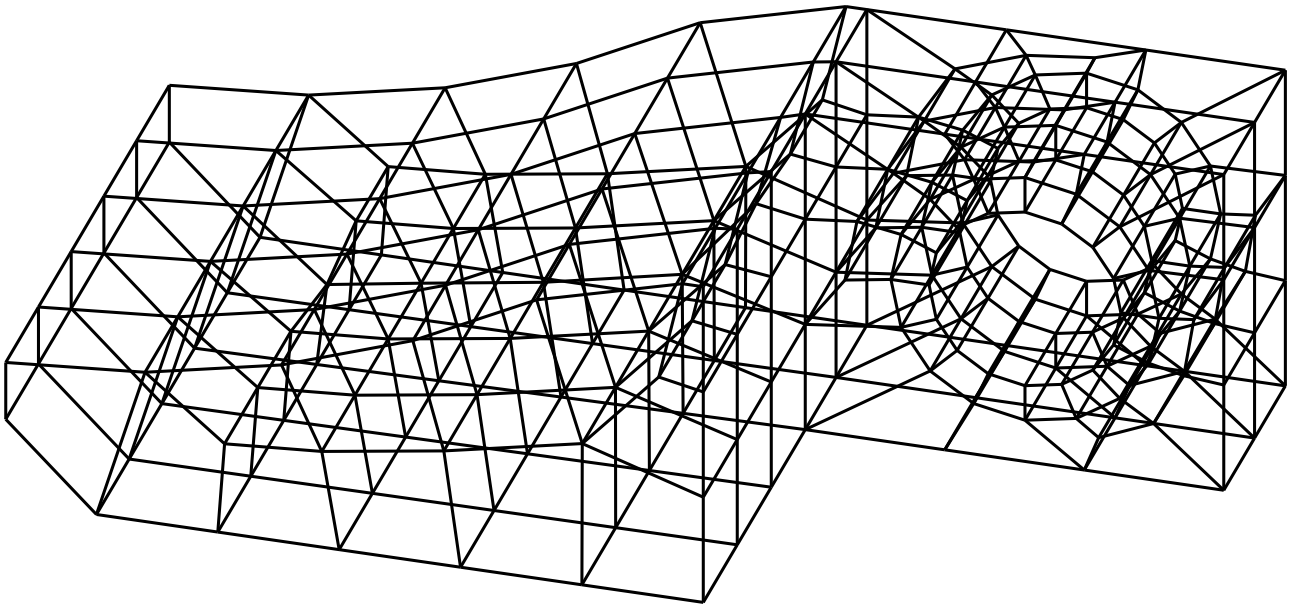
Instituto Tecnológico de Buenos Aires

Elementos Finitos

APUNTE NO OFICIAL PARA LA MATERIA 30.07 Y 31.92



PATRICIO WHITTINGSLOW

Índice general

Sobre esta obra

Este documento fue creado usando LaTeX para las materias Elementos Finitos I y Elementos Finitos II del ITBA por un alumno. Edición 13 de abril de 2020.

El código fuente se puede encontrar en mi repositorio de github ~/soypat/whittileaks. Licenciado bajo Creative Commons 4.0 CC-BY-NC-SA.

Códigos MATLAB

0.0. Introducción al Método de los Elementos Finitos

El análisis o método de elementos finitos (FEA por *finite element analysis*) es usado para obtener una solución numérica de un problema de campo (electrostático, térmico, tensiones). Matemáticamente estos problemas están definidos como ecuaciones diferenciales o como un integral. Ambas expresiones pueden formularse con elementos finitos.

Las ventajas de FEA son

- Es aplicable a cualquier problema de campo
- No hay restricciones geométricas
- No hay restricciones al tipo de cargas o condiciones de borde que se pueden aplicar
- Se puede formular para materiales que no son isotrópicos e incluso el tipo de material puede cambiar dentro de un elemento
- Se pueden combinar distintos tipos de elementos en un modelos, por ejemplo, unir barras con vigas o incluso con elementos 3D.
- La aproximación se puede mejorar fácilmente refinando la malla donde hay gradientes de tensión altos

Proceso de resolución

El primer paso es identificar y **clasificar** el problema.

- Cuales son los fenómenos físicos involucrados y que resultados se buscan del análisis
- Depende del tiempo? (estático o dinámico)
- Hace falta una resolución iterativa? (no linealidad: radiación, plasticidad)

Luego se comienza el **modelado** del problema.

- Se excluyen los detalles superfluos, dejando los esencial para describir el problema con un margen de error adecuado sin complicar las cosas innecesariamente
- Un *modelo geométrico* se convierte en un *modelo matemático* cuando se describe su comportamiento mediante ecuaciones diferenciales y condiciones de borde.*

Un modelo matemático es una idealización donde se simplifican la geometría, propiedades del material, cargas y/o condiciones de borde en base del entendimiento del analista acerca lo que tiene (o no) importancia al momento de obtener los resultados requeridos.

Finalmente llega el momento de la **discretización**. Un modelo matemático se discretiza dividiéndolo en una malla de elementos finitos. De esta forma, un campo continuo es representado como una función partida la cual es definida por una cantidad finita de variables nodales e interpolación dentro de cada elemento.

Tipos de error

Al momento de discretizar se introduce error conocido como **error de discretización**. Eso sucede porque se aproxima un campo *suave* con una función partida. Aumentar el numero de elementos puede disminuir este error pero nunca eliminarlo.

Aún reduciendo el error de discretización se tendría **error numérico** porque toda computadora usa números de finita precisión para efectuar aritmética. Este error suele ser mínimo cuando se discretiza de forma adecuada y no se tiene una situación física propensa al *mal condicionamiento*.

Cabe destacar que se introdujo error antes de hacer una sola cuenta! El **error de modelado** se introduce por necesidad de simplificar el problema. Las cargas puntuales, los soportes fijos y los materiales perfectamente homogéneos no existen en la realidad! Cook et al. [2007].

*Un modelo de FEA no es la realidad, es una *simulación*. Difícilmente se obtengan resultados buenos cuando se aplique FEA a un modelo matemático que no refleja la realidad de forma apropiada.

Principios básicos de elementos finitos

La ecuación que se resuelve es

$$M\ddot{d} + C\dot{d} + K\,d = F^{\text{externas}}$$

para un sistema mecánico. Es común tratar problemas estáticos tener como variables de entrada las fuerzas externas F^{externas} (peso propio, fuerzas aplicadas, fuerza centrifuga etc.) y la rigidez del sistema K. Los desplazamientos d sería la variable que se desea obtener. La ecuación a resolver entonces es

$$d = K^{-1} \cdot F^{\text{externas}}$$

El método de los elementos finitos entonces tiene su campo de rigidez K que se suele llamar la matriz de rigidez global **[K]**. Esta asocia rigidez con los grados de libertad de los nodos obtenidos de la discretización. Para cuerpos sólidos en el espacio hay 6 grados de libertad, 3 de desplazamiento $(u,\ v\ ,w)$ y tres de giro (φ, θ, ψ).

Antes de discretizar un modelo se eligen las direcciones x, y, z globales. Los desplazamientos obtenidos corresponderán a estas direcciones.

Capítulo 1

Programación del método de los elementos finitos

Objeto matemático	Nombre de variable	Definición
$N_{\text{nod.}}$	`Nnod`	Número de nodos
$N_{\text{elem.}}$	`Nelem`	Número de elementos
$\vartheta_{\text{nod.}}$	`Ndofpornod`	Número dof por nodo
$N_{\text{dof}} = \vartheta_{\text{nod.}} \cdot N_{\text{nod.}}$	`Ndof` o simplemente `dof`	Número de dof
$n_{\text{nod.}}$	`Nnodporelem`	Número de nodos por elemento
$n_{\text{dof}} = \vartheta_{\text{nod.}} \cdot n_{\text{nod.}}$	`Ndofporelem`	Número de dof por elemento
$N_{\text{dim.}}$	`Ndim`	Número de dimensiones
[k]	`ke`	Matriz de rigidez del elemento
[K]	`K`	Matriz de rigidez global
{R}	`R`	Vector de cargas
{D}	`D`	Vector de desplazamientos

Tabla 1.1: Parámetros importantes para la resolución de elementos finitos. Comúnmente denominados *dofinitions*.

Se sugiere al lector usar los nombres de variables mencionados en la tabla 1.1. La motivación de los nombres de variable es que sean legibles y cortos a la vez ya que aparecen seguido en un código de elementos finitos. '`N`' se lee Número y '`elem`' se lee como elementos etc. Por ejemplo, `Ndofpornod` se leería: Número de dof (grados de libertad) por nodo.

1.1. Mallado

Primero se requiere discretizar el problema. Se comienza con una matriz de nodos que va unir los elementos.

$$\texttt{nodos} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 6.0 & 0 \\ 6.0 & 8.0 & 0 \\ 12.0 & 6.0 & 0 \\ 12.0 & 0 & 0 \end{bmatrix}_{N_{\text{nod.}} \times N_{\text{dim.}}} \tag{1.1}$$

Para acceder a las coordenadas del nodo enésimo en MATLAB: `nodos(n,:)`. Por ejemplo, el cuarto nodo está ubicado en las coordenadas $(x;y;z) = (12;6;0)$.

La matriz de conectividad de los elementos:

$$\texttt{elementos} = \begin{bmatrix} 1 & 2 \\ 2 & 3 \\ 5 & 4 \\ 4 & 3 \end{bmatrix}_{N_{\text{elem.}} \times n_{\text{nod.}}} \tag{1.2}$$

El orden de la numeración es lo que orienta al elemento. Tener especial cuidado cuando se trabaje con elementos 2D y 3D, numerar los nodos contraria a la formulación escogida puede causar problemas de jacobianos negativos!

Con estas dos matrices se ha definido una malla que podría resolver el caso de la figura 2.4b.*

Las matrices `nodos` y `elementos` definen algunos parámetros importantes (código MATLAB):

```
[Nnod, Ndim] = size(nodos);
[Nelem, Nnodporelem] = size(elementos);
```

1.2. Mapeo de dof

Para resolver el problema de elementos finitos hay que asociar dof a los elementos. Definido abajo la **función de mapeo** nodo a dof. Se ingresa el nodo de interés n y la función devuelve los dof asociados a ese nodo.

$$\texttt{node2dof}(n) = \begin{bmatrix} n\cdot\vartheta_{\text{nod.}} - \vartheta_{\text{nod.}} + 1 \\ n\cdot\vartheta_{\text{nod.}} - \vartheta_{\text{nod.}} + 2 \\ \vdots \\ n\cdot\vartheta_{\text{nod.}} \end{bmatrix}_{\vartheta_{\text{nod.}}\times 1} \tag{1.3}$$

la cual se programó como una función anónima vectorizada.

Código 1.1: `node2dof` para $\vartheta_{\text{nod.}}$=3.

```
node2dof = @(n) [n*3-2; n*3-1; n*3];
```

Con lo visto hasta ahora se puede entonces obtener la matriz de **dof asociados a los elementos**, `elemdof`. En MATLAB:

Código 1.2: Obtención de matriz elemdof.

```
elemdof = zeros(Nelem,Ndofporelem);
for e = 1:Nelem
    elemdof(e,:)= reshape(node2dof(elementos(e,:)),[],1);
end
```

La función `reshape(X,m,n)` devuelve una matriz de `m`×`n` con los elementos de la matriz `X` recorriendo las columnas de arriba para abajo.

Considerando $\vartheta_{\text{nod.}} = 3$ para el ejemplo que se viene tratando se tiene:

$$\texttt{elemdof} = \begin{bmatrix} 1 & 2 & 3 & 4 & 5 & 6 \\ 4 & 5 & 6 & 7 & 8 & 9 \\ 13 & 14 & 15 & 10 & 11 & 12 \\ 10 & 11 & 12 & 7 & 8 & 9 \end{bmatrix}_{N_{\text{elem.}}\times n_{\text{dof}}} \tag{1.4}$$

1.3. Acople de rigidez

Se itera sobre los elementos, obteniendo la rigidez del elemento de alguna forma, sea cual sea (integración directa, cuadratura de Gauss, matriz predefinida, etc). Una vez obtenida se acopla **[k]** al sistema usando la matriz dof asociados a los elementos para obtener $\mathbf{[K]}_{N_{\text{dof}}\times N_{\text{dof}}}$

Código 1.3: Obtención generica de matriz de rigidez.

```
K = sparse(Ndof, Ndof); %Funcionalmente equivalente a hacer: K = zeros(Ndof);
for e = 1:Nelem
    meindof = elemdof(e,:);
    ke = int(B'*E*B,0,L);
    K(meindof, meindof) = K(meindof, meindof) + ke;
end
```

*Para problemas de vigas 3D se tienen 6 dof por nodo, osea `Ndofpornod=6;`.

donde `sparse(m,n)` crea una matriz *sparse* de tamaño `m` × `n`.[†] Almacenar la matriz en forma *sparse* no solo ahorra espacio en memoria pero también acorta tiempos de resolución. Es recomendable su uso cuando $N_{\text{dof}} > 10^4$.

Si el lector desea profundizar su conocimiento, se lo dirige a leer Chessa [2002].

1.4. Aplicación de condiciones de borde esenciales

La ecuación diferencial que se resuelve mediante el método de elementos finitos tiene la forma

$$L\mathbf{u} + \mathbf{q} = 0 \tag{1.5}$$

donde **u** es el vector de variables primario, L es el operador diferencial y **q** es el vector de funciones conocidas. Una ecuación diferencial se le pueden aplicar **condiciones de borde naturales** (del tipo Neumann) y **condiciones de borde esenciales** (del tipo Dirichlet) [Dixit, 2007]

Esenciales Es necesario por lo menos una condición de borde esencial para la resolución *completa* del problema. Se imponen a la variable primaria **u** (impuesto a los desplazamientos)

Naturales Son las condiciones de borde que involucran términos derivados de orden superior y no son suficientes por si solas para la resolución del problema. Se imponen a la variable secundaria **q** (fuerzas, tracciones etc.)

