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

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

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

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

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

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

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

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

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

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

$n=0$						1						
$n=1$						x		y				
$n=2$					x^2		xy		y^2			
$n=3$				x^3		x^2y		xy^2		y^3		
$n=4$			x^4		x^3y		x^2y^2		xy^3		y^4	
$n=5$		x^5		x^4y		x^3y^2		x^2y^3		xy^4		y^5
$n=6$	x^6		x^5y		x^4y^2		x^3y^3		x^2y^4		xy^5		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 $\mathbf{[N]}\cdot X = A$ donde $\mathbf{[N]} = [N_1 \quad N_2 \quad \ldots \quad N_n]$ 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.

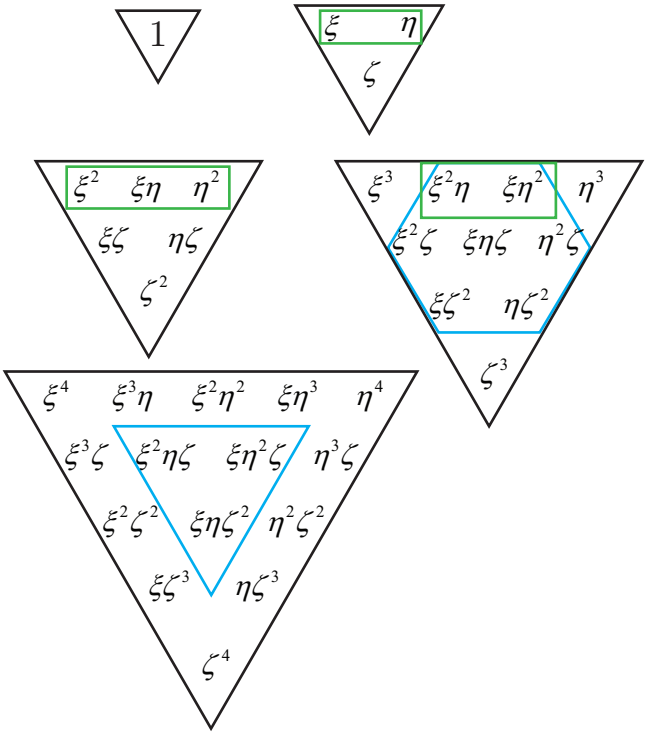

Figura 1: El tetrahedro de Pascal. Los términos de los elementos *serendipidad* están encuadrados.

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

Luego, las funciones de forma **[N]** se pueden obtener así: $\mathbf{[N]} = X^{-1}A$

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
-

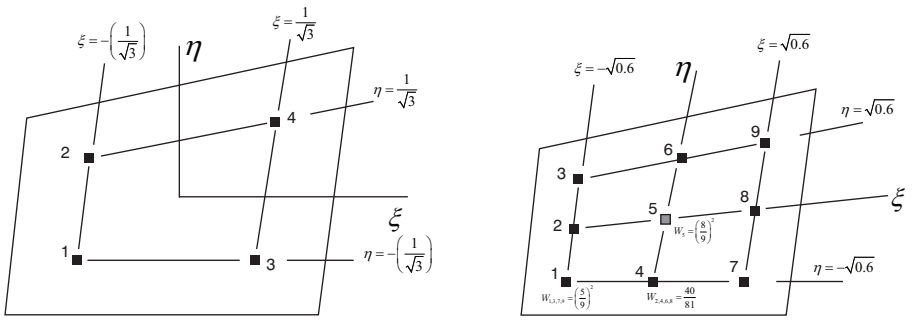

Figura 2: Puntos gauss para ordenes $n = 2$ y $n = 3$. El peso para $n = 2$ es igual en todos los puntos $W_i = 1$

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.

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

Llegado a este punto nos interesa obtener la matriz de rigidez. Si queremos lograr *"full integration"* deberíamos usar Gauss orden $n=3$ según $2n-1\geq O\left(\mathbf{[B]}^T\mathbf{[E][B]}\right)$. **[B]** es el *strain-deformation matrix*. El producto $\mathbf{[B]}^T\mathbf{[E][B]}$ da un polinomio de orden 6 (**[B]** tiene el mismo orden que la derivada de **[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 $n=2$.

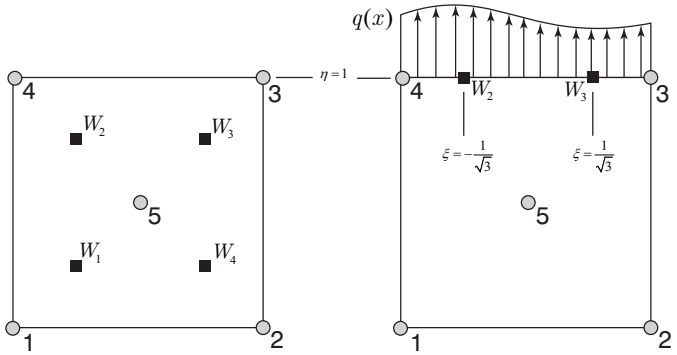

Figura 3: Elemento Q5 rectangular.

