1D-FEM in Fortran: Steady State Heat Conduction

Kengo Nakajima
RIKEN R-CCS

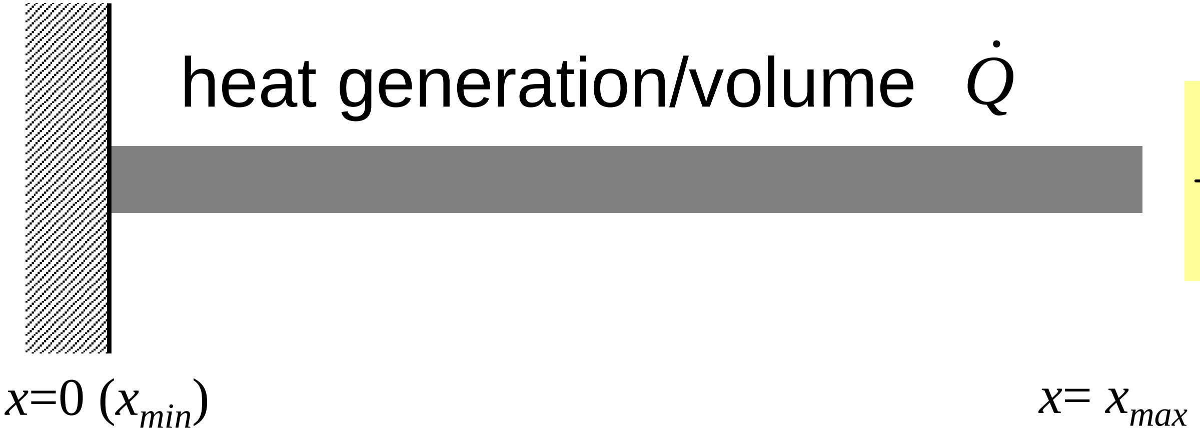
- 1D-code for Steady State Heat Conduction Problems by Galerkin FEM
- Sparse Linear Solver
  - Conjugate Gradient Method
  - Preconditioning
- Storage of Sparse Matrices

- Program

Keywords

- 1D Steady State Heat Conduction Problems
- Galerkin Method
- Linear Element
- Preconditioned Conjugate Gradient Method

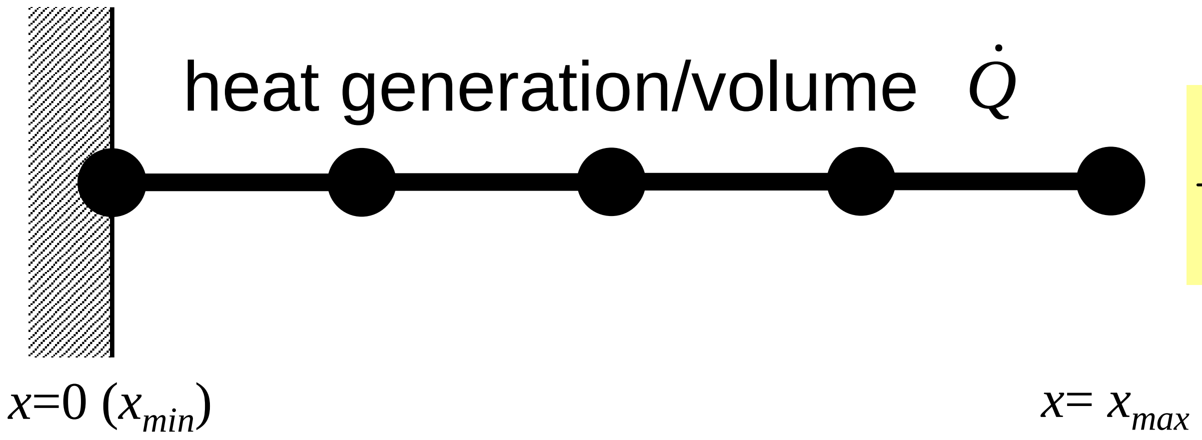
1D Steady State Heat Conduction



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

- Uniform: Sectional Area: A, Thermal Conductivity: λ
- Heat Generation Rate/Volume/Time [$QL^{-3}T^{-1}$] $\dot{Q}$
- Boundary Conditions
  - $x=0$: $T=0$ (Fixed Temperature)
  - $x=x_{max}$: $\frac{\partial T}{\partial x}=0$ (Insulated)

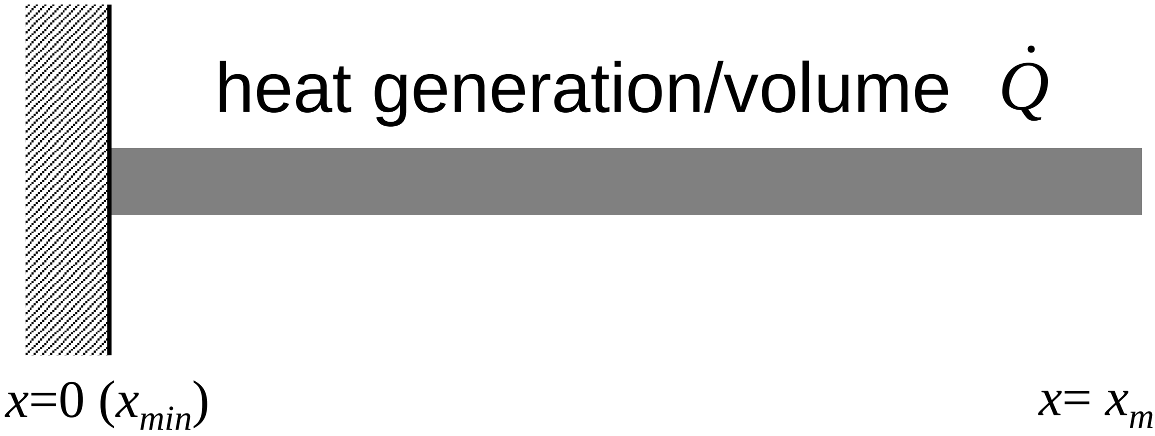
1D Steady State Heat Conduction



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

- Uniform: Sectional Area: A, Thermal Conductivity: λ
- Heat Generation Rate/Volume/Time [$QL^{-3}T^{-1}$] $\dot{Q}$
- Boundary Conditions
  - $x=0$: $T=0$ (Fixed Temperature)
  - $x=x_{max}$: $\frac{\partial T}{\partial x} = 0$ (Insulated)

Analytical Solution



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

$$T = 0 @ x = 0$$

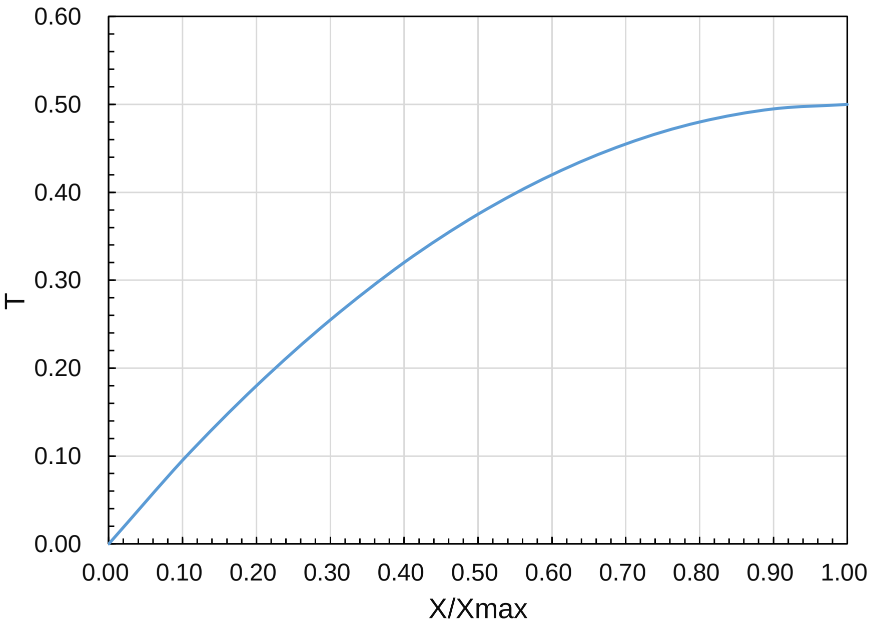
$$\frac{\partial T}{\partial x} = 0 @ x = x_{\max}$$

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

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

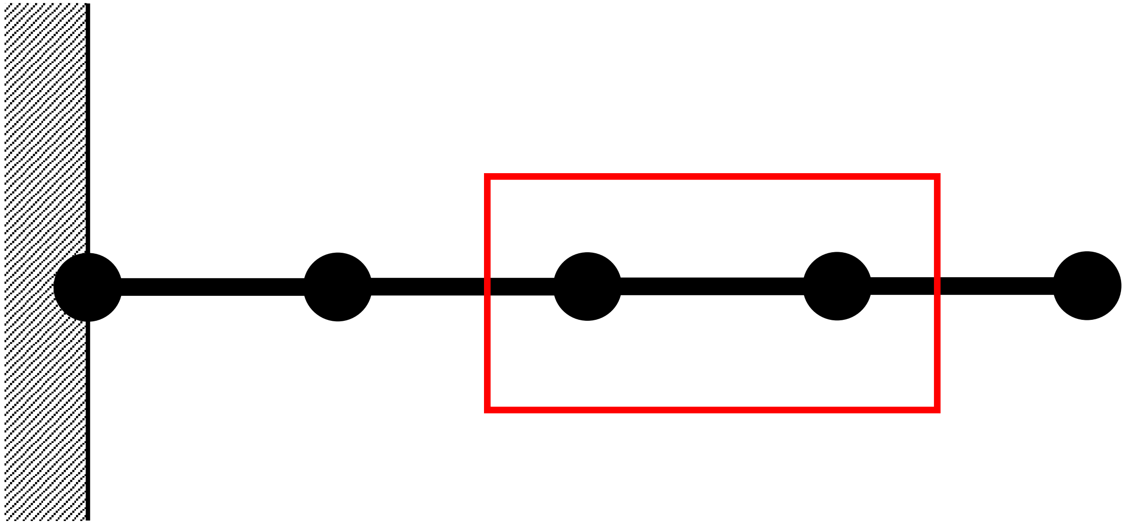
$$\lambda T = -\frac{1}{2}\dot{Q}x^2 + C_1 x + C_2 \Rightarrow C_2 = 0, \quad T = 0 @ x = 0$$

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



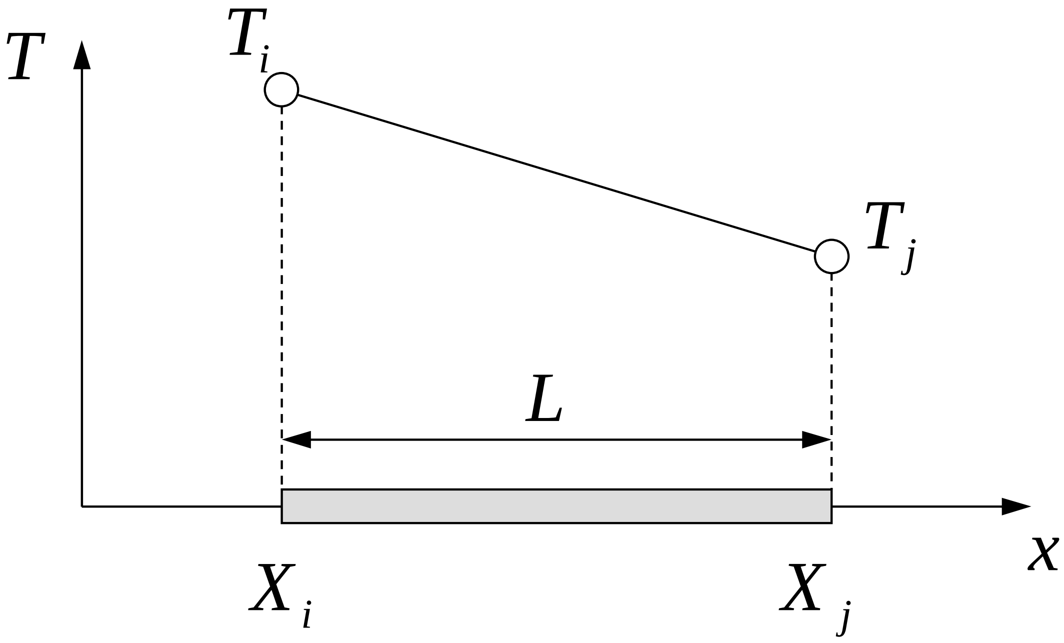
1D Linear Element (1/4)
一次元線形要素

- 1D Linear Element
  - Length= L
    - Node (Vertex)（節点）
    - Element（要素）



  - T_i Temperature at i
  - T_j Temperature at j
  - Temperature T on each element is linear function of x (Piecewise Linear):

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

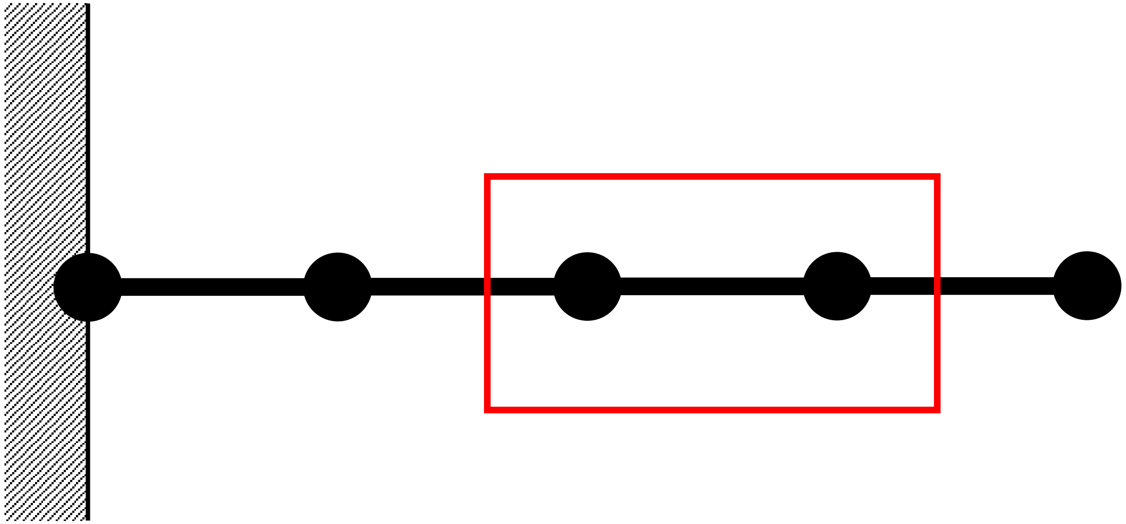


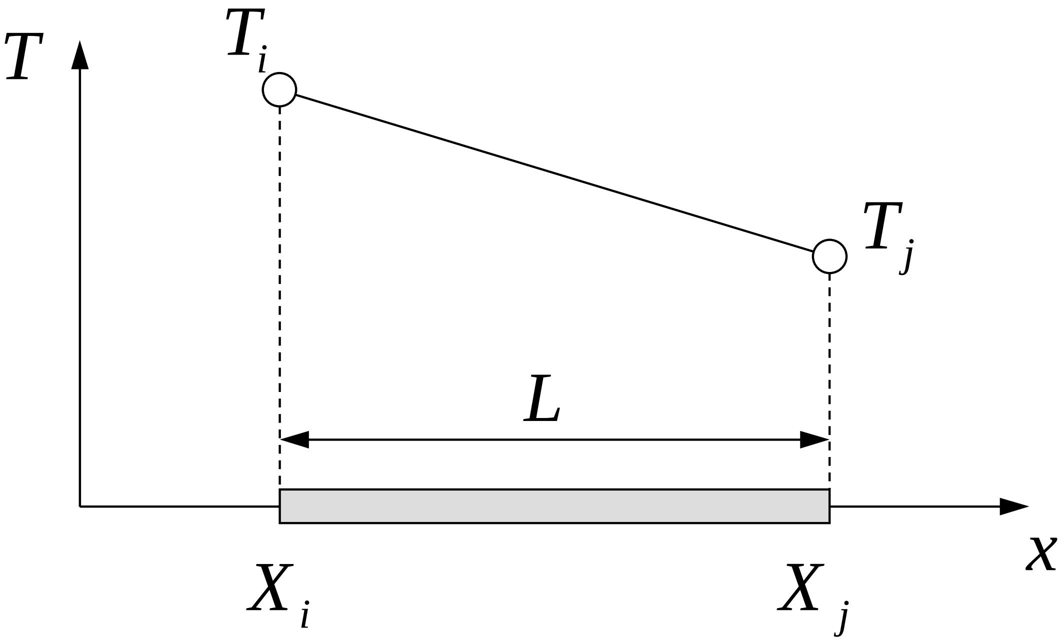
1D Linear Element (1/4)

一次元線形要素

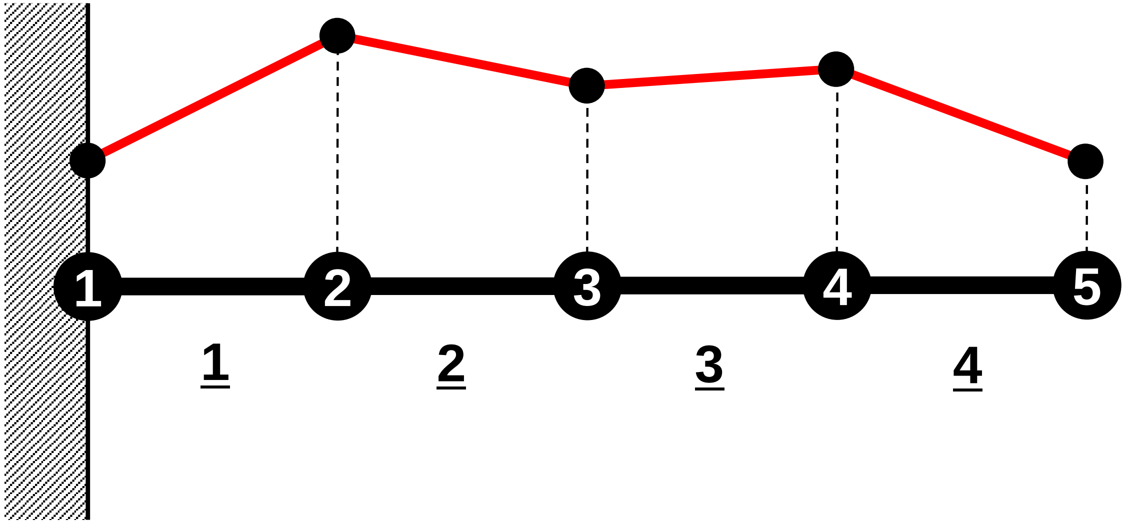
- 1D Linear Element
  - Length= L
    - Node (Vertex)
    - Element
  - T_i Temperature at i
  - T_j Temperature at j
  - Temperature T on each element is linear function of x (Piecewise Linear):

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

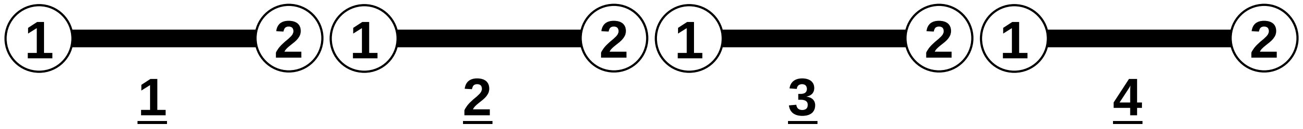




Piecewise Linear



Global Node ID
Element ID



Local Node ID
for each elem.

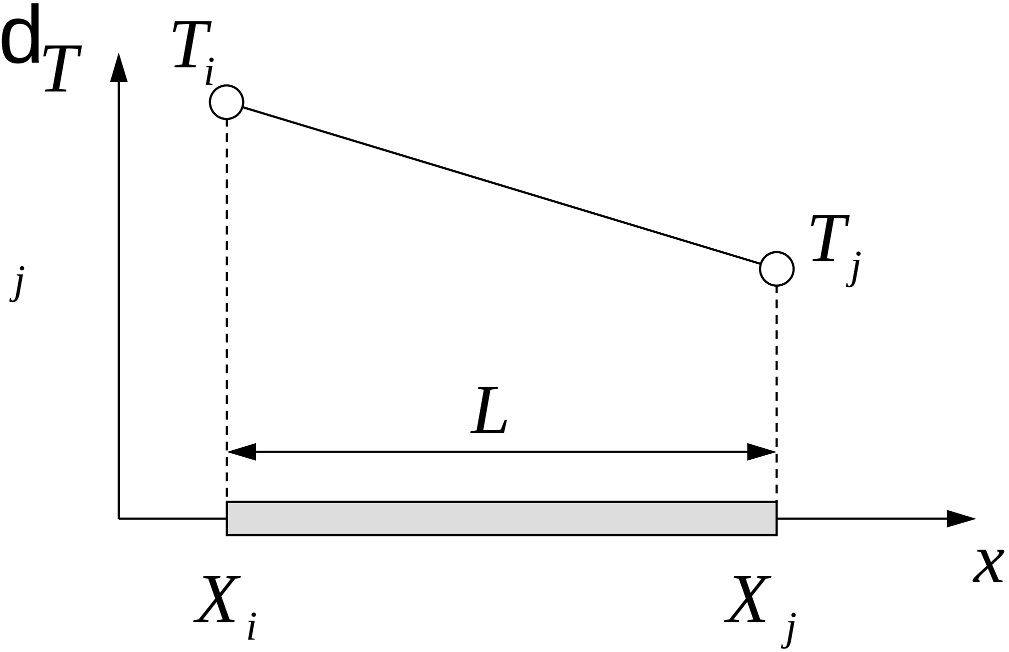
Gradient of temperature is constant in each element (might be discontinuous at each "node")

1D Linear Elem.: Shape Function (2/4)

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

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

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



- Coefficients:

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

- T can be written as follows, according to T_i and T_j:

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

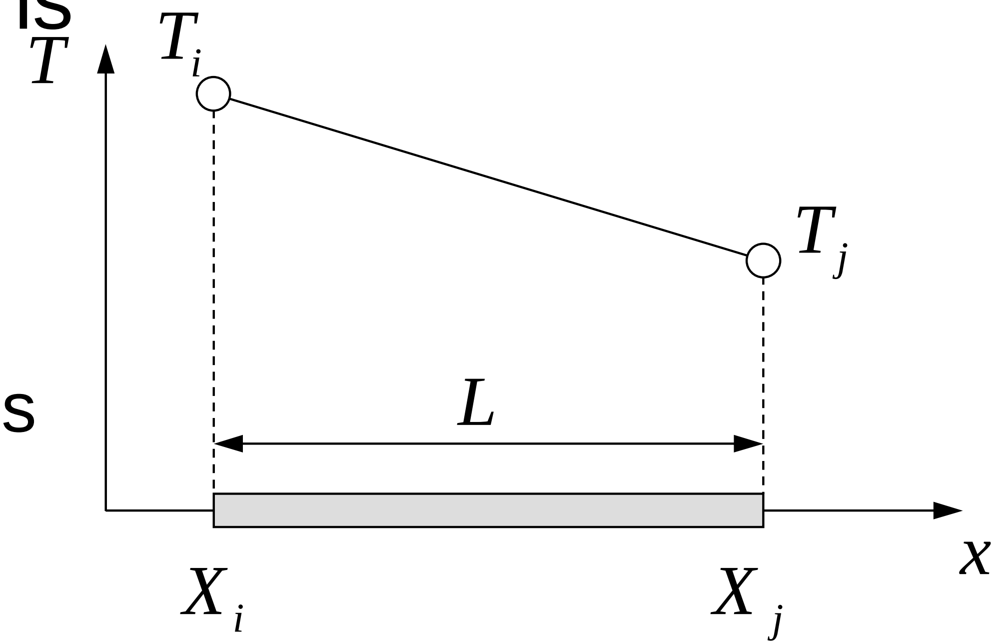
N_i N_j

N_i, N_j

Shape Function（形状関数）or Interpolation Function（内挿関数）, function of x (only)

1D Linear Elem.: Shape Function (3/4)

- Number of Shape Functions = Number of Vertices of Each Element
  - N_i: Function of Position
  - A kind of Test/Trial Functions



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

- Linear combination of shape functions provides temperature "in" each element
  - Coef's (unknows): Temperature at each node

$$T = N_i T_i + N_j T_j \longleftrightarrow$$

$$T_M = \sum_{i=1}^{M} a_i \Psi_i$$

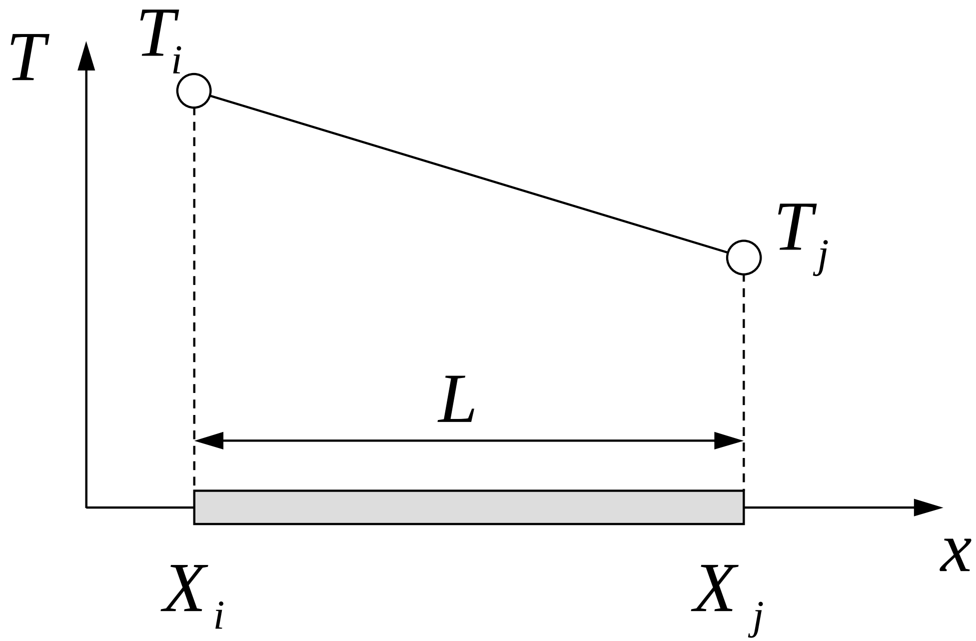
Ψ_i Trial/Test Function (known function of position, defined in domain and at boundary. "Basis" in linear algebra.