La condición de borde esencial más simple de aplicar al momento de programar es igualar el desplazamientos a 0. En MATLAB se puede trabajar este concepto usando matrices lógicas con relativa facilidad

Código 1.4: Aplicación de condiciones de borde esenciales.

```
isFixed = false(Ndof,1); % Crea un vector columna boolean
isFixed(node2dof(n)) = true;  % Se empotra' el nodo nisFixed([14 2]) = true; isFree =
    isFixed;
```

En la sección 5.1 se verá como imponer desplazamientos no nulos.

Siguiendo el ejemplo, ¿cómo aplicaría los empotramientos A y B de la figura 2.4b? Y si uno fuera un apoyo simple, ¿cómo haría?

1.5. Aplicación de condiciones de borde naturales

Ya sabemos lo que es una condición de borde natural de la última sección, queda aplicar dicho conocimiento. Llegado a este punto se remarca la importancia de usar **unidades consistentes**. El autor sugiere usar newtons, metros y pascales.

Código 1.5: Condiciones de borde naturales (cargas).

```
R = zeros(Ndof,1);
R([1 2 15]) = 9e3;  % Cargo mis dof 1 2 y 15 con 9000 unidades
```

¿Tiene sentido aplicar cargas a un dof en conjunto con condiciones de borde esenciales?

1.6. Procedimiento de resolución

El sistema a resolver es $\{\mathbf{D}\} = [\mathbf{K}]^{-1}\{\mathbf{R}\}$ aunque no es económico invertir la matriz [**K**] desde un punto de vista numérico: La matriz [**K**] es esparsa y su inversa es una matriz *full*.[‡] Hay varias formas de resolver el problema que no requieren de la inversa, MATLAB tiene incorporado el operador `mldivide(A,B)` que resuelve un sistema de ecuaciones lineal y devuelve `x` tal que $Ax = B$.

Código 1.6: Resolución del sistema lineal.

```
Dr = K(isFree,isFree)\R(isFree); % == mldivide(K(isFree,isFree),R(isFree));
```

[†]Una matriz esparsa es una matriz cuyos elementos son, en la gran mayoría, igual a cero.

[‡]Una matriz full no tiene elementos que son cero.

Se suele llamar a la matriz con condiciones de borde aplicadas la **matriz de rigidez reducida**. En el caso que el sistema sea singular (no hay suficientes condiciones de borde esenciales para hallar una solución), o que el problema esté mal condicionado, `mldivide` devuelve un aviso.

En el código de arriba se cálculo `Dr` con las matrices de carga y rigidez reducidas, por ende `Dr` tiene tamaño `Ndof-Ncb` donde `Ncb` son la cantidad de dof restringidos.[§] Es conveniente obtener el vector de desplazamientos global para el pos-procesado:

Código 1.7: Recuperación de desplazamientos globales.

```
D = zeros(Ndof,1);
D(isFree) = Dr;
```

`D` entonces será igual a `Dr` excepto que tendrá ceros en los dof donde se aplicaron condiciones de borde esenciales.

Como nadie nació leyendo vectores columnas se pueden reorganizar los desplazamientos en una matriz que muestre los desplazamientos de cada nodo en sus filas. También se puede obtener la posición deformada de los nodos sumando las matrices.

Código 1.8: Obtención de posición deformada.

```
desplazamientos = reshape(D,[],Ndofpornod)';
% si los primeros 3 dof son u,v,w puedo sumar:
posiciondeformada = nodos + mag*desplazamientos(:,1:3);
```

donde `mag` es una variable para amplificar las deformaciones y que se puedan ver en un gráfico. Suele estar entre 30-100 para problemas con pequeños desplazamientos.

Aún se desconocen las reacciones del problema. Se pueden obtener todas las fuerzas externas con:

Código 1.9: Obtención de reacciones.

```
Rext = K*D;
Rext(node2dof(n)) % Puedo visualizar reacciones en nodo n
Rext(elemdof(2)) % Fuerzas externas sobre el elemento 2
```

`Rext` tendrá las reacciones en los dof restringidos y las fuerzas externas sobre el sistema *en coordenadas globales*. Más acerca de las fuerzas sobre un elemento en la sección 2.3.

Resolución de problema de autovalores

Un problema de autovalores tiene la forma $([\mathbf{A}]-\lambda[\mathbf{B}])\{\mathbf{x}\}=\{\mathbf{0}\}$ donde hay otras soluciones ademas de la trivial. λ_i son los autovalores que satisfagan la ecuación. A cada uno de estos autovalores le corresponde un autovector $\{\mathbf{x}\}_i$.

Código 1.10: Resolución de problema de autovalores

```
[autovec, autoval] = eig(B\A);
```

[§] A pesar de no tener mucho uso en un programa de elementos finitos se deja la definición: `Ncb=sum(isFixed)`

Capítulo 2

Elementos 1D

2.1. Elemento barra

Uno puede pensar el elemento barra como un resorte que solo ejerce fuerza en la dirección en que apunta su eje local x'.

$$[\mathbf{k}']_{\text{barra}} = \begin{bmatrix} X & -X \\ -X & X \end{bmatrix} \begin{matrix} u_1 \\ u_2 \end{matrix} \qquad \text{donde} \qquad X = \frac{EA}{L} \tag{2.1}$$

si se acopla la matriz a **[K]** así como está solo otorga rigidez en la dirección x global. Si se quiere modelar una barra en el plano xy se tiene que rotar la barra según su orientación. Si el modelo es plano, osea solo se modelan x e y global, se puede rotar la barra como será visto en la sección 2.3.

Para orientar una barra que se encuentra en el espacio se puede usar la matriz de rotación para una viga 3D con una leve modificación a la matriz de rigidez de la barra. Esta tiene que incluir todos los grados de libertad del problema! Es decir, la matriz $[\mathbf{k}']_{\text{barra}}$ termina siendo de 12×12 para un problema de 6 grados de libertad por nodo.

$$\mathbf{k}_{1,1} = \mathbf{k}_{7,7} = X, \qquad \mathbf{k}_{7,1} = \mathbf{k}_{1,7} = -X, \quad \text{las demás:} \quad \mathbf{k}_{i,j} = 0$$

Un lector mosca ser dará cuenta que la matriz de rigidez de la viga Timoshenko 3-D (expresión 2.3) es la generalización para todo elemento 1D: barras e incluso vigas en el plano.*

Bar element test

Llegado a este punto en la lectura, el autor se imagina que el lector debe estar ansioso por poner a prueba sus conocimientos. La figura 2.1 describe un problema estático que se puede resolver con una breve análisis a mano alzada. Este tipo de problema se denomina *patch test* o *element test* porque sirve para ensayar la calidad del unión de elementos, el elemento en si, la aplicación de condiciones de borde y método de resolución.

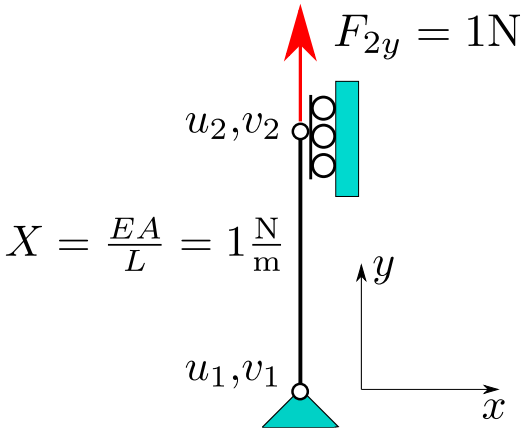


Figura 2.1: Un *bar element test* para verificar un *solver* de elementos finitos. $v_2 = 1\text{m}$.

*Las formulaciones de vigas en el plano más comunes son de dos (v, θ) y tres (u,v,θ) grados de libertad por nodo.

El *bar element test* se puede mallar como un problema plano en el plano xy con dos nodos ($x_1 = y_1 = x_2 = 0,\ y_2 = 1\text{m}$) y un elemento que los une. La rigidez del material entonces seria $EA = 1\text{N}$. Se aplican las condiciones de borde detalladas en la figura: se restringen u_1, v_1 y u_2 y se resuelve el sistema para obtener v_2. Con este element test se verifica el método de resolución, la forma en que se rota la matriz de rigidez de la barra, la aplicación de condiciones de borde y el elemento en si.

2.2. Viga 3D de Timoshenko

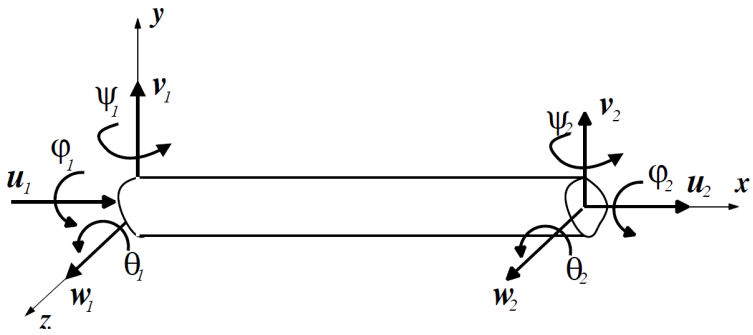


Figura 2.2: Grados de libertad (*dof*) y coordenadas locales de una viga 3D de dos nodos y 6 dof por nodo.

La viga de Timoshenko 3D tiene las siguientes funciones de forma (forma eficiente tomada de Luo [2008])

$$
\begin{cases}
\left.\begin{aligned} N_1 &= 1-\xi \\ N_2 &= \xi \end{aligned}\right\} \text{ Funciones de forma para barras} \\
H_{v_1} = \beta_y\left(2\xi^3 - 3\xi^2 + \alpha_y\xi + 1 - \alpha_y\right) \\
H_{v_2} = \beta_y\left(-2\xi^3 + 3\xi^2 - \alpha_y\xi\right) \\
H_{w_1} = \beta_z\left(2\xi^3 - 3\xi^2 + \alpha_z\xi + 1 - \alpha_z\right) \\
H_{w_2} = \beta_z\left(-2\xi^3 + 3\xi^2 - \alpha_z\xi\right) \\
H_{\theta_1} = L\beta_y\left[\xi^3 + \left(\tfrac{1}{2}\alpha_y - 2\right)\xi^2 + \left(1 - \tfrac{1}{2}\alpha_y\right)\xi\right] \\
H_{\theta_2} = L\beta_y\left[\xi^3 - \left(1 + \tfrac{1}{2}\alpha_y\right)\xi^2 + \left(\tfrac{1}{2}\alpha_y\right)\xi\right] \\
H_{\psi_1} = L\beta_z\left[\xi^3 + \left(\tfrac{1}{2}\alpha_z - 2\right)\xi^2 + \left(1 - \tfrac{1}{2}\alpha_z\right)\xi\right] \\
H_{\psi_2} = L\beta_z\left[\xi^3 - \left(1 + \tfrac{1}{2}\alpha_z\right)\xi^2 + \left(\tfrac{1}{2}\alpha_z\right)\xi\right] \\
G_{v_1} = \frac{6\beta_y}{L}\left(\xi^2 - \xi\right) \\
G_{v_2} = \frac{6\beta_y}{L}\left(-\xi^2 + \xi\right) \\
G_{w_1} = \frac{6\beta_z}{L}\left(\xi^2 - \xi\right) \\
G_{w_2} = \frac{6\beta_z}{L}\left(-\xi^2 + \xi\right) \\
G_{\theta_1} = \beta_y\left[3\xi^2 + \left(\alpha_y - 4\right)\xi + 1 - \alpha_y\right] \\
G_{\theta_2} = \beta_y\left[3\xi^2 - \left(\alpha_y + 2\right)\xi\right] \\
G_{\psi_1} = \beta_z\left[3\xi^2 + (\alpha_z - 4)\xi + 1 - \alpha_z\right] \\
G_{\psi_2} = \beta_z\left[3\xi^2 - (\alpha_z + 2)\xi\right]
\end{cases}
$$

donde x' es la coordenada local sobre la viga:

$$
\xi = \frac{x'}{L}, \qquad \alpha_y = \frac{12EI_y}{kGAL^2}, \qquad \beta_y = \frac{1}{1-\alpha_y}, \qquad \alpha_z = \frac{12EI_z}{kGAL^2}, \qquad \beta_z = \frac{1}{1-\alpha_z}
$$

donde k es el **factor de corrección por corte**[†] y depende de la sección y del modulo de Poisson ν. Algunos valores en la tabla 8.2 del anexo.

[†]Ideado por Timoshenko en 1921 para permitir un mejor cálculo de las frecuencias naturales Dong et al. [2010].

