
      if i=1
        p(1)= z(0)
       else
        βi-1= ρi-1/ρi-2
        p(i)= z(i-1) + βi-1 p(i-1)
      endif
      q(i)= [A]p(i)
      αi  = ρi-1/p(i)q(i)
      x(i)= x(i-1) + αip(i)
      r(i)= r(i-1) - αiq(i)
      check convergence |r|
end
```

$$\mathrm{Resid} = \sqrt{\frac{\mathrm{DNorm2}}{\mathrm{BNorm2}}} = \frac{|r|}{|b|} = \frac{|b - Ax|}{|b|} \leq \mathrm{Eps}$$

$$|r|, |b| : 2 / L2 / Euclidean - norm \quad (\|r\|_2, \|b\|_2)$$

制御ファイル　input.dat

4	NE（要素数）
1.0　1.0　1.0　1.0	Δx（要素長さL），Q，A，λ
100	反復回数
1.e-8	CG法の反復打切誤差　Eps

$$Ax = b \Rightarrow \alpha Ax = \alpha b$$

$$r = b - Ax \Rightarrow R = \alpha b - \alpha Ax = \alpha r$$

有限要素法の処理：プログラム

- 初期化
  - 制御変数読み込み
  - 座標読み込み⇒要素生成（N:節点数，NE：要素数）
  - 配列初期化（全体マトリクス，要素マトリクス）
  - 要素⇒全体マトリクスマッピング（Index，Item）
- マトリクス生成
  - 要素単位の処理（do icel= 1, NE）
    - 要素マトリクス計算
    - 全体マトリクスへの重ね合わせ
  - 境界条件の処理
- 連立一次方程式
  - 共役勾配法（CG）

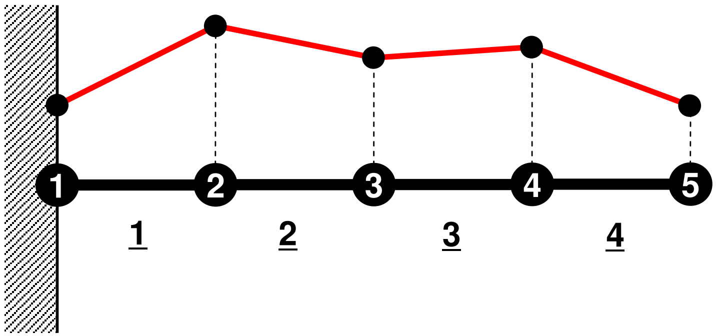
より精度をあげるには？

- 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
```

$$A = A_1 x + A_2 \quad (>0)$$

```
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
```



