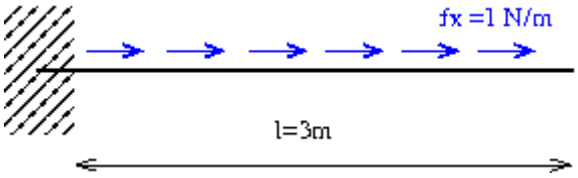
Tutorial No. 4 – Beam in tension/compression



E = A = 1

Analytical solution

Differential equation:

$$EA\frac{d^2u}{dx^2} + f_x(x) = 0$$

Integration:

$$EA\frac{du}{dx} + C_1 + 1x = 0$$

$$EAu + C_1 x + C_2 + x^2/2 = 0$$

From boundary conditions ($u(0) = 0, N(l) = EA\frac{du}{dx} = 0$):

$$EAu(0) + C_1 0 + C_2 = 0 \rightarrow C_2 = 0$$

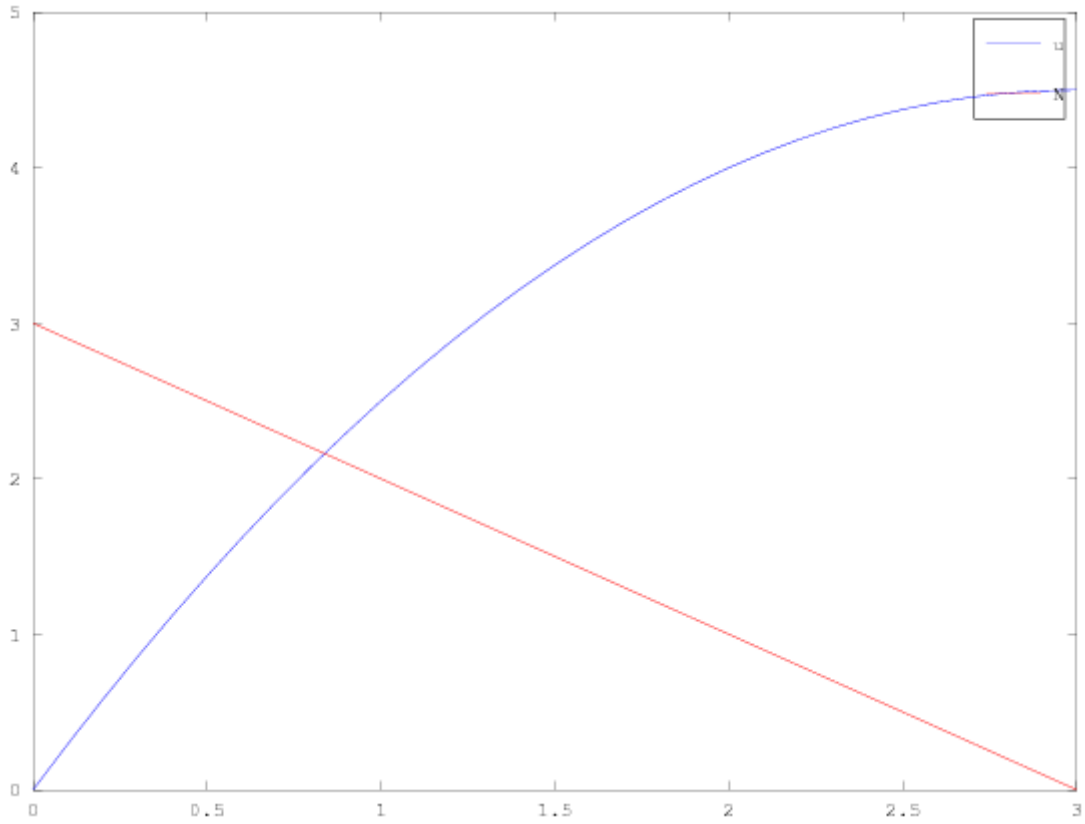
$$EA\frac{du}{dx}(l) + C_1 + l = 0 \rightarrow C_1 = -l$$

Solution: $u(x) = \frac{1}{EA}(-x^2/2 + lx)$

$$\varepsilon(x) = \frac{du}{dx} = \frac{1}{EA}(-x + l)$$

$$N(x) = EA\varepsilon(x) = -x + l$$

```
x = 0:0.1:3;
plot (x, -x.^2/2+3*x, "b;u;", x, -x+3, "r;N;")
```



