Error para Q8

```
%% Recuperacion de tensiones en los nodos
stress = zeros(nel,nNodEle,3);
uNod = [-1 -1
         1 -1
         1  1
        -1  1
         0 -1
         1  0
         0  1
        -1  0];
for iele = 1:nel
    nodesEle = nodes(elements(iele,:),:);
    for inode = 1:nNodEle
        % Punto de Gauss
        ksi = uNod(inode,1);
        eta = uNod(inode,2);
        % Derivadas de las funciones de forma respecto de ksi, eta
        dN = shapefunsder([ksi eta],'Q8');
        % Derivadas de x,y, respecto de ksi, eta
        jac = dN*nodesEle;
        % Derivadas de las funciones de forma respecto de x,y.
        dNxy = jac\dN;          % dNxy = inv(jac)*dN

        B = zeros(size(C,2),nDofNod*nNodEle);
        B(1,1:2:15) = dNxy(1,:);
        B(2,2:2:16) = dNxy(2,:);
        B(3,1:2:15) = dNxy(2,:);
        B(3,2:2:16) = dNxy(1,:);

        eleDofs = nodeDofs(elements(iele,:),:);
        eleDofs = reshape(eleDofs',[],1);
        stress(iele,inode,:) = C*B*D(eleDofs);
    end
end

%% tensiones en los puntos de superconvergencia
stressSuper = zeros(nel,4,3);
uNod = [-1  -1
         1  -1
         1   1
        -1   1]/sqrt(3);
for iele = 1:nel
    nodesEle = nodes(elements(iele,:),:);
    for inode = 1:4
        % Punto de Gauss
        ksi = uNod(inode,1);
        eta = uNod(inode,2);
        % Derivadas de las funciones de forma respecto de ksi, eta
        dN = shapefunsder([ksi eta],'Q8');
        % Derivadas de x,y, respecto de ksi, eta
        jac = dN*nodesEle;
        % Derivadas de las funciones de forma respecto de x,y.
        dNxy = jac\dN;          % dNxy = inv(jac)*dN
```

```
        B = zeros(size(C,2),nDofNod*nNodEle);
        B(1,1:2:15) = dNxy(1,:);
        B(2,2:2:16) = dNxy(2,:);
        B(3,1:2:15) = dNxy(2,:);
        B(3,2:2:16) = dNxy(1,:);

        eleDofs = nodeDofs(elements(iele,:),:);
        eleDofs = reshape(eleDofs',[],1);
        stressSuper(iele,inode,:) = C*B*D(eleDofs);
    end
end

% Extrapolo a los Nodos
a = sqrt(3);
rsExt = a*[-1  -1
            1  -1
            1   1
           -1   1
            0  -1
            1   0
            0   1
           -1   0];
stressExtra = zeros(nel,nNodEle,3);
for iele = 1:nel
    for inode = 1:nNodEle
        r = rsExt(inode,1);
        s = rsExt(inode,2);

        N = shapefuns([r s],'Q4');

        stressExtra(iele,inode,:) = N * squeeze(stressSuper(iele,:,:));
    end
end

%% Gauss
[wpg, upg, npg] = gauss([3 3]);

%% calculo de eta_el, e2, U2
invC = C\eye(3);
eta_el = zeros(nel,1);
e2_el = zeros(nel,1);
U2_el = zeros(nel,1);
for iele = 1:nel
    nodesEle = nodes(elements(iele,:),:);
    for inode = 1:nNodEle
        for ipg = 1:npg
            % Punto de Gauss
            ksi = upg(ipg,1);
            eta = upg(ipg,2);

            % Derivadas de las funciones de forma respecto de ksi, eta
            dN = shapefunsder([ksi eta],'Q8');
            % Derivadas de x,y, respecto de ksi, eta
            jac = dN*nodesEle;
            % Derivadas de las funciones de forma respecto de x,y.
