ELEMENTOS FINITOS CHEATSHEET

AUTORES

55423

1. Parcial 2

Protip: Necesitas saber que fuerza se tiene que hacer para que se mantenga en posición una arista o punto dado cargas térmicas/fuerza? Apoyalo (fix) y mira las reacciones con la carga termica/fuerza.

1.1. Expresiones útiles

$$\sigma_v = \sqrt{\sigma_x^2+\sigma_y^2+\sigma_z^2-\sigma_x\sigma_y-\sigma_x\sigma_z-\sigma_y\sigma_z+3(\sigma_{xy}^2+\sigma_{xz}^2+\sigma_{yz}^2)} \tag{1}$$

$$\sigma_v = \sqrt{\tfrac{1}{2}[(\sigma_{11}-\sigma_{22})^2+(\sigma_{11}-\sigma_{33})^2+(\sigma_{22}-\sigma_{33})^2]+3(\sigma_{12}^2+\sigma_{13}^2+\sigma_{23}^2)} \tag{2}$$

$$\lambda = \frac{E\,\nu}{(1+\nu)(1-2\nu)} \qquad \mu = G = \frac{E}{2(1+\nu)} \tag{3}$$

$$\begin{bmatrix}\sigma_{xx}\\ \sigma_{yy}\\ \sigma_{xy}\end{bmatrix} = \frac{E}{1-\nu^2}\begin{bmatrix}1 & \nu & 0\\ \nu & 1 & 0\\ 0 & 0 & \frac{1-\nu}{2}\end{bmatrix}\cdot\begin{bmatrix}\varepsilon_{xx}\\ \varepsilon_{yy}\\ 2\varepsilon_{xy}\end{bmatrix} \tag{4}$$

$$\begin{bmatrix}\sigma_{11}\\ \sigma_{22}\\ \sigma_{12}\end{bmatrix} = \frac{E}{(1+\nu)(1-2\nu)}\begin{bmatrix}1-\nu & \nu & 0\\ \nu & 1-\nu & 0\\ 0 & 0 & \frac{1-2\nu}{2}\end{bmatrix}\cdot\begin{bmatrix}\varepsilon_{11}\\ \varepsilon_{22}\\ 2\varepsilon_{12}\end{bmatrix} \tag{5}$$

$$\sigma_n = \frac{\sigma_{xx}+\sigma_{yy}}{2}+\left(\frac{\sigma_{xx}-\sigma_{yy}}{2}\right)\cos 2\theta+\tau_{xy}\sin 2\theta \tag{6}$$

$$\tau_n = -\left(\frac{\sigma_{xx}-\sigma_{yy}}{2}\right)\sin 2\theta+\tau_{xy}\cos 2\theta \tag{7}$$

$$I = \int_{-1}^{1}\phi(\xi)\mathrm{d}\xi \approx \phi(\xi_1)W_1+\phi(\xi_2)W_2\ldots\phi(\xi_n)W_n \tag{8}$$

$$I = \int_{-1}^{1}\int_{-1}^{1}\phi(\xi,\eta)\mathrm{d}\xi\,\mathrm{d}\eta \approx \sum_i\sum_j W_i W_j\phi(\xi,\eta) \tag{9}$$

$n=0$ $\quad 1$

$n=1$ $\quad x \quad y$

$n=2$ $\quad x^2 \quad xy \quad y^2$

$n=3$ $\quad x^3 \quad x^2y \quad xy^2 \quad y^3$

$n=4$ $\quad x^4 \quad x^3y \quad x^2y^2 \quad xy^3 \quad y^4$

$n=5$ $\quad x^5 \quad x^4y \quad x^3y^2 \quad x^2y^3 \quad xy^4 \quad y^5$

$n=6$ $\quad x^6 \quad x^5y \quad x^4y^2 \quad x^3y^3 \quad x^2y^4 \quad xy^5 \quad y^6$

1.2. Como obtener cualquier función de forma

Se define cuantos nodos se va tener por elemento y se los ubica en el espacio (ξ,η) que por simplicidad se trataran como (x,y). Con el triangulo de Pascal para polinomios se elige el grado del polinomio y los términos. Luego se resuelve el sistema de ecuaciones $N_i\cdot X = A$ donde $N_i = [N_1 \quad N_2 \quad \ldots \quad N_n]$ y $X = [1 \quad x \quad y \quad \ldots \quad x^{k-1}y^k \quad x^k y^k]^T$, o algo por el estilo. Se tienen que elegir los grados mas convenientes teniendo en cuenta la simetría y el número de nodos, este ultimo te limita el número de términos posibles por la naturaleza de la interpolación. La matriz A tendrá en su **espacio fila** el mismo polinomio evaluado en la posición del nodo correspondiente a esa fila.

