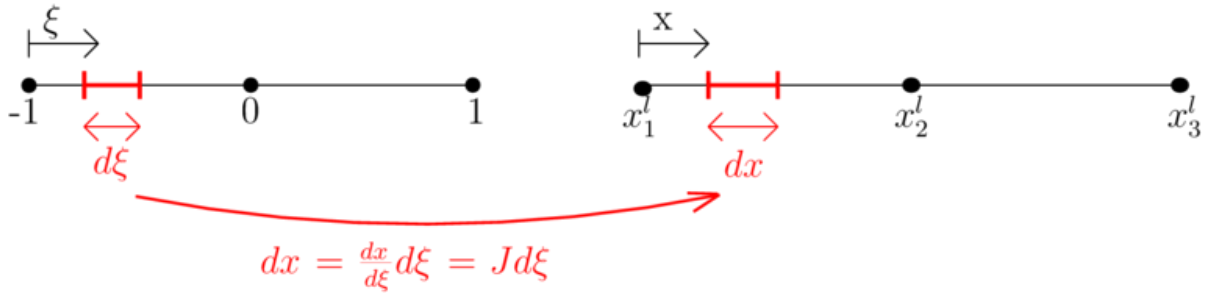
Cvičení 5a - 1D prvek v přirozených souřadnicích

Transformace do lokálních souřadnic

Protože systém přirozených souřadnic je jednorozměrný, ale konstrukce se nachází v rovině, provedeme nejdřív transformaci uzlových souřadnic prvku do lokální soustavy dané střednicí podle vztahu

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

In [1]:
```
function T = transformation_matrix(x)
    # Transformation matrix for a given vector of global element nodal coordinates 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
```



Transformace do přirozených souřadnic

Osu přirozených souřadnic pro jednorozměrný prvek zavedeme ve směru střednice. Počáteční uzel bude mít souřadnici $\xi_1 = -1$, koncový uzel $\xi_n = 1$ kde n je počet uzlů prvku.

Izoparametrický prvek

Protože prvek popisujeme v přirozených souřadnicích, potřebujeme transformační vztah vyjadřující po délce prvku kartézské souřadnice v závislosti na přirozených. Tzv. izoparametrické prvky používají pro aproximaci souřadnic stejný vztah jako pro uzlové posuny, platí tedy

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

Jakobián transformace spočteme jako derivaci skutečných souřadnic podle přirozených

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

Derivace bázových funkcí získáme s použitím pravidla pro derivaci složené funkce jako

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

Matice tuhosti prvku je definována v kartézských souřadnicích jako

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

což můžeme vyjádřit v přirozených souřadnicích jako

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

po dosazení za $B(\xi) = \frac{dN}{d\xi} J^{-1}$ dostáváme

$$K^{le} = EA \int_{-1}^{1} \frac{dN(\xi)}{d\xi}^T \frac{dN(\xi)}{d\xi} J^{-1} d\xi$$

Obdobně vyjádříme vektor zatížení jako

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

Význam přirozených souřadnic

Použití transformace prvku do přirozených souřadnic umožňuje provádět numerickou integraci na konstantním intervalu. Místo toho, aby se meze integrálu měnily v závislosti na délce prvku, integrujeme vždy na intervalu $< -1, 1 >$.