```

```
            dNxy = jac\dN;          % dNxy = inv(jac)*dN

            % funciones de forma
            N = shapefuns([ksi eta],'Q8');
            eleStress =  squeeze(stress(iele,inode,:));            %
tensiones "directas"
            starStress = squeeze(stressExtra(iele,inode,:)); % tensiones
mejoradas

            e2_el(iele) = e2_el(iele) + (starStress - eleStress)' * ...
                        invC * (starStress - eleStress) * wpg(ipg) *
det(jac);

            U2_el(iele) = U2_el(iele) + eleStress' * invC * eleStress * ...
                          wpg(ipg) * det(jac);
        end
    end
    eta_el(iele) = sqrt( e2_el(iele) / (e2_el(iele) + U2_el(iele)) );

end

etaG = sqrt( sum(e2_el) / (sum(e2_el) + sum(U2_el)) )


%% Configuracion deformada
D = (reshape(D,nDofNod,[]))';
nodePosition = nodes + D(:,1:2);


%Graficacion
limites = [min(min(stressExtra(:,:,2))),max(max(stressExtra(:,:,2)))];

figure(1)
bandplot(elements,nodePosition,stress(:,:,2),limites,'k');
title('Tensiones en los nodos.')

figure(2)
bandplot(elements,nodePosition,stressExtra(:,:,2),limites,'k');
title('Tensiones extrapoladas de los puntos de superconvergencia.')

figure(3)
scalarbandplot(elements,nodePosition,eta_el,[],'k',[],'flat');
```

Error para Q4

```
%% Recuperacion de tensiones en los nodos
stressNod = zeros(nel,nNodEle,3);
uNod = [-1 -1
         1 -1
         1  1
        -1  1];
for iele = 1:nel
    nodesEle = nodes(elements(iele,:),:);
    for inode = 1:nNodEle
        % Punto de Gauss
        ksi = uNod(inode,1);
        eta = uNod(inode,2);
        % Derivadas de las funciones de forma respecto de ksi, eta
        dN = shapefunsder([ksi eta],'Q4');
        % Derivadas de x,y, respecto de ksi, eta
        jac = dN*nodesEle;
        % Derivadas de las funciones de forma respecto de x,y.
        dNxy = jac\dN;          % dNxy = inv(jac)*dN

        B = zeros(size(C,2),nDofNod*nNodEle);
        B(1,1:2:nNodEle*2-1) = dNxy(1,:);
        B(2,2:2:nNodEle*2)   = dNxy(2,:);
        B(3,1:2:nNodEle*2-1) = dNxy(2,:);
        B(3,2:2:nNodEle*2)   = dNxy(1,:);

        eleDofs = nodeDofs(elements(iele,:),:);
        eleDofs = reshape(eleDofs',[],1);
        stressNod(iele,inode,:) = C*B*D(eleDofs);
    end
end

figure
bandplot(elements,nodes,stressNod(:,:,3),[],'k')

%% Tensiones en los puntos de superconvergencia
stressSuper = zeros(nel,nNodEle,3);
for iele = 1:nel
    nodesEle = nodes(elements(iele,:),:);
    for inode = 1:nNodEle
        % Punto de Gauss
        ksi = 0;
        eta = 0;
        % Derivadas de las funciones de forma respecto de ksi, eta
        dN = shapefunsder([ksi eta],'Q4');
        % Derivadas de x,y, respecto de ksi, eta
        jac = dN*nodesEle;
        % Derivadas de las funciones de forma respecto de x,y.