La rigidez de un elemento está dada por

$$\mathbf{[k]}=\int \mathbf{[B]}^T\mathbf{[E][B]}\mathrm{d}V \tag{10}$$

para un elemento plano la ecuación anterior es

$$\mathbf{[k]}_{2\mathrm{D}}=\iint \mathbf{[B]}^T\mathbf{[E][B]}t\,\mathrm{d}x\mathrm{d}y=\int_{-1}^{1}\int_{-1}^{1}\mathbf{[B]}^T\mathbf{[E][B]}t\mid\mathbf{J}\mid\mathrm{d}\xi\mathrm{d}\eta$$

donde **[B]** es la matriz deformación-desplazamiento del elemento, **[E]** es la matriz constitutiva, y $\mid\mathbf{J}\mid$ 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 **{r}** las cargas nodales, **{F}** fuerzas volumétricas, **{Φ}** fuerzas de tracción superficiales, $\boldsymbol{\{\varepsilon_0\}}$ las deformaciones iniciales y $\boldsymbol{\{\sigma_0\}}$ las tensiones iniciales (pg. 228)

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

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

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

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)

Código MATLAB

Dofinitions

Codigo 1.1

```
%% Definiciones    (N=Número)
Ndofpornod = 2;                     % grados de libertad por nodo
Nelem = size(elementos,1);          % elementos
Nnod = size(nodos,1);            % nodos
Nnodporelem = size(elementos,2);     % nodos por elemento
doftot = Ndofpornodo*Nnod;          % grados de libertad
DOF=reshape(1:doftot,2,[])';     % Matriz de grados de libertad
Ndims = size(nodos,2);           % dimensiones del problema
```

Form Functions

Codigo 1.2

```
%% Obtención de funciones de forma
syms x y real
X = [1 x y x^2 x*y y^2]; %LST
A = [1 0 0 0 0 0
     1 1 0 1 0 0
     1 0 1 0 0 1
     1 .5 0 .25 0 0
     1 .5 .5 .25 .25 .25
     1 0 .5 0 0 .25];
N = X/A;
dN = [diff(N,x); diff(N,y)];
sum(N);  % == 1
sum(dN,2); % == 0
```

Codigo 1.3 (Q8)

```
clear dN N dNaux X dNxx dNyy dNxy Bs% Ns2 Ns3 dNs3 dNs2
syms ksi eta real
X = [1 eta ksi ksi.^2 ksi.*eta eta.^2 (ksi.^2).*eta ksi.*(eta.^2)];%
Cambia de Q4 a Q8
% X = [1 ksi eta ksi.*eta];
% w1 t1 t2
Xdx = diff(X,ksi);
Xdy = diff(X,eta);
uNod = [-1 -1
1 -1
1 1
-1 1
0 -1
1 0
0 1
-1 0];
```

```
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);
dN = [dNx;dNy];
```

Preparación para rutina elementos

Codigo 1.4

```
Cstrain = ...
Cstress = ...
Cterm  =  ...
Kglobal = zeros(doftot);
P = zeros(doftot,1);

fixity = false(doftot,1);
fixity([m n ... z]) = true;
free = ~fixity;

W=[5/9   8/9   5/9]; %El producto W'*W visualiza en 2D los pesos
wpg=reshape(W'*W,[],1);% gauss n=3
```

Matriz rigidez con rutina elementos (tabla 6.3-2)

Codigo 1.5

```
%% Matriz de rigidez
Kt = zeros(doftotT);
K=zeros(doftot);
jmin = 1E10;
Areas=zeros(Nelem,1);
for iele = 1:Nelem
    A = 0;
    index=elementos(iele,:);
    Ket = zeros(NdofpornodT*Nnodporelem);
    Ke = zeros(Ndofpornodo*Nnodporelem);
    meindof = reshape(DOF(index,:)',1,[]);
    meindofT=DOFT(index)';%ahora lo hago para carga termica
    nodesEle = nodos(index,:);
    for ipg = 1:npg
        ksi = upg(ipg,1);
        eta = upg(ipg,2);
        N = shapefuns([ksi eta],eleType);
        dN = shapefunsder([ksi eta],eleType);
        jac = dN*nodesEle;
        dNxy = jac\dN;       % dNxy = inv(jac)*dN
        B = zeros(size(C,2),Ndofpornodo*Nnodporelem);
        B(1,1:2:Ndofpornodo*Nnodporelem-1) = dNxy(1,:);
        B(2,2:2:Ndofpornodo*Nnodporelem) = dNxy(2,:);
        B(3,1:2:Ndofpornodo*Nnodporelem-1) = dNxy(2,:);
        B(3,2:2:Ndofpornodo*Nnodporelem) = dNxy(1,:);
        Bt=dNxy;%el B para cargas termicas
        Djac = det(jac);

        Ke = Ke + B'*C*B*wpg(ipg)*Djac;
        Ket = Ket + Bt'*Ct*Bt*wpg(ipg)*Djac;
        A = A + wpg(ipg)*Djac;
        if Djac < jmin
            jmin = Djac;
        end
    end
    Areas(iele)=A;
    Kt(meindofT,meindofT) = Kt(meindofT,meindofT) + Ket;
    K(meindof,meindof) = K(meindof,meindof) + Ke;
end
```

Carga de Linea Q4

Codigo 1.6

```
R = zeros(doftot,1);% Vector de cargas
DOF=reshape(1:doftot,2,[])';
surfnod=[4 3];
wpg=[1 1];
a=-sqrt(3)^-1;
upg=[-a 1;a 1];
npg=2;
sig=-.02;
for e=1:Nelem
    index=elementos(e,:);
    meindof=DOF(index,:);
    elenod=nodos(index,:);
    A=elenod(:,2)==60; % y=60 => Cargo
    if  isempty(find(A,1))
        continue
    end
    for ipg=1:npg
        ksi = upg(ipg,1);
        eta = upg(ipg,2);
        N = shapefuns([ksi eta],eleType);
        dN = shapefunsder([ksi eta],eleType);
        jac = dN*elenod;
        W=wpg(ipg);
        for snod=surfnod
           R(meindof(snod,1))=R(meindof(snod,1))+t*N(snod)*(-jac(1,2))*sig*W;
           R(meindof(snod,2))=R(meindof(snod,2))+t*N(snod)*jac(1,1)*sig*W;
           qcheck=qcheck+t*N(snod)*jac(1,1)*sig*W; %Sería de las dos
        end
    end
end
R=reshape(R,2,[])';
```