Estas funciones de forma interpolan los desplazamientos sobre la viga:

$$\begin{cases} u = N_1 u_1 + N_2 u_2 \\ v = H_{v_1} v_1 + H_{\theta_1} \theta_1 + H_{v_2} v_2 + H_{\theta_2} \theta_2 \\ w = H_{w_1} w_1 + H_{\psi_1} \psi_1 + H_{w_2} w_2 + H_{\psi_2} \psi_2 \\ \varphi = N_1 \varphi_1 + N_2 \varphi_2 \\ \theta = G_{v_1} v_1 + G_{\theta_1} \theta_1 + G_{v_2} v_2 + G_{\theta_2} \theta_2 \\ \psi = G_{w_1} w_1 + G_{\psi_1} \psi_1 + G_{w_2} w_2 + G_{\psi_2} \psi_2 \end{cases} \tag{2.2}$$

para luego calcular los esfuerzos usando las mismas formulas vistas en estática y resistencia de materiales.

$$M_z = E I_z \frac{\mathrm{d}^2 v}{\mathrm{d}x^2}, \qquad V_y = \frac{\mathrm{d}M_z}{\mathrm{d}x} = E I_z \frac{\mathrm{d}^3 v}{\mathrm{d}x^3}, \qquad N_x = AE \frac{u_2 - u_1}{L}$$
$$T = G J_T \frac{\varphi_2 - \varphi_1}{L}, \qquad M_y = E I_y \frac{\mathrm{d}^2 w}{\mathrm{d}x^2}, \qquad V_z = E I_y \frac{\mathrm{d}^3 w}{\mathrm{d}x^3}$$

donde J_T es la constante torsional de la viga y A es la sección. [‡]

La matriz de rigidez se puede obtener[§] integrando analíticamente a la funciones de forma mencionadas anteriormente sobre el largo de la viga. En este documento no se va tratar la matriz de rigidez que resulta de dicha integración. Se presenta al lector la matriz de rigidez de una viga Timoshenko 3D clásica Cook et al. [2007]:

$$[\mathbf{k}']_{1\mathrm{D}} = \begin{bmatrix} X & 0 & 0 & 0 & 0 & 0 & -X & 0 & 0 & 0 & 0 & 0 \\ 0 & Y_1 & 0 & 0 & 0 & Y_2 & 0 & -Y_1 & 0 & 0 & 0 & Y_2 \\ 0 & 0 & Z_1 & 0 & -Z_2 & 0 & 0 & 0 & -Z_1 & 0 & -Z_2 & 0 \\ 0 & 0 & 0 & S & 0 & 0 & 0 & 0 & 0 & -S & 0 & 0 \\ 0 & 0 & -Z_2 & 0 & Z_3 & 0 & 0 & 0 & Z_2 & 0 & Z_4 & 0 \\ 0 & Y_2 & 0 & 0 & 0 & Y_3 & 0 & -Y_2 & 0 & 0 & 0 & Y_4 \\ -X & 0 & 0 & 0 & 0 & 0 & X & 0 & 0 & 0 & 0 & 0 \\ 0 & -Y_1 & 0 & 0 & 0 & -Y_2 & 0 & Y_1 & 0 & 0 & 0 & -Y_2 \\ 0 & 0 & -Z_1 & 0 & Z_2 & 0 & 0 & 0 & Z_1 & 0 & Z_2 & 0 \\ 0 & 0 & 0 & -S & 0 & 0 & 0 & 0 & 0 & S & 0 & 0 \\ 0 & 0 & -Z_2 & 0 & Z_4 & 0 & 0 & 0 & Z_2 & 0 & Z_3 & 0 \\ 0 & Y_2 & 0 & 0 & 0 & Y_4 & 0 & -Y_2 & 0 & 0 & 0 & Y_3 \end{bmatrix} \begin{matrix} u_1 \\ v_1 \\ w_1 \\ \varphi_1 \\ \psi_1 \\ \theta_1 \\ u_2 \\ v_2 \\ w_2 \\ \varphi_2 \\ \psi_2 \\ \theta_2 \end{matrix} \tag{2.3}$$

donde

$$X = \frac{AE}{L}, \qquad Y_4 = \frac{2EI_z}{L}, \qquad Y_3 = 2Y_4, \qquad Y_2 = \frac{3Y_4}{L}, \qquad Y_1 = \frac{2Y_2}{L}$$
$$Z_4 = \frac{2EI_y}{L}, \qquad Z_3 = 2Z_4, \qquad Z_2 = \frac{2Z_4}{L}, \qquad Z_1 = \frac{2Z_2}{L}, \qquad S = \frac{GJ_T}{L}$$

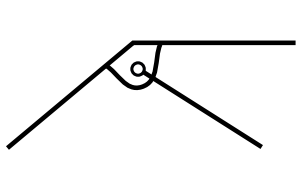
Rótulas

Las uniones rotuladas permiten que ciertos elementos giren libremente manteniendo rigidez ante la rotación de otros elementos. El primer paso consiste en **desacoplar** el/los[¶] giro/s del elemento rotulado. Esto se logra generando un nuevo grado de libertad para cada giro desacoplado, el cual *no pertenece a ningún nodo*, un *nodeless dof*. Este nuevo dof se puede introducir al final de la matriz de rigidez para no estropear el mapeo de dof visto anteriormente. Al momento de acoplar la rigidez del elemento a la matriz de rigidez global la matriz `elemdof` (sección 1.2) tiene que mostrar este cambio con los dof desacoplados. Para el caso de la figura 2.3a con grados de libertad u, v, θ por nodo:

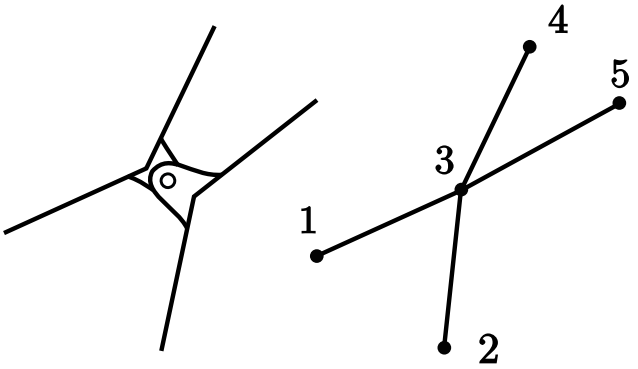
[‡]La constante torsional J_T es igual a I_p para secciones de viga circulares. Para perfiles abiertos de paredes delgadas, como por ejemplo un perfil doble T o un perfil 'C', J_T es mucho mas chico que I_p.

[§]Para una viga uniforme con sección simétrica y de material en su rango elástico.

[¶]Suponiendo que la figura 2.3a está contenida en el plano xy: el giro a desacoplar sería en z (θ) dado la geometría de la rótula.



(a) Una unión rotulada y una posible discretización.



(b) Dos sistemas de vigas unidos por una rótula y una posible discretización.

Figura 2.3

$$\texttt{elementos} = \begin{bmatrix} 1 & 2 \\ 2 & 4 \\ 3 & 2 \end{bmatrix}, \qquad \texttt{elemdof} = \begin{bmatrix} 1 & 2 & 3 & 4 & 5 & 6 \\ 4 & 5 & 6 & 10 & 11 & 12 \\ 7 & 8 & 9 & 4 & 5 & 13 \end{bmatrix}$$

note que el elemento rotulado comparte los desplazamientos u, v sobre el nodo 2 (dof 4 y 5) con los otros dos elementos pero tiene su giro θ desacoplado.

¿Como quedaría la matriz `elemdof` *del sistema de la figura 2.3b?*

2.3. Orientación de elementos 1D

Los elementos detallados en la sección anterior están acostados sobre el eje x. Para orientar un elemento en el espacio se tiene que empezar de hablar de una *matriz de rotación* $[\mathbf{T}]$.

$$[\mathbf{k}]_{\text{rotada}} = [\mathbf{T}]^T [\mathbf{k}'][\mathbf{T}] \tag{2.4}$$

donde $[\mathbf{k}']$ es la matriz de rigidez local del elemento 1D antes de ser rotado. Luego de ser rotada la matriz puede ser acoplada a la rigidez global del sistema $[\mathbf{K}]$.

Cabe agregar que los desplazamientos relacionados con el sistema global corresponden a los ejes globales. Para obtener los desplazamientos locales de los elementos se tienen que rotar estos también!

$$\{\mathbf{d}'\} = [\mathbf{T}]\{\mathbf{d}\} \tag{2.5}$$

donde $\{\mathbf{d}'\}$ son los desplazamientos del elemento en las coordenadas locales. Obtener $\{\mathbf{d}'\}$ resulta útil al momento de querer calcular las fuerzas[◇] locales del elemento: $[\mathbf{k}']\{\mathbf{d}'\} = -\{\mathbf{r}'\}$ o para interpolar los desplazamientos usando las expresiones en (2.2).

Orientación de barras en el plano

La matriz de rigidez de una barra es definida horizontal. Esto significa que toda la rigidez está en x.

Una barra rotada ϕ grados va tener una nueva rigidez en el eje global x, e incluso puede no tener rigidez si se rota 90 grados.

La matriz transformación de una barra en el plano está dada por

$$[\mathbf{T}]_{\text{Barra}} = \begin{bmatrix} \cos\phi & 0 \\ \sin\phi & 0 \\ 0 & \cos\phi \\ 0 & \sin\phi \end{bmatrix}$$

[◇]El signo negativo es para obtener las fuerzas **internas** actuando en el elemento.

Orientación de vigas en el espacio

Para orientar una viga se tiene que aportar más información que para una barra. Mientras que una barra se puede orientar con el simple input de sus nodos, la viga de Timoshenko necesita orientar su eje y' además de su eje x' .

Algunos programas como ADINA, por ejemplo, permiten al usuario especificar un nodo auxiliar o un vector auxiliar que apunta en la dirección en la cual se desea que la viga tenga su eje y'.

Supongamos que elegimos este vector auxiliar $s_y = [0,1,0]$.[8] Esto nos va dar el mayor momento ante la flexión para una carga en y global (suponiendo que $I_z > I_y$).

Para hallar la matriz de transformación **[T]** debemos buscar los cosenos directores de nuestra viga en el espacio. Una vez que obtenemos el versor de orientación[3] $\hat{v}_x$ es trivial obtener $\hat{v}_z$

$$\hat{v}_z = \frac{\hat{v}_x \times s_y}{||\hat{v}_x \times s_y||}$$

Luego de armar los versores $\hat{v}_x$, $\hat{v}_y$, $\hat{v}_z$ se define la matriz de los cosenos directores $\boldsymbol{\lambda}$

$$\boldsymbol{\lambda} = \begin{bmatrix} \hat{v}_x \\ \hat{v}_y \\ \hat{v}_z \end{bmatrix}_{3\times 3}$$

La matriz de transformación entonces es

$$[\mathbf{T}]_{1D} = \begin{bmatrix} \boldsymbol{\lambda} & & \cdots & 0 \\ & \boldsymbol{\lambda} & & \vdots \\ \vdots & & \boldsymbol{\lambda} & \\ 0 & \cdots & & \boldsymbol{\lambda} \end{bmatrix}_{12\times 12} \tag{2.6}$$

2.4. Cargas

Cargas Consistentes

Una carga distribuida en $-y'$ sobre una viga que está fija en sus extremos tiene las cargas nodales

$$\{\mathbf{r}'\} = \begin{Bmatrix} 0 \\ -qL/2 \\ 0 \\ 0 \\ 0 \\ -qL^2/12 \\ 0 \\ -qL/2 \\ 0 \\ 0 \\ 0 \\ qL^2/12 \end{Bmatrix} \qquad q \text{ en } \left[\frac{\text{Fuerza}}{\text{Longitud}}\right] \tag{2.7}$$

estas son las cargas ***consistentes*** porque toman en cuenta los momentos generados. Si se omiten los momentos se llaman *cargas reducidas*.

Si la viga tiene una carga distribuida sobre su eje y' local pero está arbitrariamente orientada se pueden obtener las cargas globales a aplicar sobre los nodos del elemento usando la matriz de rotación usada para orientar el elemento

$$\{\mathbf{r}\} = [\mathbf{T}]^T \{\mathbf{r}'\} \tag{2.8}$$

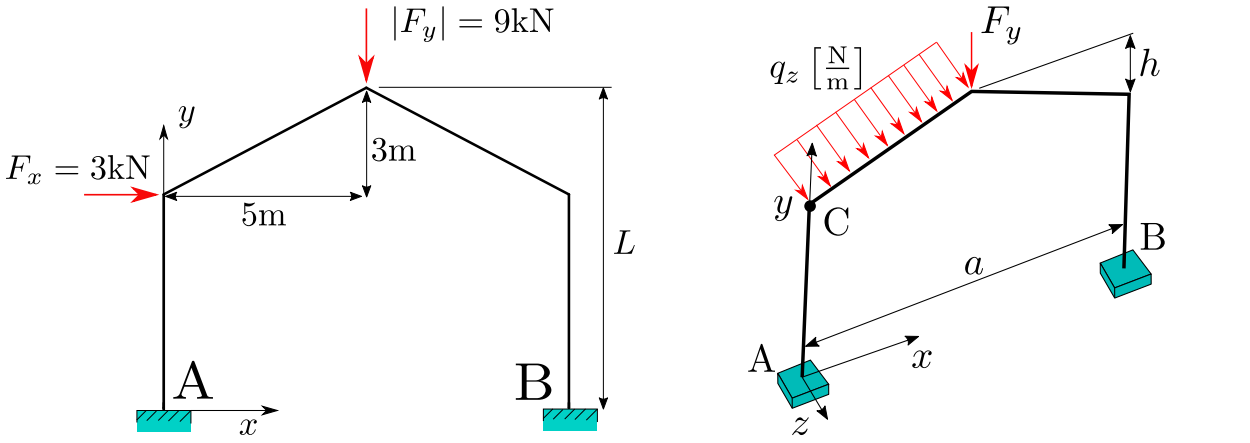
[8] **Ser cauteloso al elegir el vector s_y para que bajo ninguna circunstancia quede colineal al eje x' del elemento. Esto causará problemas insanables.**

[3] Es el que resulta de la resta entre los nodos del elemento.

usando $\{\mathbf{r}'\}$ de (2.7). Si la carga no está sobre el eje y' de la viga, se deberá usar una matriz de rotación diferente, o cambiar el vector de cargas predefinido. *¿Como aplicaría un par torsor a una viga arbitrariamente orientada?*

2.5. Ejercicios

Pórticos



(a) Pórtico de altura L a escala para el problema 1*c* (b) Pórtico con una carga distribuida en dirección z

Figura 2.4

1. Para la figura 2.4a efectuar un modelo matemático y luego discretizar para resolver:
   a) Verificar que para el caso dado con $F_x = 0$ las reacciones en y de los empotramientos son $\frac{F_y}{2}$.
   b) Se diseño el pórtico con una altura $L = 7$ metros con perfiles IPN 160. Un análisis preliminar de una consultora sugiere que el punto superior supera los desplazamientos máximos permitidos de $d_{\text{máx}} = 15\text{mm}$. Verificar.
   c) ¿Que altura debería el pórtico así el punto superior no se desplaza mas que 26.7mm? Usar perfil IPB[♭] 120. ¿Es esta la configuración de la figura 2.4a? (*hint:* si lo es)
2. Para la figura 2.4b considerar perfiles HEB 340.
   a) Las reacciones considerando $a = 12\text{m}$, $h = 2\text{m}$ y una altura de 8m. $q_z = 500\left[\frac{\text{N}}{\text{m}}\right]$ y $F_y = 20\text{kN}$.
   b) ¿Que orientación de vigas es favorable para reducir el desplazamiento del punto C?