        dNxy = jac\dN;          % dNxy = inv(jac)*dN

        B = zeros(size(C,2),nDofNod*nNodEle);
        B(1,1:2:nNodEle*2-1) = dNxy(1,:);
        B(2,2:2:nNodEle*2)   = dNxy(2,:);
        B(3,1:2:nNodEle*2-1) = dNxy(2,:);
        B(3,2:2:nNodEle*2)   = dNxy(1,:);
```

```
        eleDofs = nodeDofs(elements(iele,:),:);
        eleDofs = reshape(eleDofs',[],1);
        stressSuper(iele,inode,:) = C*B*D(eleDofs);
    end
end

%% Promediado de tensiones en los nodos
avgStress = zeros(nNod,3);
for inode = 1:nNod
    [I,J] = find(elements == inode);
    nShare = length(I);
    for ishare = 1:nShare
        avgStress(inode,:) = avgStress(inode,:) +
squeeze(stressSuper(I(ishare),J(ishare),:))';
    end
    avgStress(inode,:) = avgStress(inode,:) / nShare;
end

%% calculo de eta_el, e2, U2
invC = C\eye(3);
eta_el = zeros(nel,1);
e2_el = zeros(nel,1);
U2_el = zeros(nel,1);
for iele = 1:nel
    nodesEle = nodes(elements(iele,:),:);
    for inode = 1:nNodEle
        for ipg = 1:npg
            % Punto de Gauss
            ksi = upg(ipg,1);
            eta = upg(ipg,2);

            % Derivadas de las funciones de forma respecto de ksi, eta
            dN = shapefunsder([ksi eta],'Q4');
            % Derivadas de x,y, respecto de ksi, eta
            jac = dN*nodesEle;
            % Derivadas de las funciones de forma respecto de x,y.
            dNxy = jac\dN;          % dNxy = inv(jac)*dN

            % funciones de forma
            N = shapefuns([ksi eta],'Q4');
            eleStress =  squeeze(stressNod(iele,inode,:));    % tensiones
"directas"
            starStress = ( N * avgStress(elements(iele,:),:) )';    %
tensiones mejoradas

            e2_el(iele) = e2_el(iele) + (starStress - eleStress)' * ...
                        invC * (starStress - eleStress) * wpg(ipg) *
det(jac);

            U2_el(iele) = U2_el(iele) + eleStress' * invC * eleStress * ...
                          wpg(ipg) * det(jac);
        end
    end
    eta_el(iele) = sqrt( e2_el(iele) / (e2_el(iele) + U2_el(iele)) );
```

```
end

etaG = sqrt( sum(e2_el) / (sum(e2_el) + sum(U2_el)) );

disp('Error global por energía de deformación: ')
disp (etaG)
fprintf('\n')
disp('Error elemental por energía de deformación: ')
disp(([1:nel; eta_el']))
```

AxisQ4 con P

```
clear
clc
close all
format short g

load('Nodos2x6Q4.mat');
load('Elementos2x6Q4.mat');

nDofNod = 2;                    % Numero de grados de libertad por nodo
nNodEle = size(elements,2);     % Numero de nodos por elemento
nel = size(elements,1);         % Numero de elementos
nNod = size(nodes,1);           % Numero de nodos
nDofTot = nDofNod*nNod;         % Numero de grados de libertad

bc = false(nNod,nDofNod);       % Matriz de condiciones de borde
bc(abs(nodes(:,2)-0) <= 1e-4 ,2) = true;
bc(abs(nodes(:,2)-2) <= 1e-4 ,2) = true;



r1 = 4; %mm
r2 = 10; %mm
d = 2; %mm

%% Gauss
a   = 1/sqrt(3);
% Ubicaciones puntos de Gauss
upg = [ -a  -a
         a  -a
         a   a
        -a   a ];
% Numero de puntos de Gauss
npg = size(upg,1);
wpg = ones(4,1);

%% Matriz Constitutiva
E = 1;
NU = 0.3;
f = NU/(1 - NU);
g = (1 - 2*NU)/(2*(1 - NU));
C = [ 1 f f 0
      f 1 f 0
      f f 1 0
      0 0 0 g ] * (1 - NU)*E/((1 + NU)*(1 - 2*NU));

%% Matriz de rigidez
K = zeros(nDofTot);
nodeDofs = reshape(1:nDofTot,nDofNod,nNod)';
for iele = 1:nel
    Ke = zeros(nDofNod*nNodEle);
    nodesEle = nodes(elements(iele,:),:);
    for ipg = 1:npg
        % Punto de Gauss
        ksi = upg(ipg,1);
        eta = upg(ipg,2);
```

```
        % Derivadas de las funciones de forma respecto de ksi, eta
        dN = shapefunsder([ksi eta],'Q4');
        N = shapefuns([ksi eta],'Q4');
        % Derivadas de x,y, respecto de ksi, eta
        jac = dN*nodesEle;
        % Derivadas de las funciones de forma respecto de x,y.
