Funciones de forma

```
syms x y real
X = [1 x y x^2 x*y];
A = [1 -1 -1  1  1
     1  1 -1  1 -1
     1  1  1  1  1
     1 -1  1  1 -1
     1  0  0  0  0];
N = X/A;
dNx = diff(N,x);
dNy = diff(N,y);
dN = [dNx; dNy];
sum(N);  % == 1
sum(dN); % == 0
```

Constitutiva

```
% Plane strain
C = [1-nu  nu  0
     nu 1-nu 0
     0 0 (1-2*nu)/2]*E/(1+nu)/(1-2*nu);
% Plane stress
C = [1 nu 0
     nu 1 0
     0 0 (1-nu)/2]*E/(1-nu^2);
```

Nodos, elementos y DOF

```
%% Nodos y elementos
nod = [];
nnod = size(nod,1);
elem = [];
nelem = size(elem,1);
nnodelem = size(elem,2);
% meshplot(elem,nod,'b')
%% DOF
ndofnod = 2;
doftot = ndofnod*nnod;
dof = reshape(1:doftot,ndofnod,nnod)';
```

Integración por gauss

```
%% Gauss1D 2 puntos
a   = 1/sqrt(3);
upg = [-a a];
npg = size(upg,2);
wpg = ones(npg,1);

%% Gauss1D 3 puntos
a = sqrt(0.6);
upg = [-a 0 a];
npg = size(upg,2);
wpg = [5 8 5]/9;

%% Gauss grado 2 tringular
a   = 1/2;
upg = [a  0
       0  a
       a  a];
npg = size(upg,1);
wpg = ones(npg,1)/3;

%% Gauss para regla de 2x2
a   = 1/sqrt(3);
upg = [ -a  -a
        -a   a
         a  -a
         a   a ];
npg = size(upg,1);
wpg = ones(npg,1);

%% Gauss para regla de 3x3
a   = sqrt(0.6);
upg = [ -a  -a
        -a   0
        -a   a
         0  -a
         0   0
         0   a
         a  -a
         a   0
         a   a ];
npg = size(upg,1);
wpg = [5/9, 5/9, 5/9, 5/9, 8/9, 5/9, 5/9, 5/9, 5/9];
```

Matriz de rigidez

```
%% Matriz de rigidez Q4
Kglobal = zeros(doftot);
for e = 1:nelem
    nodelem = nod(elem(e,:),:);
    Ke = zeros(nodporelem*dofpornodo);
    for ipg = 1:npg
        ksi = upg(ipg,1);
        eta = upg(ipg,2);

        dNke = shapefunsder([ksi eta],eleT);
        J = dNke*nodelem;
        dNxy = J\dNke;

        B = zeros(size(C,2),nodporelem*dofpornodo);
        B(1,1:2:7) = dNxy(1,:);
        B(2,2:2:8) = dNxy(2,:);
        B(3,1:2:7) = dNxy(2,:);
        B(3,2:2:8) = dNxy(1,:);

        Ke = Ke + B'*C*B*t*wpg(ipg)*det(J);
    end
    dofs = reshape(dof(elem(e,:),:)',[],1);
    Kglobal(dofs,dofs) = Kglobal(dofs,dofs) + Ke;
end
```

BC y Solver

```
%% BC
fijo = zeros(nnod,ndofnod);
fijo = logical(reshape(fijo',[],1));
libre = ~fijo;

%% Solver
Dred = Kglobal(libre,libre)\P(libre);

D = zeros(doftot,1);
D(libre) = Dred;

nodfinal = nod+reshape(D,dofpornodo,nnod)'*1000;
figure
hold on; grid on; axis equal;
plot(nod(:,1),nod(:,2),'.')
plot(nodfinal(:,1),nodfinal(:,2),'.')
```

Cargas superficiales

```
%% Cargas superficiales
q =@(y) 1; % [N/m^2]
Q = integral(q,0,1);
qcheck = 0;

for iele = 1:nel
    nodesEle = nodes(elements(iele,:),:);
    if nodesEle(4,2)==60
        for ipg = 1:npg
            ksi = 1;
            eta = upg(ipg);
            N = [];
            dN = [];
            jac = dN*nodesEle;
            N = [N(2) N(3)];
            Q = [q(nodesEle(2,2)) q(nodesEle(3,2))]';
            R(elements(iele,2:3),1) = R(elements(iele,2:3),1) ...
                + N'*N*t*wpg(ipg)*Q*jac(2,2);
            qcheck = qcheck + sum(N'*N*t*wpg(ipg)*Q*jac(2,2));
        end
    end
end


% Evaluando en puntos de gauss estructurales
    if nodesEle(3,1)==30&nodesEle(3,2)<=52.5
        for ipg = 1:npg
            ksi = 1;
            eta = upg(ipg);
            N = [];
            dN = [];
            delt = (nodesEle(3,2)-nodesEle(2,2))/2;
            cennod = (nodesEle(3,2)+nodesEle(2,2))/2;
            jac = dN*nodesEle;
            R(elements(iele,2),1) = R(elements(iele,2),1) + ...
                N(2)*t*wpg(ipg)*q(cennod+delt*eta)*jac(2,2);
            R(elements(iele,3),1) = R(elements(iele,3),1) + ...
                N(3)*t*wpg(ipg)*q2(cennod+delt*eta)*jac(2,2);
            qcheck = qcheck + ...
                (N(2)+N(3))*t*wpg(ipg)*q(cennod+delt*eta)*jac(2,2);
        end
    end
```

Cargas volumétricas

```
%% Cargas volumétricas
R = zeros(nNod,nDofNod);
for e = 1:nel
    nodesEle = nodes(elements(e,:),:);
    for ipg = 1:npg
        ksi = upg(ipg,1);
        eta = upg(ipg,2);
        N = [];
        dN = [];
        jac = dN*nodesEle;
        R(elements(e,:),2) = R(elements(e,:),2) + N'*rho*g*t*wpg(ipg)*det(jac);
    end
end
```