Algunas Respuestas considerando $E = 200\text{GPa}$:

1*b*) $d = 0,0155\text{m}$, $\{\mathbf{R_A}\} \approx \{1,1;\ \ 4,1;\ \ -1,28\}$ kN o kNm

2*a*) $\{\mathbf{R_B}\} \approx \{-6,23;\ \ 10,0;\ \ -0,79;\ \ -6,35;\ \ -0,47;\ \ 16,1\}$ kN o kNm, con cargas reducidas

2*b*) $d_C = 2,38\text{mm}$

[♭]Se puede buscar también como perfil HEB.

Capítulo 3

Elementos 2D

Este capítulo trata de la formulación de elementos 2D simples en base a desplazamientos. Se explica como interpolar variables de campo sobre el elemento 2D usando las funciones de forma.

3.1. Mallado

Cosas a tener en mente antes de mallar

- **¿Donde está la sección de mi dominio que quiero estudiar**? ¿Puedo refinar en esa zona?
- ¿Qué orientación va tener mi malla para cada división de dominio?
- ¿Qué forma tendrán mis elementos con una dada división? ¿Habrá una mejor forma de dividir mi dominio así no tengo jacobianos singulares?

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 térmica/fuerza.

3.2. Funciones de formas para elementos 2D

Las funciones de forma caracterizan un campo sobre un elemento y permiten interpolar valores dentro de este mismo. Esto nos permite integrar campos sobre ele elemento para la resolución del problema de elementos finitos.

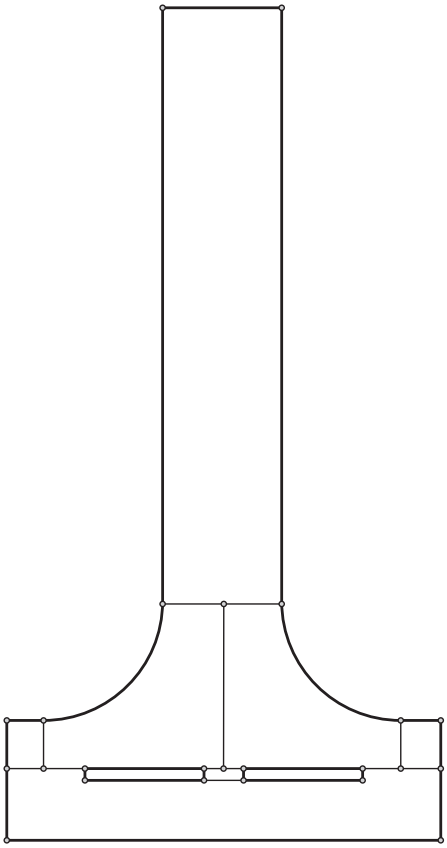
Cálculo

Para calcular las funciones de forma primero se define cuantos nodos se va tener por elemento y se los ubica en el espacio (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 $\mathbf{[N]} = [N_1 \quad N_2 \quad \dots \quad N_n]$ y $X = [1 \quad x \quad y \quad \dots \quad x^{k-1}y^k \quad x^k y^k]^T$, 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.

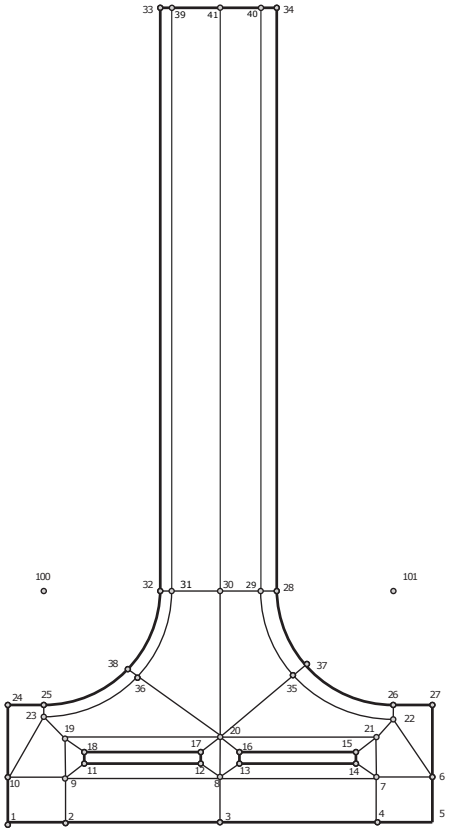
$$A = \begin{bmatrix} 1 & x_1 & y_1 & \dots & x_1^{k-1}y_1^k & x_1^k y_1^k \\ 1 & x_2 & y_2 & \dots & x_2^{k-1}y_2^k & x_2^k y_2^k \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 1 & x_n & y_n & \dots & x_n^{k-1}y_n^k & x_n^k y_n^k \end{bmatrix}$$

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

Se habla de interpolación de campo en la sección de elementos isoparamétricos.



(a) Mallable con triángulos solamente.



(b) Se puede mallar con elementos cuadrilateros. No hay esquinas angulosas. Es posible obtener *una buena malla*. Los puntos 100 y 101 son auxiliares y no forman parte de la malla.

Figura 3.1: Dos formas de dividir una paleta de una turbina pelton que recibe un chorro incidente en su esquina superior derecha.

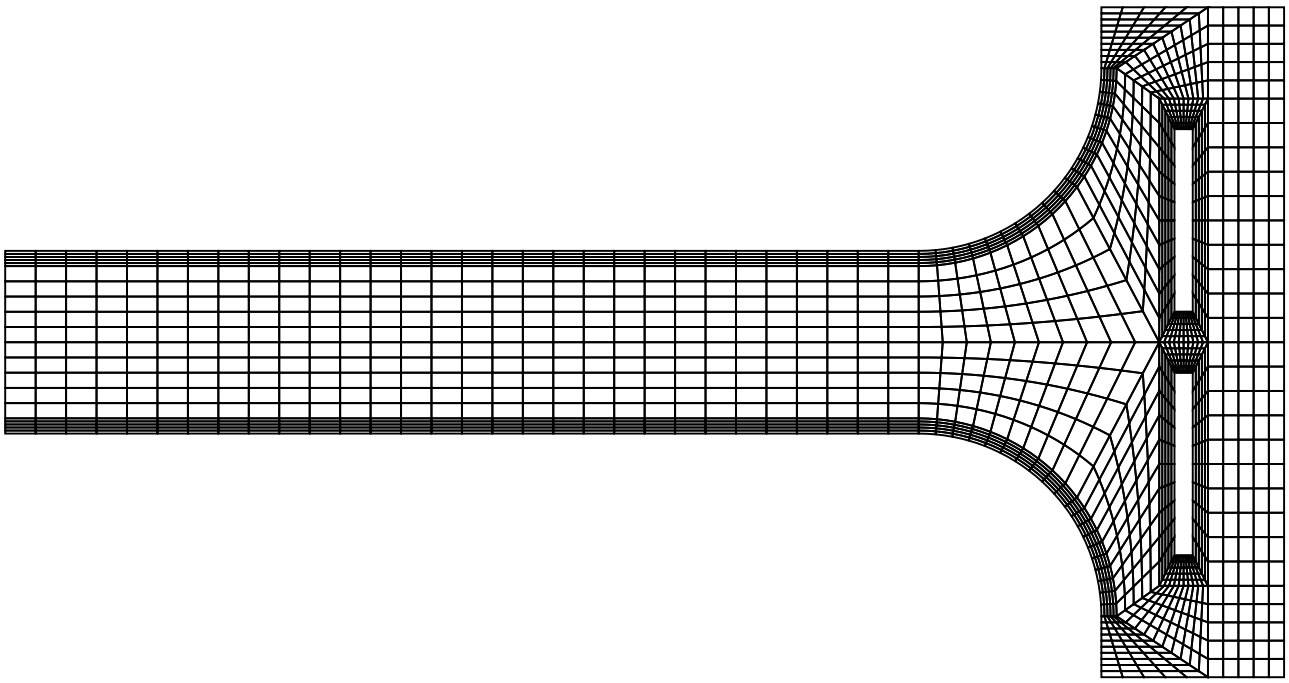


Figura 3.2: Ejemplo de malla usando divisiones de 3.1b. Se refina en los costados donde se tiene flexión.

Cargas 2-D

La ecuación que rige como se cargan elementos, siendo **{r}** las cargas nodales del elemento, **{F}** fuerzas volumétricas, **{Φ}** fuerzas de tracción superficiales, $\{\boldsymbol{\varepsilon_0}\}$ las deformaciones iniciales y $\{\boldsymbol{\sigma_0}\}$ las tensiones

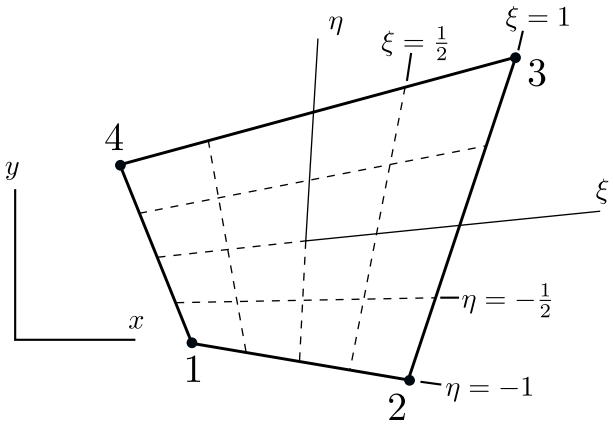


Figura 3.3: Elemento Q4 en el espacio físico (x, y). Sistema de referencia auxiliar (ξ, η) visible.

iniciales (pg. 228)

$$\{\mathbf{r}\} = \overbrace{\int [\mathbf{N}]^T \{\mathbf{F}\} \mathrm{d}V}^{\text{Volumétricas}} + \overbrace{\int [\mathbf{N}]^T \{\Phi\} \mathrm{d}S}^{\text{Superficiales}} + \int [\mathbf{B}]^T [\mathbf{E}] \{\boldsymbol{\varepsilon_0}\} \mathrm{d}V - \int [\mathbf{B}] \{\boldsymbol{\sigma_0}\} \mathrm{d}V \tag{3.1}$$

3.3. Elementos isoparamétricos

La formulación isoparamétrica permite interpolar campos sobre elementos cuadriláteros no rectangulares usando las mismas funciones de forma.

Se utiliza un sistema de coordenadas auxiliar (ξ, η) en dos dimensiones y (ξ, η, ζ) en tres dimensiones. Para integrar se utiliza un método numérico llamado *Cuadratura de Gauss*.

- Un elemento que no esta distorsionado (sigue siendo un rectangulo o paralelogramo) tiene **J** constante
- Cuidado con modo espurio. Ver tabla 6.8-1 pg. 226 el tema de full/reduced integration Cook et al. [2007]
- Como cargar tu elemento isoparamétrico en pg. 228

Interpolación de campo

Para interpolar el campo de interés (desplazamientos, coordenadas, tensión, etc.) es necesario conocer lo que vale el campo en los nodos del elemento.

Para interpolar el valor desconocido del campo ϕ sobre el punto (x_d, y_d) se suma el producto de la función de forma evaluada en (x_d, y_d) multiplicada por el campo en el nodo correspondiente

$$\phi(x_d, y_d) = \sum_{i}^{n_{\text{nod.}}} N_i(x_d, y_d) \cdot \phi(x_i, y_i)$$

Por ende, si se interpola sobre un nodo se debería obtener el valor del campo en el mismo nodo.

Integración usando cuadratura

Para integrar un campo dentro de un elemento isoparamétrico se emplea el método de la cuadratura de Gauss. Se precisa integrar para obtener la matriz de rigidez de un sistema, las tensiones a partir de los desplazamientos y para obtener las cargas.

Para integrar se necesita conocer cuanto vale el campo sobre los denominados *puntos Gauss* y multiplicar el valor por los pesos W asociados al punto Gauss. Se suman estos valores para obtener la aproximación de la integral.

$$I = \int_{-1}^{1} \int_{-1}^{1} \phi(\xi, \eta) \, \mathrm{d}\xi \, \mathrm{d}\eta \approx \sum_i \sum_j W_i W_j \cdot \phi_{ij}(\xi, \eta) \cdot |\mathbf{J}| \tag{3.2}$$

donde $|\mathbf{J}|$ es el determinante del jacobiano evaluado en el punto Gauss. Los pesos y las posiciones de los puntos Gauss se pueden calcular usando la función `gauss.m` del anexo (código 8.5).

