

Finite Element Method in Engineering

Dr. Movahedi Rad Majid

Introduction

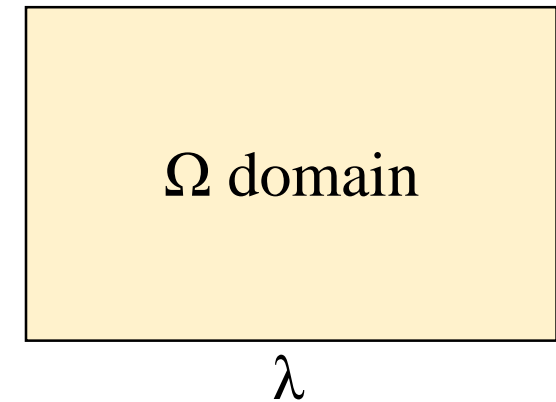
In FEM, the aim is to solve the following differential equation (in Ω domain):

$$L(\phi) + A = 0$$

L : an operator for ϕ variable.

Boundary conditions for λ : $m(\phi) + B = 0$

m : an operator for ϕ variable.



Example

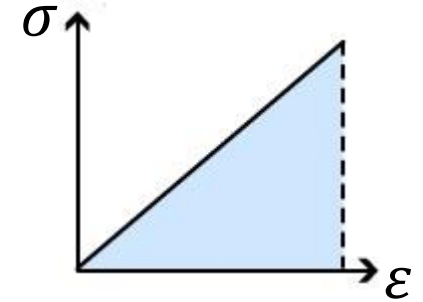
$$\partial^2 \phi / \partial x^2 + 2 = 0 \quad \text{on} \quad \Omega \text{ domain}$$

$$\phi + 2 = 0 \quad \text{on} \quad \lambda$$

Pre requirements

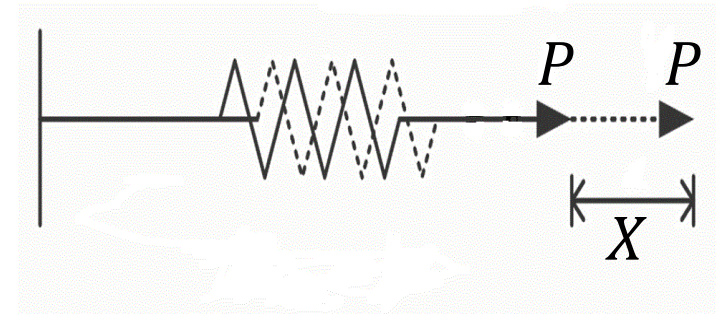
Strain energy:

$$U = \int \frac{1}{2} \sigma \varepsilon dv$$



External work:

$$W_{ext} = PX$$



Total potential energy:

$$\pi = U - W_{ext}$$



Minimum total potential energy:

$$\delta\pi = 0$$

Pre requirements

Functional equation:

$$I(u) = \int F(x, u, u', u'') dx$$

input  function
output  scalar

Differential equation:

$$L(u) + A = 0$$

Rayleigh–Ritz method

Functional equation is used:

$$\pi(u) = \int F(x, u, u', u'') dx$$

Let's assume the solution of functional equation can be defined as a series:

$$u(x) = \sum_{i=1}^n a_i N_i(x) + \psi(x)$$

$N_i(x)$: are trial functions $\psi(x)$: these functions define the boundary conditions

The aim is to find a_i multipliers, where $\pi(u) = \int F(x, u, u', u'') dx = \pi(a_1, \dots, a_n)$

Minimum total potential energy definition:

$$\delta\pi(a_1, \dots, a_n) = 0 \qquad \frac{\partial\pi}{\partial a_1} = 0, \dots, \frac{\partial\pi}{\partial a_n} = 0$$

Rayleigh–Ritz method

$N_i(x)$ (trial functions):

Properties:

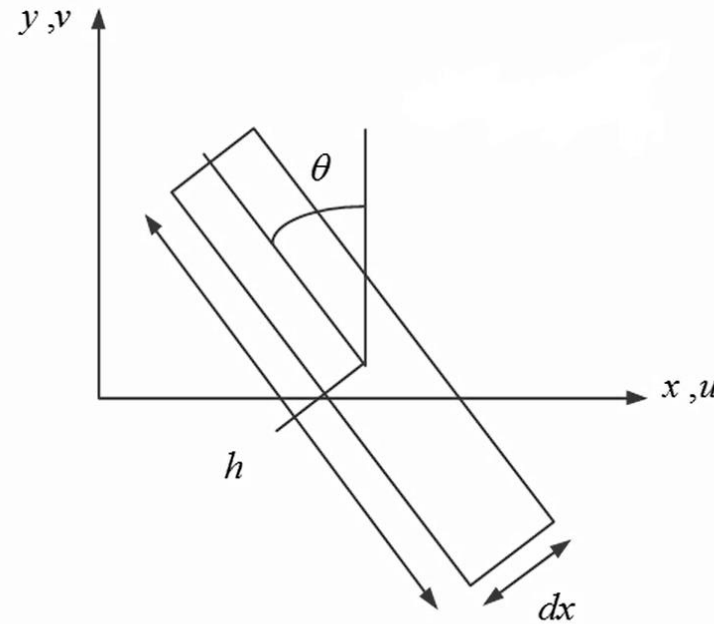
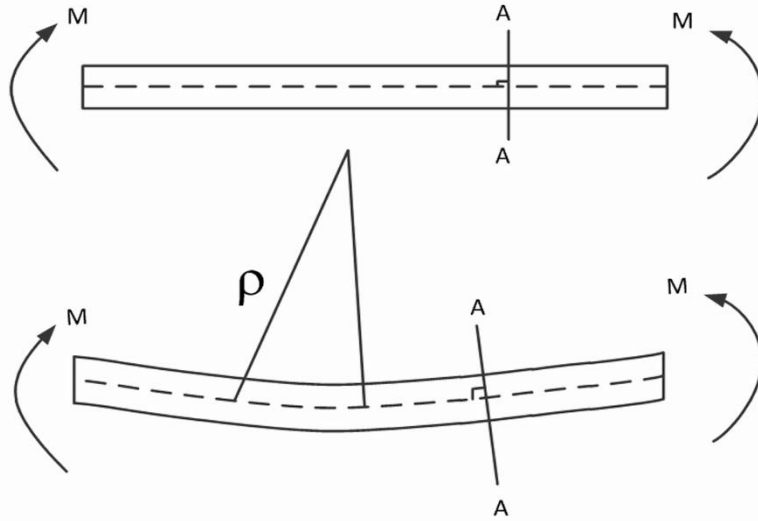
- Independent,
- Cover boundary conditions (and any other physical constraints).

Trial functions can be:

- Polynomial $N_i(x) = x^i$
 - Trigonometric $N_i(x) = \sin(ix)$
- } examples

Euler–Bernoulli beam

Functional equation for Euler–Bernoulli beam:



$$\left. \begin{aligned} \theta &= \frac{dv}{dx} \\ u &= -y \theta \end{aligned} \right\} \Rightarrow u = -y \frac{dv}{dx}$$

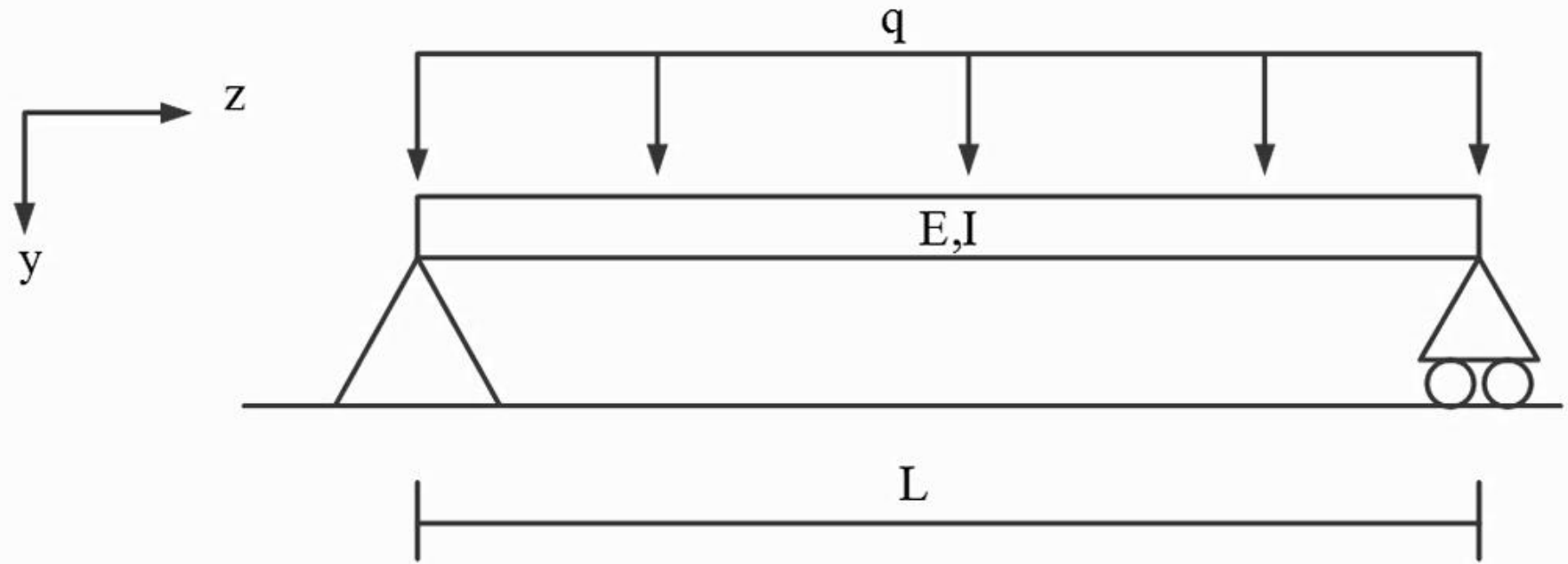
$$\varepsilon_x = \frac{du}{dx} \Rightarrow \boxed{\varepsilon_x = -y \frac{d^2v}{dx^2}}$$

$$u = \int \frac{1}{2} E \varepsilon_x^2 dv = \iint \frac{1}{2} E \varepsilon_x^2 b dx dy$$

$$u = \iint \frac{1}{2} E \left(-y \frac{d^2v}{dx^2} \right)^2 b dx dy \Rightarrow \boxed{u = \int \frac{1}{2} E I_z \left(\frac{d^2v}{dx^2} \right)^2 dx} \quad I_z = \int y^2 b dy$$

Rayleigh–Ritz method

Example 1-1: Euler–Bernoulli beam:



$$\pi = \int_0^L \left[\frac{1}{2} EI (y'')^2 - qy \right] dx \quad B.C. \begin{cases} y = 0 \text{ at } z = 0 \\ y = 0 \text{ at } z = L \end{cases}$$

Rayleigh–Ritz method

Example 1-1: Euler–Bernoulli beam:

Trial function:

$$N_n(z) = \sin \frac{(2(n-1)+1)\pi z}{L}$$
$$N_1(z) = \sin \frac{\pi z}{L}$$
$$N_2(z) = \sin \frac{3\pi z}{L}$$
$$y(z) = a_1 \sin \frac{\pi z}{L} + a_2 \sin \frac{3\pi z}{L} + \underbrace{\psi(z)}_0$$

Boundary conditions are satisfied by trial function!

Rayleigh–Ritz method

Example 1-1: Euler–Bernoulli beam:

MATLAB essential commands:

`syms L E I` to create symbolic variables

`diff(y,'z',1)` first derivation of y with respect to the variable z

`int(y,'z')` integration of y with respect to the variable z

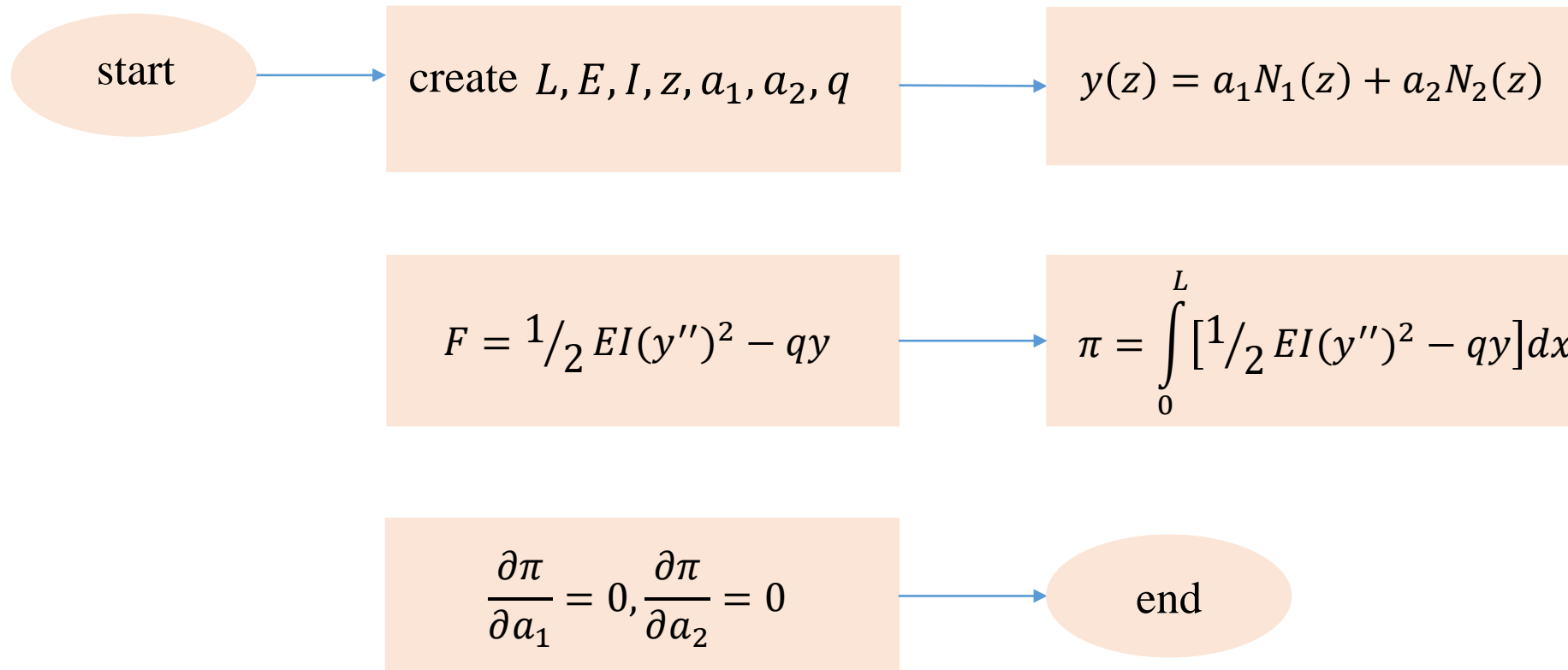
`int(y,'z',0,L)` integration of y with respect to the variable z for the given interval

`x=1;eval(y)` evaluate y function when x=1

Rayleigh–Ritz method

Example 1-1: Euler–Bernoulli beam:

Algorithm:



Rayleigh–Ritz method

Example 1-1: Euler–Bernoulli beam:

Algorithm:

$$\begin{array}{l} \text{M} \\ \text{N} \end{array} \left\{ \begin{array}{l} A(q, L, E, I) a_1 + B(q, L, E, I) = 0 \\ C(q, L, E, I) a_2 + D(q, L, E, I) = 0 \end{array} \right. \Rightarrow \left\{ \begin{array}{l} a_1 = -B(q, L, E, I) / A(q, L, E, I) \\ a_2 = -D(q, L, E, I) / C(q, L, E, I) \end{array} \right.$$

$$\begin{array}{l} a_1 : \\ a_2 : \end{array} \left\{ \begin{array}{l} den1 = \frac{\partial M}{\partial a_1} = A(q, L, E, I) \\ num1 = M|_{a_1=0} = B(q, L, E, I) \\ den2 = \frac{\partial N}{\partial a_2} = C(q, L, E, I) \\ num2 = N|_{a_2=0} = D(q, L, E, I) \end{array} \right. \Rightarrow \left\{ \begin{array}{l} a_1 = -\frac{num1}{den1} = -M|_{a_1=0} / \frac{\partial M}{\partial a_1} \\ a_2 = -\frac{num2}{den2} = -N|_{a_2=0} / \frac{\partial N}{\partial a_2} \end{array} \right.$$

Rayleigh–Ritz method

Example 1-1: Euler–Bernoulli beam:

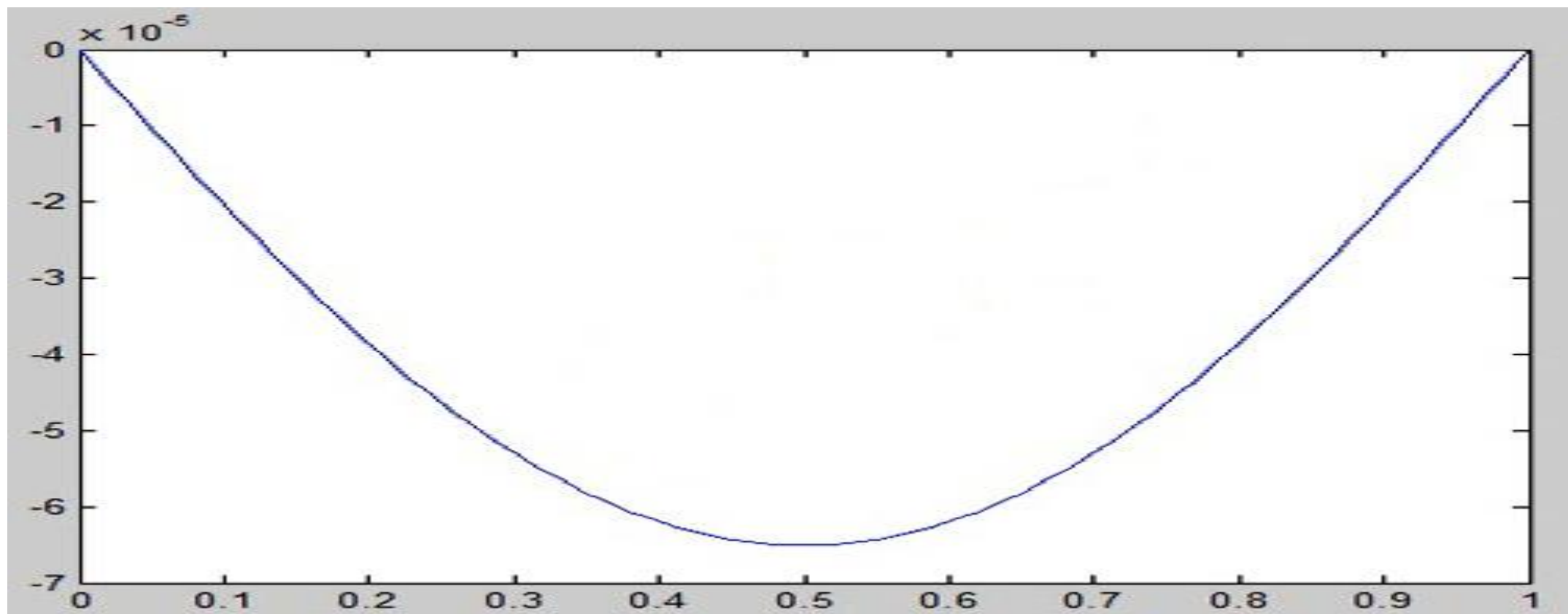
MATLAB code:

```
tic
clc
clear all
close all
%%
syms L E I z a1 a2 q
N1=sin(pi*z/L);
N2=sin(3*pi*z/L);
y=a1*N1+a2*N2;
ydd=diff(y,'z',2);
F=(E*I*ydd^2)/2-q*y;
FUNCTIONAL=int(F,'z',0,1);
%%
EQ_1=diff(FUNCTIONAL,'a1',1);
a1=0;num1=eval(EQ_1);
den1=diff(EQ_1,'a1',1);
a1=-num1/den1
```

```
EQ_2=diff(FUNCTIONAL,'a2',1);
a2=0;num2=eval(EQ_2);
den2=diff(EQ_2,'a2',1);
a2=-num2/den2
%%
y=a1*N1+a2*N2;
E=200e9;
I=1e-6;
L=1;
q=-1000;
z=0:0.01:L;
Y=eval(y)
plot(z,Y)

toc
```

Rayleigh–Ritz method

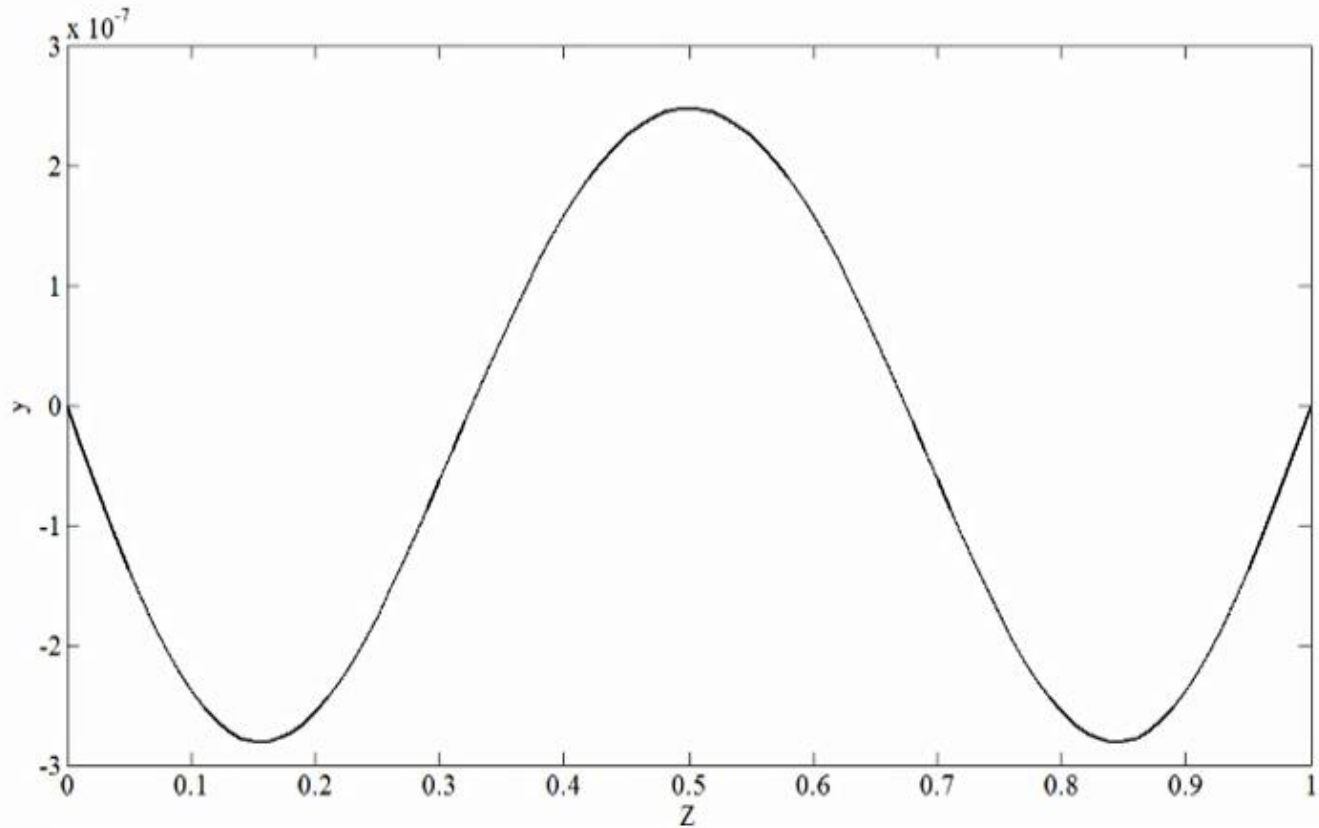


Rayleigh–Ritz method

Example 1-1: Euler–Bernoulli beam:

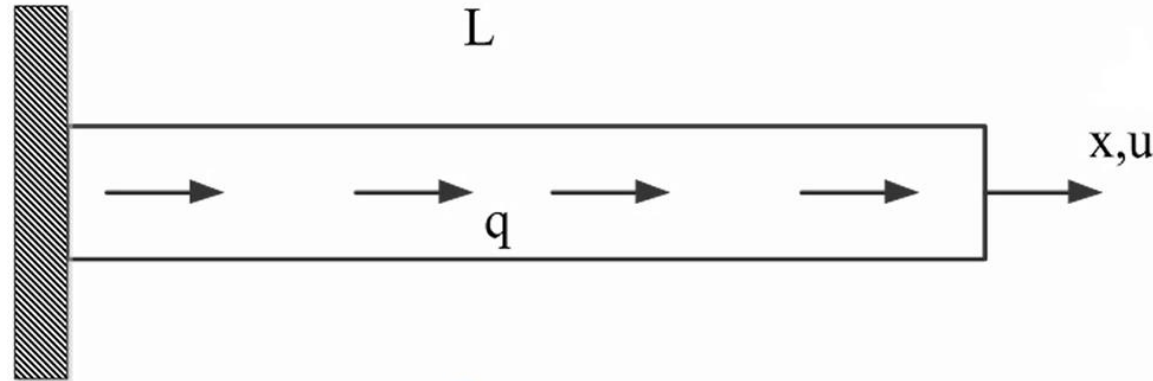
$$N_1(x) = \sin \frac{3\pi z}{L}$$

$$N_2(x) = \sin \frac{5\pi z}{L}$$



Rayleigh–Ritz method

Functional equation for truss element:

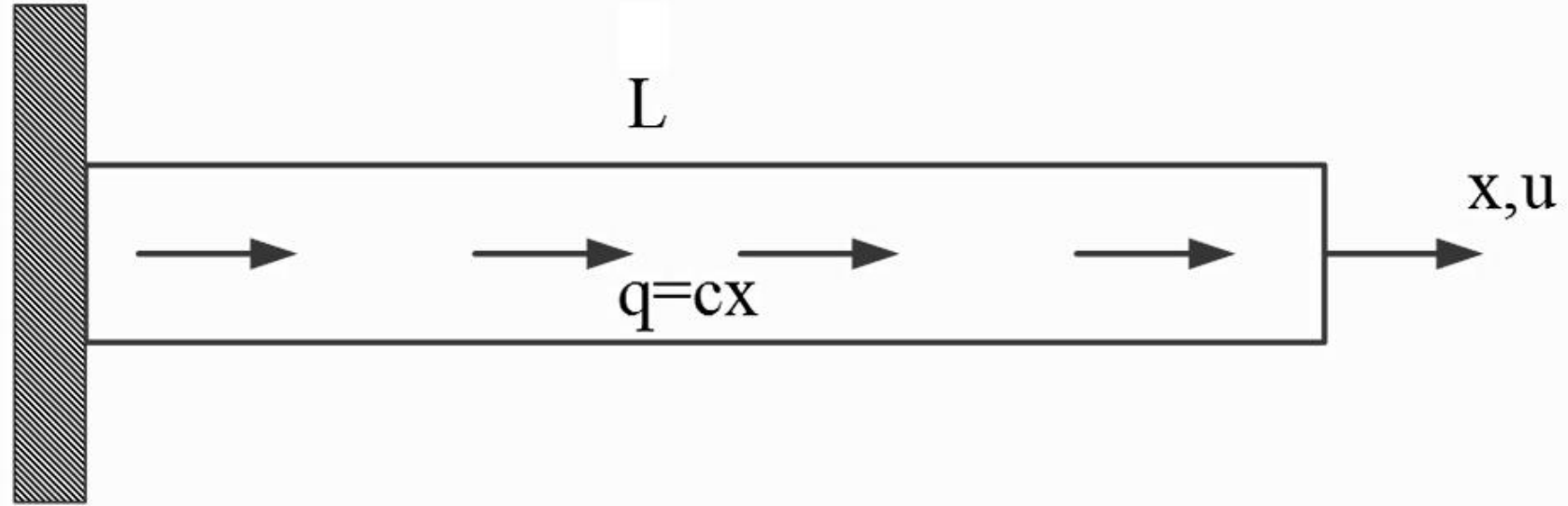


$$\pi = u - W_{ext}$$

$$\left. \begin{aligned} u &= \int \frac{1}{2} \sigma \varepsilon_x dv \\ \sigma &= E \varepsilon_x \end{aligned} \right\} \Rightarrow \left. \begin{aligned} u &= \int \frac{1}{2} E \varepsilon_x^2 A dx \\ \varepsilon_x &= \frac{du}{dx} \end{aligned} \right\} \Rightarrow u = \int \frac{1}{2} E A \left(\frac{du}{dx} \right)^2 dx$$
$$W_{ext} = \int q u dx \Rightarrow \pi = \int \left[\frac{1}{2} E A \left(\frac{du}{dx} \right)^2 - q u \right] dx$$

Rayleigh–Ritz method

Example 1-2:



$$\pi = \int_0^L \left[\frac{1}{2} A E \left(\frac{du}{dx} \right)^2 - cxu \right] dx \quad B.C. \, u = 0 \text{ at } x = 0$$

Rayleigh–Ritz method

Example 1-2:

$$N_n(x) = x^{n-1} \longrightarrow \begin{aligned} N_1(x) &= 1 \\ N_2(x) &= x \\ N_3(x) &= x^2 \end{aligned}$$

$$u(x) = \underset{\downarrow 0}{a_0} + a_1 x + a_2 x^2 + \psi(\underset{\downarrow 0}{x})$$

Boundary conditions are satisfied by trial function!