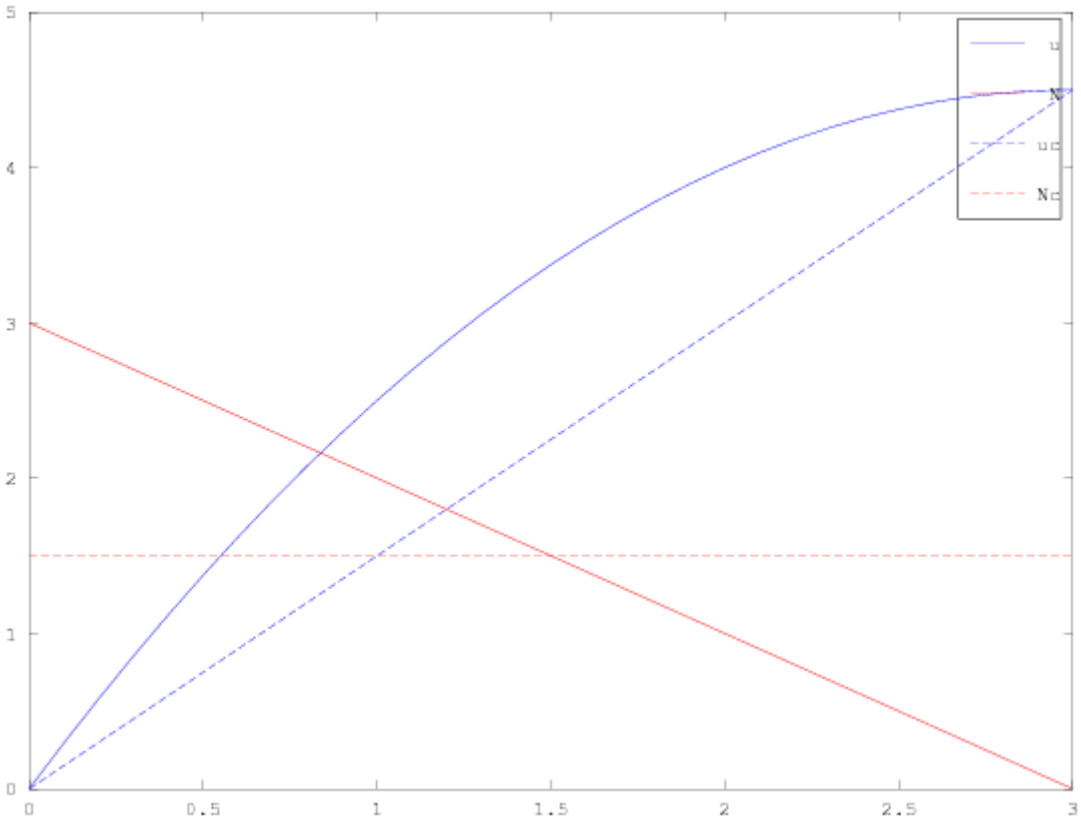
Solution with one element with linear approximation functions

Loading vector:

$$f_e = \int_0^l N^T f_x \, dx = \int_0^l \begin{bmatrix} \frac{l-x}{l} \\ \frac{x}{l} \end{bmatrix} 1 dx = \begin{bmatrix} \frac{l^2}{2l} \\ \frac{l^2}{2l} \end{bmatrix}$$

```
E = 1;
A = 1;
l1 = 3;

k1 = (E*A/l1)*[1 -1; -1 1];
loc1 = [0 1];
K = k1;
f1 = [l1^2/(2*l1) l1^2/(2*l1)];
F = f1

u = F(2)/K(2,2);
U = [0 u]
eps1 = (U(2)-U(1))/l1;
N1 = E*A*eps1;

hold on;
plot (x, -x.^2/2+3*x, "b;u;", x, -x+3, "r;N;")
plot ([0 l1], [U(1) U(2)], "b--;uc;")
plot ([0 l1], [N1 N1], "r--;Nc;")
```