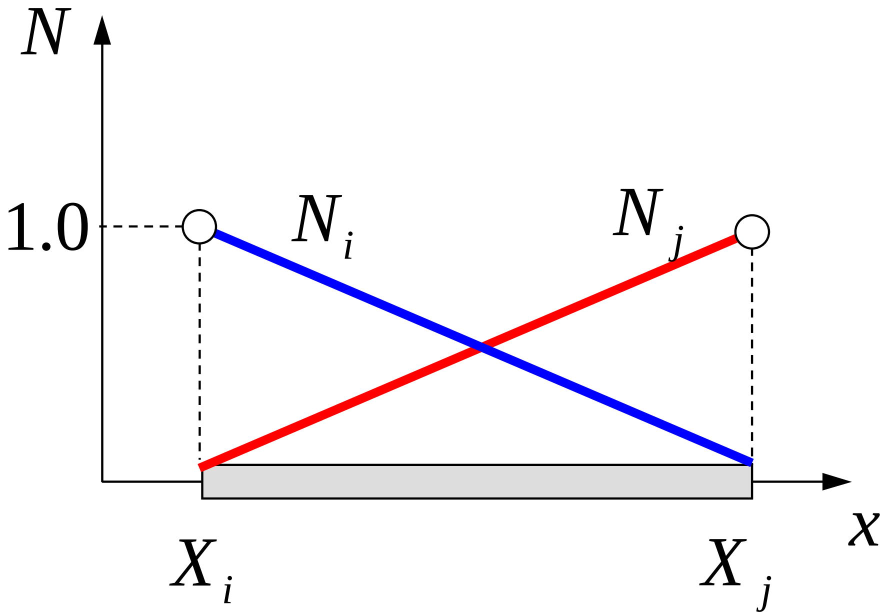
a_i Coefficients (unknown)

1D Linear Elem.: Shape Function (4/4)

- Value of N_i
  - =1 at one of the nodes in element
  - =0 on other nodes

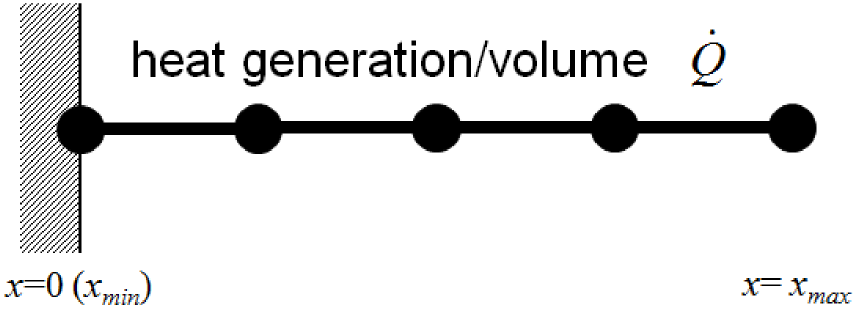
$$N_i = \left(\frac{X_j - x}{L} \right), \quad N_j = \left(\frac{x - X_i}{L} \right)$$





Galerkin Method (1/4)

- Governing Equation for 1D Steady State Heat Conduction Problems (Uniform λ):



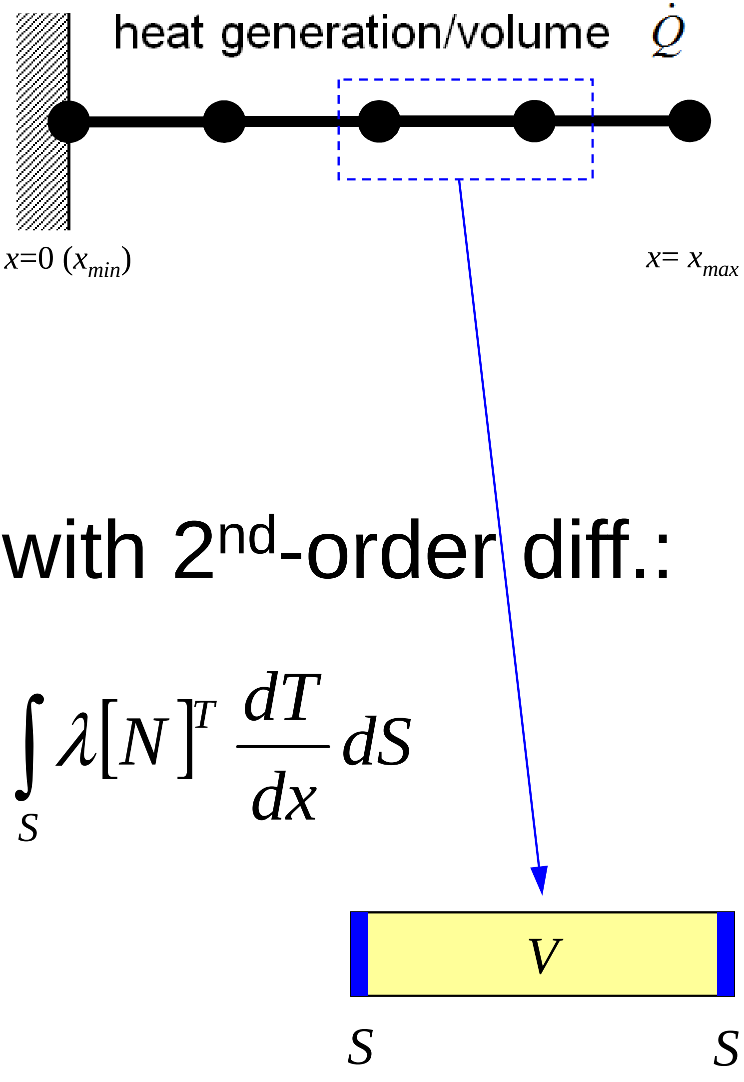
$$\lambda\left(\frac{d^2T}{dx^2}\right)+\dot{Q}=0$$

$$T=[N]\{\phi\}$$ Distribution of temperature in each element (matrix form), ϕ: Temperature at each node

- Following integral equation is obtained at each element by Galerkin method, where $[N]$'s are also weighting functions:

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

Galerkin Method (2/4)



- Green’s Theorem (1D)

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

- Apply this to the 1st part of eqn with 2nd-order diff.:

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

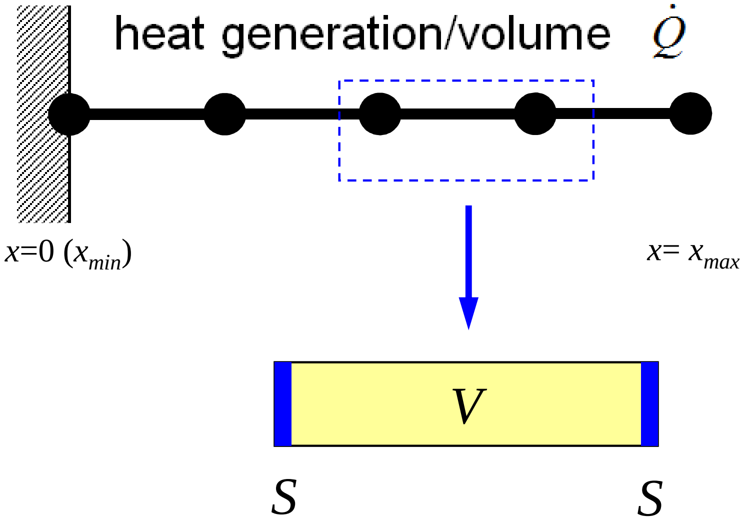
- Consider the following terms:

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

: Heat flux at element surface[$QL^{-2}T^{-1}$]

Galerkin Method (3/4)

- Finally, following eqn is obtained by considering heat generation term $\dot{Q}$:



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

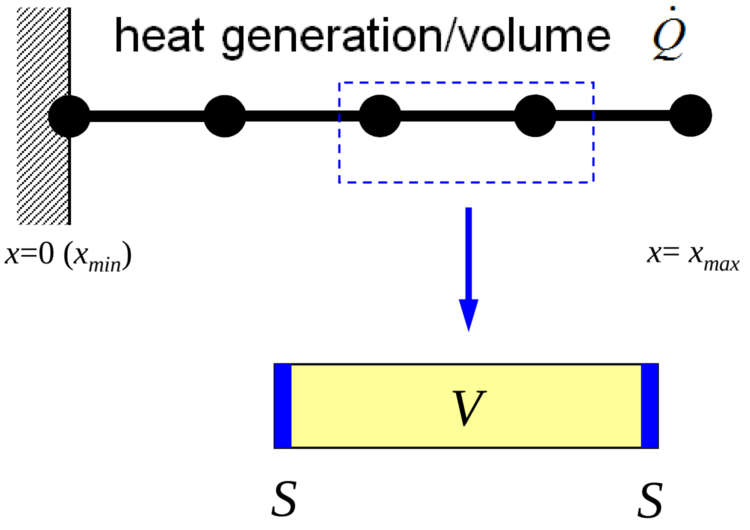
$$-\int_S \bar{q}[N]^T dS + \int_V Q[N]^T dV = 0$$

- This is called “weak form（弱形式）”. Original PDE consists of terms with 2nd-order diff., but this “weak form” only includes 1st-order diff by Green’s theorem.
  - Requirements for shape functions are “weaker” in “weak form”. Linear functions can describe effects of 2nd-order differentiation.

Galerkin Method (4/4)

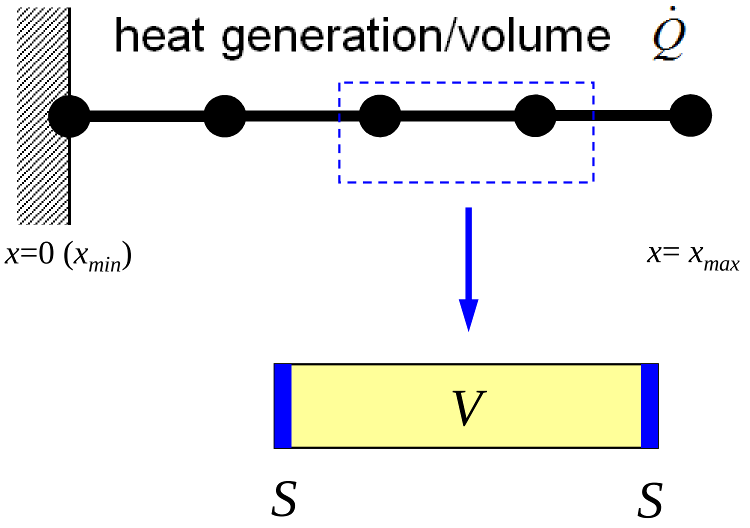
$$-\int_V \lambda \left(\frac{d[N]^T}{dx} \frac{d[N]}{dx} \right) dV \cdot \{\phi\}$$

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



- These terms coincide at element boundaries and disappear. Finally, only terms on the domain boundaries remain.

Weak Form and Boundary Conditions



- Value of dependent variable is defined (Dirichlet)
  - Weighting Function = 0
  - Principal B.C. (Boundary Condition)（第一種境界条件）
  - Essential B.C.（基本境界条件）
- Derivatives of Unknowns (Neumann)
  - Naturally satisfied in weak form
  - Secondary B.C.（第二種境界条件）
  - Natural B.C（自然境界条件）

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

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

- (Robin)
  - Linear Combination of Dirichlet & Neumann
  - Third Kind B.C.（第三種境界条件）
  - Electromagnetics

where $\bar{q} = -\lambda \dfrac{dT}{dx}$

Weak Form with B.C.: on each elem.

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

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

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

Integration over Each Element: [*k*]

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

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

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

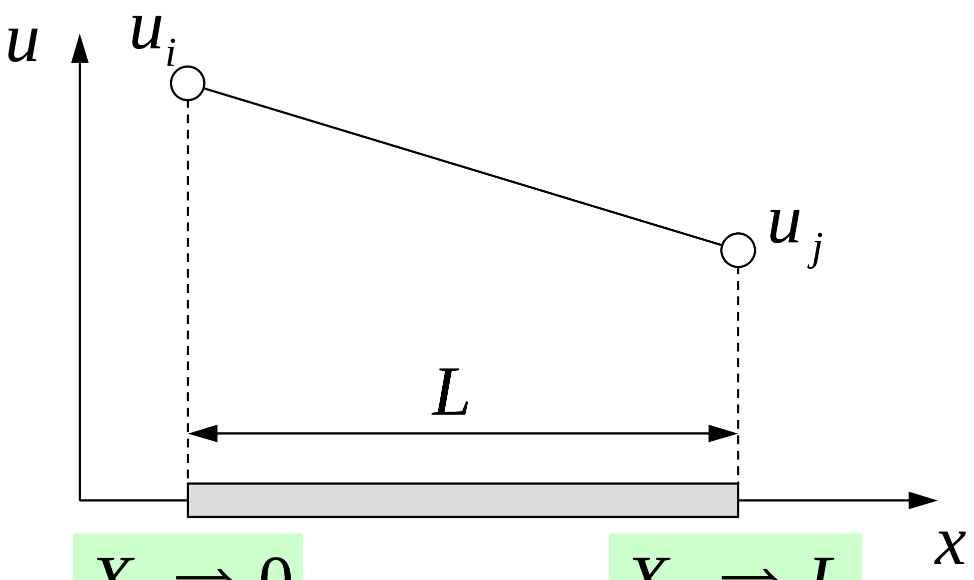
2x1 matrix

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: Sectional Area

L: Length



$X_i \Rightarrow 0$

$X_j \Rightarrow L$

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

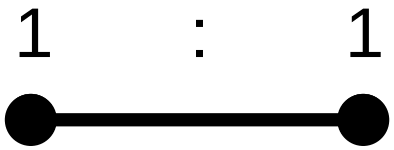
Integration over Each Element: {*f*} (1/2)

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

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

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

Heat Generation (Volume)



A: Sectional Area

L: Length

Integration over Each Element: {*f*} (2/2)

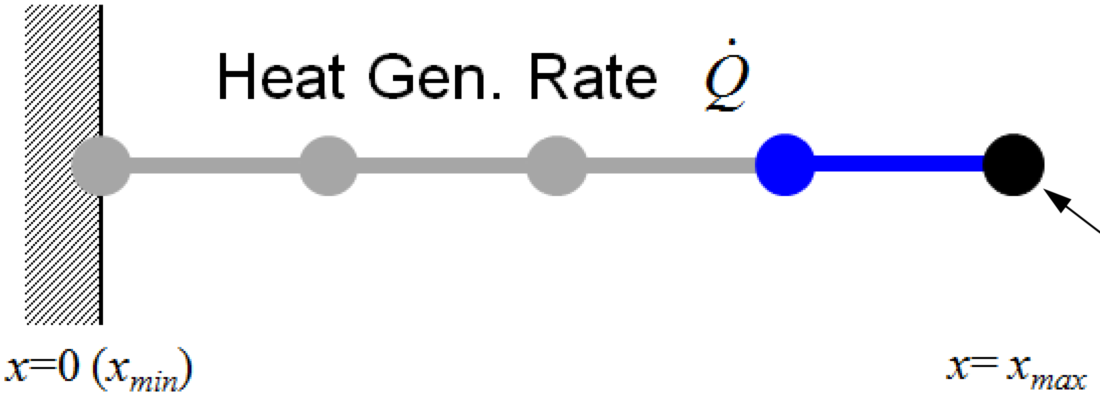
$$N_i = \left(\frac{X_j - x}{L}\right), \quad N_j = \left(\frac{x - X_i}{L}\right) \qquad \frac{dN_i}{dx} = \left(\frac{-1}{L}\right), \quad \frac{dN_j}{dx} = \left(\frac{1}{L}\right)$$

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

Heat Generation (Volume)

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

Surface Heat Flux



when surface heat flux acts on only this surface.

Global Equations

- Accumulate Element Equations:

$$[k]^{(e)}\{\phi\}^{(e)}=\{f\}^{(e)}$$ Element Matrix, Element Equations

↓

$$[K]\cdot\{\Phi\}=\underline{\{F\}}$$ Global Matrix, Global Equations

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

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

This is the final linear equations (global equations) to be solved.

Your PC

Download the Files

`(download http://nkl.cc.u-tokyo.ac.jp/files/fem-f.tar)`

`Copy to ¥Cygwin¥home¥YourName Directory on Windows`

1D Code for Steady-State Heat Conduction Problems

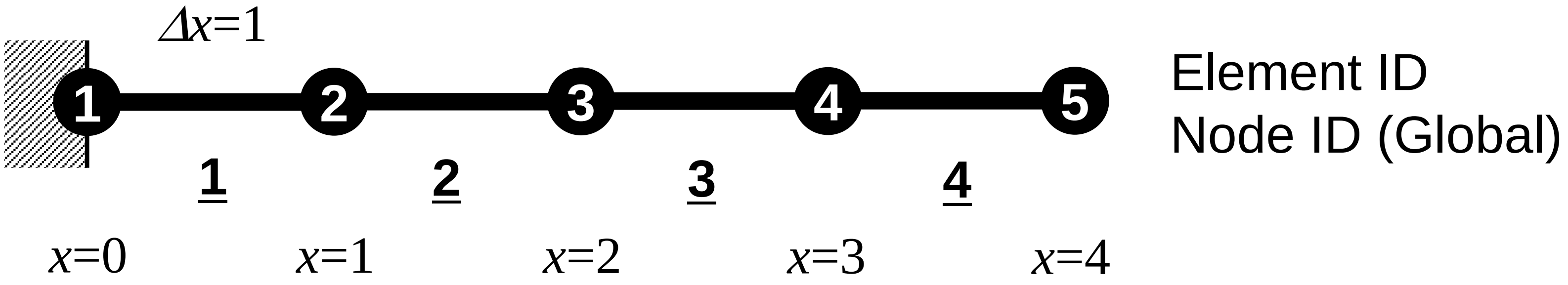
```
  >$ cd
  >$ tar xvf fem-f.tar
  >$ cd fem-f/1d
```

Compile & GO !

```
>$ cd
>$ cd fem-f/1d
>$ gfortran –O 1d.f
>$ ./a.exe
```

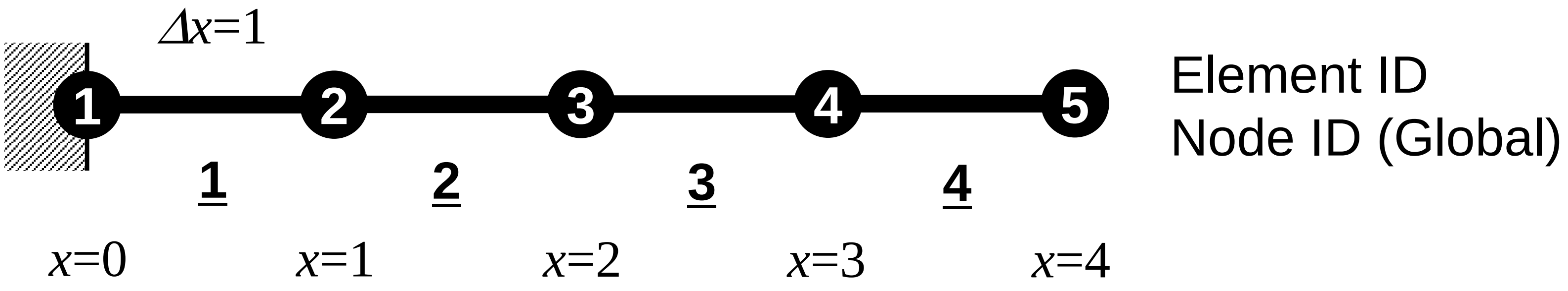
Control Data `input.dat`

4	**NE** (Number of Elements)
1.0 1.0 1.0 1.0	Δ**x** (Length of Each Elem.: **L**) , **Q, A,** λ
100	Number of MAX. Iterations for CG Solver
1.e-8	Convergence Criteria for CG Solver



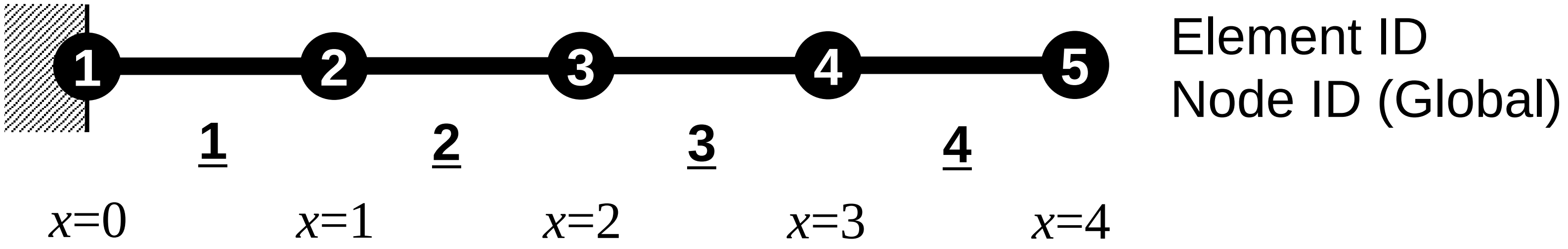
Results

```
>$ ./a.exe (or ./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

             Computational    Analytical
```



Element Eqn's/Accumulation (1/3)

- 4 elements, 5 nodes



- $[k]$ and $\{f\}$ of Element-1:

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

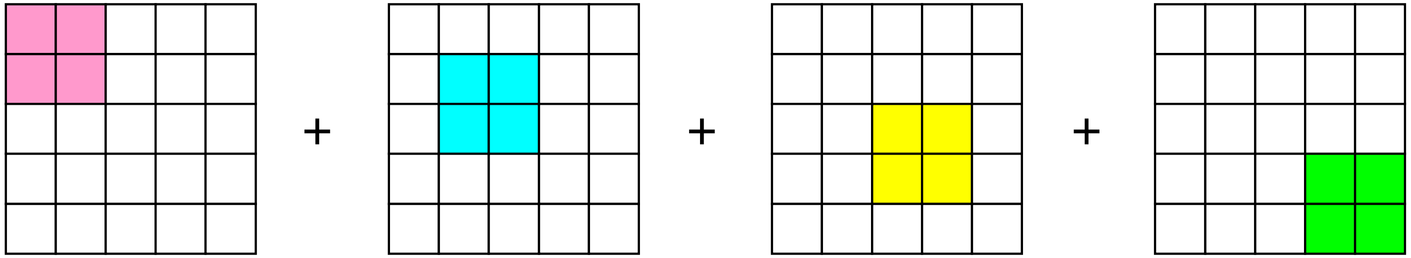
- As for Element-4:

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

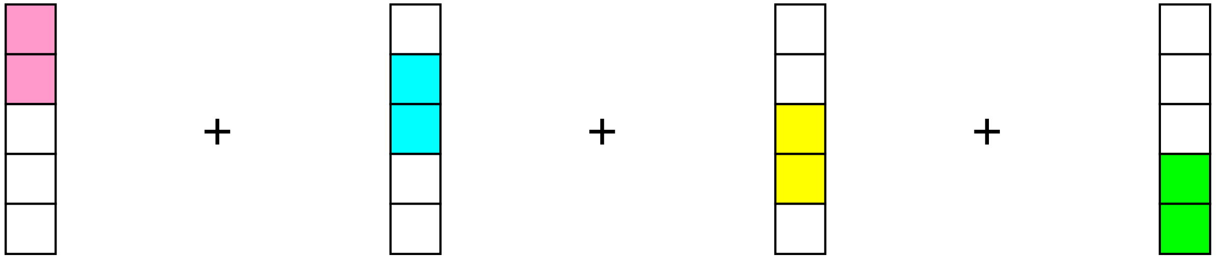
Element Eqn’s/Accumulation (2/3)