Rayleigh–Ritz method

Example 1-2:

$$\begin{cases} A(q, L, E, I) a_1 + B(q, L, E, I) a_2 = R1 \\ C(q, L, E, I) a_1 + D(q, L, E, I) a_2 = R2 \end{cases}$$

$$a_1 : \begin{cases} A(q, L, E, I) = \frac{\partial(Eq.1)}{\partial a_1} \\ B(q, L, E, I) = \frac{\partial(Eq.1)}{\partial a_2} \\ R1 = -(Eq.1) \Big|_{\substack{a_1=0 \\ a_2=0}} \end{cases}$$

$$a_2 : \begin{cases} C(q, L, E, I) = \frac{\partial(Eq.2)}{\partial a_1} \\ D(q, L, E, I) = \frac{\partial(Eq.2)}{\partial a_2} \\ R2 = -(Eq.2) \Big|_{\substack{a_1=0 \\ a_2=0}} \end{cases}$$

Rayleigh–Ritz method

Example 1-2:

$$\begin{aligned} Coef_matrix &= \begin{bmatrix} A(q, L, E, I) & B(q, L, E, I) \\ C(q, L, E, I) & D(q, L, E, I) \end{bmatrix} \\ R &= \begin{bmatrix} R1 \\ R2 \end{bmatrix} \end{aligned} \quad \Rightarrow \quad [Coef_matrix] \begin{Bmatrix} a_1 \\ a_2 \end{Bmatrix} = [R]$$

$$\Rightarrow \begin{Bmatrix} a_1 \\ a_2 \end{Bmatrix} = [Coef_matrix]^{-1} [R]$$

Rayleigh–Ritz method

Example 1-2:

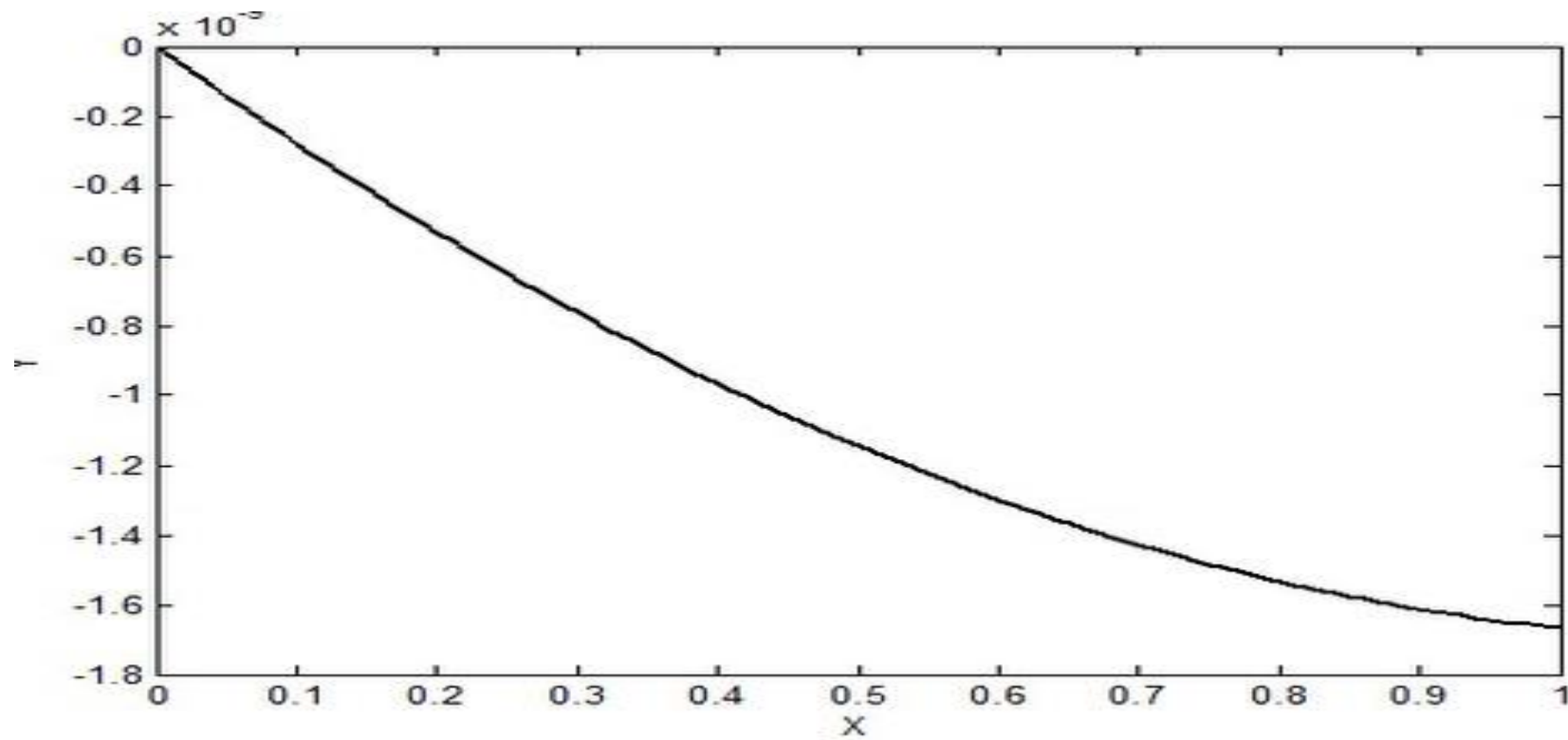
MATLAB code:

```
tic
clc
clear all
close all
%%
syms L E A x a1 a2 c
N1=x;
N2=x^2;
u=a1*N1+a2*N2;
ud=diff(u,'x',1);
F=(E*A*ud^2)/2-c*x*u;
FUNCTIONAL=int(F,'x',0,1);
%%
EQ_1=diff(FUNCTIONAL,'a1',1);
A=diff(EQ_1,'a1',1);
B=diff(EQ_1,'a2',1);
a1=0;a2=0;R1=eval(EQ_1);
```

```
EQ_2=diff(FUNCTIONAL,'a2',1);
C=diff(EQ_2,'a1',1);
D=diff(EQ_2,'a2',1);
a1=0;a2=0;R2=eval(EQ_2);
```

```
Coef_matrix=[A B;C D];
R=[R1;R2];
Coef=Coef_matrix^-1*R;
a1=Coef(1,1)
a2=Coef(2,1)
%%
u=a1*N1+a2*N2;
E=200e9;
A=1e-2;
L=1;
c=10;
x=0:0.01:L;
U=eval(u);
plot(x,U)
toc
```

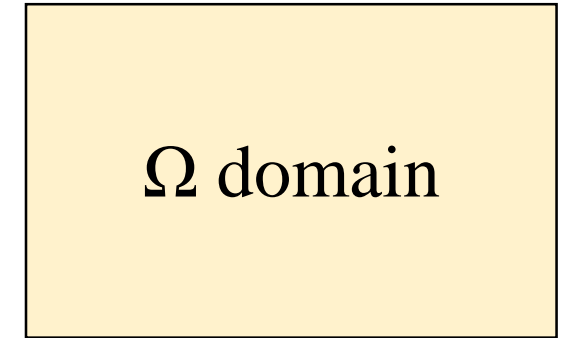
Rayleigh–Ritz method



Rayleigh–Ritz method with Penalty Function Method

Boundary conditions for λ : $m(\phi) + B = 0$

$$\pi_1(u) = \pi(u) + \alpha \oint_{\lambda} [m(\phi) + B]^2 dx$$



λ

α : penalty number, a large number (for ex. $\alpha = 10^6$)

The aim is to find a_i multipliers, where $\pi_1(u) = \pi_1(a_1, \dots, a_n)$

Minimum total potential energy definition:

$$\delta \pi_1(a_1, \dots, a_n) = 0 \quad \longrightarrow \quad \frac{\partial \pi_1}{\partial a_1} = 0, \dots, \frac{\partial \pi_1}{\partial a_n} = 0$$

$$\pi = \int_0^1 \frac{1}{2} \left[\left(\frac{d\phi}{dx} \right)^2 + \phi^2 \right] dx \quad B.C. \begin{cases} \phi = 0 & \text{at } x = 0 \\ \phi = 1 & \text{at } x = 1 \end{cases}$$

$$0 \quad \xrightarrow{0 \leq \Omega \leq 1} \quad 1$$

$$\lambda = 0$$

$$\lambda = 1$$

$$N_n(x) = x^{n-1}$$

$$N_1(x) = 1$$

$$N_2(x) = x$$

$$N_3(x) = x^2$$

$$u(x) = a_1 + a_2 x + a_3 x^2$$

$$\pi_1 = \int_0^1 \frac{1}{2} \left[\left(\frac{d\phi}{dx} \right)^2 + \phi^2 \right] dx + \alpha \left[\phi^2 \right]_{x=0} + \alpha \left[(\phi - 1)^2 \right]_{x=1}$$

$$B.C. \begin{cases} \phi = 0 \text{ at } x = 0 \\ \phi = 1 \text{ at } x = 1 \end{cases}$$

Rayleigh–Ritz method with Penalty Function Method

Example 1-3:

MATLAB code:

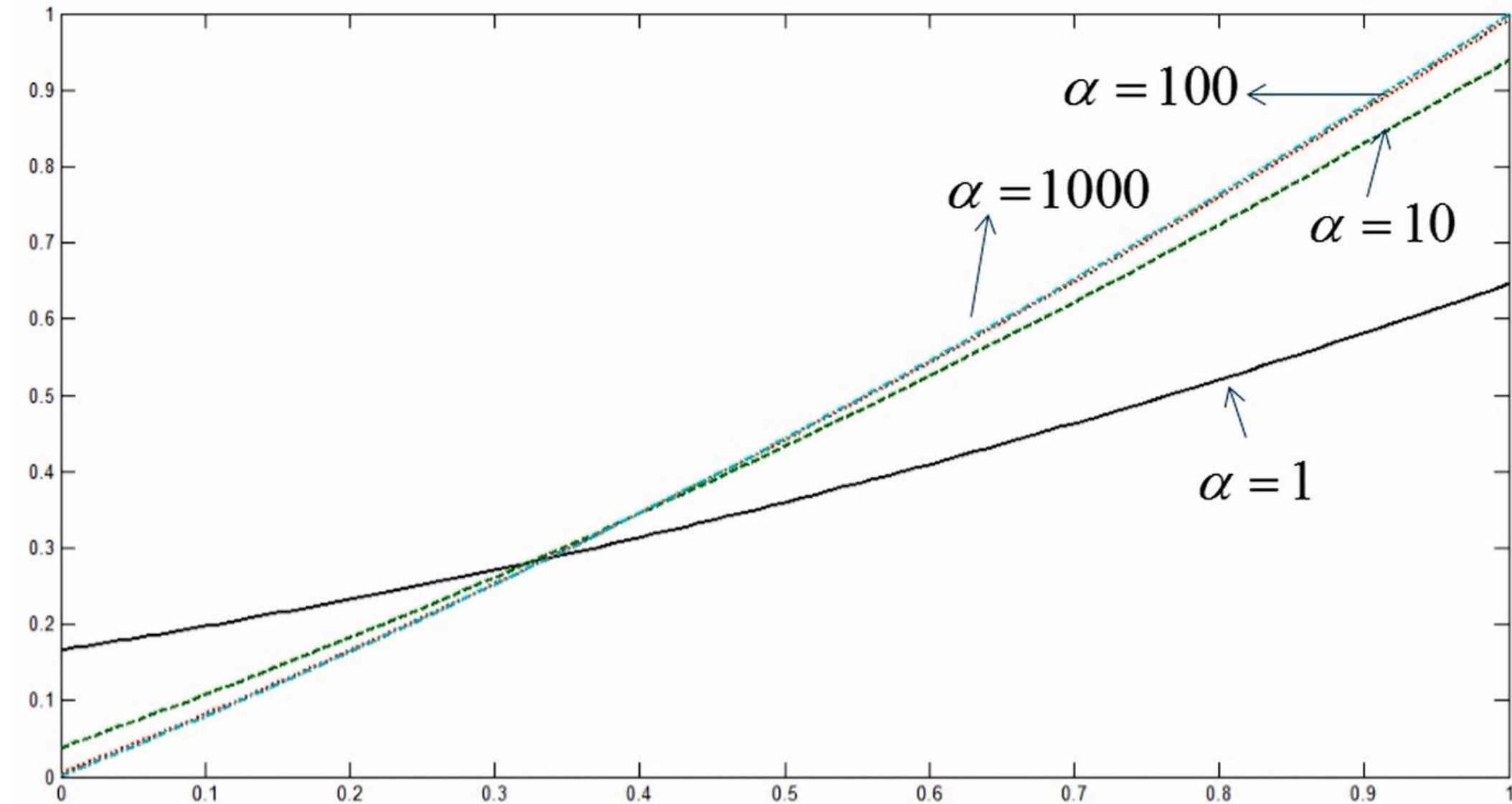
```
% example 13
tic
clc
clear all
close all
%%
syms x a1 a2 a3 alpha
N1=1;
N2=x;
N3=x^2;
phi=a1*N1+a2*N2+a3*N3;
phid=diff(phi,'x',1);
F=(phid^2+phi^2)/2;
FUNCTIONAL=int(F,'x',0,1);
x=0;BC1=eval(phi^2);
x=1;BC2=eval((phi-1)^2);
FUNCTIONAL_BC=alpha*BC1+alpha*BC2;
FUNCTIONAL=FUNCTIONAL+FUNCTIONAL_BC;
%%
```

```
EQ_1=diff(FUNCTIONAL,'a1',1);
A=diff(EQ_1,'a1',1);
B=diff(EQ_1,'a2',1);
C=diff(EQ_1,'a3',1);
a1=0;a2=0;a3=0;R1=-eval(EQ_1);
EQ_2=diff(FUNCTIONAL,'a2',1);
D=diff(EQ_2,'a1',1);
E=diff(EQ_2,'a2',1);
F=diff(EQ_2,'a3',1);
a1=0;a2=0;a3=0;R2=-eval(EQ_2);
EQ_3=diff(FUNCTIONAL,'a3',1);
G=diff(EQ_3,'a1',1);
H=diff(EQ_3,'a2',1);
I=diff(EQ_3,'a3',1);
a1=0;a2=0;a3=0;R3=-eval(EQ_3);
Coef_matrix=[A B C;D E F;G H I];
R=[R1;R2;R3];
Coef=Coef_matrix^-1*R;
a1=Coef(1,1);
a2=Coef(2,1);
a3=Coef(3,1);
%%
```

Rayleigh–Ritz method with Penalty Function Method

```
phi=a1*N1+a2*N2+a3*N3;  
x=0:0.01:1;  
alpha=1;  
U1=eval(phi);  
alpha=10;  
U2=eval(phi);  
alpha=100;  
U3=eval(phi);  
alpha=1000;  
U4=eval(phi);  
plot(x,U1,x,U2,x,U3,x,U4)  
toc
```

Rayleigh–Ritz method with Penalty Function Method



Galerkin method

Differential equation is used:

$$L(u) + A = 0$$

Let's assume the solution of functional equation can be defined as a series:

$$\tilde{u}(x) = \sum_{i=1}^n a_i N_i(x) + \psi(x)$$

$N_i(x)$: are trial functions

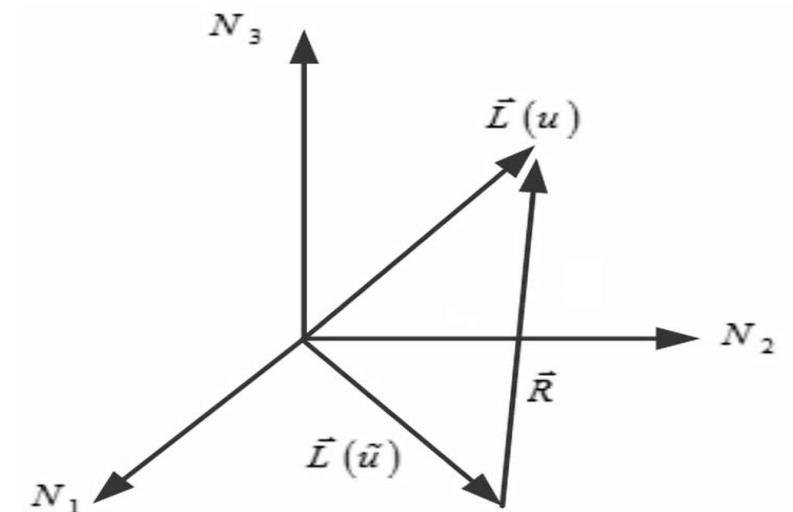
$\psi(x)$: these functions define the boundary conditions

Minimization of errors:

$$L(\tilde{u}) + A = R$$

Minimization of errors should be in Ω domain:

$$L(\tilde{u}) + A = R_{\Omega}$$



Galerkin method

The aim is to find a_i multipliers, where errors are minimized: $\int W_i R_\Omega d\Omega = 0$

If $W_i = N_i$  Babnov Galerking

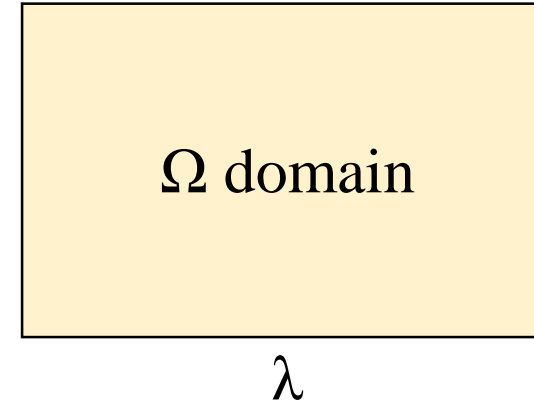
If $W_i \neq N_i$  Petrov Galerking

$$\left. \begin{array}{l} \int W_i \textcircled{R_\Omega} d\Omega = 0 \\ L(\tilde{u}) + A = R_\Omega \end{array} \right\} \Rightarrow \int W_i [L(\tilde{u}) + A] d\Omega = 0$$

$$\Rightarrow \int W_i [L(\tilde{u})] d\Omega = - \int W_i [A] d\Omega \Rightarrow [K] \{a_i\} = \{F_i\}$$

Extended Galerkin method

$$m(\tilde{u}) + B = R_\lambda \quad \text{on} \quad \lambda$$



The aim is to find a_i multipliers,
where errors are minimized:

$$\int W_i R_\Omega d\Omega + \oint \bar{W}_i R_\lambda d\lambda = 0$$

$$W_i = \bar{W}_i$$
$$W_i = -\bar{W}_i$$

Extended Galerkin method

$$\left. \begin{aligned} \int W_i \circledast R_\Omega d\Omega + \oint_{\lambda} \bar{W}_i \circledast R_\lambda d\lambda &= 0 \\ L(\tilde{u}) + A &= R_\Omega \\ m(\tilde{u}) + B &= R_\lambda \end{aligned} \right\} \Rightarrow \int W_i [L(\tilde{u}) + A] d\Omega + \oint_{\lambda} \bar{W}_i [m(\tilde{u}) + B] d\lambda = 0$$

$$\Rightarrow \underbrace{\int W_i [L(\tilde{u})] d\Omega}_{[K]} + \underbrace{\oint_{\lambda} \bar{W}_i [m(\tilde{u})] d\lambda}_{[K']} = - \underbrace{\int W_i [A] d\Omega}_{\{F_i\}} - \underbrace{\oint_{\lambda} \bar{W}_i [B] d\lambda}_{\{F'_i\}}$$

$$\Rightarrow [K + K'] \{a_i\} = \{F_i + F'_i\}$$

Galerkin method

$$\left(\frac{d^2 u}{dx^2} \right) - u = 0 \quad B.C. \begin{cases} u = 0 \text{ at } x = 0 \\ u = 1 \text{ at } x = 1 \end{cases}$$

$$0 \leq \Omega \leq 1$$

$\lambda = 0$
 $\lambda = 1$

$$N_n(x) = \sin(n\pi x)$$

$$N_n(x) = \sin(\pi x)$$

$$N_n(x) = \sin(2\pi x)$$

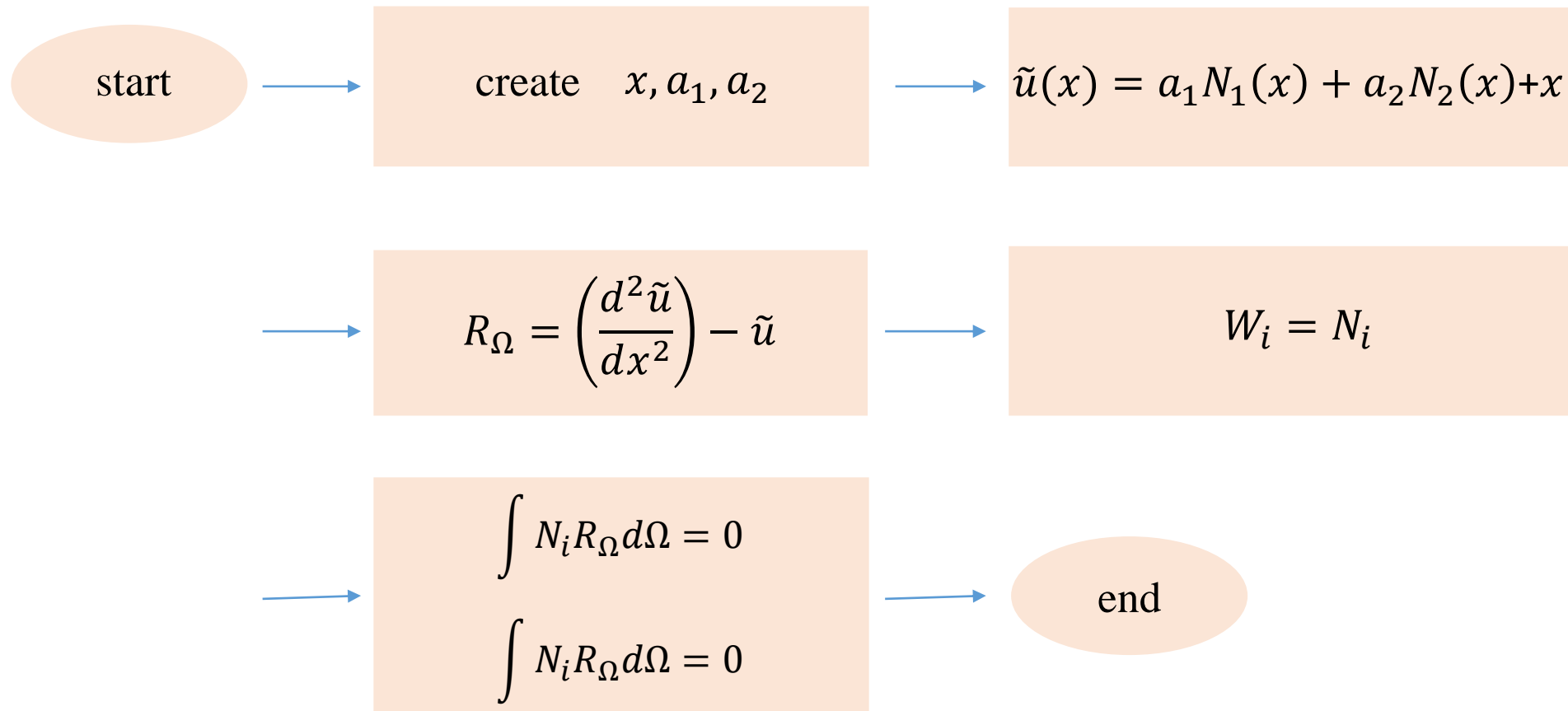
$$u(x) = \underset{0}{\cancel{a_0}} + a_1 \sin(2\pi x) + a_2 \sin(\pi x) + \cancel{\psi}(x)$$