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

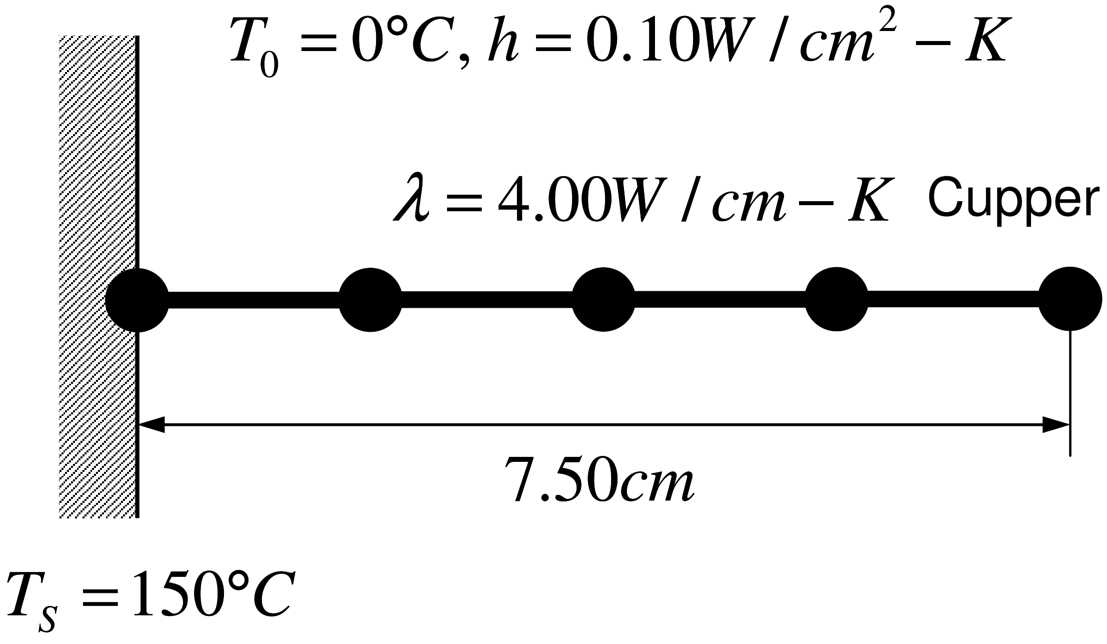
より精度をあげるには？

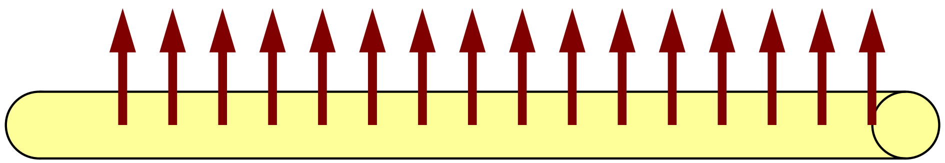
- メッシュを細かくする
- 高次の補間関数（形状関数）を使用する
  - 高次要素
  - 線形要素，一次要素は低次要素と呼ばれる
- n次微分係数の連続性を保証する定式化を適用する
  - C^n連続性

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
- 高次要素の例：
  - 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

Example: 1D Heat Transfer (1/2)





Convective Heat Transfer on Cylindrical Surface

- Temp. Thermal Fins
- Circular Sectional Area, r=1cm
- Boundary Condition
  - x=0 : Fixed Temperature
  - x=7.5 : Insulated
- Convective Heat Transfer on Cylindrical Surface
  - q= h (T-T_0)
  - q : Heat Flux
    - Heat Flow/Unit Surface Area/sec.

Example: 1D Heat Transfer (2/2)

RESULTS (linear interpolation)

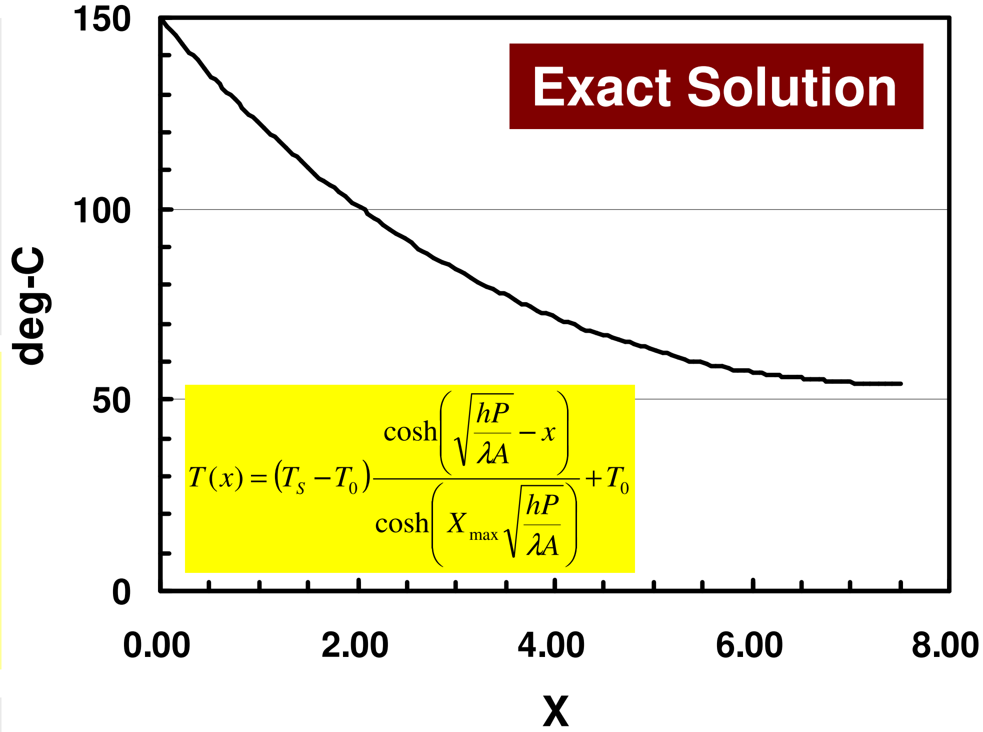
ID	X	FEM.	ANALYTICAL		
1	0.00000	150.00000	150.00000	ERR(%):	0.00000
2	1.87500	102.62226	103.00165	ERR(%):	0.25292
3	3.75000	73.82803	74.37583	ERR(%):	0.36520
4	5.62500	58.40306	59.01653	ERR(%):	0.40898
5	7.50000	53.55410	54.18409	ERR(%):	0.41999