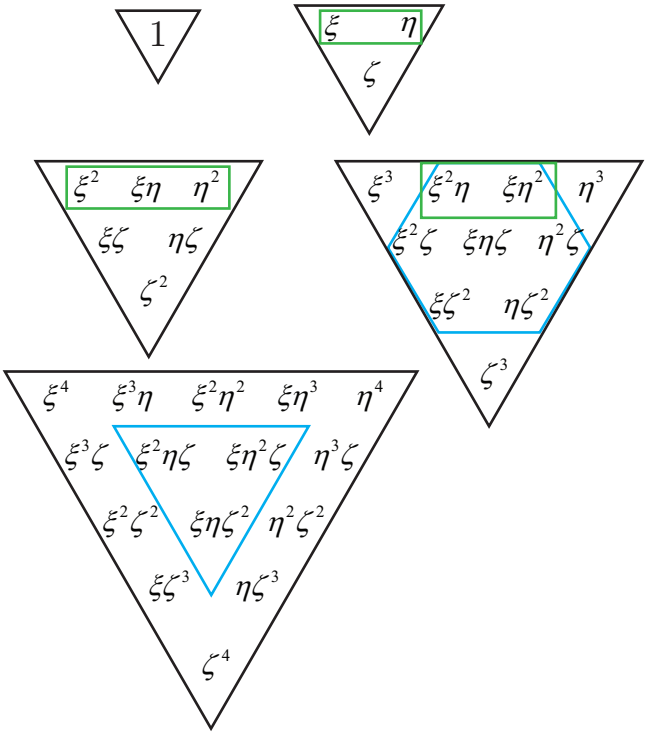

Figura 1: El tetrahedro de Pascal. Los términos de los elementos *serendipidad* están encuadrados.

$$A = \begin{bmatrix} 1 & x_1 & y_1 & \dots & x_1^{k-1} y_1^k & x_1^k y_1^k \\ 1 & x_2 & y_2 & \dots & x_2^{k-1} y_2^k & x_2^k y_2^k \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 1 & x_n & y_n & \dots & x_n^{k-1} y_n^k & x_n^k y_n^k \end{bmatrix}$$

Luego, las funciones de forma N_i se pueden obtener así: $N_i = X^{-1} A$

1.3. Elementos isoparamétricos

- Un elemento que no esta distorsionado (sigue siendo rectangular) tiene J constante
- Cuidado con modo espurio. Ver tabla 6.8-1 pg. 226 el tema de full/reduced integration.
- Todo sobre como cargar tu elemento isoparam. en pg. 228
-

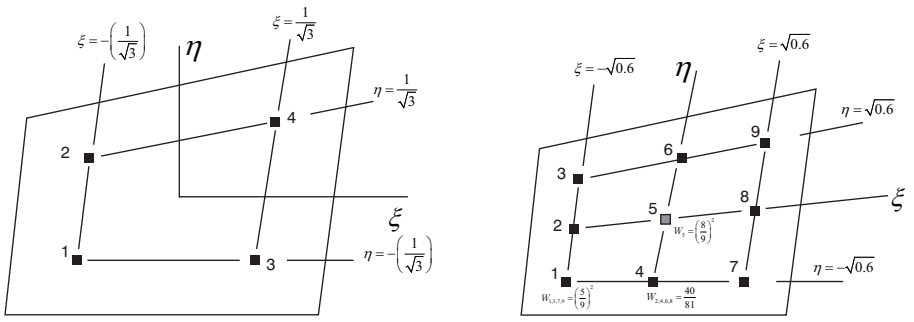

Figura 2: Puntos gauss para ordenes $n = 2$ y $n = 3$. El peso para $n = 2$ es igual en todos los puntos $W_i = 1$

1.4. Ejemplo elemento exótico

Matriz de Rigidez

Imaginemos un elementos Q5 cuadrado de 2×2 con espesor t (igual al Q4 con un nodo en su centro). Si fuéramos a obtener las funciones de formas de dicho elemento quedarían iguales para (x,y) y para (ξ,η) por las dimensiones usadas. La funcionalidad que uno estaría tentado a seleccionar sería $[1\ x\ y\ x^2\ y^2]$, pero está trae problemas inesperados debido a que tiene varias soluciones en la interpolación. Como nuestra prioridad siempre es mantener la simetría la funcionalidad será $[1\ x\ y\ xy\ x^2y^2]$. Tomando el orden de la figura 3.