$\psi(x) = x$

Galerkin method

Example 1-4: Galerkin method:

Algorithm:



Galerkin method

Example 1-4: Galerkin method:

MATLAB code:


```
% example14
tic
clc
clear all
close all
%%
syms x a1 a2
N1=sin(pi*x);
N2=sin(2*pi*x);
u=a1*N1+a2*N2+x;
udd=diff(u,'x',2);
R=udd-u;
%%
F1=R*N1;
EQ_1=int(F1,'x',0,1);
A=diff(EQ_1,'a1',1);
B=diff(EQ_1,'a2',1);
a1=0;a2=0;R1=-eval(EQ_1);
```

```
F2=R*N2;
EQ_2=int(F2,'x',0,1);
C=diff(EQ_2,'a1',1);
D=diff(EQ_2,'a2',1);
a1=0;a2=0;R2=-eval(EQ_2);

Coef_matrix=[A B;C D];
R=[R1;R2];
Coef=Coef_matrix*R;
a1=Coef(1,1)
a2=Coef(2,1)
%%
u=a1*N1+a2*N2+x;
x=0:0.01:1;
U=eval(u);
createfigure(x,U)
toc
```

Extended Galerkin method

Example 1-5: Extended Galerkin method:

$$\left(\frac{d^2 u}{dx^2} \right) - u = 0 \quad B.C. \begin{cases} u = 0 \text{ at } x = 0 \\ u = 1 \text{ at } x = 1 \end{cases}$$


A horizontal line segment representing the domain $0 \leq \Omega \leq 1$. The left endpoint is labeled '0' and the right endpoint is labeled '1'. The label $0 \leq \Omega \leq 1$ is centered above the line.

$$\lambda = 0$$
$$\lambda = 1$$

$$N_n(x) = x^{n-1}$$

$$N_1(x) = 1$$

$$N_2(x) = x$$

$$N_3(x) = x^2$$

$$\tilde{u}(x) = a_0 + a_1 x + a_2 x^2$$

Extended Galerkin method

$$R_{\Omega} = \left(\frac{d^2 \tilde{u}}{dx^2} \right) - \tilde{u}$$

$$u = 0 \text{ at } x = 0$$

$$R_{\lambda_1} = \tilde{u} \Big|_{x=0}$$

$$u = 1 \text{ at } x = 1$$

$$R_{\lambda_2} = \tilde{u} \Big|_{x=1}$$

$$\int W_i R_{\Omega} d\Omega + \oint_{\lambda} \bar{W}_i R_{\lambda} d\lambda = 0$$

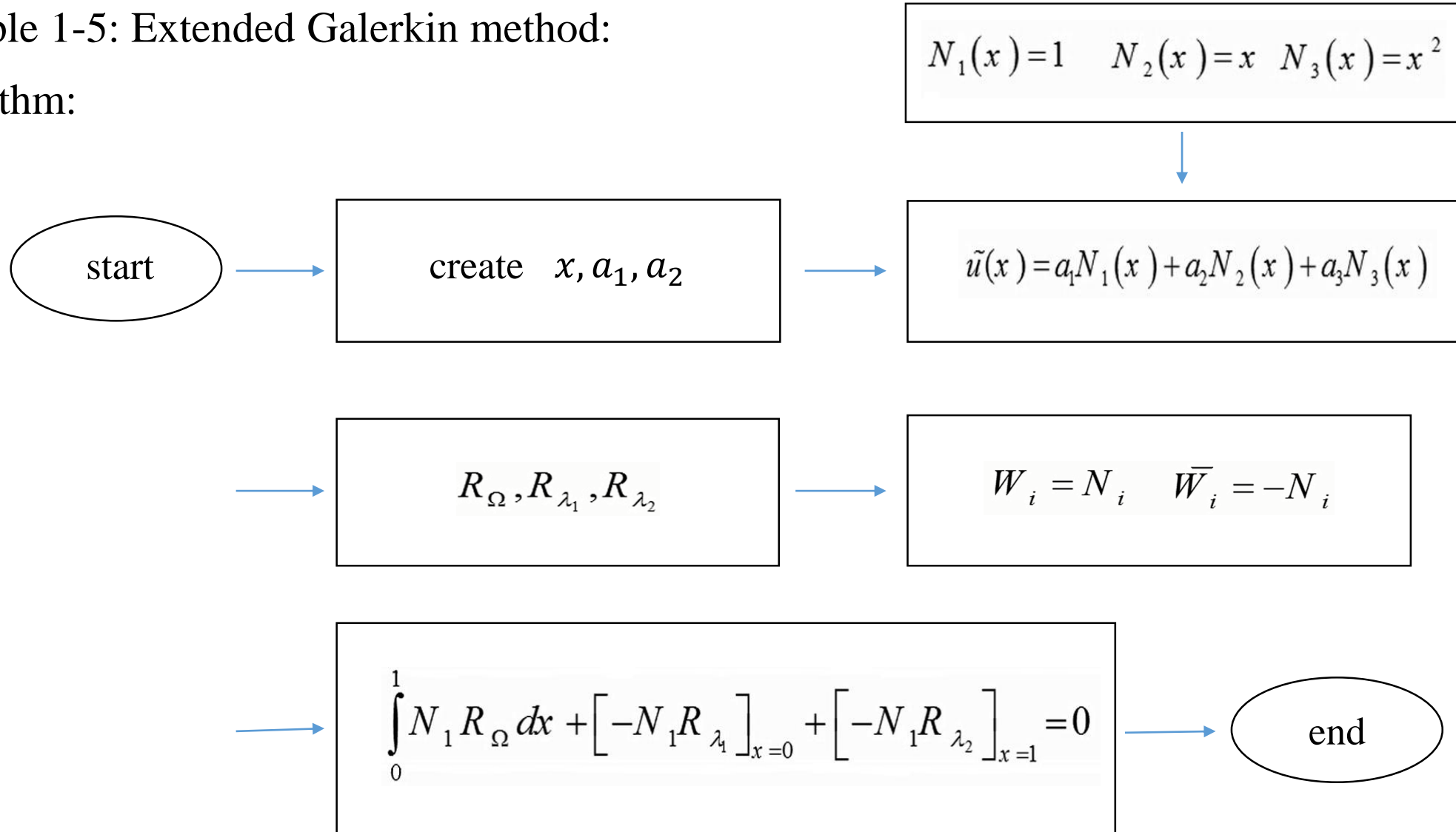
$$W_i = N_i$$

$$\bar{W}_i = -N_i$$

Extended Galerkin method

Example 1-5: Extended Galerkin method:

Algorithm:



Extended Galerkin method

Example 1-5:

MATLAB code:

```
% example 1-5
tic
clc
clear all
close all
%%
syms x a1 a2 a3
N1=1;
N2=x;
N3=x^2;
u=a1*N1+a2*N2+a3*N3;
udd=diff(u,'x',2);
R_domain=udd-u;
%%
```

```
x=0;R_B_1_1=eval(u);
x=1;R_B_2_1=eval(u)-1;
x=0;R_B_1_2=eval(u)*eval(N2);
x=1;R_B_2_2=(eval(u)-1)*eval(N2);
x=0;R_B_1_3=eval(u)*eval(N3);
x=1;R_B_2_3=(eval(u)-1)*eval(N3);
%%
F1=R_domain*N1;
EQ_1=int(F1,'x',0,1)-R_B_1_1-R_B_2_1;
A=diff(EQ_1,'a1',1);
B=diff(EQ_1,'a2',1);
C=diff(EQ_1,'a3',1);
a1=0;a2=0;a3=0;R1=-eval(EQ_1);
F2=R_domain*N2;
EQ_2=int(F2,'x',0,1)-R_B_1_2-R_B_2_2;
D=diff(EQ_2,'a1',1);
E=diff(EQ_2,'a2',1);
F=diff(EQ_2,'a3',1);
a1=0;a2=0;a3=0;R2=-eval(EQ_2);
```

Extended Galerkin method

```
F3=R_domain*N3;
EQ_3=int(F3,'x',0,1)-R_B_1_3-R_B_2_3;
G=diff(EQ_3,'a1',1);
H=diff(EQ_3,'a2',1);
I=diff(EQ_3,'a3',1);
a1=0;a2=0;a3=0;R3=-eval(EQ_3);
Coef_matirx=[A B C;D E F;G H I];
R=[R1;R2;R3];
Coef=Coef_matirx^-1*R;
a1=Coef(1,1);
a2=Coef(2,1);
a3=Coef(3,1);
%%
u=a1*N1+a2*N2+a3*N3;
x=0:0.01:1;
U=eval(u);
createfigure(x,U)
toc
```

Differential equation



Functional equation

Differential equation: $L(u) + A = 0$

Conditions:

$$\int_{\Omega} \theta L(u) d\Omega = \int_{\Omega} u L(\theta) d\Omega$$

Symmetric (Self-adjoint)

$$\int_{\Omega} u L(u) d\Omega \geq 0$$

Positive definite

Differential equation to Functional equation

Differential equation:

$$L(u(x)) + A = 0$$

$$0 \leq x \leq L$$

n : order for differential equation

Boundary conditions:

$$m(u(x)) \Big|_{x=0} = B$$

$$n(u(x)) \Big|_{x=L} = C$$

Differential equation to Functional equation

Weighted-Residual Integral:
$$\int_0^L W(x) [L(u(x)) + A] dx = 0$$

Weight Function:
$$W(x)$$

$$\int_0^L W(x) [L(u(x))] dx + \int_0^L W(x) A dx = 0$$

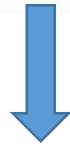
Differential equation to Functional equation

Weak-Form by Fractional Calculus

Fractional Calculus:

$$\int_0^L u \, dv = [uv] \Big|_0^L - \int_0^L v \, du$$

$$\int_0^L W(x) [L(u(x))] dx + \int_0^L W(x) A dx = 0$$



$$V(x) = \int_0^L L(u(x)) dx$$

Differential equation to Functional equation

Weak-Form by Fractional Calculus

$$\left[W(x) V(x) \right] \Big|_0^L - \int_0^L \frac{dW(x)}{dx} V(x) dx + \int_0^L W(x) A dx = 0$$

Applying boundary conditions:

$$-\left[W(x) \cancel{V(x)} \right] \Big|_{x=L} + \left[W(x) \cancel{V(x)} \right] \Big|_{x=0} + \int_0^L \left[-\frac{dW(x)}{dx} V(x) + W(x) A \right] dx = 0$$

\swarrow \searrow
 0

$$W(L) V(L) = W(L) C$$

Differential equation to Functional equation

$$-W(L)C + \int_0^L \left[-\frac{dW(x)}{dx} \cancel{V(x)} + W(x)A \right] dx = 0$$

$$V(x) = \int_0^L L(u(x)) dx$$

$$\int_0^L \left[-\frac{dW(x)}{dx} \left[\int_0^L L(u(x)) dx \right] \right] dx + \int_0^L [W(x)A] dx - W(L)C = 0$$

$B(u, W)$

$L(W)$

Differential equation to Functional equation

$$\left. \begin{aligned} B(u, W) &= \int_0^L \left[-\frac{dW(x)}{dx} \left[\int_0^L L(u(x)) dx \right] \right] dx \\ L(W) &= \int_0^L [W(x)A] dx - W(L)C \end{aligned} \right\} \Rightarrow \pi(u) = \frac{1}{2} B(u, u) + L(u)$$

Differential equation to Functional equation

Differential equation:

$$-\frac{d}{dx} \left[AE \frac{du}{dx} \right] = q$$

Boundary conditions:

$$\begin{aligned} u(x) \Big|_{x=0} &= 0 \\ AE \frac{du}{dx} \Big|_{x=L} &= Q_0 \end{aligned}$$

First Step:

Weighted-Residual Integral:

$$\int_0^L W(x) \left[-\frac{d}{dx} \left[AE \frac{du}{dx} \right] - q \right] dx = 0$$

Differential equation to Functional equation

Second step:

Fractional Calculus:

$$-\left[AEW(x) \frac{du}{dx} \right] \Big|_0^L + \int_0^L AE \frac{dW(x)}{dx} \frac{du}{dx} dx - \int_0^L W(x) q dx = 0$$

$$\left[AEW(x) \frac{du}{dx} \right] \Big|_{x=0} - \left[AEW(x) \frac{du}{dx} \right] \Big|_{x=L} + \int_0^L \left[AE \frac{dW(x)}{dx} \frac{du}{dx} - W(x) q \right] dx = 0$$

Differential equation to Functional equation

Third step:

Boundary conditions:

$$\left[\cancel{AEW(x)} \frac{du}{dx} \right] \Big|_{x=0} - \left[AEW(x) \frac{du}{dx} \right] \Big|_{x=L} + \int_0^L \left[AE \frac{dW(x)}{dx} \frac{du}{dx} - W(x)q \right] dx = 0$$

0 $AE \frac{du}{dx} \Big|_{x=L} = Q_0$

$$-W(L)Q_0 + \int_0^L \left[AE \frac{dW(x)}{dx} \frac{du}{dx} - W(x)q \right] dx = 0$$

Differential equation to Functional equation

Fourth step:

Functional equation:

$$\int_0^L \left[AE \frac{dW(x)}{dx} \frac{du}{dx} \right] dx + \int_0^L \left[-W(x)q \right] dx - W(L)Q_0 = 0$$

$$B(u, W)$$

$$L(W)$$

$$B(u, u) = \int_0^L \left[AE \frac{du}{dx} \frac{du}{dx} \right] dx = \int_0^L \left[AE \left(\frac{du}{dx} \right)^2 \right] dx$$

$$L(u) = \int_0^L \left[-u(x)q \right] dx - u(L)Q_0$$

$$\pi(u) = \frac{1}{2} B(u, u) + L(u)$$

Functional equation to Differential equation

Minimum total potential energy definition:

$$I(u) = \int_a^b F(x, u, u', u'') dx$$

$$\delta I(u) = 0 \quad \Rightarrow \quad \delta I(u) = \int_a^b \left[\cancel{\frac{\partial F}{\partial x}} \delta x + \frac{\partial F}{\partial u} \delta u + \boxed{\frac{\partial F}{\partial u'} \delta u'} + \boxed{\frac{\partial F}{\partial u''} \delta u''} \right] = 0$$

Changes just considered for the functions

Fractional calculus

Functional equation to Differential equation

Minimum total potential energy definition:

Fractional calculus

$$\int_a^b \left[\frac{\partial F}{\partial u'} \delta u' \right] = \left[\frac{\partial F}{\partial u'} \delta u \right]_a^b - \int_a^b \left[\frac{d}{dx} \left(\frac{\partial F}{\partial u'} \right) \delta u \right]$$

$$\int_a^b \left[\frac{\partial F}{\partial u''} \delta u'' \right] = \left[\frac{\partial F}{\partial u''} \delta u' \right]_a^b - \int_a^b \left[\frac{d}{dx} \left(\frac{\partial F}{\partial u''} \right) \delta u' \right] \int_a^b \left[\frac{d}{dx} \left(\frac{\partial F}{\partial u''} \right) \delta u' \right] =$$

$$\left[\frac{d}{dx} \left(\frac{\partial F}{\partial u''} \right) \delta u \right]_a^b - \int_a^b \left[\frac{d^2}{dx^2} \left(\frac{\partial F}{\partial u''} \right) \delta u \right]$$

$$\int_a^b \left[\frac{\partial F}{\partial u''} \delta u'' \right] = \left[\frac{\partial F}{\partial u''} \delta u' \right]_a^b - \left[\frac{d}{dx} \left(\frac{\partial F}{\partial u''} \right) \delta u \right]_a^b + \int_a^b \left[\frac{d^2}{dx^2} \left(\frac{\partial F}{\partial u''} \right) \delta u \right]$$

Functional equation to Differential equation

Minimum total potential energy definition:

Substitution:

$$\delta I(u) = \left[\left(\frac{\partial F}{\partial u'} - \frac{d}{dx} \left(\frac{\partial F}{\partial u''} \right) \right) \delta u \right]_a^b + \left[\frac{\partial F}{\partial u''} \delta u' \right]_a^b \quad \begin{matrix} u(a) = A \\ u(b) = B \end{matrix} \Rightarrow [\delta u]_a^b = 0$$

$$+ \int_a^b \left[\left(\frac{\partial F}{\partial u} - \frac{d}{dx} \left(\frac{\partial F}{\partial u'} \right) + \frac{d^2}{dx^2} \left(\frac{\partial F}{\partial u''} \right) \right) \delta u \right] = 0 \quad \begin{matrix} u'(a) = A' \\ u'(b) = B' \end{matrix} \Rightarrow [\delta u']_a^b = 0$$

$$\frac{\partial F}{\partial u} - \frac{d}{dx} \left(\frac{\partial F}{\partial u'} \right) + \frac{d^2}{dx^2} \left(\frac{\partial F}{\partial u''} \right) = 0$$

Euler Equation

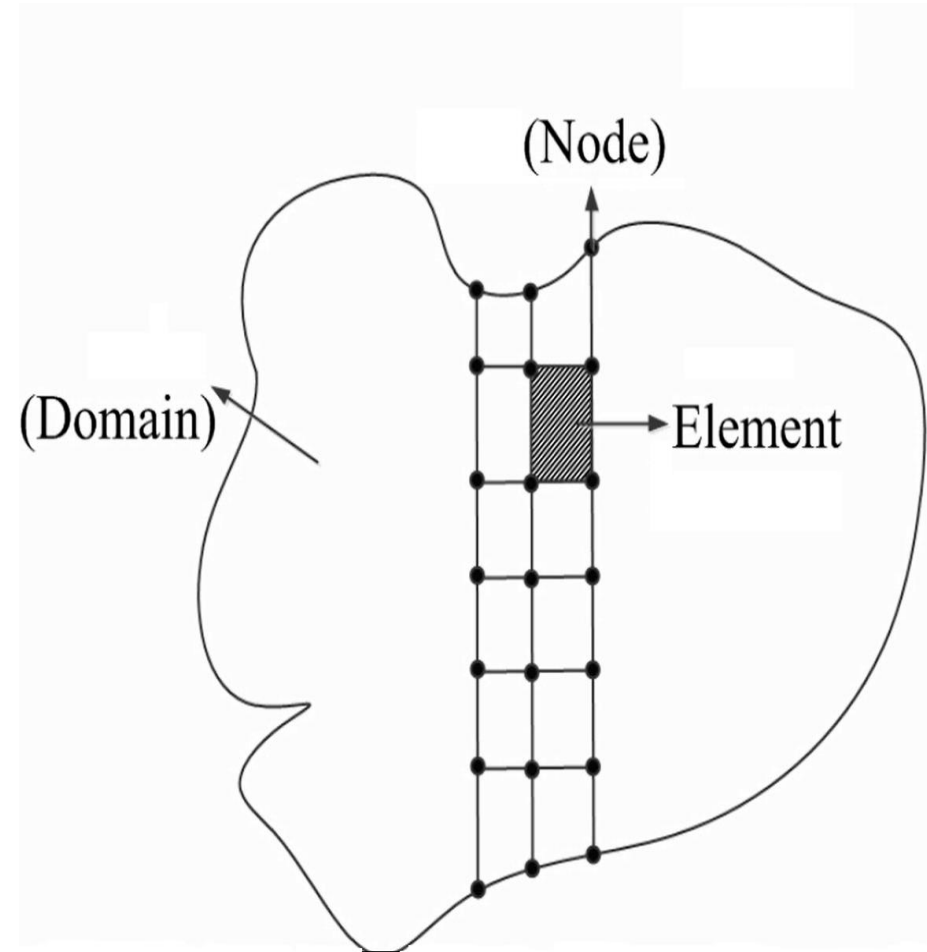
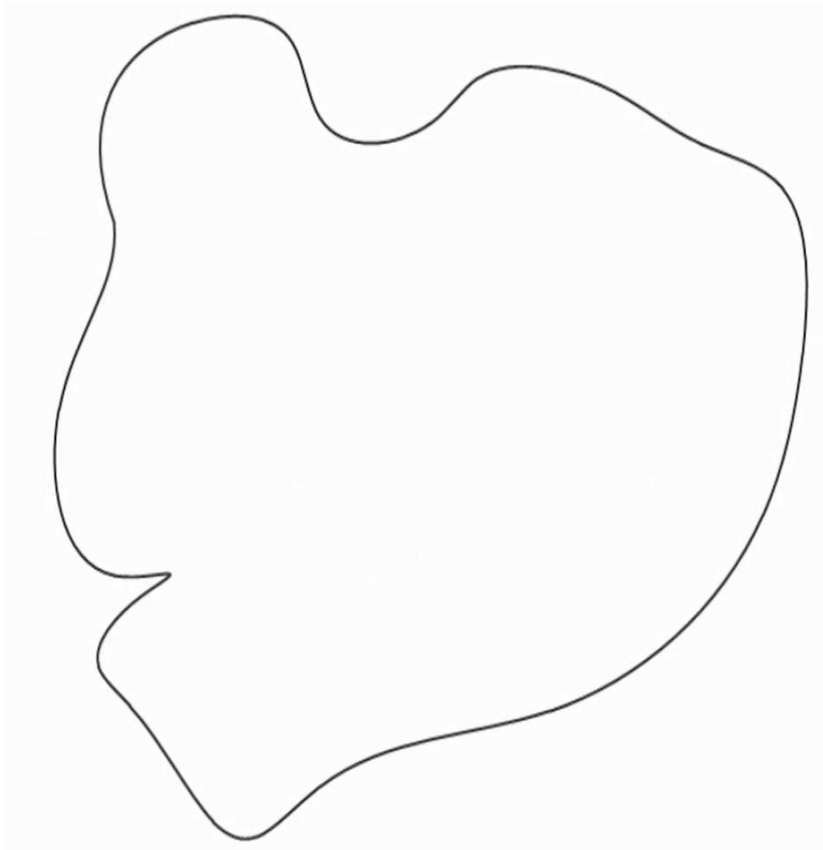
Functional equation to Differential equation

Example:

$$\pi = \int \left[\frac{1}{2} E A \left(\frac{du}{dx} \right)^2 - qu \right] dx \quad \text{Functional equation}$$

$$F = \left[\frac{1}{2} E A \left(\frac{du}{dx} \right)^2 - qu \right] \Rightarrow \left\{ \frac{\partial F}{\partial u} = -q \quad \frac{\partial F}{\partial u'} = AE \frac{du}{dx} \right\}$$

$$\text{Euler Equation} \Rightarrow -q - \frac{d}{dx} \left(AE \frac{du}{dx} \right) = 0 \Rightarrow \boxed{AE \frac{d^2 u}{dx^2} + q = 0}$$



Node:
Continuity between elements,
Define boundary conditions and loading

node

Ω_1

Ω_2

Ω_3

element

Ω

Galerkin method:

$$\int_{\Omega} N_i R_{\Omega} d\Omega = 0$$

Galerkin method in FEM:

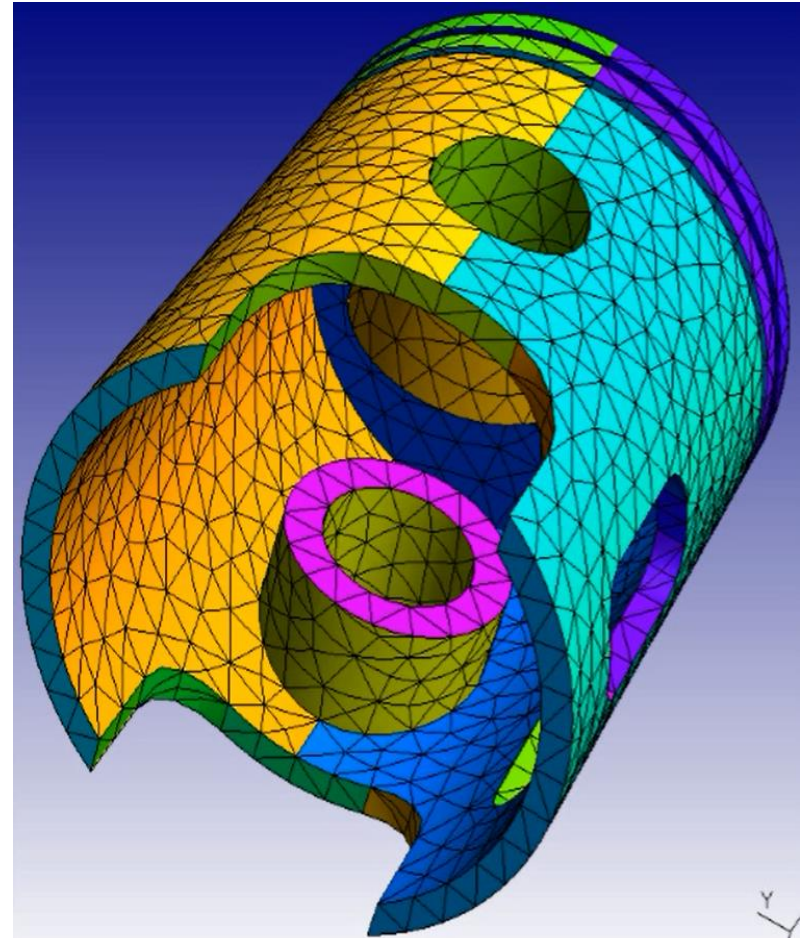
$$\int_{\Omega_1} N_i R_{\Omega_1} d\Omega_1 + \int_{\Omega_2} N_i R_{\Omega_2} d\Omega_2 + \int_{\Omega_3} N_i R_{\Omega_3} d\Omega_3 = 0$$

Galerkin method: by increasing N_i functions  errors are increasing

Galerkin method in FEM: number of element increasing with constant N_i  errors are not increasing

Meshing

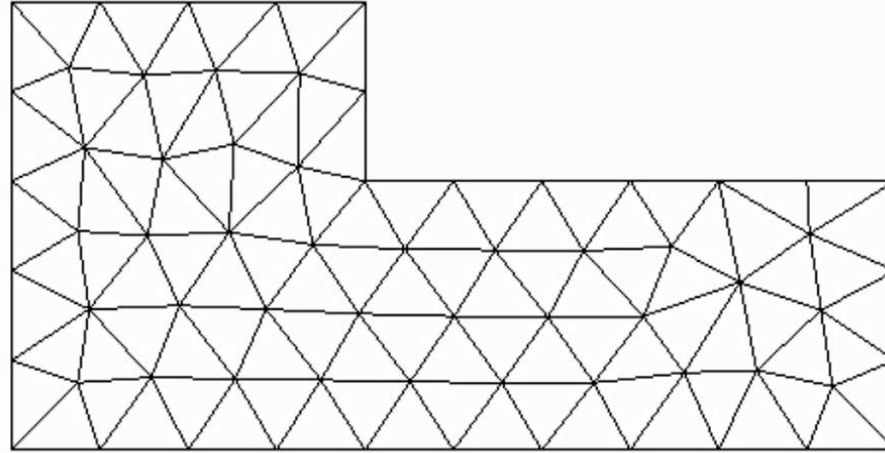
The domain is broken up into small pieces called elements.