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)

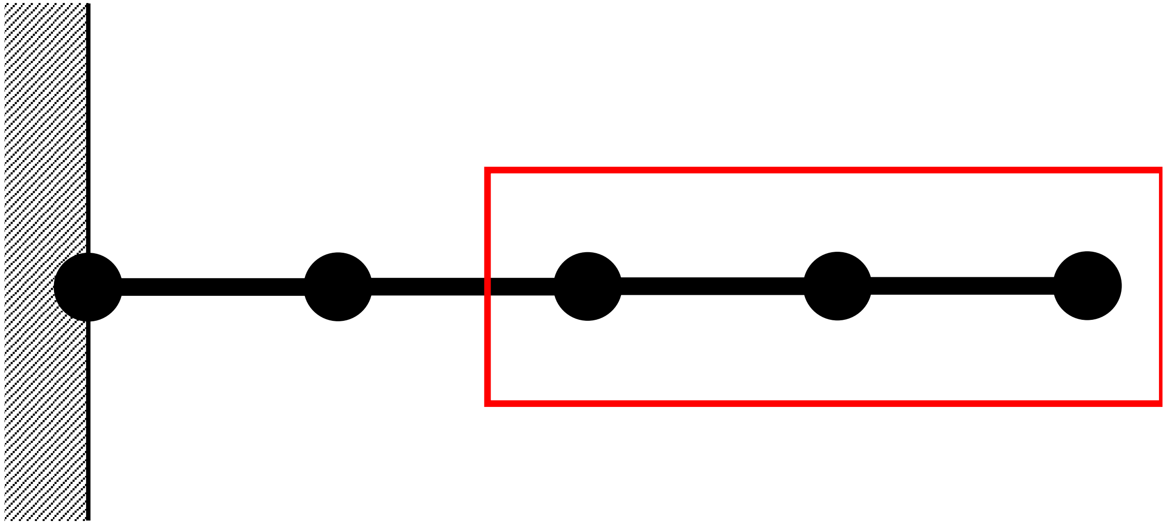
ID	X	FEM.	ANALYTICAL		
1	0.00000	150.00000	150.00000	ERR(%):	0.00000
2	0.93750	123.71561	123.77127	ERR(%):	0.03711
3	1.87500	102.90805	103.00165	ERR(%):	0.06240
4	2.81250	86.65618	86.77507	ERR(%):	0.07926
5	3.75000	74.24055	74.37583	ERR(%):	0.09019
6	4.68750	65.11151	65.25705	ERR(%):	0.09703
7	5.62500	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



Quadratic interpolation provides more accurate solution, especially if X is close to 7.50cm.