        dNxy = jac\dN;          % dNxy = inv(jac)*dN

        r = N*nodesEle(:,1);

        B = zeros(size(C,2),nDofNod*nNodEle);
        B(1,1:2:7) = dNxy(1,:);
        B(2,1:2:7) = N/r;
        B(3,2:2:8) = dNxy(2,:);
        B(4,1:2:7) = dNxy(2,:);
        B(4,2:2:8) = dNxy(1,:);

        Ke = Ke + B'*C*B*wpg(ipg)*det(jac)*r;
    end
    eleDofs = nodeDofs(elements(iele,:),:);
    eleDofs = reshape(eleDofs',[],1);
    K(eleDofs,eleDofs) = K(eleDofs,eleDofs) + Ke;
end

%% Carga
p = 1e-1; % [N/mm^2]
% Ubicaciones puntos de Gauss
upg = [-a a];
% Numero de puntos de Gauss
npg = size(upg,2);
wpg = [1 1];
R = zeros(nNod,nDofNod);        % Vector de cargas
nodeDofs = reshape(1:nDofTot,nDofNod,nNod)';
for iele = 1:nel
    nodesEle = nodes(elements(iele,:),:);
    if abs(nodesEle(1,1)-r1) <= 1e-4
        for ipg = 1:npg
            % Punto de Gauss
            ksi = -1;
            eta = upg(ipg);
            % Derivadas de las funciones de forma respecto de ksi, eta
            dN = shapefunsder([ksi eta],'Q4');
            % Derivadas de x,y, respecto de ksi, eta
            jac = dN*nodesEle;
            % Derivadas de las funciones de forma respecto de x,y.
            dNxy = jac\dN;          % dNxy = inv(jac)*dN
            N = shapefuns([ksi eta],'Q4');
            r = N*nodesEle(:,1);
            N = N([1 4]);
            R(elements(iele,[1 4]),1) = R(elements(iele,[1 4]),1) + ...
                N'*N*p*ones(2,1)*jac(2,2)*wpg(ipg)*r;
        end
    end
end
%% Reduccion Matriz
isFixed = reshape(bc',[],1);
```

```
isFree = ~isFixed;

Rr = reshape(R',[],1);

% Solver
Dr = K(isFree,isFree)\Rr(isFree);

% Reconstruccion
D = zeros(nDofTot,1);
D(isFree) = D(isFree) + Dr;

% Reacciones
Rv = K(isFixed,isFree)*D(isFree);
reacciones = nan(nDofTot,1);
reacciones(isFixed) = Rv;
reacciones = (reshape(reacciones,nDofNod,[]))';

% D = (reshape(D,nDofNod,[]))';
% nodePosition = nodes + D(:,1:2);

% figure
% meshplot(elements,nodes,'b')
% hold on
% meshplot(elements,nodePosition,'r')

%% Recuperacion de tensiones en los nodos
stress = zeros(nel,nNodEle,4);
uNod = [-1 -1
         1 -1
         1  1
        -1  1];
for iele = 1:nel
    nodesEle = nodes(elements(iele,:),:);
    for inode = 1:nNodEle
        % Punto de Gauss
        ksi = uNod(inode,1);
        eta = uNod(inode,2);
        % Derivadas de las funciones de forma respecto de ksi, eta
        dN = shapefunsder([ksi eta],'Q4');
        N = shapefuns([ksi eta],'Q4');
        % Derivadas de x,y, respecto de ksi, eta
        jac = dN*nodesEle;
        % Derivadas de las funciones de forma respecto de x,y.