$$N_i = \left[\ \frac{x^2y^2}{4}+\frac{xy}{4}-\frac{x}{4}-\frac{y}{4},\quad \frac{x^2y^2}{4}-\frac{xy}{4}+\frac{x}{4}-\frac{y}{4},\quad \frac{x^2y^2}{4}+\frac{xy}{4}+\frac{x}{4}+\frac{y}{4},\quad \frac{x^2y^2}{4}-\frac{xy}{4}-\frac{x}{4}+\frac{y}{4},\quad 1-x^2y^2\ \right]$$

Llegado a este punto nos interesa obtener la matriz de rigidez. Si queremos lograr *"full integration"* deberíamos usar Gauss orden $n=3$ según $2n-1\geq O\left([\mathbf{B}]^T[\mathbf{E}][\mathbf{B}]\right)$. El producto $[\mathbf{B}]^T[\mathbf{E}][\mathbf{B}]$ da un polinomio de orden 6 ($[\mathbf{B}]$ tiene el mismo orden que la derivada de $[\mathbf{N}]$). De esta forma nos aseguramos que nuestro resultado va ser exacto para el elemento sin distorsionar.

Para esté ejemplo, no se pide *full integration* entonces no pasa nada si queremos *underintegrate*. Usamos Gauss orden $n=2$.

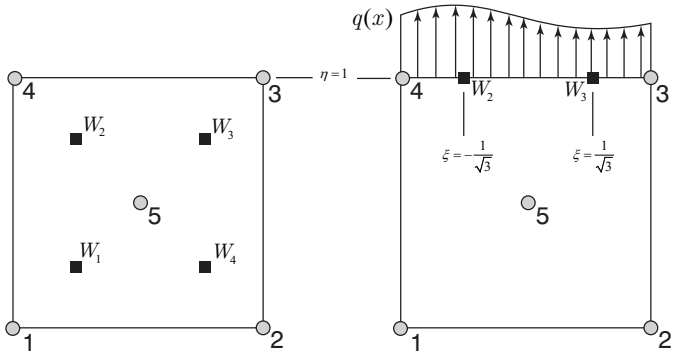

Figura 3: Elemento Q5 rectangular.

La rigidez de un elemento está dada por

$$[\mathbf{k}] = \int [\mathbf{B}]^T[\mathbf{E}][\mathbf{B}]\mathrm{d}V = \iint [\mathbf{B}]^T[\mathbf{E}][\mathbf{B}]t\,\mathrm{d}x\,\mathrm{d}y = \int_{-1}^{1}\int_{-1}^{1} [\mathbf{B}]^T[\mathbf{E}][\mathbf{B}]t\ |\mathbf{J}|\,\mathrm{d}\xi\,\mathrm{d}\eta \tag{10}$$

donde $[\mathbf{B}]$ es la matriz deformación-desplazamiento del elemento, $[\mathbf{E}]$ es la matriz constitutiva, y $|\mathbf{J}|$ es el determinante de la matriz Jacobiana, el cual se le suele decir simplemente el Jacobiano.

Este ultimo se calcula a partir de la derivada de las funciones de forma

Cargas 2-D

La ecuación que rige como se cargan elementos, siendo $\{\mathbf{r}\}$ las cargas nodales, $\{\mathbf{F}\}$ fuerzas volumétricas, $\{\Phi\}$ fuerzas de tracción superficiales, $\{\boldsymbol{\varepsilon_0}\}$ las deformaciones iniciales y $\{\boldsymbol{\sigma_0}\}$ las tensiones iniciales (pg. 228)

$$\{\mathbf{r}\} = \int [\mathbf{N}]^T\{\mathbf{F}\}\mathrm{d}V + \int [\mathbf{N}]^T\{\Phi\}\mathrm{d}S + \int [\mathbf{B}]^T[\mathbf{E}]\{\boldsymbol{\varepsilon_0}\}\mathrm{d}V - \int [\mathbf{B}]\{\boldsymbol{\sigma_0}\}\mathrm{d}V \tag{11}$$

Carga de linea. Si el elemento está cargado sobre la linea 4-3 con una distribuida $q(x)$ (en $[\mathrm{N\,m^{-1}}]$) entonces procedemos de la siguiente manera según el segundo término de (11):

$$r_{xi} = \int_{-1}^{1} N_i(\tau\mathbf{J}_{11}-\sigma\mathbf{J}_{12})t\,\mathrm{d}\xi \tag{12}$$

$$r_{yi} = \int_{-1}^{1} N_i(\sigma\mathbf{J}_{11}+\tau\mathbf{J}_{12})t\,\mathrm{d}\xi \tag{13}$$

donde σ es la solicitación normal a la superficie y τ es la tangencial. Para la fuerza sobre el nodo 4 se tiene

$$r_{y4} = N_4(\xi_2)t\,[\sigma(\xi_2)\,\mathbf{J}_{11} + \tau(\xi_2)\,\mathbf{J}_{12}]\cdot W_2 + N_4(\xi_3)t\,[\sigma(\xi_3)\,\mathbf{J}_{11} + \tau(\xi_3)\,\mathbf{J}_{12}]\cdot W_3$$