Tensiones en los nodos

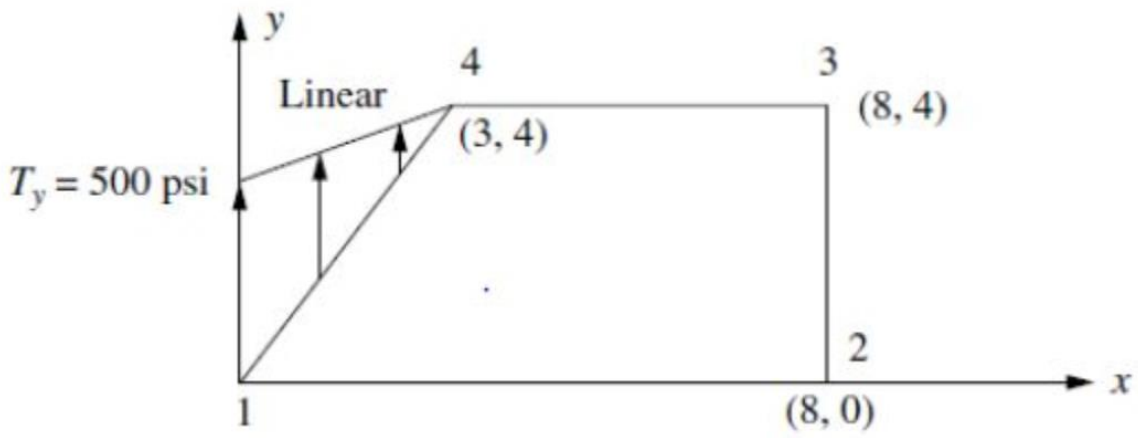
```
%% Tensiones
stress = zeros(nelem,nodporelem,3);
unod = [ -1 -1
          1 -1
          1  1
         -1  1 ];
for e = 1:nelem
    nodelem = nod(elem(e,:),:);
    for in = 1:nodporelem
        ksi = unod(in,1);
        eta = unod(in,2);
        dNke = [];
        J = dNke*nodelem;
        dNxy = J\dNke;
        B = zeros(size(C,2),dofpornodo*nodporelem);
        B(1,1:2:7) = dNxy(1,:);
        B(2,2:2:8) = dNxy(2,:);
        B(3,1:2:7) = dNxy(2,:);
        B(3,2:2:8) = dNxy(1,:);
        dofs = reshape(dof(elem(e,:),:)',[],1);
        stress(e,in,:) = C*B*D(dofs);
    end
end
% En lo puntos de Gauss es igual pero evaluado en upg.
```

Caso 2

```
%% Carga distribuida en elementos isoparamétricos
clear; close all; clc
%% Datos
E = 210E9; %[Pa]
nu = 0.3;
lam = E*nu/(1+nu)/(1-2*nu);
mu = E/2/(1+nu);
t = 1;
eleT = 'Q4';

%% Nodos y elementos
nod = [0 0
       8 0
       3 4
       8 4]; % [m]
nnod = size(nod,1);
elem = [1 2 4 3];
nelem = size(elem,1);
nodporelem = 4;

%% Funciones de forma
N1 =@(x,y) (1-x)*(1-y)/4;
N2 =@(x,y) (1+x)*(1-y)/4;
N3 =@(x,y) (1+x)*(1+y)/4;
N4 =@(x,y) (1-x)*(1+y)/4;

%% DOF
dofpornodo = 2;
doftot = dofpornodo*nnod;
dof = reshape((1:doftot)',dofpornodo,nnod)';

%% Constitutiva
% Plane stress
C = [1 nu 0
     nu 1 0
     0 0 1-nu]*E/(1-nu^2);

%% Matriz
[wpg, upg, npg] = gauss([2 2]);
Kglobal = zeros(doftot);
for e = 1:nelem
    nodelem = nod(elem(e,:),:);
    Ke = zeros(nodporelem*dofpornodo);
    for ipg = 1:npg
        ksi = upg(ipg,1);
        eta = upg(ipg,2);
        dNke = shapefunsder([ksi eta],eleT);
        J = dNke*nodelem;
        dNxy = J\dNke;
        B = zeros(size(C,2),nodporelem*dofpornodo);
        B(1,1:2:7) = dNxy(1,:);
        B(2,2:2:8) = dNxy(2,:);
```

```
        B(3,1:2:7) = dNxy(2,:);
        B(3,2:2:8) = dNxy(1,:);
        Ke = Ke + B'*C*B*t*wpg(ipg)*det(J);
    end
    dofs = reshape(dof(elem(e,:),:)',[],1);
    Kglobal(dofs,dofs) = Kglobal(dofs,dofs) + Ke;
end

%% Cargas
P = zeros(doftot/2,2);
f =@(x) 5000000000*(1-x/3);
tita = atan(3/4);
npg = 2;
[wpg, upg] = gauss1D(npg);
for e = 1:nelem
    nodelem = nod(elem(e,:),:);
    if nodelem(4,1)/nodelem(4,2) == 3/4
        for ig = 1:npg
            ksi = -1;
            eta = upg(ig);
            dNke = shapefunsder([ksi eta],eleT);
            J = dNke*nodelem;
            delta = (nodelem(4,1)-nodelem(1,1))/2*(1-upg(2));
            phiy = [N1(ksi,eta) N4(ksi,eta)]*[f(nodelem(1,1)+delta)
f(nodelem(4,1)-delta)]';
            sig = phiy*sin(tita);
            tau = phiy*cos(tita);
            P(elem(e,1),:) = P(elem(e,1),:) +
N1(ksi,eta)*wpg(ig)*t*[tau*J(2,1)-sig*J(2,2) tau*J(2,2)+sig*J(2,1)];
            P(elem(e,4),:) = P(elem(e,4),:) +
N4(ksi,eta)*wpg(ig)*t*[tau*J(2,1)-sig*J(2,2) tau*J(2,2)+sig*J(2,1)];
        end
    end
end
P = reshape(P',[],1);

%% BC
fijo = zeros(doftot/2,2);
fijo(nod(:,1)==8,:) = true;
fijo = reshape(fijo',[],1);
libre = ~fijo;

%% Solver
Dred = Kglobal(libre,libre)\P(libre);
D = zeros(doftot,1);
D(libre) = Dred;
meshplot(elem,nod,'r')
hold on
nodfinal = nod+reshape(D,dofpornodo,nnod)';
meshplot(elem,nodfinal,'b')

%% Tensiones
stress = zeros(nelem,nodporelem,3);
unod = [ -1 -1
          1 -1
          1  1
```

```
         -1   1 ];
for e = 1:nelem
    nodelem = nod(elem(e,:),:);
    for in = 1:nodporelem
        % Punto de Gauss
        ksi = unod(in,1);
        eta = unod(in,2);
        % Derivadas de las funciones de forma respecto de ksi, eta
        dNke = shapefunsder([ksi eta],eleT);
        % Derivadas de x,y, respecto de ksi, eta
        J = dNke*nodelem;
        % Derivadas de las funciones de forma respecto de x,y.
        dNxy = J\dNke;
        B = zeros(size(C,2),dofpornodo*nodporelem);
        B(1,1:2:7) = dNxy(1,:);
        B(2,2:2:8) = dNxy(2,:);
        B(3,1:2:7) = dNxy(2,:);
        B(3,2:2:8) = dNxy(1,:);
        dofs = reshape(dof(elem(e,:),:)',[],1);
        stress(e,in,:) = C*B*D(dofs);
    end
end

bandplot(elem,nodfinal,stress(:,:,1),[],'k')
```