        dNxy = jac\dN;          % dNxy = inv(jac)*dN

        r = N*nodesEle(:,1);

        B = zeros(size(C,2),nDofNod*nNodEle);
        B(1,1:2:7) = dNxy(1,:);
        B(2,1:2:7) = N/r;
        B(3,2:2:8) = dNxy(2,:);
        B(4,1:2:7) = dNxy(2,:);
        B(4,2:2:8) = dNxy(1,:);

        eleDofs = nodeDofs(elements(iele,:),:);
```

```
        eleDofs = reshape(eleDofs',[],1);
        stress(iele,inode,:) = C*B*D(eleDofs);
    end
end

%% tensiones en los puntos de superconvergencia
stressSuper = zeros(nel,nNodEle,4);
for iele = 1:nel
    nodesEle = nodes(elements(iele,:),:);
    ksi = 0;
    eta = 0;
    dN = shapefunsder([ksi eta],'Q4');
    N = shapefuns([ksi eta],'Q4');
    for inode = 1:nNodEle
        jac = dN*nodesEle;
        % Derivadas de las funciones de forma respecto de x,y.
        dNxy = jac\dN;          % dNxy = inv(jac)*dN

        r = N*nodesEle(:,1);

        B = zeros(size(C,2),nDofNod*nNodEle);
        B(1,1:2:7) = dNxy(1,:);
        B(2,1:2:7) = N/r;
        B(3,2:2:8) = dNxy(2,:);
        B(4,1:2:7) = dNxy(2,:);
        B(4,2:2:8) = dNxy(1,:);

        eleDofs = nodeDofs(elements(iele,:),:);
        eleDofs = reshape(eleDofs',[],1);
        stressSuper(iele,inode,:) = C*B*D(eleDofs);
    end
end

%% promediado de tensiones en los nodos
avgStress = zeros(nNod,4);
for inode = 1:nNod
    [I,J] = find(elements == inode);
    nShare = length(I);
    for ishare = 1:nShare
        avgStress(inode,:) = avgStress(inode,:) +
squeeze(stressSuper(I(ishare),J(ishare),:))';
    end
    avgStress(inode,:) = avgStress(inode,:) / nShare;
end

%% Configuracion deformada
D = (reshape(D,nDofNod,[]))';
nodePosition = nodes + D(:,1:2);

%Graficacion
% limites = [min(min(stressExtra(:,:,2))),max(max(stressExtra(:,:,2)))];

figure(1)
bandplot(elements,nodePosition,stress(:,:,2),[],'k');
title('Tensiones en el centro de cada elemento.')
```

```
figure(2)
scalarbandplot(elements,nodePosition,avgStress(:,2),[],'k',[],'interp');
title('Tensiones promediadas.')

% figure(3)
% scalarbandplot(elements,nodePosition,eta_el,[],'k',[],'flat');
```

Axis Q8 con omega

```
%% Gauss
[wpg, upg, npg] = gauss([3 3]);

%% Matriz Constitutiva (plane stress)
E = 1000;
NU = 0.3;
f = NU/(1 - NU);
g = (1 - 2*NU)/(2*(1 - NU));
C = [ 1 f f 0
      f 1 f 0
      f f 1 0
      0 0 0 g ] * (1 - NU)*E/((1 + NU)*(1 - 2*NU));

%% Matriz de rigidez
K = zeros(nDofTot);
nodeDofs = reshape(1:nDofTot,nDofNod,nNod)';
for iele = 1:nel
    Ke = zeros(nDofNod*nNodEle);
    nodesEle = nodes(elements(iele,:),:);
    for ipg = 1:npg
        % Punto de Gauss
        ksi = upg(ipg,1);
        eta = upg(ipg,2);
        % Derivadas de las funciones de forma respecto de ksi, eta
        dN = shapefunsder([ksi eta],'Q8');
        N = shapefuns([ksi eta],'Q8');
        % Derivadas de x,y, respecto de ksi, eta
        jac = dN*nodesEle;
        % Derivadas de las funciones de forma respecto de x,y.