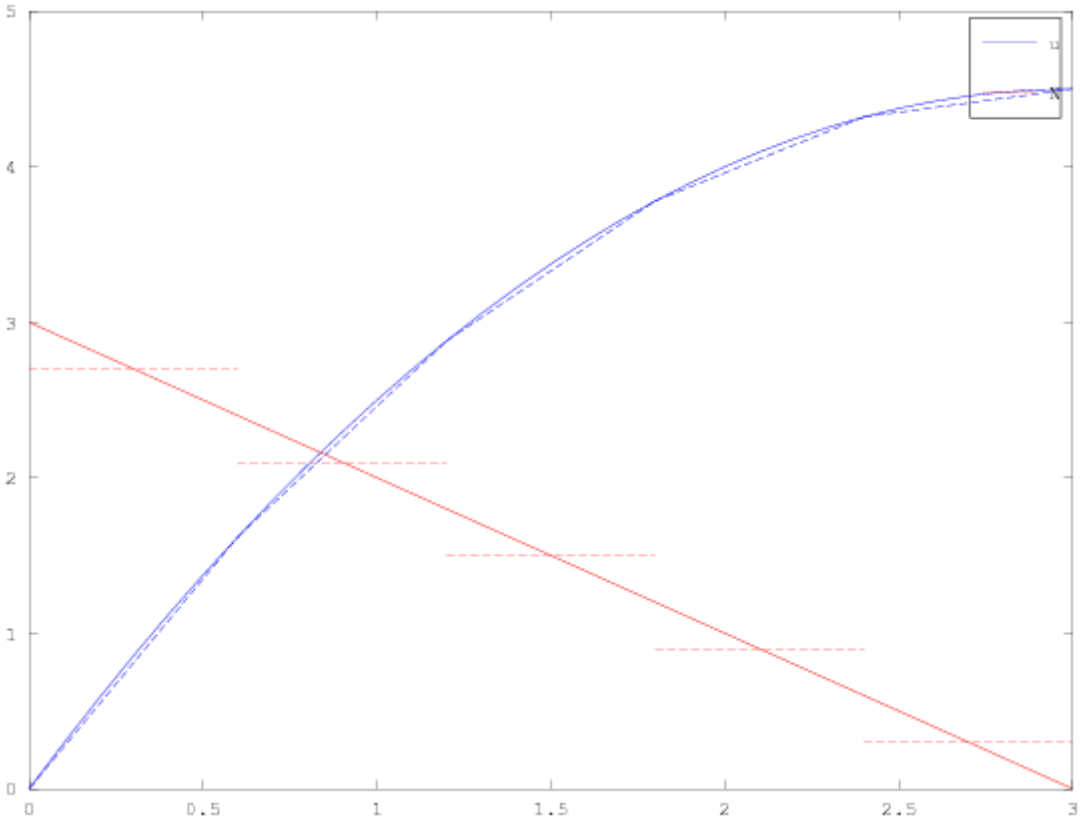
```
F =

   1.5000   1.5000

U =

   0.00000   4.50000
```



Solution with *n* elements:

```
n = 5
l = 3/n;

E = 1;
A = 1;

ki = (E*A/l)*[1 -1; -1 1];
fi = [l^2/(2*l) l^2/(2*l)];
K = zeros (n+1);
F = zeros (n+1, 1);
for i=1:n
    loc = [i i+1];
    K(loc, loc) += ki;
    F(loc)+= fi';
endfor
u = K(2:n+1, 2:n+1)\(F(2:n+1,1));
U = [0 ; u]

#plot analytical solution
hold on;
plot (x, -x.^2/2+3*x, "b;u;", x, -x+3, "r;N;")

#plot obtained solution
for i=1:n
    eps = (U(i+1)-U(i))/l;
    N = E*A*eps;
    plot ([(i-1)*l i*l], [U(i) U(i+1)], "b--")
    plot ([(i-1)*l i*l], [N N], "r--")
endfor
```

```
n =  5
U =

   0.00000
   1.62000
   2.88000
   3.78000
   4.32000
   4.50000
```