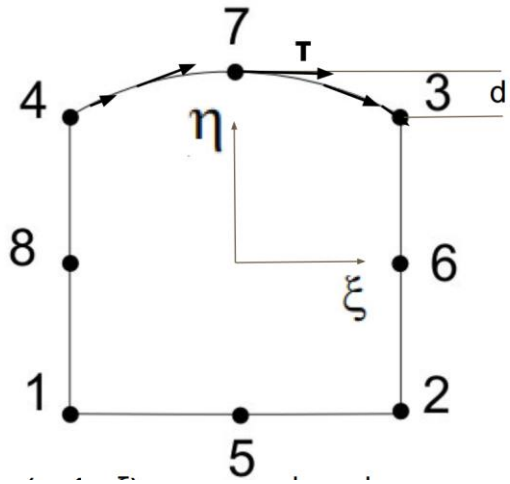
Caso 3

```
%% Caso 3, Q8 con corte tipo piramide de guiza, no de guiso
clear; close all; clc
%% Datos
E = 210E9; %[Pa]
nu = 0.3;
lam = E*nu/(1+nu)/(1-2*nu);
mu = E/2/(1+nu);
t = 1;
eleT = 'Q8';
cargaT = 'Superficie';
%% Nodos y elementos
d = .2; %[m]
nod = [0 0
       1 0
       2 0
       0 1
       2 1
       0 2
       1 2+d
       2 2]; % [m]
nnod = size(nod,1);
elem = [1 3 8 6 2 5 7 4];
nelem = size(elem,1);
nodporelem = 8;

%% DOF
dofpornodo = 2;
doftot = dofpornodo*nnod;
dof = reshape((1:doftot)',dofpornodo,nnod)';

%% Constitutiva
% Plane stress
C = [1 nu 0
     nu 1 0
     0 0 1-nu]*E/(1-nu^2);

%% Matriz
[wpg, upg, npg] = gauss([3 3]);
Kglobal = zeros(doftot);
for e = 1:nelem
    nodelem = nod(elem(e,:),:);
    Ke = zeros(nodporelem*dofpornodo);
    for ipg = 1:npg
        ksi = upg(ipg,1);
        eta = upg(ipg,2);
        dNke = shapefunsder([ksi eta],eleT);
        J = dNke*nodelem;
        dNxy = J\dNke;
        B = zeros(size(C,2),nodporelem*dofpornodo);
        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,:);
```

```
        Ke = Ke + B'*C*B*t*wpg(ipg)*det(J);
    end
    dofs = reshape(dof(elem(e,:),:)',[],1);
    Kglobal(dofs,dofs) = Kglobal(dofs,dofs) + Ke;
end

%% Cargas
switch cargaT
%% Cargas
    case 'Superficie'
P = zeros(doftot/2,2);
T = 10000000000; %[Pa]
% tauizq =@(x) T*(x+1)/2;
% tauder =@(x) T*(1-x)/2;
tauizq =@(x) T*x;
tauder =@(x) T*(2-x);
npg = 2;
[wpg, upg] = gauss1D(npg);
for e = 1:nelem
    nodelem = nod(elem(e,:),:);
%     if nodelem(4,:) == [0, 2]
%         for ig = 1:npg %izquierda
%             ksi = upg(ig);
%             eta = 1;
%             N = shapefuns([ksi,eta],eleT);
%             dNke = shapefunsder([ksi eta],eleT);
%             J = dNke*nodelem;
%             tau = tauizq(ksi);
%             P(elem(e,4),:) = P(elem(e,4),:) + N(4)*wpg(ig)*t*[tau*J(1,1)
tau*J(1,2)];
%             P(elem(e,7),:) = P(elem(e,7),:) + N(7)*wpg(ig)*t*[tau*J(1,1)
tau*J(1,2)];
%         end
%         for ig = 1:npg %derecha
%             ksi = upg(ig);
%             eta = 1;
%             N = shapefuns([ksi,eta],eleT);
%             dNke = shapefunsder([ksi eta],eleT);
%             J = dNke*nodelem;
%             tau = tauder(ksi);
%             P(elem(e,7),:) = P(elem(e,7),:) + N(7)*wpg(ig)*t*[tau*J(1,1)
tau*J(1,2)];
%             P(elem(e,3),:) = P(elem(e,3),:) + N(3)*wpg(ig)*t*[tau*J(1,1)
tau*J(1,2)];
%         end
%     end
    if nodelem(4,:) == [0, 2]
        for ig = 1:npg
            ksi = upg(ig);
            eta = 1;
            N = shapefuns([ksi eta],eleT);
            dNke = shapefunsder([ksi eta],eleT);
            J = dNke*nodelem;
            tau = [N(4) N(7) N(3)]*[tauizq(nodelem(4,1)) tauizq(nodelem(7,1))
tauder(nodelem(3,1))]';
            P(elem(e,4),:) = P(elem(e,4),:) + N(4)*wpg(ig)*t*[tau*J(1,1)
tau*J(1,2)];
```

```
            P(elem(e,7),:) = P(elem(e,7),:) + N(7)*wpg(ig)*t*[tau*J(1,1)
tau*J(1,2)];
            P(elem(e,3),:) = P(elem(e,3),:) + N(3)*wpg(ig)*t*[tau*J(1,1)
tau*J(1,2)];
        end
    end
end
P = reshape(P',[],1);


%% Cargas volumétricas
    case 'Volumnen'
P = zeros(doftot/2,2);
f = 50000; % [N/m^3]
[wpg, upg, npg] = gauss([3 3]);
for e=1:nelem
    nodelem = nod(elem(e,:),:);
    for ig= 1:npg
        ksi = upg(ig,1);
        eta = upg(ig,2);
        dNxe = shapefunsder([ksi eta],eleT)
        J = dNxe*nodelem;
        N = [N1(ksi,eta) N2(ksi,eta) N3(ksi,eta) N4(ksi,eta)];
        P(elem(e,:),1) = P(elem(e,:),1) + N'*f1*t*wpg(ig)*det(J);
        P(elem(e,:),2) = P(elem(e,:),2) + N'*f2*t*wpg(ig)*det(J);
    end
end
P = reshape(P',[],1);
end

%% BC
fijo = zeros(doftot/2,2);
fijo([1 3],:) = true;
fijo = reshape(fijo',[],1);
libre = ~fijo;

%% Solver
Dred = Kglobal(libre,libre)\P(libre);
D = zeros(doftot,1);
D(libre) = Dred;
meshplot(elem,nod,'r')
hold on
nodfinal = nod+reshape(D,dofpornodo,nnod)';
meshplot(elem,nodfinal,'b')

%% Tensiones
stress = zeros(nelem,nodporelem,3);
unod = [ -1 -1
          1 -1
          1  1
         -1  1
          0 -1
          1  0
          0  1
         -1  0];
for e = 1:nelem
```

```
    nodelem = nod(elem(e,:),:);
    for in = 1:nodporelem
        % Punto de Gauss
        ksi = unod(in,1);
        eta = unod(in,2);
        % Derivadas de las funciones de forma respecto de ksi, eta
        dNke = shapefunsder([ksi eta],eleT);
        % Derivadas de x,y, respecto de ksi, eta
        J = dNke*nodelem;
        % Derivadas de las funciones de forma respecto de x,y.
        dNxy = J\dNke;
        B = zeros(size(C,2),dofpornodo*nodporelem);
        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,:);
        dofs = reshape(dof(elem(e,:),:)',[],1);
        stress(e,in,:) = C*B*D(dofs);
    end
end

bandplot(elem,nodfinal,stress(:,:,1),[],'k')
```