- Element-by-Element Accumulation:

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



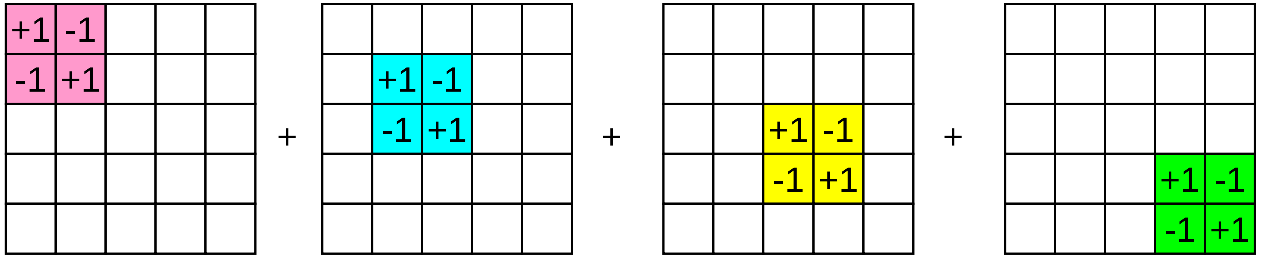
$$\{F\}=\sum_{e=1}^{4}\{f\}^{(e)}=$$



Element Eqn's/Accumulation (3/3)

- Relations to FDM

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



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

$$= \begin{array}{|c|c|c|c|c|} \hline +1 & -1 & & & \\ \hline -1 & +2 & -1 & & \\ \hline & -1 & +2 & -1 & \\ \hline & & -1 & +2 & -1 \\ \hline & & & -1 & +1 \\ \hline \end{array} \times \frac{\lambda A}{L}$$

三重対角行列
Tri-Diagonal Matrix

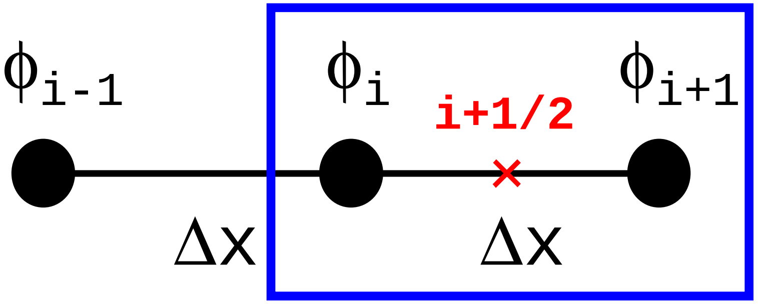
$$-\int_V \left(\frac{d^2T}{dx^2}\right) dV = -\int_V \left(\frac{T_{i+1}-2T_i+T_{i-1}}{L^2}\right) dV$$

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

Something familiar …
FEM: Coefficient Matrices are generally sparse (many ZERO's)

2nd –Order Differentiation in FDM

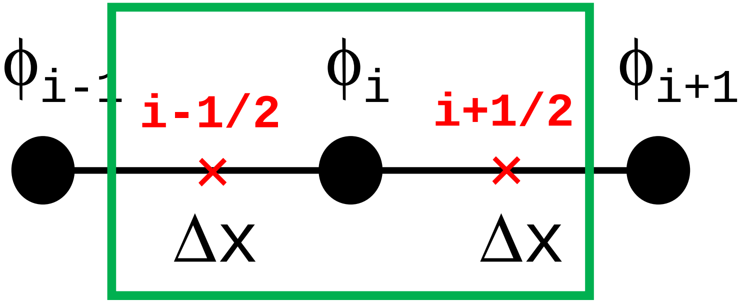
- Approximate Derivative at × (center of *i* and *i+1*)



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

Δx→0: Real Derivative

- 2nd-Order Diff. at *i*

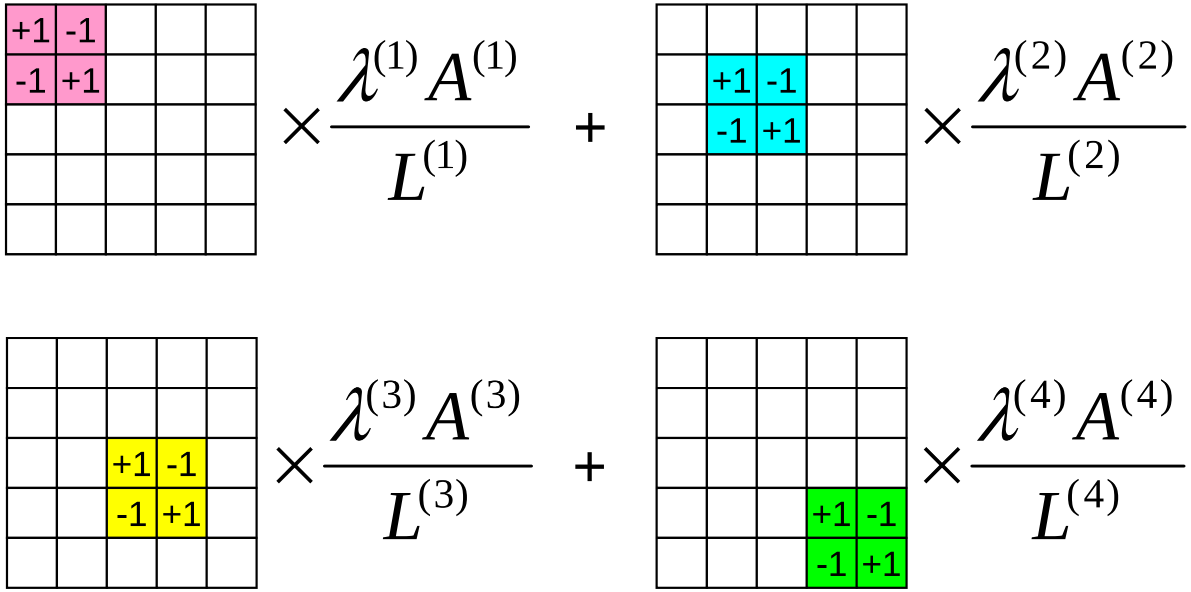


$$\left(\frac{d^2\phi}{dx^2}\right)_i \approx \frac{\left(\frac{d\phi}{dx}\right)_{i+1/2}-\left(\frac{d\phi}{dx}\right)_{i-1/2}}{\Delta x} = \frac{\frac{\phi_{i+1}-\phi_i}{\Delta x}-\frac{\phi_i-\phi_{i-1}}{\Delta x}}{\Delta x} = \frac{\phi_{i+1}-2\phi_i+\phi_{i-1}}{\Delta x^2}$$

Element-by-Element Operation

very flexible if each element has different material property, size, etc.

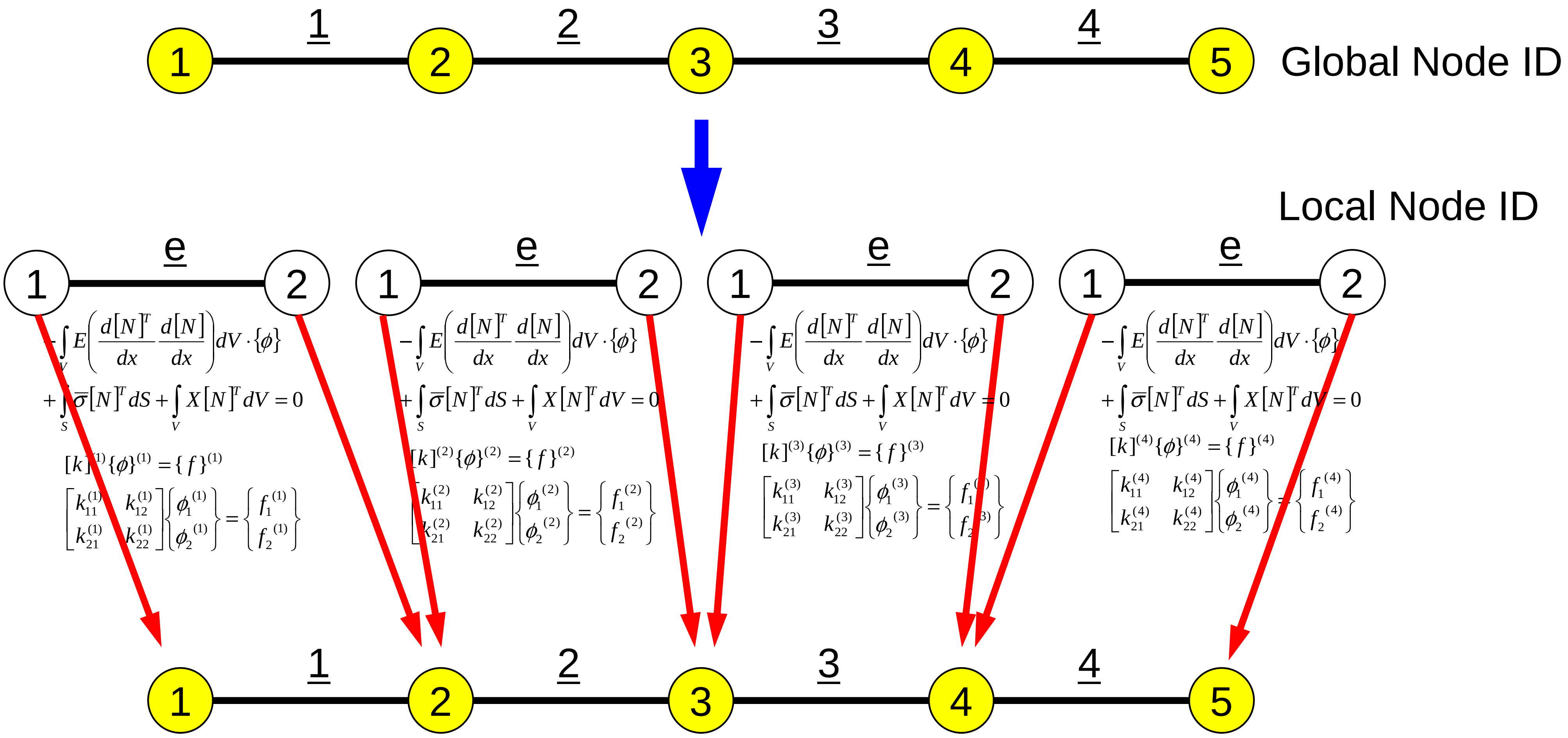
$$[k]^{(e)}=\frac{\lambda^{(e)}A^{(e)}}{L^{(e)}}\begin{bmatrix}+1 & -1\\ -1 & +1\end{bmatrix}$$



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

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

Element/Global Operations



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

$$[k]^{(1)}\{\phi\}^{(1)} = \{f\}^{(1)}$$

$$\begin{bmatrix} k_{11}^{(1)} & k_{12}^{(1)} \\ k_{21}^{(1)} & k_{22}^{(1)} \end{bmatrix}\begin{Bmatrix} \phi_1^{(1)} \\ \phi_2^{(1)} \end{Bmatrix} = \begin{Bmatrix} f_1^{(1)} \\ f_2^{(1)} \end{Bmatrix}$$

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

$$[k]^{(2)}\{\phi\}^{(2)} = \{f\}^{(2)}$$

$$\begin{bmatrix} k_{11}^{(2)} & k_{12}^{(2)} \\ k_{21}^{(2)} & k_{22}^{(2)} \end{bmatrix}\begin{Bmatrix} \phi_1^{(2)} \\ \phi_2^{(2)} \end{Bmatrix} = \begin{Bmatrix} f_1^{(2)} \\ f_2^{(2)} \end{Bmatrix}$$

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

$$[k]^{(3)}\{\phi\}^{(3)} = \{f\}^{(3)}$$

$$\begin{bmatrix} k_{11}^{(3)} & k_{12}^{(3)} \\ k_{21}^{(3)} & k_{22}^{(3)} \end{bmatrix}\begin{Bmatrix} \phi_1^{(3)} \\ \phi_2^{(3)} \end{Bmatrix} = \begin{Bmatrix} f_1^{(3)} \\ f_2^{(3)} \end{Bmatrix}$$

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

$$[k]^{(4)}\{\phi\}^{(4)} = \{f\}^{(4)}$$

$$\begin{bmatrix} k_{11}^{(4)} & k_{12}^{(4)} \\ k_{21}^{(4)} & k_{22}^{(4)} \end{bmatrix}\begin{Bmatrix} \phi_1^{(4)} \\ \phi_2^{(4)} \end{Bmatrix} = \begin{Bmatrix} f_1^{(4)} \\ f_2^{(4)} \end{Bmatrix}$$

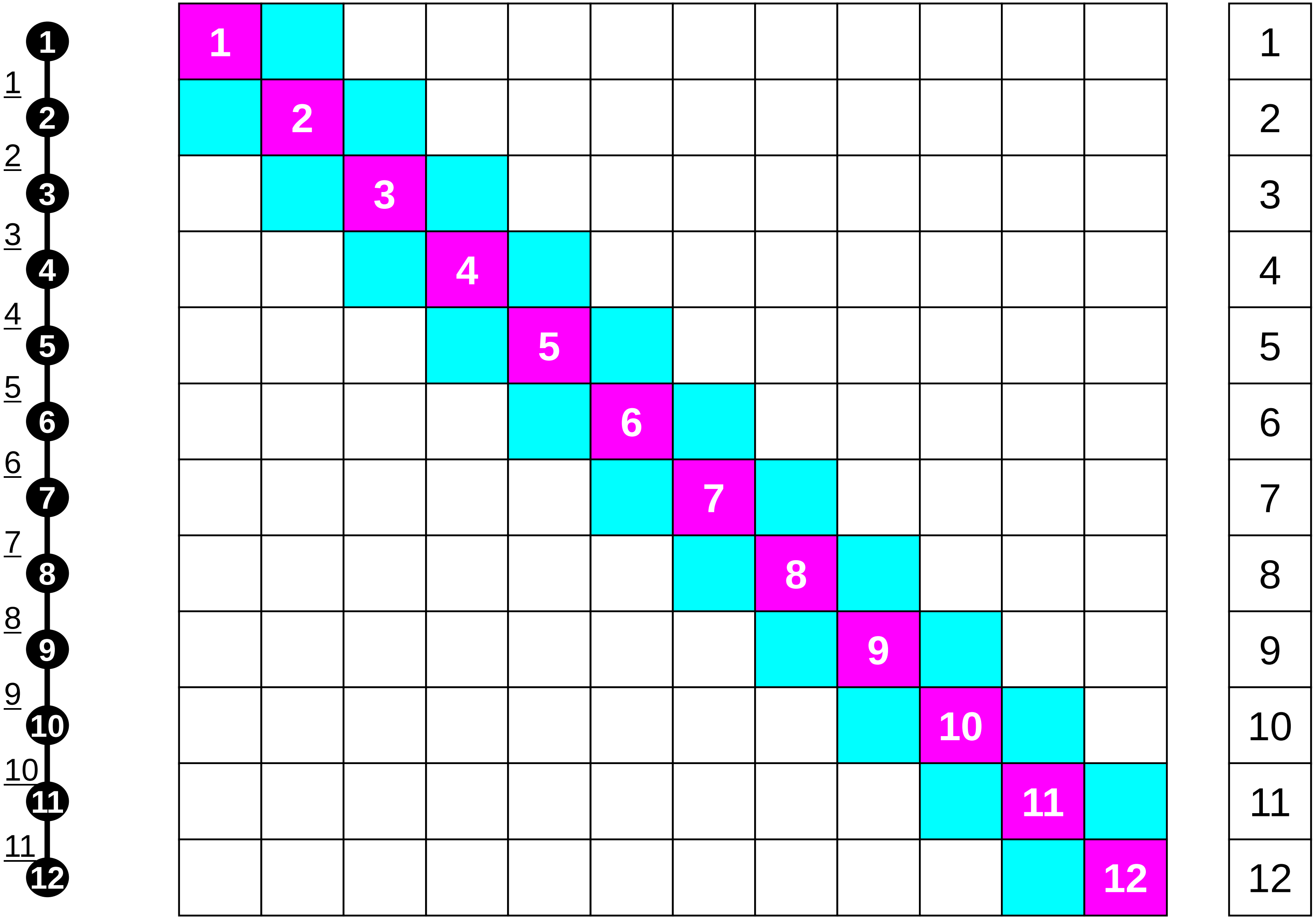
$$[K]\{\Phi\} = \{F\}$$

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

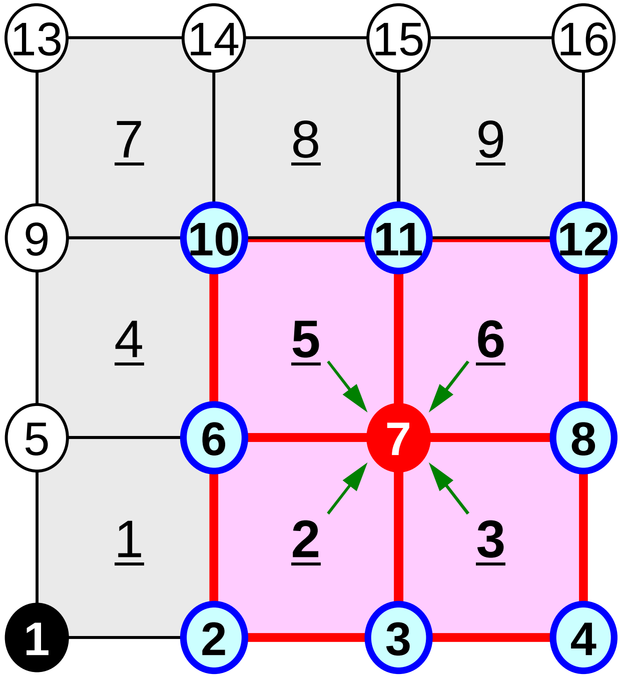
Mapping Information needed,
from element-matrix
to global-matrix.

1D FEM: 12 nodes/11 elem’s

三重対角行列：Tri-Diagonal Matrix



Accumulation to Global/overall Matrix



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

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

- 1D-code for Static Linear-Elastic Problems by Galerkin FEM
- Sparse Linear Solver
  - Conjugate Gradient Method
  - Preconditioning
- Storage of Sparse Matrices

- Program

Large-Scale Linear Equations in Scientific Applications

- Solving large-scale linear equations **Ax=b** is the most important and **expensive** part of various types of scientific computing.
  - for both linear and nonlinear applications
- Various types of methods proposed & developed.
  - for dense and sparse matrices
  - classified into ***direct*** and ***iterative*** methods
- Dense Matrices:密行列: Globally Coupled Problems
  - BEM, Spectral Methods, MO/MD (gas, liquid)
- Sparse Matrices:疎行列: Locally Defined Problems
  - FEM, FDM, DEM, MD (solid), BEM w/FMM

Direct Method
直接法

- Gaussian Elimination/LU Factorization.
  - compute **A**$^{-1}$ directly (or equivalent operations)

Good

  - Robust for wide range of applications.
  - Good for both dense and sparse matrices

Bad

  - More expensive than iterative methods (memory, CPU)
    - not scalable

What is Iterative Method ?
反復法

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

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

Iterative Method
反復法

- Stationary Method
  - Only $\mathbf{x}$ (solution vector) changes during iterations.
  - SOR, Gauss-Seidel, Jacobi
  - Generally slow, impractical

$$\mathbf{Ax} = \mathbf{b} \Rightarrow$$

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

- Non-Stationary Method
  - With restriction/optimization conditions
  - Krylov-Subspace
  - CG: Conjugate Gradient
  - BiCGSTAB: Bi-Conjugate Gradient Stabilized
  - GMRES: Generalized Minimal Residual

Iterative Method (cont.)

Good

- Less expensive than direct methods, especially in memory.
- Suitable for parallel and vector computing.

Bad

- Convergence strongly depends on problems, boundary conditions (condition number etc.)
- **Preconditioning is required : Key Technology for Parallel FEM**

Non-Stationary/Krylov Subspace Method (1/2)

非定常法・クリロフ部分空間法

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

Compute $\mathbf{x_0}$, $\mathbf{x_1}$, $\mathbf{x_2}$, ..., $\mathbf{x_k}$ by the following iterative procedures:

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

$where\ \mathbf{r}_k=\mathbf{b}-\mathbf{Ax}_k$: residual

⬇

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

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

Non-Stationary/Krylov Subspace Method (2/2)

非定常法・クリロフ部分空間法

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

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

$\mathbf{z}_\mathbf{k}$ is a vector which belongs to kth Krylov Subspace（クリロフ部分空間）, approximate solution vector $\mathbf{x}_\mathbf{k}$ is derived by the Krylov Subspace:

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

Conjugate Gradient Method
共役勾配法

- Conjugate Gradient: CG
  - Most popular “non-stationary” iterative method
- for Symmetric Positive Definite (SPD) Matrices
  - 対称正定
  - $\{x\}^{\mathrm{T}}[A]\{x\}>0$ for arbitrary $\{x\}$
  - All of diagonal components, eigenvaules and leading principal minors > 0 （主小行列式・首座行列式）
  - Matrices of Galerkin-based FEM: heat conduction, Poisson, static linear elastic problems

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

- Algorithm
  - “Steepest Descent Method”
  - $\mathbf{x}^{(i)}=\mathbf{x}^{(i-1)}+\alpha_i\mathbf{p}^{(i)}$
    - $x^{(i)}$: solution, $p^{(i)}$: search direction, α_i: coefficient
  - Solution $\{x\}$ minimizes $\{\boldsymbol{x}\text{-}\boldsymbol{y}\}^{\mathrm{T}}[\boldsymbol{A}]\{\boldsymbol{x}\text{-}\boldsymbol{y}\}$, where $\{y\}$ is exact solution.

Procedures of Conjugate Gradient

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

- Mat-Vec. Multiplication
- Dot Products
- DAXPY (Double Precision: a{X} + {Y})

$x^{(i)}$: Vector

α_i : Scalar

Procedures of Conjugate Gradient

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

- Mat-Vec. Multiplication
- Dot Products
- DAXPY

$x^{(i)}$: Vector

α_i : Scalar

Procedures of Conjugate Gradient

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

- Mat-Vec. Multiplication
- Dot Products
- DAXPY

$x^{(i)}$: Vector

α_i : Scalar

Procedures of Conjugate Gradient

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

- Mat-Vec. Multiplication
- Dot Products
- DAXPY
  - Double
  - a{x} + {y}

$x^{(i)}$: Vector

α_i : Scalar

Procedures of Conjugate Gradient

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

$x^{(i)}$: Vector

α_i : Scalar

Derivation of CG Algorithm (1/5)

Solution x minimizes the following equation if y is the exact solution ($Ay=b$)

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

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

Const.

Therefore, the solution x minimizes the following $f(x)$:

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

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

Arbitrary vector h

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

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

Arbitrary vector *h*

$$\begin{aligned}
f\left(x+h\right)&=\frac{1}{2}\left(x+h,A(x+h)\right)-\left(x+h,b\right)\\
&=\frac{1}{2}\left(x+h,Ax\right)+\frac{1}{2}\left(x+h,Ah\right)-\left(x,b\right)-\left(h,b\right)\\
&=\frac{1}{2}\left(x,Ax\right)+\underline{\frac{1}{2}\left(h,Ax\right)}+\underline{\frac{1}{2}\left(x,Ah\right)}+\frac{1}{2}\left(h,Ah\right)-\left(x,b\right)-\left(h,b\right)\\
&=\frac{1}{2}\left(x,Ax\right)-\left(x,b\right)+\left(h,Ax\right)-\left(h,b\right)+\frac{1}{2}\left(h,Ah\right)\\
&=f\left(x\right)+\left(h,Ax-b\right)+\frac{1}{2}\left(h,Ah\right)
\end{aligned}$$

Derivation of CG Algorithm (2/5)

CG method minimizes $f(x)$ at each iteration.
Assume that approximate solution: $x^{(0)}$, and
search direction vector $p^{(k)}$ is defined at k-th iteration.