1.4.1. Cálculo de tensiones iterando sobre elementos pg. 230

Codigo 1.7

```
stress = zeros(Nnodporelem,3,Nelem);

meindof=reshape(DOF(index,:)',[],1);
stress(inode,:,iele) = C*B*D(meindof);
%...

S=2; % para sigma_yy
plotvar=squeeze(stress(:,S,:))';
bandplot(elementos,nodePosition,plotvar,[],'k');
```

1.4.2. Triangulos pg. 260 Gauss Triangulos (267)

Codigo 1.8

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

Resolución de problema de Calor con tensiones iniciales

Codigo 1.9

```
Co=[1,6,5,10,11,12,13,14,15];
X=[2,3,4,7,8,9]; %numel(Co)+numel(X)==doftotT
Kxx=Kt(X,X);Kxc=Kt(X,Co);
Tc=zeros(length(Co),1);
Tc([1,2,5])=100;%los nodos con respecto a Co
Rt=Kxc*Tc;
Tx=Kxx\-Rt;
T=zeros(Nnod,1);
T(X)=Tx;
T(Co)=Tc;
for i=1:Nelem
    temp(i,:)=T(elementos(i,:)); %temp es graficable
end
%% Encuentro FUERZAS GENERADAS POR GRADIENTE CALOR(rutina elementos[RE])
index=elementos(iele,:)
R(index,:) = R(index,:) + ...
reshape(-B'*C*(N*T(index)*[1 1 0]')*alfa*wpg(ipg)*Djac,2,Nnodporelem)';
%% Busco D con estás fuerzas con K\R->Tensiones térmicas (RE sobre NODOS)
```

(cont.) Tensiones y flujo de calor. Rutina Elementos para tensiones sobre nodos

Codigo 1.10

```
uNod = [-1 -1;1 -1;1  1;-1  1];%Q4
q=zeros(Nnodporelem*Ndofpornodo,1);
stress = zeros(Nnodporelem,3,Nelem);
for iele = 1:Nelem
    index = elementos(iele,:);
    nodesEle = nodos(index,:);
    indexT=index;
    meindof = reshape(DOF(index,:)',1,[]);
    TnodosEle=T(index); %Del perfil temp.
    for inod = 1:Nnodporelem
        ksi,eta = uNod(inod,1),uNod(inod,2);
```

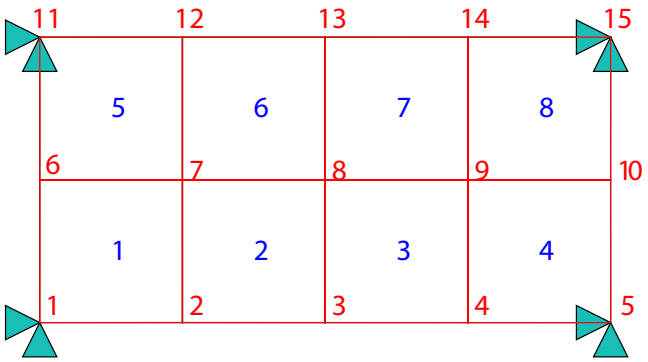

Figura 4: Problema depicto en el código 1.9

```
        dN  = shapefunsder([ksi eta],eleType);
%Obtengo jacobiano, B
        stress(inod,:,iele) = C*(B*D(meindof)-(alfa*TnodosEle(inod)*[1 1 0]'));
        q([2*indexT(inod)-1,2*indexT(inod)])=-k*Bt*T(indexT);
    end
end
qx=q(1:2:end);
qy=q(2:2:end);
quiver(nodos(:,1),nodos(:,2),qx,qy) %Plot Quiver
```

$$dN_i = \begin{pmatrix} 4x+4y-3, & 4x-1, & 0, & 4-4y-8x, & 4y, & -4y \\ 4x+4y-3, & 0, & 4y-1, & -4x, & 4x, & 4-8y-4x \end{pmatrix}_{LST}$$

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. Constraints

Rigid Links

2.3. Subestructuras

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

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

2.4. Pitfalls

MATLAB

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

2.5. 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 $\eta \leq 0,05$

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

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

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

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

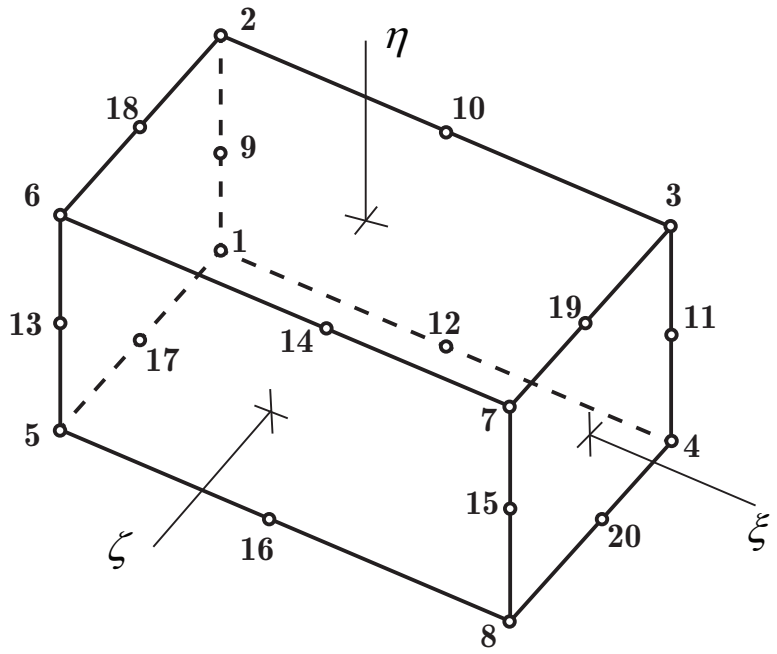

Figura 5: Numeración de nodos H20 en ADINA(Ejes sugeridos)

ADINA

2.6. Formulacion elemento Hexahedro de 8 nodos

El modelado con elementos isoparametricos hexaedros de 8 nodos es un buen punto de partida para comenzar a manejar los elementos finitos en 3 dimensiones. Un codigo hecho para obtener **[K]** con elementos H8 se adapta con facilidad para los H20.