        dNxy = jac\dN;          % dNxy = inv(jac)*dN

        r = N*nodesEle(:,1);

        B = zeros(size(C,2),nDofNod*nNodEle);
        B(1,1:2:nNodEle*nDofNod-1) = dNxy(1,:);
        B(2,1:2:nNodEle*nDofNod-1) = N/r;
        B(3,2:2:nNodEle*nDofNod)   = dNxy(2,:);
        B(4,1:2:nNodEle*nDofNod-1) = dNxy(2,:);
        B(4,2:2:nNodEle*nDofNod)   = dNxy(1,:);

        Ke = Ke + B'*C*B*wpg(ipg)*det(jac)*r;
    end
    eleDofs = nodeDofs(elements(iele,:),:);
    eleDofs = reshape(eleDofs',[],1);
    K(eleDofs,eleDofs) = K(eleDofs,eleDofs) + Ke;
```

```
end

%% Carga
omega = 10; % [rad/s]
rho = 3; % [kg/m^3]
o2r = omega^2*rho;
R = zeros(nNod,nDofNod);         % Vector de cargas
[wpg, upg, npg] = gauss([2 2]);
% upg = upg([1 3 4 2]);
for iele = 1:nel
    nodesEle = nodes(elements(iele,:),:);
    for ipg = 1:npg
        % Punto de Gauss
        ksi = upg(ipg,1);
        eta = upg(ipg,2);
        % Derivadas de las funciones de forma respecto de ksi, eta
        dN = shapefunsder([ksi eta],'Q8');
        % Derivadas de x,y, respecto de ksi, eta
        jac = dN*nodesEle;
        % Derivadas de las funciones de forma respecto de x,y.
        dNxy = jac\dN;           % dNxy = inv(jac)*dN
        N = shapefuns([ksi eta],'Q8');
        r = N*nodesEle(:,1);
        R(elements(iele,:),1) = R(elements(iele,:),1) + ...
            N'*o2r*r^2*det(jac)*wpg(ipg);
    end
end
%% Reduccion Matriz
isFixed = reshape(bc',[],1);
isFree = ~isFixed;

Rr = reshape(R',[],1);

% Solver
Dr = K(isFree,isFree)\Rr(isFree);

% Reconstruccion
D = zeros(nDofTot,1);
D(isFree) = D(isFree) + Dr;

% Reacciones
Rv = K(isFixed,isFree)*D(isFree);
reacciones = nan(nDofTot,1);
reacciones(isFixed) = Rv;
reacciones = (reshape(reacciones,nDofNod,[]))';

% D = (reshape(D,nDofNod,[]))';
% nodePosition = nodes + D(:,1:2);
%
% figure
% meshplot(elements,nodes,'b')
% hold on
% meshplot(elements,nodePosition,'r')

%% Recuperacion de tensiones en los nodos
```

```
stress = zeros(nel,nNodEle,4);
uNod = [-1 -1
         1 -1
         1  1
        -1  1
         0 -1
         1  0
         0  1
        -1  0];
for iele = 1:nel
    nodesEle = nodes(elements(iele,:),:);
    for inode = 1:nNodEle
        % Punto de Gauss
        ksi = uNod(inode,1);
        eta = uNod(inode,2);
        % Derivadas de las funciones de forma respecto de ksi, eta
        dN = shapefunsder([ksi eta],'Q8');
        N = shapefuns([ksi eta],'Q8');
        % Derivadas de x,y, respecto de ksi, eta
        jac = dN*nodesEle;
        % Derivadas de las funciones de forma respecto de x,y.