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

Minimization of $f(x^{(k+1)})$ is done as follows:

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

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

$r^{(k)} = b - Ax^{(k)}$ residual vector

Derivation of CG Algorithm (3/5)

Residual vector at $(k+1)$-th iteration:

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

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

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

Search direction vector p is defined by the following recurrence formula:

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

It's lucky if we can get exact solution y at $(k+1)$-th iteration:

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

Derivation of CG Algorithm (4/5)

BTW, we have the following (convenient) orthogonality relation:

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

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

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

Thus, following relation is obtained:

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

Derivation of CG Algorithm (5/5)

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

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

$\left(p^{(k+1)}, Ap^{(k)}\right) = 0$ $p^{(k)}$ and $p^{(k+1)}$ are “conjugate（共役）” for matrix A

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

    check convergence |r|
    (if not converged)
    calc. β_{i-1}
    p^(i)= r^(i) + β_{i-1} p^(i-1)
end
```

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

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

Properties of CG Algorithm

Following “conjugate（共役）” relationship is obtained for arbitrary (i,j):

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

Following relationships are also obtained for $p^{(k)}$ and $r^{(k)}$:

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

In N-dimensional space, only N sets of orthogonal and linearly independent residual vector $r^{(k)}$. This means CG method converges after N iterations if number of unknowns is N. Actually, round-off error sometimes affects convergence.

Proof (1/3)

Mathematical Induction
数学的帰納法

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

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

(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

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

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

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

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

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

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

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

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

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

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

Proof (3/3)

Mathematical Induction

数学的帰納法

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

(＊)

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

$\alpha_k, \ \beta_k$

Usually, we use simpler definitions of $\alpha_k, \ \beta_k$ as follows:

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

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

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

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

Procedures of Conjugate Gradient

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

$x^{(i)}$: Vector

α_i : Scalar

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

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

Preconditioning for Iterative Solvers

- Convergence rate of iterative solvers strongly depends on the spectral properties (eigenvalue distribution) of the coefficient matrix **A**.
  - Eigenvalue distribution is small, eigenvalues are close to 1
  - In “<u>ill-conditioned</u>” problems, “<u>condition number</u>” (ratio of max/min eigenvalue if **A** is symmetric) is large（条件数）.
- A preconditioner **M** (whose properties are similar to those of **A**) transforms the linear system into one with more favorable spectral properties （前処理）
  - **M** transforms **Ax=b** into **A'x=b'** where $\mathbf{A'}=\mathbf{M}^{-1}\mathbf{A}$, $\mathbf{b'}=\mathbf{M}^{-1}\mathbf{b}$
  - If **M~A**, $\mathbf{M}^{-1}\mathbf{A}$ is close to identity matrix.
  - If $\mathbf{M}^{-1}=\mathbf{A}^{-1}$, this is the best preconditioner (Gaussian Elim.)
  - Generally, **A'x’=b’** where $\mathbf{A'}=\mathbf{M}_L^{-1}\mathbf{A}\mathbf{M}_R^{-1}$, $\mathbf{b'}=\mathbf{M}_L^{-1}\mathbf{b}$, $\mathbf{x'}=\mathbf{M}_R\mathbf{x}$
  - $\mathbf{M}_L/\mathbf{M}_R$: Left/Right Preconditioning（左／右前処理）

Preconditioned CG Solver

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

$$[M]= [M_1][M_2]$$

$$[A']x'=b'$$
$$[A']=[M_1]^{-1}[A][M_2]^{-1}$$
$$x'=[M_2]x,\ b'=[M_1]^{-1}b$$

$$p'=>[M_2]p,\ r'=>[M_1]^{-1}r$$

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

$$[M_2]p^{(i)}= [M_1]^{-1}r^{(i-1)} + \beta'_{i-1}\ [M_2]p^{(i-1)}$$

$$p^{(i)}= [M_2]^{-1}[M_1]^{-1}r^{(i-1)} + \beta'_{i-1}\ p^{(i-1)}$$
$$p^{(i)}= [M]^{-1}r^{(i-1)} + \beta'_{i-1}\ p^{(i-1)}$$

$$\beta'_{i-1}= ([M]^{-1}r^{(i-1)},r^{(i-1)})/ \\ ([M]^{-1}r^{(i-2)},r^{(i-2)})$$
$$\alpha'_{i-1}= ([M]^{-1}r^{(i-1)},r^{(i-1)})/ \\ (\ p^{(i-1)},[A]p^{(i-1)})$$

In CG method, preconditioner usually satisfies $[\mathrm{M}_2]=[\mathrm{M}_1]^T$, such as Incomplete Cholesky/Incomplete Modified Cholesky Factorizations.
In this problem, let us define $[\mathrm{M}_1]$ and $[\mathrm{M}_2]$ as follows:

$$[\mathrm{M}_1]=[\mathrm{X}]^T,[\mathrm{M}_2]=[\mathrm{X}],[\mathrm{M}]=[\mathrm{M}_1][\mathrm{M}_2]$$

$$\left[\mathrm{A}'\right]x'=b'$$

$$\left[\mathrm{A}'\right]=[\mathrm{M}_1]^{-1}\left[\mathrm{A}\right][\mathrm{M}_2]^{-1}=[\left[\mathrm{X}\right]^T]^{-1}\left[\mathrm{A}\right][\mathrm{X}]^{-1}=[\mathrm{X}]^{-T}\left[\mathrm{A}\right][\mathrm{X}]^{-1}$$

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

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

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

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

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

Preconditioning in PCG

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

Solving the following equation:

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

“Approximate Inverse Matrix”

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

Ultimate Preconditioning:
Inverse Matrix

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

Diagonal Scaling: Simple but weak

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

ILU(0), IC(0)

- Widely used Preconditioners for Sparse Matrices
  - Incomplete LU Factorization（不完全LU分解）
  - Incomplete Cholesky Factorization (for Symmetric Matrices)（不完全コレスキー分解）

- Incomplete Direct Method
  - Even if original matrix is sparse, inverse matrix is not necessarily sparse.
  - fill-in
  - ILU(0)/IC(0) without fill-in have same non-zero pattern with the original (sparse) matrices

Diagonal Scaling, Point-Jacobi

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

- **`solve [M]`**$\mathbf{z^{(i-1)} = r^{(i-1)}}$ is very easy.
- Provides fast convergence for simple problems.
- 1d.f, 1d.c

- 1D-code for Static Linear-Elastic Problems by Galerkin FEM
- Sparse Linear Solver
  - Conjugate Gradient Method
  - Preconditioning
- Storage of Sparse Matrices

- Program

Coef. Matrix derived from FEM

- Sparse Matrix
  - Many “0”’s
- Storing all components (e.g. A(*i*,*j*)) is not efficient for sparse matrices
  - A(*i*,*j*) is suitable for dense matrices

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

- Number of non-zero off-diagonal components is O(100) in FEM
  - If number of unknowns is 10^8 :
    - A(*i*,*j*): $O(10^{16})$ words ~ $O(10^{17})$ bytes for DP: 100PB (400x Odyssey)
    - Actual Non-zero Comp. : $O(10^{10})$ words: 100GB (Odyssey: 32GB/node)
- Only (really) non-zero off-diag. components should be stored on memory

Variables/Arrays in 1d.f, 1d.c related to coefficient matrix

name	type	size	description
`N`	`I`	`-`	# Unknowns
`NPLU`	`I`	`-`	# Non-Zero Off-Diagonal Components
`Diag(:)`	`R`	`N`	Diagonal Components
`PHI(:)`	`R`	`N`	Unknown Vector
`Rhs(:)`	`R`	`N`	RHS Vector
`Index(:)`	`I`	`0:N` `N+1`	Off-Diagonal Components (Number of Non-Zero Off-Diagonals at Each ROW)
`Item(:)`	`I`	`NPLU`	Off-Diagonal Components (Corresponding Column ID)
`AMat(:)`	`R`	`NPLU`	Off-Diagonal Components (Value)

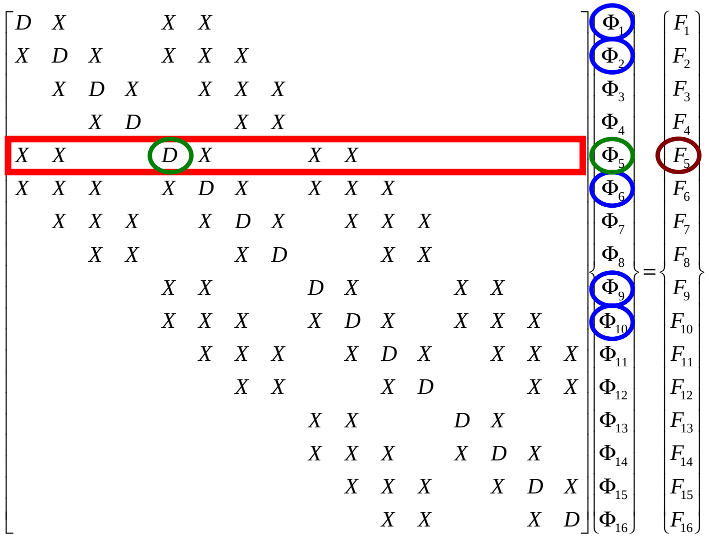
Only non-zero components are stored according to “Compressed Row Storage”.

Mat-Vec. Multiplication for Sparse Matrix

Compressed Row Storage (CRS)

`Diag (i)`	Diagonal Components (REAL, i=1~N)
`Index(i)`	Number of Non-Zero Off-Diagonals at Each ROW (INT, i=0~N)
`Item(k)`	Off-Diagonal Components (Corresponding Column ID) （INT, k=1, index(N)）
`AMat(k)`	Off-Diagonal Components (Value) （ REAL, k=1, index(N) ）

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

do i= 1, N
  Y(i)= Diag(i)*X(i)
  do k= Index(i-1)+1, Index(i)
    Y(i)= Y(i) + Amat(k)*X(Item(k))
  enddo
enddo
```



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

Mat-Vec. Multiplication for Dense Matrix

Very Easy, Straightforward

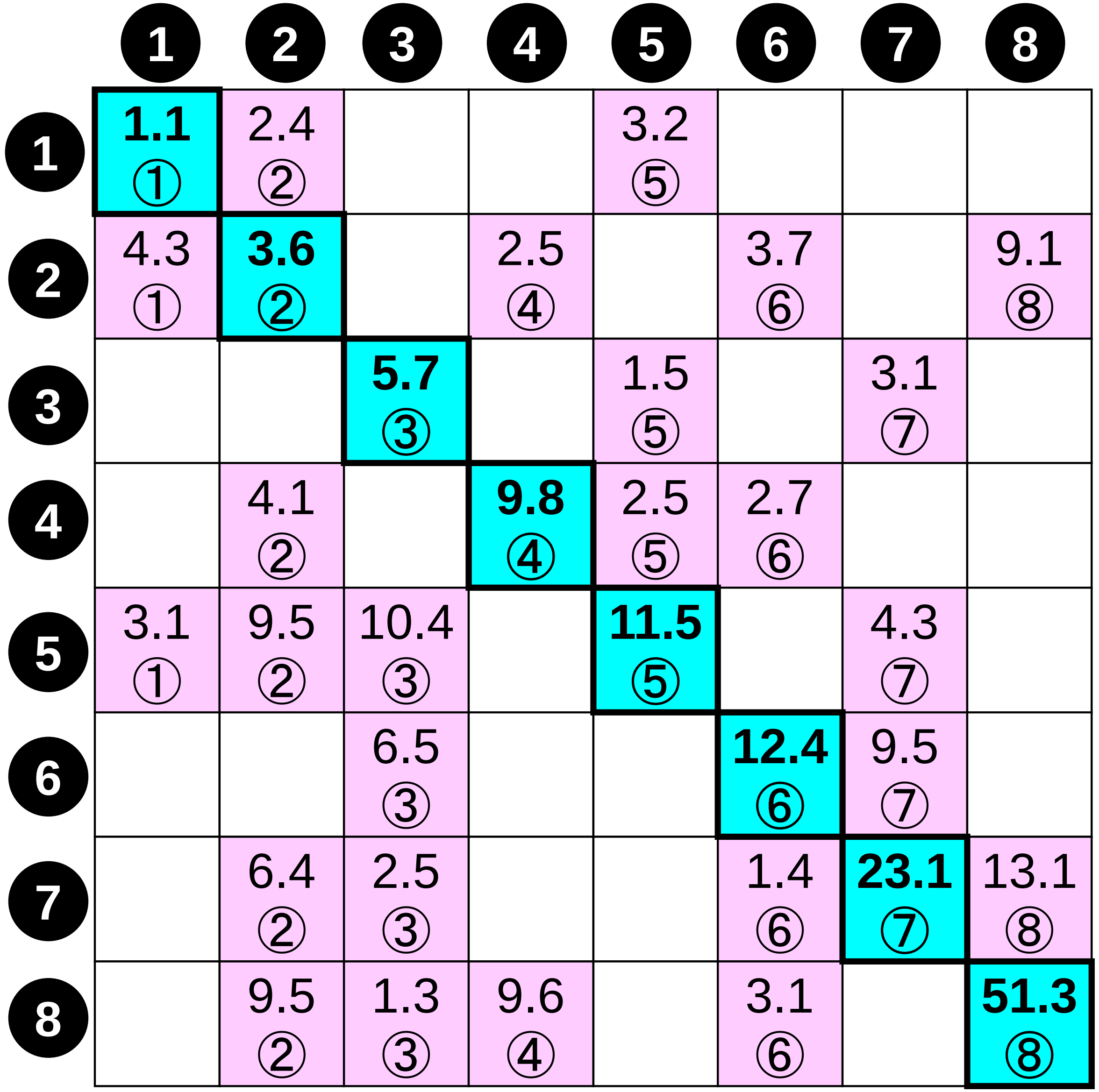
$$\begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1,N-1} & a_{1,N} \\ a_{21} & a_{22} & & a_{2,N-1} & a_{2,N} \\ \cdots & & \cdots & & \cdots \\ a_{N-1,1} & a_{N-1,2} & & a_{N-1,N-1} & a_{N-1,N} \\ a_{N,1} & a_{N,2} & \cdots & a_{N,N-1} & a_{N,N} \end{bmatrix} \begin{Bmatrix} x_1 \\ x_2 \\ \vdots \\ x_{N-1}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

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

do j= 1, N
  Y(j)= 0.d0
  do i= 1, N
    Y(j)= Y(j) + A(i,j)*X(i)
  enddo
enddo
```

Compressed Row Storage (CRS)

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

Compressed Row Storage (CRS): Fortran



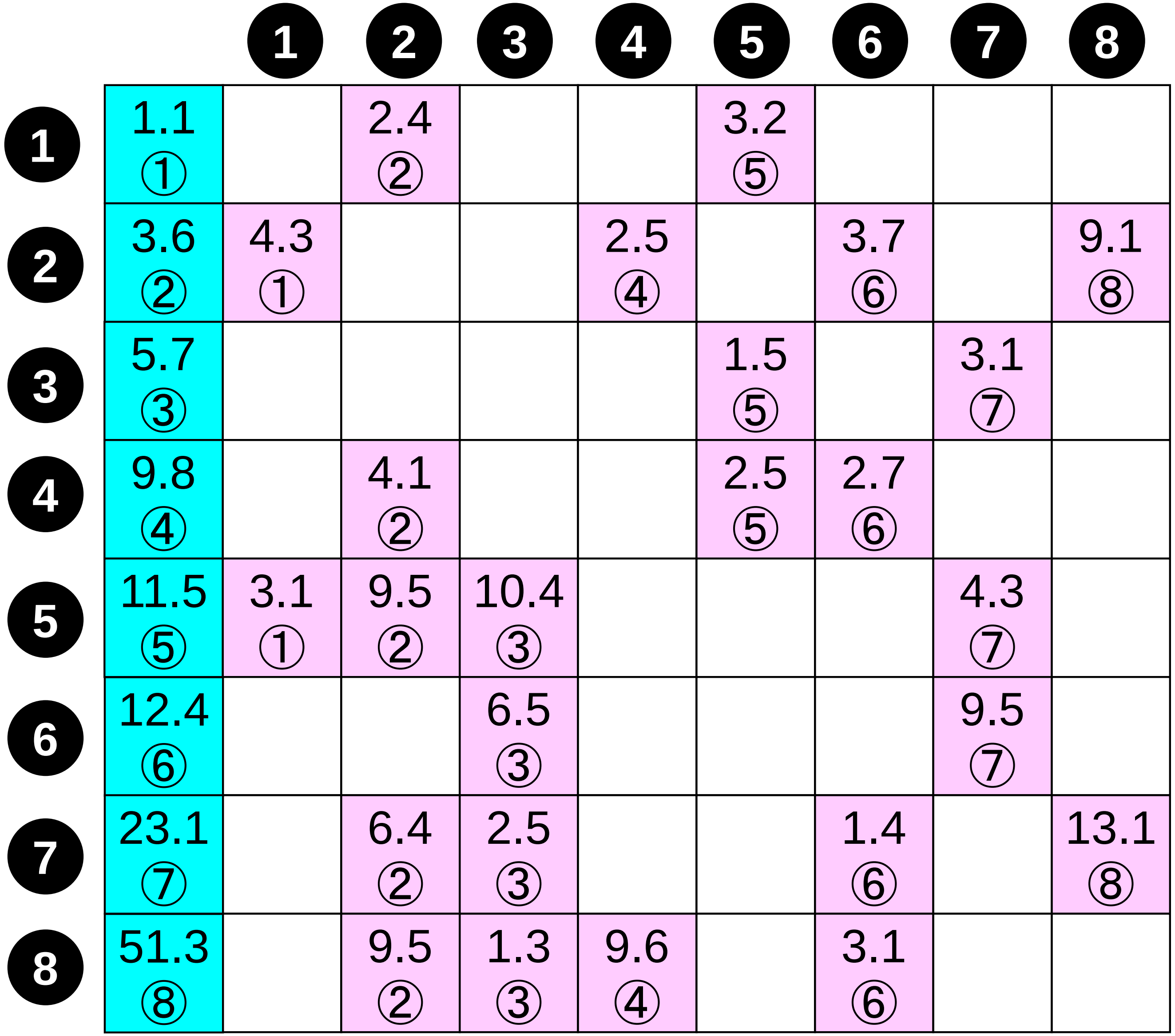
	1	2	3	4	5	6	7	8
1	**1.1** ①	2.4 ②			3.2 ⑤			
2	4.3 ①	**3.6** ②		2.5 ④		3.7 ⑥		9.1 ⑧
3			**5.7** ③		1.5 ⑤		3.1 ⑦	
4		4.1 ②		**9.8** ④	2.5 ⑤	2.7 ⑥		
5	3.1 ①	9.5 ②	10.4 ③		**11.5** ⑤		4.3 ⑦	
6			6.5 ③			**12.4** ⑥	9.5 ⑦	
7		6.4 ②	2.5 ③			1.4 ⑥	**23.1** ⑦	13.1 ⑧
8		9.5 ②	1.3 ③	9.6 ④		3.1 ⑥		**51.3** ⑧