Si consideramos que solo hay una *carga distribuida de linea* a tracción/compresión como indica la figura 3, se reduce la ecuación anterior

$$r_{y4} = N_4(\xi_2)\,\mathbf{J}_{11}\,q(\xi_2) + N_4(\xi_3)\,\mathbf{J}_{11}\,q(\xi_3) = N_4\,q\,\mathbf{J}_{11}\Big|_{\xi_2} + N_4\,q\,\mathbf{J}_{11}\Big|_{\xi_3}$$

similarmente $r_{y3} = N_3\,q\,\mathbf{J}_{11}\big|_{\xi_2} + N_3\,q\,\mathbf{J}_{11}\big|_{\xi_3}$ donde la matriz Jacobiana también se evalúa para cada punto de Gauss!

Carga volumétrica.

Tensiones

- Las tensiones en los nodos suele ser de mayor interés que sobre los puntos de gauss (mas comprometidas, permiten estimar error)

2. Parcial 3

2.1. Axisimetría

Resuelvo problema 3-D en el plano. Los resultados son por cada unidad radian. Como sigo teniendo dos grados de libertad tengo las mismas funciones de forma. Cambia mi operador derivada.

$$\begin{Bmatrix} \sigma_r \\ \sigma_\theta \\ \sigma_z \\ \tau_{zr} \end{Bmatrix} = \frac{(1-\nu)E}{(1+\nu)(1-2\nu)} \begin{bmatrix} 1 & f & f & 0 \\ & 1 & f & 0 \\ & & 1 & 0 \\ \text{sim.} & & & g \end{bmatrix} \left(\begin{Bmatrix} \varepsilon_r \\ \varepsilon_\theta \\ \varepsilon_z \\ \gamma_{rz} \end{Bmatrix} - \begin{Bmatrix} \alpha T \\ \alpha T \\ \alpha T \\ 0 \end{Bmatrix} \right)$$

donde

$$f = \frac{\nu}{1-\nu} \qquad \text{y} \qquad g = \frac{1-2\nu}{2(1-\nu)}$$

Una carga puntual P aplicada sobre un elemento axisimétrico no tiene el mismo significado físico que en elementos plane stress/strain.

$$P = 2\pi r q$$

donde q es la carga distribuida en [N/m], r es la distancia al eje de revolución y 2π es el resultado de integrar la fuerza distribuida sobre θ.

$$\{\mathbf{r_e}\} = \int \int_{-\pi}^{\pi} [\mathbf{N}]^T \begin{Bmatrix} \rho r \omega^2 \\ 0 \end{Bmatrix} r \, \mathrm{d}\theta \, \mathrm{d}A$$

2.2. Subestructuras

$$\begin{bmatrix} K_{AA} & K_{AB} \\ K_{BA} & K_{BB} \end{bmatrix} \begin{bmatrix} D_A \\ D_B \end{bmatrix} = \begin{bmatrix} R_A \\ R_B \end{bmatrix}$$

$$D_B = K_{BB}^{-1}(R_B - K_{BA} D_A)$$
$$D_A = K_{AA}^{-1}(R_A - K_{AB} D_B)$$

2.3. Pitfalls

MATLAB

- Antes de aplicar carga distribuida, verificar la orientación del elemento con sus nodos (ξ, η)
-

2.4. Error

Tipos de error:

- Modelado
- Bugs
- Error de usuario
- Error de discretización
- Error de redondeo/truncado
- Error de manipulación
- Error numérico (combinación de los dos anteriores)

Cálculo Tensiones en puntos superconvergencia (Gauss orden 1 para Q4, y Gauss orden 2 aproxima para Q8).

Extrapolo tensiones superconvergentes a los nodos (σ^*).

Energía de deformación.