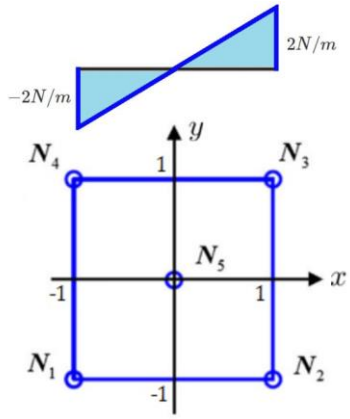
P21 2016

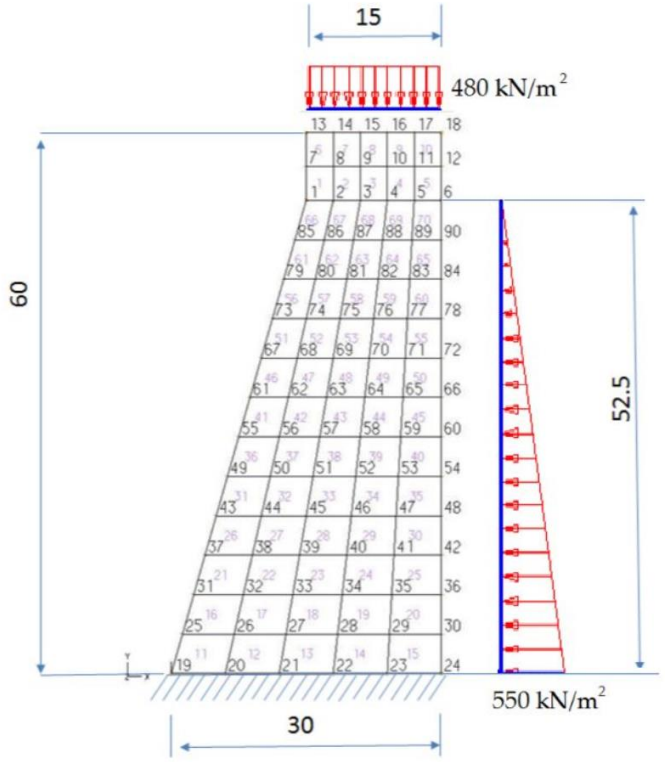
```
%% Parcial 2
clear; close all; clc
%% Ej1.a
syms x y real
X = [1 x y x^2 x*y];
A = [1 -1 -1  1  1
     1  1 -1  1 -1
     1  1  1  1  1
     1 -1  1  1 -1
     1  0  0  0  0];
N = X/A;
dNx = diff(N,x);
dNy = diff(N,y);
dN = [dNx; dNy];
sum(N);  % == 1
sum(dN); % == 0

%% Datos
E = 200E3; %[MPa]
nu = 0.3;
lam = E*nu/(1+nu)/(1-2*nu);
mu = E/2/(1+nu);
t = 1;

%% Nodos y Elementos
nod = [-1 -1
        1 -1
        0  0
       -1  1
        1  1]*1E3;
nnod = size(nod,1);
elem = [1 2 5 4 3];
nelem = size(elem,1);

%% DOF
dofpornodo = 2;
nodporelem = 5;
doftot = dofpornodo*nnod;
dof = reshape((1:doftot)',dofpornodo,nnod)';

%% Constitutiva
Cstrain = E/((1 + nu)*(1 - 2*nu))*[ 1 - nu      nu            0.0
                                    nu       1 - nu           0.0
                                    0.0        0.0     (1 - 2*nu)/2 ];

%% Gauss para regla de 2x2 (numeración de nodos como figura 6.3-3 pág 212)
a   = 1/sqrt(3);
% Ubicaciones puntos de Gauss
upg = [ -a  -a
        -a   a
         a  -a
         a   a ];
```

```
% Número de puntos de Gauss
npg = size(upg,1);
wpg = ones(npg,1);

%% Matriz de rigidez
Kglobal = zeros(doftot);
for e = 1:nelem
    Ke = zeros(dofpornodo*nodporelem);
    nodelem = nod(elem(e,:),:);
    for ipg = 1:npg
        % Punto de Gauss
        x = upg(ipg,1);
        y = upg(ipg,2);
        % Derivadas de las funciones de forma respecto de ksi, eta
        dN = [ x/2 + y/4 - 1/4, x/2 - y/4 + 1/4, x/2 + y/4 + 1/4, x/2 - y/4 -
1/4, -2*x
                      x/4 - 1/4,     - x/4 - 1/4,       x/4 + 1/4,       1/4 -
x/4,    0];
        % Derivadas de x,y, respecto de ksi, eta
        J = dN*nodelem;
        % Derivadas de las funciones de forma respecto de x,y.
        dNxy = J\dN;

        B = zeros(size(Cstrain,2),size(Ke,1));
        B(1,1:2:9) = dNxy(1,:);
        B(2,2:2:10) = dNxy(2,:);
        B(3,1:2:9) = dNxy(2,:);
        B(3,2:2:10) = dNxy(1,:);

        Ke = Ke + B'*Cstrain*B*wpg(ipg)*det(J);
    end
    dofs = dof(elem(e,:),:);
    dofs = reshape(dofs',[],1);
    Kglobal(dofs,dofs) = Kglobal(dofs,dofs) + Ke;
end

%% BC
fijo = zeros(doftot/2,2);
fijo([1 2],:) = [1 1; 0 1];
fijo = logical(reshape(fijo',[],1));
libre = ~fijo;

%% Cargas
P = zeros(doftot/2,2);
Q = 0.002; %[N/mm]
f =@(x) Q*x;
a   = 1/sqrt(3);
% a = sqrt(0.6);
upg = [-a a];
npg = size(upg,2);
wpg = ones(npg,1);
% wpg = [5 8 5]/9;
for e = 1:nelem
    nodelem = nod(elem(e,:),:);
    for ipg = 1:npg
        % Punto de Gauss
```

```
        x = upg(ipg);
        y = 1;
        N = [ (x*y)/4 - y/4 - x/4 + x^2/4, x/4 - y/4 - (x*y)/4 + x^2/4, x/4 +
y/4 + (x*y)/4 + x^2/4, y/4 - x/4 - (x*y)/4 + x^2/4, 1 - x^2];
        dN = [ x/2 + y/4 - 1/4, x/2 - y/4 + 1/4, x/2 + y/4 + 1/4, x/2 - y/4 -
1/4, -2*x
                    x/4 - 1/4,     - x/4 - 1/4,       x/4 + 1/4,       1/4 -
x/4,    0];
        sig = f(x);
        J = dN*nodelem;
        P(elem(e,4),:) = P(elem(e,4),:) + N(4)*wpg(ipg)*t*[0 sig*J(1,1)];
        P(elem(e,3),:) = P(elem(e,3),:) + N(3)*wpg(ipg)*t*[0 sig*J(1,1)];
    end
end
P = reshape(P',[],1);

%% Solver
Dred = Kglobal(libre,libre)\P(libre);
D = zeros(doftot,1);
D(libre) = Dred;
cnodfinal = nod+reshape(D,dofpornodo,nnod)'*10000000;
figure
hold on; grid on; axis equal;
plot(nod(:,1),nod(:,2),'.')
plot(cnodfinal(:,1),cnodfinal(:,2),'.')

%% Gauss para regla de 2x2 (numeración de nodos como figura 6.3-3 pág 212)
a   = 1/sqrt(3);
% Ubicaciones puntos de Gauss
upg = [ -a  -a
        -a   a
         a  -a
         a   a ];
% Número de puntos de Gauss
npg = size(upg,1);
wpg = ones(npg,1);

%% Tensiones
stress = zeros(nelem,nodporelem-1,3);
sigvm = zeros(nelem,nodporelem-1);
for e = 1:nelem
    nodelem = nod(elem(e,1:4),:);
    for in = 1:npg
        x = upg(in,1);
        y = upg(in,2);
        dN = 1/4*[-(1-y)   1-y    1+y  -(1+y)
                  -(1-x) -(1+x)   1+x    1-x ];
        J = dN*nodelem;
        dNxy = J\dN;
        B = zeros(size(Cstrain,2),dofpornodo*npg);
        B(1,1:2:7) = dNxy(1,:);
        B(2,2:2:8) = dNxy(2,:);
        B(3,1:2:7) = dNxy(2,:);
        B(3,2:2:8) = dNxy(1,:);
        dofs = reshape(dof(elem(e,1:4),:)',[],1);
        stress(e,in,:) = Cstrain*B*D(dofs);
```

```
        sigmavm(e,in) = sqrt(stress(e,in,1)^2-
stress(e,in,1)*stress(e,in,2)+stress(e,in,2)^2+3*stress(e,in,3)^2);
    end
end
```