Cada nodo tendrá 3 grados de libertad, dándonos 24 dof por elemento. El elemento H20 de la figura 5 esta planteado de tal forma que los primeros nodos del 1 al 8 son los nodos del H8 que se va formular a continuación.

La funcionalidad a usar es la siguiente

$$X_{\mathrm{H8}} = \left[1, \xi, \eta, \zeta, \xi\eta, \xi\eta, \eta\zeta, \xi\eta\zeta\right]$$

y la matriz constitutiva para el espacio 3D con 3 dof por nodo se escribe:

$$\mathbf{[E]} = \begin{bmatrix} 2G+\lambda & \lambda & \lambda & 0 & 0 & 0 \\ \lambda & 2G+\lambda & \lambda & 0 & 0 & 0 \\ \lambda & \lambda & 2G+\lambda & 0 & 0 & 0 \\ 0 & 0 & 0 & G & 0 & 0 \\ 0 & 0 & 0 & 0 & G & 0 \\ 0 & 0 & 0 & 0 & 0 & G \end{bmatrix} \qquad \mathbf{[E]}^{-1} = \frac{1}{E}\begin{bmatrix} 1 & -\nu & -\nu & 0 & 0 & 0 \\ -\nu & 1 & -\nu & 0 & 0 & 0 \\ -\nu & -\nu & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & f & 0 & 0 \\ 0 & 0 & 0 & 0 & f & 0 \\ 0 & 0 & 0 & 0 & 0 & f \end{bmatrix} \tag{14}$$

donde $\lambda = \frac{E\,\nu}{(1+\nu)(1-2\,\nu)}$ y $G = \frac{E}{2(1+\nu)}$ y $f = 2+2\,\nu$.

2.6.1. Formulación elementos

El jacobiano tiene la forma

$$\mathbf{J} = \begin{bmatrix} \frac{\partial x}{\partial \xi} & \frac{\partial y}{\partial \xi} & \frac{\partial z}{\partial \xi} \\ \frac{\partial x}{\partial \eta} & \frac{\partial y}{\partial \eta} & \frac{\partial z}{\partial \eta} \\ \frac{\partial x}{\partial \zeta} & \frac{\partial y}{\partial \zeta} & \frac{\partial z}{\partial \zeta} \end{bmatrix}$$

pudiendo ser calculado de la siguiente forma

$$\mathbf{J} = \begin{bmatrix} \frac{\partial}{\partial \xi}[\mathbf{N}] \\ \frac{\partial}{\partial \eta}[\mathbf{N}] \\ \frac{\partial}{\partial \zeta}[\mathbf{N}] \end{bmatrix} \cdot \begin{bmatrix} x_1 & y_1 & z_1 \\ x_2 & y_2 & z_2 \\ x_3 & y_3 & z_3 \\ x_4 & y_4 & z_4 \\ x_5 & y_5 & z_5 \\ x_6 & y_6 & z_6 \\ x_7 & y_7 & z_7 \\ x_8 & y_8 & z_8 \end{bmatrix} \tag{15}$$

donde la primer matriz termina siendo 3×8 para un elemento H8. La segunda matriz son las posiciones *globales* de los nodos del elemento. El jacobiano se puede entonces utilizar para calcular

$$[\partial \mathbf{N}] = \begin{bmatrix} \frac{\partial}{\partial x}[\mathbf{N}] \\ \frac{\partial}{\partial y}[\mathbf{N}] \\ \frac{\partial}{\partial z}[\mathbf{N}] \end{bmatrix} = \mathbf{J}^{-1} \begin{bmatrix} \frac{\partial}{\partial \xi}[\mathbf{N}] \\ \frac{\partial}{\partial \eta}[\mathbf{N}] \\ \frac{\partial}{\partial \zeta}[\mathbf{N}] \end{bmatrix} \tag{16}$$

con lo obtenido se puede calcular la matriz $[\mathbf{B}]$.

La matriz *strain-deformation* queda

$$[\mathbf{B}] = \begin{bmatrix} B_1 & B_2 & B_3 & B_4 & B_5 & B_6 & B_7 & B_8 \end{bmatrix}$$

donde

$$B_i = \begin{bmatrix} \partial N_i/\partial x & 0 & 0 \\ 0 & \partial N_i/\partial y & 0 \\ 0 & 0 & \partial N_i/\partial z \\ 0 & \partial N_i/\partial z & \partial N_i/\partial y \\ \partial N_i/\partial z & 0 & \partial N_i/\partial x \\ \partial N_i/\partial y & \partial N_i/\partial x & 0 \end{bmatrix} \tag{17}$$

Finalmente un calcula la rigidez del elemento usando (10).