        dNxy = jac\dN;          % dNxy = inv(jac)*dN

        r = N*nodesEle(:,1);

        B = zeros(size(C,2),nDofNod*nNodEle);
        B(1,1:2:nNodEle*nDofNod-1) = dNxy(1,:);
        B(2,1:2:nNodEle*nDofNod-1) = N/r;
        B(3,2:2:nNodEle*nDofNod)   = dNxy(2,:);
        B(4,1:2:nNodEle*nDofNod-1) = dNxy(2,:);
        B(4,2:2:nNodEle*nDofNod)   = dNxy(1,:);

        eleDofs = nodeDofs(elements(iele,:),:);
        eleDofs = reshape(eleDofs',[],1);
        stress(iele,inode,:) = C*B*D(eleDofs);
    end
end

%% tensiones en los puntos de superconvergencia
stressSuper = zeros(nel,4,4);
uNod = [-1 -1
         1 -1
         1  1
        -1  1]/sqrt(3);
for iele = 1:nel
    nodesEle = nodes(elements(iele,:),:);
    for inode = 1:4
        ksi = uNod(inode,1);
        eta = uNod(inode,2);
        dN = shapefunsder([ksi eta],'Q8');
        N = shapefuns([ksi eta],'Q8');
        % Derivadas de x,y, respecto de ksi, eta
        jac = dN*nodesEle;
        % Derivadas de las funciones de forma respecto de x,y.
        dNxy = jac\dN;          % dNxy = inv(jac)*dN
```

```
        r = N*nodesEle(:,1);

        B = zeros(size(C,2),nDofNod*nNodEle);
        B(1,1:2:nNodEle*nDofNod-1) = dNxy(1,:);
        B(2,1:2:nNodEle*nDofNod-1) = N/r;
        B(3,2:2:nNodEle*nDofNod)   = dNxy(2,:);
        B(4,1:2:nNodEle*nDofNod-1) = dNxy(2,:);
        B(4,2:2:nNodEle*nDofNod)   = dNxy(1,:);

        eleDofs = nodeDofs(elements(iele,:),:);
        eleDofs = reshape(eleDofs',[],1);
        stressSuper(iele,inode,:) = C*B*D(eleDofs);
    end
end

% Extrapolación a los Nodos
rsExt = [-1  -1
          1  -1
          1   1
         -1   1
          0  -1
          1   0
          0   1
         -1   0]*sqrt(3);
stressExtra = zeros(nel,nNodEle,4);
for iele = 1:nel
    for inode = 1:nNodEle
        rr = rsExt(inode,1);
        s = rsExt(inode,2);

        N = shapefuns([rr s],'Q4');

        stressExtra(iele,inode,:) = N * squeeze(stressSuper(iele,:,:));
    end
end

%% promediado de tensiones en los nodos
avgStress = zeros(nNod,4);
for inode = 1:nNod
    [I,J] = find(elements == inode);
    nShare = length(I);
    for ishare = 1:nShare
        avgStress(inode,:) = avgStress(inode,:) +
squeeze(stressExtra(I(ishare),J(ishare),:))';
    end
    avgStress(inode,:) = avgStress(inode,:) / nShare;
end



%% Configuracion deformada
D = (reshape(D,nDofNod,[]))';
nodePosition = nodes + D(:,1:2);

%Graficacion
```

```
% limites = [min(min(stressExtra(:,:,2))),max(max(stressExtra(:,:,2)))];

figure(1)
bandplot(elements,nodePosition,stress(:,:,2),[],'k');
title('Tensiones en los nodos.')

figure(2)
bandplot(elements,nodePosition,stressExtra(:,:,2),[],'k');
title('Tensiones extrapoladas.')