In [2]:
```
# 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
```

Lineární prvek

Vektor bázových funkcí pro lineární prvek definujeme jako

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

, jeho derivaci v přirozených souřadnicích jako

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

In [3]:
```
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
```

Kvadratický prvek

Vektor bázových funkcí pro kvadratický prvek definujeme jako

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

, jeho derivaci v přirozených souřadnicích jako

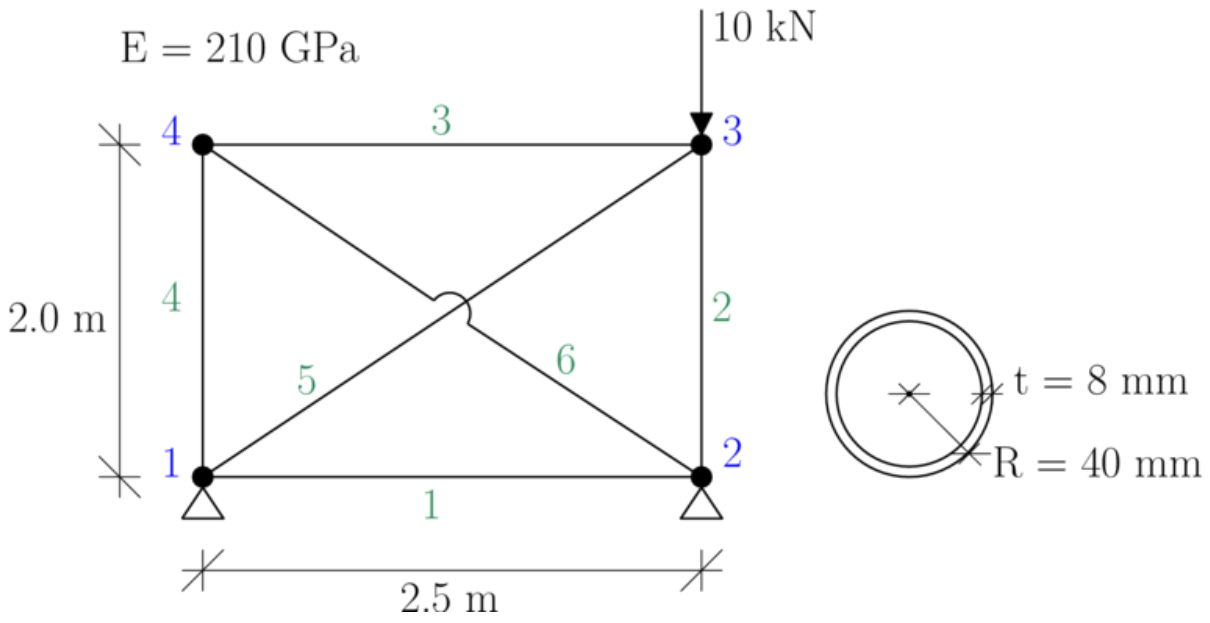
$$\frac{dN(\xi)}{d\xi} = \begin{bmatrix} \xi - \frac{1}{2} & 0 & -2\xi & 0 & \xi + \frac{1}{2} & 0 \end{bmatrix}$$

In [4]:
```
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
```

Cvičení 5b - Automatizace řešení příhradové konstrukce ve 2D



Popis geometrie konstrukce

K reprezentaci konstrukce v programu používáme uzly a prvky. Uzly musíme umístit v rovině a poté je pospojovat pomocí prvků.

In [5]:
```
% Geometrie
nodes = [0, 0;
    2.5, 0;
    2.5, 2;
    0,2];
% Konektivita
element_nodes = [1, 2;
                 2, 3;
                 3, 4;
                 4, 1;
                 1, 3;
                 2, 4];

nelems = size(element_nodes)(1);
nnodes = size(nodes)(1);
```

Ostatní data modelu

Veškerý další popis konstrukce vztahujeme na vytvořené prvky a uzly. Prvkům předepisujeme materiál resp. tuhost a případně prvkové zatížení. V uzlech specifikujeme podepření, popř. uzlové zatížení.

In [6]:
```
% Podepreni
supports = zeros(nnodes,2);
supports(1,:) = [1,1];
supports(2,:) = [0,1];

% Uzlove zatizeni
nodal_loads = zeros(nnodes,2);
nodal_loads(3,:) = [0,-10e3];

% Material / prurez
E = 210e9
A = pi * (0.04^2-0.032^2)
EA = E*A
element_stiffnesses = EA*ones(nelems,1);

% Prvkove zatizeni
%element_force_loads = zeros(nelems,1);
%element_temperature_loads = zeros(nelems,1);
```

```
E =    2.1000e+11
A =  0.0018096
EA =    3.8001e+08
```

Kódová čísla

Pro jednoznačnou identifikaci stupňů volnosti při lokalizaci používáme kódová čísla. Očíslujeme všechny stupně volnosti a zároveň podle informací o podepření vytvoříme seznam neznámých (nepodepřených) stupňů volnosti.

In [7]:
```
% Kodova cisla - projdu vsechny uzly, ocisluji posuny a vytvorim seznam neznamych
function [nodes_enum,unknowns_enum] = enumerate_unknowns(supports)
    nodes_enum = [];
    unknowns_enum = [];
    for i = 1:size(supports)(1)
        nodes_enum = [nodes_enum; [2*i-1 2*i]];
        for j = 1:2
            if supports(i,j) == 0
                unknowns_enum = [unknowns_enum,nodes_enum(i,j)];
            endif
        endfor
    endfor
end
```

Lokalizační matice

Na prvcích známe čísla odpovídajících uzlů, na uzlech čísla odpovídajících stupňů volnosti. Kombinací těchto informací získáme pro každý prvek vektor čísel odpovídajících stupňů volnosti, který udává pozici prvkových stupňů volnosti v globální matici tuhosti.

In [8]:
```
% Lokalizacni matice - projdu vsechny prvky, vytvorim vektory cisel neznamych,
% ktere prisluseji uzlum prvku
function ans = locvec(nodes,nodes_enum)
    ans = [nodes_enum(nodes(1),:), nodes_enum(nodes(2),:)];
end
```

Řešení úlohy

K řešení potřebujeme globální matici tuhosti konstrukce, kterou získáme výpočtem a následnou lokalizací prvkových matic. Druhou výchozí informací je vektor zatížení, který se skládá z příspěvků uzlových a prvkových zatížení. Výslednou soustavu tvoří rovnice příslušné neznámým stupňům volnosti.