Se suele requerir que $\eta \leq 0,05$

$$\|U\|^2 = \sum_{i=1}^{m} \int_{v_e} \{\boldsymbol{\varepsilon}\}_i^T [\mathbf{E}] \{\boldsymbol{\varepsilon}\}_i \mathrm{d}V$$

$$\|e\|^2 = \sum_{i=1}^{m} \int_{v_e} (\{\boldsymbol{\varepsilon}^*\}_i - \{\boldsymbol{\varepsilon}\}_i)^T [\mathbf{E}] (\{\boldsymbol{\varepsilon}^*\}_i - \{\boldsymbol{\varepsilon}\}_i) \mathrm{d}V$$

$$\|e\|^2 = \sum_{i=1}^{m} \int_{v_e} (\{\boldsymbol{\sigma}^*\}_i - \{\boldsymbol{\sigma}\}_i)^T [\mathbf{E}]^{-1} (\{\boldsymbol{\sigma}^*\}_i - \{\boldsymbol{\sigma}\}_i) \mathrm{d}V$$

$$\eta = \sqrt{\frac{\|e\|^2}{\|e\|^2 + \|U\|^2}}$$

ADINA

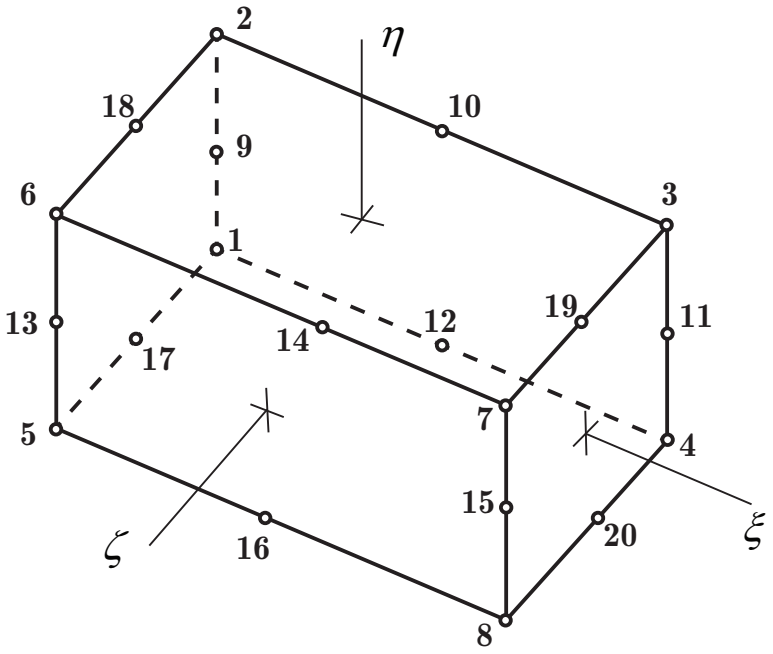

Figura 4: Numeración de nodos H20 en ADINA(Ejes sugeridos)

3. Finitos II

Modal:

$$\{\mathbf{Z}\} = \frac{\{\mathbf{R_\Phi}\}}{\{\Omega\}^2 \sqrt{(1-\chi^2)^2 + (2\varsigma\chi)^2}}$$

donde $\chi = \frac{\omega_{\text{exc}}}{\{\Omega\}}$ y ς se elige por el usuario.

Proporcional:

$$\{\mathbf{D}\} = \frac{\{\mathbf{R}\}}{\{\Omega\}^2 \sqrt{(1-\chi^2)^2 + (2\varsigma\chi)^2}}$$

donde $\chi = \frac{\omega_{\text{exc}}}{\{\Omega\}}$ y $\varsigma = \frac{1}{2}\left(\frac{\alpha}{\omega_{\text{exc}}} + \beta\,\omega_{\text{exc}}\right)$

Si se quiere estudiar un rango de frecuencias de excitación tal que $\omega_{\text{exc}} \in [\omega_1, \omega_2]$ y eligiendo dos valores de damping para ambas frecuencias ς_1 y ς_2 se tiene:

$$\alpha = 2\omega_1\omega_2(\varsigma_1\omega_2 - \varsigma_2\omega_1)/(\omega_2^2 - \omega_1^2)$$
$$\beta = 2(\varsigma_2\omega_2 - \varsigma_1\omega_1)/(\omega_2^2 - \omega_1^2)$$

3.1. Sine Sweep

A medida que la frecuencia de excitación aumenta la *amplitud del sistema disminuye*[1]. Es interesante pensar que si aumentara no tendría sentido buscar las frecuencias naturales porque estas son caracterizadas por un máximo de amplitud. Las curvas del barrido de frecuencia son decrecientes en lejanía de una frecuencia natural porque para una fuerza cíclica $F(t) = F_0 \sin \omega t$ el tiempo que actúa en una dirección es inversamente proporcional a la frecuencia. Por ende la estructura no tiene tiempo para moverse lejos antes de que se invierta la dirección de la fuerza.