La matriz jacobiana, o simplemente *jacobiano*, es un factor de escala que relaciona el incremento de longitud por la transformación $(\xi,\eta)\rightarrow(x,y)$. Para la transformación de coordenadas Cartesianas a polares $\mathrm{d}x\ \mathrm{d}y = r\cdot \mathrm{d}r\ \mathrm{d}\theta$ el determinante es $|\mathbf{J}| = r$.

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

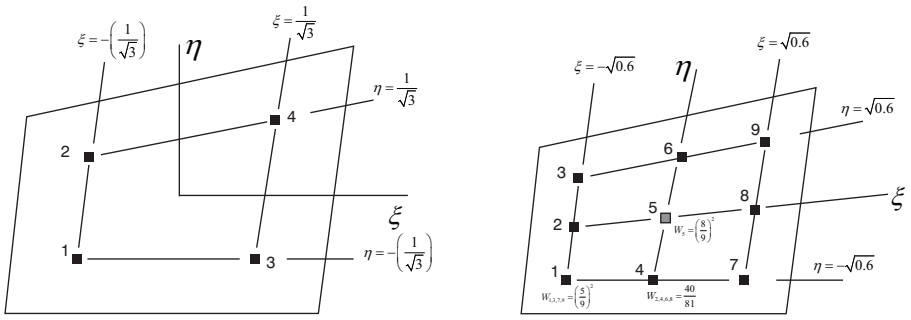


Figura 3.4: Puntos Gauss para reglas 2×2 y 3×3. El peso vale 1 en todos los puntos Gauss para la regla 2×2.

Los elementos triangulares y tetraedricos son un caso particular. El código 8.5 no está generalizado para estos casos. En lugar de una función `gausstri` se ofrece al usuario las tablas 3.1 y 4.2.

Nro. Puntos	Orden	Coordenadas	Pesos
1	0	$\left(\frac{1}{3},\frac{1}{3}\right)$	1
3	1	$\left(\frac{2}{3},\frac{1}{6}\right),\left(\frac{1}{6},\frac{1}{6}\right),\left(\frac{1}{6},\frac{2}{3}\right)$	$\frac{1}{3}$
4	2	$\left(\frac{1}{3},\frac{1}{3}\right)$	$-\frac{27}{48}$
		$\left(\frac{3}{5},\frac{1}{5}\right),\left(\frac{1}{5},\frac{1}{5}\right),\left(\frac{1}{5},\frac{3}{5}\right)$	$\frac{25}{48}$

Tabla 3.1: Valores para la integración numérica de triangulos usando cuadratura de Gauss.

Ejemplo elemento exótico

Matriz de Rigidez

Imaginemos un elementos Q5 (figura 3.5) cuadrado de dimensiones 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 3.5:

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

Llegado a este punto nos interesa obtener la matriz de rigidez. Si queremos lograr *"full integration"* deberíamos usar Gauss orden $n=3$ según (3.3)

$$2n-1 \geq O\left([\mathbf{B}]^T[\mathbf{E}][\mathbf{B}]\right) \tag{3.3}$$

[B] es el *strain-deformation matrix*. El producto $[\mathbf{B}]^T[\mathbf{E}][\mathbf{B}]$ 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 $n=2$.

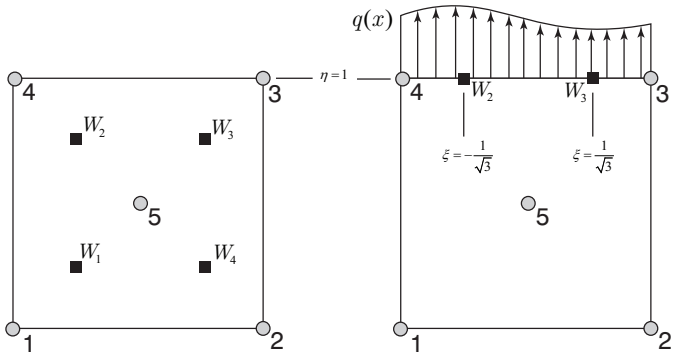


Figura 3.5: Elemento Q5 rectangular.

La rigidez de un elemento está dada por

$$[\mathbf{k}] = \int [\mathbf{B}]^T [\mathbf{E}][\mathbf{B}] \mathrm{d}V \tag{3.4}$$

para un elemento plano la ecuación anterior es

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

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

Carga de linea

Si el elemento está cargado sobre la linea 4-3 con una distribuida $q(x)$ (en [N/m]) entonces procedemos de la siguiente manera según el segundo término de (3.1):

$$r_{xi} = \int_{-1}^{1} N_i (\tau \mathbf{J}_{11} - \sigma \mathbf{J}_{12}) \cdot t \; \mathrm{d}\xi \tag{3.5}$$

$$r_{yi} = \int_{-1}^{1} N_i (\sigma \mathbf{J}_{11} + \tau \mathbf{J}_{12}) \cdot t \; \mathrm{d}\xi \tag{3.6}$$

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 3.5, 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!

Tensiones

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

3.4. Elementos Axisimetricos

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.

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

Capítulo 4

Elementos 3D

Para elementos tridimensionales se van a tener tres desplazamientos por nodo u, v, w. La matriz constitutiva para sólidos 3D se calcula según (4.1)

$$[\mathbf{E}] = \begin{bmatrix} 2G+\lambda & \lambda & \lambda & 0 & 0 & 0 \\ \lambda & 2G+\lambda & \lambda & 0 & 0 & 0 \\ \lambda & \lambda & 2G+\lambda & 0 & 0 & 0 \\ 0 & 0 & 0 & G & 0 & 0 \\ 0 & 0 & 0 & 0 & G & 0 \\ 0 & 0 & 0 & 0 & 0 & G \end{bmatrix} \qquad [\mathbf{E}]^{-1} = \frac{1}{E}\begin{bmatrix} 1 & -\nu & -\nu & 0 & 0 & 0 \\ -\nu & 1 & -\nu & 0 & 0 & 0 \\ -\nu & -\nu & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & f & 0 & 0 \\ 0 & 0 & 0 & 0 & f & 0 \\ 0 & 0 & 0 & 0 & 0 & f \end{bmatrix} \tag{4.1}$$

donde

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

Un esquema general para calcular la matriz de rigidez de un sistema compuesto por elementos 3D es el siguiente. Tenga en cuenta que falta la obtención de los puntos Gauss.

Código 4.1: Programa generalizado para obtener [**K**] para elementos 3D.

```
%% Dofinitions
Ndofpornod = 3; Ndn = Ndofpornod;
n2d = @(n) repmat(n*Ndn,Ndn,1) - (Ndn-1:-1:0)'; % forma generalizada
[Nnod, Ndim] = size(nodos);
[Nelem, Nnodporelem] = size(elementos);
Ndofporelem  =  Ndofpornod*Nnodporelem;
dof = Ndofpornod*Nnod;
DOF = reshape(1:dof,Ndofpornod,[])';
%% Integracion
K = sparse(dof,dof);
for e = 1:Nelem
    ke = zeros(Ndofporelem,Ndofporelem);
    meindof = reshape(DOF(elementos(e,:),:)',1,[]);
    elenod = nodos(elementos(e,:),:);
    for ipg = 1:pg.n
        r = pg.u(ipg,1); s = pg.u(ipg,2); t = pg.u(ipg,3);
        Ns = eval(N);
        dNs = eval(dN);
        jac = dNs*elenod;
        Djac = det(jac)/factor; % Ver tabla 4.1
        dNxyz = jac\dNs;
        B=zeros(size(E,2),Ndofpornod * Nnodporelem);
        B(1,1:3:end-2) = dNxyz(1,:); %dx
        B(2,2:3:end-1) = dNxyz(2,:); %dy
        B(3,3:3:end)   = dNxyz(3,:); %dz VER PAGINA 80 Cook. Ec (3.1-9)
        B(4,1:3:end-2) = dNxyz(2,:); %dy
```

```
27          B(4,2:3:end-1) = dNxyz(1,:); %dx
28          B(5,2:3:end-1) = dNxyz(3,:); %dz
29          B(5,3:3:end)   = dNxyz(2,:); %dy
30          B(6,1:3:end-2) = dNxyz(3,:); %dz
31          B(6,3:3:end)   = dNxyz(1,:); %dx
32          ke = ke + B'*E*B*pg.w(ipg)*Djac;
33      end
34      K(meindof,meindof) = K(meindof,meindof) + ke;
35  end
```

donde la variable `factor` es aplicada según la geometría del elemento. Para elementos rectangulares o hexaedros el valor del factor es 1.

Elemento	Factor
Triángulos	2
Tetraedros	6
Pirámides	3
Wedges	2

Tabla 4.1: Factor que divide el determinante del jacobiano para varios elementos. Ver linea 20 del código 4.1

4.1. Formulación elemento hexaedro lineal (H8)

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

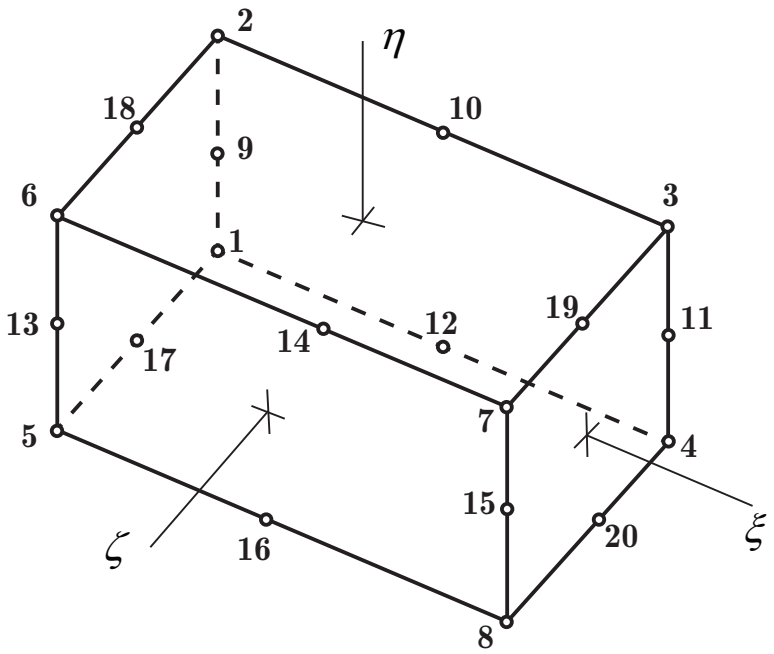


Figura 4.1: Numeración de nodos H20 en ADINA(Ejes sugeridos)

Cada nodo tendrá 3 grados de libertad, dándonos 24 dof por elemento. El elemento H20 de la figura 4.1 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

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

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

$$\mathbf{J} = \begin{bmatrix} \frac{\partial}{\partial \xi}[\mathbf{N}] \\ \frac{\partial}{\partial \eta}[\mathbf{N}] \\ \frac{\partial}{\partial \zeta}[\mathbf{N}] \end{bmatrix} \cdot \begin{bmatrix} x_1 & y_1 & z_1 \\ x_2 & y_2 & z_2 \\ x_3 & y_3 & z_3 \\ x_4 & y_4 & z_4 \\ x_5 & y_5 & z_5 \\ x_6 & y_6 & z_6 \\ x_7 & y_7 & z_7 \\ x_8 & y_8 & z_8 \end{bmatrix} \tag{4.2}$$

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

$$[\partial \mathbf{N}] = \begin{bmatrix} \frac{\partial}{\partial x}[\mathbf{N}] \\ \frac{\partial}{\partial y}[\mathbf{N}] \\ \frac{\partial}{\partial z}[\mathbf{N}] \end{bmatrix} = \mathbf{J}^{-1} \begin{bmatrix} \frac{\partial}{\partial \xi}[\mathbf{N}] \\ \frac{\partial}{\partial \eta}[\mathbf{N}] \\ \frac{\partial}{\partial \zeta}[\mathbf{N}] \end{bmatrix} \tag{4.3}$$

con lo obtenido se puede calcular la matriz [**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

$$B_i = \begin{bmatrix} \partial N_i/\partial x & 0 & 0 \\ 0 & \partial N_i/\partial y & 0 \\ 0 & 0 & \partial N_i/\partial z \\ 0 & \partial N_i/\partial z & \partial N_i/\partial y \\ \partial N_i/\partial z & 0 & \partial N_i/\partial x \\ \partial N_i/\partial y & \partial N_i/\partial x & 0 \end{bmatrix} \tag{4.4}$$

Finalmente un calcula la rigidez del elemento usando (3.4). Usando 8 puntos gauss se podría efectuar una integración *full* del elemento. Usando la función 8.5:

```
pg = gauss([2 2 2]); % 8 pg para H8
```

4.2. Elemento tetraedro lineal (T4)

Se espera que el lector ya haya programado elementos 2D y que tenga experiencia obteniendo las funciones de formas de elementos diversos. Esto dicho, el salto de elementos 2D a elementos 3D puede presentar dificultades debido a la falta de bibliografía acerca del tema, o la omisión de la aplicación de teoría por parte de los autores.

A continuación se dejan rutinas de MATLAB para que el usuario pueda rápidamente empezar a experimentar con tetraedros lineales. Se empieza procesando una malla de tetraedros cuadráticos de diez nodos para que se pueda resolver con un programa para tetraedros de cuatro nodos.