P22 2016

```
%% Ej 2
clear; close all; clc

%% Datos
E = 30e9; %[Pa]
NU = 0.18;
rho = 2000; %[kg/m^3]
g = -9.81; %[m/s^2]
t = 1;
%% PROGRAMA DE ELEMENTOS FINITOS PARA ISOPARAMÉTRICOS Q4

elements = load('ElementosT1.txt');

nodes = load('NodosT1.txt');

nDofNod = 2;                    % Número de grados de libertad por nodo
nNodEle = 4;                    % Número de nodos por elemento
nel = size(elements,1);         % Número de elementos
nNod = size(nodes,1);           % Número de nodos
nDofTot = nDofNod*nNod;         % Número de grados de libertad

bc = false(nNod,nDofNod);       % Matriz de condiciones de borde

R = zeros(nNod,nDofNod);        % Vector de cargas

% Propiedades del material


meshplot(elements,nodes,'b')

%% Matriz Constitutiva (plane strain)

C = E/((1 + NU)*(1 - 2*NU))*[ 1 - NU      NU            0.0
                              NU       1 - NU          0.0
                             0.0        0.0     (1 - 2*NU)/2 ];

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

%% Matriz de rigidez (Para elementos Q4 con regla de Gauss de 2x2)
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 = 1/4*[-(1-eta)   1-eta    1+eta  -(1+eta)
                  -(1-ksi) -(1+ksi)   1+ksi    1-ksi ];
        % 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(1,1:2:7) = dNxy(1,:);
        B(2,2:2:8) = dNxy(2,:);
        B(3,1:2:7) = dNxy(2,:);
        B(3,2:2:8) = dNxy(1,:);

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

%% BC
bc = zeros(nNod,nDofNod);
bc(nodes(:,2)==0,:) = true;

%% Cargas volumétricas
R = zeros(nNod,nDofNod);
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
        N = 1/4*[(1 - ksi)*(1 - eta) (1 + ksi)*(1 - eta) (1 + ksi)*(1 + eta)
(1 - ksi)*(1 + eta)];
        % Derivadas de las funciones de forma respecto de ksi, eta
        dN = 1/4*[-(1-eta)   1-eta    1+eta  -(1+eta)
                  -(1-ksi) -(1+ksi)   1+ksi    1-ksi ];
        % Derivadas de x,y, respecto de ksi, eta
        jac = dN*nodesEle;
        R(elements(iele,:),2) = R(elements(iele,:),2) +
N'*rho*g*t*wpg(ipg)*det(jac);
    end
end

%% Cargas superficiales
a   = 1/sqrt(3);
% Ubicaciones puntos de Gauss
upg = [-a  a];
% Número de puntos de Gauss
npg = size(upg,2);
```

```
wpg = ones(npg,1);
q1 = -480e3; % [N/m^2]
q2 =@(y) -550e3*(1-y/52.5); % [N/m^2]
Q1 = q1*15;
Q2 = integral(q2,0,52.5);
qcheck1 = 0;
qcheck2 = 0;
for iele = 1:nel
    nodesEle = nodes(elements(iele,:),:);
    if nodesEle(4,2)==60
        for ipg = 1:npg
            % Punto de Gauss
            ksi = upg(ipg);
            eta = 1;
            dN = 1/4*[-(1-eta)   1-eta    1+eta  -(1+eta)
                      -(1-ksi) -(1+ksi)   1+ksi    1-ksi ];
            % Derivadas de x,y, respecto de ksi, eta
            jac = dN*nodesEle;
            N = 1/4*[(1 - ksi)*(1 - eta) (1 + ksi)*(1 - eta) (1 + ksi)*(1 +
eta) (1 - ksi)*(1 + eta)];
            R(elements(iele,4),2) = R(elements(iele,4),2) +
N(4)*t*wpg(ipg)*q1*jac(1,1);
            R(elements(iele,3),2) = R(elements(iele,3),2) +
N(3)*t*wpg(ipg)*q1*jac(1,1);
            qcheck1 = qcheck1 + (N(4)+N(3))*t*wpg(ipg)*q1*jac(1,1);
        end
    end
%     if nodesEle(3,1)==30&nodesEle(3,2)<=52.5
%         for ipg = 1:npg
%             % Punto de Gauss
%             ksi = 1;
%             eta = upg(ipg);
%             % Derivadas de las funciones de forma respecto de ksi, eta
%             N = 1/4*[(1 - ksi)*(1 - eta) (1 + ksi)*(1 - eta) (1 + ksi)*(1 +
eta) (1 - ksi)*(1 + eta)];
%             % Derivadas de las funciones de forma respecto de ksi, eta
%             dN = 1/4*[-(1-eta)   1-eta    1+eta  -(1+eta)
%                       -(1-ksi) -(1+ksi)   1+ksi    1-ksi ];
%             % Derivadas de x,y, respecto de ksi, eta
%             delta = (nodesEle(3,2)-nodesEle(2,2))/2;
%             centronodo = (nodesEle(3,2)+nodesEle(2,2))/2;
%             jac = dN*nodesEle;
%             R(elements(iele,2),1) = R(elements(iele,2),1) +
N(2)*t*wpg(ipg)*q2(centronodo+delta*eta)*jac(2,2);
%             R(elements(iele,3),1) = R(elements(iele,3),1) +
N(3)*t*wpg(ipg)*q2(centronodo+delta*eta)*jac(2,2);
%             qcheck2 = qcheck2 +
(N(2)+N(3))*t*wpg(ipg)*q2(centronodo+delta*eta)*jac(2,2);
%         end
%     end
    if nodesEle(3,1)==30&nodesEle(3,2)<=52.5
        for ipg = 1:npg
            % Punto de Gauss
            ksi = 1;
            eta = upg(ipg);
            % Derivadas de las funciones de forma respecto de ksi, eta
```

```
            N = 1/4*[(1 - ksi)*(1 - eta) (1 + ksi)*(1 - eta) (1 + ksi)*(1 +
eta) (1 - ksi)*(1 + eta)];
            % Derivadas de las funciones de forma respecto de ksi, eta
            dN = 1/4*[-(1-eta)   1-eta    1+eta  -(1+eta)
                      -(1-ksi) -(1+ksi)   1+ksi    1-ksi ];
            % Derivadas de x,y, respecto de ksi, eta
            jac = dN*nodesEle;
            N = [N(2) N(3)];
            Q = [q2(nodesEle(2,2)) q2(nodesEle(3,2))]';
            R(elements(iele,2:3),1) = R(elements(iele,2:3),1) +
N'*N*t*wpg(ipg)*Q*jac(2,2);
            qcheck2 = qcheck2 + sum(N'*N*t*wpg(ipg)*Q*jac(2,2));
        end
    end
end
%% Reduccion Matriz
isFixed = reshape(bc',[],1);
isFree = ~isFixed;

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

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

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

meshplot(elements,nodes+reshape(D,nDofNod,nNod)'*1000,'r')
Dr = reshape(D,nDofNod,nNod)';

uMax = max(abs(Dr(:,1)));
nodoUMax = find( abs(Dr(:,1)) == uMax)
uMax = Dr(nodoUMax,1)
vMax = max(abs(Dr(:,2)));
nodoVMax = find( abs(Dr(:,2)) == vMax)
vMax = Dr(nodoVMax,2)
```