```
N= 8
```

対角成分

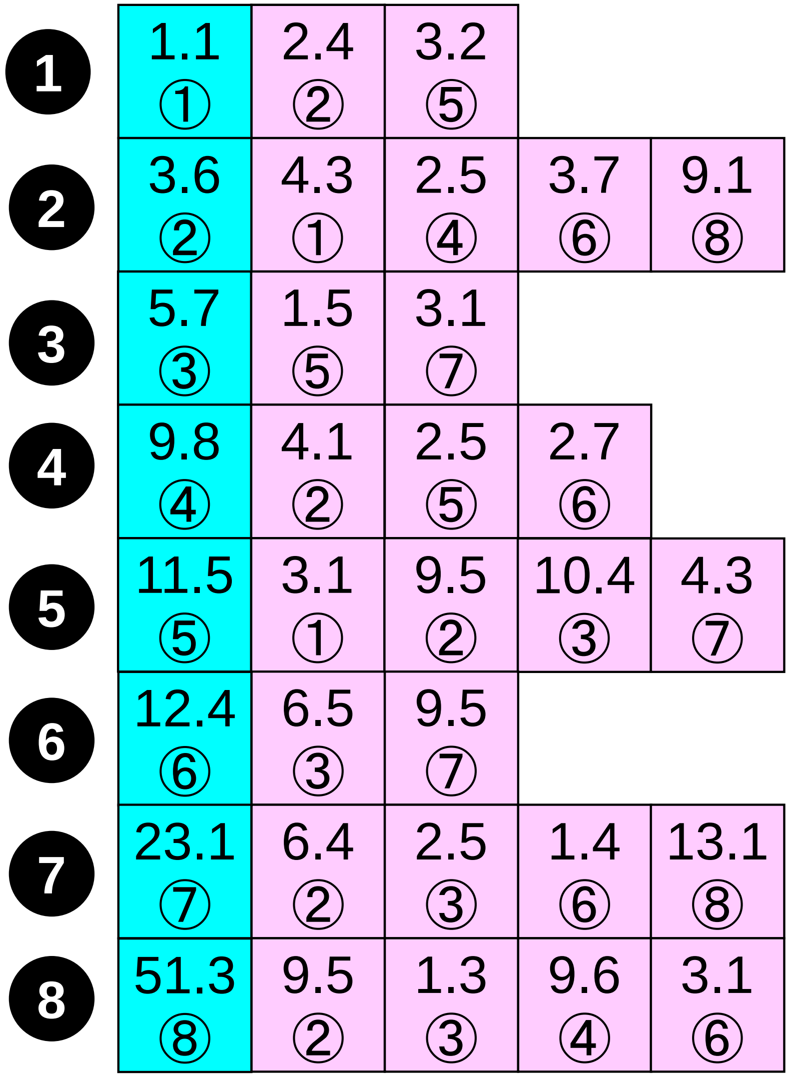
```
 Diag(1)=  1.1
 Diag(2)=  3.6
 Diag(3)=  5.7
 Diag(4)=  9.8
 Diag(5)= 11.5
 Diag(6)= 12.4
 Diag(7)= 23.1
 Diag(8)= 51.3
```

Compressed Row Storage (CRS)



		❶	❷	❸	❹	❺	❻	❼	❽
❶	1.1 ①		2.4 ②			3.2 ⑤			
❷	3.6 ②	4.3 ①			2.5 ④		3.7 ⑥		9.1 ⑧
❸	5.7 ③					1.5 ⑤		3.1 ⑦	
❹	9.8 ④		4.1 ②			2.5 ⑤	2.7 ⑥		
❺	11.5 ⑤	3.1 ①	9.5 ②	10.4 ③				4.3 ⑦	
❻	12.4 ⑥			6.5 ③				9.5 ⑦	
❼	23.1 ⑦		6.4 ②	2.5 ③			1.4 ⑥		13.1 ⑧
❽	51.3 ⑧		9.5 ②	1.3 ③	9.6 ④		3.1 ⑥		

Compressed Row Storage (CRS)

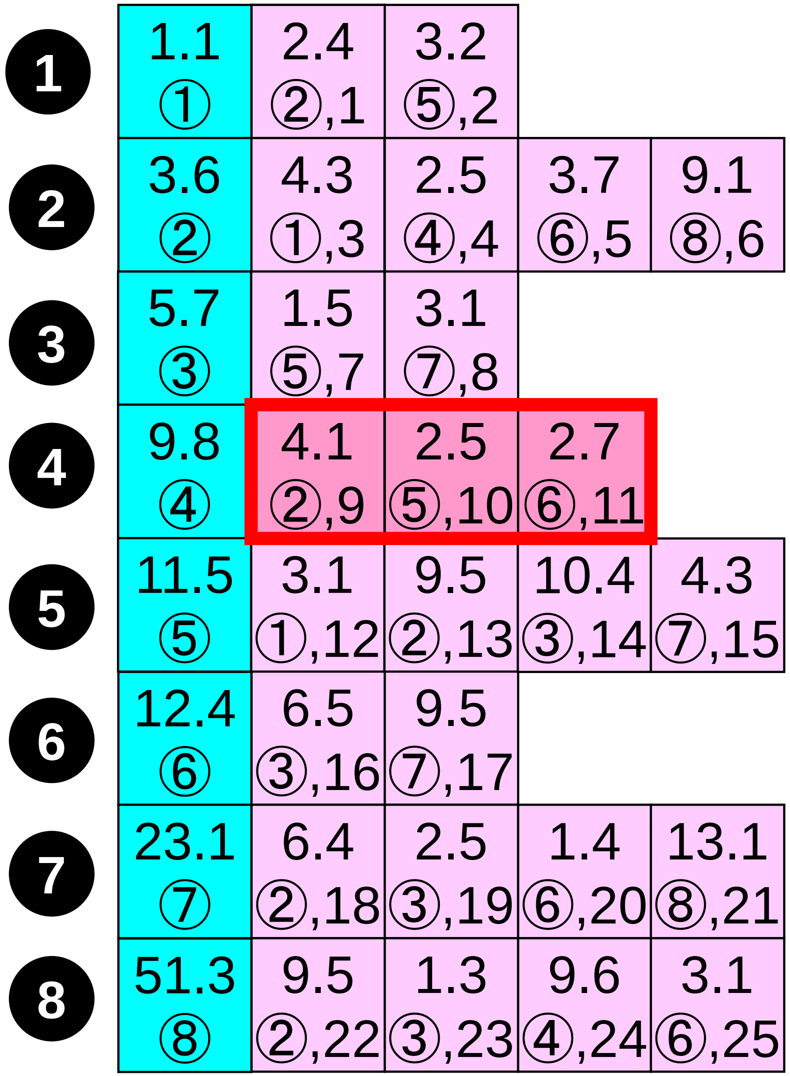


						# Non-Zero Off-Diag.	index(0)= 0
❶	1.1 ①	2.4 ②	3.2 ⑤			2	index(1)= 2
❷	3.6 ②	4.3 ①	2.5 ④	3.7 ⑥	9.1 ⑧	4	index(2)= 6
❸	5.7 ③	1.5 ⑤	3.1 ⑦			2	index(3)= 8
❹	9.8 ④	4.1 ②	2.5 ⑤	2.7 ⑥		3	index(4)= 11
❺	11.5 ⑤	3.1 ①	9.5 ②	10.4 ③	4.3 ⑦	4	index(5)= 15
❻	12.4 ⑥	6.5 ③	9.5 ⑦			2	index(6)= 17
❼	23.1 ⑦	6.4 ②	2.5 ③	1.4 ⑥	13.1 ⑧	4	index(7)= 21
❽	51.3 ⑧	9.5 ②	1.3 ③	9.6 ④	3.1 ⑥	4	index(8)= 25

NPLU= 25
(=index(N))

`index(i-1)+1`th~`index(i)` th
Non-Zero Off-Diag. Components corresponding to *i*-th row

Compressed Row Storage (CRS)

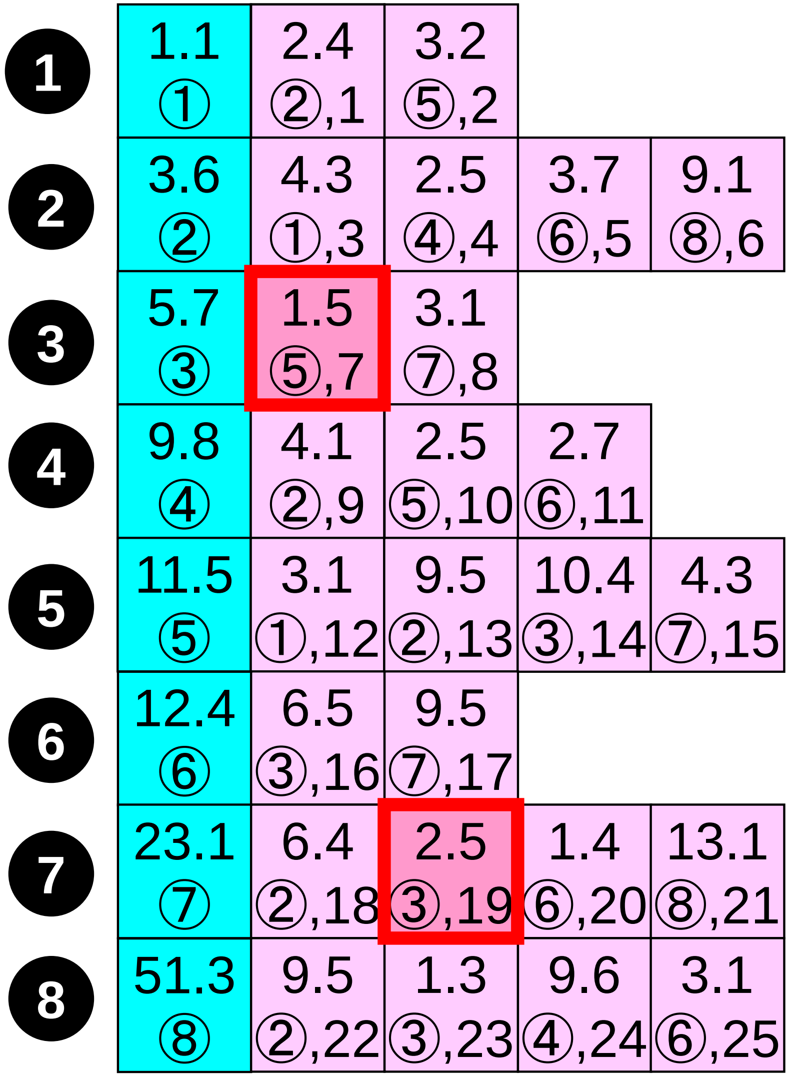


						# Non-Zero Off-Diag.	index(0)= 0
❶	1.1 ①	2.4 ②,1	3.2 ⑤,2			2	index(1)= 2
❷	3.6 ②	4.3 ①,3	2.5 ④,4	3.7 ⑥,5	9.1 ⑧,6	4	index(2)= 6
❸	5.7 ③	1.5 ⑤,7	3.1 ⑦,8			2	**index(3)= 8**
❹	9.8 ④	4.1 ②,9	2.5 ⑤,10	2.7 ⑥,11		**3**	**index(4)= 11**
❺	11.5 ⑤	3.1 ①,12	9.5 ②,13	10.4 ③,14	4.3 ⑦,15	4	index(5)= 15
❻	12.4 ⑥	6.5 ③,16	9.5 ⑦,17			2	index(6)= 17
❼	23.1 ⑦	6.4 ②,18	2.5 ③,19	1.4 ⑥,20	13.1 ⑧,21	4	index(7)= 21
❽	51.3 ⑧	9.5 ②,22	1.3 ③,23	9.6 ④,24	3.1 ⑥,25	4	index(8)= 25

NPLU= 25
(=index(N))

index(i-1)+1th**~index(i)** th
Non-Zero Off-Diag. Components corresponding to *i*-th row

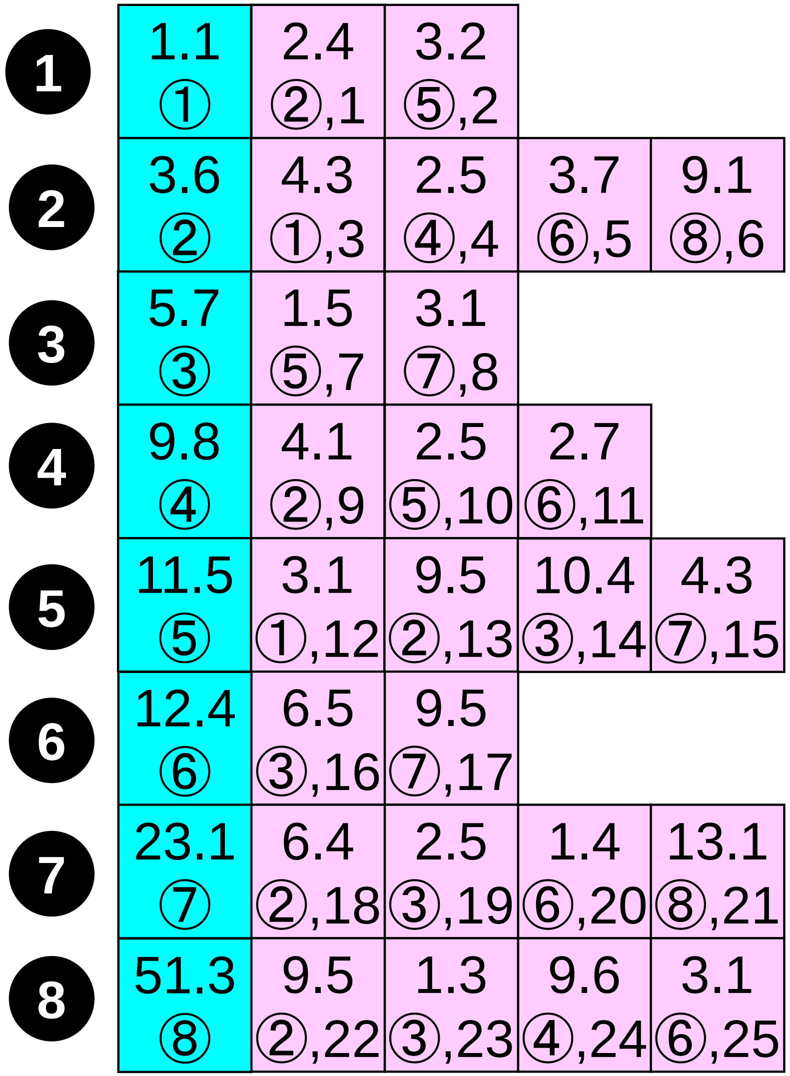
Compressed Row Storage (CRS)



❶	1.1 ①	2.4 ②,1	3.2 ⑤,2		
❷	3.6 ②	4.3 ①,3	2.5 ④,4	3.7 ⑥,5	9.1 ⑧,6
❸	5.7 ③	1.5 ⑤,7	3.1 ⑦,8		
❹	9.8 ④	4.1 ②,9	2.5 ⑤,10	2.7 ⑥,11	
❺	11.5 ⑤	3.1 ①,12	9.5 ②,13	10.4 ③,14	4.3 ⑦,15
❻	12.4 ⑥	6.5 ③,16	9.5 ⑦,17		
❼	23.1 ⑦	6.4 ②,18	2.5 ③,19	1.4 ⑥,20	13.1 ⑧,21
❽	51.3 ⑧	9.5 ②,22	1.3 ③,23	9.6 ④,24	3.1 ⑥,25

```
Example:
item( 7)= 5, AMAT( 7)= 1.5
item(19)= 3, AMAT(19)= 2.5
```

Compressed Row Storage (CRS)



❶	1.1 ①	2.4 ②,1	3.2 ⑤,2		
❷	3.6 ②	4.3 ①,3	2.5 ④,4	3.7 ⑥,5	9.1 ⑧,6
❸	5.7 ③	1.5 ⑤,7	3.1 ⑦,8		
❹	9.8 ④	4.1 ②,9	2.5 ⑤,10	2.7 ⑥,11	
❺	11.5 ⑤	3.1 ①,12	9.5 ②,13	10.4 ③,14	4.3 ⑦,15
❻	12.4 ⑥	6.5 ③,16	9.5 ⑦,17		
❼	23.1 ⑦	6.4 ②,18	2.5 ③,19	1.4 ⑥,20	13.1 ⑧,21
❽	51.3 ⑧	9.5 ②,22	1.3 ③,23	9.6 ④,24	3.1 ⑥,25

`Diag (i)`	Diagonal Components (REAL, i=1~N)
`Index(i)`	Number of Non-Zero Off-Diagonals at Each ROW (INT, i=0~N)
`Item(k)`	Off-Diagonal Components (Corresponding Column ID) （INT, k=1, index(N)）
`AMat(k)`	Off-Diagonal Components (Value) （ REAL, k=1, index(N) ）

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

do i= 1, N
  Y(i)= D(i)*X(i)
  do k= index(i-1)+1, index(i)
    Y(i)= Y(i) + AMAT(k)*X(item(k))
  enddo
enddo
```

- 1D-code for Static Linear-Elastic Problems by Galerkin FEM
- Sparse Linear Solver
  - Conjugate Gradient Method
  - Preconditioning
- Storage of Sparse Matrices

- Program

Finite Element Procedures

- Initialization
  - Control Data
  - Node, Connectivity of Elements (N: Node#, NE: Elem#)
  - Initialization of Arrays (Global/Element Matrices)
  - Element-Global Matrix Mapping (Index, Item)
- Generation of Matrix
  - Element-by-Element Operations (do icel= 1, NE)
    - Element matrices
    - Accumulation to global matrix
  - Boundary Conditions
- Linear Solver
  - Conjugate Gradient Method

Program: 1d.f (1/6)

variables and arrays

```
!C
!C   1D Steady-State Heat Transfer
!C   FEM with Piece-wise Linear Elements
!C   CG (Conjugate Gradient) Method
!C
!C   d/dx(CdT/dx) + Q = 0
!C   T=0@x=0
!C
      program heat1D
      implicit REAL*8 (A-H,O-Z)

      integer :: N, NPLU, ITERmax
      integer :: R, Z, P, Q, DD

      real(kind=8) :: dX, RESID, EPS
      real(kind=8) :: AREA, QV, COND
      real(kind=8), dimension(:), allocatable :: PHI, RHS, X
      real(kind=8), dimension(:  ), allocatable :: DIAG, AMAT
      real(kind=8), dimension(:,:), allocatable :: W

      real(kind=8), dimension(2,2) :: KMAT, EMAT

      integer, dimension(:), allocatable :: ICELNOD
      integer, dimension(:), allocatable :: INDEX, ITEM
```

Variable/Arrays (1/2)

Name	Type	Size	I/O	Definition
NE	I		I	# Element
N	I		O	# Node
NPLU	I		O	# Non-Zero Off-Diag. Components
IterMax	I		I	MAX Iteration Number for CG
errno	I		O	ERROR flag
R, Z, Q, P, DD	I		O	Name of Vectors in CG
dX	R		I	Length of Each Element
Resid	R		O	Residual for CG
Eps	R		I	Convergence Criteria for CG
Area	R		I	Sectional Area of Element
QV	R		I	Heat Generation Rate/Volume/Time $\dot{Q}$
COND	R		I	Thermal Conductivity

Variable/Arrays (2/2)

Name	Type	Size	I/O	Definition
X	R	N	O	Location of Each Node
PHI	R	N	O	Temperature of Each Node
Rhs	R	N	O	RHS Vector
Diag	R	N	O	Diagonal Components
W	R	(N,4)	O	Work Array for CG
Amat	R	NPLU	O	Off-Diagonal Components (Value)
Index	I	0:N	O	Number of Non-Zero Off-Diagonals at Each ROW
Item	I	NPLU	O	Off-Diagonal Components (Corresponding Column ID)
Icelnod	I	2*NE	O	Node ID for Each Element
Kmat	R	(2,2)	O	Element Matrix [*k*]
Emat	R	(2,2)	O	Element Matrix

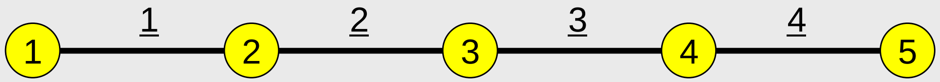
Program: 1d.f (2/6)

Initialization, Allocation of Arrays

```
!C
!C +-------+
!C | INIT. |
!C +-------+
!C===
      open  (11, file='input.dat', status='unknown')
       read (11,*) NE
       read (11,*) dX, QV, AREA, COND
       read (11,*) ITERmax
       read (11,*) EPS
      close (11)


      N= NE + 1
      allocate (PHI(N), DIAG(N), AMAT(2*N-2), RHS(N))
      allocate (ICELNOD(2*NE), X(N))
      allocate (INDEX(0:N), ITEM(2*N-2), W(N,4))

      PHI  = 0.d0
      AMAT= 0.d0
      DIAG= 0.d0
       RHS= 0.d0
         X= 0.d0
```

Control Data **`input.dat`**

4	**NE** (Number of Elements)
1.0 1.0 1.0 1.0	**Δx** (Length of Each Elem.: **L**) , **Q, A, λ**
100	Number of MAX. Iterations for CG Solver
1.e-8	Convergence Criteria for CG Solver



NE: # Element

N : # Node (NE+1)

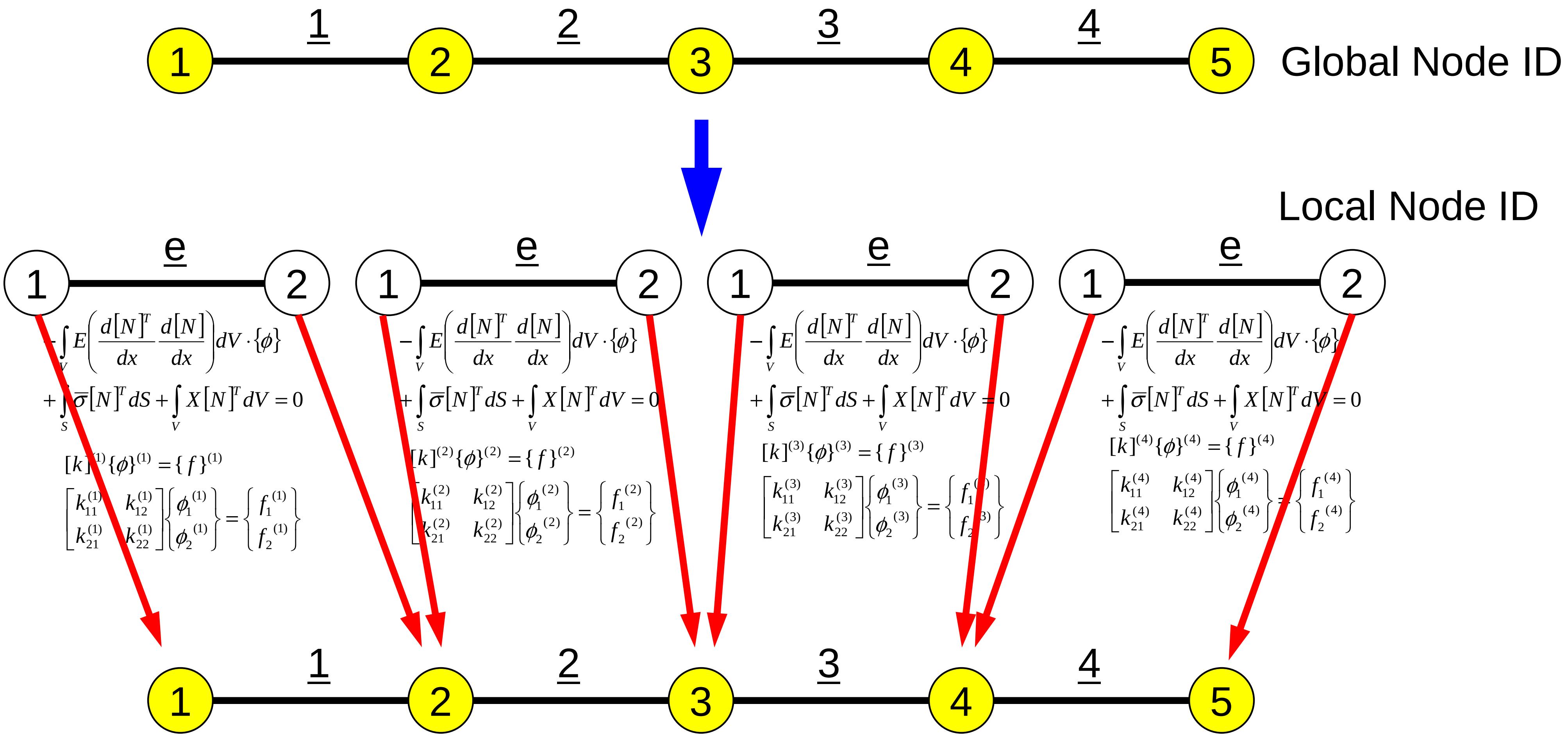
Program: 1d.f (2/6)

Initialization, Allocation of Arrays

```
!C
!C +--------+
!C | INIT.  |
!C +--------+
!C===
      open  (11, file='input.dat', status='unknown')
       read (11,*) NE
       read (11,*) dX, QV, AREA, COND
       read (11,*) ITERmax
       read (11,*) EPS
      close (11)


      N= NE + 1
      allocate (PHI(N), DIAG(N), AMAT(2*N-2), RHS(N))
      allocate (ICELNOD(2*NE), X(N))
      allocate (INDEX(0:N), ITEM(2*N-2), W(N,4))

      PHI  = 0.d0
      AMAT= 0.d0
      DIAG= 0.d0
       RHS= 0.d0
         X= 0.d0
```

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

Element/Global Operations



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

$$[k]^{(1)}\{\phi\}^{(1)}=\{f\}^{(1)}$$

$$\begin{bmatrix} k_{11}^{(1)} & k_{12}^{(1)} \\ k_{21}^{(1)} & k_{22}^{(1)} \end{bmatrix}\begin{Bmatrix} \phi_1^{(1)} \\ \phi_2^{(1)} \end{Bmatrix}=\begin{Bmatrix} f_1^{(1)} \\ f_2^{(1)} \end{Bmatrix}$$

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

$$[k]^{(2)}\{\phi\}^{(2)}=\{f\}^{(2)}$$

$$\begin{bmatrix} k_{11}^{(2)} & k_{12}^{(2)} \\ k_{21}^{(2)} & k_{22}^{(2)} \end{bmatrix}\begin{Bmatrix} \phi_1^{(2)} \\ \phi_2^{(2)} \end{Bmatrix}=\begin{Bmatrix} f_1^{(2)} \\ f_2^{(2)} \end{Bmatrix}$$

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

$$[k]^{(3)}\{\phi\}^{(3)}=\{f\}^{(3)}$$

$$\begin{bmatrix} k_{11}^{(3)} & k_{12}^{(3)} \\ k_{21}^{(3)} & k_{22}^{(3)} \end{bmatrix}\begin{Bmatrix} \phi_1^{(3)} \\ \phi_2^{(3)} \end{Bmatrix}=\begin{Bmatrix} f_1^{(3)} \\ f_2^{(3)} \end{Bmatrix}$$

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

$$[k]^{(4)}\{\phi\}^{(4)}=\{f\}^{(4)}$$

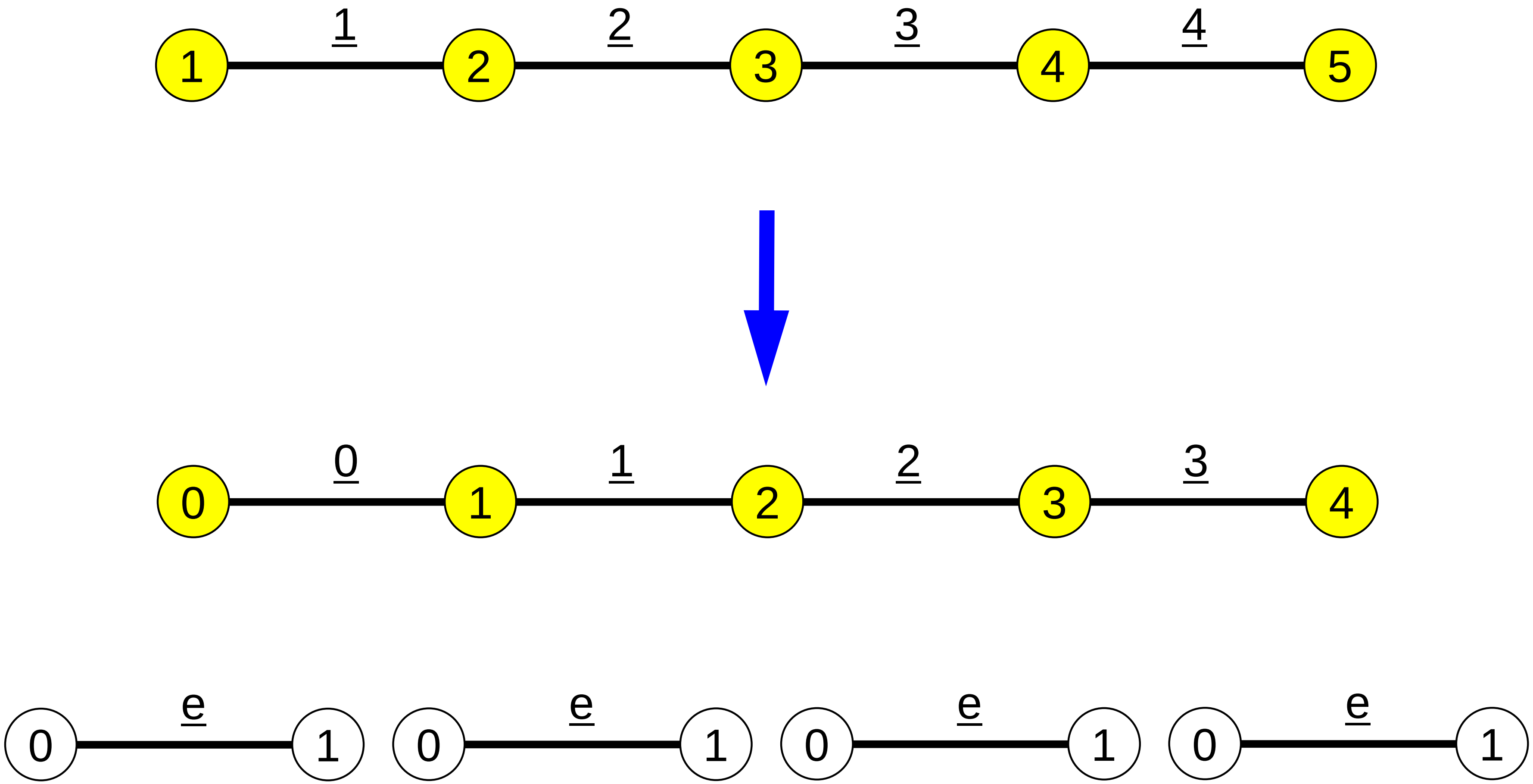
$$\begin{bmatrix} k_{11}^{(4)} & k_{12}^{(4)} \\ k_{21}^{(4)} & k_{22}^{(4)} \end{bmatrix}\begin{Bmatrix} \phi_1^{(4)} \\ \phi_2^{(4)} \end{Bmatrix}=\begin{Bmatrix} f_1^{(4)} \\ f_2^{(4)} \end{Bmatrix}$$

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

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

Number of non-zero off-diag. components is 2 for each node. This number is 1 at boundary nodes).