Código 4.2: Descarga y procesado de malla T10 para uso con rutinas para elementos T4.

```
%% Definir Dominio. Nodos + elementos
fprintf(fopen('nod.txt','wt'),webread('https://pastebin.com/raw/0BLPxUuu'));
fprintf(fopen('ele.txt','wt'),webread('https://pastebin.com/raw/hfcg7wc4'));
fclose('all');
```

```
nodosT10 = load('nod.txt');
elementosT10 = load('ele.txt'); % La malla es para elementos T10
%% Post process para convertir malla T10 a T4
elementos = elementosT10(:,2:5); % La primera columna es indice
losNodosT4 = intersect(nodosT10(:,1),elementos(:));
nodos = nodosT10(ismember(nodosT10(:,1),losNodosT4),1:4); %columna1=indice
for en = 1:numel(elementos)
    elementos(en) = find(elementos(en)==nodos(:,1));
end
nodos = nodos(:,2:4); % eliminamos columna indice
```

El próximo paso es definir el material en la matriz **[E]** según la ecuación (4.1). Antes de integrar usando la rutina 4.1 se tienen que obtener las funciones de forma (Código 4.3) y definir la integración numérica según la tabla 4.2

Código 4.3: Obtención de las funciones de forma para elementos T4.

```
%% FForma T4
syms r s t
unod = [0  0  0
        1  0  0
        0  1  0
        0  0  1];
X = [1 r s t];
A = nan(length(X));
k=0;
for inod = unod'
   k = k + 1;
   r = inod(1); s = inod(2); t = inod(3);
   A(k,:) = subs(X);
end
syms r s t
X = subs(X);
N = X/A;
dN = [diff(N,r); diff(N,s); diff(N,t)];
```

Nro. Puntos	Orden	Coordenadas	Pesos
1	0	$\left(\frac{1}{4},\frac{1}{4},\frac{1}{4}\right)$	1
4	1	$(a,b,b),(b,b,b),(b,b,a),(b,a,b)$	$\frac{1}{4}$
		donde $a=\frac{5+3\sqrt{5}}{20}$, $b=\frac{5-3\sqrt{5}}{20}$	
5	2	$\left(\frac{1}{4},\frac{1}{4},\frac{1}{4}\right)$	$-\frac{4}{5}$
		$\left(\frac{1}{2},\frac{1}{6},\frac{1}{6}\right),\left(\frac{1}{6},\frac{1}{6},\frac{1}{6}\right),\left(\frac{1}{6},\frac{1}{6},\frac{1}{2}\right),\left(\frac{1}{6},\frac{1}{2},\frac{1}{6}\right)$	$\frac{9}{20}$

Tabla 4.2: Valores para la integración numérica de tetraedros usando cuadratura de Gauss.

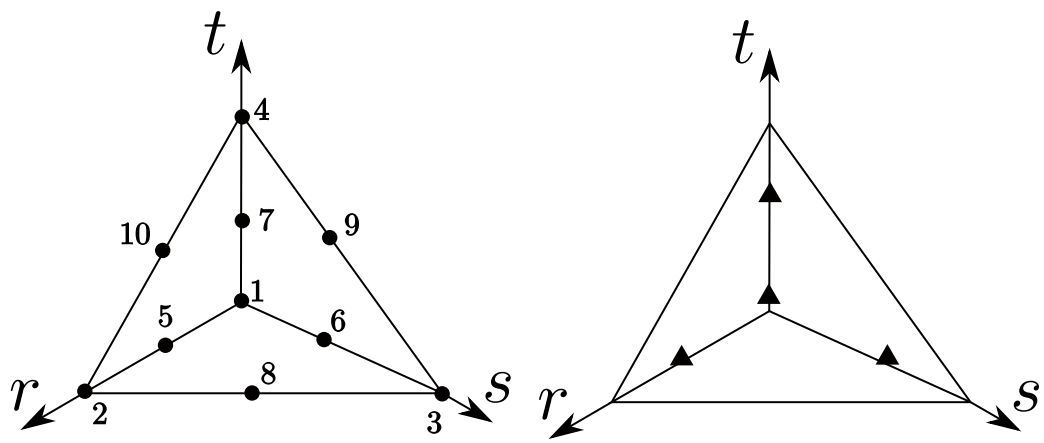


Figura 4.2: Numeración de un tetraedro de diez nodos y la ubicación de los puntos Gauss para un grado de precisión 2 (orden 1).

Capítulo 5

Otros temas

5.1. Desplazamientos iniciales

Si nuestro sistema tiene desplazamientos iniciales conocidos se puede formular un sistema a resolver:

$$\begin{bmatrix} [\mathbf{K}_{xx}] & [\mathbf{K}_{xc}] \\ [\mathbf{K}_{cx}] & [\mathbf{K}_{cc}] \end{bmatrix} \begin{Bmatrix} \{\mathbf{D}_x\} \\ \{\mathbf{D}_c\} \end{Bmatrix} = \begin{Bmatrix} \{\mathbf{R}_c\} \\ \{\mathbf{R}_x\} \end{Bmatrix} \tag{5.1}$$

Note que para los nodos donde se conocen los desplazamientos (los nodos c) *se desconocen las cargas*, por ende las cargas sobre los nodos con desplazamientos conocidos tienen subíndice x. La rigidez del sistema **[K]** es conocida; se usan los subindices x y c solo para tener referencia. El sistema expandido tiene la forma

$$[\mathbf{K}_{xx}]\{\mathbf{D}_x\} + [\mathbf{K}_{xc}]\{\mathbf{D}_c\} = \{\mathbf{R}_c\}$$
$$[\mathbf{K}_{cx}]\{\mathbf{D}_x\} + [\mathbf{K}_{cc}]\{\mathbf{D}_c\} = \{\mathbf{R}_x\}$$

La matriz $[\mathbf{K}_{xx}]$ no es singular si se impusieron suficientes desplazamientos como para prevenir movimiento de cuerpo rígido/mecanismos. Se pueden entonces obtener los desplazamientos desconocidos

$$\{\mathbf{D}_x\} = [\mathbf{K}_{xx}]^{-1}(\{\mathbf{R}_c\} - [\mathbf{K}_{xc}]\{\mathbf{D}_c\})$$

luego de obtener los desplazamientos desconocidas se puede obtener las cargas desconocidas $\{\mathbf{R}_x\}$.

5.2. Restricciones

Las restricciones sirven para imponer una nueva relación a los dof del sistema ya existente $K\,d = F$. Las condiciones de borde esenciales son restricciones de un punto. Se pueden tener restricciones *multi punto* al relacionar los dof entre si como es el caso para rigid links.

Multiplicadores de Lagrange

$$\begin{bmatrix} \mathbf{K} & \mathbf{C}^T \\ \mathbf{C} & 0 \end{bmatrix} \begin{Bmatrix} \mathbf{D} \\ \boldsymbol{\lambda} \end{Bmatrix} = \begin{Bmatrix} \mathbf{R} \\ \mathbf{Q} \end{Bmatrix} \tag{5.2}$$

El sistema de desconocidas se vuelve los desplazamientos **{D}** y un vector de multiplicadores de Lagrange $\boldsymbol{\lambda}$. **Q** es el lado derecho de la ecuación de restricciones. La ventaja de este método es su resultado exacto.

Método del penal

$$(\mathbf{K} + \alpha \mathbf{C}^T \mathbf{C})\mathbf{D} = \mathbf{R} + \alpha \mathbf{C}^T \mathbf{Q} \tag{5.3}$$

α es una constante de magnitud relativamente grande en comparación al máximo de la diagonal de **K**. Lo que se hace es agregar un valor alto a matriz de rigidez y una fuerza correspondiente para cumplir la restricción.

La gran ventaja de este método es que se conserva el tamaño del sistema a resolver resultando en tiempos de resolución cortos para sistemas con muchas restricciones. Las desventajas también son importantes tomar en cuenta: Para un resultado más exacto es necesario agrandar α. El usuario también tiene que tener cuidado que si se aumenta un α que multiplica un elemento no-diagonal entonces pueden llegar a haber problemas numéricos de acondicionamiento.

Armado de C

Si se quiere restringir dos dof con una relación del tipo $u_n = u_m$ entonces la ecuación es

$$[\mathbf{c}]^T_{N_{\text{dof}}\times 1} = \begin{bmatrix} 0 \\ \vdots \\ 1 \\ \vdots \\ -1 \\ \vdots \\ 0 \end{bmatrix} \begin{matrix} u_1 \\ \vdots \\ u_m \\ \vdots \\ u_s \\ \vdots \\ u_{N_{\text{nod.}}} \end{matrix}, \qquad \mathbf{q} = 0 \qquad \Rightarrow \quad u_m - u_s = \mathbf{q} = 0$$

donde el nodo m es el **master** y el nodo s es el **slave**. Cuando se tengan varias restricciones unidas a un nodo, este nodo será el **master** y se tendrán que plantear las ecuaciones adecuadamente para que sea así.

Algunas reglas generales para guiarse con elementos rígidos:

1. Un nodo slave restringido tiene solo un master para que hay a una solución definida
2. Un nodo no puede ser slave y master a la vez

Rigid Bar en el plano

Un rigid bar o un rigid beam (son restricciones inherentemente distintas) cumple la función de unir dos nodos con un elemento 1D totalmente rígido. Aunque se podría hacer con un elemento barra con rigidez alterada artificialmente, esto puede traer problemas numéricos si la rigidez alterada es muy alta. Además, para el estudio de dinámica estructural y vibraciones es indeseable tener elementos con rigidez artificialmente alta pues agregan modos de alta frecuencia. Estos modos después hacen que el análisis dinámico transitorio sea imposible de efectuar. Para estos casos se emplean los multiplicadores de Lagrange.

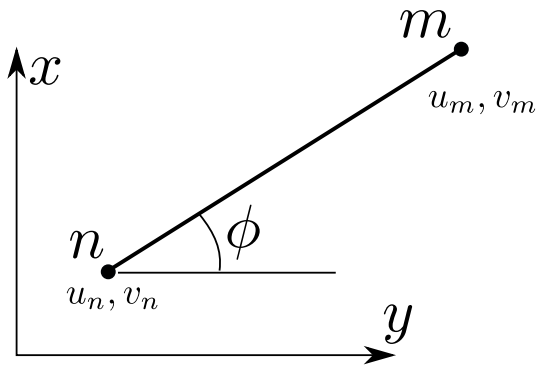


Figura 5.1: Un rigid bar que une el nodo n con el nodo m.

El motivo del rigid bar es un elemento que *no se acorta ni se alarga* con las fuerzas que transmite. Un usuario del método de elementos finitos podría pensar que alcanza con plantear dos ecuaciones $u_n = u_m; v_n = v_m$ sin embargo esto lograría que los nodos n y m se trasladen como un cuerpo rígido sin permitir rotaciones, efectivamente logrando una rigidez más alta que la propuesta en el motivo del rigid bar.

Si el rigid bar de la figura 5.1 permite rotaciones sin acortarse, entonces la distancia que se mueve el nodo m sobre el eje local x' del rigid bar debe ser igual al del nodo n, dando una única ecuación

$$u_n \cos\phi + v_n \sin\phi = u_m \cos\phi + v_m \sin\phi$$

esta restricción se puede agregar al sistema usando los multiplicadores de Lagrange o con el método del penal.

Rigid beam en el plano

En el análisis ingenieril por FEA es común el uso de restricciones para modelar roscas, unir elementos con dof disimiles, etcætera. Para el segundo caso se puede usar un *rigid link* (rigid beam) para transmitir rotaciones de un elemento/nodo con giro a un elemento sin giros en sus nodos, como podría ser el caso de unión viga-elemento 3D.

La formulación de un rigid link consiste en plantear ecuaciones de cinemática para una viga rígida (considerando desplazamientos pequeños). Los desplazamientos cartesianos del nodo slave serán condicionados por los giros del nodo master

$$\begin{aligned} u_s &= u_m + L_e\,\hat{v}_y\,\theta_m \\ v_s &= v_m - L_e\,\hat{v}_x\,\theta_m \\ \theta_s &= \theta_m \end{aligned}$$

donde L_e es la distancia entre nodo slave y master.

La formulación para un rigid link en el espacio puede ser deducida de la formulación RBE2 en el anexo de este documento planteando desplazamientos pequeños.

5.3. Error

Para calcular error ZZ primero se obtiene un campo de tensiones que "mejor aproxima la solucion" llamado el **smoothed field** o campo suavizado (σ^*). Hay varios metodos para obtener σ^*, en este documento se va tratar el método de suavizado por elemento.

Element smoothing

Se obtienen las tensiones sobre los puntos gauss superconvergentes mediante la ecuacion

$$\{\boldsymbol{\sigma}^*\}_i = [\mathbf{E}][\mathbf{B}^*]\{\mathbf{d}\}$$

donde $[\mathbf{B}^*]$ es evaluada sobre dichos puntos de gauss. Esto nos da el campo suavizado σ^* del elemento.

Error ZZ

Para poder integrar las siguientes expresiones se tendrá que obtener $\{\boldsymbol{\sigma}\}_i$ sobre los puntos de cuadratura, es decir, habrá que interpolar el campo no-suavizado sobre los puntos gauss. En general no son idénticas las dos tensiones. Recordar que no tiene sentido obtener η sobre un dominio que presenta discontinuidades de tensiones inherentes, como cambios abruptos de sección y cambio de material.

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

donde m es el numero de elementos del dominio donde se quiere calcular η.

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

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

El error relativo (η) esta comprendido entre 1 y 0. Un valor aceptable de η suele ser tomado como $\eta \leq 0,05$ [Cook et al., 2007].

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

5.4. 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 complicado 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.

Para aplicar convección

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

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

Resolución de problemas típico

Condiciones de borde de temperatura. Idéntico a lo visto en la sección 5.1.