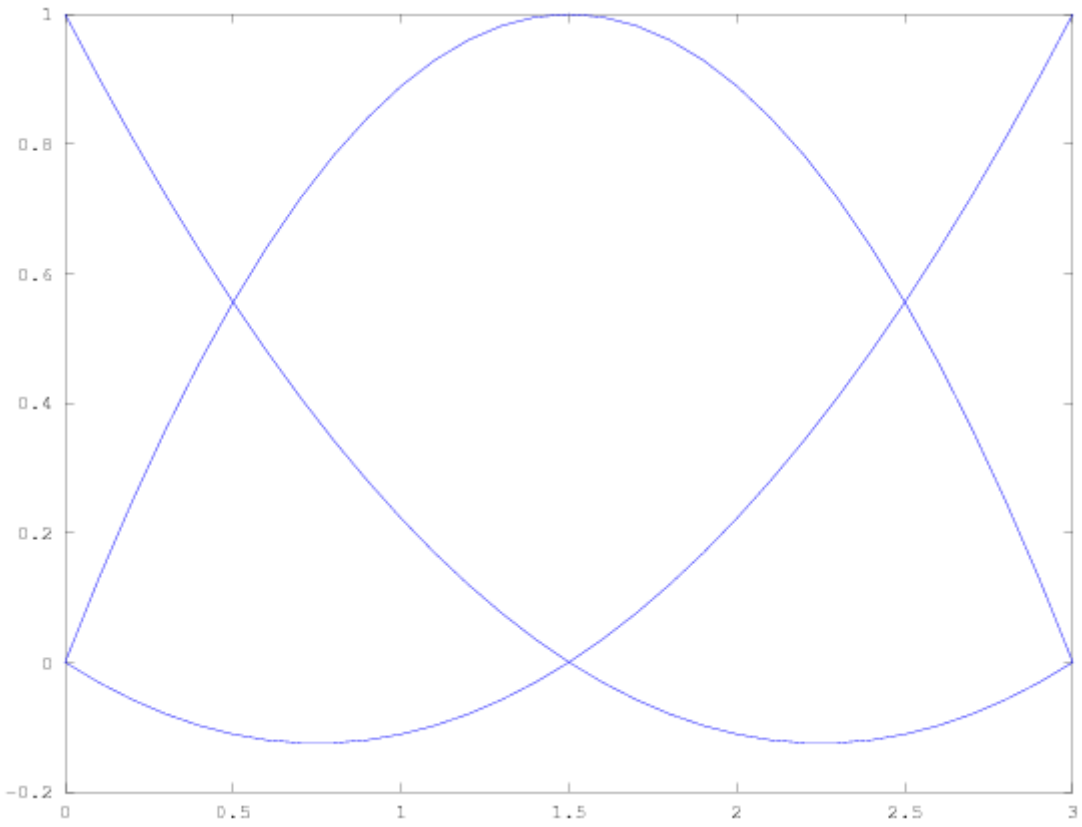


Solution with one element with quadratic approximation functions

Quadratic approximation requires 3 nodes, 2 in end points and one in the center.

- $N_1(x) = \frac{2}{l^2}(x - l/2)(x - l)$
- $N_2(x) = -\frac{4}{l^2}(x)(x - l)$
- $N_3(x) = \frac{2}{l^2}(x)(x - l/2)$

```
l = 3;
x=0:0.1:l;
hold on;

plot (x, (2/l^2).*(x-l/2).*(x-l))
plot (x, -(4/l^2).*x.*(x-l))
plot (x, (2/l^2).*x.*(x-l/2))
```



Stiffness matrix assembly

Derivatives of interpolation functions

- $\frac{dN_1}{dx}(x) = \frac{2}{l^2}((x-l)+(x-l/2))$
- $\frac{dN_2}{dx}(x) = -\frac{4}{l^2}((x-l)+x)$
- $\frac{dN_3}{dx}(x) = \frac{2}{l^2}((x-l/2)+x)$

Derivatives of interpolation functions gives geometric matrix ***B***:

$B = [\frac{dN_1}{dx}(x), \frac{dN_2}{dx}(x), \frac{dN_3}{dx}(x)] = \frac{1}{l^2}[4x-3l, -8x+4l, 4x-l]$

Stiffness matrix: $K = \int_0^l B^T EAB\, dx.$

For individual components:

- $K_{11} = EA\int_0^3 (\frac{dN_1}{dx})^2\, dx = EA\int_0^3 \frac{4}{81}((x-3)+(x-1.5))^2\, dx = EA\frac{4}{81}\int_0^3 (2x-4.5)^2\, dx = EA\frac{4}{81}\int_0^3 (4x^2-18x+4.5^2)\, dx$
  $= EA\frac{4}{81}[\frac{4x^3}{3} - 18\frac{x^2}{2} + 20.25x]_0^3 = EA\frac{7}{9}$
- $K_{12} = K_{21} = EA\int_0^3 (\frac{dN_1}{dx})(\frac{dN_2}{dx})\, dx = -EA\frac{8}{9}$
- $K_{13} = K_{31} = EA\int_0^3 (\frac{dN_1}{dx})(\frac{dN_3}{dx})\, dx = EA\frac{1}{9}$
- ....

$$K = EA\begin{bmatrix} 7/9 & -8/9 & 1/9 \\ -8/9 & 16/9 & -8/9 \\ 1/9 & -8/9 & 7/9 \end{bmatrix}$$

Loading vector:

Loading vector is calculated from the relation:

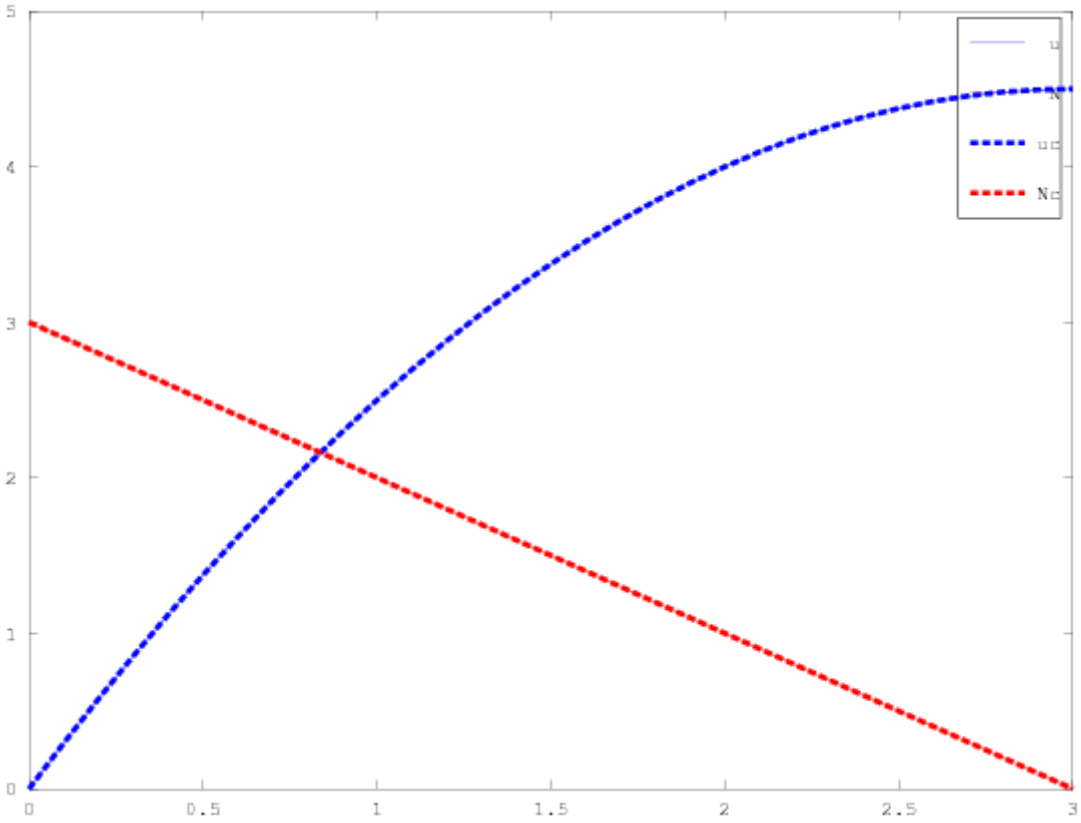
$f = \int_0^l N^T f\, dx$

where matrix ***N*** has interpolation functions.

- $f_1 = \int_0^1 \frac{2}{l^2}(x-l/2)(x-l)\, 1\, dx = \frac{2}{l^2}[x^3/3 - (3/4)lx^2 + xl^2/2]_0^3 = 1/2$
- $f_2 = \int_0^1 -\frac{4}{l^2}(x)(x-l)\, 1\, dx = 2$
- $f_3 = \int_0^1 \frac{2}{l^2}(x)(x-l/2)\, 1\, dx = 1/2$

```
E = 1;
A = 1;

k1 = (E*A)*[7/9, -8/9, 1/9;
            -8/9, 16/9, -8/9;
            1/9, -8/9, 7/9]
loc1 = [0 1 2];
K = k1;
f1 = [1/2; 2; 1/2];
F = f1

u = K(2:3,2:3)\F(2:3);
U = [0;u]
eps1 = @(x) (1/9).*((4.*x-9)*U(1)+(-8.*x+12)*U(2)+(4.*x-3)*U(3));
u1x = @(x) (2/9).*(((x-1.5).*(x-3))*U(1)+(-2.*x.*(x-3))*U(2)+(x.*(x-1.5))*U(3));

hold on;
x = 0:0.1:3;
plot (x, -x.^2/2+3*x, "b;u;", x, -x+3, "r;N;")
plot (x, u1x(x), "b--;uc;", "linewidth",10)
plot (x, eps1(x)*E*A, "r--;Nc;", "linewidth", 10)
```

```
k1 =

   0.77778  -0.88889   0.11111
  -0.88889   1.77778  -0.88889
   0.11111  -0.88889   0.77778

F =

   0.50000
   2.00000
   0.50000

U =

   0.00000
   3.37500
   4.50000
```