Bimetalico

```
%% An application with high stress gradient
close all; clear; clc
%% Datos
% 1.Acero 2.Aluminio
E = [200 70]*1e3; % [MPa]
NU = [.29 .33];
alfa = [12 24]*1e-6; %[1/C]
eleT = 'Q8';
dT = 1000; % [C]
t = 20; % [mm]
%% Nodos y elementos

elements = load('elembi.txt');
elements = elements(:,2:9);

nodes = load('nodosbi.txt'); % [mm]
nodes = nodes(:,2:3);

nDofNod = 2;                    % Número de grados de libertad por nodo
nNodEle = 8;                    % Número de nodos por elemento
nel = size(elements,1);         % Número de elementos
nNod = size(nodes,1);           % Número de nodos
nDofTot = nDofNod*nNod;         % Número de grados de libertad

bc = false(nNod,nDofNod);       % Matriz de condiciones de borde
R = zeros(nNod,nDofNod);        % Vector de cargas

%meshplot(elements,nodes,'b')

%% Gauss para regla de 3x3 (numeración de nodos como figura 6.3-3 pág 212)
a   = sqrt(0.6);
% Ubicaciones puntos de Gauss
upg = [ -a  -a
        -a   0
        -a   a
         0  -a
         0   0
         0   a
         a  -a
         a   0
         a   a ];
% Número de puntos de Gauss
npg = size(upg,1);
wpg = [5/9, 5/9, 5/9, 5/9, 8/9, 5/9, 5/9, 5/9, 5/9];

%% Matriz de rigidez y deformaciones térmicas
K = zeros(nDofTot);
nodeDofs = reshape(1:nDofTot,nDofNod,nNod)';
epsilon0 = dT*alfa;
P = zeros(nDofTot,1);
for iele = 1:nel
    Ke = zeros(nDofNod*nNodEle);
```

```
    Pe = zeros(nDofNod*nNodEle,1);
    nodesEle = nodes(elements(iele,:),:);
    if nodesEle(4,2) == 60
        C = E(1)/(1 - NU(1)^2)*[ 1.0     NU(1)        0.0
                                 NU(1)     1.0        0.0
                                  0.0      0.0    (1 - NU(1))/2 ];
        eps = epsilon0(1);
        epsilon = [eps;eps;0];
    else
        C = E(2)/(1 - NU(2)^2)*[ 1.0     NU(2)        0.0
                                 NU(2)     1.0        0.0
                                  0.0      0.0    (1 - NU(2))/2 ];
        eps = epsilon0(2);
        epsilon = [eps;eps;0];
    end
    for ipg = 1:npg
        ksi = upg(ipg,1);
        eta = upg(ipg,2);
        dN = shapefunsder([ksi eta],eleT);
        jac = dN*nodesEle;
        dNxy = 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,:);

        Ke = Ke + B'*C*B*wpg(ipg)*det(jac)*t;
        Pe = Pe + B'*C*epsilon*wpg(ipg)*det(jac)*t;
    end
    eleDofs = nodeDofs(elements(iele,:),:);
    eleDofs = reshape(eleDofs',[],1);
    K(eleDofs,eleDofs) = K(eleDofs,eleDofs) + Ke;
    P(eleDofs) = P(eleDofs) + Pe;
end


%% BC
bc(1,:) = true;
bc(2,2) = true;

%% Reduccion Matriz
isFixed = reshape(bc',[],1);
isFree = ~isFixed;

Rr = reshape(P,[],1);

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

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

```
%% 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,:),:);
    if nodesEle(4,2) == 60
        C = E(1)/(1 - NU(1)^2)*[ 1.0     NU(1)        0.0
                                 NU(1)     1.0         0.0
                                  0.0      0.0    (1 - NU(1))/2 ];
        eps = epsilon0(1);
        epsilon = [eps;eps;0];
    else
        C = E(2)/(1 - NU(2)^2)*[ 1.0     NU(2)        0.0
                                 NU(2)     1.0         0.0
                                  0.0      0.0    (1 - NU(2))/2 ];
        eps = epsilon0(2);
        epsilon = [eps;eps;0];
    end
    for in = 1:nNodEle
        % Punto de Gauss
        ksi = unod(in,1);
        eta = unod(in,2);
        % Derivadas de las funciones de forma respecto de ksi, eta
        dNke = shapefunsder([ksi eta],eleT);
        % Derivadas de x,y, respecto de ksi, eta
        J = dNke*nodesEle;
        % Derivadas de las funciones de forma respecto de x,y.
        dNxy = J\dNke;
        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,:);
        dofs = reshape(nodeDofs(elements(iele,:),:)',[],1);
        stress(iele,in,:) = C*(B*D(dofs)-epsilon);
    end
end

%% Tensiones en los puntos de superconvergencia
RSextrapolation = unod*sqrt(3);
a   = 1/sqrt(3);
% Ubicaciones puntos de Gauss
upg = [ -a  -a
        -a   a
         a  -a
         a   a ];
% Número de puntos de Gauss
npg = size(upg,1);
wpg = ones(npg,1);
stress2 = zeros(nel,nNodEle,3);
```

```
for iele = 1:nel
    nodesEle = nodes(elements(iele,:),:);
    stressgauss = zeros(npg,3);
    if nodesEle(4,2) == 60
        C = E(1)/(1 - NU(1)^2)*[ 1.0     NU(1)       0.0
                                NU(1)     1.0         0.0
                                 0.0      0.0    (1 - NU(1))/2 ];
        eps = epsilon0(1);
        epsilon = [eps;eps;0];
    else
        C = E(2)/(1 - NU(2)^2)*[ 1.0     NU(2)       0.0
                                NU(2)     1.0         0.0
                                 0.0      0.0    (1 - NU(2))/2 ];
        eps = epsilon0(2);
        epsilon = [eps;eps;0];
    end
    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
        dNke = shapefunsder([ksi eta],eleT);
        % Derivadas de x,y, respecto de ksi, eta
        J = dNke*nodesEle;
        % Derivadas de las funciones de forma respecto de x,y.
        dNxy = J\dNke;
        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,:);
        dofs = reshape(nodeDofs(elements(iele,:),:)',[],1);
        stressgauss(ipg,:) = (C*(B*D(dofs)-epsilon))';
    end
    StressElem = zeros(nNodEle,3);
    for iNod = 1:8
        r = RSextrapolation(iNod,1);
        s = RSextrapolation(iNod,2);
        N = shapefuns([r s],'Q4');
        StressElem(iNod,:) = N*stressgauss;
    end
    stress2(iele,:,:) = StressElem;
end