[1] Excepto en cercanías de una frecuencia natural

Código MATLAB Finitos II

Código de Placas

Se va tratar con un método para el almacenado eficiente de las funciones de forma usando structures de MATLAB. Esto acorta el tiempo de corrido en un factor mayor a 100 para problemas medianos/grandes. Código en anexo.

Codigo 3.1 (Funciones de Forma Mindlin Q4)

```
clear dN N dNaux X dNxx dNyy dNxy
syms ksi eta real
X = [1 ksi eta ksi.*eta];
Xdx = diff(X,ksi);
Xdy = diff(X,eta);
uNod = [-1 -1;1 -1;1 1;-1 1];
Nnodporelem = length(uNod);
Ndofpornod = 3; %Placas Mindlin
Ndofporelem = Nnodporelem*Ndofpornod;
A = zeros(Nnodporelem,length(X));
for i=1:Nnodporelem
  ksi=uNod(i,1); eta = uNod(i,2);
  A(i,:) = double(subs(X));
end

syms ksi eta real
shapefuns = X*inv(A);
N = shapefuns;
dNx = diff(shapefuns,ksi);
dNy = diff(shapefuns,eta);
B =  sym('noImporta',[5,Ndofporelem]);
for i = 1:Nnodporelem
  B(:,(i*3-2):(i*3)) = [0 dNx(i) 0;0 0 dNy(i);0 dNy(i) dNx(i);
-dNx(i) N(i) 0;-dNy(i) 0 N(i)];
end
dN = [dNx;dNy];
```

Codigo 3.2 (Dofinitions y elemDof)

```
[Nelem, Nnodporelem]= size(elementos);
[Nnod, Ndim] = size(nodos);
Ndofpornod = 3; %Para placa Kirchoff
dof = Nnod*Ndofpornod;
DOF = (1:dof)'; %vector columna
n2d = @(nodo) [nodo*3-2, nodo*3-1, nodo*3]; % Función Node a DOF.
elemDof = zeros(Nelem,Ndofpornod*Nnodporelem);
for e = 1:Nelem
for n = 1:Nnodporelem
  elemDof(e,n2d(n)) = n2d(elementos(e,n));
end
```

```
end
```

Codigo 3.3 (Constitutiva)

```
F = E*t^3/(12*(1 - NU^2)); %Rigidez ante la flexion
G = E/(2+2*NU); % Rigidez a la torsion
Cb = [F NU*F 0;
NU*F F 0
0 0 (1-NU)*F/2];
Cs = 5/6*[G*t 0;0 G*t];
C = blkdiag(Cb,Cs);
```

Codigo 3.4 (Acople matriz rigidez)

```
Kg = sparse(dof,dof);
for e = 1:Nelem
  Kb = zeros(Ndofporelem);
  dofIndex = elemDof(e,:);
  storeTo = elemDof(e,:);
  nodesEle = nodos(elementos(e,:),:);
  Bb = zeros(3,Ndofporelem);
  Bs = zeros(2,Ndofporelem);
  for ipg = 1:npg2
    ksi = upg2(ipg,1); eta = upg2(ipg,2);
    jac = dNs2{ipg}*nodesEle;
    dNxy = jac\dNs2{ipg};   % dNxy = inv(jac)*dN
    for i = 1:Nnodporelem
     Bb(:,(i*3-2):(i*3)) = [0 dNxy(1,i) 0;0 0 dNxy(2,i);0 dNxy(2,i) dNxy(1,i)];
    end
    Kb = Kb + Bb'*Cb*Bb*wpg2(ipg)*det(jac);
  end
  jac = dNs1*nodesEle;
  dNxy = jac\dNs1;   % dNxy = inv(jac)*dN
  for i = 1:Nnodporelem
    Bs(:,(i*3-2):(i*3)) = [-dNxy(1,i) Ns1(i) 0;-dNxy(2,i) 0 Ns1(i)];
  end
  Ks = Bs'*Cs*Bs*wpg1*det(jac);

  Kg(storeTo,storeTo) = Kg(storeTo,storeTo) + Kb+ Ks;
end
```

Codigo 3.5 (Cargas)

```
p0 = -0.05e6; %MPa
R = zeros(dof,1);
for e = 1:Nelem
   storeTo = elemDof(e,1:3:end);
   nodesEle = nodos(elementos(e,:),:);
   for ipg = 1:npg2
```

```
      jac = dNs2{ipg}*nodesEle;
      Q = Ns2{ipg}'*p0*wpg2(ipg)*det(jac);
      R(storeTo)=R(storeTo)+Q;
  end
end
```