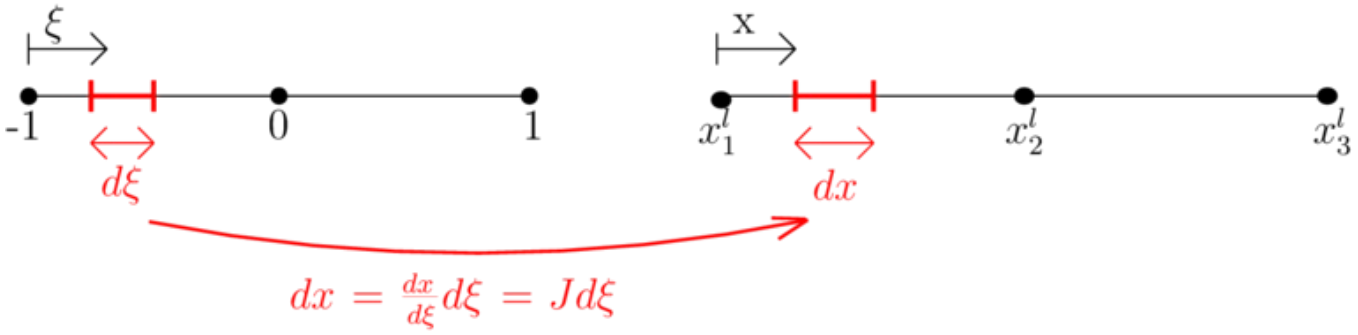
Solution using numerical integration extended in 2D

Transformation of local coordinates

The natural coordinate system is 1D but the truss is 2D structure! The transformation of nodal coordinates must be transformed into the local coordinate system:

$x^{le} = T^{e} x^{ge}$

```
function T = transformation_matrix(x)
    # Transformation matrix for a given vector of global element nodal coordinat
es into local coordinates
    # Vector xeg contains sequentially ordered nodal vectors - xeg = [x1g, y1g,
 x2g, y2g, ..., xng, yng]
    # n = number of element nodes / (order of approximation - 1)
    # Minimum n = 2 (linear approximation)
    # nodes 1 and 2 define the local coordinate system (angle of rotation fi)
    length=sqrt((x(3)-x(1))^2+(x(4)-x(2))^2);
    c = (x(3)-x(1))/length; c2=c^2;
    s = (x(4)-x(2))/length; s2=s^2;
    t1 = [c, s;
          -s,c];
    number_of_nodes = size(x)(2)/2;
    T = zeros(number_of_nodes*2);
    for i = 1:number_of_nodes
        loc = 2*i-1:2*i;
        T(loc,loc) += t1;
    end
end
```



Transformation into natural coordinate system

The axis of the natural coordinate is defined in the beam axis. The initial node (left) has the coordinate -1 and the end node (right) has the coordinate +1.

Isoparametric element

When the element is described in natural coordinate system, the transformation relationship expressing the Kartesian coordinate is needed in natural coordinates.
Isoparametric elements define approximation for nodal values and nodal coordinates:

- $u(\xi) = N(\xi)^T u^{le}$
- $x(\xi) = N(\xi)^T x^{le}$

Jacobian of transformation is calculated as the derivative of absolute (Kartesian) coordinates with respect to natural coordinates.

$$\frac{dx}{d\xi} = \frac{dN}{d\xi} x^{le} = J$$

The derivative of shape functions is obtained using the derivative of compound function:

$$B(\xi) = \frac{dN(\xi(x))}{dx} = \frac{dN(\xi)}{d\xi}\frac{d\xi}{dx} = \frac{dN}{d\xi} J^{-1}$$

Stiffness matrix defined in Kartesian coordinate system

$$K^{le} = EA \int_{x_1^{le}}^{x_n^{le}} B(x)^T B(x) dx$$

Which can be expressed in the natural coordinate system:

$$K^{le} = EA \int_{\xi(x_1^{le})}^{\xi(x_n^{le})} B(\xi(x))^T B(\xi(x)) dx = EA \int_{-1}^{1} B(\xi)^T B(\xi) J d\xi.$$

