Functions

This function returns the abscissas and weights of the Gauss points for ngp equal up to 4

```
function [samp]=gauss(ngp)
%
% This function returns the abscissas and weights of the Gauss points for ngp equal up to 4
%
%
samp=zeros(ngp,2);
%
if ngp==1
    samp=[0. 2];

elseif ngp==2
    samp=[-1./sqrt(3) 1.;...
        1./sqrt(3) 1.];

elseif ngp==3
    samp= [-.2*sqrt(15.) 5./9; ...
        0. 8./9.;...
        .2*sqrt(15.) 5./9];

elseif ngp==4
    samp=[-0.861136311594053 0.347854845137454; ...
        -0.339981043584856 0.652145154862546; ...
        0.339981043584856 0.652145154862546; ...
        0.861136311594053 0.347854845137454];
end

end
```

This function returns the coordinates of the nodes of element i and its steering vector

```
function [coord,g]=Element_Q8(i)
%
% This function returns the coordinates of the nodes of element i
% and its steering vector
%
global geom connec nne nodof  nf
%
l=0;
coord=zeros(nne,nodof);
for k=1:nne
    for j=1:nodof
        coord(k,j)=geom(connec(i,k),j);
        l=l+1;
        g(l)=nf(connec(i,k),j);
    end
end
```

```
end
```

This function returns the vector of the shape function and their derivatives with respect to xi and eta at the gauss points for an 8-nodded quadrilateral

```
function[der,fun]=fmquad(samp,ig,jg)
%
% This function returns the vector of the shape function and their derivatives with respect to
% the gauss points for an 8-nodded quadrilateral
%
xi=samp(ig,1);
eta=samp(jg,1);
etam=(1.-eta);
etap=(1.+eta);
xim=(1.-xi);
xip=(1.+xi);
%
fun(1) = -0.25*xim*etam*(1.+ xi + eta);
fun(2) = 0.5*(1.- xi^2)*etam;
fun(3) = -0.25*xip*etam*(1. - xi + eta);
fun(4) = 0.5*xip*(1. - eta^2);
fun(5) = -0.25*xip*etap*(1. - xi - eta);
fun(6) = 0.5*(1. - xi^2)*etap;
fun(7) = -0.25*xim*etap*(1. + xi - eta);
fun(8) = 0.5*xim*(1. - eta^2);
%
der(1,1)=0.25*etam*(2.*xi + eta);    der(1,2)=-1.*etam*xi;
der(1,3)=0.25*etam*(2.*xi-eta);      der(1,4)=0.5*(1-eta^2);
der(1,5)=0.25*etap*(2.*xi+eta);      der(1,6)=-1.*etap*xi;
der(1,7)=0.25*etap*(2.*xi-eta);      der(1,8)=-0.5*(1.-eta^2);
%
der(2,1)=0.25*xim*(2.*eta+xi); der(2,2)=-0.5*(1. - xi^2);
der(2,3)=-0.25*xip*(xi-2.*eta); der(2,4)=-1.*xip*eta;
der(2,5)=0.25*xip*(xi+2.*eta); der(2,6)=0.5*(1.-xi^2);
der(2,7)=-0.25*xim*(xi-2.*eta); der(2,8)=-1.*xim*eta;
%
end
```

This function assembles the matrix [bee] from the derivatives of the shape functions in global coordinates

```
function [bee]=formbee(deriv,nne,eldof)
%
% This function assembles the matrix [bee] from the
% derivatives of the shape functions in global coordinates
%
bee=zeros(3,eldof);
for m=1:nne
    k=2*m;
    l=k-1;
```

```
    x=deriv(1,m);
    bee(1,l)=x;
    bee(3,k)=x;
    y=deriv(2,m);
    bee(2,k)=y;
    bee(3,l)=y;
end
%
end
```

This function assembles the global stiffness matrix

```
function[KK]=form_KK(KK, kg, g)
%
% This function assembles the global stiffness matrix
%
global eldof
%
% This function assembles the global stiffness matrix
%
for i=1:eldof
    if g(i) ~= 0
        for j=1: eldof
            if g(j) ~= 0
                KK(g(i),g(j))= KK(g(i),g(j)) + kg(i,j);
            end
        end
    end
end
end
```

This function averages the stresses at the nodes

```
function[ZX, ZY, ZT, Z1, Z2]=stresses_at_nodes_Q8(SIGMA)
%
% This function averages the stresses at the nodes
%
global nnd nel nne connec
%
for k = 1:nnd
    sigx = 0. ;sigy = 0.; tau = 0.;
    ne = 0;
    for iel = 1:nel
        for jel=1:nne
            if connec(iel,jel) == k
                ne=ne+1;
                sigx = sigx + SIGMA(iel,1);
                sigy = sigy + SIGMA(iel,2);
                tau = tau + SIGMA(iel,3);
            end
```

```
        end
    end
    ZX(k,1) = sigx/ne;
    ZY(k,1) = sigy/ne;
    ZT(k,1)=  tau/ne;
    Z1(k,1)= ((sigx+sigy)/2 + sqrt(((sigx+sigy)/2)^2 +tau^2))/ne;
    Z2(k,1)= ((sigx+sigy)/2 - sqrt(((sigx+sigy)/2)^2 +tau^2))/ne;
end

end
```