1D Quadratic Element (1/2)
一次元二次要素

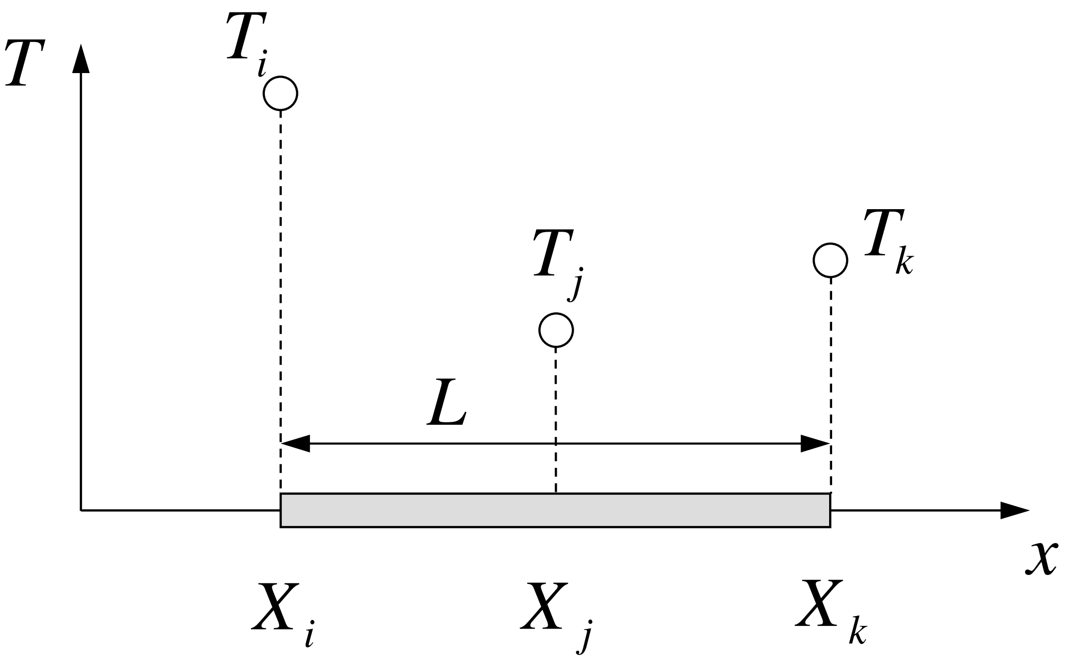
- Length= L
- (i,k): Both Ends
- (j): Intermediate Node
  - ✓ Mid-Point（中間節点）



- Distribution of T in each element:

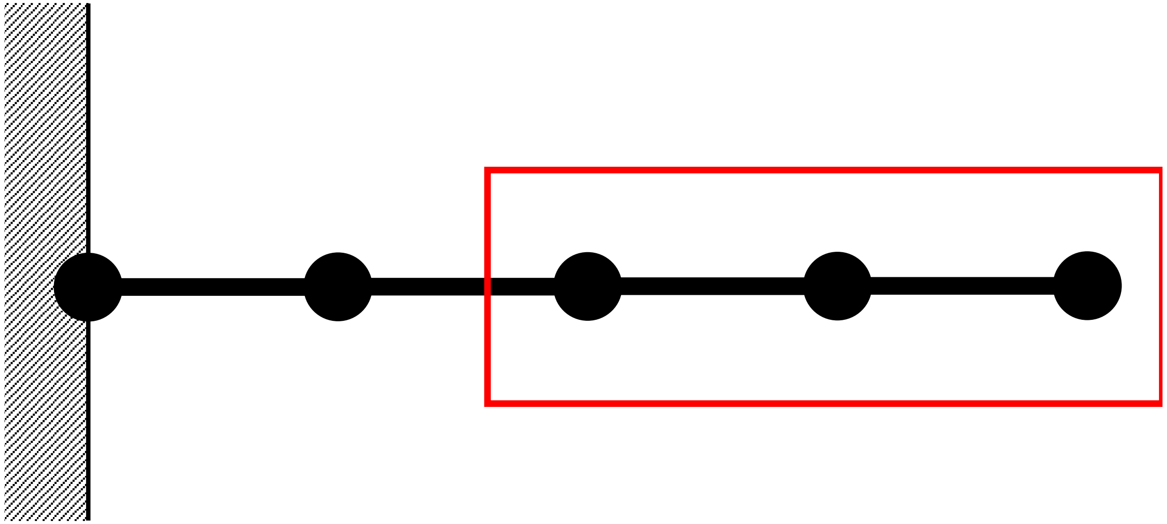
$$T = \alpha_1 + \alpha_2 x + \alpha_3 x^2$$

$$\begin{aligned} T_i &= \alpha_1 + X_i \alpha_2 + X_i^{\ 2} \alpha_3 \\ T_j &= \alpha_1 + X_j \alpha_2 + X_j^{\ 2} \alpha_3 \\ T_k &= \alpha_1 + X_k \alpha_2 + X_k^{\ 2} \alpha_3 \end{aligned}$$