3. Finitos II

$$[\mathbf{M}]\{\ddot{\mathbf{D}}\} + [\mathbf{C}]\{\dot{\mathbf{D}}\} + [\mathbf{K}]\{\mathbf{D}\} = \{\mathbf{R}\} \tag{18}$$

Amortiguamiento $[\mathbf{C}] = \alpha[\mathbf{M}] + \beta[\mathbf{K}]$ cede una matriz no diagonal. Se complica la resolución. Existen dos otros modelos que tratan con una matriz $[\mathbf{C_\Phi}]$ diagonal donde las ecuaciones se desacoplan.

Amortiguamiento Modal: Se elige un ς para cada modo

$$[\mathbf{C_\Phi}] = \begin{bmatrix} 2\omega_n\varsigma_n & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & 2\omega_1\varsigma_1 \end{bmatrix} \tag{19}$$

Amortiguamiento proporcional. Se basa el análisis

$$[\mathbf{C_\Phi}] = [\mathbf{\Phi}]^T(\alpha[\mathbf{M}] + \beta[\mathbf{K}])[\mathbf{\Phi}] = \alpha\delta[\mathbf{I}] + \beta[\mathbf{\Omega^2}] \tag{20}$$

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:

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

Una vez obtenida $[\mathbf{C_\Phi}]$ se pueden obtener los desplazamientos modales $\{\mathbf{Z}\}$. Tome en cuenta que debido a la diagonalidad de $[\mathbf{\Omega^2}]$ y $\{\mathbf{R_\Phi}\}$ se desacoplan las ecuaciones de 18 y por ende se pasa a tratar dichas matrices diagonales como vectores columnas. Una vez desacopladas se tiene

$$\{\ddot{\mathbf{Z}}\} + 2\{\Omega\}\{\mathbf{C_\Phi}\}\{\dot{\mathbf{Z}}\} + \{\Omega\}^2\{\mathbf{Z}\} = \{\mathbf{R_\Phi}\}$$

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

donde $\chi = \frac{\omega_{\text{exc}}}{\{\Omega\}}$.

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.

4. Apuntes Segundo Parcial

Es cuasiestatico cuando la deformacion avanza

Cargas armonicas: Analizo respuesta de la estructura ante cada modo.

Aumentar damping disminuye la amplitud del motor, pero a la vez lo que le otorga damping se convierte rigido y toma reacciones.

Para simular transitorio podemos usar la descomposicion modal con metodos numericos usando runge kutta... porque ahora tengo una particular... la funcion

Ahora pueden variar de cualquier manera:

$$D = D(x,y,z,t) = N(x,y,z)D(t)$$

Desarrollamos D en serie Taylor.

$f(x_0+\Delta x) = f(x_0) + f'(x_0)\Delta x + \frac{1}{2}...$

Con las ecuaciones de taylor despejadas para $n+1$ y $n-1$ puedo adivinar el futuro con el presente (y pasado).

$$\{D\}n+1 = \{D\}n-1+2\Delta t\{\dot{D}\}_n$$

Sin conocer $\dot{D}_n$ esta dificil... uso taylor otra vez!

$$\left[\frac{M}{\Delta t^2}+\frac{C}{2\Delta t}\right]\{D\}n+1 = \left\{R^{ext}\right\}n-[K]\{D\}n+\frac{2}{\Delta t^2}[M]\}Dn-\left[\frac{M}{\Delta t^2}-\frac{C}{2\Delta t}\right]\{D\}_{n-1}$$

donde $\left\{R^{ext}\right\}$ puede ser una función en el tiempo también! Una funcion heaviside excita TODOS los modos.

Optimización: No hay mucho que se pueda hacer. Es super complicated

Se presentan al lector dos formas de resolver la siguiente ecuacion:

$$\begin{cases} [M]\dot{V}\}+[C]V\}+[K]\{D\} = \left\{R^{\text{ert}}\right\} \\ \{\dot{D}\} = \{V\} \end{cases}$$

Integración directa

$$\begin{cases} \{\mathbf{V}\}^{n+1} & = \{\mathbf{V}\}^n + \frac{\Delta t}{2}[\mathbf{M}]^{-1}\left(\{\mathbf{R}^{\text{ext}}\}-[\mathbf{C}]\{\mathbf{V}\}-[\mathbf{K}]\{\mathbf{D}\}\right) \\ \{\mathbf{D}\}^{n+1} & = \{\mathbf{D}\}^n + \frac{\Delta t}{2}\{\mathbf{V}\} \end{cases}$$

RUNGE KUTTA

Tengo que resolver las dos ecuaciones simultáneamente

$$\begin{cases} \{\mathbf{V}\}^{n+1} = \{\mathbf{V}\}^n + \frac{\Delta t}{6}\left(\{k_1^{\dot{V}}\}+2\{k_2^{\dot{V}}\}+2\{k_3^{\dot{V}}\}+\{k_4^{\dot{V}}\}\right) \\ \{\mathbf{D}\}^{n+1} = \{\mathbf{D}\}^n + \frac{\Delta t}{6}\left(\{k_1^{\dot{D}}\}+2\{k_2^{\dot{D}}\}+2\{k_3^{\dot{D}}\}+\{k_4^{\dot{D}}\}\right) \end{cases}$$