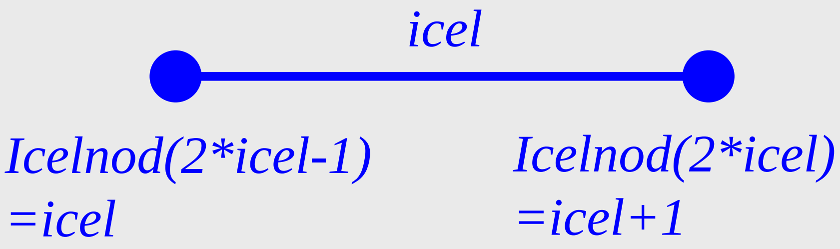
Attention: In C program, node and element ID’s start from 0.



Program: 1d.f (2/6)

Initialization, Allocation of Array

```
!C
!C +-------+
!C | INIT. |
!C +-------+
!C===
      open  (11, file='input.dat', status='unknown')
       read (11,*) NE
       read (11,*) dX, QV, AREA, COND
       read (11,*) ITERmax
       read (11,*) EPS
      close (11)


      N= NE + 1
      allocate (PHI(N), DIAG(N), AMAT(2*N-2), RHS(N))
      allocate (ICELNOD(2*NE), X(N))
      allocate (INDEX(0:N), ITEM(2*N-2), W(N,4))

      PHI = 0.d0
      AMAT= 0.d0
      DIAG= 0.d0
       RHS= 0.d0
         X= 0.d0
```



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

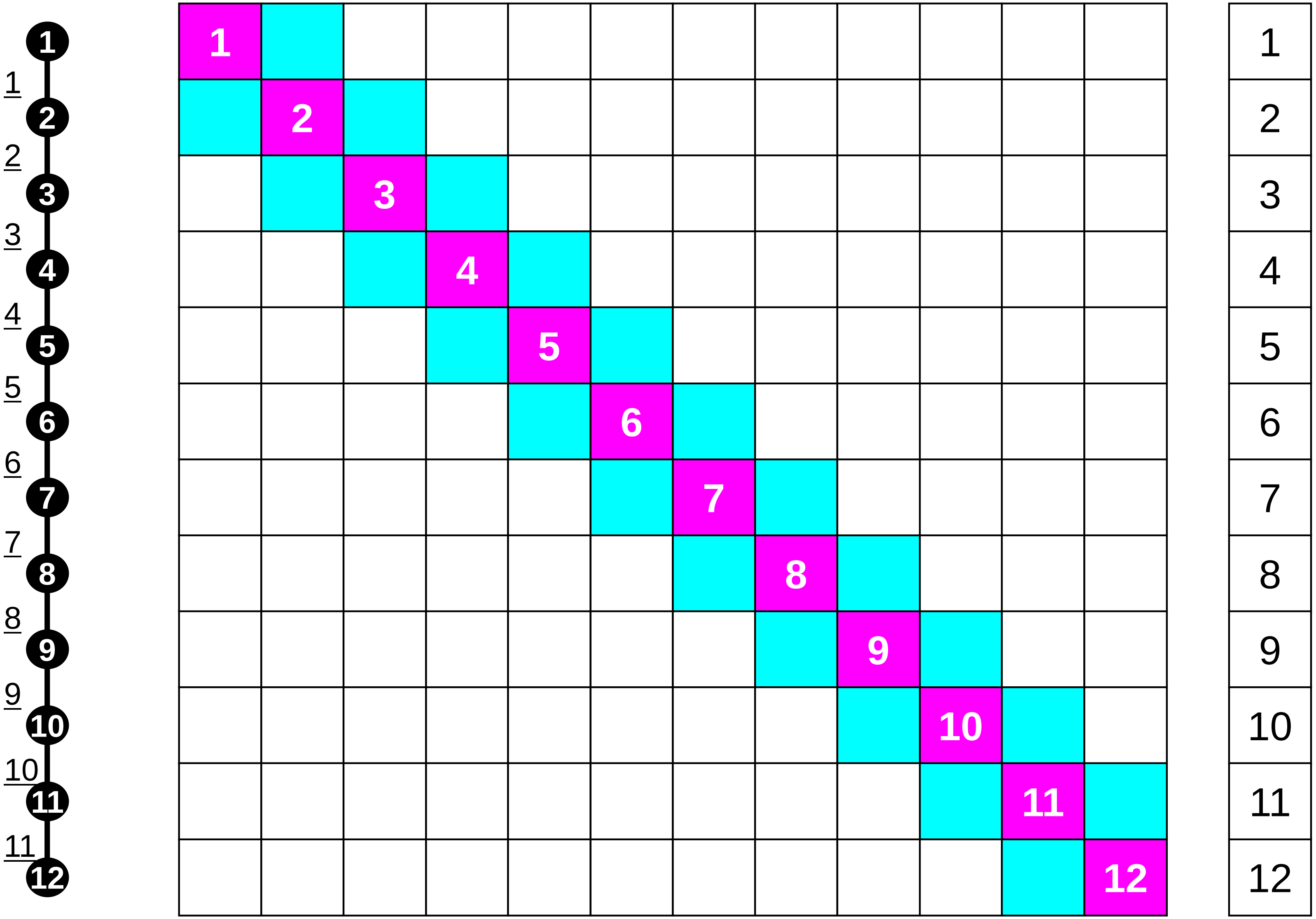
Number of non-zero off-diag. components is 2 for each node. This number is 1 at boundary nodes).

Total Number of Non-Zero Off-Diag. Components:

2*(N-2)+1+1= 2*N-2

1D FEM: 12 nodes/11 elem's

三重対角行列：Tri-Diagonal Matrix



Program: 1d.f (3/6)

Initialization, Allocation of Arrays (cont.)

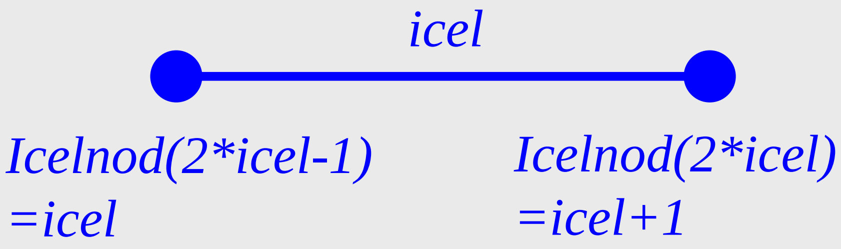
```
      do i= 1, N
        X(i)= dfloat(i-1)*dX
      enddo

      do icel= 1, NE
        ICELNOD(2*icel-1)= icel
        ICELNOD(2*icel  )= icel + 1
      enddo

      KMAT(1,1)= +1.d0
      KMAT(1,2)= -1.d0
      KMAT(2,1)= -1.d0
      KMAT(2,2)= +1.d0
```

X: X-coordinate component of each node

Program: 1d.f (3/6)

Initialization, Allocation of Arrays (cont.)

```
      do i= 1, N
        X(i)= dfloat(i-1)*dX
      enddo

      do icel= 1, NE
        ICELNOD(2*icel-1)= icel
        ICELNOD(2*icel  )= icel + 1
      enddo

      KMAT(1,1)= +1.d0
      KMAT(1,2)= -1.d0
      KMAT(2,1)= -1.d0
      KMAT(2,2)= +1.d0
```



Program: 1d.f (3/6)

Initialization, Allocation of Arrays (cont.)

```
      do i= 1, N
        X(i)= dfloat(i-1)*dX
      enddo

      do icel= 1, NE
        ICELNOD(2*icel-1)= icel
        ICELNOD(2*icel  )= icel + 1
      enddo

      KMAT(1,1)= +1.d0
      KMAT(1,2)= -1.d0
      KMAT(2,1)= -1.d0
      KMAT(2,2)= +1.d0
```

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

Program: 1d.f (4/6)

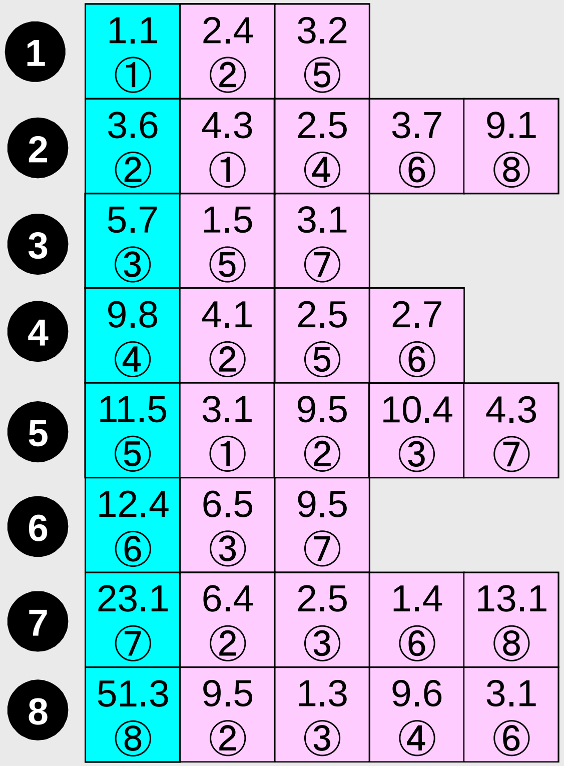
Global Matrix: Column ID for Non-Zero Off-Diag’s

```
!C
!C +----------------+
!C | CONNECTIVITY |
!C +----------------+
!C===
      INDEX   = 2

      INDEX(0)= 0
      INDEX(1)= 1
      INDEX(N)= 1

      do i= 1, N
        INDEX(i)= INDEX(i) + INDEX(i-1)
      enddo

      NPLU= INDEX(N)

      do i= 1, N
        jS= INDEX(i-1)
        if (i.eq.1) then
          ITEM(jS+1)= i+1
         else if
     &      (i.eq.N) then
          ITEM(jS+1)= i-1
         else
          ITEM(jS+1)= i-1
          ITEM(jS+2)= i+1
        endif
      enddo
!C===
```

Number of non-zero off-diag. components is 2 for each node. This number is 1 at boundary nodes).

Total Number of Non-Zero Off-Diag. Components:

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



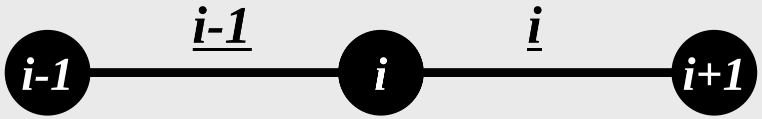
						# Non-Zero Off-Diag.	index(0)= 0
1	1.1 ①	2.4 ②	3.2 ⑤			2	index(1)= 2
2	3.6 ②	4.3 ①	2.5 ④	3.7 ⑥	9.1 ⑧	4	index(2)= 6
3	5.7 ③	1.5 ⑤	3.1 ⑦			2	index(3)= 8
4	9.8 ④	4.1 ②	2.5 ⑤	2.7 ⑥		3	index(4)= 11
5	11.5 ⑤	3.1 ①	9.5 ②	10.4 ③	4.3 ⑦	4	index(5)= 15
6	12.4 ⑥	6.5 ③	9.5 ⑦			2	index(6)= 17
7	23.1 ⑦	6.4 ②	2.5 ③	1.4 ⑥	13.1 ⑧	4	index(7)= 21
8	51.3 ⑧	9.5 ②	1.3 ③	9.6 ④	3.1 ⑥	4	index(8)= 25

index(i-1)+1th~index(i) th
Non-Zero Off-Diag. Components corresponding to *i*-th row

Program: 1d.f (4/6)

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

```
!C
!C +--------------+
!C | CONNECTIVITY |
!C +--------------+
!C===
      INDEX   = 2

      INDEX(0)= 0
      INDEX(1)= 1
      INDEX(N)= 1

      do i= 1, N
        INDEX(i)= INDEX(i) + INDEX(i-1)
      enddo

      NPLU= INDEX(N)

      do i= 1, N
        jS= INDEX(i-1)
        if (i.eq.1) then
          ITEM(jS+1)= i+1
         else if
     &      (i.eq.N) then
          ITEM(jS+1)= i-1
         else
          ITEM(jS+1)= i-1
          ITEM(jS+2)= i+1
        endif
      enddo
!C===
```



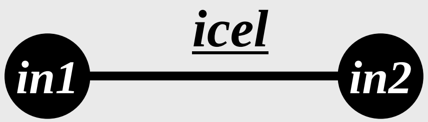
						# Non-Zero Off-Diag.	`index(0)=  0`
1	1.1 ①	2.4 ②	3.2 ⑤			2	`index(1)=  2`
2	3.6 ②	4.3 ①	2.5 ④	3.7 ⑥	9.1 ⑧	4	`index(2)=  6`
3	5.7 ③	1.5 ⑤	3.1 ⑦			2	`index(3)=  8`
4	9.8 ④	4.1 ②	2.5 ⑤	2.7 ⑥		3	`index(4)= 11`
5	11.5 ⑤	3.1 ①	9.5 ②	10.4 ③	4.3 ⑦	4	`index(5)= 15`
6	12.4 ⑥	6.5 ③	9.5 ⑦			2	`index(6)= 17`
7	23.1 ⑦	6.4 ②	2.5 ③	1.4 ⑥	13.1 ⑧	4	`index(7)= 21`
8	51.3 ⑧	9.5 ②	1.3 ③	9.6 ④	3.1 ⑥	4	`index(8)= 25`

index(i-1)+1th~index(i) th
Non-Zero Off-Diag. Components corresponding to *i*-th row

Program: 1d.f (5/6)

Element Matrix ~ Global Matrix

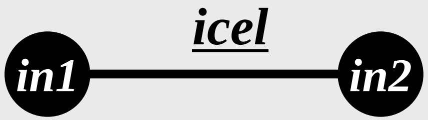
```
!C +-----------------+
!C | MATRIX ASSEMBLE |
!C +-----------------+
!C===
      do icel= 1, NE
        in1= ICELNOD(2*icel-1)
        in2= ICELNOD(2*icel  )
        X1 = X(in1)
        X2 = X(in2)
        DL = dabs(X2-X1)

        cK= AREA*COND/DL
        EMAT(1,1)= Ck*KMAT(1,1)
        EMAT(1,2)= Ck*KMAT(1,2)
        EMAT(2,1)= Ck*KMAT(2,1)
        EMAT(2,2)= Ck*KMAT(2,2)

        DIAG(in1)= DIAG(in1) + EMAT(1,1)
        DIAG(in2)= DIAG(in2) + EMAT(2,2)

        if (icel.eq.1) then
          k1= INDEX(in1-1) + 1
         else
          k1= INDEX(in1-1) + 2
        endif
        k2= INDEX(in2-1) + 1

        AMAT(k1)= AMAT(k1) + EMAT(1,2)
        AMAT(k2)= AMAT(k2) + EMAT(2,1)

        QN= 0.50d0*QV*AREA*DL
        RHS(in1)= RHS(in1) + QN
        RHS(in2)= RHS(in2) + QN
      enddo
!C===
```



Program: 1d.f (5/6)

Element Matrix ~ Global Matrix

```
!C +-------------------+
!C | MATRIX ASSEMBLE |
!C +-------------------+
!C===
      do icel= 1, NE
        in1= ICELNOD(2*icel-1)
        in2= ICELNOD(2*icel  )
        X1 = X(in1)
        X2 = X(in2)
        DL = dabs(X2-X1)

        cK= AREA*COND/DL
        EMAT(1,1)= Ck*KMAT(1,1)
        EMAT(1,2)= Ck*KMAT(1,2)
        EMAT(2,1)= Ck*KMAT(2,1)
        EMAT(2,2)= Ck*KMAT(2,2)

        DIAG(in1)= DIAG(in1) + EMAT(1,1)
        DIAG(in2)= DIAG(in2) + EMAT(2,2)

        if (icel.eq.1) then
          k1= INDEX(in1-1) + 1
         else
          k1= INDEX(in1-1) + 2
        endif
        k2= INDEX(in2-1) + 1

        AMAT(k1)= AMAT(k1) + EMAT(1,2)
        AMAT(k2)= AMAT(k2) + EMAT(2,1)

        QN= 0.50d0*QV*AREA*DL
        RHS(in1)= RHS(in1) + QN
        RHS(in2)= RHS(in2) + QN
      enddo
!C===
```

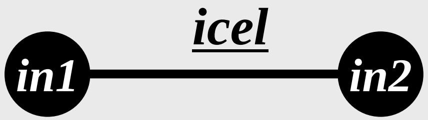


$$[Emat]=[k]^{(e)}=\frac{\lambda A}{L}\begin{bmatrix}+1 & -1\\ -1 & +1\end{bmatrix}=\frac{\lambda A}{L}[Kmat]$$

Program: 1d.f (5/6)

Element Matrix ~ Global Matrix

```
!C +-----------------+
!C | MATRIX ASSEMBLE |
!C +-----------------+
!C===
      do icel= 1, NE
        in1= ICELNOD(2*icel-1)
        in2= ICELNOD(2*icel  )
        X1 = X(in1)
        X2 = X(in2)
        DL = dabs(X2-X1)

        cK= AREA*COND/DL
        EMAT(1,1)= Ck*KMAT(1,1)
        EMAT(1,2)= Ck*KMAT(1,2)
        EMAT(2,1)= Ck*KMAT(2,1)
        EMAT(2,2)= Ck*KMAT(2,2)

        DIAG(in1)= DIAG(in1) + EMAT(1,1)
        DIAG(in2)= DIAG(in2) + EMAT(2,2)

        if (icel.eq.1) then
          k1= INDEX(in1-1) + 1
         else
          k1= INDEX(in1-1) + 2
        endif
        k2= INDEX(in2-1) + 1

        AMAT(k1)= AMAT(k1) + EMAT(1,2)
        AMAT(k2)= AMAT(k2) + EMAT(2,1)

        QN= 0.50d0*QV*AREA*DL
        RHS(in1)= RHS(in1) + QN
        RHS(in2)= RHS(in2) + QN
      enddo
!C===
```

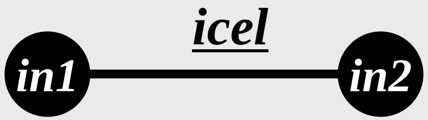


$$[Emat]=[k]^{(e)}=\frac{EA}{L}\begin{bmatrix}\enclose{circle}{+1} & -1\\ -1 & \enclose{circle}{+1}\end{bmatrix}$$

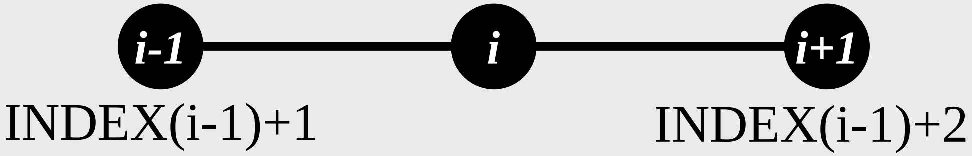
Program: 1d.f (5/6)

Element Matrix ~ Global Matrix

```
!C +-------------------+
!C | MATRIX ASSEMBLE |
!C +-------------------+
!C===
      do icel= 1, NE
        in1= ICELNOD(2*icel-1)
        in2= ICELNOD(2*icel  )
        X1 = X(in1)
        X2 = X(in2)
        DL = dabs(X2-X1)

        cK= AREA*COND/DL
        EMAT(1,1)= Ck*KMAT(1,1)
        EMAT(1,2)= Ck*KMAT(1,2)
        EMAT(2,1)= Ck*KMAT(2,1)
        EMAT(2,2)= Ck*KMAT(2,2)

        DIAG(in1)= DIAG(in1) + EMAT(1,1)
        DIAG(in2)= DIAG(in2) + EMAT(2,2)

        if (icel.eq.1) then
          k1= INDEX(in1-1) + 1
         else
          k1= INDEX(in1-1) + 2
        endif
        k2= INDEX(in2-1) + 1

        AMAT(k1)= AMAT(k1) + EMAT(1,2)
        AMAT(k2)= AMAT(k2) + EMAT(2,1)

        QN= 0.50d0*QV*AREA*DL
        RHS(in1)= RHS(in1) + QN
        RHS(in2)= RHS(in2) + QN
      enddo
!C===
```



Non-zero Off-Diag. at *i*-th row:
`Index(i-1)+1, Index(i-1)+2`



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

k1

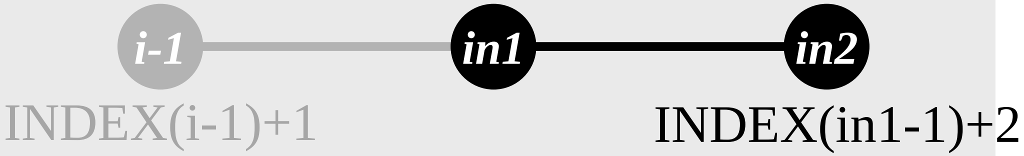
k2

General Elements: k1

“in2” as a off-diag. component of “in1”

```
!C +-----------------+
!C | MATRIX ASSEMBLE |
!C +-----------------+
!C===
      do icel= 1, NE
        in1= ICELNOD(2*icel-1)
        in2= ICELNOD(2*icel  )
        X1 = X(in1)
        X2 = X(in2)
        DL = dabs(X2-X1)

        cK= AREA*COND/DL
        EMAT(1,1)= Ck*KMAT(1,1)
        EMAT(1,2)= Ck*KMAT(1,2)
        EMAT(2,1)= Ck*KMAT(2,1)
        EMAT(2,2)= Ck*KMAT(2,2)

        DIAG(in1)= DIAG(in1) + EMAT(1,1)
        DIAG(in2)= DIAG(in2) + EMAT(2,2)

        if (icel.eq.1) then
          k1= INDEX(in1-1) + 1
         else
          k1= INDEX(in1-1) + 2
        endif
        k2= INDEX(in2-1) + 1

        AMAT(k1)= AMAT(k1) + EMAT(1,2)
        AMAT(k2)= AMAT(k2) + EMAT(2,1)

        QN= 0.50d0*QV*AREA*DL
        RHS(in1)= RHS(in1) + QN
        RHS(in2)= RHS(in2) + QN
      enddo
!C===
```

Non-zero Off-Diag. at *i*-th row:
`Index(i-1)+1, Index(i-1)+2`



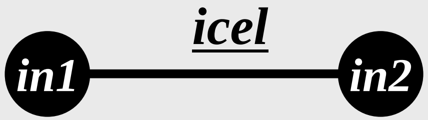
$$[Emat]=[k]^{(e)}=\frac{\lambda A}{L}\begin{bmatrix}+1 & \boxed{-1}\\ -1 & +1\end{bmatrix}$$

k1

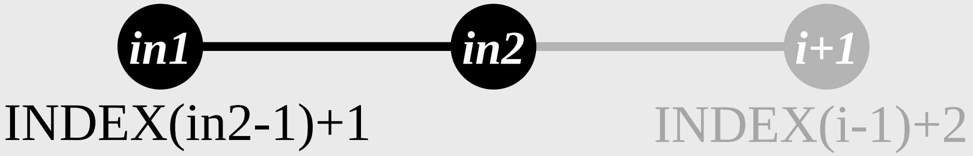
General Elements: k2

“in1” as a off-diag. component of “in2”

```
!C +-----------------+
!C | MATRIX ASSEMBLE |
!C +-----------------+
!C===
      do icel= 1, NE
        in1= ICELNOD(2*icel-1)
        in2= ICELNOD(2*icel  )
        X1 = X(in1)
        X2 = X(in2)
        DL = dabs(X2-X1)

        cK= AREA*COND/DL
        EMAT(1,1)= Ck*KMAT(1,1)
        EMAT(1,2)= Ck*KMAT(1,2)
        EMAT(2,1)= Ck*KMAT(2,1)
        EMAT(2,2)= Ck*KMAT(2,2)

        DIAG(in1)= DIAG(in1) + EMAT(1,1)
        DIAG(in2)= DIAG(in2) + EMAT(2,2)

        if (icel.eq.1) then
          k1= INDEX(in1-1) + 1
         else
          k1= INDEX(in1-1) + 2
        endif
        k2= INDEX(in2-1) + 1

        AMAT(k1)= AMAT(k1) + EMAT(1,2)
        AMAT(k2)= AMAT(k2) + EMAT(2,1)

        QN= 0.50d0*QV*AREA*DL
        RHS(in1)= RHS(in1) + QN
        RHS(in2)= RHS(in2) + QN
      enddo
!C===
```