1D Quadratic Element (1/2)
一次元二次要素

- Length= L
- (i,k): Both Ends
- (j): Intermediate Node
  - ✓ Mid-Point（中間節点）

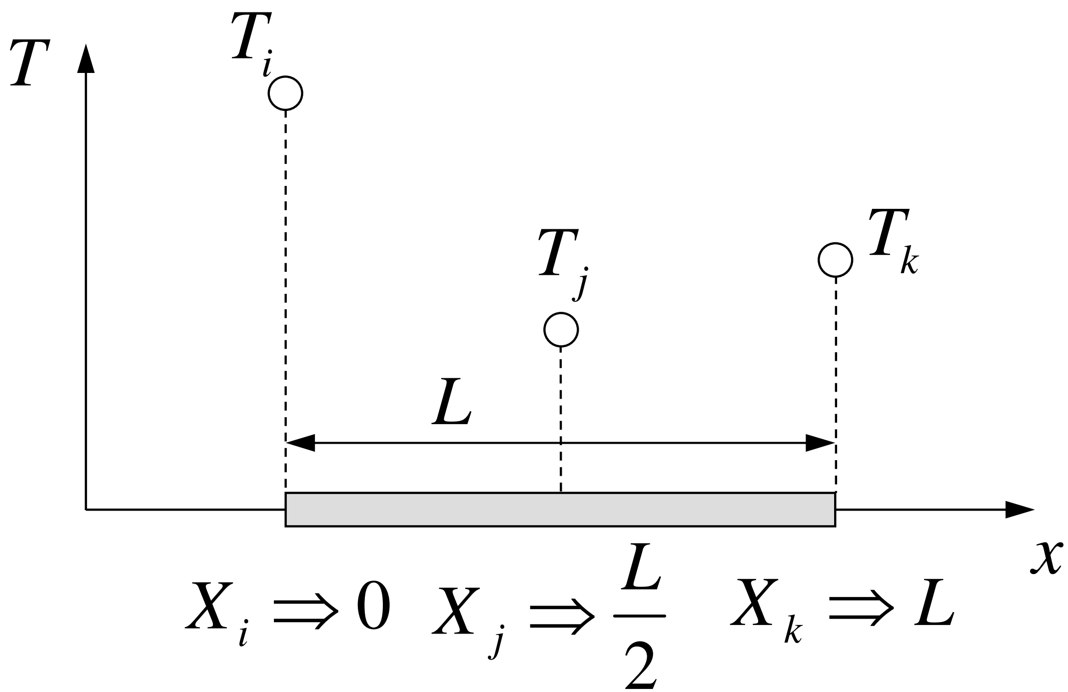


- Distribution of T in each element:

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

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

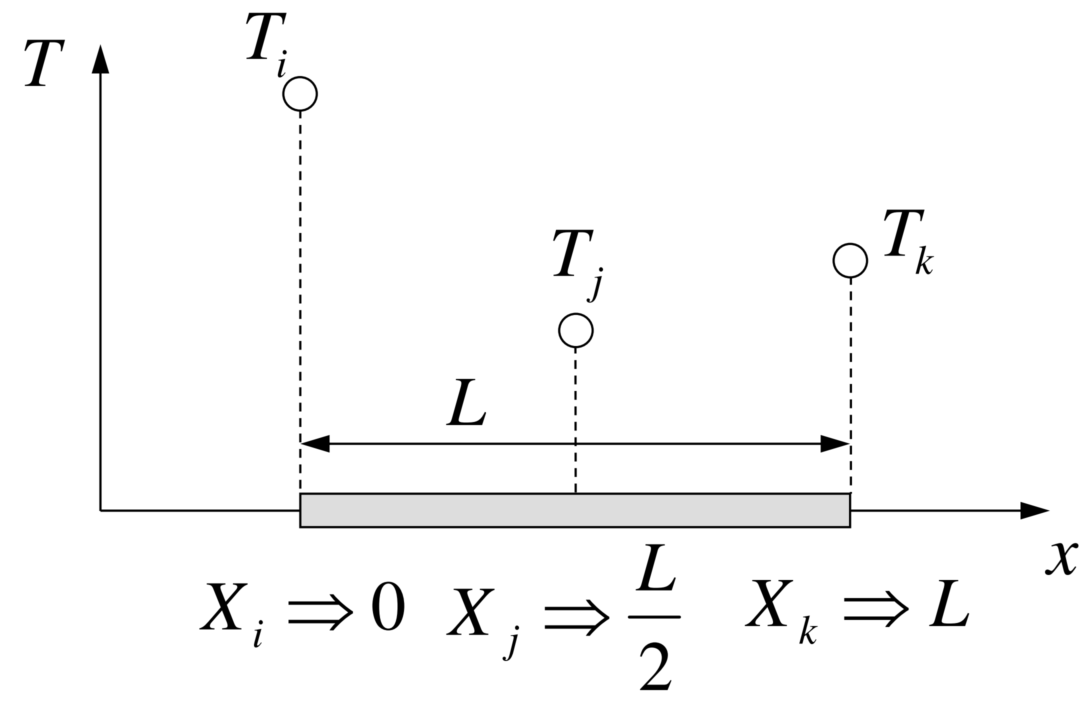
$$u_k = \alpha_1 + L\alpha_2 + L^2\alpha_3$$



1D Quadratic Element (2/2)
一次元二次要素

- Coef's are calculated based on info. at each node:

$$\alpha_1 = T_i,\ \alpha_2 = \frac{4T_i - 3T_j - T_k}{L},$$
$$\alpha_3 = \frac{2}{L^2}\left(T_i - 2T_j + T_k\right)$$