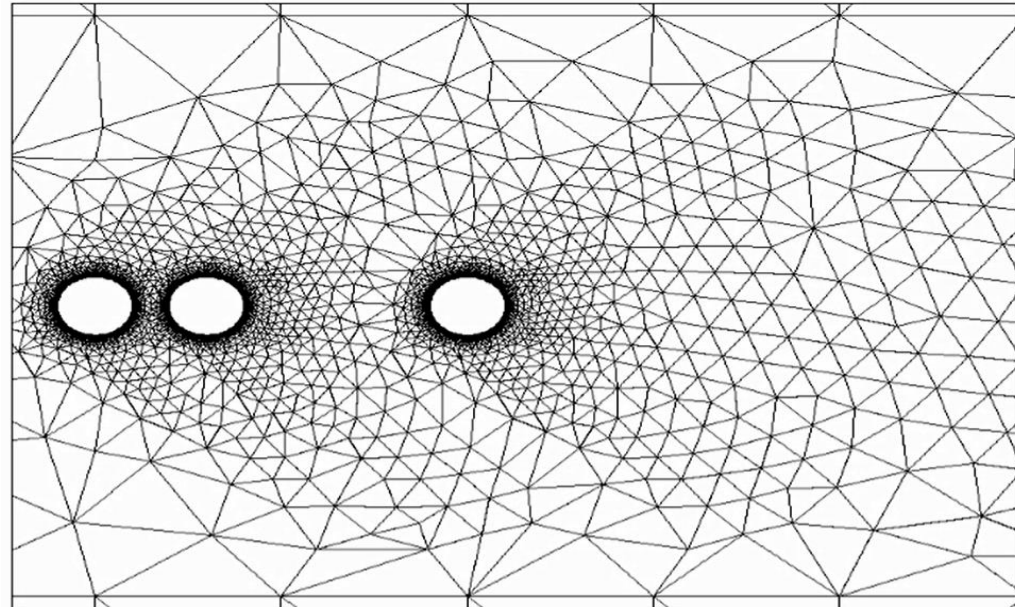
Meshing

Mesh types:

a) Uniform:



b) Non-uniform:



Meshing

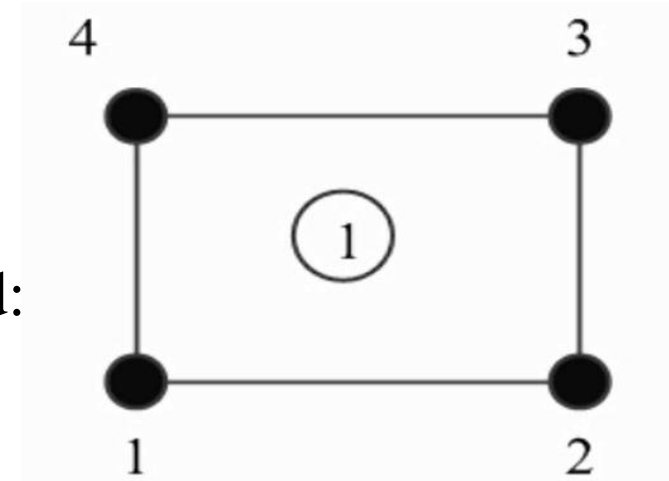
Information which are needed for meshing:

- Nodes coordinates.
- Number of nodes in each element.
- Connection between nodes in each element

Coordinates for nodes 1, 2, 3, 4.

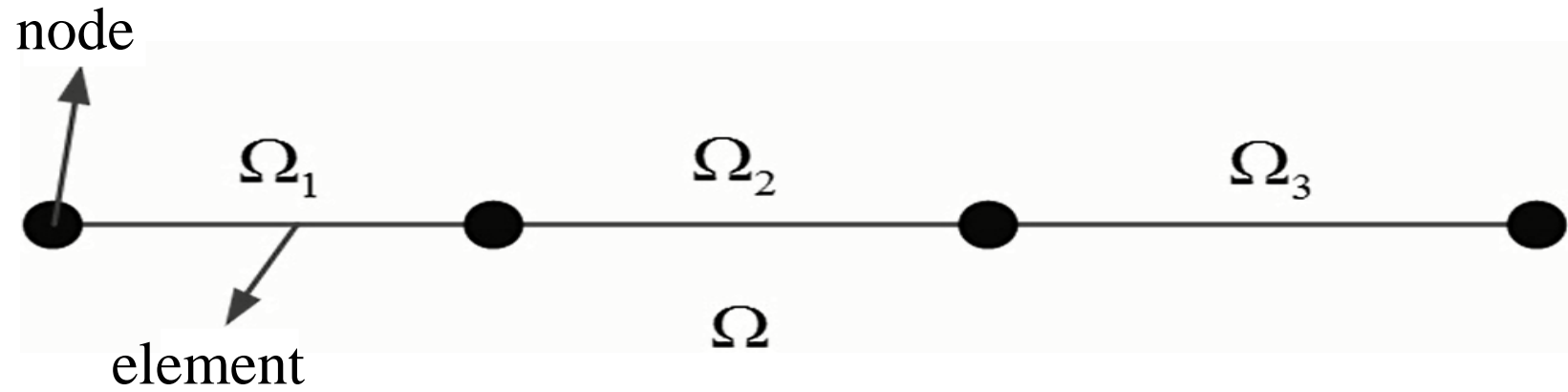
Four nodes for element 1.

In element 1 nodes are connected:
1-2-3-4.



Meshing

1-D meshing



N = number of elements

Meshing

1-D meshing:

Matlab example:

$$node_cord = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 2 & 0.1 & 0 & 0 \\ 3 & 0.2 & 0 & 0 \\ 4 & 0.3 & 0 & 0 \end{bmatrix}$$

$$element_inf = \begin{bmatrix} 1 & 1 & 2 \\ 2 & 2 & 3 \\ 3 & 3 & 4 \end{bmatrix}$$

Meshing

Matlab example:

```
node_inf=xlsread('node.xlsx','sheet1');  
element_inf=xlsread('element.xlsx','sheet1');
```

Meshing

Matlab example:

```
function [X_m,Y_m]=linearmesh(X,Y,N)
X_start=X(1,1);
Y_start=Y(1,1);
X_final=X(2,1);
Y_final=Y(2,1);
X_E=X_final-X_start;
Y_E=Y_final-Y_start;
L_E=sqrt(X_E^2+Y_E^2);
S=(Y_E/L_E);
C=(X_E/L_E);
dL=L_E/N;
dX=dL*C;
dY=dL*S;
%%
```

```
X_m(1,1)=X_start;
Y_m(1,1)=Y_start;
for i=2:N+1
    X_m(i,1)=X_m(i-1,1)+dX;
    Y_m(i,1)=Y_m(i-1,1)+dY;
end
end
```

Meshing

Command Window

```
>> X=[0;1];  
>> Y=[0;0];  
>> N=10;  
>> [X_m,Y_m]=linearmesh(X,Y,N);
```

fx >>

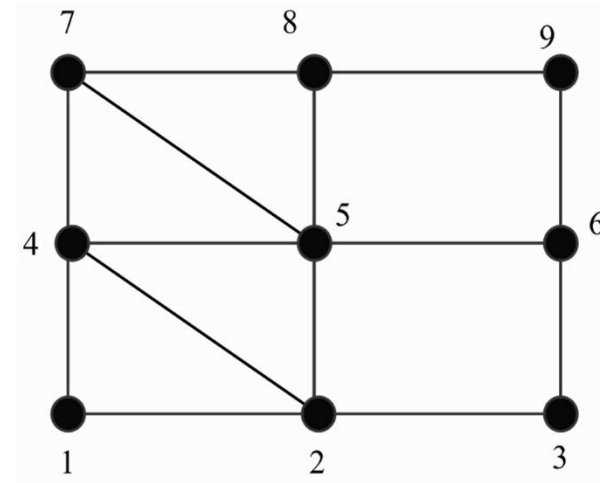
Meshing

2D Meshing:

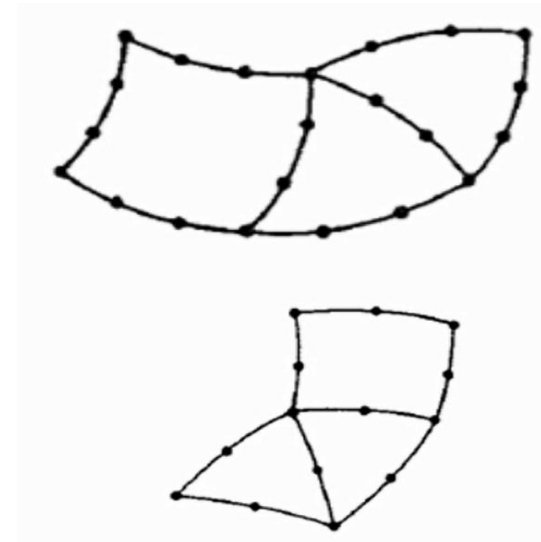
Linear 2D elements:

Rectangular elements

Triangular elements

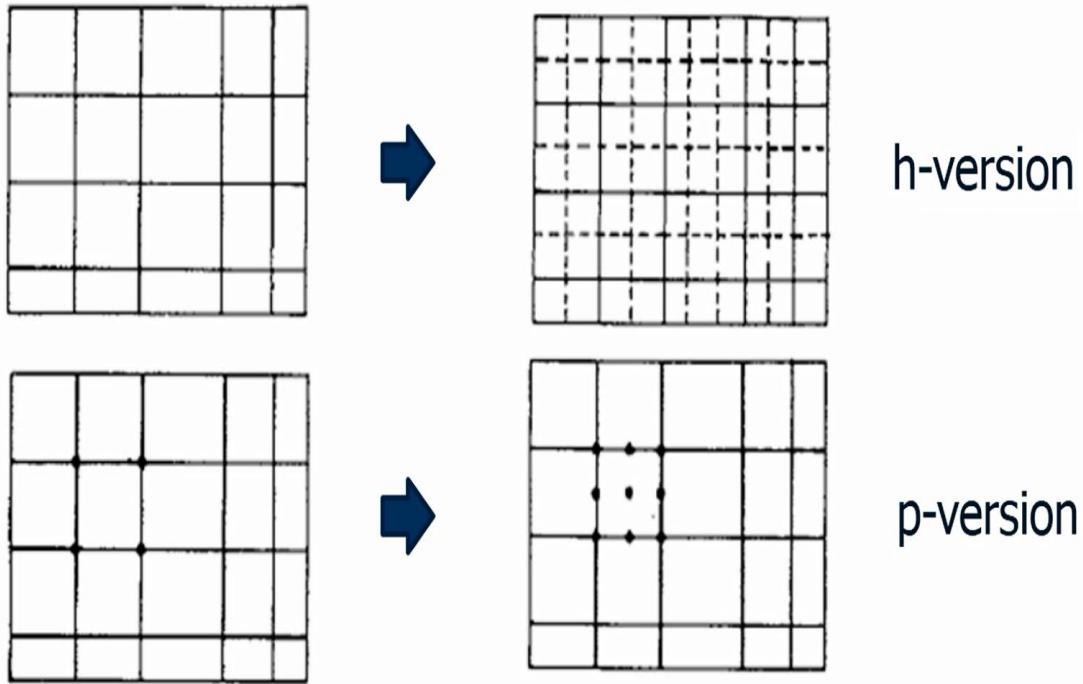


Non-linear 2D elements:

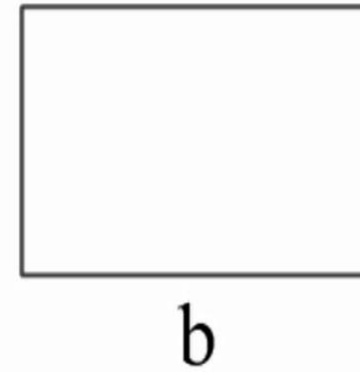


Meshing

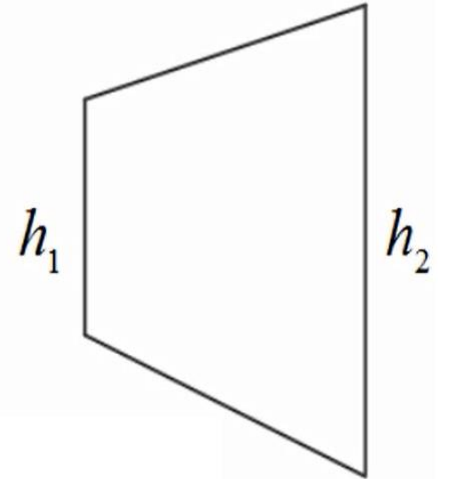
Mesh quality improvement:



Aspect ratio: $b/h \approx 1$



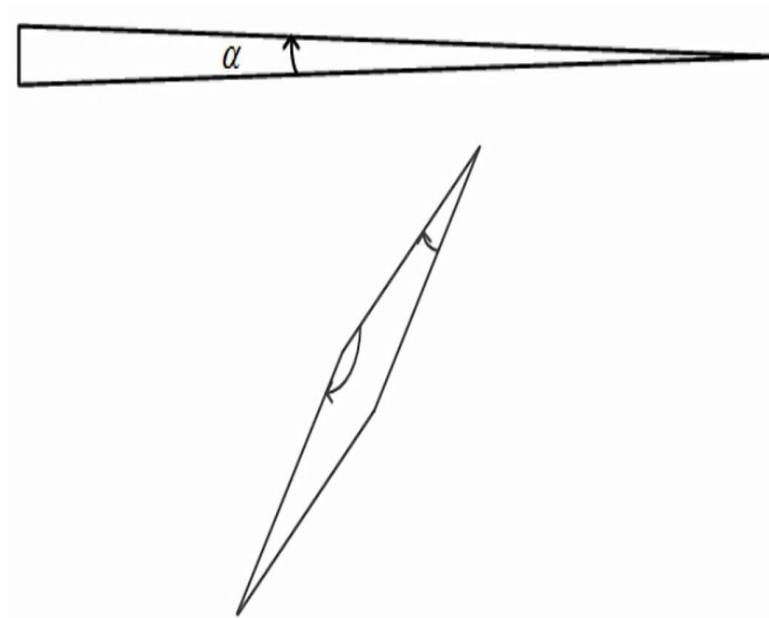
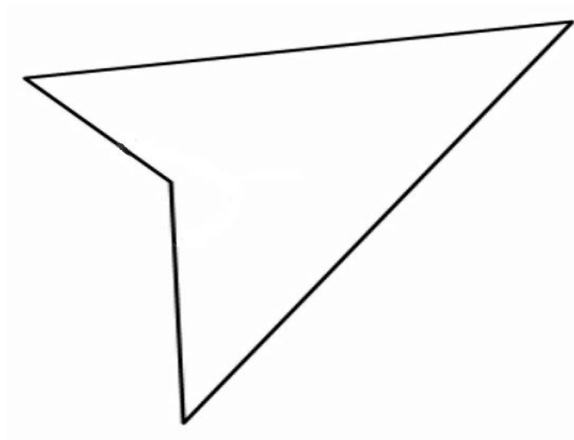
$$b \approx h$$



$$h_1 \approx h_2$$

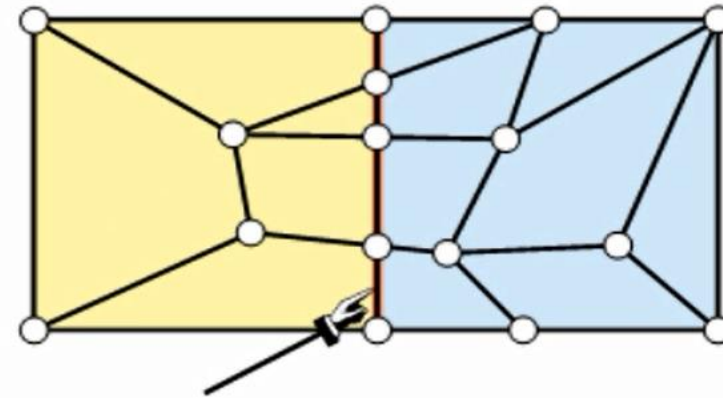
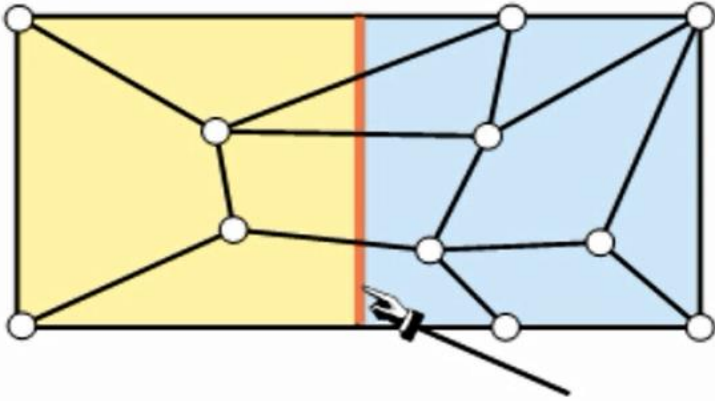
Meshing

Angel:



Meshing

For different layers and materials:



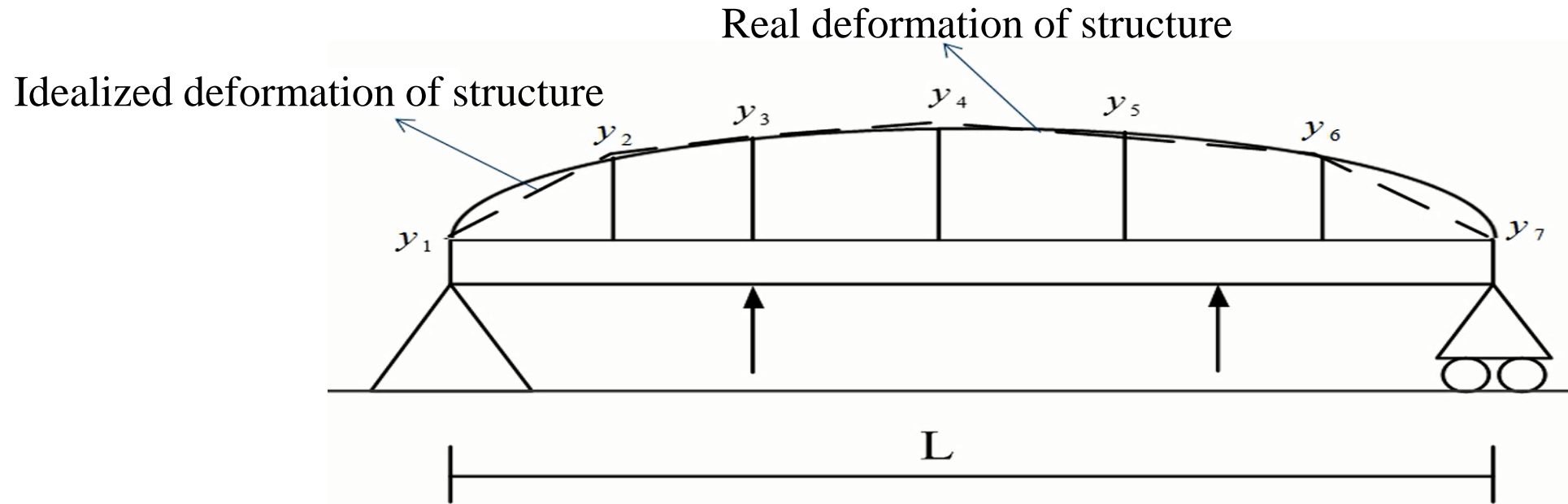
Sudden changes in elements



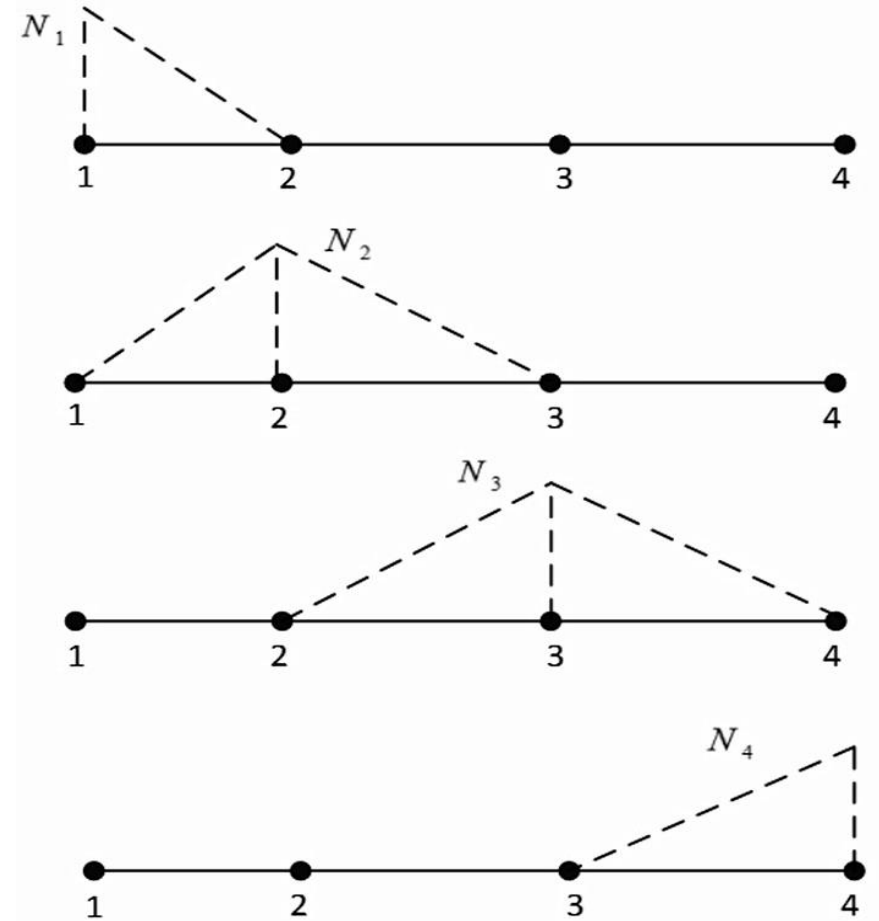
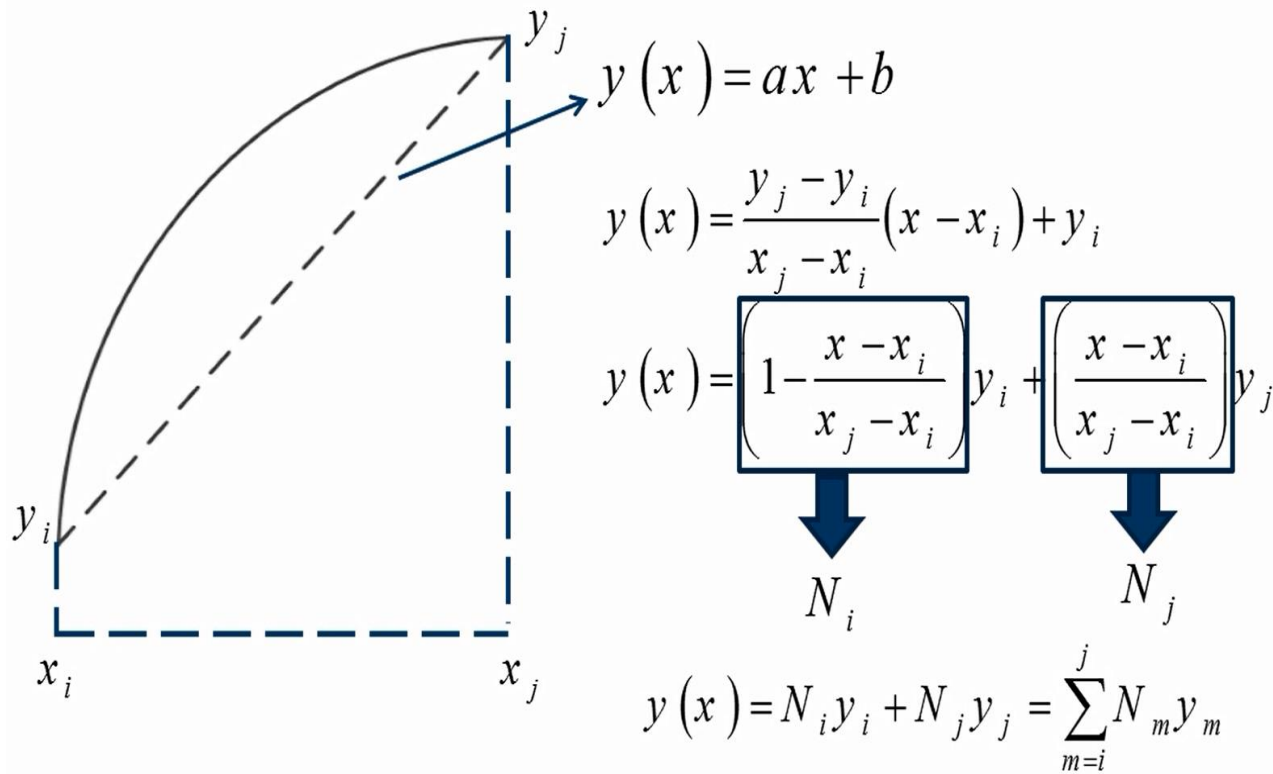
Gradual changes in elements