[1] Excepto en cercanías de una frecuencia natural

donde

$$\begin{cases} \{k_1^{\dot{V}}\} &= [\mathbf{M}]^{-1}\left(\{\mathbf{R}^{\text{ext}}\}^n - [\mathbf{C}]\{\mathbf{V}\}^n - [\mathbf{K}]\{\mathbf{D}\}^n\right) \\ \{k_1^{\dot{D}}\} &= \{\mathbf{V}\}^n \\ \hline \{k_2^{\dot{V}}\} &= [\mathbf{M}]^{-1}\left(\{\mathbf{R}^{\text{ext}}\}^{n+\frac{1}{2}} - [\mathbf{C}]\left(\{\mathbf{V}\}^n + \frac{\Delta t}{2}\{k_1^{\dot{V}}\}\right) - [\mathbf{K}]\left(\{\mathbf{D}\}^n + \frac{\Delta t}{2}\{k_1^{\dot{D}}\}\right)\right) \\ \{k_2^{\dot{D}}\} &= \{\mathbf{V}\}^n + \frac{\Delta t}{2}\{k_1^{\dot{V}}\} \\ \hline \{k_3^{\dot{V}}\} &= [\mathbf{M}]^{-1}\left(\{\mathbf{R}^{\text{ext}}\}^{n+\frac{1}{2}} - [\mathbf{C}]\left(\{\mathbf{V}\}^n + \frac{\Delta t}{2}\{k_2^{\dot{V}}\}\right) - [\mathbf{K}]\left(\{\mathbf{D}\}^n + \frac{\Delta t}{2}\{k_2^{\dot{D}}\}\right)\right) \\ \{k_3^{\dot{D}}\} &= \{\mathbf{V}\}^n + \frac{\Delta t}{2}\{k_2^{\dot{V}}\} \\ \hline \{k_4^{\dot{V}}\} &= [\mathbf{M}]^{-1}\left(\{\mathbf{R}^{\text{ext}}\}^{n+1} - [\mathbf{C}]\left(\{\mathbf{V}\}^n + \Delta t\{k_3^{\dot{V}}\}\right) - [\mathbf{K}]\left(\{\mathbf{D}\}^n + \Delta t\{k_3^{\dot{D}}\}\right)\right) \\ \{k_4^{\dot{D}}\} &= \{\mathbf{V}\}^n + \Delta t\{k_3^{\dot{V}}\} \end{cases}$$

Espectro de respuesta

Gráfico $\ddot{u}$–ω muestra las cargas que podría estar sometido tu sistema.

A frecuencias mas altas la diferencia entre las Z es mayor:

$$S_i = \frac{Z_i^{\text{mâx}}}{Z_i^{\text{st}}}$$

Ataco el sistema con un “Paquete“ de cargas y cada modo responde de su propia forma. Este metodo Sebas le dice Random. Se suele usar para sistemas tipo placas en satelites, donde tenes todas las plaquetas vibrando y no conoces que esta pasando ahi. Para estructuras no tanto porque son más estables y se conoce mejor que puede llegar a ser el modo de vibración porque los modos de vibracion estan bien separados->cosa que no es verdad en satelites.

$$\mathrm{D(t)j} = [\Phi]\{\mathrm{Z(t)}\} = \sum \mathrm{j}\Phi_{\mathrm{ji}} Z_{\mathrm{i}}(\mathrm{t}) = \sum_{\mathrm{j}} \Delta_{\mathrm{ji}}(\mathrm{t})$$

Voy a tener un desplazamiento máximo cuando los modos esten todos en su máximo... pero eso sucede cuando están en fase, cosa que nunca sucede porque siempre hay algun desfasaje. Hay criterios para determinar el desplazamiento máximo, una incluye sumando la primer autoforma que es la más grande y luego sumando cuadrados mínimos de las otras. Con esto te cubris de la tensión que podría llegar a ocurrir ().

5. No linealidad

Metodo variacional: Lo que buscamos es un estado de energia minima para el sistema, y como para obtener la energia tenemos que integrar se requiere integrar todo el sistema. Buscamos un punto donde el funcional sea estable. Decimos que toda la energia interna es igual al trabajo externo.

Formula resuelta en Resistencia de materiales

$$EI\frac{d^4 v}{dx^4} - q = 0$$

Formula resuelta en elementos finitos

$$\Phi = \int_0^L \frac{EI}{2}\left(\frac{d^2 v}{dx^2}\right)^2 - q\,v\,dx \rightarrow \delta\Phi = 0$$

Para resolver variacionalmente necesitamos un potencial. Necesitamos saber como se almacena la energia. Con plasticidad pierdo energia, se entrega a la pieza de la forma de deformacion permanente.

Segun el principio de los trabajos virtuales. Lo que en verdad estamos escribiendo es un equilibrio de fuerzas, porque me estoy preguntando el estado final y si esta en equilibrio haciendo el calculo "si me muevo un cachito cuanto trabajo hago?". Entonces no estoy mirando los procesos disipativos!

Como hago para los fluidos si quiero aplicar PTV a fluidos? Tenes que plantear el Principio de trabajo virtual, lo mismo pero ahora con el tiempo ahi metido.

Hu-Washizu

$$\int_{V^e}\left[\frac{1}{2}\varepsilon^T C\varepsilon - \sigma^T\varepsilon + \sigma^T(\nabla u) - \overline{p}^T u\right]dV - \int_{S_\sigma^e}\overline{T}^T u\,dS$$

Separa parte elastica, deformaciones iniciales, etc. Sebas lo uso para que elementos Q6 pasen el Patch test.

5.1. Residuos Ponderados

Y si no tengo la fisica? Balance de una viga de seccion A con carga q.

$$A\frac{\partial \sigma_x}{\partial x} + q = 0$$

Pero el ultimo elemento de la viga esta sometido a carga P. La condicion de borde natural es:

$$AE\frac{\partial u}{\partial x} = P$$

Una condicion de borde comun seria $u = 0$ sobre el empotramiento.

Soy vidente y digo que $\widetilde{u} = a_1 x + a_2 x^2$. el mon o es por aproximada

Metiendolo en el PDE obtengo ecuaciones y resuelvo por los a. Si tenes mucha suerte los Residuos te dan cero tenes mucha suerte y lo que tenes en tus manos es una solucion al sistema. Wl residuo te dice cuanto error estas cometiendo. $R_s = 0$ vale cero siempre porque es una condicion que impusimos.