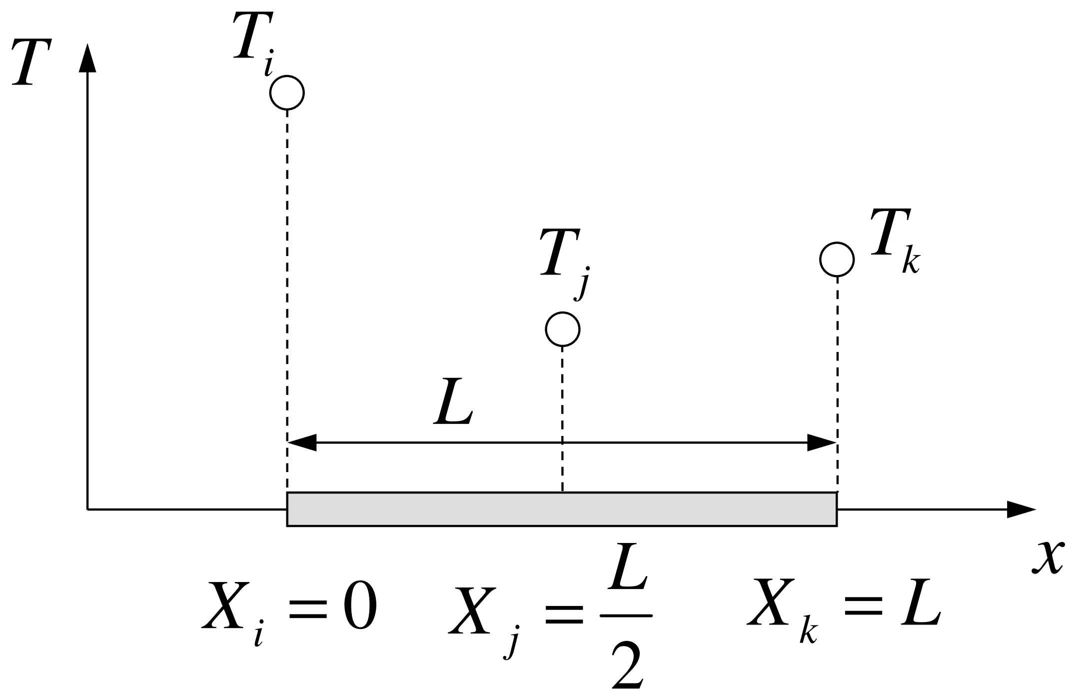


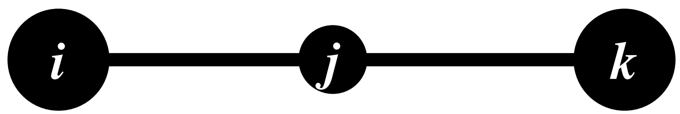
- Shape Functions: N_i, N_j, N_k

$$T = N_i T_i + N_j T_j + N_k T_k$$
$$= \left(1 - \frac{2x}{L}\right)\left(1 - \frac{x}{L}\right)T_i + \left(\frac{4x}{L}\right)\left(1 - \frac{x}{L}\right)T_j + \left(-\frac{x}{L}\right)\left(1 - \frac{2x}{L}\right)T_k$$

1D Quadratic Element

一次元二次要素





Intermediate Node
Mid Point: j

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

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

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

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

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

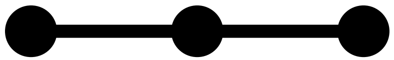
$$\int_V \lambda \left(\frac{d[N]^T}{dx} \frac{d[N]}{dx} \right) dV = \int_0^L \begin{bmatrix} dN_i / dx \\ dN_j / dx \\ dN_k / dx \end{bmatrix} \lambda \left[\frac{dN_i}{dx}, \frac{dN_j}{dx}, \frac{dN_k}{dx} \right] A\, dx$$

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

Integration over Each Element: $\{f\}$

$$\int_V \dot{Q}[N]^T dV = \dot{Q}A\int_0^L \begin{bmatrix} N_i \\ N_j \\ N_k \end{bmatrix} dx = \dot{Q}A\int_0^L \begin{bmatrix} 1-\dfrac{3x}{L}+\dfrac{2x^2}{L^2} \\ \dfrac{4x}{L}-\dfrac{4x^2}{L^2} \\ -\dfrac{x}{L}+\dfrac{2x^2}{L^2} \end{bmatrix} dx = \frac{\dot{Q}AL}{6}\begin{Bmatrix} 1 \\ 4 \\ 1 \end{Bmatrix}$$

1 : 4 : 1



The Ratio was 1:1 in Linear Element

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

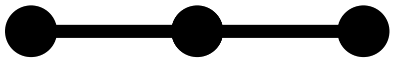
$$\int_V \dot{Q} [N]^T dV = \dot{Q} A \int_0^L \begin{bmatrix} 1 - x/L \\ x/L \end{bmatrix} dx = \frac{\dot{Q} A L}{2} \begin{Bmatrix} 1 \\ 1 \end{Bmatrix}$$

1 : 1

Integration over Each Element: $\{f\}$

$$\int_V \dot{Q}[N]^T dV = \dot{Q}A\int_0^L \begin{bmatrix} N_i \\ N_j \\ N_k \end{bmatrix} dx = \dot{Q}A\int_0^L \begin{bmatrix} 1-\dfrac{3x}{L}+\dfrac{2x^2}{L^2} \\ \dfrac{4x}{L}-\dfrac{4x^2}{L^2} \\ -\dfrac{x}{L}+\dfrac{2x^2}{L^2} \end{bmatrix} dx = \frac{\dot{Q}AL}{6}\begin{Bmatrix} 1 \\ 4 \\ 1 \end{Bmatrix}$$