Function of linear deformation



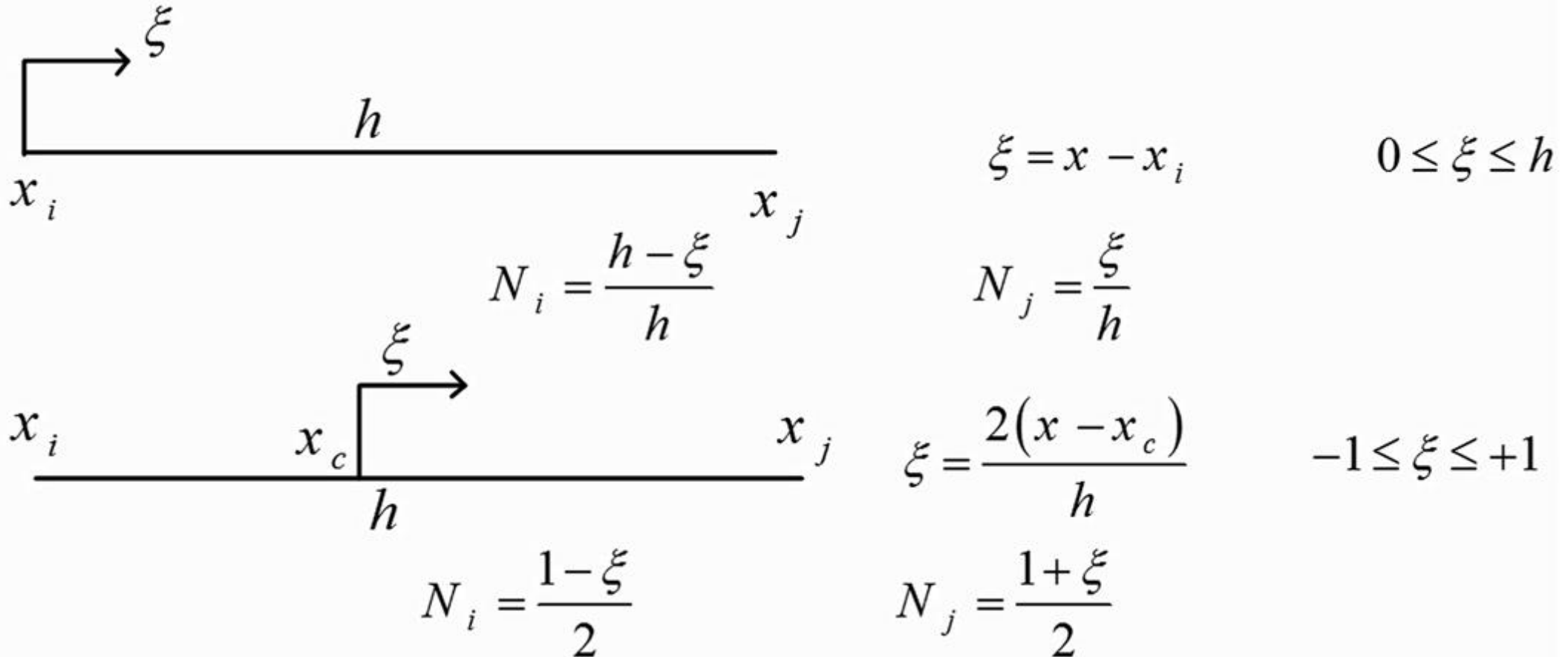
Function of linear deformation



$$N_1 = \left(1 - \frac{x - x_i}{x_j - x_i}\right)$$

First order shape function

First order shape function in local coordinate

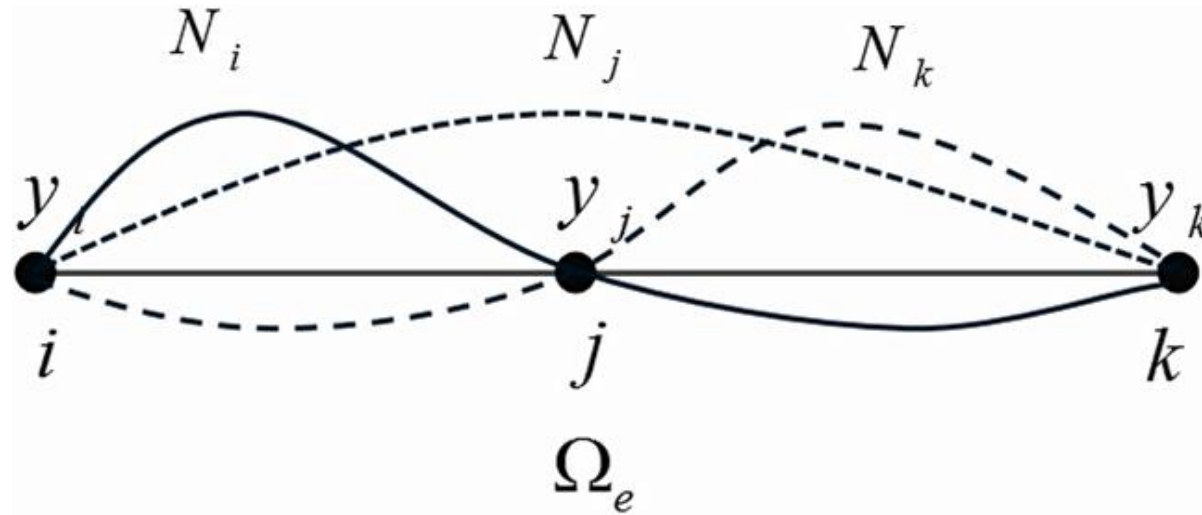


Lagrange Interpolation Function:

$$N_k = \frac{\prod_{\substack{i=1 \\ i \neq k}}^N (x_i - x)}{\prod_{\substack{i=1 \\ i \neq k}}^N (x_i - x_k)}$$

$$N_k = \frac{(x_\ell - x)(x_m - x) \dots (x_n - x)}{(x_\ell - x_k)(x_m - x_k) \dots (x_n - x_k)}$$

Second order shape functions



$$y(x) = N_i y_i + N_j y_j + N_k y_k$$

$$N_i, N_j, N_k$$

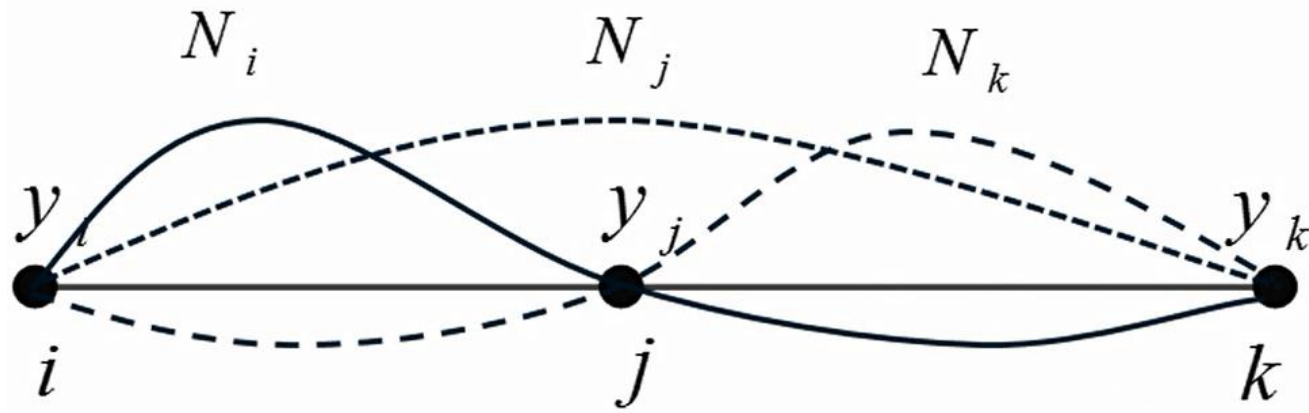
second order shape functions

$$N_i = \alpha_i + \beta_i x + \gamma_i x^2 \Rightarrow \begin{cases} N_i = 1 \text{ at } x = x_i \\ N_i = 0 \text{ at } x = x_j \\ N_i = 0 \text{ at } x = x_k \end{cases}$$

$$N_j = \alpha_j + \beta_j x + \gamma_j x^2 \Rightarrow \begin{cases} N_j = 0 \text{ at } x = x_i \\ N_j = 1 \text{ at } x = x_j \\ N_j = 0 \text{ at } x = x_k \end{cases}$$

$$N_k = \alpha_k + \beta_k x + \gamma_k x^2 \Rightarrow \begin{cases} N_k = 0 \text{ at } x = x_i \\ N_k = 0 \text{ at } x = x_j \\ N_k = 1 \text{ at } x = x_k \end{cases}$$

Lagrange Interpolation Function application:

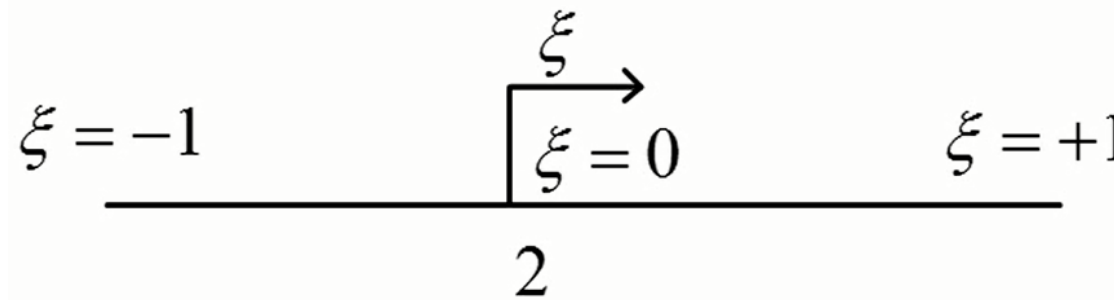


$$N_j = \frac{\prod_{\substack{m=1 \\ m \neq j}}^N (x_m - x)}{\prod_{\substack{m=1 \\ m \neq j}}^N (x_m - x_j)} = \frac{(x_i - x)(x_k - x)}{(x_i - x_j)(x_k - x_j)}$$

$$N_i = \frac{\prod_{\substack{m=1 \\ m \neq i}}^N (x_m - x)}{\prod_{\substack{m=1 \\ m \neq i}}^N (x_m - x_i)} = \frac{(x_j - x)(x_k - x)}{(x_j - x_i)(x_k - x_i)}$$

$$N_k = \frac{\prod_{\substack{m=1 \\ m \neq k}}^N (x_m - x)}{\prod_{\substack{m=1 \\ m \neq k}}^N (x_m - x_k)} = \frac{(x_i - x)(x_j - x)}{(x_i - x_k)(x_j - x_k)}$$

Second order shape function in local coordinate

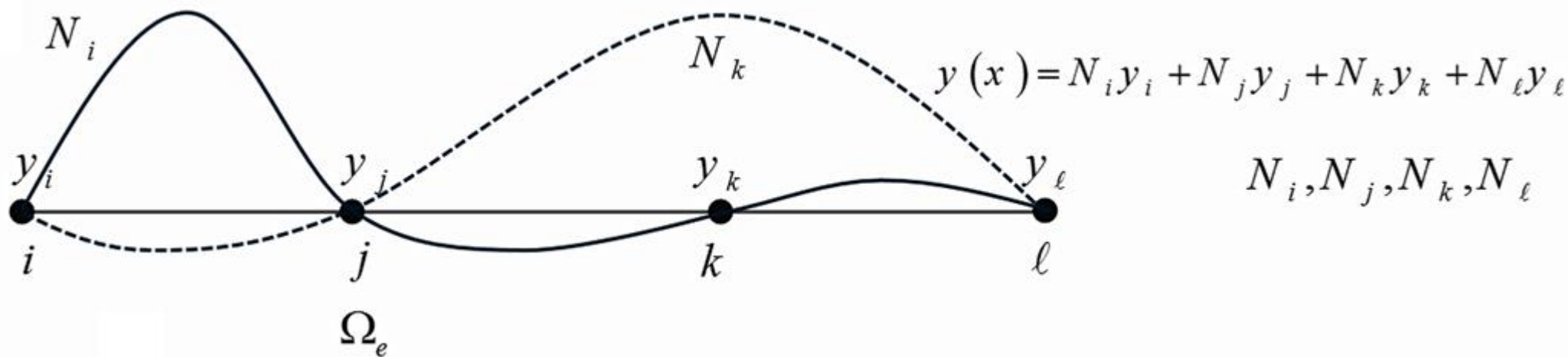

$$\xi = -1 \quad \xi = 0 \quad \xi = +1 \quad \xi = \frac{2(x - x_c)}{h} \quad -1 \leq \xi \leq +1$$

$$N_i = \frac{(\xi_j - \xi)(\xi_k - \xi)}{(\xi_j - \xi_i)(\xi_k - \xi_i)} = \frac{(0 - \xi)(1 - \xi)}{(0 - (-1))(1 - (-1))} = \frac{\xi(\xi - 1)}{2}$$

$$N_k = \frac{(\xi_i - \xi)(\xi_j - \xi)}{(\xi_i - \xi_k)(\xi_j - \xi_k)} = \frac{(-1 - \xi)(0 - \xi)}{(-1 - (1))(0 - (1))} = \frac{\xi(1 + \xi)}{2}$$

$$N_j = \frac{(\xi_i - \xi)(\xi_k - \xi)}{(\xi_i - \xi_j)(\xi_k - \xi_j)} = \frac{(-1 - \xi)(1 - \xi)}{(-1 - (0))(1 - (0))} = (1 - \xi)(1 + \xi)$$

Third order shape functions



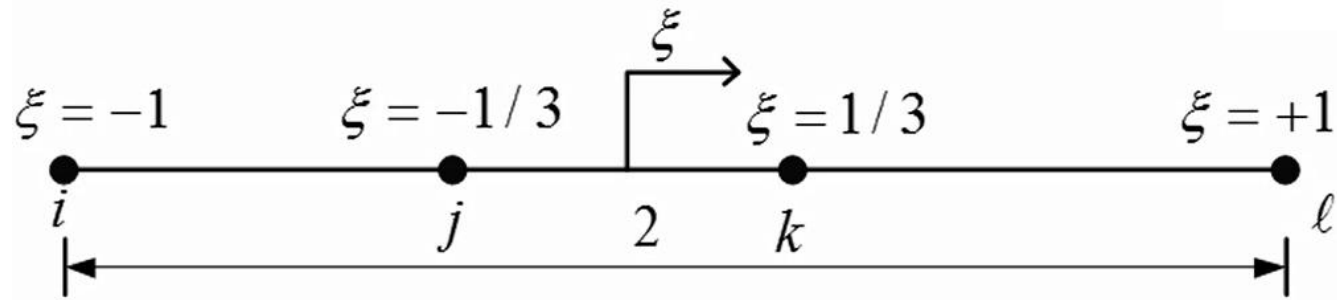
$$N_i = \frac{\prod_{\substack{m=1 \\ m \neq i}}^N (x_m - x)}{\prod_{\substack{m=1 \\ m \neq i}}^N (x_m - x_i)} = \frac{(x_j - x)(x_k - x)(x_\ell - x)}{(x_j - x_i)(x_k - x_i)(x_\ell - x_i)}$$

$$N_j = \frac{\prod_{\substack{m=1 \\ m \neq j}}^N (x_m - x)}{\prod_{\substack{m=1 \\ m \neq j}}^N (x_m - x_j)} = \frac{(x_i - x)(x_k - x)(x_\ell - x)}{(x_i - x_j)(x_k - x_j)(x_\ell - x_j)}$$

$$N_k = \frac{\prod_{\substack{m=1 \\ m \neq k}}^N (x_m - x)}{\prod_{\substack{m=1 \\ m \neq k}}^N (x_m - x_k)} = \frac{(x_i - x)(x_j - x)(x_\ell - x)}{(x_i - x_k)(x_j - x_k)(x_\ell - x_k)}$$

$$N_\ell = \frac{\prod_{\substack{m=1 \\ m \neq \ell}}^N (x_m - x)}{\prod_{\substack{m=1 \\ m \neq \ell}}^N (x_m - x_\ell)} = \frac{(x_i - x)(x_j - x)(x_k - x)}{(x_i - x_\ell)(x_j - x_\ell)(x_k - x_\ell)}$$

Third order shape function in local coordinate



$$\xi = \frac{2(x - x_c)}{h}$$
$$-1 \leq \xi \leq +1$$

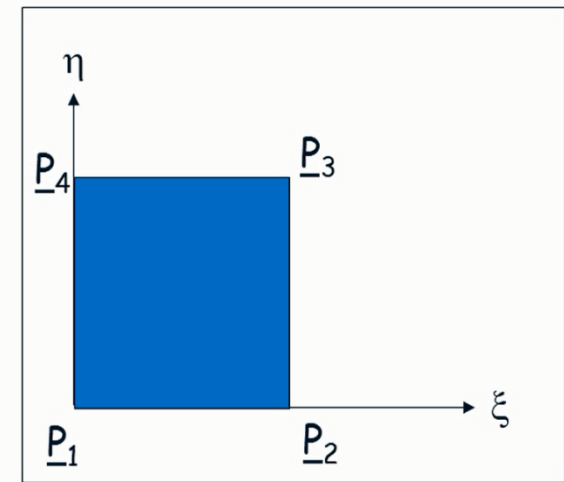
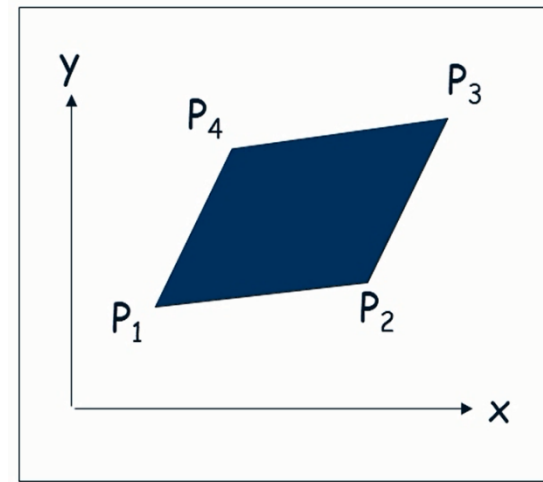
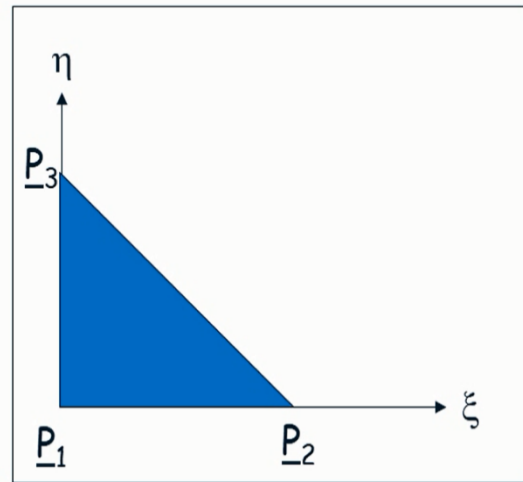
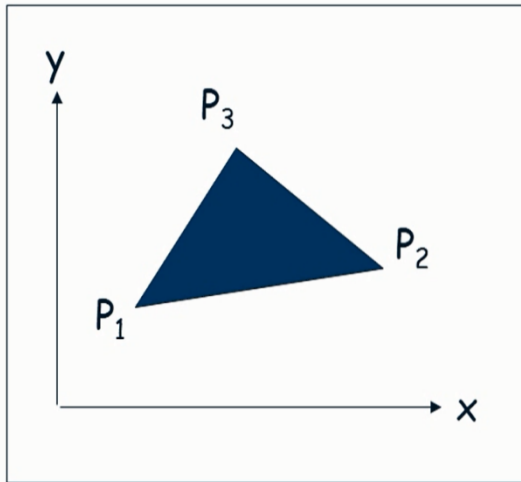
$$N_i = -\frac{9}{16} \left(\xi + \frac{1}{3} \right) \left(\xi - \frac{1}{3} \right) (\xi - 1)$$

$$N_\ell = \frac{9}{16} (\xi + 1) \left(\xi + \frac{1}{3} \right) \left(\xi - \frac{1}{3} \right)$$

$$N_j = \frac{27}{16} (\xi + 1) \left(\xi - \frac{1}{3} \right) (\xi - 1)$$

$$N_k = -\frac{27}{16} (\xi + 1) \left(\xi + \frac{1}{3} \right) (\xi - 1)$$

2D elements



Triangular elements

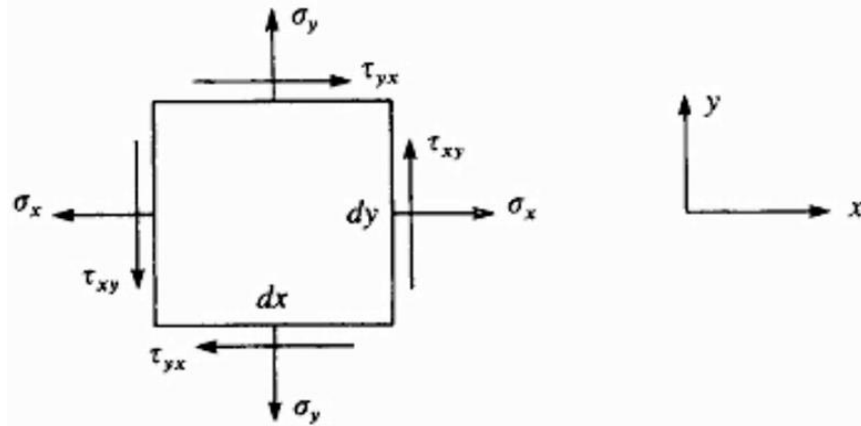
From strength of materials:

Stress tensor:

3D tensor:

$$[\sigma] = \begin{bmatrix} \sigma_{xx} & \sigma_{xy} & \sigma_{xz} \\ \sigma_{yx} & \sigma_{yy} & \sigma_{yz} \\ \sigma_{zx} & \sigma_{zy} & \sigma_{zz} \end{bmatrix} \quad \rightarrow \quad \begin{aligned} \sigma_{xy} &= \sigma_{yx} \\ \sigma_{yz} &= \sigma_{zy} \\ \sigma_{xz} &= \sigma_{zx} \end{aligned}$$

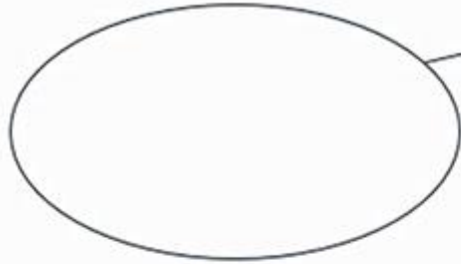
2D tensor:



The diagram shows a square stress element in a 2D Cartesian coordinate system with axes x and y . The element has dimensions dx and dy . On the left face, normal stress σ_x acts to the left and shear stress τ_{xy} acts downwards. On the right face, normal stress σ_x acts to the right and shear stress τ_{xy} acts upwards. On the top face, normal stress σ_y acts upwards and shear stress τ_{yx} acts to the right. On the bottom face, normal stress σ_y acts downwards and shear stress τ_{yx} acts to the left. To the right of the element is the 2D stress tensor:

$$[\sigma] = \begin{bmatrix} \sigma_{xx} & \sigma_{xy} \\ \sigma_{yx} & \sigma_{yy} \end{bmatrix}$$

From strength of materials:


 \vec{t}

$$t_x = \sigma_{xx}n_x + \sigma_{xy}n_y$$

$$t_y = \sigma_{xy}n_x + \sigma_{yy}n_y$$

(Traction)

$$\sum \vec{F} = 0$$



$$\frac{\partial \sigma_{xx}}{\partial x} + \frac{\partial \sigma_{xy}}{\partial y} + f_x = 0$$

$$\frac{\partial \sigma_{yx}}{\partial x} + \frac{\partial \sigma_{yy}}{\partial y} + f_y = 0$$

Statics

$$\frac{\partial \sigma_{xx}}{\partial x} + \frac{\partial \sigma_{xy}}{\partial y} + f_x = \rho \ddot{u}_x$$

$$\frac{\partial \sigma_{yx}}{\partial x} + \frac{\partial \sigma_{yy}}{\partial y} + f_y = \rho \ddot{u}_y$$

Dynamics

From strenght of materials:

Strain:

$$\varepsilon_{xx} = \frac{\partial u_x}{\partial x} \quad \varepsilon_{xy} = \frac{1}{2} \gamma_{xy} = \frac{1}{2} \left(\frac{\partial u_x}{\partial y} + \frac{\partial u_y}{\partial x} \right) \quad \varepsilon_{yy} = \frac{\partial u_y}{\partial y}$$

Hooke's law:

$$G = \frac{E}{2(1+\nu)} \quad \lambda = \frac{\nu E}{(1+\nu)(1-2\nu)}$$

$$\varepsilon_{xx} = \frac{\sigma_{xx}}{E} - \nu \frac{\sigma_{yy}}{E} - \nu \frac{\sigma_{zz}}{E}$$

$$\varepsilon_{yy} = -\nu \frac{\sigma_{xx}}{E} + \frac{\sigma_{yy}}{E} - \nu \frac{\sigma_{zz}}{E}$$

$$\varepsilon_{zz} = -\nu \frac{\sigma_{xx}}{E} - \nu \frac{\sigma_{yy}}{E} + \frac{\sigma_{zz}}{E}$$

$$\tau_{xy} = G \gamma_{xy} \quad \tau_{yz} = G \gamma_{yz} \quad \tau_{zx} = G \gamma_{zx}$$

$$\sigma_{xx} = \lambda e + 2G \varepsilon_{xx}$$

$$\sigma_{yy} = \lambda e + 2G \varepsilon_{yy}$$

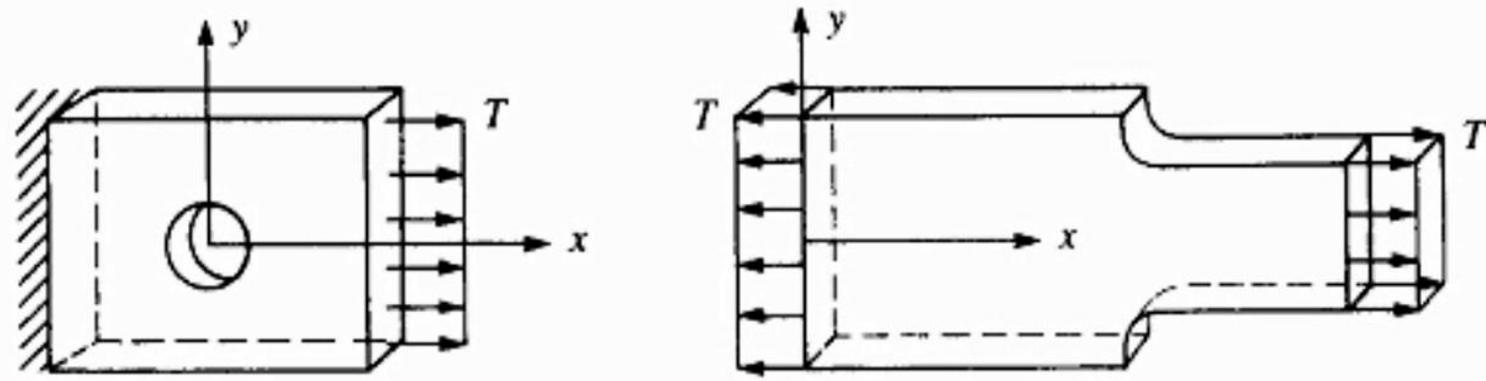
$$\sigma_{zz} = \lambda e + 2G \varepsilon_{zz}$$

$$e = \varepsilon_{xx} + \varepsilon_{yy} + \varepsilon_{zz}$$

From strength of materials:

Plane stresses:

$$\sigma_z = \tau_{xz} = \tau_{yz} = 0$$



$$\begin{Bmatrix} \sigma_x \\ \sigma_y \\ \tau_{xy} \end{Bmatrix} = \begin{bmatrix} C_{11} & C_{12} & 0 \\ C_{12} & C_{22} & 0 \\ 0 & 0 & C_{33} \end{bmatrix} \begin{Bmatrix} \varepsilon_x \\ \varepsilon_y \\ \gamma_{xy} \end{Bmatrix}$$



$$\begin{Bmatrix} \sigma_x \\ \sigma_y \\ \tau_{xy} \end{Bmatrix} = \begin{bmatrix} \frac{E}{1-\nu^2} & \frac{\nu E}{1-\nu^2} & 0 \\ \frac{\nu E}{1-\nu^2} & \frac{E}{1-\nu^2} & 0 \\ 0 & 0 & \frac{E}{2(1+\nu)} \end{bmatrix} \begin{Bmatrix} \varepsilon_x \\ \varepsilon_y \\ \gamma_{xy} \end{Bmatrix}$$

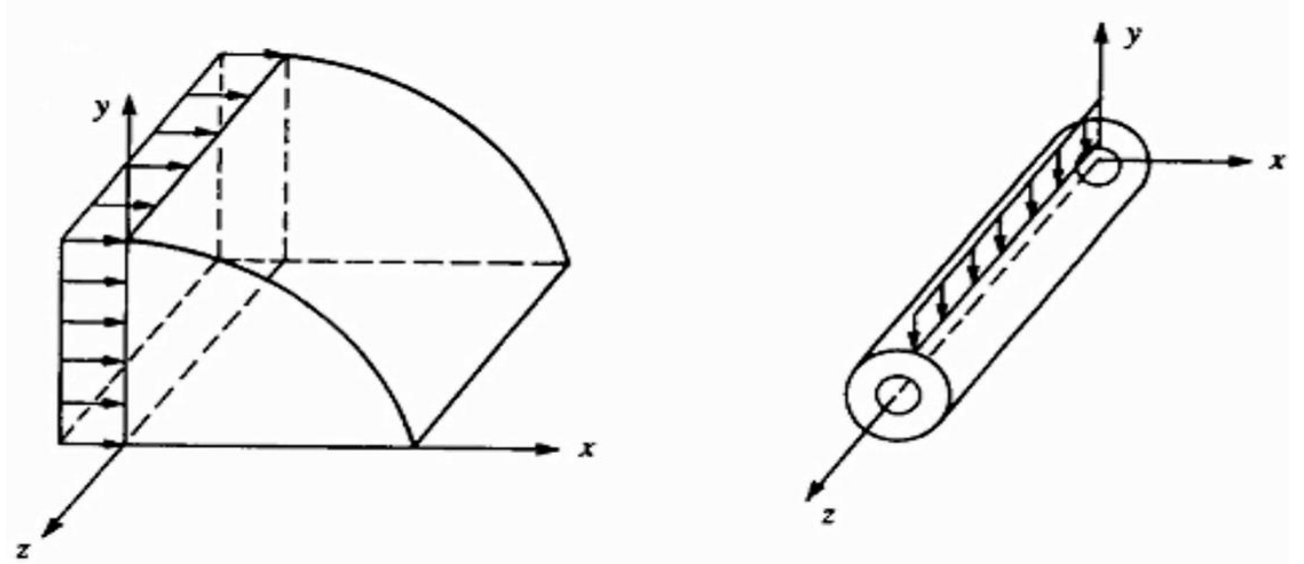
$$\begin{Bmatrix} \sigma_x \\ \sigma_y \\ \tau_{xy} \end{Bmatrix} = [D] \begin{Bmatrix} \varepsilon_x \\ \varepsilon_y \\ \gamma_{xy} \end{Bmatrix}$$

$$[D] = \frac{E}{1-\nu^2} \begin{bmatrix} 1 & \nu & 0 \\ \nu & 1 & 0 \\ 0 & 0 & \frac{1-\nu}{2} \end{bmatrix}$$

From strength of materials:

Plane strains:

$$\epsilon_z = \gamma_{xz} = \gamma_{yz} = 0$$

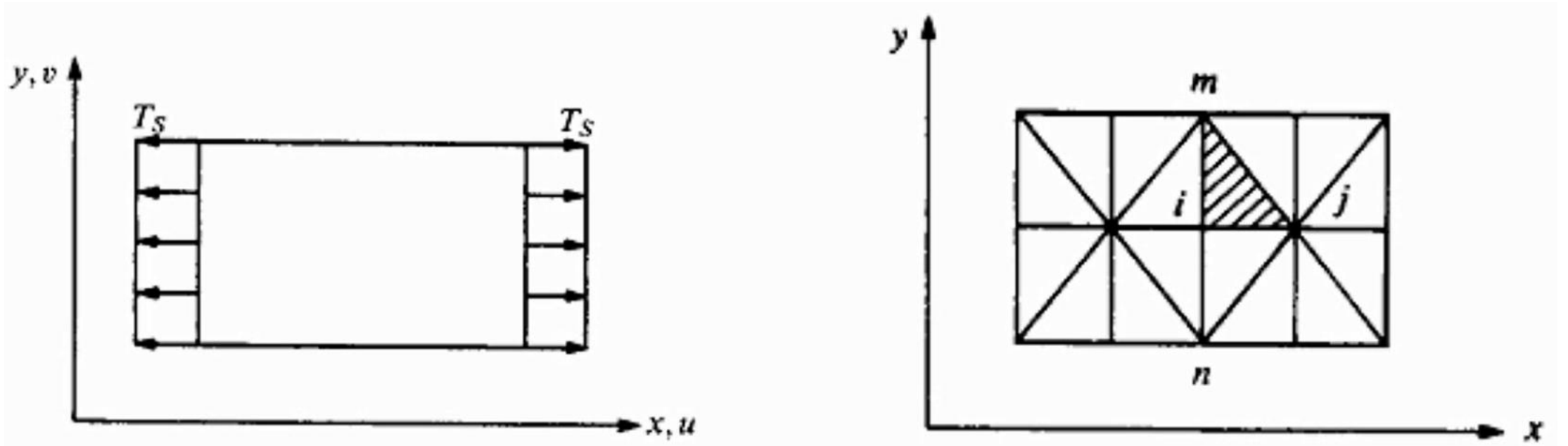


$$\begin{Bmatrix} \sigma_x \\ \sigma_y \\ \tau_{xy} \end{Bmatrix} = \begin{bmatrix} C_{11} & C_{12} & 0 \\ C_{12} & C_{22} & 0 \\ 0 & 0 & C_{33} \end{bmatrix} \begin{Bmatrix} \epsilon_x \\ \epsilon_y \\ \gamma_{xy} \end{Bmatrix} \Rightarrow \begin{Bmatrix} \sigma_x \\ \sigma_y \\ \tau_{xy} \end{Bmatrix} = \begin{bmatrix} \frac{(1-\nu)E}{(1+\nu)(1-2\nu)} & \frac{\nu E}{(1+\nu)(1-2\nu)} & 0 \\ \frac{\nu E}{(1+\nu)(1-2\nu)} & \frac{(1-\nu)E}{(1+\nu)(1-2\nu)} & 0 \\ 0 & 0 & \frac{E}{2(1+\nu)} \end{bmatrix} \begin{Bmatrix} \epsilon_x \\ \epsilon_y \\ \gamma_{xy} \end{Bmatrix}$$

$$\begin{Bmatrix} \sigma_x \\ \sigma_y \\ \tau_{xy} \end{Bmatrix} = [D] \begin{Bmatrix} \epsilon_x \\ \epsilon_y \\ \gamma_{xy} \end{Bmatrix}$$

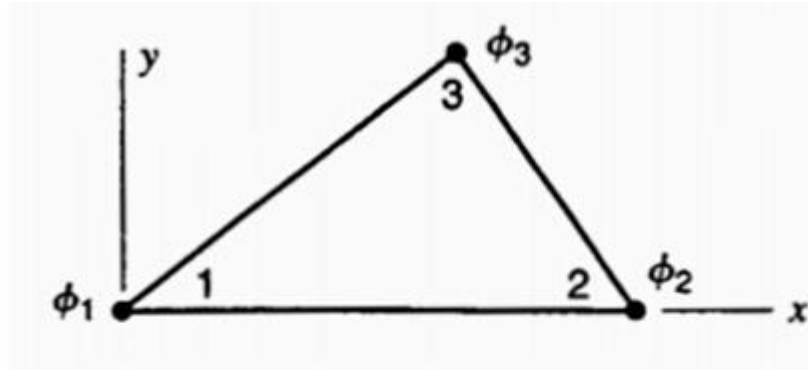
Finite Elements for 2-D Problems

Constant Strain Triangle (CST): This is the simplest 2-D element, which is also called linear triangular element

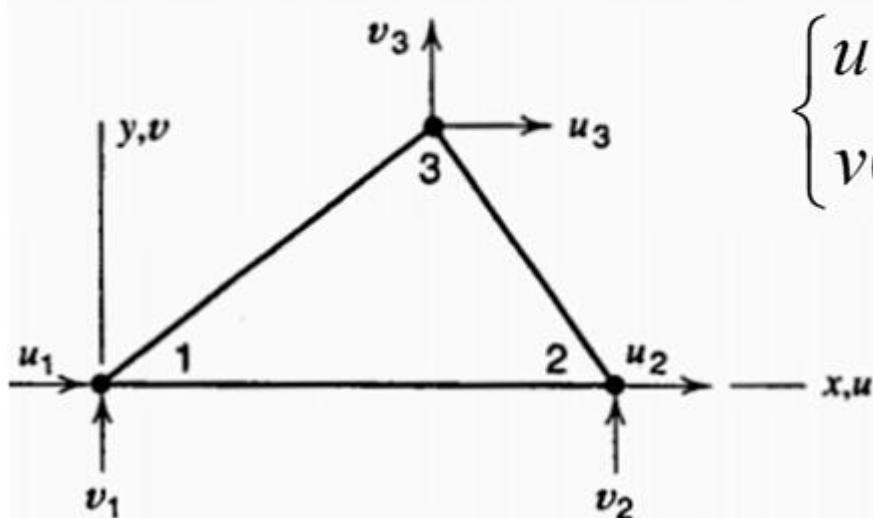


Finite Elements for 2-D Problems

Constant Strain Triangle (CST):

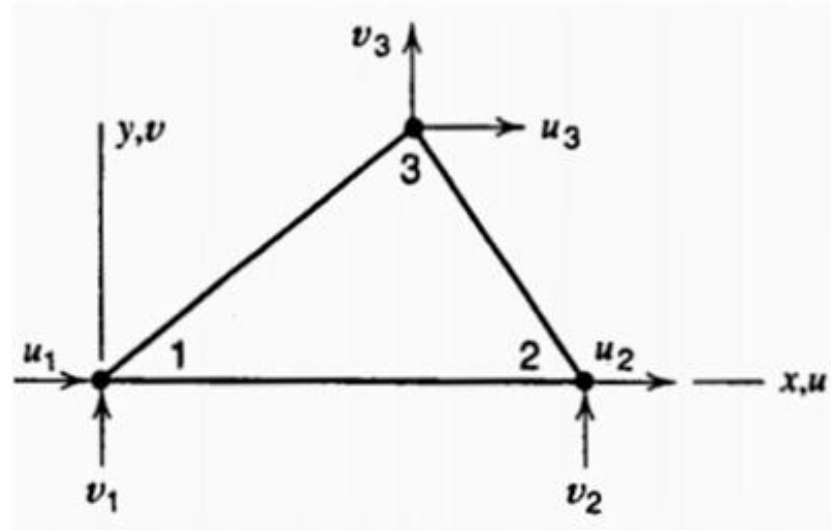


$$\{d\} = \begin{Bmatrix} \underline{d}_i \\ \underline{d}_j \\ \underline{d}_m \end{Bmatrix} = \begin{Bmatrix} u_i \\ v_i \\ u_j \\ v_j \\ u_m \\ v_m \end{Bmatrix}$$



$$\begin{cases} u(x, y) = c_1 + c_2x + c_3y = \phi_1u_1 + \phi_2u_2 + \phi_3u_3 \\ v(x, y) = c_5 + c_6x + c_7y = \phi_1v_1 + \phi_2v_2 + \phi_3v_3 \end{cases}$$

Displacement domain:



$$\{\psi\} = \begin{Bmatrix} a_1 + a_2x + a_3y \\ a_4 + a_5x + a_6y \end{Bmatrix} = \begin{bmatrix} 1 & x & y & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & x & y \end{bmatrix} \begin{Bmatrix} a_1 \\ a_2 \\ a_3 \\ a_4 \\ a_5 \\ a_6 \end{Bmatrix}$$

$$u_i = u(x_i, y_i) = a_1 + a_2x_i + a_3y_i$$

$$u_j = u(x_j, y_j) = a_1 + a_2x_j + a_3y_j$$

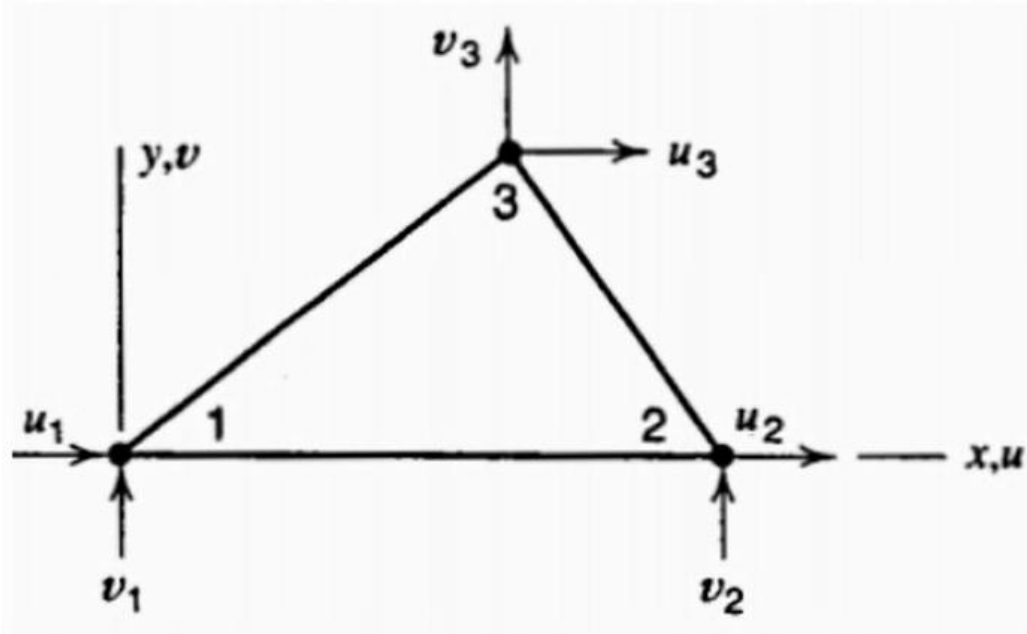
$$u_m = u(x_m, y_m) = a_1 + a_2x_m + a_3y_m$$

$$v_i = v(x_i, y_i) = a_4 + a_5x_i + a_6y_i$$

$$v_j = v(x_j, y_j) = a_4 + a_5x_j + a_6y_j$$

$$v_m = v(x_m, y_m) = a_4 + a_5x_m + a_6y_m$$

Shape function:



$$\begin{Bmatrix} u_i \\ u_j \\ u_m \end{Bmatrix} = \begin{bmatrix} 1 & x_i & y_i \\ 1 & x_j & y_j \\ 1 & x_m & y_m \end{bmatrix} \begin{Bmatrix} a_1 \\ a_2 \\ a_3 \end{Bmatrix}$$

$$\{a\} = [x]^{-1} \{u\}$$

$$2A = \begin{vmatrix} 1 & x_i & y_i \\ 1 & x_j & y_j \\ 1 & x_m & y_m \end{vmatrix}$$

$$[x]^{-1} = \frac{1}{2A} \begin{bmatrix} \alpha_i & \alpha_j & \alpha_m \\ \beta_i & \beta_j & \beta_m \\ \gamma_i & \gamma_j & \gamma_m \end{bmatrix}$$

$$[x]^{-1} = \frac{1}{2A} \begin{bmatrix} \alpha_i & \alpha_j & \alpha_m \\ \beta_i & \beta_j & \beta_m \\ \gamma_i & \gamma_j & \gamma_m \end{bmatrix}$$

$$\alpha_i = x_j y_m - y_j x_m$$

$$\alpha_j = y_i x_m - x_i y_m$$

$$\alpha_m = x_i y_j - y_i x_j$$

$$\beta_i = y_j - y_m$$

$$\beta_j = y_m - y_i$$

$$\beta_m = y_i - y_j$$

$$\gamma_i = x_m - x_j$$

$$\gamma_j = x_i - x_m$$

$$\gamma_m = x_j - x_i$$

$$\begin{Bmatrix} a_1 \\ a_2 \\ a_3 \end{Bmatrix} = \frac{1}{2A} \begin{bmatrix} \alpha_i & \alpha_j & \alpha_m \\ \beta_i & \beta_j & \beta_m \\ \gamma_i & \gamma_j & \gamma_m \end{bmatrix} \begin{Bmatrix} u_i \\ u_j \\ u_m \end{Bmatrix}$$

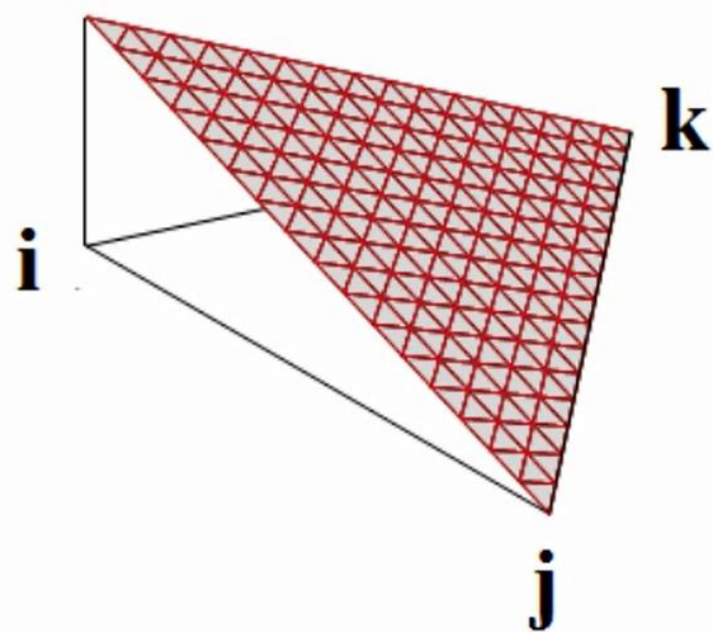
$$\left\{ \begin{matrix} \{u\} = [1 \quad x \quad y] \begin{Bmatrix} a_1 \\ a_2 \\ a_3 \end{Bmatrix} \\ \begin{Bmatrix} a_1 \\ a_2 \\ a_3 \end{Bmatrix} = \frac{1}{2A} \begin{bmatrix} \alpha_i & \alpha_j & \alpha_m \\ \beta_i & \beta_j & \beta_m \\ \gamma_i & \gamma_j & \gamma_m \end{bmatrix} \begin{Bmatrix} u_i \\ u_j \\ u_m \end{Bmatrix} \end{matrix} \right\} \Rightarrow \{u\} = \frac{1}{2A} [1 \quad x \quad y] \begin{bmatrix} \alpha_i & \alpha_j & \alpha_m \\ \beta_i & \beta_j & \beta_m \\ \gamma_i & \gamma_j & \gamma_m \end{bmatrix} \begin{Bmatrix} u_i \\ u_j \\ u_m \end{Bmatrix}$$

$$\Rightarrow \{u\} = \frac{1}{2A} [1 \quad x \quad y] \begin{Bmatrix} \alpha_i u_i + \alpha_j u_j + \alpha_m u_m \\ \beta_i u_i + \beta_j u_j + \beta_m u_m \\ \gamma_i u_i + \gamma_j u_j + \gamma_m u_m \end{Bmatrix}$$

$$\Rightarrow u(x, y) = \frac{1}{2A} \{ (\alpha_i + \beta_i x + \gamma_i y) u_i + (\alpha_j + \beta_j x + \gamma_j y) u_j + (\alpha_m + \beta_m x + \gamma_m y) u_m \}$$

$$u(x, y) = \frac{1}{2A} \{ (\alpha_i + \beta_i x + \gamma_i y) u_i + (\alpha_j + \beta_j x + \gamma_j y) u_j + (\alpha_m + \beta_m x + \gamma_m y) u_m \}$$

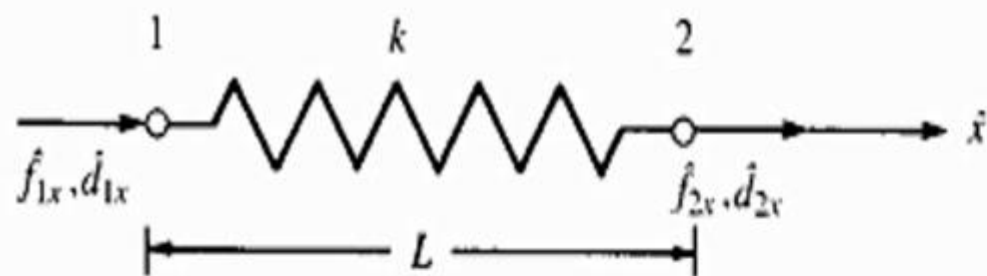
$$N_i = \frac{1}{2A} (\alpha_i + \beta_i x + \gamma_i y) \quad N_j = \frac{1}{2A} (\alpha_j + \beta_j x + \gamma_j y) \quad N_m = \frac{1}{2A} (\alpha_m + \beta_m x + \gamma_m y)$$



$$\left\{ \begin{array}{l} u(x, y) = N_i u_i + N_j u_j + N_m u_m \\ v(x, y) = N_i v_i + N_j v_j + N_m v_m \end{array} \right\} \Rightarrow \{\psi\} = \left\{ \begin{array}{l} u(x, y) \\ v(x, y) \end{array} \right\} = \left\{ \begin{array}{l} N_i u_i + N_j u_j + N_m u_m \\ N_i v_i + N_j v_j + N_m v_m \end{array} \right\}$$

$$\{\psi\} = \underbrace{\begin{bmatrix} N_i & 0 & N_j & 0 & N_m & 0 \\ 0 & N_i & 0 & N_j & 0 & N_m \end{bmatrix}}_{[N]} \left\{ \begin{array}{l} u_i \\ v_i \\ u_j \\ v_j \\ u_m \\ v_m \end{array} \right\}$$

$$\{\psi\} = [N] \{d\}$$



$$T = -\hat{f}_{1x} = k(\hat{d}_{2x} - \hat{d}_{1x}) \quad \Rightarrow \quad \hat{f}_{1x} = k(\hat{d}_{1x} - \hat{d}_{2x})$$

$$T = \hat{f}_{2x} = k(\hat{d}_{2x} - \hat{d}_{1x}) \quad \Rightarrow \quad \hat{f}_{2x} = k(\hat{d}_{2x} - \hat{d}_{1x})$$

$$\begin{Bmatrix} \hat{f}_{1x} \\ \hat{f}_{2x} \end{Bmatrix} = \begin{bmatrix} k & -k \\ -k & k \end{bmatrix} \begin{Bmatrix} \hat{d}_{1x} \\ \hat{d}_{2x} \end{Bmatrix}$$

force

degree
of
freedom

stiffness matrix

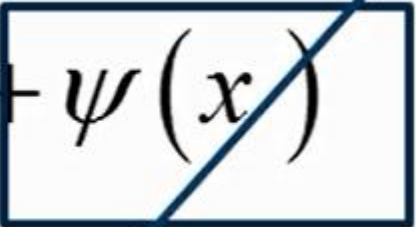
$$\underline{\hat{k}} = \begin{bmatrix} k & -k \\ -k & k \end{bmatrix}$$

Force = Degrees of freedom \times  Stiffness Matrix

differential equation (in Ω domain)

$$L(u) + A = 0 \text{ on } \Omega$$

Galerkin method:

$$\tilde{u}(x) = \sum_{i=1}^n a_i N_i(x) + \psi(x)$$



Minimization of errors in Ω domain:

$$L(\tilde{u}) + A = R_{\Omega}$$

$$\int_{\Omega_1} N_i R_{\Omega} d\Omega_1 + \int_{\Omega_2} N_i R_{\Omega} d\Omega_2 + \int_{\Omega_3} N_i R_{\Omega} d\Omega_3 = 0$$

Galerkin method in FEM:

$$\int_{x_1}^{x_2} N_i \boxed{R_\Omega} dx + \int_{x_2}^{x_3} N_i R_\Omega dx + \dots + \int_{x_{n-1}}^{x_n} N_i R_\Omega dx = 0$$


 $L(\tilde{u}) + A = R_\Omega \quad \Rightarrow \quad L\left(\sum_{i=1}^n a_i N_i(x)\right) + A = R_\Omega$

$$\int_{x_1}^{x_2} N_i \left[L\left(\sum_{i=1}^n a_i N_i(x)\right) + A \right] dx + \int_{x_2}^{x_3} N_i \left[L\left(\sum_{i=1}^n a_i N_i(x)\right) + A \right] dx + \dots$$

$$+ \int_{x_{n-1}}^{x_n} N_i \left[L\left(\sum_{i=1}^n a_i N_i(x)\right) + A \right] dx = 0 \quad i = 1, \dots, n$$

$$\int_{x_1}^{x_2} N_i \left[L\left(\sum_{i=1}^n a_i N_i(x)\right) \right] dx + \int_{x_2}^{x_3} N_i \left[L\left(\sum_{i=1}^n a_i N_i(x)\right) \right] dx$$

$$= - \left[\int_{x_1}^{x_2} N_i [L(A)] dx + \int_{x_2}^{x_3} N_i [L(A)] dx \right]$$

For two elements: $\tilde{u}(x) = a_1 N_1(x) + a_2 N_2(x)$

$$k_{i1} \int_{x_1}^{x_2} N_i \left[L(a_1 N_1(x) + a_2 N_2(x)) \right] dx + \int_{x_2}^{x_3} N_i \left[L(a_1 N_1(x) + a_2 N_2(x)) \right] dx \quad k_{i2}$$

$$F_i = \int_{x_1}^{x_2} N_i \left[L(A) \right] dx - \int_{x_2}^{x_3} N_i \left[L(A) \right] dx$$

$$\begin{aligned} i=1 & \Rightarrow k_{11}u_1 + k_{12}u_2 = F_1 \\ i=2 & \Rightarrow k_{21}u_1 + k_{22}u_2 = F_2 \end{aligned} \Rightarrow \begin{bmatrix} k_{11} & k_{12} \\ k_{21} & k_{22} \end{bmatrix} \begin{Bmatrix} u_1 \\ u_2 \end{Bmatrix} = \begin{Bmatrix} F_1 \\ F_2 \end{Bmatrix}$$