Codigo 3.6 (Funciones de Forma Kirchoff. dx e dy son ancho y alto del elemento)

```
syms x y real
X = [1 x y x.^2 x.*y y.^2 x.^3 (x.^2).*y x.*(y.^2) y.^3 x^3*y x.*(y^3)];
Xdx = diff(X,x);
Xdy = diff(X,y);
uDof = [-1 -1;-1 -1;-1 -1;
1 -1; 1 -1; 1  -1
1  1;1  1;1  1
-1 1;-1 1;-1 1];
uDof(:,1)=uDof(:,1)*dx/2;
uDof(:,2)=uDof(:,2)*dy/2;
uNod =[-1 -1;1 -1;1 1;-1 1];
A = zeros(size(uDof,1),length(X));
for i=1:size(uDof,1)
x=uDof(i,1); y = uDof(i,2);
if mod(i,3)==0
A(i,:) = double(subs(Xdy));
elseif mod(i,3)==2
A(i,:) = double(subs(Xdx));
elseif mod(i,3)==1
A(i,:) = double(subs(X));
end
end
syms x y real
N = X*inv(A);
dNxx = diff(N,x,x);
dNyy = diff(N,y,y);
dNxy = diff(N,x,y);
B = [dNxx; dNyy; 2*dNxy];
```

Codigo 3.7 (Acople matriz rigidez y cargas Kirchoff)

```
Kg = sparse(dof,dof);
Ke = double(int(int(B'*C*B,x,[-dx dx]),y,[-dy dy]));
for e = 1:Nelem
storeTo = elemDof(e,:);
Ke = double(int(int(B'*C*B,x,[-dx dx]),y,[-dy dy]));
Kg(storeTo,storeTo) = Kg(storeTo,storeTo) + Ke;
end
p0=-0.05e6; %0.05 MPa
R = zeros(dof,1);
```

```
Nint = int(int(N.',x,[-dx dx]),y,[-dy dy]);
Q = double(p0*Nint);
for e = 1:Nelem
storeTo = elemDof(e,:);
R(storeTo)=R(storeTo)+Q;
end
```

Codigo 3.8 (Recuperación de tensiones)

```
Sb = zeros(Nnodporelem,3,Nelem);
Ss = zeros(Nnodporelem,2,Nelem);
% Funciones de forma en nodos
Nsn = cell(Nnodporelem,1);
dNsn = cell(Nnodporelem,1);
for inod = 1:Nnodporelem
  ksi = uNod(inod,1); eta = uNod(inod,2);
  dNsn{inod} = double(subs(dN));
  Nsn{inod} = double(subs(N));
end

for e = 1:Nelem
  storeTo = elemDof(e,:);
  nodesEle = nodos(elementos(e,:),:);
  Bb = zeros(3,Ndofporelem);
  Bs = zeros(2,Ndofporelem);
  for  inod = 1:Nnodporelem
    ksi = uNod(inod,1); eta = uNod(inod,2);
    Nder=dNsn{inod};
    jac = Nder*nodesEle;
    dNxy = jac\Nder;  % dNxy = inv(jac)*dN
    for i = 1:Nnodporelem % Armo matriz B de bending
        Bb(:,(i*3-2):(i*3)) = [0 dNxy(1,i) 0
        0 0 dNxy(2,i)
        0 dNxy(2,i) dNxy(1,i)];
    end
    for i = 1:Nnodporelem
      Bs(:,(i*3-2):(i*3)) = [-dNxy(1,i) Nsn{inod}(i) 0
     -dNxy(2,i) 0 Nsn{inod}(i)]; % Ver cook 15.3-3
    end
    Sb(inod,:,e) = Cb*(Bb*D(storeTo));
    Ss(inod,:,e) = Cs*(Bs*D(storeTo));
  end
end
```

Codigo 3.9 (Plot Von Mises y desplazamientos)

```
Svm1 = (((Sb(:,1,:)-Sb(:,2,:)).^2+(Sb(:,3,:).^2 + ...
   Ss(:,1,:).^2+Ss(:,2,:).^2) )./2).^(.5);
```

```
bandplot(elementos,nodos,squeeze(Svm1)',[],'k')

Dz = zeros(divisionesx,divisionesy); % Matriz superficie
xv =[];
yv = [];
for n=1:Nnod
   xv = [xv nodos(n,1)];yv = [yv nodos(n,2)];
   Dz(n) = D(n*3-2);
end
Dz=reshape(Dz,[],1)';
scatter3(xv,yv,Dz)
```

3.1.1. Vibraciones

La rutina para acoplar a la matriz de rigidez y la matriz es la misma.