Non-zero Off-Diag. at *i*-th row:
`Index(i-1)+1, Index(i-1)+2`



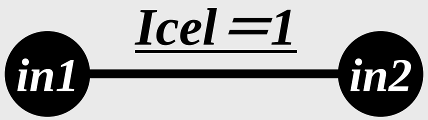
$$[Emat]=[k]^{(e)}=\frac{\lambda A}{L}\begin{bmatrix}+1 & -1\\ -1 & +1\end{bmatrix}$$

k2

0-th Element: k1

"in2" as a off-diag. component of "in1"

```
!C +-----------------+
!C | MATRIX ASSEMBLE |
!C +-----------------+
!C===
      do icel= 1, NE
        in1= ICELNOD(2*icel-1)
        in2= ICELNOD(2*icel  )
        X1 = X(in1)
        X2 = X(in2)
        DL = dabs(X2-X1)

        cK= AREA*COND/DL
        EMAT(1,1)= Ck*KMAT(1,1)
        EMAT(1,2)= Ck*KMAT(1,2)
        EMAT(2,1)= Ck*KMAT(2,1)
        EMAT(2,2)= Ck*KMAT(2,2)

        DIAG(in1)= DIAG(in1) + EMAT(1,1)
        DIAG(in2)= DIAG(in2) + EMAT(2,2)

        if (icel.eq.1) then
          k1= INDEX(in1-1) + 1
         else
          k1= INDEX(in1-1) + 2
        endif
        k2= INDEX(in2-1) + 1

        AMAT(k1)= AMAT(k1) + EMAT(1,2)
        AMAT(k2)= AMAT(k2) + EMAT(2,1)

        QN= 0.50d0*QV*AREA*DL
        RHS(in1)= RHS(in1) + QN
        RHS(in2)= RHS(in2) + QN
      enddo
!C===
```



Non-zero Off-Diag. at *i*-th row:
Index(i-1)+1 only

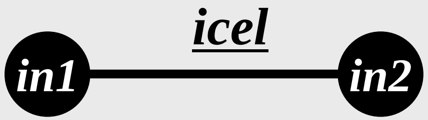
$$[Emat]=[k]^{(e)}=\frac{\lambda A}{L}\begin{bmatrix}+1 & \boxed{-1}\\ -1 & +1\end{bmatrix}$$

k1

Program: 1d.f (5/6)

RHS: Heat Generation Term

```
!C +-------------------+
!C | MATRIX ASSEMBLE |
!C +-------------------+
!C===
      do icel= 1, NE
        in1= ICELNOD(2*icel-1)
        in2= ICELNOD(2*icel  )
        X1 = X(in1)
        X2 = X(in2)
        DL = dabs(X2-X1)

        cK= AREA*COND/DL
        EMAT(1,1)= Ck*KMAT(1,1)
        EMAT(1,2)= Ck*KMAT(1,2)
        EMAT(2,1)= Ck*KMAT(2,1)
        EMAT(2,2)= Ck*KMAT(2,2)

        DIAG(in1)= DIAG(in1) + EMAT(1,1)
        DIAG(in2)= DIAG(in2) + EMAT(2,2)

        if (icel.eq.1) then
          k1= INDEX(in1-1) + 1
         else
          k1= INDEX(in1-1) + 2
        endif
        k2= INDEX(in2-1) + 1

        AMAT(k1)= AMAT(k1) + EMAT(1,2)
        AMAT(k2)= AMAT(k2) + EMAT(2,1)

        QN= 0.50d0*QV*AREA*DL
        RHS(in1)= RHS(in1) + QN
        RHS(in2)= RHS(in2) + QN
      enddo
!C===
```



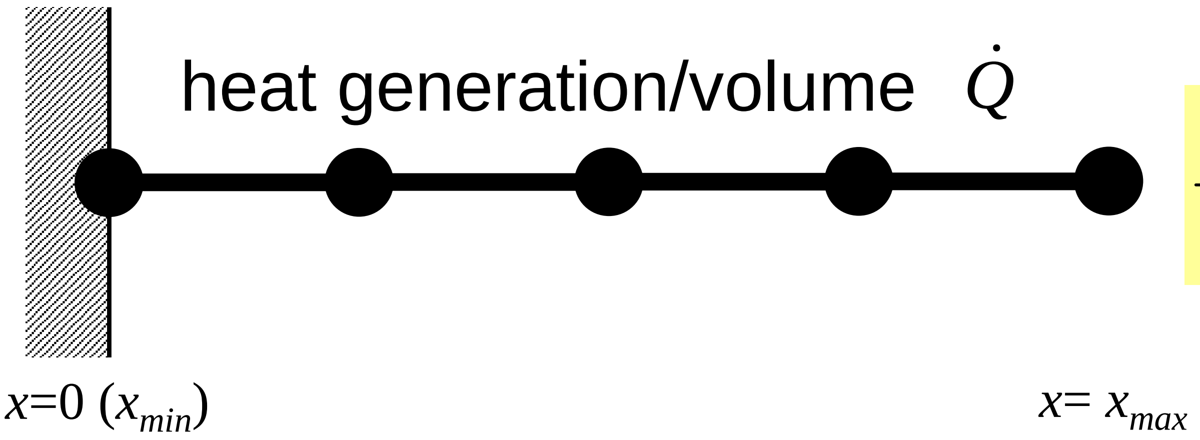
$$\int_V \dot{Q}[N]^T dV = \dot{Q}A\int_0^L \begin{bmatrix} 1-x/L \\ x/L \end{bmatrix} dx = \frac{\dot{Q}AL}{2}\begin{Bmatrix} 1 \\ 1 \end{Bmatrix}$$

Program: 1d.f (6/6)

Dirichlet B.C. @ X=0

```
!C
!C +---------------------+
!C | BOUNDARY CONDITIONS |
!C +---------------------+
!C===

!C
!C-- X=Xmin
      i= 1
      jS= INDEX(i-1)

      AMAT(jS+1)= 0.d0
      DIAG(i)= 1.d0
      RHS (i)= 0.d0

      do k= 1, NPLU
        if (ITEM(k).eq.1) AMAT(k)= 0.d0
      enddo
!C===
```

1D Steady State Heat Conduction

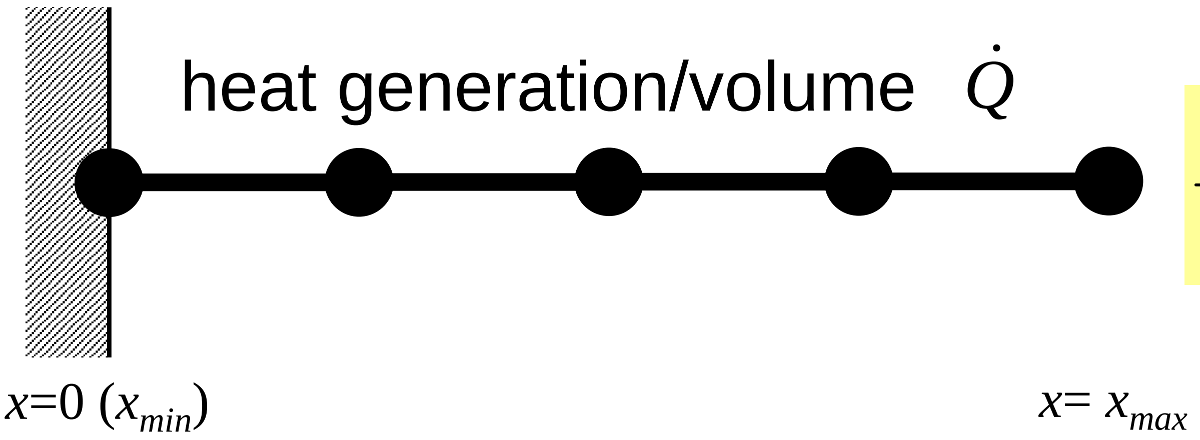


$$\frac{\partial}{\partial x}\left(\lambda \frac{\partial T}{\partial x}\right)+\dot{Q}=0$$

- Uniform: Sectional Area: A, Thermal Conductivity: λ
- Heat Generation Rate/Volume/Time [$QL^{-3}T^{-1}$] $\dot{Q}$
- Boundary Conditions
  - $x=0$: $T=0$ (Fixed Temperature)
  - $x=x_{max}$: $\frac{\partial T}{\partial x}=0$ (Insulated)

(Linear) Equation at x=0

$T_1 = 0$ (or $T_0 = 0$)



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

- Uniform: Sectional Area: A, Thermal Conductivity: λ
- Heat Generation Rate/Volume/Time [$QL^{-3}T^{-1}$] $\dot{Q}$
- Boundary Conditions
  - $x=0$: $T=0$ (Fixed Temperature)
  - $x=x_{max}$: $\dfrac{\partial T}{\partial x}=0$ (Insulated)

Program: 1d.f (6/6)

Dirichlet B.C. @ X=0

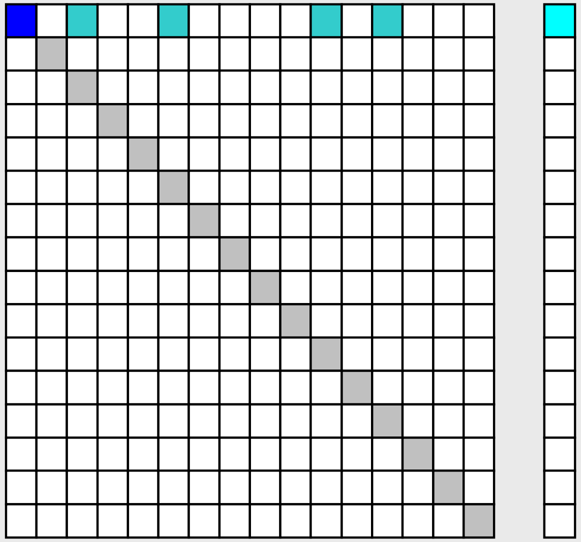
```
!C
!C +---------------------+
!C | BOUNDARY CONDITIONS |
!C +---------------------+
!C===

!C
!C-- X=Xmin
      i= 1
      jS= INDEX(i-1)

      AMAT(jS+1)= 0.d0
      DIAG(i)= 1.d0
      RHS (i)= 0.d0

      do k= 1, NPLU
        if (ITEM(k).eq.1) AMAT(k)= 0.d0
      enddo
!C===
```

$T_1=0$

Diagonal Component=1

RHS=0

Off-Diagonal Components= 0.



Program: 1d.f (6/6)

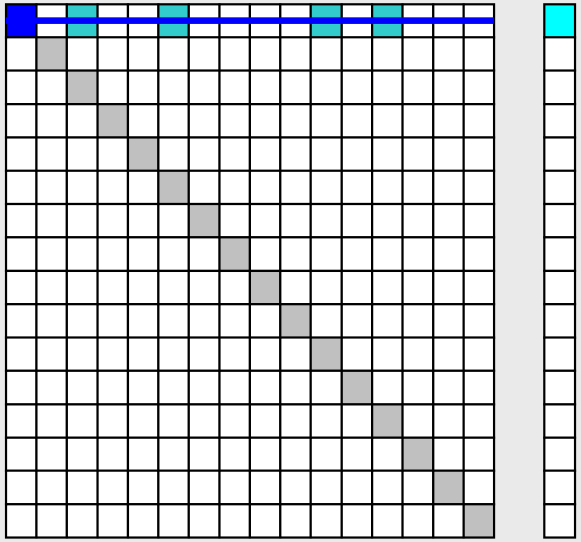
Dirichlet B.C. @ X=0

```
!C
!C +---------------------+
!C | BOUNDARY CONDITIONS |
!C +---------------------+
!C===

!C
!C-- X=Xmin
      i= 1
      jS= INDEX(i-1)

      AMAT(jS+1)= 0.d0
      DIAG(i)= 1.d0
      RHS (i)= 0.d0

      do k= 1, NPLU
        if (ITEM(k).eq.1) AMAT(k)= 0.d0
      enddo
!C===
```

$T_1=0$

Diagonal Component=1
RHS=0
Off-Diagonal Components= 0.

Erase !



Program: 1d.f (6/6)

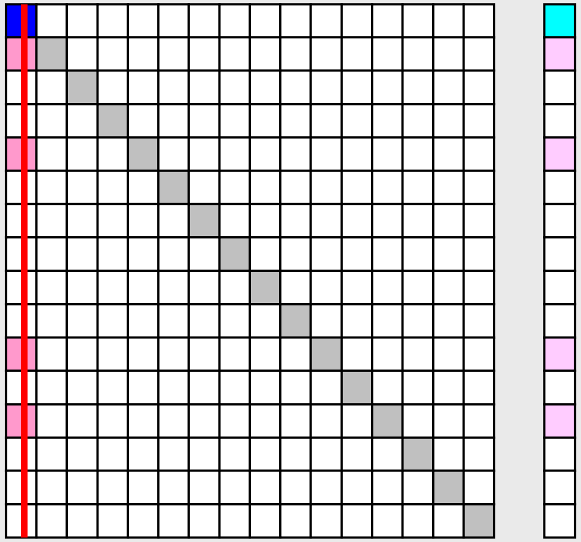
Dirichlet B.C. @ X=0

```
!C
!C +---------------------+
!C | BOUNDARY CONDITIONS |
!C +---------------------+
!C===

!C
!C-- X=Xmin
      i= 1
      jS= INDEX(i-1)

      AMAT(jS+1)= 0.d0
      DIAG(i)= 1.d0
      RHS (i)= 0.d0

      do k= 1, NPLU
        if (ITEM(k).eq.1) AMAT(k)= 0.d0
      enddo
!C===
```

$T_1=0$

Diagonal Component=1
RHS=0
Off-Diagonal Components= 0.

Elimination and Erase



Column components of boundary nodes (Dirichlet B.C.) are moved to RHS and eliminated for keeping symmetrical feature of the matrix (in this case just erase off-diagonal components)

if $T_1 \neq 0$

```
!C
!C +----------------------+
!C | BOUNDARY CONDITIONS |
!C +----------------------+
!C===

!C
!C-- X=Xmin
      i= 1
      jS= INDEX(i-1)

      AMAT(jS+1)= 0.d0
      DIAG(i)= 1.d0
      RHS (i)= PHImin

      do i= 1, N
        do k= INDEX(i-1)+1, INDEX(i)
          if (ITEM(k).eq.1) then
            RHS (i)= RHS(i) - AMAT(k)*PHImin
            AMAT(k)= 0.d0
          endif
        enddo
      enddo
!C===
```

Column components of boundary nodes (Dirichlet B.C.) are moved to RHS and eliminated for keeping symmetrical feature of the matrix.

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

if $T_1 \neq 0$

```
!C
!C +----------------------+
!C | BOUNDARY CONDITIONS |
!C +----------------------+
!C===

!C
!C-- X=Xmin
      i= 1
      jS= INDEX(i-1)

      AMAT(jS+1)= 0.d0
      DIAG(i)= 1.d0
      RHS (i)= PHImin

      do i= 1, N
        do k= INDEX(i-1)+1, INDEX(i)
          if (ITEM(k).eq.1) then
            RHS (i)= RHS(i) - AMAT(k)*PHImin
            AMAT(k)= 0.d0
          endif
        enddo
      enddo
!C===
```

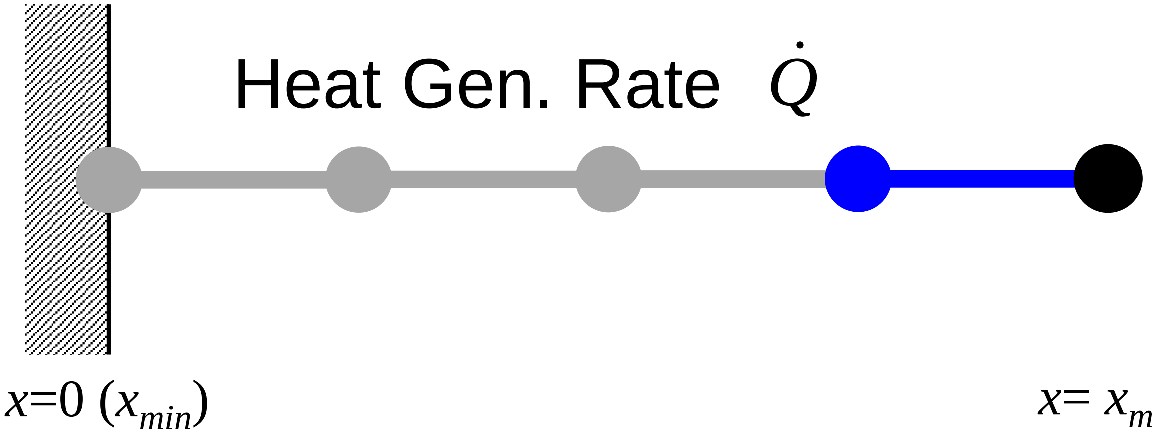
$$Diag_j \phi_j + \sum_{k=Index[j],k\neq k_s}^{Index[j+1]-1} Amat_k \, \phi_{Item[k]}$$

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

$$= Rhs_j - Amat_{k_s} \phi_{\min} \quad where\ Item[k_s]=1$$

Column components of boundary nodes (Dirichlet B.C.) are moved to RHS and eliminated for keeping symmetrical feature of the matrix.

Secondary B.C. (Insulated)



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

$$T=0@x=0$$

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

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

Surface Flux

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

According to insulated B.C., $\bar{q}=0$
is satisfied. No contribution by this term.
Insulated B.C. is automatically satisfied
without explicit operations
-> Natural B.C.

Preconditioned CG Solver

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

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

Diagonal Scaling, Point-Jacobi

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

- **`solve [M]z(i-1)= r(i-1)`** is very easy.
- Provides fast convergence for simple problems.
- 1d.f, 1d.c

CG Solver (1/6)

```
!C
!C +---------------+
!C | CG iterations |
!C +---------------+
!C===
      R = 1
      Z = 2
      Q = 2
      P = 3
      DD= 4

      do i= 1, N
        W(i,DD)= 1.0D0 / DIAG(i)
      enddo
```

W(i,1)= W(i,R) ⇒ {r}
W(i,2)= W(i,Z) ⇒ {z}
W(i,2)= W(i,Q) ⇒ {q}
W(i,3)= W(i,P) ⇒ {p}
W(i,4)= W(i,DD) ⇒ 1/{D}

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

CG Solver (1/6)

```
!C
!C +---------------+
!C | CG iterations |
!C +---------------+
!C===
      R = 1
      Z = 2
      Q = 2
      P = 3
      DD= 4

      do i= 1, N
        W(i,DD)= 1.0D0 / DIAG(i)
      enddo
```

Reciprocal numbers（逆数）of diagonal components are stored in `W[DD][i]`. Computational cost for division is usually expensive.

Although it was said (division):(+,-,*) is 10:1 before, the difference is much smaller now. Generally, multiplying is still faster than division.

W(i,1)= W(i,R) ⇒ {r}
W(i,2)= W(i,Z) ⇒ {z}
W(i,2)= W(i,Q) ⇒ {q}
W(i,3)= W(i,P) ⇒ {p}
W(i,4)= W(i,DD) ⇒ 1/{D}

CG Solver (2/6)

```
!C
!C-- {r0}= {b} - [A]{xini} |

!C 初期残差
      do i= 1, N
        W(i,R) = DIAG(i)*PHI(i)
        do j= INDEX(i-1)+1, INDEX(i)
          W(i,R) = W(i,R) + AMAT(j)*PHI(ITEM(j))
        enddo
      enddo

      BNRM2= 0.0D0
      do i= 1, N
        BNRM2 = BNRM2  + RHS(i) **2
        W(i,R)= RHS(i) - W(i,R)
      enddo
```

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

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

BNRM2=$|b|^2$
for convergence criteria
of CG solvers

CG Solver (3/6)

```
      do iter= 1, ITERmax

!C
!C-- {z}= [Minv]{r}

      do i= 1, N
        W(i,Z)= W(i,DD) * W(i,R)
      enddo

!C
!C-- RHO= {r}{z}

      RHO= 0.d0
      do i= 1, N
        RHO= RHO + W(i,R)*W(i,Z)
      enddo
```

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

CG Solver (4/6)

```
!C
!C-- {p} = {z} if      ITER=1
!C   BETA= RHO / RHO1  otherwise

      if ( iter.eq.1 ) then
        do i= 1, N
          W(i,P)= W(i,Z)
        enddo
       else
         BETA= RHO / RHO1
         do i= 1, N
           W(i,P)= W(i,Z) + BETA*W(i,P)
         enddo
      endif

!C
!C-- {q}= [A]{p}

      do i= 1, N
        W(i,Q) = DIAG(i)*W(i,P)
        do j= INDEX(i-1)+1, INDEX(i)
          W(i,Q) = W(i,Q) + AMAT(j)*W(ITEM(j),P)
        enddo
      enddo
```

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

CG Solver (5/6)

```
!C
!C-- ALPHA= RHO / {p}{q}

      C1= 0.d0
      do i= 1, N
        C1= C1 + W(i,P)*W(i,Q)
      enddo
      ALPHA= RHO / C1

!C
!C-- {x}= {x} + ALPHA*{p}
!C   {r}= {r} - ALPHA*{q}

      do i= 1, N
        PHI(i)= PHI(i) + ALPHA * W(i,P)
        W(i,R)= W(i,R) - ALPHA * W(i,Q)
      enddo
```

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

CG Solver (6/6)

```
         DNRM2 = 0.0
         do i= 1, N
           DNRM2= DNRM2 + W(i,R)**2
         enddo

         RESID= dsqrt(DNRM2/BNRM2)

         if ( RESID.le.EPS) goto 900
         RHO1 = RHO     ρi-2


       enddo
 900   continue
```

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

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

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

Control Data **input.dat**

4	**NE** (Number of Elements)
1.0 1.0 1.0 1.0	**Δx** (Length of Each Elem.: **L**) , **Q, A, λ**
100	Number of MAX. Iterations for CG Solver
1.e-8	Convergence Criteria for CG Solver

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

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

Finite Element Procedures

- Initialization
  - Control Data
  - Node, Connectivity of Elements (N: Node#, NE: Elem#)
  - Initialization of Arrays (Global/Element Matrices)
  - Element-Global Matrix Mapping (Index, Item)
- Generation of Matrix
  - Element-by-Element Operations (do icel= 1, NE)
    - Element matrices
    - Accumulation to global matrix
  - Boundary Conditions
- Linear Solver
  - Conjugate Gradient Method

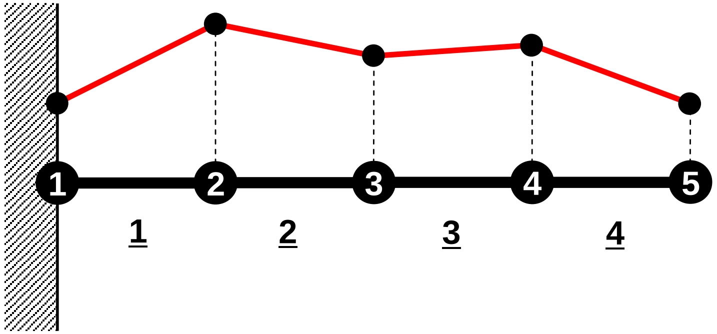
Remedies for Higher Accuracy

- Finer Meshes

```
NE=8, dX=12.5

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

### Temperature
         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 PHI(N)=   1.975734E-01

### Temperature
         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
```



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

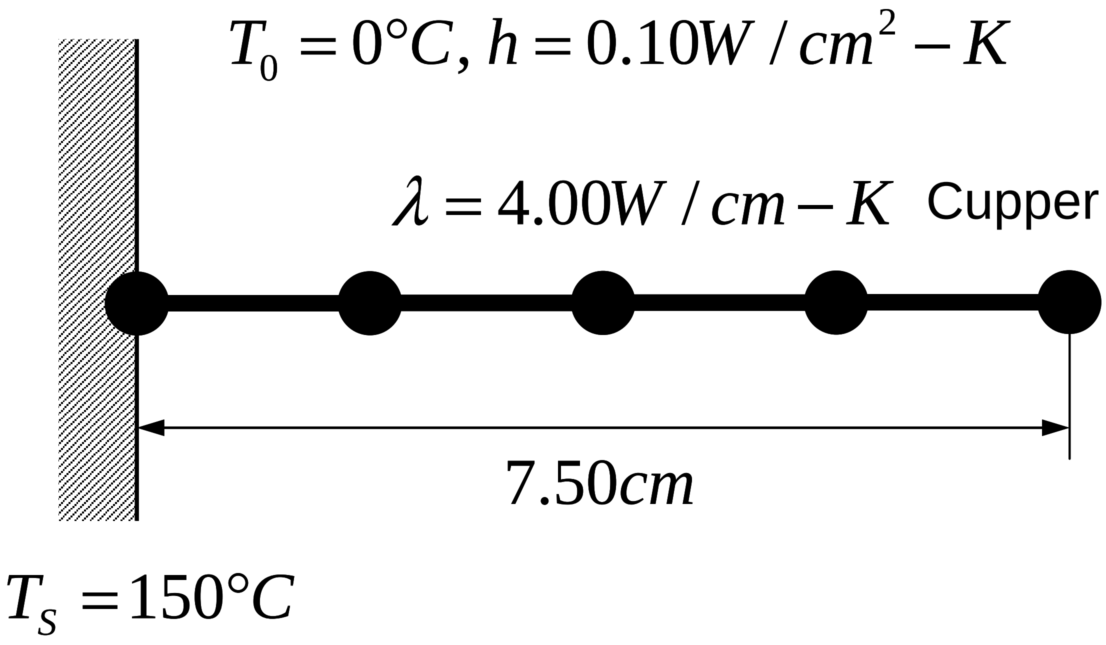
Remedies for Higher Accuracy

- Finer Meshes
- Higher Order Shape/Interpolation Function（高次補間関数・形状関数）
  - Higher-Order Element（高次要素）
  - Linear-Element, 1st-Order Element: Lower Order（低次要素）
- Formulation which assures continuity of n-th order derivatives
  - C^n Continuity（C^n連続性）