Applying boundary conditions:

- Elimination method
- Penalty method
- Lagrange multiplier method

Applying boundary conditions with elimination method

$$[K]_{n \times n} = \begin{bmatrix} k_{11} & \cdot & \cdot & \cdot & k_{1n} \\ k_{21} & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ k_{n1} & & & & k_{nn} \end{bmatrix}$$

Constraint on the first degree of freedom



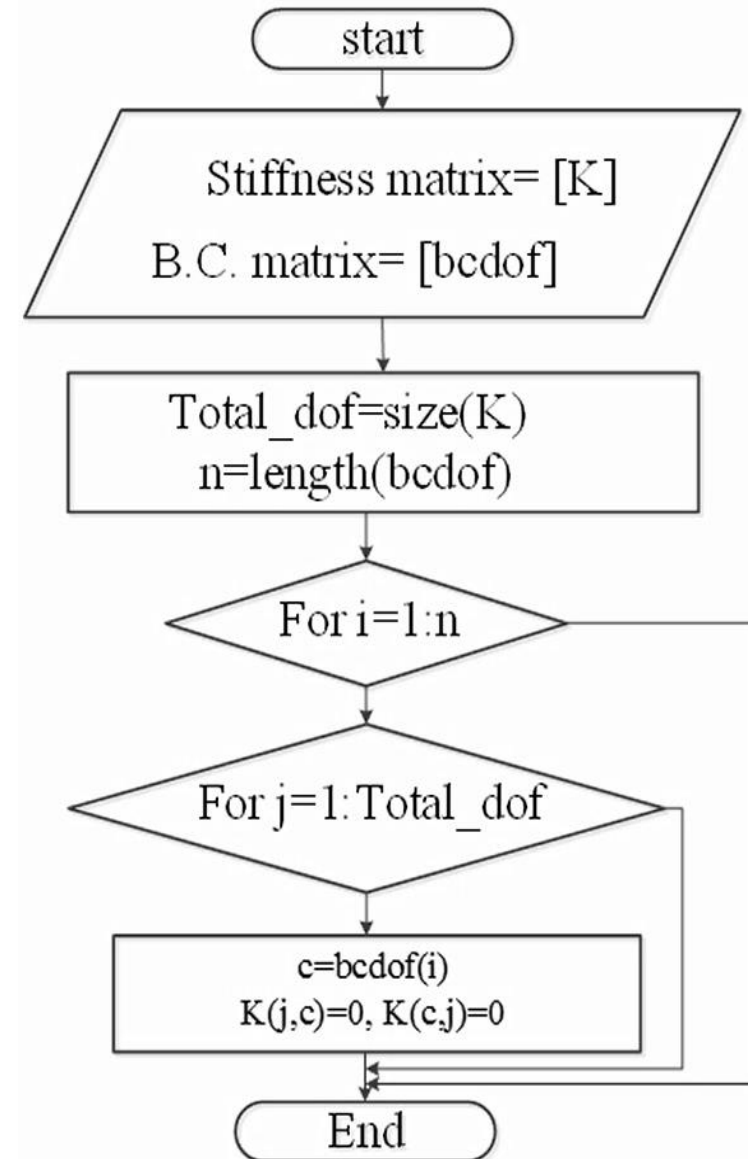
First row and first column of K matrix is zero

Applying boundary conditions with elimination method

Algorithm:

$$K = 10^6 \times \begin{bmatrix} 1 & -1 & 0 & 0 \\ -1 & 2 & -1 & 0 \\ 0 & -1 & 3 & -2 \\ 0 & 0 & -2 & 2 \end{bmatrix}$$

$$bcdof = \begin{bmatrix} 1 \\ 2 \end{bmatrix}$$



Applying boundary conditions with elimination method

Example:

Import :

$$K = 10^6 \times \begin{bmatrix} 1 & -1 & 0 & 0 \\ -1 & 2 & -1 & 0 \\ 0 & -1 & 3 & -2 \\ 0 & 0 & -2 & 2 \end{bmatrix}$$
$$bcdof = \begin{bmatrix} 1 \\ 2 \end{bmatrix}$$

Results:

$$K = 10^6 \times \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 3 & -2 \\ 0 & 0 & -2 & 2 \end{bmatrix}$$

Applying boundary conditions with elimination method

Matlab code:

```
function [ K_T ] = eliminationbc( K,bcdof )
Total_dof=size(K,1);
n=length(bcdof);
K_T=K;
for i=1:n
    for j=1:Total_dof
        c=bcdof(i,1);
        K_T(j,c)=0;
        K_T(c,j)=0;
    end
end
end
```

Applying boundary conditions with elimination method

Import :

```
>> K=10^6*[1 -1 0 0;...  
-1 2 -1 0;...  
0 -1 3 -2;...  
0 0 -2 2];  
>> bcdof=[1;2];  
>> K_T=eliminationbc(K,bcdof)
```

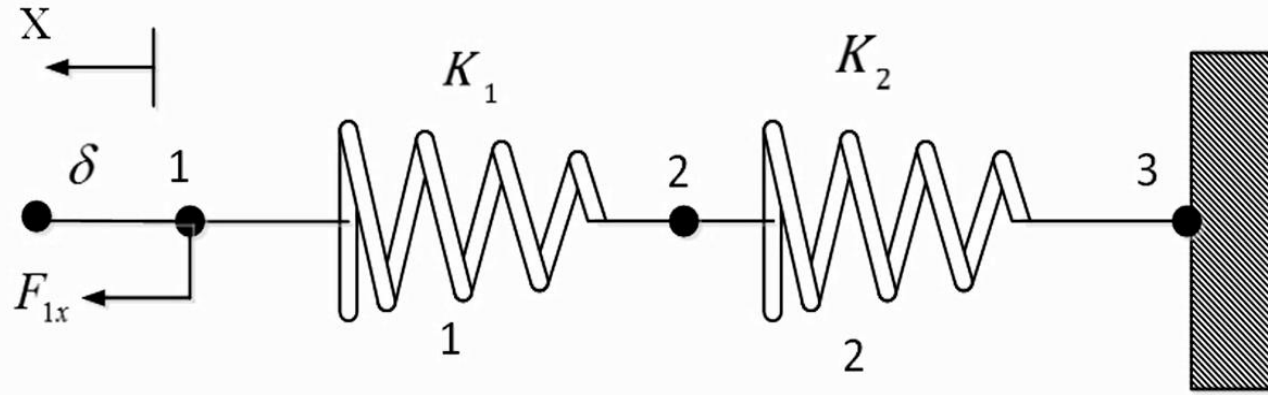
Results:

$K_T =$

0	0	0	0
0	0	0	0
0	0	3000000	-2000000
0	0	-2000000	2000000

Applying boundary conditions with penalty method

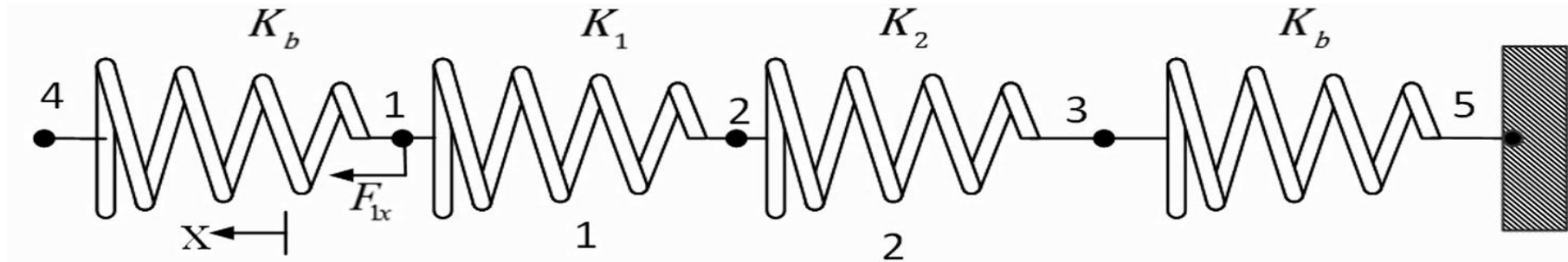
$$K_b = \max([K_{ii}]_G) \times 10^6 \longrightarrow (\text{A big number})$$



$$[K]_G = \begin{bmatrix} K_1 & -K_1 & 0 \\ -K_1 & K_1 + K_2 & -K_2 \\ 0 & -K_2 & K_2 \end{bmatrix}$$

$$\{F\} = \begin{bmatrix} F_1 \\ F_2 \\ F_3 \end{bmatrix}$$

Applying boundary conditions with penalty method



$$[K]_G = \begin{matrix} & \begin{matrix} 4 \\ 1 \\ 2 \\ 3 \\ 5 \end{matrix} \\ \begin{matrix} 4 \\ 1 \\ 2 \\ 3 \\ 5 \end{matrix} & \begin{bmatrix} K_b & -K_b & 0 & 0 & 0 \\ -K_b & K_1 + K_b & -K_1 & 0 & 0 \\ 0 & -K_1 & K_1 + K_2 & -K_2 & 0 \\ 0 & 0 & -K_2 & K_2 + K_b & -K_b \\ 0 & 0 & 0 & -K_b & K_b \end{bmatrix} \end{matrix}$$

$$\{F\} = \begin{matrix} & \begin{matrix} 4 \\ 1 \\ 2 \\ 3 \\ 5 \end{matrix} \\ \begin{matrix} 4 \\ 1 \\ 2 \\ 3 \\ 5 \end{matrix} & \left\{ \begin{array}{c} F_4 \\ F_1 + K_b \delta \\ F_2 \\ (F_3 = 0) + K_b \times 0 \\ F_5 \end{array} \right\} \end{matrix}$$

Applying boundary conditions with penalty method

$$[K]_G = \begin{bmatrix} K_b & -K_b & 0 & 0 & 0 \\ -K_b & K_1 + K_b & -K_1 & 0 & 0 \\ 0 & -K_1 & K_1 + K_2 & -K_2 & 0 \\ 0 & 0 & -K_2 & K_2 + K_b & -K_b \\ 0 & 0 & 0 & -K_b & K_b \end{bmatrix}$$



$$[K]_G = \begin{bmatrix} K_1 + K_b & -K_1 & 0 \\ -K_1 & K_1 + K_2 & -K_2 \\ 0 & -K_2 & K_2 + K_b \end{bmatrix}$$

$$\{F\} = \begin{Bmatrix} F_4 \\ F_1 + K_b \delta \\ F_2 \\ (F_3 = 0) + K_b \times 0 \\ F_5 \end{Bmatrix}$$



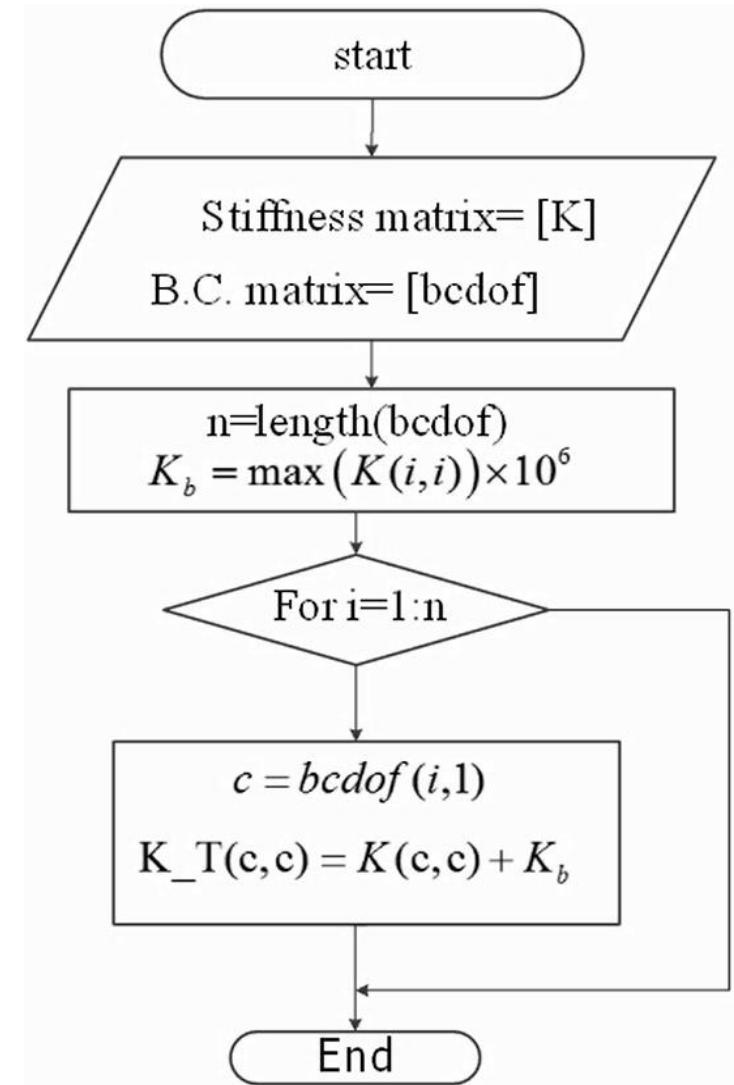
$$\{F\} = \begin{bmatrix} F_{1x} + K_b \delta_1 \\ F_2 \\ 0 \end{bmatrix}$$

Applying boundary conditions with penalty method

Algorithm:

$$K = 10^6 \times \begin{bmatrix} 1 & -1 & 0 & 0 \\ -1 & 2 & -1 & 0 \\ 0 & -1 & 3 & -2 \\ 0 & 0 & -2 & 2 \end{bmatrix}$$

$$bcdof = \begin{bmatrix} 1 \\ 2 \end{bmatrix}$$



Applying boundary conditions with penalty method method

Matlab code:

```
function [ K_t, F_t ] = penaltybc( K, bcdof, F, delta)
n=length(bcdof);
KB=max(diag(K))*1e6;
K_t=K;
F_t=F;
for i=1:n
    c=bcdof(i,1);
    D=delta(i,1);
    K_t(c,c)=K(c,c)+KB;
    F_t(c,1)=F(c,1)+KB*D;
end
end
```

Applying boundary conditions with penalty method

Import :

```
>> K=10^6*[1 -1 0 0;...  
-1 2 -1 0;...  
0 -1 3 -2;...  
0 0 -2 2];  
>> bcdof=[1;2];  
>> delta=[0;0];  
>> F=[0;0;100;100];  
>> [K_t,F_t]=penaltybc(K,bcdof,F,delta)
```

Results:

K_t =

1.0e+12 *

3.0000	-0.0000	0	0
-0.0000	3.0000	-0.0000	0
0	-0.0000	0.0000	-0.0000
0	0	-0.0000	0.0000

F_t =

0
0
100
100

Applying boundary conditions with lagrange multiplier method

$$[C]\{D\} - \{Q\} = \{0\}$$

$$\pi = U - W \quad \Rightarrow \quad \tilde{\pi} = \pi + \lambda^T ([c]\{d\} - \{Q\})$$

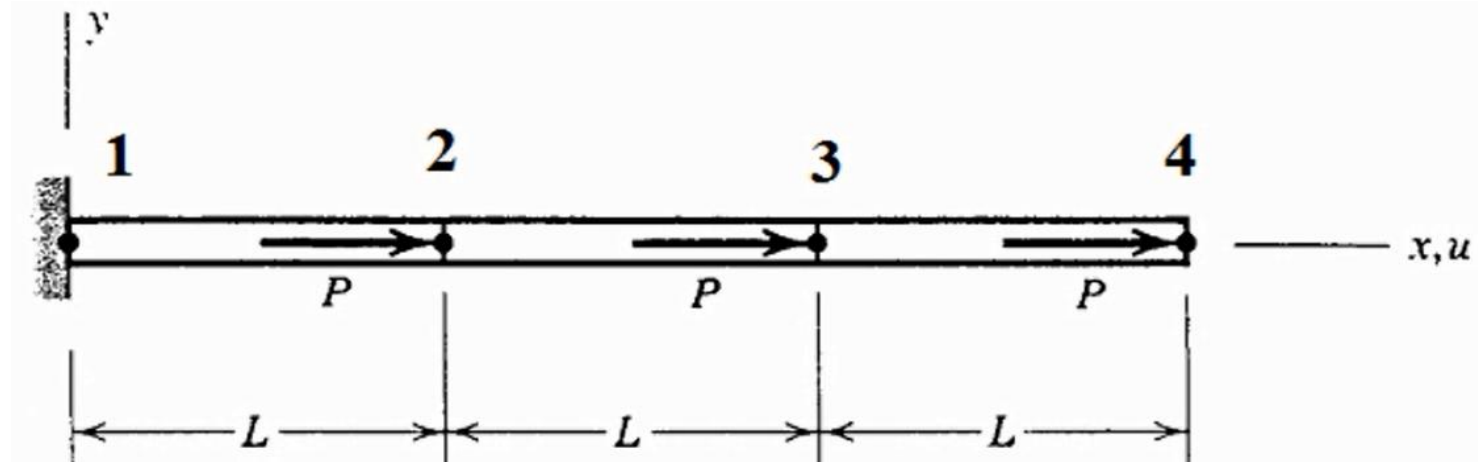
$$\lambda^T = (\lambda_1, \lambda_2, \dots, \lambda_n)$$

$$\frac{\partial \tilde{\pi}}{\partial \{d\}} = 0$$

$$\frac{\partial \tilde{\pi}}{\partial \{\lambda\}} = 0$$

$$\begin{bmatrix} [K] & [c]^T \\ [c] & [0] \end{bmatrix} \begin{Bmatrix} d \\ \lambda \end{Bmatrix} = \begin{Bmatrix} \{F\} \\ \{Q\} \end{Bmatrix}$$

Applying boundary conditions with lagrange multiplier method



$$\begin{array}{l} u_1 = 0 \\ u_3 = u_4 \end{array} \quad \Rightarrow \quad \begin{array}{l} u_1 = 0 \\ u_3 - u_4 = 0 \end{array}$$

$$\begin{array}{c} \boxed{\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & -1 \end{bmatrix}}_{2 \times 4} \begin{Bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \end{Bmatrix}_{4 \times 1} = \begin{Bmatrix} 0 \\ 0 \end{Bmatrix}_{2 \times 1}$$

$[c]$ \swarrow

Applying boundary conditions with lagrange multiplier method

Example:

$$K = \begin{matrix} & \begin{matrix} 1 & 2 & 3 & 4 \end{matrix} \\ \begin{matrix} 1 \\ 2 \\ 3 \\ 4 \end{matrix} & \begin{bmatrix} k_{11} & k_{12} & k_{13} & k_{14} \\ & k_{22} & k_{23} & k_{24} \\ & & k_{33} & k_{34} \\ Sym. & & & k_{44} \end{bmatrix} \end{matrix} \rightarrow \begin{matrix} \begin{bmatrix} k_{11} & k_{12} & k_{13} & k_{14} & 1 & 0 \\ & k_{22} & k_{23} & k_{24} & 0 & 0 \\ & & k_{33} & k_{34} & 0 & 1 \\ Sym & & & k_{44} & 0 & -1 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & -1 & 0 & 0 \end{bmatrix} \end{matrix} \begin{Bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \\ \lambda_1 \\ \lambda_2 \end{Bmatrix} = \begin{Bmatrix} F_1 \\ F_2 \\ F_3 \\ F_4 \\ 0 \\ 0 \end{Bmatrix}$$

Assembly of stiffness matrix

Diagram of a two-spring system. Node 1 is fixed to a wall. Node 2 is free. Node 3 is the junction between the two springs. Spring 1 has stiffness k_1 and connects Node 1 and Node 3. Spring 2 has stiffness k_2 and connects Node 3 and Node 2. A force F_{3x} is applied at Node 3, and a force F_{2x} is applied at Node 2. The displacement at Node 3 is $d_{3x}^{(1)}$ and at Node 2 is $d_{2x}^{(2)}$.

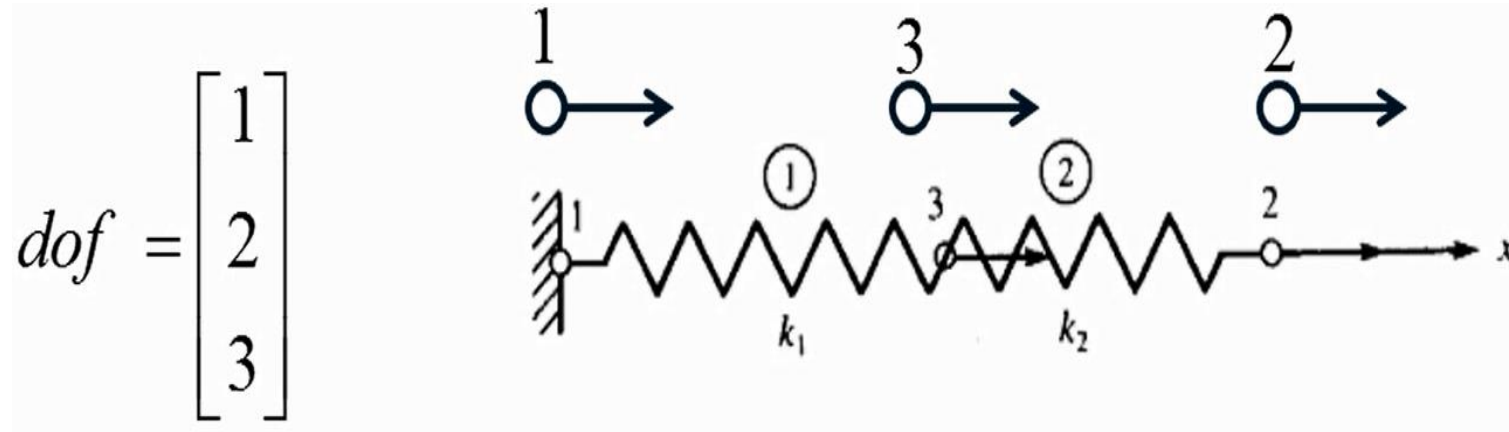
$$\begin{Bmatrix} f_{1x} \\ f_{3x} \end{Bmatrix} = \begin{bmatrix} k_1 & -k_1 \\ -k_1 & k_1 \end{bmatrix} \begin{Bmatrix} d_{1x}^{(1)} \\ d_{3x}^{(1)} \end{Bmatrix}$$
$$\begin{Bmatrix} f_{3x} \\ f_{2x} \end{Bmatrix} = \begin{bmatrix} k_2 & -k_2 \\ -k_2 & k_2 \end{bmatrix} \begin{Bmatrix} d_{3x}^{(2)} \\ d_{2x}^{(2)} \end{Bmatrix}$$

$$d_{3x}^{(1)} = d_{3x}^{(2)} = d_{3x}$$

continuity between nodes

$$\underline{K} = \begin{bmatrix} k_1 & 0 & -k_1 \\ 0 & k_2 & -k_2 \\ -k_1 & -k_2 & k_1 + k_2 \end{bmatrix}$$

Assembly of stiffness matrix



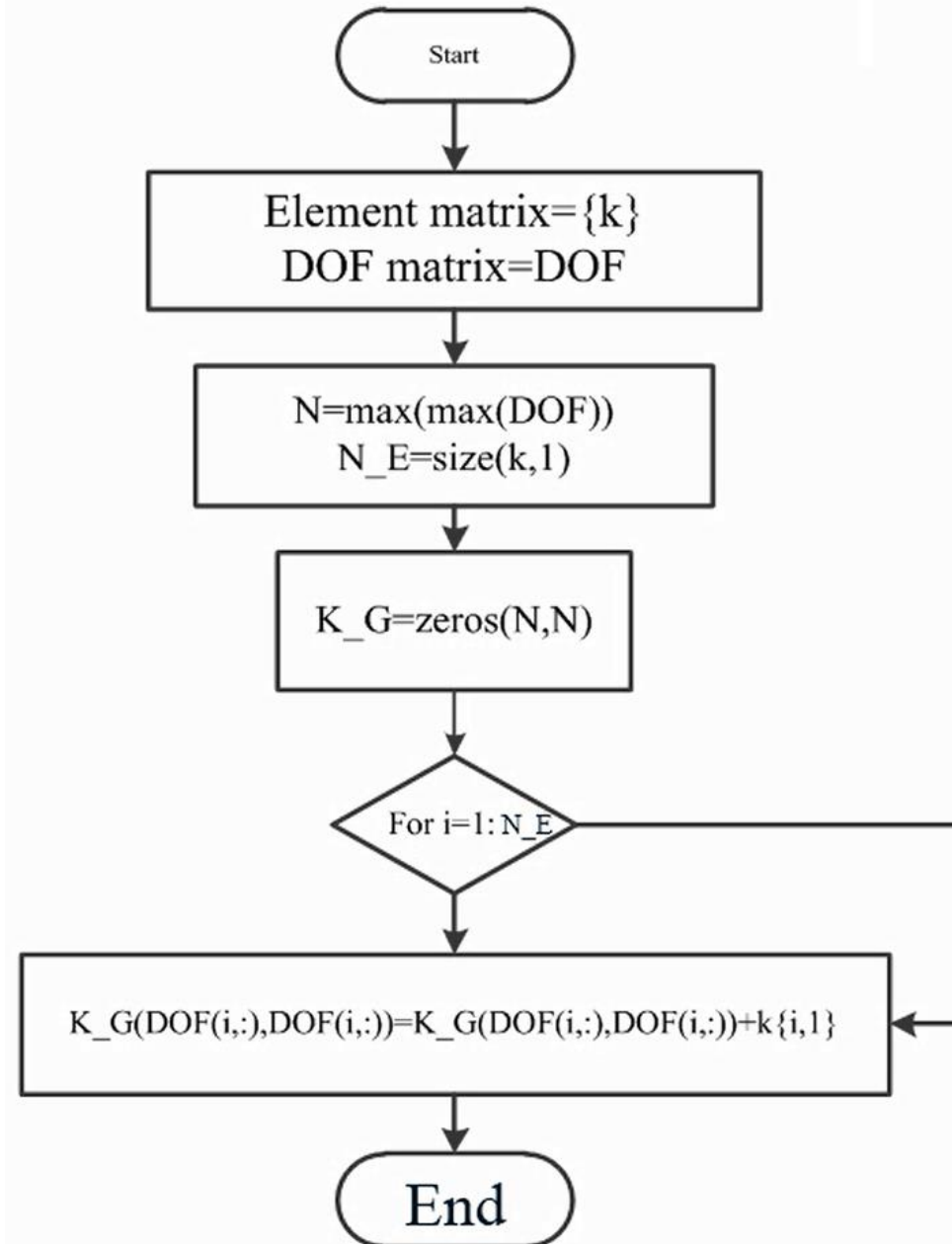
For assembly of stiffness matrix the order of degree of freedom is important!