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

luego de obtener las temperaturas desconocidas se puede obtener el flujo desconocido (sobre los nodos conocidos):

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

Capítulo 6

No-linealidad y análisis dinámico

6.1. Respuesta dinámica estructural

$$[\mathbf{M}]\{\ddot{\mathbf{D}}\}+[\mathbf{C}]\{\dot{\mathbf{D}}\}+[\mathbf{K}]\{\mathbf{D}\}=\{\mathbf{R}^{\text{ext}}\} \tag{6.1}$$

Se puede resolver la ecuación de arriba para un sistema dado sin amortiguamiento y sin cargas externas* para obtener sus frecuencias naturales y los modos asociados a estos.

$$\{\mathbf{D}\}=\{\bar{\mathbf{D}}\}\sin\omega t, \qquad \{\ddot{\mathbf{D}}\}=-\omega^2\{\bar{\mathbf{D}}\}\sin\omega t$$

sin considerar la matriz de amortiguamiento se obtiene el problema de autovalores:

$$\big([\mathbf{K}]-\omega^2[\mathbf{M}]\big)\{\bar{\mathbf{D}}\}=\{\mathbf{0}\}$$

donde ω^2 es un autovalor y la raíz de estos son las frecuencias naturales.

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

$$[\mathbf{C}_\Phi]=\begin{bmatrix} 2\omega_n\varsigma_n & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & 2\omega_1\varsigma_1 \end{bmatrix} \tag{6.2}$$

Amortiguamiento proporcional. Se basa el análisis

$$[\mathbf{C}_\Phi]=[\Phi]^T(\alpha[\mathbf{M}]+\beta[\mathbf{K}])[\Phi]=\alpha\delta[\mathbf{I}]+\beta[\mathbf{\Omega^2}] \tag{6.3}$$

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 6.1 y por ende se pasa a tratar dichas matrices diagonales como vectores columnas. Una vez desacopladas se tiene

$$\{\ddot{\mathbf{Z}}\}+2\{\mathbf{\Omega}\}\{\mathbf{C}_\Phi\}\{\dot{\mathbf{Z}}\}+\{\mathbf{\Omega^2}\}\{\mathbf{Z}\}=\{\mathbf{R}_\Phi\}$$

$$\{\mathbf{Z}\}=\frac{\{\mathbf{R}_\Phi\}}{\{\mathbf{\Omega^2}\}\sqrt{(1-\chi^2)^2+(2\{\mathbf{C}_\Phi\}\chi)^2}}$$

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

*Se denomina vibración "libre" cuando no hay cargas asociadas. Si no hay amortiguamiento el desplazamiento es regido por $u=\bar{u}\sin\omega t$, donde $\bar{u}$ es la amplitud de vibración y ω es la frecuencia circular Cook et al. [2007]. ω es obtenida en radianes por segundo.

Sine Sweep

A medida que la frecuencia de excitación aumenta la *amplitud del sistema disminuye*[†]. 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.

Matriz de masa consistente para una viga 3D

$$[\mathbf{m}]_{1D} = \int_0^L [\mathbf{N}]^T [\mathbf{N}] \rho A \mathrm{d}x \tag{6.4}$$

La matriz de masa según una fuente desconocida

$$[\mathbf{m}']_{1D} = \frac{\rho A L}{420} \begin{bmatrix} 140 & 0 & 0 & 0 & 0 & 0 & 35 & 0 & 0 & 0 & 0 & 0 \\ 0 & 156 & 0 & 0 & 0 & 22L & 0 & 27 & 0 & 0 & 0 & -13L \\ 0 & 0 & 156 & 0 & -22L & 0 & 0 & 0 & 27 & 0 & 13L & 0 \\ 0 & 0 & 0 & 140\,{r_x}^2 & 0 & 0 & 0 & 0 & 0 & -35\,{r_x}^2 & 0 & 0 \\ 0 & 0 & -22L & 0 & 16L^2 & 0 & 0 & 0 & -13L & 0 & -6L^2 & 0 \\ 0 & 22L & 0 & 0 & 0 & 16L^2 & 0 & 13L & 0 & 0 & 0 & -6L^2 \\ 35 & 0 & 0 & 0 & 0 & 0 & 140 & 0 & 0 & 0 & 0 & 0 \\ 0 & 27 & 0 & 0 & 0 & 13L & 0 & 156 & 0 & 0 & 0 & -22L \\ 0 & 0 & 27 & 0 & -13L & 0 & 0 & 0 & 156 & 0 & 22L & 0 \\ 0 & 0 & 0 & -35\,{r_x}^2 & 0 & 0 & 0 & 0 & 0 & 140\,{r_x}^2 & 0 & 0 \\ 0 & 0 & 13L & 0 & -6L^2 & 0 & 0 & 0 & 22L & 0 & 16L^2 & 0 \\ 0 & -13L & 0 & 0 & 0 & -6L^2 & 0 & -22L & 0 & 0 & 0 & 16L^2 \end{bmatrix} \tag{6.5}$$

donde $r_x = \sqrt{\frac{I_z}{A}}$.

La matriz de masa según un paper escrito por Matas Hidalgo [2014]

$$\boldsymbol{m_{11}} = \begin{bmatrix} \frac{1}{3} & 0 & 0 & 0 & 0 & 0 \\ 0 & \frac{13}{35} & 0 & 0 & 0 & \frac{11L}{210} \\ 0 & 0 & \frac{13}{35} & 0 & \frac{-11L}{210} & 0 \\ 0 & 0 & 0 & \frac{I_y+I_z}{3A} & 0 & 0 \\ 0 & 0 & \frac{-11L}{210} & 0 & \frac{L^2}{105} & 0 \\ 0 & \frac{11L}{210} & 0 & 0 & 0 & \frac{L^2}{105} \end{bmatrix}, \quad \boldsymbol{m_{12}} = \begin{bmatrix} \frac{1}{6} & 0 & 0 & 0 & 0 & 0 \\ 0 & \frac{9}{70} & 0 & 0 & 0 & \frac{-13L}{420} \\ 0 & 0 & \frac{9}{70} & 0 & \frac{-13L}{420} & 0 \\ 0 & 0 & 0 & \frac{I_y+I_z}{6A} & 0 & 0 \\ 0 & 0 & \frac{-13L}{420} & 0 & \frac{-L^2}{140} & 0 \\ 0 & \frac{13L}{420} & 0 & 0 & 0 & \frac{-L^2}{140} \end{bmatrix}$$

$$\boldsymbol{m_{21}} = \begin{bmatrix} \frac{1}{6} & 0 & 0 & 0 & 0 & 0 \\ 0 & \frac{9}{70} & 0 & 0 & 0 & \frac{13L}{420} \\ 0 & 0 & \frac{9}{70} & 0 & \frac{-13L}{420} & 0 \\ 0 & 0 & 0 & \frac{I_y+I_z}{6A} & 0 & 0 \\ 0 & 0 & \frac{13L}{420} & 0 & \frac{-L^2}{140} & 0 \\ 0 & \frac{-13L}{420} & 0 & 0 & 0 & \frac{-L^2}{140} \end{bmatrix}, \quad \boldsymbol{m_{22}} = \begin{bmatrix} \frac{1}{3} & 0 & 0 & 0 & 0 & 0 \\ 0 & \frac{13}{35} & 0 & 0 & 0 & \frac{-11L}{210} \\ 0 & 0 & \frac{13}{35} & 0 & \frac{11L}{210} & 0 \\ 0 & 0 & 0 & \frac{I_y+I_z}{3A} & 0 & 0 \\ 0 & 0 & \frac{11L}{210} & 0 & \frac{L^2}{105} & 0 \\ 0 & \frac{-11L}{210} & 0 & 0 & 0 & \frac{L^2}{105} \end{bmatrix}$$

donde

$$[\mathbf{m}']_{1D} = \rho A L \begin{bmatrix} \boldsymbol{m_{11}} & \boldsymbol{m_{12}} \\ \boldsymbol{m_{21}} & \boldsymbol{m_{22}} \end{bmatrix} \tag{6.6}$$

[†]Excepto en cercanías de una frecuencia natural

6.2. Transferencia de calor no-lineal y transitoria

Radiación

Cuando se tienen problemas de radiación se puede iterar para obtener el perfil usando **relajación.**

$$\begin{cases} \{\mathbf{T}_x\}^{n+1}_{\text{unrelaxed}} = [\mathbf{K}_{xx}]^{-1}\left(\{\mathbf{R}_x\}^n - [\mathbf{K}_{xc}]\{\mathbf{T}_c\}^n\right) \\ \{\mathbf{R}\}^n = \{\mathbf{R}_{\text{generado}}\} + \{\mathbf{R}_{\text{rad}}\}^n \\ \{\mathbf{T}\}^{n+1} = \{\mathbf{T}\}^n + \frac{1}{k_R}\cdot\left(\{\mathbf{T}\}^{n+1}_{\text{unrelaxed}} - \{\mathbf{T}\}^n\right) \end{cases} \tag{6.7}$$

donde k_R es la relajación o factor de atenuación de temperaturas. Cuanto mayor es más "amortiguada" es la convergencia del perfil. Usando mayores k_R se puede asegurar la convergencia de la solución a costo de ser más lenta.

Transitorio

Matriz capacidad

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

Temperature-Heat flux matrix **[B]** para Q4:

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

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:

$$\mathbf{\{T\}}^{n+1} = \left(\mathbf{[C]} + \Delta t\,\beta\mathbf{[K]}\right)^{-1}\left[\left(\mathbf{[C]} - \Delta t(1-\beta)\mathbf{[K]}\right)\mathbf{\{T\}}^n + \Delta t\left((1-\beta)\mathbf{\{R\}}^n + \beta\mathbf{\{R\}}^{n+1}\right)\right] \tag{6.8}$$

$\beta = 0$	Euler Forward Difference
$\beta = 0.5$	Crank–Nicholson
$\beta = 0.666$	Galerkin
$\beta = 1$	Backward Difference

6.3. Análisis Dinámico Explicito

Método Integración Directa

Resuelve 6.1.

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

Método Runge–Kutta K4

Tengo que resolver las dos ecuaciones simultáneamente

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

donde

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

6.4. Análisis Dinámico Implícito

Método Newmark

El método Newmark consiste en plantear las ecuaciones de equilibrio en el instante de tiempo $t + \Delta t$. Es un método **implícito** que goza de ser incondicionalmente estable cuando usado en conjunto con el esquema de aceleración promediada constante propuesto por Newmark (6.12)

$$\{\mathbf{V}\}^{n+1} = \{\mathbf{V}\}^n + \left[(1-\delta)\{\dot{\mathbf{V}}\}^n + \delta\{\dot{\mathbf{V}}\}^{n+1}\right]\Delta t \tag{6.9}$$

$$\{\mathbf{U}\}^{n+1} = \{\mathbf{U}\}^n + \{\mathbf{V}\}^n\Delta t + \left[(\tfrac{1}{2}-\alpha)\{\dot{\mathbf{V}}\}^n + \alpha\{\dot{\mathbf{V}}\}^{n+1}\right]\Delta t^2 \tag{6.10}$$

$$\{\mathbf{R}\}^{n+1} = [\mathbf{M}]\{\dot{\mathbf{V}}\}^{n+1} + [\mathbf{C}]\{\mathbf{V}\}^{n+1} + [\mathbf{K}]\{\mathbf{U}\}^{n+1} \tag{6.11}$$

$$\alpha = \tfrac{1}{4}, \qquad \delta = \tfrac{1}{2} \tag{6.12}$$

Capítulo 7

Métodos avanzados para la programación

Esta sección incluye técnicas para facilitar la programación de modelos complejos (OOP) y estructuras de datos que optimizan el tiempo de corrida. Ver la tabla 1.1 para la parte de definición de variables.

7.1. OOP

La programación orientada a los objetos (Objected Oriented Programming) tiene sus ventajas al momento de tratar programas complejos. MATLAB tiene una implementada una estructura de datos llamada `struct` cuyo comportamiento es similar al de otros lenguajes OOP.

Si se piensa que una variable es un cajón donde se puede guardar una matriz o string, un struct seria un mueble con amplio espacio para varios cajones. Los cajones se llaman `fields` o campos. Crear un struct es tan fácil como agregar un punto entre dos nombres de variables:

Código 7.1: Ejemplos de structs y como acceder a sus campos.

```
mueble.cajon1 = [0 1 2];  % mi struct se llama mueble
mueble.c2 = {'Ke','Ep','k'}; % cajon1 y c2 son campos del struct
fprintf( '%d %s',mueble.cajon1(2),mueble.c2{1} )
```

Las ventajas solo resultan aparente cuando se tiene un código largo donde se tienen muchas variables. Su aplicación en los elementos finitos son numerosas, pues no presentan desventaja en el tiempo de corrida y suelen disminuir la complejidad del código.

- Mas facilidad para funciones que llaman a structs. Resulta mas fácil modificar estas dado un cambio en el código
- Usar una struct para cada tipo o grupo de elemento para almacenar su matriz de rigidez, matriz elementos etc. reduce la complejidad del código
- Guardar resultados de una corrida es tan fácil como `save('solucion.mat','structSolucion')` donde `structSolucion` puede contener los desplazamientos, tensiones, reacciones etc. Luego de guardar a archivo se puede cargar a memoria las variables guardadas usando la función `load`.