1 : 4 : 1



Volume
Heat Flux

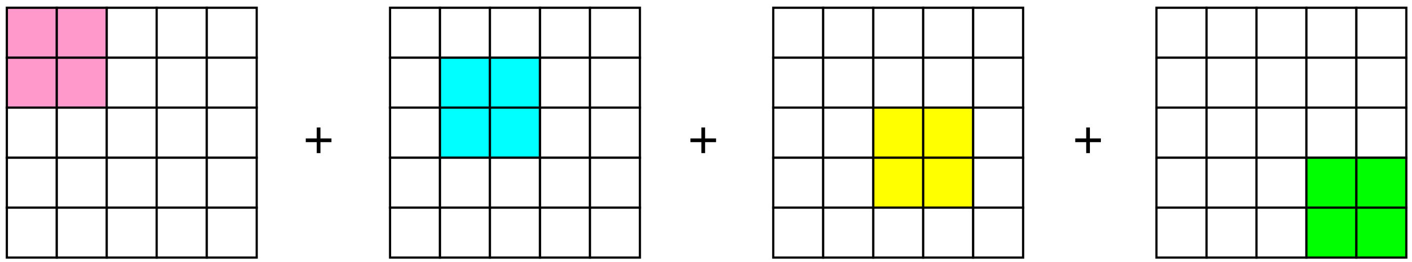
$$\int_S \bar{q}[N]^T dS = \bar{q}A\Big|_{x=L} = \bar{q}A\begin{Bmatrix} 0 \\ 0 \\ 1 \end{Bmatrix}$$

Surface
Heat Flux

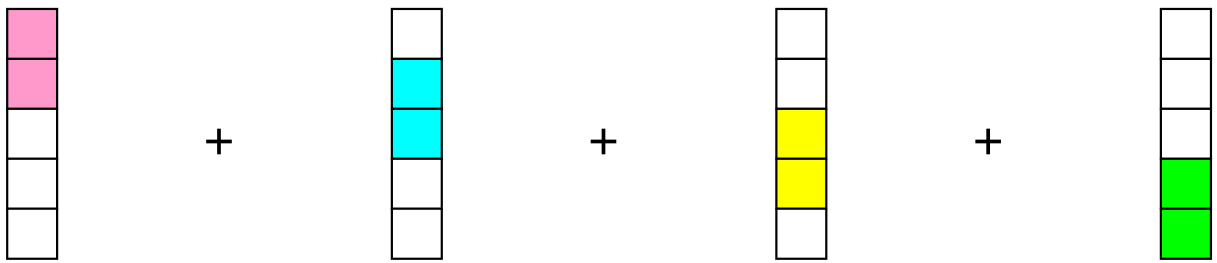
Element Eqn's/Accumulation

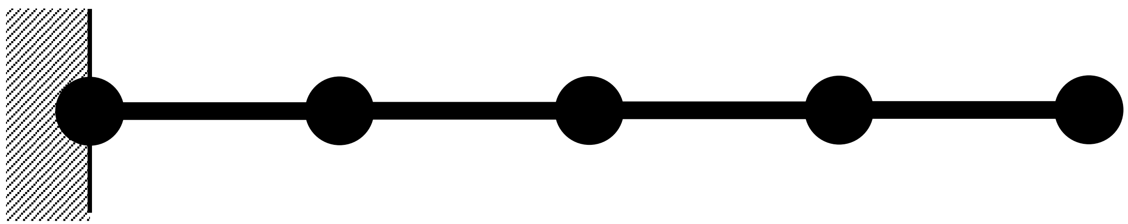
1D Linear Element

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



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

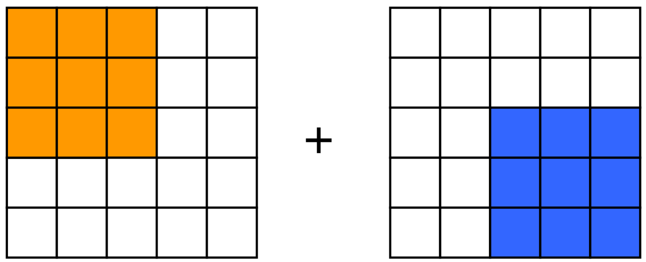




Element Eqn's/Accumulation

1D Quadratic Element, 2 Elements

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



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