Que hago? Calculo para que en $x = x_0$ el residuo valga cero (Sebas dice en $x = L/3$).

Hago una grilla de puntos y pido que se cumpla la ecuacion en ellos.

Si planteo que el integral de R valga cero entonces como que coloc un punto en $x = L/2$. Colocacion por puntos, es rapido y medio chotengui, pero anda bien en una malla fina.

Metodo cuadrado minimos busca reducir el R violentamente. Integro el residuo al cuadrado mas el residuo en la condicion de borde al cuadrado multiplicado por un factor que podes multiplicarlo por grandes numero par a que se cumpla forzadamente. $1/L^2$ te da unidades correctas para dicho factor α

Galerkin: Las condiciones de borde naturales se llaman asi porque no las ves, pero estan metidas ahi en la ecuacion. a diferencia de las comunes quelas eligo yo.

Porque co'no meto una funcion a multiplicar el residuo? Porque esta proyectando de tal forma de que valga cero. Si soy inteligente para proectar voy a lograr que las proyecciones sean nulas.

Al vector puedo descomponerlo en dos direcciones. Proyectamos tal que nos de cero en ciertas direcciones. Depende de cuantos coeficientes pusistes, proyectas tantas veces.

En la filmina 12 Sebas reemplaza = $\int_V -\frac{\partial W_i}{\partial x}\frac{\partial u}{\partial x} d\upsilon$ y reemplaza cada peso correpsondiente.

Galerkin 1D pg 13: Proyecto las funciones de forma del nodo respecto las otras y si misma. vector de cargas $\sum_{nels} \int_L N_i q\, dx$

Si es sistema es par es estable. Fluidos con alta intercia, solidos. Si es impar entonces tenes problemas al tener ecuaciones cubicas (impares) mucho problema, no bueno.

Tener cuidado al elegir el grado. En fluidos se eligen diferentes grados para modelar la velocidad y la presion porque si no se traban.

6. Transferencia de Calor

Tenemos el flujo de calor que es $\underline{K}\nabla T$, la divergencia de esto es el flujo neto que pasa por un punto.

$$\nabla(\underline{K}\nabla T) + Q = \rho c \frac{\partial T}{\partial \tau}$$

Ley de Newton:

$$q_c = h_c(s,T)(T - T_\infty)$$

Ley de Stefan Boltzmann

$$q_r = \varepsilon_r \sigma \left(T^4 - T_\infty^4\right)$$

Es mas jodido el tema de modelar temperaturas fijas e imponer calor transferido, pues son aseveraciones no tan reales como imponer desplazamiento cero sobre un apoyo y fuerzas sobre vigas.

Metodo Galerkin

$$\int_\Omega -N_i k\left(\frac{\partial^2 T}{\partial x^2} + \frac{\partial^2 T}{\partial y^2} + \frac{\partial^2 T}{\partial z^2}\right) - N_i q \quad d\omega = 0$$

Se le ocurrio a alguien derivar respecto a K

$$(I)\frac{\partial\left(N_i k\frac{\partial T}{\partial x}\right)}{\partial x}=\frac{\partial N_i}{\partial x}k\frac{\partial T}{\partial x}+N_i k\frac{\partial^2 T}{\partial x^2}$$

e integro en el volumen

Lo que tenemos abajo es el integral de flujo de calor en la frontera (ultimo termino). $k\frac{\partial T}{\partial x}$ es el flujo de calor en direccion x.

$$\int_\Omega^\partial \frac{\partial\left(N_i k\frac{\partial T}{\partial x}\right)}{\partial x}+\frac{\partial\left(N_i k\frac{\partial T}{\partial y}\right)}{\partial y}+\frac{\partial\left(N_i k\frac{\partial T}{\partial z}\right)}{\partial z}d\omega=\oint_\Gamma N_i k\frac{\partial T}{\partial \overline{n}}ds$$

$T=N_j T_j$

N_j y N_i representan dos cosas. N_j nos da el perfil termico. Abajo multiplican entre ellas cruzadamente.

$$\int_\Omega \frac{\partial N_i}{\partial x}k\frac{\partial N_j}{\partial x}T_j+\frac{\partial N_i}{\partial y}k\frac{\partial N_j}{\partial y}T_j+\frac{\partial N_i}{\partial z}k\frac{\partial N_j}{\partial z}T_j-\int_\Omega N_i Q\,d\omega-\oint_\Gamma N_i q_{\vec{n}}\,ds=0$$

Temperature-Heat flux matrix **[B]**:

Para 2D Q4

$$\mathbf{[B]}=\begin{Bmatrix}\frac{\partial N}{\partial\xi}\frac{\partial\xi}{\partial x}+\frac{\partial N}{\partial\eta}\frac{\partial\eta}{\partial x}\\ \frac{\partial N}{\partial\xi}\frac{\partial\xi}{\partial y}+\frac{\partial N}{\partial\eta}\frac{\partial\eta}{\partial y}\end{Bmatrix}$$

Para aplicar conveccion

Hay una parte que depende de la temperatura interna, la parte que no depende la dejo como vector de cargas. La parte que depende la tengo que sumar a mi matriz de conductividades

$$R_{C_i}=\oint_{\Gamma_{comv}} N_i h\left(T(x)-T_{fl}\right)d\gamma$$

Lo escribe sebas: $kT=q_c=HT-RT_{fl}\longrightarrow(K-H)T=-RT_{fl}$

$$\text{Cargas}:R_H:\oint_{\Gamma_{conv}} N_i h T_{fl}\,d\gamma\qquad\qquad\text{Conductividad}:K:\oint_{\Gamma_{cm}} N_i h N_j\,d\gamma$$

La radiacion se hace de forma iterativa y converge porque es monotona. Aumenta la temperatura de mi sistema y por lo tanto aumenta la radiacion.