Loading vector is derived similarly:

$$f^{le} = \int_{x_1^{le}}^{x_n^{le}} N(x)^T f_x(x) dx = \int_{\xi(x_1^{le})}^{\xi(x_n^{le})} N(\xi(x))^T f_x(\xi(x)) dx = \int_{-1}^{1} N(\xi)^T f_x(\xi) J d\xi$$

Importance of natural coordinates

Transformation of the element into natual coordinate system enables the numerical integration in the constant interval $< -1, 1 >$.

```
# Weights for gaussian integration
global w_switch = cell(3,1);
w_switch{1} = [2.];
w_switch{2} = [1., 1.];
w_switch{3} = [0.555556 0.888889 0.555556];
# Locations for gaussian integration
global xi_switch = cell(3,1);
xi_switch{1} = [0.];
xi_switch{2} = [-1./sqrt(3.), 1./sqrt(3.)];
xi_switch{3} = [-0.774597, 0., 0.774597];

function ans = numerical_integration(nip, f)
    # Numerical integration using the Gauss quadrature rule
    # f - function to be integrated, defined in natural coordinates
    # nip - number of integration points
    global w_switch
    global xi_switch
    w = w_switch{nip};
    xi = xi_switch{nip};
    ans = 0.0*f(0.);
    for i=1:nip
        ans = ans + w(i)*f(xi(i));
    end
end
```

Linear element:

Vector of interpolation functions:

$$N(\xi) = \begin{bmatrix} \frac{-(\xi-1)}{2} & 0 & \frac{\xi+1}{2} & 0 \end{bmatrix}$$

and its derivative:

$$\frac{dN(\xi)}{d\xi} = \begin{bmatrix} \frac{-1}{2} & 0 & \frac{1}{2} & 0 \end{bmatrix}$$

```
global N_lin = @(xi) [-(xi-1.)/2., 0., (xi+1.)/2., 0.];

global dN_lin = @(xi) [-1./2., 0., 1./2., 0.];
global J_lin = @(xi, xe) dN_lin(xi)*xe';

function ans = K_truss2d_linear(xeg, EA)
    # Local stiffness matrix of linear truss element
    # in natural coordinates
    global dN_lin;
    global J_lin;
    T = transformation_matrix(xeg);
    xel = T*xeg';
    integrand = @(xi) dN_lin(xi)'*dN_lin(xi)/J_lin(xi,xel');
    ans = EA*numerical_integration(2, integrand);
    ans = T'*ans*T;
end

function ans = f_linear(xeg, f)
    global N_lin;
    global J_lin;
    T = transformation_matrix(xeg);
    xel = T*xeg';
    integrand = @(xi) N_lin(xi)'*f(xi)*J_lin(xi,xel');
    ans = numerical_integration(2, integrand);
    ans = T'*ans;
end
```

Quadratic element:

Vector of interpolation functions:

$$N(\xi) = \begin{bmatrix} \frac{\xi(\xi-1)}{2} & 0 & -(\xi+1)(\xi-1) & 0 & \frac{\xi(\xi+1)}{2} & 0 \end{bmatrix}$$

and its derivative:

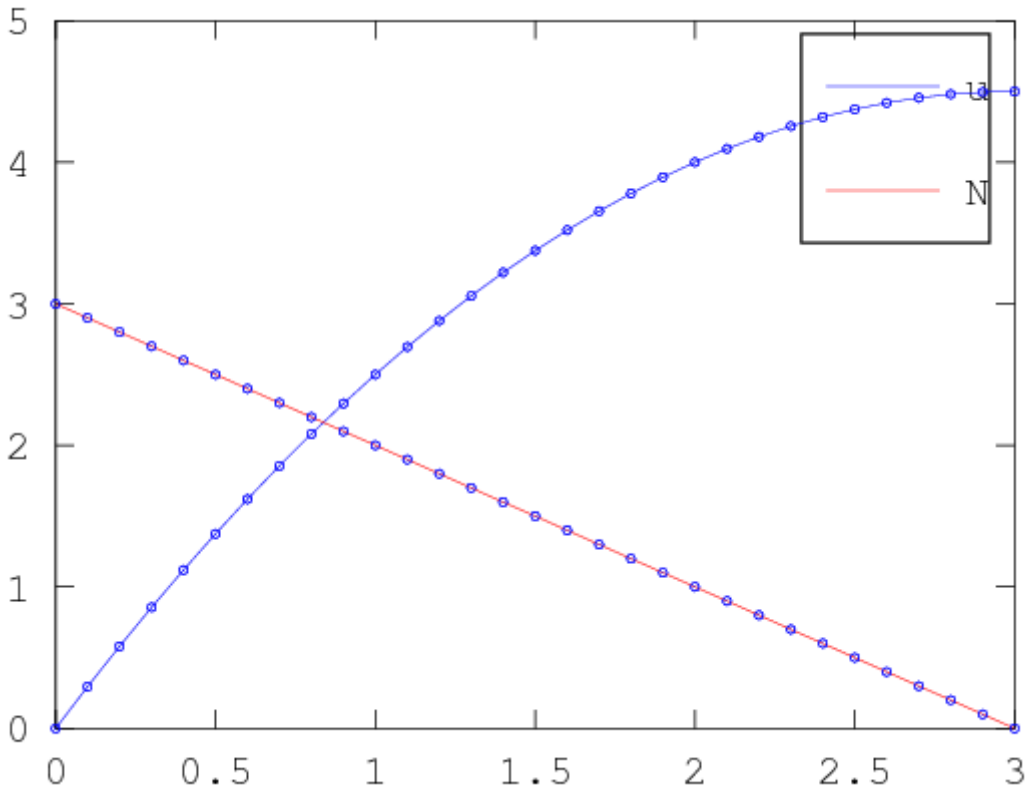
$$\frac{dN(\xi)}{d\xi} = \begin{bmatrix} \xi - \frac{1}{2} & 0 & -2\xi & 0 & \xi + \frac{1}{2} & 0 \end{bmatrix}$$

```
global N_qua = @(xi) [xi*(xi-1.)/2., 0.,-(xi+1.)*(xi-1.), 0., (xi+1.)*xi/2., 0.
];
global dN_qua = @(xi) [xi-1./2., 0., -2*xi, 0., xi+1./2., 0.];
global J_qua = @(xi, xe) dN_qua(xi)*xe';

function ans = K_truss2d_quadratic(xeg, EA)
    # Local stiffness matrix of quadratic truss element
    global dN_qua;
    global J_qua;
    T = transformation_matrix(xeg);
    xel = T*xeg';
    integrand = @(xi) dN_qua(xi)'*dN_qua(xi)/J_qua(xi,xel');
    ans = EA*numerical_integration(3, integrand);
    ans = T'*ans*T;
end

function ans = f_quadratic(xeg, f)
    global N_qua;
    global J_qua;
    T = transformation_matrix(xeg);
    xel = T*xeg';
    integrand = @(xi) N_qua(xi)'*f(xi)*J_qua(xi,xel');
    ans = numerical_integration(2, integrand);
    ans = T'*ans;
end
```

Cantilever beam:

```
function ans = N(x,l)
    ans = [(2/l^2).*(x-l/2).*(x-l), -(4/l^2).*x.*(x-l), (2/l^2).*x.*(x-l/2)];
end

function ans = B(x, l)
    ans = (1/l^2)*[4*x-3*l, -8*x+4*l, 4*x-l];
end

function ans = K_bar_2(l, EA)
    # integration points + weights
    w=[1 1]; ip=[-1/sqrt(3) 1/sqrt(3)];

    nip = size(w)(2);
    ans = zeros (3,3);
    for i=1:nip
        x = l/2+ip(i)*l/2;
        b = B(x, l);
        ans=ans+EA*b'*b*w(i)*l/2.;
    end
end

function ans = f_bar_2(l,f)
    # integration points + weights
    w=[1 1]; ip=[-1/sqrt(3) 1/sqrt(3)];

    nip = size(w)(2);
    ans = zeros (3,1);
    for i=1:nip
        x = l/2+ip(i)*l/2;
        n = N(x, l);
        ans=ans+n'*f(x)*w(i)*l/2.;
    end
end

####### Main #########
E = 1;
A = 1;
l =3;
K = K_bar_2(l, E*A);
F = f_bar_2(l, @(x) 1);

u = K(2:3,2:3)\F(2:3);
U = [0;u]

hold on;
xlim=[0 3];
x=0:0.1:l;
#analytical solution for reference
plot (x, -x.^2/2+3*x, "b;u;", x, -x+3, "r;N;");
#computed results
for x=0:0.1:3
    scatter (x, N(x,l)*U);
    scatter (x, E*A*B(x,l)*U);
end
```

```
U =

    0.00000
    3.37500
    4.50000
```



Reference: Czech course of “Numerická analýza konstrukcí” (Numerical analysis of structures) by B. Patzák (borek.patzak@fsv.cvut.cz)