% figure(3)
% scalarbandplot(elements,nodePosition,eta_el,[],'k',[],'flat');
```

Línea recta rígida

```
K2 = K;
R2 = reshape(R',[],1);
bc2 = reshape(bc',[],1);
[i,j]=find((abs(nodes(:,1)-3) <= 1e-3) & (abs(nodes(:,2)-0) <= 1e-3));
masterDof = nodeDofs(i,j);
c = 0;
for iele = 1:nel
    nodesEle = nodes(elements(iele,:),:);
    if abs(nodesEle(2,1)-3) <= 1e-3
        retained = zeros(size(K2,1),1);
        eleDofs = nodeDofs(elements(iele,3),1);
        eleDofs = reshape(eleDofs',[],1); % Si fuese mas de un valor sirve
esto
        retained([eleDofs masterDof]) = [1; -1];
        K2 = [K2 retained
              retained' 0];
        R2 = [R2; 0];
        bc2 = [bc2; 0];
        c = c + 1;
    end
end

%% Reduccion Matriz

isFixed = logical(bc2);
isFree = ~isFixed;

% Solver
Dr2 = K2(isFree,isFree)\R2(isFree);

% Reconstruccion
D2 = zeros(nDofTot+c,1);
D2(isFree) = D2(isFree) + Dr2;

% Reacciones
Rv2 = K2(isFixed,isFree)*D2(isFree);
reacciones2 = nan(nDofTot+c,1);
reacciones2(isFixed) = Rv2;
reacciones2 = (reshape(reacciones2(1:end-c),nDofNod,[]))';

%% Graficos
figure
meshplot(elements,nodes,'b')
hold on
Dreshaped2 = (reshape(D2(1:end-c),nDofNod,[]))';
nodePosition2 = nodes + Dreshaped2(:,1:2);
meshplot(elements,nodePosition2,'r')
```

Rigid link por lagrange

```
%% Rigid Links

CM = zeros(7,GdLTot);

u30 = GdLNodos(30,1);
u31 = GdLNodos(31,1);
u32 = GdLNodos(32,1);
u38 = GdLNodos(38,1);
u44 = GdLNodos(44,1);

v30 = GdLNodos(30,2);
v31 = GdLNodos(31,2);
v32 = GdLNodos(32,2);
v38 = GdLNodos(38,2);
v44 = GdLNodos(44,2);

CM(1,[u30 u38 u44])=[1 -2 1];
CM(2,[v31 v30 u30 u38])=[1 -1 -1 1];
CM(3,[v38 v30])=[1 -1];
CM(4,[v44 v30])=[1 -1];
CM(5,[u31 u30])=[1 -1];
CM(6,[u32 u30])=[1 -1];
CM(7,[v32 v30 u30 u38])=[1 -1 -2 2];
% CM(8,[u45 u38 u30])=[1 -2 1];
% CM(9,[v45 v30 u38 u30])=[1 -1 2 -2];

Q = zeros(7,1);

constrained = false(GdLTot,1);
constrained([v31 v32 v38 v44 u31 u32 u44 ]) = true;
released = ~constrained;

nconst = length(find(constrained));
Kc = [ K CM' ; CM zeros(nconst) ];
R = [ R ; zeros(nconst,1) + Q];

%% Resolucion
% Reduccion
Fijo = reshape(bc',[],1);
Libre = ~Fijo;
Libre = [ Libre ; true(nconst,1) ];
cond( Kc(Libre,Libre) );        %% Numero de condicionamiento: Infinito = No
Inversible (o sea MAL)
% Solver
Dr = Kc(Libre,Libre)\R(Libre);
% Reconstruyo
D = zeros(GdLTot,1);
Libre = ~Fijo;
D(Libre) = D(Libre) + Dr(1 : end - nconst);
% Desplazamientos
D = (reshape(D,GdL,[]))';
CD = Nodos + D(:,1:2);
```