6.1. Resolución de problemas típico

Condiciones de borde de temperatura

$$\mathbf{\{T_x\}=[K_{xx}]^{-1}(\{Q_c\}-[K_{xc}]\{T_c\})}$$

luego de obtener las temperaturas desconocidas se obtiene el flujo desconocido

$$\mathbf{\{Q_x\}=[K_{cx}]\{T_x\}+[K_{cc}]\{T_c\}}$$

6.2. Transitorio

Matriz capacidad

$$[C]=\int_\Omega [N]^T\rho\,c[N]d\omega$$

El calculin

$$[C]\{\dot{T}\}+[K]\{T\}=\{R_B\}+\left\{R_Q\right\}=\{R_T\}=[C]=\int_\Omega [N]^T\rho\,c[N]d\omega$$

Sebas dice $C\dot{T}+kT=R$. Cuando esta en equilibrio? En tiempo n esta en equilibrio tal que $C\dot{T}^n+kT^n=R^n$, si podemos calcular el tiempo $n+1$ entonces podemos resolver el problema transitorio. Con un *peso*

(que llamamos beta) podemos darle mas bola al tiempo n o el tiempo $n+1$. $(T^{n+1}-T^n)/\Delta t = \dot{T}^{n+\beta}$ es lo que puedo calcular. Si $\beta = 1$ entonces

$$\frac{T^{n+1}-T^n}{\Delta t} = \dot{T}^{n+1}$$

y si $\beta = 0$

$$\frac{T^{n+1}-T^n}{\Delta t} = \dot{T}^{n}$$

$$\dot{T}^{n+\beta} = (1-\beta)\dot{T}^n + \beta\,\dot{T}^{n+1}$$

$$\dot{T}^{n+1}$$

Si a beta le digo que vale cero el futuro muere en cambio si beta vale 1 entonces TODO depende del futuro.

$$\beta[C]\{\dot{T}\}^{n+1} + (1-\beta)[C]\{\dot{T}\}^n + [K]\beta\{T\}^{n+1} + [K](1-\beta)\{T\}^n = (1-\beta)\{R_T\}^n + \beta\,\{R_T\}^{n+1}$$

Ecuación iterativa:

$$\mathbf{\{T\}}^{n+1} = \left(\mathbf{[C]} + \Delta t\beta\mathbf{[K]}\right)^{-1}\left[\left(\mathbf{[C]} - \Delta t(1-\beta)\mathbf{[K]}\right)\mathbf{\{T\}}^n + \Delta t\left((1-\beta)\mathbf{\{R\}}^n + \beta\mathbf{\{R\}}^{n+1}\right)\right] \tag{21}$$

$\beta = 0$	Euler Forward Difference
$\beta = 0{,}5$	Crank Nicholson
$\beta = 0{,}666$	Galerkin
$\beta = 1$	Backward Difference

La estabilidad es pedir que los autovalores de A sean menores que 1. donde

$$[A] = ([C] + \Delta t\beta[K])^{-1}([C] - \Delta t(1-\beta)[K])$$

EStabilidad 1-D:

$$A = \frac{c - \Delta t(1-\beta)k}{c + \Delta t\beta k} = \frac{1 - \Delta t(1-\beta)\omega}{1 + \Delta t\beta\omega}$$

Donde

$$\omega = \frac{k}{c}$$

tal que

$$\left|\frac{1 - \Delta t\omega + \Delta t\beta\omega}{1 + \Delta t\omega}\right| < 1$$

$$1 - \Delta t\omega + \Delta t\beta\omega > -1 - \Delta t\beta\omega$$

$$2 - \Delta t\omega + 2\Delta t\beta\omega > 0$$

$$2 + (-1 + 2\beta)\Delta t\omega > 0$$

$$(-1 + 2\beta)\Delta t\omega > -2$$

$$\Delta t\omega > \frac{-2}{-1 + 2\beta}$$

Si beta es menor a un medio entonces siempre es positivo y por lo tanto no importa el paso de tiempo que tome siempre va converger.

Si beta es 0 te queda que es inestable. CORREGIDO $\lambda > -1$ por ende la desigualdad de abajo te puede quedar positiva o negativa dependiendo del valor de β

$$\Delta t\,\omega(2\beta - 1) > -2$$

7. No-linealidad

Porque cuando deformas una regla aumenta la rigidez? En parte porque cambia la geometria (a una viga curva) y porque hay tensiones remanentes al deformarse. Las tensiones remanentes se pueden pensar como una cuerda de guitarra que al ser tensionada son mas 'rigida' las cuerdas, cuesta mas deformar...

Cargando por desplazamitos no me pierdo de soluciones intermedias, como por ejemplo una viga que pandea deja de tomar fuerzas hasta que se la vuelve a deformar suficiente.

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 7.1 (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 7.2 (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 7.3 (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 7.4 (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 7.5 (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 7.6 (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 7.7 (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 7.8 (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 7.9 (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)
```

7.0.1. Vibraciones

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

Codigo 7.10 (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 7.11 (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 7.12 (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 7.13 (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
```

7.1. Pandeo

Codigo 7.14 (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 7.15 (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);
```

Codigo 7.16 (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
```

7.1.1. Matriz de transformación para elementos 1D

Codigo 7.17 (Barra. ϕ en grados)

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

Codigo 7.18 (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];
```