$$\begin{Bmatrix} F_{3x} \\ F_{2x} \\ F_{1x} \end{Bmatrix} = \begin{matrix} 3 & 2 & 1 \\ 1 & 2 & 3 \end{matrix} \begin{bmatrix} k_1 + k_2 & -k_2 & -k_1 \\ -k_2 & k_2 & 0 \\ -k_1 & 0 & k_1 \end{bmatrix} \begin{Bmatrix} d_{3x} \\ d_{2x} \\ d_{1x} \end{Bmatrix}$$

$$\begin{Bmatrix} F_{1x} \\ F_{2x} \\ F_{3x} \end{Bmatrix} = \begin{matrix} 1 & 2 & 3 \\ 1 & 2 & 3 \end{matrix} \begin{bmatrix} k_1 & 0 & -k_1 \\ 0 & k_2 & -k_2 \\ -k_1 & -k_2 & k_1 + k_2 \end{bmatrix} \begin{Bmatrix} d_{1x} \\ d_{2x} \\ d_{3x} \end{Bmatrix}$$

Assembly of stiffness matrix

Algorithm:



Assembly of stiffness matrix

Matlab code:

```
function [K_G] = assemble(k,DOF)
N=max(max(DOF));
N_E=size(k,1);
K_G=zeros(N,N);
for i=1:N_E

K_G(DOF(i,:),DOF(i,:))=K_G(DOF(i,:),DOF(i,:
))+k{i,1};
end

end
```

Assembly of stiffness matrix

Import :

```
>> k{1,1}=1000*[1 -1;-1 1];  
>> k{2,1}=2000*[1 -1;-1 1];  
>> DOF=[1 2;2 3];  
>> K_G=assembl(k,DOF);
```

Results:

	1	2	3
1	1000	-1000	0
2	-1000	3000	-2000
3	0	-2000	2000

Differential equation for heat transfer

Helmholtz equation:

$$-\left(\frac{\partial q_x}{\partial x} + \frac{\partial q_y}{\partial y} + \frac{\partial q_z}{\partial z}\right) + Q = \rho c \frac{\partial T}{\partial t}$$

The local heat flux:

$$\left\{ \begin{array}{c} q_x \\ q_y \\ q_z \end{array} \right.$$

The heat generated per unit time per unit volume: $Q = Q(x, y, z, t)$

c : is the specific heat

T is the temperature

ρ is the density

Fourier's law of heat transfer:

$$q_x = -k \frac{\partial T}{\partial x}$$

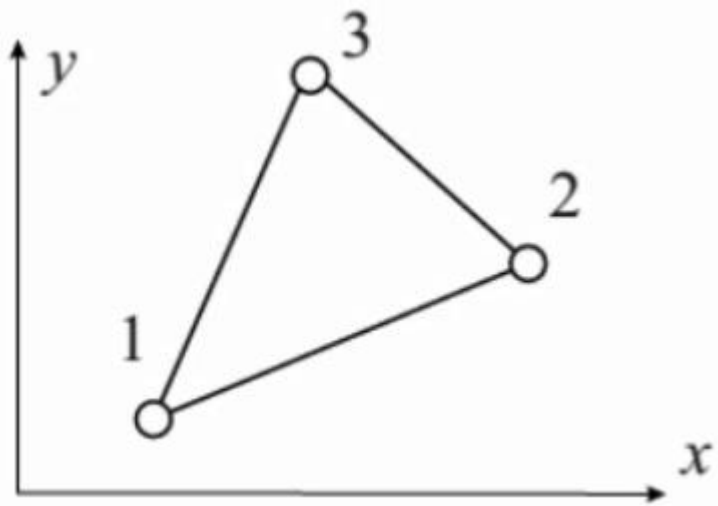
$$q_y = -k \frac{\partial T}{\partial y}$$

$$q_z = -k \frac{\partial T}{\partial z}$$

k: thermal conductivity

$$\frac{\partial}{\partial x} \left(k \frac{\partial T}{\partial x} \right) + \frac{\partial}{\partial y} \left(k \frac{\partial T}{\partial y} \right) + \frac{\partial}{\partial z} \left(k \frac{\partial T}{\partial z} \right) + Q = \rho c \frac{\partial T}{\partial t}$$

Let's say we have triangular elements:



Temperature:

$$T(x,y) = N_1(x,y)T_1 + N_2(x,y)T_2 + N_3(x,y)T_3$$

$$N_i = \frac{1}{2\Delta}(a_i + b_i x + c_i y),$$

$$a_i = x_{i+1}y_{i+2} - x_{i+2}y_{i+1},$$

$$b_i = y_{i+1} - y_{i+2},$$


$$c_i = x_{i+2} - x_{i+1},$$

$$\Delta = \frac{1}{2}(x_2y_3 + x_3y_1 + x_1y_2 - x_2y_1 - x_3y_2 - x_1y_3)$$

$$N_i(x,y) = \alpha_i + \beta_i x + \gamma_i y$$

Stiffness matrix for each element

Galerkin method in FEM:

$$\int N_i \boxed{R_\Omega} d\Omega = 0$$


$$R_\Omega = \left(\frac{\partial q_x}{\partial x} + \frac{\partial q_y}{\partial y} + \frac{\partial q_z}{\partial z} - Q + \rho c \frac{\partial T}{\partial t} \right)$$

$$\int_V \left(\frac{\partial q_x}{\partial x} + \frac{\partial q_y}{\partial y} + \frac{\partial q_z}{\partial z} - Q + \rho c \frac{\partial T}{\partial t} \right) N_i dV = 0$$

$$\{q\} = -k[B]\{T\}$$

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

$$[N] = [N_1 \ N_2 \ \dots],$$

$$\{T\} = \{T_1 \ T_2 \ \dots\}.$$

$$\left\{ \begin{array}{c} \frac{\partial T}{\partial x} \\ \frac{\partial T}{\partial y} \\ \frac{\partial T}{\partial z} \end{array} \right\} = \left[\begin{array}{ccc} \frac{\partial N_1}{\partial x} & \frac{\partial N_2}{\partial x} & \dots \\ \frac{\partial N_1}{\partial y} & \frac{\partial N_2}{\partial y} & \dots \\ \frac{\partial N_1}{\partial z} & \frac{\partial N_2}{\partial z} & \dots \end{array} \right] \{T\} = [B]\{T\}$$

Weak-Form by Fractional Calculus:

$$\int_V \rho c \frac{\partial T}{\partial t} N_i dV - \int_V \left[\frac{\partial N_i}{\partial x} \frac{\partial N_i}{\partial y} \frac{\partial N_i}{\partial z} \right] \{q\} dV = \int_V Q N_i dV - \int_{S_1} \{q\}^T \{n\} N_i dS$$

$$+ \int_{S_2} q_s N_i dS - \int_{S_3} h(T - T_e) N_i dS - \int_{S_4} (\sigma \epsilon T^4 - \alpha q_r) N_i dS.$$



$$[C]\{\dot{T}\} + ([K_c] + [K_h] + [K_r])\{T\} = \{R_T\} + \{R_Q\} + \{R_q\} + \{R_h\} + \{R_r\}$$

$$[C] = \int_V \rho c [N]^T [N] dV$$

$$[K_c] = \int_V k [B]^T [B] dV$$

$$[K_h] = \int_{S_3} h [N]^T [N] dS.$$

$$[K_r]\{T\} = \int_{S_4} \sigma \epsilon T^4 [N]^T dS.$$

Stiffness matrix for each element

$$[C] = \int_V \rho c [N]^T [N] dV$$

t: thickness of plate

$$[C] = \int_A \rho c \begin{bmatrix} N_1 N_1 & N_1 N_2 & N_1 N_3 \\ N_2 N_1 & N_2 N_2 & N_2 N_3 \\ N_3 N_1 & N_3 N_2 & N_3 N_3 \end{bmatrix} t dA$$

$$[C] = \rho c t \int \begin{bmatrix} N_1 N_1 & N_1 N_2 & N_1 N_3 \\ N_2 N_1 & N_2 N_2 & N_2 N_3 \\ N_3 N_1 & N_3 N_2 & N_3 N_3 \end{bmatrix} dA$$

$$N_i N_j = \frac{1}{4\Delta^2} \left[a_i a_j + (a_i b_j + a_j b_i) x + (a_i c_j + a_j c_i) y + (b_i c_j + b_j c_i) xy + b_i b_j x^2 + c_i c_j y^2 \right]$$

positions of nodes in coordinate system: (x_3, y_3) (x_2, y_2) (x_1, y_1)

Stiffness matrix for each element

[C]

m	n	$I = \int_A x^m y^n dx dy$
0	0	$\int dA = A = [x_1(y_2 - y_3) + x_2(y_3 - y_1) + x_3(y_1 - y_2)]/2$
0	1	$\int y dA = A\bar{y} = A(y_1 + y_2 + y_3)/3$
1	0	$\int x dA = A\bar{x} = A(x_1 + x_2 + x_3)/3$
0	2	$\int y^2 dA = A(y_1^2 + y_2^2 + y_3^2 + 9\bar{y}^2)/12$
1	1	$\int xy dA = A(x_1 y_1 + x_2 y_2 + x_3 y_3 + 9\bar{x}\bar{y})/12$
2	0	$\int x^2 dA = A(x_1^2 + x_2^2 + x_3^2 + 9\bar{x}^2)/12$

Stiffness matrix for each element

$$[K_c] = \int_V k[B]^T [B] dV$$
$$[B] = \begin{bmatrix} \frac{\partial N_1}{\partial x} & \frac{\partial N_2}{\partial x} & \dots \\ \frac{\partial N_1}{\partial y} & \frac{\partial N_2}{\partial y} & \dots \\ \frac{\partial N_1}{\partial z} & \frac{\partial N_2}{\partial z} & \dots \end{bmatrix}$$
$$[B] = \begin{bmatrix} \frac{\partial N_1}{\partial x} & \frac{\partial N_2}{\partial x} & \frac{\partial N_3}{\partial x} \\ \frac{\partial N_1}{\partial y} & \frac{\partial N_2}{\partial y} & \frac{\partial N_3}{\partial y} \end{bmatrix} = \frac{1}{2\Delta} \begin{bmatrix} b_1 & b_2 & b_3 \\ c_1 & c_2 & c_3 \end{bmatrix}$$

$$[k_c] = \frac{k}{4\Delta} \begin{bmatrix} b_1^2 + c_1^2 & b_1 b_2 + c_1 c_2 & b_1 b_3 + c_1 c_3 \\ b_1 b_2 + c_1 c_2 & b_2^2 + c_2^2 & b_2 b_3 + c_2 c_3 \\ b_1 b_3 + c_1 c_3 & b_2 b_3 + c_2 c_3 & b_3^2 + c_3^2 \end{bmatrix}$$

Stiffness matrix for each element

$$[K_h] = \int_{S_3} h[N]^T [N] dS.$$

$$[K_h] = \frac{htL}{6} \begin{bmatrix} 0 & 0 & 0 \\ 0 & 2 & 1 \\ 0 & 1 & 2 \end{bmatrix}$$

Matlab code:

K_C function:

```
function Kc=kconductivity(kxx,kyy,b,c,A,t)
B=[b(1,1) b(1,2) b(1,3);c(1,1) c(1,2)
c(1,3)]/(2*A);
D=[kxx 0;0 kyy];
Kc=t*A*B'*D*B;
end
```

K_h function:

```
function [Kh,F]=kconvection(h,L,t,edge,Te)
Kh=(h*L*t/6)*[2 1 0;1 2 0;0 0 0];
alpha=h*Te*L*t/2;
switch edge
    case 1
        F=alpha*[1 1 0];
    case 2
        F=alpha*[0 1 1];
    case 3
        F=alpha*[1 0 1];
end
end
```

C function:

```
function
```

```
C=Ctransient(rho,Cheat,Area,a,b,c,x,y,t)
```

```
alpha=(rho*Cheat*t);
```

```
x0y0=Area;
```

```
yB=(y(1,1)+y(1,2)+y(1,3))/3;
```

```
xB=(x(1,1)+x(1,2)+x(1,3))/3;
```

```
y1=Area*yB;
```

```
x1=Area*xB;
```

```
y2=Area*(y(1,1)^2+y(1,2)^2+y(1,3)^2+9*yB  
^2)/12;
```

```
x1y1=Area*(x(1,1)*y(1,1)+x(1,2)*y(1,2)+x  
(1,3)*y(1,3)+9*xB*yB)/12;
```

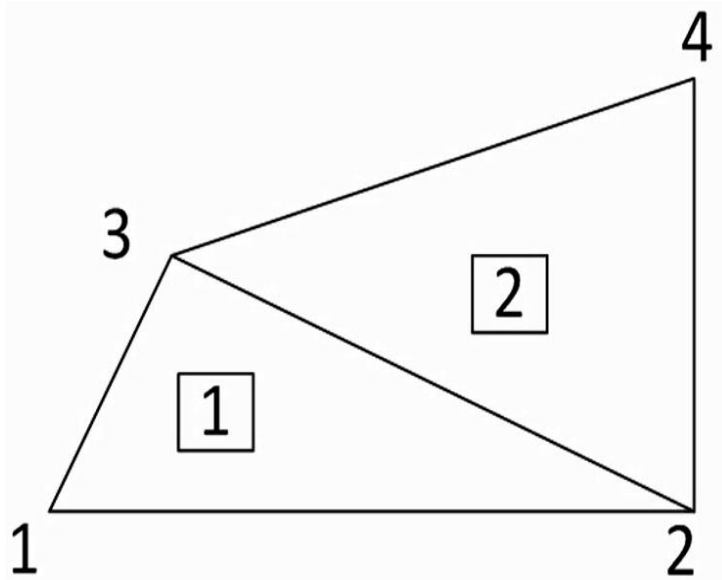
```
x2=Area*(x(1,1)^2+x(1,2)^2+x(1,3)^2+9*xB  
^2)/12;
```

```
for i=1:3
    for j=1:3

N(i,j)=(a(1,i)*a(1,j)*x0y0+(a(1,i)*b(1,j)+(a(1,j)*b(1,i))*
x1+(a(1,i)*c(1,j)+a(1,j)*c(1,i))*y1+...

(b(1,i)*c(1,j)+b(1,j)*c(1,i))*x1y1+b(1,i)*b(1,j)*x2+c(1,i)
*c(1,j)*y2))/(4*Area^2);
    end
end
C=alpha*N;
end
```

Assembly of stiffness matrix



1: 1-2-3

2: 2-4-3

$$K_e^{(1)} = \frac{1}{2} \begin{bmatrix} k_{11}^{(1)} & k_{12}^{(1)} & k_{13}^{(1)} \\ & k_{22}^{(1)} & k_{23}^{(1)} \\ 3 & \text{sym.} & k_{33}^{(1)} \end{bmatrix}$$

$$K_e^{(2)} = \frac{1}{4} \begin{bmatrix} k_{22}^{(2)} & k_{23}^{(2)} & k_{24}^{(2)} \\ & k_{33}^{(2)} & k_{34}^{(2)} \\ 4 & \text{sym.} & k_{44}^{(2)} \end{bmatrix}$$

$$[K] = \begin{bmatrix} 1 & k_{11}^{(1)} & k_{12}^{(1)} & k_{13}^{(1)} & 0 \\ 2 & & k_{22}^{(1)} + k_{22}^{(2)} & k_{23}^{(1)} + k_{23}^{(2)} & k_{24}^{(2)} \\ 3 & & & k_{33}^{(1)} + k_{33}^{(2)} & k_{34}^{(2)} \\ 4 & & & & k_{44}^{(2)} \end{bmatrix} \text{Sym.}$$

Boundary conditions for heat transfer

1. For constant temperature:

$$T_s = T_1(x, y, z, t) \text{ on } S_1$$

T_s : temperature on the surface

2. For constant heat flux

$$q_x n_x + q_y n_y + q_z n_z = -q_s \text{ on } S_2$$

q_s : heat flux on the surface

3. Convection boundary condition

$$q_x n_x + q_y n_y + q_z n_z = h(T_s - T_e) \text{ on } S_3$$

h : coefficient of convective heat transfer

4. Radiation

$$q_x n_x + q_y n_y + q_z n_z = \sigma \varepsilon T_s^4 - \alpha q_r \text{ on } S_4$$

α : the surface absorption coefficient

ε : the surface emission coefficient

σ : The Stefan–Boltzmann constant

q_r : the incident radiant heat flow per unit surface area

Boundary conditions for heat transfer

Boundary conditions:

$$\{R_T\} = - \int_{S_1} \{q\}^T \{n\} [N]^T dS$$

$$\{R_Q\} = \int_V Q [N]^T dV$$

$$\{R_q\} = \int_{S_2} q_s [N]^T dS$$

$$\{R_r\} = \int_{S_4} \alpha q_r [N]^T dS$$

$$\{R_h\} = \int_{S_3} h T_e [N]^T dS$$

Boundary conditions for heat transfer

$$\{R_h\} = \int_L hT_e [N]^T dL = \int_0^1 hT_e [N_1 \ N_2]^T L d\xi = hT_e L \int_0^1 [N_1 \ N_2]^T d\xi$$

$$N_1 = 1 - \xi, \quad N_2 = \xi$$

$$\{R_h\} = \frac{hT_e L}{2} \begin{Bmatrix} 1 \\ 1 \end{Bmatrix}$$

Matlab code:

K_h function:

```
function [Kh,F]=kconvection(h,L,t,edge,Te)
Kh=(h*L*t/6)*[2 1 0;1 2 0;0 0 0];
alpha=h*Te*L*t/2;
switch edge
    case 1
        F=alpha*[1 1 0];
    case 2
        F=alpha*[0 1 1];
    case 3
        F=alpha*[1 0 1];
end
end
```

Matlab code:

Main body:

```
clc
clear
tic
format long
%% Input
input=xlsread('input.xlsx','SHEET1');
mesh=xlsread('mesh1.xlsx','SHEET1');

n_node=input(1,1);
n_element=input(1,2);
n_bc_conv=input(1,3);
n_bc_temp=input(1,4);
BC_conv=input(1:n_bc_conv,5:8);
BC_temp=input(1:n_bc_temp,9:10);
thickness=input(1,12);
kxx=input(1,14);
kyy=input(1,15);
rho=input(1,16);
Cheat=input(1,17);
```

```

x(:,1)=mesh(1:n_node,2);
y(:,1)=mesh(1:n_node,3);
element_info=mesh(1:n_element,7:10);
meshplot(x,y,element_info,n_element)

%% Shape function
K_global=zeros(n_node);
C_global=zeros(n_node);
for i=1:n_element
    jj=element_info(i,2);
    kk=element_info(i,3);
    ll=element_info(i,4);
    x_element(i,1:3)=[x(jj,1) x(kk,1) x(ll,1)];
    y_element(i,1:3)=[y(jj,1) y(kk,1) y(ll,1)];

    [a(i,:),b(i,:),c(i,:),Area(i,:),L(i,:)]=shapefunction(x_element(i,1:3),
y_element(i,1:3));
    Kc{i,1}=kconductivity(kxx,kyy,b(i,:),c(i,:),Area(i,:),thickness);

    K_global(element_info(i,2:4),element_info(i,2:4))=K_global(element_info
(i,2:4),element_info(i,2:4))+Kc{i,1};

    C{i,1}=Ctransient(rho,Cheat,Area(i,:),a(i,:),b(i,:),c(i,:),x_element(i,
1:3),y_element(i,1:3),thickness);

    C_global(element_info(i,2:4),element_info(i,2:4))=C_global(element_info
(i,2:4),element_info(i,2:4))+C{i,1};
end

```

```

%% Fh Kh
F_global=zeros(n_node,1);
for i=1:n_bc_conv
    element_conv=BC_conv(i,1);
    edge=BC_conv(i,2);
    LL=L(element_conv,edge);
    h=BC_conv(i,3);
    Te=BC_conv(i,4);
    [Kh{i,1},F{i,1}]=kconvection(h,LL,thickness,edge,Te);

K_global(element_info(element_conv,2:4),element_info(element_conv,2:4))=K_global(element_info(element_conv,2:4),element_info(element_conv,2:4))+Kh{i,1};

F_global(element_info(element_conv,2:4),1)=F_global(element_info(element_conv,2:4),1)+F{i,1}';
end

```

```

%% Constant Temperature
[KK_global,FF_global]=boundary(K_global,F_global,BC_temp);
%% Steady State Solution
disp('*****')
;
disp('* Steady State Nodal Temperatures are ... *');
disp('*****')
;
T=(KK_global)\FF_global;
for i=1:n_node
    dd=[num2str(i),'= ',num2str(T(i,1))];
    disp(dd)
end
%% Transient Solution
ti=input(6,1);
tf=input(6,2);
dt=input(6,3);
beta=input(6,4);
Tinitial(1:n_node,1)=input(8,2);

```

```

disp('*****');
disp('* Transient solution is doing ... *');
disp('*****');
Tt=transientbeta(KK_global,C_global,FF_global,Tinitial,beta,ti,tf,dt);
AAA=[' The variable "Tt" has saved the transient responses from ',
num2str(ti),'to', num2str(tf),'sec'];
disp(AAA);
time=[ti:dt:tf]';
n_X=input(10,2);
for jjj=1:n_X
    X=input(10+jjj,2);
    for iii=1:(tf-ti)/dt+1
        temp(iii,1)=Tt{iii,2}(X,1);
    end
    figure
    plot(time,temp)
    legend(num2str(X))
end
toc

```

Matlab code:

Mesh plot:

```
function
meshplot(x,y,element_info,n_element)
NODETINFO=['Number of nodes are =
',num2str(size(x,1))];
disp(NODETINFO);
ELEMENTINFO=['Number of elements are =
',num2str(n_element)];
disp(ELEMENTINFO);
close all
step=0.0005;
% Create figure
figure1=figure('Color',[1 1 1]);
% Creat axis
axes1=axes('Parent',figure1);
box(axes1,'on');
hold(axes1,'all');
```

```

for i=1:n_element
    XA=x(element_info(i,2),1);
    XB=x(element_info(i,3),1);
    XC=x(element_info(i,4),1);
    YA=y(element_info(i,2),1);
    YB=y(element_info(i,3),1);
    YC=y(element_info(i,4),1);
    E=XB-XA;
    if E>0
        X1=XA:step:XB;
        M=(YA-YB)/(XA-XB);
        Y1=M*(X1-XA)+YA;
    else E<0
        X1=XB:step:XA;
        M=(YA-YB)/(XA-XB);
        Y1=M*(X1-XA)+YA;
    end
    if XB<XC
        X2=XB:step:XC;
        M=(YB-YC)/(XB-XC);
        Y2=M*(X2-XB)+YB;

```

```

    else
        X2=XC:step:XB;
        M=(YB-YC)/(XB-XC);
        Y2=M*(X2-XB)+YB;
    end
    if XA<XC
        X3=XA:step:XC;
        M=(YA-YC)/(XA-XC);
        Y3=M*(X3-XA)+YA;
    else
        X3=XC:step:XA;
        M=(YA-YC)/(XA-XC);
        Y3=M*(X3-XA)+YA;
    end
    plot(X1,Y1,X2,Y2,X3,Y3);
end

end

```


Matlab code:

Shape function:

```
function [a,b,c,Area,L]=shapefunction(x,y)
a=zeros(1,3);
b=zeros(1,3);
c=zeros(1,3);
a(1,1)=x(1,2)*y(1,3)-x(1,3)*y(1,2);
a(1,2)=x(1,3)*y(1,1)-x(1,1)*y(1,3);
a(1,3)=x(1,1)*y(1,2)-x(1,2)*y(1,1);
b(1,1)=y(1,2)-y(1,3);
b(1,2)=y(1,3)-y(1,1);
b(1,3)=y(1,1)-y(1,2);
c(1,1)=x(1,3)-x(1,2);
c(1,2)=x(1,1)-x(1,3);
c(1,3)=x(1,2)-x(1,1);
A=[1 x(1,1) y(1,1);...
    1 x(1,2) y(1,2);...
    1 x(1,3) y(1,3)];
```

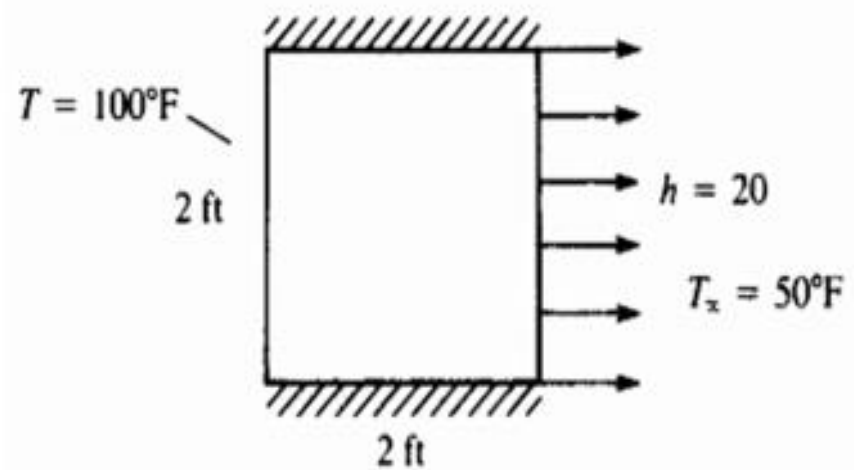
```
Area=det(A)/2;
L(1,1)=sqrt((x(1,2)-x(1,1))^2+(y(1,2)-y(1,1))^2);
L(1,2)=sqrt((x(1,3)-x(1,2))^2+(y(1,3)-y(1,3))^2);
L(1,3)=sqrt((x(1,3)-x(1,1))^2+(y(1,3)-y(1,1))^2);
end
```

Matlab code:

Boundary function:

```
function [K_T,F_T]=boundary(K,F,BC)
Kb=max(diag(K))*(1e6);
d=BC(:,2);
F_T=F;
K_T=K;
for i=1:size(BC,1)
    m=BC(i,1);
    K_T(m,m)=K(m,m)+Kb;
    F_T(m,1)=Kb*d(i,1);
end
end
```

Example:



$$K_{xx} = K_{yy} = 25$$