In [9]:
```
% Sestaveni globalni matice
[nodes_enum,unknowns_enum] = enumerate_unknowns(supports);
locmat = locmat(element_nodes,nodes_enum);

kdim = nnodes*2;
k = zeros(kdim);
for i = 1:nelems
    xe1 = nodes(element_nodes(i,1),:);
    xe2 = nodes(element_nodes(i,2),:);
    ke = K_truss2d_linear([xe1 xe2],element_stiffnesses(i));
    lvec = locvec(element_nodes(i,:),nodes_enum)
    k(lvec,lvec) += ke;
endfor

% Vektor uzlovych zatizeni
f_nodal = [];
for i = 1:nnodes
    f_nodal = [f_nodal;nodal_loads(i,:)'];
endfor

% Vysledna soustava rovnic
ku = k(unknowns_enum,unknowns_enum);
ku
fu = f_nodal(unknowns_enum);
fu
u = ku\fu;
```

```
error: 'locmat' undefined near line 1 column 10
lvec =

   1   2   3   4

lvec =

   3   4   5   6

lvec =

   5   6   7   8

lvec =

   7   8   1   2

lvec =

   1   2   5   6

lvec =

   3   4   7   8

ku =

   2.2438e+08   0.0000e+00   0.0000e+00  -7.2375e+07   5.7900e+07
   0.0000e+00   2.2438e+08   5.7900e+07  -1.5200e+08   0.0000e+00
   0.0000e+00   5.7900e+07   2.3632e+08   0.0000e+00   0.0000e+00
  -7.2375e+07  -1.5200e+08   0.0000e+00   2.2438e+08  -5.7900e+07
   5.7900e+07   0.0000e+00   0.0000e+00  -5.7900e+07   2.3632e+08

fu =

       0
       0
  -10000
       0
       0
```

Vyhodnocení výsledků (postprocessing)

In [19]:

```
%% Vyhodnoceni
U = zeros(kdim,1);
U(unknowns_enum) = u
R = k*U - f_nodal

function [eps,n] = truss2d_postpro (x,EA,u,xi)
    global dN_lin
    global J_lin
    T = transformation_matrix(x);
    xl = T*x';
    ul = T*u;
    eps = dN_lin(xi)*ul/J_lin(xi,xl');
    n   = EA*eps;
end

for i=1:nelems
    lvec = locvec(element_nodes(i,:),nodes_enum);
    d=U(lvec);
    xe1 = nodes(element_nodes(i,1),:);
    xe2 = nodes(element_nodes(i,2),:);
    [eps,sig]=truss2d_postpro ([xe1 xe2], EA, d, 0);
    fprintf(1,'Element %2d\t\teps=%g, s=%g\n',i,eps,sig);
endfor

% Vykresleni posunu
factor = 5e3;
for i=1:nelems
    lvec = locvec(element_nodes(i,:),nodes_enum);
    d=U(lvec)';
    xe1 = nodes(element_nodes(i,1),:);
    xe2 = nodes(element_nodes(i,2),:);
    xdef1 = xe1 + factor*d(1:2);
    xdef2 = xe2 + factor*d(3:4);
    hold on;
    if i==nelems
        % Print with legend
        plot ([xe1(1) xe2(1)], [xe1(2) xe2(2)], "b;Default shape;", ...
        [xdef1(1) xdef2(1)], [xdef1(2) xdef2(2)], "r;Deformed shape;")
    else
        plot ([xe1(1) xe2(1)], [xe1(2) xe2(2)], "b;;", ...
        [xdef1(1) xdef2(1)], [xdef1(2) xdef2(2)], "r;;")
    endif
endfor
```

```
U =

   0.0000e+00
   0.0000e+00
   5.8281e-06
   0.0000e+00
   2.6880e-05
  -4.8901e-05
   2.1052e-05
   3.7300e-06

R =

  -5.5622e-13
  -7.7660e-14
  -2.3731e-14
   1.0000e+04
  -9.4147e-14
  -3.6380e-12
  -4.3102e-13
  -1.0048e-14

Element  1              eps=2.33122e-06, s=885.881
Element  2              eps=-2.44503e-05, s=-9291.3
Element  3              eps=2.33122e-06, s=885.881
Element  4              eps=1.86498e-06, s=708.705
Element  5              eps=-2.98542e-06, s=-1134.48
Element  6              eps=-2.98542e-06, s=-1134.48
```