Codigo 3.10 (Matriz de masa viga 2 dof)

```
Me= m/420 * [156    22*Le     54      -13*Le
   22*Le    4*Le^2    13*Le    -3*Le^2
   54      13*Le     156     -22*Le
  -13*Le   -3*Le^2   -22*Le   4*Le^2];
```

Codigo 3.11 (El problema de autovalores)

```
A = Mg(isFree,isFree)\Kg(isFree,isFree);
[Vr, eigVal]=eig(A);
V=zeros(dof);
V(isFree,isFree)=Vr;
dofred = size(Vr,2);
```

Codigo 3.12 (5 formas de obtener {Ω})

```
Phi = zeros(dofred,dofred);
omega = zeros(dofred,1);
omegaray = zeros(dofred,1);
ray = zeros(dofred,1);
for i = 1:dofred
   Dbi=Db(:,i);
   Phi(:,i) = Dbi/sqrt(Dbi'*Mr*Dbi);
   Phii = Phi(:,i);
   omegaray(i) = sqrt((Dbi' * Kg(isFree,isFree) *Dbi)/aux);% Cook (11.4-13)
   ray(i)= sqrt((Phii' * Kg(isFree,isFree) *Phii)/(Phii' *...
        Mg(isFree,isFree) *Phii)); %Cook (11.7-1)b
   omega(i) = sqrt(Phii'* Kg(isFree,isFree)*Phii); % Idem
end
ESP  = Phi' * Kg(isFree,isFree) *Phi;
omegaespectral = sqrt(diag(ESP));
omegaeig = sqrt(diag(eigVal)); %y una quinta para que tengas
```

Codigo 3.13 (Rutina barrido de frecuencia)

```
input_ksi = 0.05:0.05:0.3;
Nmodos = 3;
input_omega = 1:1:10000;
for k = 1:Nksi
   ksi = input_ksi(k);
   for f = 1:Nfrec
      Z=zeros(dofred,1);
      for i=1:dofred
         chi = input_omega(f)/omega(i);
         z(i) = (Rmodalenfrecuencia(i,f)/omega(i)^2)/ ...
              sqrt((1-chi^2)^2+(2*input_ksi(k)*chi)^2);
      end
      Amp(f,k) = abs(sum(z));
   end
   semilogy(input_omega/(2*pi),Amp(:,k))% HZ
   hold on
end
```

3.2. Pandeo

Codigo 3.14 (Matriz rigidez de tensiones para elementos)

```
ksbar =@(P,L) P/L*[1 -1;-1 1];
ksviga =@(P,L) P/30/L*[36 3*L -36 3*L; 3*L 4*L^2 -3*L -L^2
                  -36 -3*L 36 -3*L; 3*L -L^2 -3*L 4*L^2];
```

Codigo 3.15 (Recuperación de fuerzas para barras y λ_{crit})

```
for e = 1:Nelem
   n1 = nodos(elementos(e,1),:);
   n2 = nodos(elementos(e,2),:);
   T = Tbu(atan2d(n2(2)-n1(2),n2(1)-n1(1)));
   Ke = ke(E,A,Le);
   Kerot =T*Ke*T';
   Dlocal=D(elemDof(e,:));
   flocal=Ke*T'*Dlocal;
   Kes = T*kes(flocal(2),Le)*T';
   Ks(storeTo,storeTo) = Ks(storeTo,storeTo) + Kes;
end
Ksr = Ks(isFree,isFree);
A = Ksr\Kr;
[Vr, lambdacrit] = eig(A);
```

Código MATLAB Anexo

Algoritmo rápido para funciones de formas simbólicas

Codigo 3.16 (aplicación a Gauss 2×2)

```
k   = 1/sqrt(3);
upg2 = [ -k  -k
          k  -k
          k   k
         -k   k ];
npg2 = size(upg2,1);
wpg2 = ones(npg2,1);

Ns2 = cell(npg2,1);
dNs2 = cell(npg2,1);
for ipg = 1:npg2
   ksi = upg2(ipg,1); eta = upg2(ipg,2);
   dNs2{ipg} = double(subs(dN));
   Ns2{ipg} = double(subs(N));
end
```

3.2.1. Matriz de transformación para elementos 1D

Codigo 3.17 (Barra. ϕ en grados)

```
T=[cosd(phi) 0;
 sind(phi) 0;
 0 cosd(phi);
 0 sin(phi)];
```

Codigo 3.18 (Viga 3 dof por nodo. ϕ en grados)

```
T=[cosd(phi) sind(phi) 0 0 0 0;
  -sind(phi) cosd(phi) 0 0 0 0;
   0 0 1 0 0 0;
   0 0 0 cosd(phi) sind(phi) 0;
   0 0 0 -sind(phi) cosd(phi) 0;
   0 0 0 0 0 1];
```