Remedies for Higher Accuracy

- Finer Meshes
- Higher Order Shape/Interpolation Function（高次補間関数・形状関数）
  - Higher-Order Element（高次要素）
  - Linear-Element, 1st-Order Element: Lower Order（低次要素）
- Formulation which assures continuity of n-th order derivatives
  - C^n Continuity（C^n連続性）
- Linear Elements
  - Piecewise Linear
  - C^0 Continuity
    - Only dependent variables are continuous at element boundary

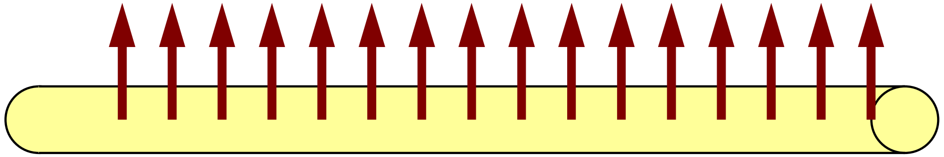
Example: 1D Heat Transfer (1/2)



$T_0 = 0°C, h = 0.10 W/cm^2 - K$

$\lambda = 4.00 W/cm - K$ Cupper

$T_S = 150°C$



Convective Heat Transfer on Cylindrical Surface

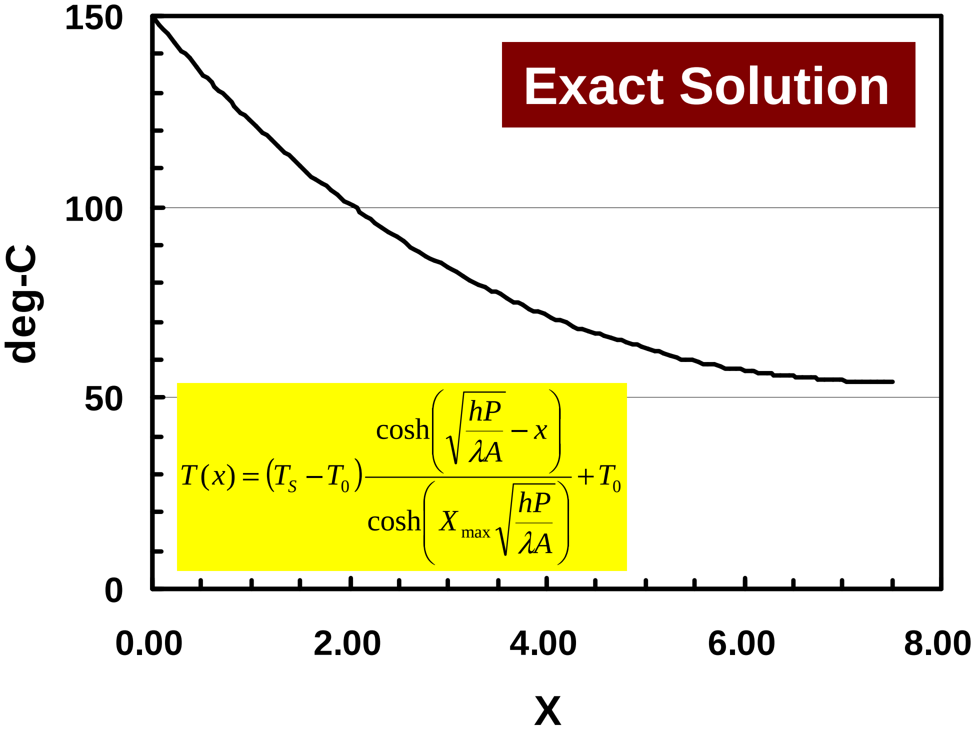
- Temp. Thermal Fins
- Circular Sectional Area, r=1cm
- Boundary Condition
  - x=0 : Fixed Temperature
  - x=7.5 : Insulated
- Convective Heat Transfer on Cylindrical Surface
  - $q= h (T-T_0)$
  - q : Heat Flux
    - Heat Flow/Unit Surface Area/sec.

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
```



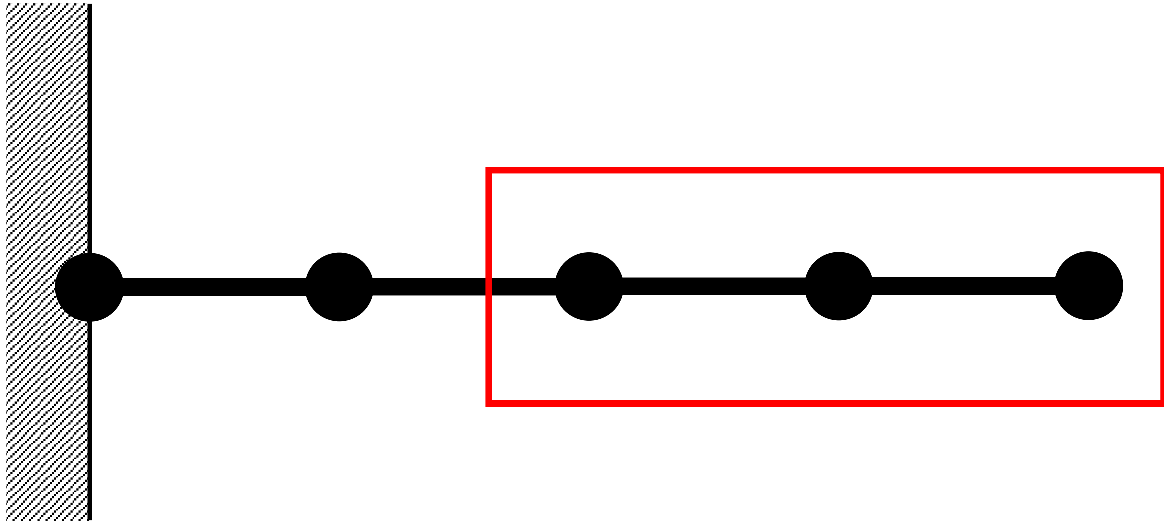
$$T(x) = (T_S - T_0)\frac{\cosh\left(\sqrt{\frac{hP}{\lambda A}} - x\right)}{\cosh\left(X_{\max}\sqrt{\frac{hP}{\lambda A}}\right)} + T_0$$

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

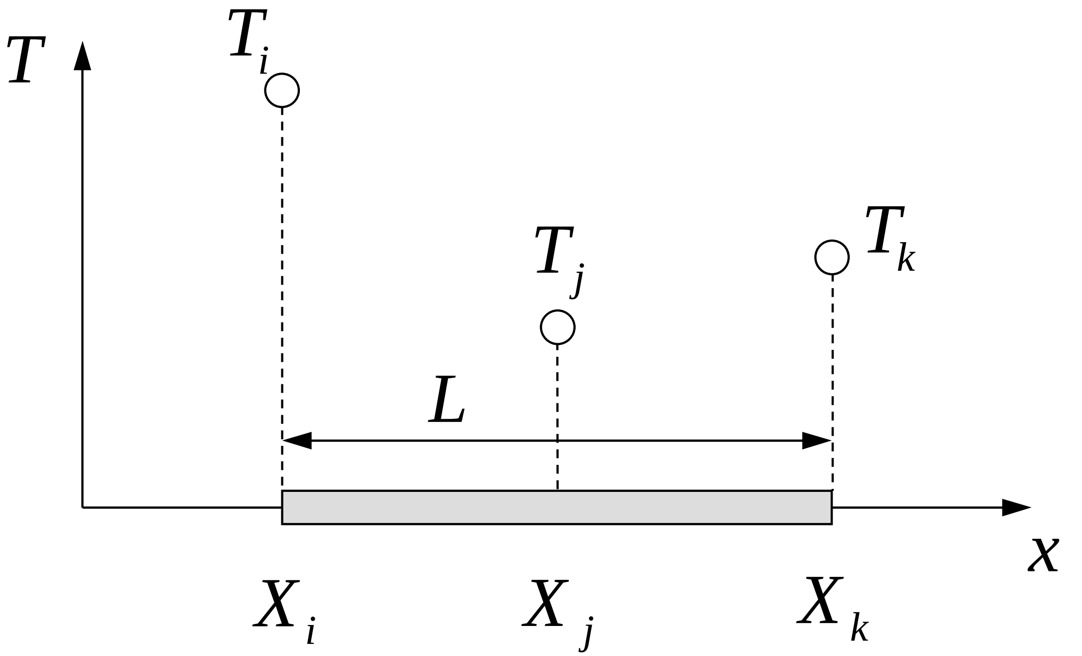
- Length= L
- (i,k): Both Ends
- (j): Intermediate Node
  - ✓ Mid-Point（中間節点）



- Distribution of T in each element:

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

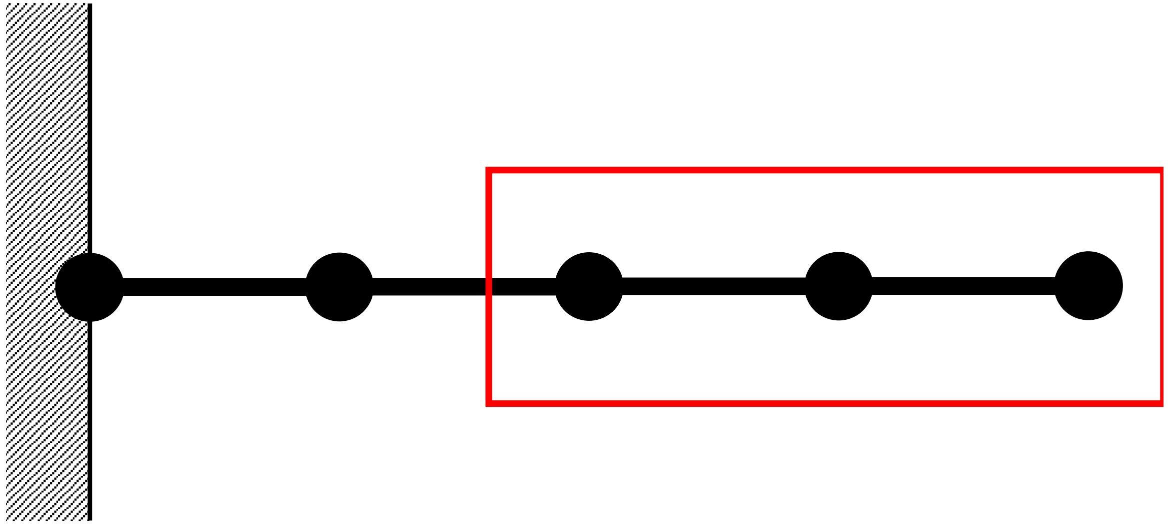
$$\begin{aligned} T_i &= \alpha_1 + X_i \alpha_2 + X_i^{\,2} \alpha_3 \\ T_j &= \alpha_1 + X_j \alpha_2 + X_j^{\,2} \alpha_3 \\ T_k &= \alpha_1 + X_k \alpha_2 + X_k^{\,2} \alpha_3 \end{aligned}$$



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）

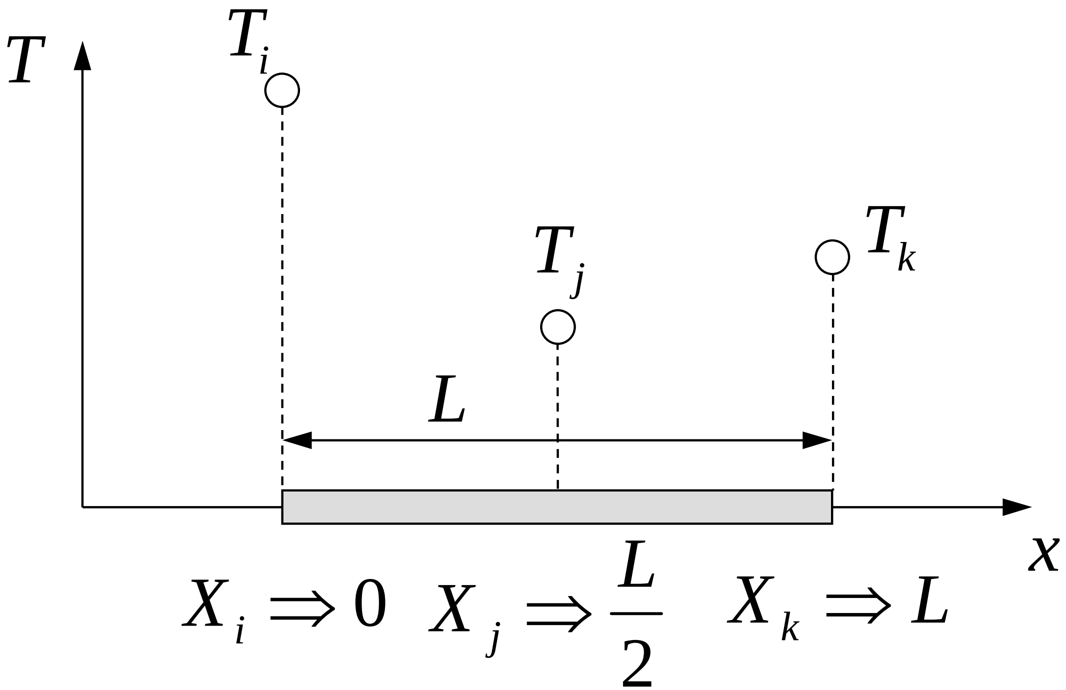


- Distribution of T in each element:

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

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

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

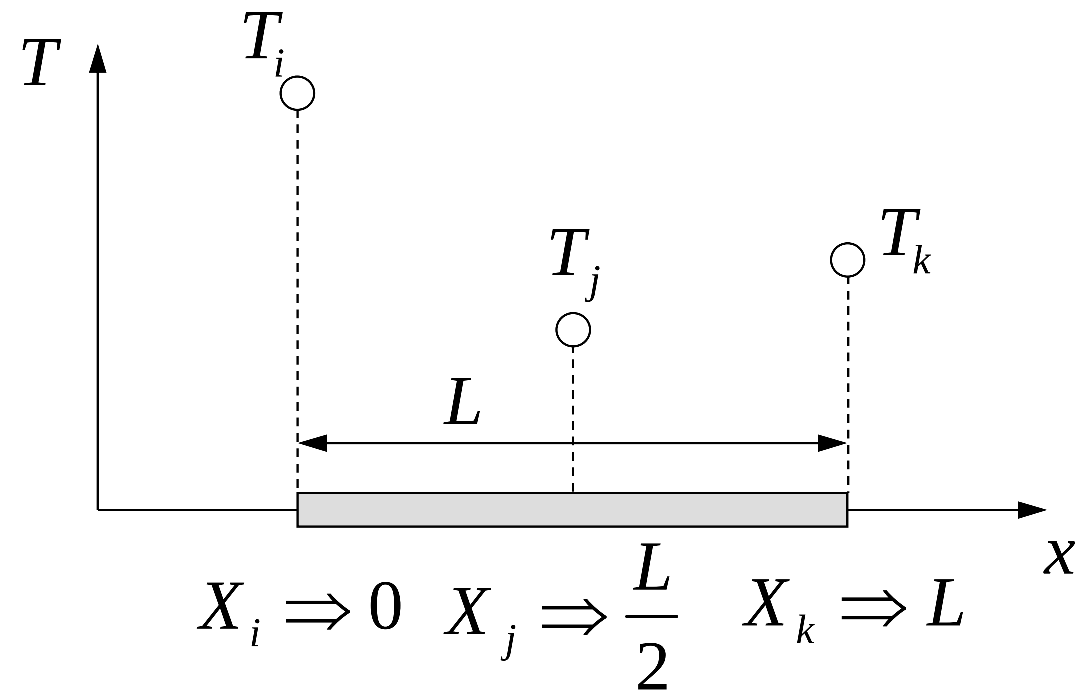


1D Quadratic Element (2/2)

一次元二次要素

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

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

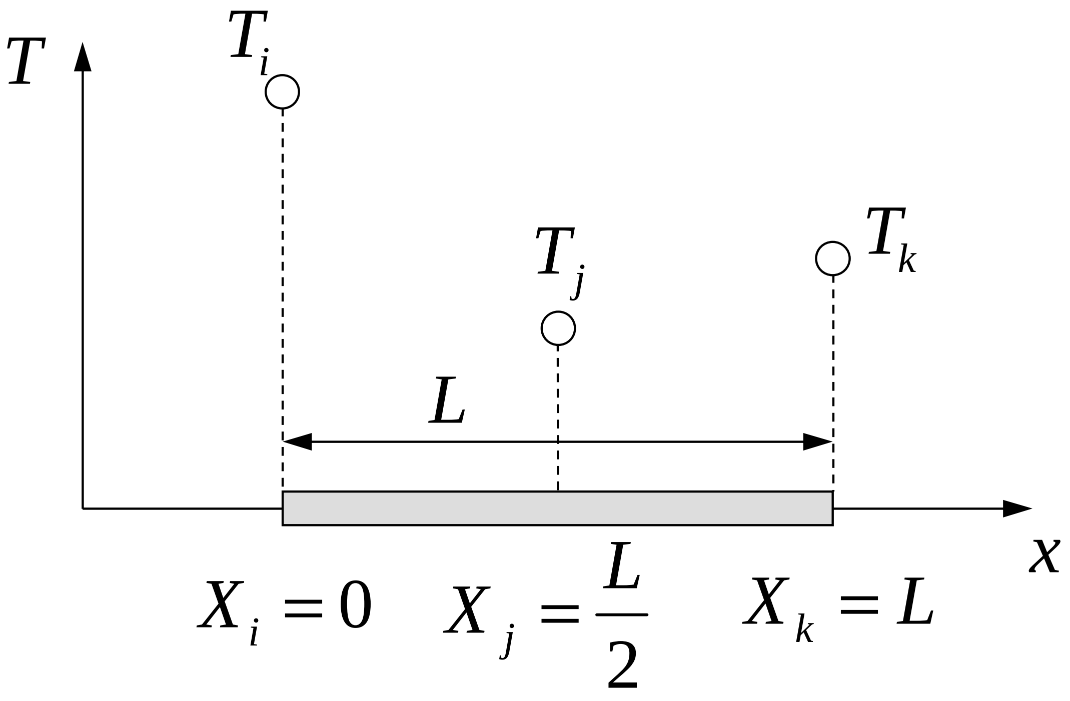


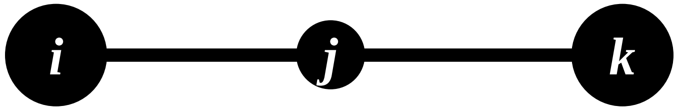
- Shape Functions: N$_i$, N$_j$, N$_k$

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

1D Quadratic Element

一次元二次要素





Intermediate Node
Mid Point: j

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

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

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

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

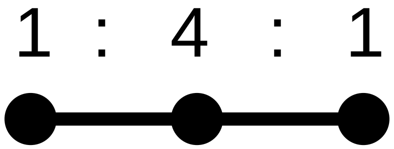
Integration over Each Element: $[k]$ (2/2)

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

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

Integration over Each Element: {*f*}

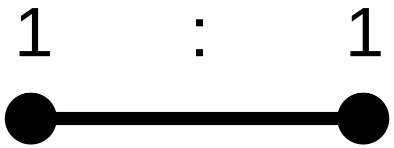
$$\int_V \dot{Q}\left[N\right]^T dV = \dot{Q}A\int_0^L \begin{bmatrix} N_i \\ N_j \\ N_k \end{bmatrix} dx = \dot{Q}A\int_0^L \begin{bmatrix} 1-\frac{3x}{L}+\frac{2x^2}{L^2} \\ \frac{4x}{L}-\frac{4x^2}{L^2} \\ -\frac{x}{L}+\frac{2x^2}{L^2} \end{bmatrix} dx = \frac{\dot{Q}AL}{6}\begin{Bmatrix} 1 \\ 4 \\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



The Ratio was 1:1 in Linear Element

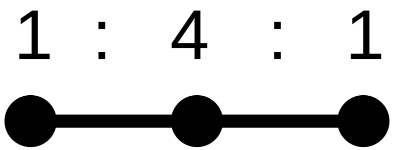
$$N_i = \left(\frac{X_j - x}{L} \right), \quad N_j = \left(\frac{x - X_i}{L} \right) \qquad \frac{dN_i}{dx} = \left(\frac{-1}{L} \right), \quad \frac{dN_j}{dx} = \left(\frac{1}{L} \right)$$

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



Integration over Each Element: {*f*}

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



Volume
Heat Flux

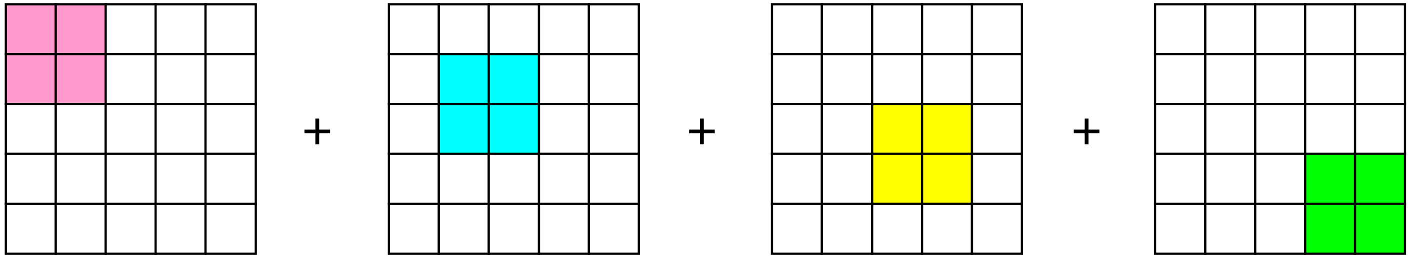
$$\int_S \bar{q}[N]^T dS = \bar{q}A\Big|_{x=L} = \bar{q}A\begin{Bmatrix} 0 \\ 0 \\ 1 \end{Bmatrix}$$

Surface
Heat Flux

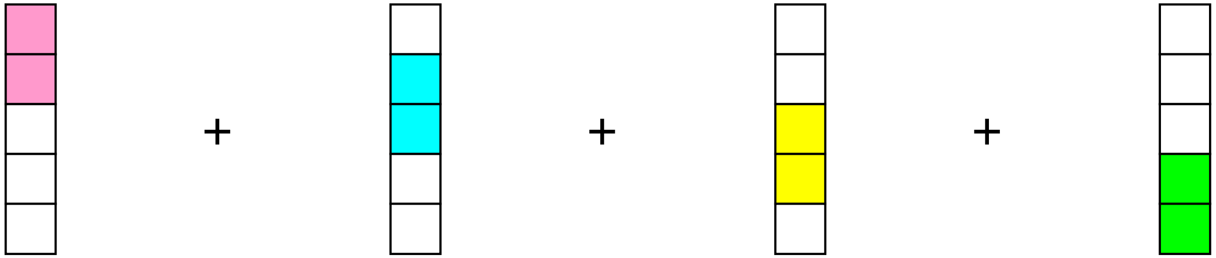
Element Eqn's/Accumulation

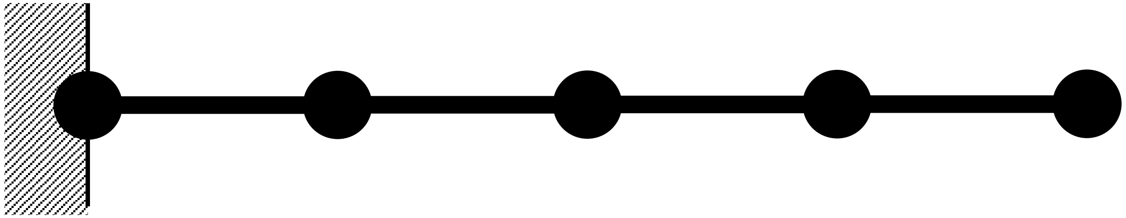
1D Linear Element

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



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

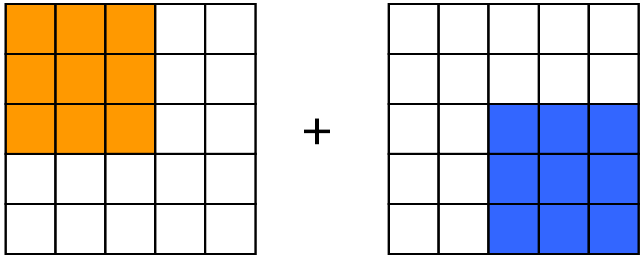




Element Eqn's/Accumulation

1D Quadratic Element, 2 Elements

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



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