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

%Gráfico
bandplot(elements,nodePosition,stress(:,:,1),[],'k');
meshplot(elements,nodes,'b')
```

Patch test

```
% Patch Test - Q8 iosparamétrico.
clear
close all
format short g
clc
% UNIDADES N - mm
TestType = 'Sigma_x' ; % 'Sigma_y' ; %  ; %'Tau_xy'
Integracion = 'Subintegrada'; % 'Full'; %
StressCalc  = 'PG'; %'PG' ; %'Nodos'
Lados = 'Rectos'; % 'Curvos'; %

% Discretizacion
switch Lados
    case 'Rectos'
        nod = [ 0.0   0.0
            0.5   0.0
            1.0   0.0
            1.5   0.0
            2.0   0.0
            0.0   0.5
            1.125 0.375
            2.0   0.5
            0.0   1.0
            0.625 0.875
            1.25  0.75
            1.625 0.875
            2.0   1.0
            0.0   1.5
            1.125 1.375
            2.0   1.5
            0.0   2.0
            0.5   2.0
            1.0   2.0
            1.5   2.0
            2.0   2.0 ];          % Coordenadas nodales lados rectos

    case 'Curvos'
        nod = [ 0.0   0.0
            0.5   0.0
            1.0   0.0
            1.5   0.0
            2.0   0.0
            0.0   0.5
            1.0   0.375
            2.0   0.5
            0.0   1.0
            0.625 1.0
            1.25  0.75
            1.625 0.75
            2.0   1.0
            0.0   1.5
            1.0   1.375
            2.0   1.5
```

```
            0.0   2.0
            0.5   2.0
            1.0   2.0
            1.5   2.0
            2.0   2.0 ];          % Coordenadas nodales lados curvos
end

      elem = [ 1  3  11  9   2  7  10   6
               3  5  13  11  4  8  12  7
               9  11  19  17 10 15 18 14
               11  13  21  19  12  16  20  15];  %Matriz de conectividades:
ojo el orden! muy importante!

nDofNod = 2;                    % Número de grados de libertad por nodo
nNodEl = 8;                     % Número de nodos por elemento
nElem = size(elem,1);           % Número de elementos
nNod = size(nod,1);             % Número de nodos
nDofTot = nDofNod*nNod;         % Número de grados de libertad


R = zeros(nNod,nDofNod);        % Vector de cargas
bc = false(nNod,nDofNod);       % Matriz de condiciones de borde
switch TestType
    case 'Sigma_x' %sigma x constante

        bc(1,1:2) = true;
        bc([6 9 14],1) = true;


        R([5 21],1) = 1/6;
        R([8 16],1) = 2/3;
        R(13,1) = 1/3;
        R(17,1) = -1/6;

    case 'Sigma_y' %sigma y constante
      bc(1,1:2) = true;
        bc(2:4,2) = true;

        R(5,2) = -1/6;
        R([17 21],2) = 1/6;
        R([18 20],2) = 2/3;
        R(19,2) = 1/3;

    case 'Tau_xy' %corte xy constante
        bc(1,1:2) = true;
        bc([6 9 14],1) = true;

        R([5 21],2)= 1/6;
        R([8 16],2)= 2/3;
        R(13,2) = 1/3;

        R(17,2)= -1/6;
        R([6 14],2) = -2/3;
        R(9,2) = -1/3;
```

```
        R([17 21],1) = 1/6;
        R([18 20],1) = 2/3;
        R(19,1) = 1/3;

        R(5,1)= -1/6;
        R([2 4],1)= -2/3;
        R(3,1) = -1/3;

end

meshplot(elem,nod,'b')
axis equal

% Propiedades del Material
E = 1;
NU = 0.33;
C = E/(1 - NU^2)*[ 1.0     NU         0.0
                   NU    1.0         0.0
                  0.0    0.0     (1 - NU)/2 ];

%% Matriz de rigidez

switch Integracion
    case 'Subintegrada'
        PG = [-1/sqrt(3) 1
              1/sqrt(3)   1]; %La segunda columna es el peso
    case 'Full'
        PG = [-sqrt(0.6) 5/9
              0          8/9
              sqrt(0.6)  5/9]; %La segunda columna es el peso
end

ordInt = size(PG,1); %Orden de integración Gaussiana

for ipg = 1:ordInt
    csi = PG(ipg,1);
    for jpg = 1:ordInt
        eta = PG(jpg,1);
        varName = genvarname(['dN' num2str(ipg) num2str(jpg)]);
        %genvarname es una función de matlab que te permite guardar un
        %string como un nombre de variable. Así, puedo incluir los
        %contadores del for en el nombre de la variable que quiero guardar.
        %Guardo dN11, dN12, dN21, dN22
        eval([varName ' = dNQ8(csi, eta)']);
    end
end

%Guardé las derivadas de las funciones de forma respecto a csi y eta, que
%es la "parte independiente del Jacobiano", evaluada en los 4 o 9 puntos de
%integración Gaussiana. La llamo independiente porque va a ser igual para
%cualquier elemento, si bien depende de csi y eta, no depende en absoluto
%de x ni de y.

% Buscamos matriz de rigidez y ensamblamos
```

```
K = zeros(nDofTot);
nodeDofs = reshape(1:nDofTot,nDofNod,nNod)';
for iElem = 1:nElem
    %Matriz de rigidez elemental (elementos rectangulares)
    valNod = nod(elem(iElem,:),:);
    ke = zeros(16);
    for ipg = 1:ordInt
        for jpg = 1:ordInt
        dN = eval(['dN' num2str(ipg) num2str(jpg)]); %El "Jacobiano
independiente" evaluada en el PG que nos interesa
        jac = dN*valNod; %Jacobiano es el producto entre la "parte
independiente" y los valores nodales.
        detJ = det(jac);
        dNxy = jac\dN; %Derivadas de las funciones de forma respecto a x e y.

        B = zeros(3,16);
        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,:);

        w = [PG(ipg,2) PG(jpg,2)];
        ke = ke + w(1)*w(2)*B.'*C*B*detJ;
        end
    end
    eleDofs = reshape(nodeDofs(elem(iElem,:),:)',1,[]);
    K(eleDofs,eleDofs) = K(eleDofs,eleDofs) + ke;
end

%% Calculo desplazamientos

%Reducción matriz
isFixed = reshape(bc',[],1);
isFree = ~isFixed;

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

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

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

%Configuración deformada
Dreshape = reshape(D',2,[])';
meshplot(elem,nod + Dreshape,'r') %Dibujo deformación en rojo
axis equal

switch StressCalc
    case 'PG'
        %% Tensión en los puntos de superconvergencia

        PG = [-1/sqrt(3) 1
```

```
            1/sqrt(3)   1]; %buscamos un orden menor al ´full´, para lograr
la superconvergencia

        %en el caso de integracion full, calculamos nuevamente el valor de
los
        %jacobianos en los puntos de gauss, pq nos interesa la integracion
        %gausseana de un orden menor que el full
        %Extrapolamos multiplicando por raiz de 3 desde los puntos de
        %integracion gausseana hacia los nodos.
        if strcmp(Integracion, 'Full')
            for ipg = 1:2
                csi = PG(ipg,1);
                for jpg = 1:2
                    eta = PG(jpg,1);
                    varName = genvarname(['dN' num2str(ipg) num2str(jpg)]);
                    %genvarname es una función de matlab que te permite
guardar un
                    %string como un nombre de variable. Así, puedo incluir
los
                    %contadores del for en el nombre de la variable que
quiero guardar.
                    %Guardo dN11, dN12, dN21, dN22
                    eval([varName ' = dNQ8(csi, eta)']);
                end
            end
        end

        % Buscamos matriz deformacion-desplazamientos
        Stress = zeros(8*nElem,3); %8 filas por cada elementos: las tres
tensiones del nodo 1, las tres tensiones del nodo 2, etc.
        StressAvg = zeros(nNod,3);
        ind = zeros(nNod,1); %Cuenta cuantas veces está compartido un nodo,
para después dividir por eso al promediar
        for iElem = 1:nElem
            valNod = nod(elem(iElem,:),:);
            PGstress = zeros(4,3); % 4 puntos PG, 3 tensiones
            eleDofs = reshape(nodeDofs(elem(iElem,:),:)',1,[]);
            % 4-------3
            % |       |
            % |       |
            % 1-------2 Estos son los PG de superconvergencia donde
calculamos
            % tensiones. 11 es el PG 1, 12 es el PG 4, 21 es el PG 2, 22 es
el PG
            % 3.
            RSextrapolation = [-sqrt(3)  -sqrt(3)
                sqrt(3)  -sqrt(3)
                sqrt(3)   sqrt(3)
                -sqrt(3)  sqrt(3)
                0      -sqrt(3)
                sqrt(3)     0
                0       sqrt(3)
                -sqrt(3)    0];
            for ipg = 1:2
                for jpg = 1:2
```

```
                    dN = eval(['dN' num2str(ipg) num2str(jpg)]); %El
"Jacobiano independiente" evaluada en el PG que nos interesa
                    jac = dN*valNod; %Jacobiano es el producto entre la
"parte independiente" y los valores nodales.
                    dNxy = jac\dN; %Derivadas de las funciones de forma
respecto a x e y.
                    B = zeros(3,16);
                    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,:);

                    if ipg == jpg
                        PGstress(ipg + jpg - 1,:) = C*B*D(eleDofs);
                    else
                        PGstress(ipg * jpg^2,:) = C*B*D(eleDofs);
                    end
                    %Ahora que tenemos las tensiones en los puntos Gaussianos
dentro
                    %del elemento, extrapolamos a los nodos y guardamos esa
data
                    StressElem = zeros(8,3);
                    for iNod = 1:8
                        r = RSextrapolation(iNod,1);
                        s = RSextrapolation(iNod,2);
                        N1 = (1 - r)*(1 - s) / (4);
                        N2 = (1 + r)*(1 - s) / (4);
                        N3 = (1 + r)*(1 + s) / (4);
                        N4 = (1 - r)*(1 + s) / (4);
                        N = [N1 N2 N3 N4];
                        StressElem(iNod,:) = N * PGstress;
                    end
                end
            end
            Stress(8*iElem-7:8*iElem,:) = StressElem;
            % Las tensiones están ordenadas por elemento, queremos tenerlas
en orden
            % por nodo, promediando entre los valores.
            NodosAfectados = elem(iElem,:);
            StressAvg(NodosAfectados,:) = StressAvg(NodosAfectados,:) +
StressElem;
            ind(NodosAfectados,:) = ind(NodosAfectados,:) + 1;
        end
        for iNod = 1:nNod
            StressAvg(iNod,:) = StressAvg(iNod,:)./ind(iNod);
        end
    %% Tensión calculada en los nodos
    case 'Nodos'
        % Para conocer las tensiones en los nodos, debemos evaluar la
        % matriz B en los nodos. Para eso, tenemos que conocer el valor del
        % jacobiano en los nodos.
        CsiEtaNod = [-1 -1
            1  -1
            1  1
            -1  1
            0  -1
            1  0
```

```
            0  1
            -1 0];
        Stress = zeros(8*nElem,3); %8 filas por cada elemento: las tres
tensiones del nodo 1, las tres tensiones del nodo 2, etc.
        StressAvg = zeros(nNod,3);
        ind = zeros(nNod,1); %Cuenta cuantas veces está compartido un nodo,
para después dividir por eso al promediar
        for i = 1:8
            csi = CsiEtaNod(i,1);
            eta = CsiEtaNod(i,2);
            varName = genvarname(['dN' num2str(i)]);
            %genvarname es una función de matlab que te permite guardar un
            %string como un nombre de variable. Así, puedo incluir los
            %contadores del for en el nombre de la variable que quiero
guardar.
            %Guardo dN1, dN2, dN3, dN4, etc
            eval([varName ' = dNQ8(csi, eta)']);
        end
        for iElem = 1:nElem
            valNod = nod(elem(iElem,:),:);
            NodStress = zeros(8,3); % 8 nodos, 3 tensiones
            eleDofs = reshape(nodeDofs(elem(iElem,:),:)',1,[]);
            for i = 1:8
                dN = eval(['dN' num2str(i)]); %El "Jacobiano independiente"
evaluada en el nodo que nos interesa
                jac = dN*valNod; %Jacobiano es el producto entre la "parte
independiente" y los valores nodales.
                dNxy = jac\dN; %Derivadas de las funciones de forma respecto
a x e y.
                B = zeros(3,16);
                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,:);

                NodStress(i,:) = C*B*D(eleDofs);
            end
            %  Ya tenemos las tensiones nodales del elemento. Ahora lo
            %  guardamos en la matriz global de tensiones.
            Stress(8*iElem-7:8*iElem,:) = NodStress;
            % Las tensiones están ordenadas por elemento, queremos tenerlas
en orden
            % por nodo, promediando entre los valores.
            NodosAfectados = elem(iElem,:);
            StressAvg(NodosAfectados,:) = StressAvg(NodosAfectados,:) +
NodStress;
            ind(NodosAfectados,:) = ind(NodosAfectados,:) + 1;
        end
        for iNod = 1:nNod
            StressAvg(iNod,:) = StressAvg(iNod,:)./ind(iNod);
        end

end

disp(StressAvg)
```

```
%% Ploteo de deformada
figure
for iElem = 1:nElem
    eleDofs = reshape(nodeDofs(elem(iElem,:),:)',1,[]); %Ubicacion de los
dofs nodales
    dispElem = D(eleDofs);
    plotDef(nod(elem(iElem,:),:),dispElem,'r','Q8')
    hold on
end
```