7.2. Integración pre-cargada

No es necesario calcular las funciones de forma sobre los puntos gauss o nodos cada vez que se itera sobre un elemento, se pueden guardar estas en un cell array y llamarlas en el momento. El uso de este método puede reducir por un factor mayor a 10 el tiempo de corrida. A continuación se declara una estructura `gp` que contiene toda la información relacionada a los puntos gauss, incluyendo

- `pg.n` La cantidad de puntos gauss a integrar
- `pg.w` Los pesos de los puntos gauss
- `pg.u` Las posiciones de los puntos gauss
- `pg.N` Las funciones de forma de los nodos evaluadas en los puntos gauss (un cell array)

Creación del cell array

Código 7.2: Creación de struct relacionada a los puntos de Gauss.

```
pg.N=cell(pg.n,1);
pg.dN= cell(pg.n,1);
for ipg =1:pg.n %Optimiza para problemas grandes
   xi = pg.u(ipg,1); eta = pg.u(ipg,2);
   pg.N{ipg} = N(xi, eta);
   pg.dN{ipg} = dN(xi, eta);% funciones de formas derivadas
end
```

Integración pre-cargada

Código 7.3: Aplicación del método de integración pre-cargada.

```
DOF = reshape(1:dof,Ndofpornod,[])';
for e = 1:Nelem
    ke = zeros(Ndofporelem);
    meindof = reshape(DOF(elementos(e,:),:)',1,[]);
    elenod = nodos(elementos(e,:),:);
    for ipg = 1:pg.n
        jac = pg.dN{ipg}*elenod; % estas dos lineas corren
        dNxyz = jac\pg.dN{ipg}; %mucho mas rapido que integracion lenta
        Djac = det(jac);
        B = zeros(size(E,2),Ndofporelem);
        %% Calculo B para mi elemento
        ke = ke + B'*E*B*pg.w(ipg)*Djac;
    end
    %Acople a matriz de rigidez K
end
```

7.3. Acople rápido de [K]

Se puede optimizar mucho el espacio en memoria cambiando la forma de almacenamiento de **[K]** a esparsa. Lo que no se suele saber es que importa mucho la forma en que se acoplan los valores a **[K]** para optimizar el *tiempo* de corrida. A continuación hay un esquema para el armado rápido de **[K]**.

Declaración de indices y vector valor

```
I = zeros( Ndofporelem*Nelem ,1 ); %indice i de K
J = zeros( Ndofporelem*Nelem ,1 ); %indice j de K
V = zeros( Ndofporelem*Nelem ,1 ); %Valores de K
Nentryporelem = Ndofporelem^2; %Cantidad de valores en ke
DOF = reshape(1:dof,Ndofpornod,[])';
n2de = @(e) repmat(Nentryporelem*e ,Nentryporelem ,1 ) - (Nentryporelem-1:-1:0)';
```

Acople Rápido

Código 7.4: Aplicación del método de acople rápido de la matriz de rigidez.

```
for e = 1:Ndofporelem
    ke = zeros(Ndofporelem,Ndofporelem);
    meindof = reshape(DOF(elementos(e,:),:)',1,[]);% == elemdof(e,:);
```

```
    elenod = nodos(elementos(e,:),:);
    % obtengo ke de alguna forma
    idx = n2de(e);
    I(idx) = repmat(meindof,1,Ndofporelem)' ;
    J(idx) = reshape(repmat(meindof,Ndofporelem,1),[],1);
    V(idx) = ke(:);
end

K = sparse( I, J, V , dof, dof); % spalloc rapido
```

Este método funciona muy bien para problemas con cientos de miles de dof. Para un problema de 600000 dof con elementos tetraedros de diez nodos (T10), este método permite pasar de un acople de 15.600 segundos (con integración rápida) a tan solo 50 segundos.

7.4. Verificación de [K]

Autovalores de [K]

Tomando los autovalores de **[K]** nos puede dar una buena idea de si nos falta aplicar condiciones de borde o si hay mal condicionamiento numérico. El problema de este método es que deja de funcionar para modelos con muchos dof ya que el calculo de autovalores es muy intensivo sobre los recursos de la computadora.

```
    [autoval, autovec] = eigs(K(isfree,isfree));
```

La existencia de autovalores nulos o muy cercanos a cero puede implicar

- Existencia de modos rígidos de desplazamiento
- Una razón alta entre valores de la diagonal de **[K]** y los no-diagonales (mal condicionamiento numérico)

Mecanismos y singularidades en NASTRAN

Una vez armada la matriz de rigidez global resulta útil poder encontrar dofs que actúen como mecanismos o problemas de condicionamiento. Para esto NASTRAN implementa un método para hallar estos dofs. Primero se restringe **[K]** aplicando condiciones de borde esenciales y ecuaciones de restricciones. Luego se divide cada elemento de su diagonal ($[\mathbf{K}]_{ii}$) por su elemento correspondiente de la matriz D de la descomposición LDL, obteniéndose así los *pivot-ratios*.

$$\text{PIVOTRATIO} = \frac{[\mathbf{K}]_{ii}}{D_{ii}}$$

en MATLAB se puede vectorizar el problema

Código 7.5: Rutina MAXPIVOT de NASTRAN.

```
KLL = K(isfree, isfree); %Condiciones de borde
pivots = decomposition(KLL,'ldl','lower')\diag(KLL);
```

son de interés los dof con pivot ratio mas alto. NASTRAN pone un limite superior de 10^8 antes de finalizar la corrida con un error.

Tenga cuidado al buscar el dof que causa el problema, pues redujo el problema aplicando condiciones de borde esenciales!

Capítulo 8

Anexo

8.1. Solver genérico para elementos 2D (Q4)

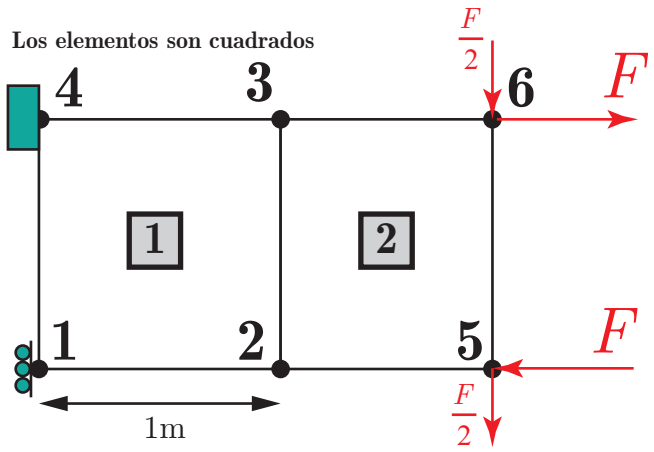


Figura 8.1: Problema plane-stress a resolver. $F = 10\text{kN}$, $t = 1\text{cm}$.

Código 8.1: q4shapefun.m

```
%% FuncForm Q4
syms x y real
X = [1 x y x*y];
q4nod = [-1  -1
          1  -1
          1   1
         -1   1];
A = zeros(size(q4nod,1),length(X));
for i=1:size(q4nod,1)
    x=q4nod(i,1); y = q4nod(i,2);
    A(i,:) = eval(X);
end
syms x y real
N = X*inv(A);
dN = [diff(N,x);diff(N,y)];
```

Código 8.2: q4solve.m

```
clear % Conviene siempre empezar con un espacio de trabajo liberado
q4shapefun % se crean las funciones de forma
nodos = [0 0;
         1 0
         1 1
         0 1
         2 0
         2 1]; %[metros]
elementos = [1 2 3 4;
             2 5 6 3];
%% DOFINITIONS
Ndofpornod = 2;
[Nnod, Ndim] = size(nodos);
[Nelem, Nnodporelem] = size(elementos);
dof = Ndofpornod*Nnod;
Ndofporelem = Ndofpornod*Nnodporelem;
n2d = @(n) [n*2 - 1; n*2];
%% Obtencion de elemdof
elemdof = zeros(Nelem,Ndofporelem);
for e = 1:Nelem
    elemdof(e,:)= reshape(n2d(elementos(e,:)),[],1);
end
%% Propiedades del material
young = 200E9;                              % Modulo de Young acero [Pa]
nu = 0.3;                                   % Poisson
t = 1E-2;                                   % espesor [metros]
E = young/(1-nu^2)*[1 nu 0;nu 1 0;0 0 (1-nu)/2]; %plane stress
%% Gauss para regla de 2x2
a   = 1/sqrt(3);
upg = [ -a  -a;a  -a;a   a;-a   a ];% Ubicaciones puntos de Gauss
npg = size(upg,1);
wpg = ones(npg,1); %Weight vale 1 para orden n=1, grado precision 2
%% Obtencion Matriz Rigidez
K = zeros(dof);
for e = 1:Nelem
    meindof = elemdof(e,:);
    ke = zeros(Ndofporelem);
    elemnod = nodos(elementos(e,:),:);
    for ipg = 1:npg
        x = upg(ipg,1);
        y = upg(ipg,2);
        J = eval(dN)*elemnod; % se podrian precargar los resultados de
        dNxy = J\eval(dN);    % la integracion para problemas con muchos elementos
        B=zeros(size(E,2),size(ke,1)); %Stress-Strain Matrix
        B(1,1:2:end) = dNxy(1,:);
        B(2,2:2:end) = dNxy(2,:);
        B(3,1:2:end) = dNxy(2,:); % 3.6-6 Cook
        B(3,2:2:end) = dNxy(1,:);
        ke = ke + B'*E*B*wpg(ipg)*det(J)*t;
    end
    K(meindof,meindof) = K(meindof,meindof) + ke;
end
%% Obtencion vector columna de cargas R
R = zeros(dof,1);
```

```
R(n2d(5)) = [-10e3 -5e3]; % -20kN en x, -10kN en y
R(n2d(6)) = [10e3 -5e3];  % [Newton]
%% Condiciones de Borde isostaticas
isFixed = false(dof,1);
isFixed(n2d(1)) = [true false]; % Roller apoyado en x nodo 1
isFixed(n2d(4)) = [true true];  % Apoyo simple nodo 4
%% Solucion del sistema lineal
Dr = K(~isFixed,~isFixed)\R(~isFixed);
D = zeros(dof,1);
D(~isFixed) = Dr;
Dxy = reshape(D,2,[])';
escala = 1000; % Escala para amplificar desplazamientos
posdef = nodos + escala*Dxy;
q4stress
```

Código 8.3: q4stress.m

```
%% Tensiones en nodos directo
% Matriz 3D para guardar tensiones
stress = zeros(size(E,2),Nnodporelem,Nelem);
for e = 1:Nelem
    elemnod = nodos(elementos(e,:),:);
    meindof = elemdof(e,:);
    for n = 1:Nnodporelem % Obtengo tensiones en nodos directamente
        x = q4nod(n,1);  y = q4nod(n,2);
        J =subs(dN)*elemnod;
        dNxy = J\subs(dN);
        B = zeros(size(E,2),size(ke,1)); % Strain Displacement Matrix
        B(1,1:2:end-1) = dNxy(1,:);
        B(2,2:2:end)   = dNxy(2,:);
        B(3,1:2:end-1) = dNxy(2,:);
        B(3,2:2:end)   = dNxy(1,:);
        stress(:,n,e)  = E*B*D(meindof);
    end
end
Sxx = squeeze(stress(1,:,:))'; %Tensiones xx por elemento, por nodo
Syy = squeeze(stress(2,:,:))';
Sxy = squeeze(stress(3,:,:))';
Svm2d = sqrt( Sxx.^2 + Syy.^2 + 3*Sxy.^2 ); % Von mises. Tension efectiva
```

Código 8.4: q4graphstress.m

```
%% Graficador de tensiones para Q4
for e = 1:Nelem
    elenod = elementos(e,:);
    h=patch('Faces',1:4,'Vertices',posdef(elenod,:),...
            'FaceVertexCData',Sxx(e,:)');
    set(h,'FaceColor','interp','CDataMapping','scaled');
end
colormap(jet(10)); %10 es el numero de colores en el bandplot
Slims = [min(min(Sxx)) max(max(Sxx))];
daspect([1 1 1]); % Visualizacion a escala
title('\sigma_{xx} [Pa]');xlabel('x [m]');ylabel('y [m]')
colorbar('YTick',Slims(1):diff(Slims)/5:Slims(2));
```

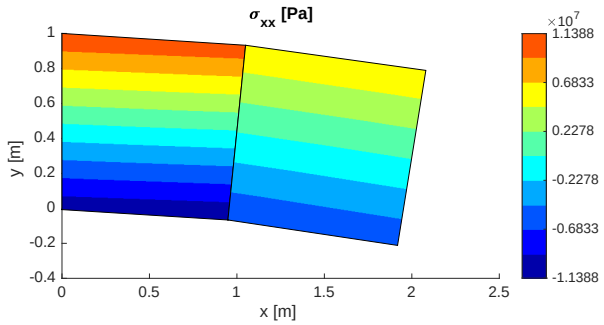


Figura 8.2: Resultado de los códigos de la sección 8.1.

8.2. Cuadratura de Gauss

Código 8.5: gauss.m

```
function [pg] = gauss(quad)
% GAUSS Obtiene puntos de cuadratura sobre una linea, superficie o volumen
% para los elementos finitos. El output contiene los siguientes campos:
%
% u: Posiciones de los puntos gauss
% w: Pesos de los puntos gauss
% n: Cantidad de puntos gauss
%
% Ejemplos
%   pg = gauss([3 3]) %Cuadratura superficie 3x3 de 9 puntos gauss (Orden 2)
%   pg = gauss([5 5 5]) % Cuadratura volumen 5x5x5 de 125 puntos gauss (Orden 4)

    if max(quad) > 5 || min(quad) < 1
        error('Ingresar un vector con numero indicando cantidad de puntos gauss en una
            dimension.');
    elseif length(quad)>3
        error('Longitud de vector ingresado no puede exceder 3 (dimension)!');
    end
    x4 = [sqrt( (3-2*sqrt(6/5))/7 ), sqrt( (3+2*sqrt(6/5))/7 )];
    x5 = [1/3*sqrt(5 - 2*sqrt(10/7)), 1/3*sqrt(5 + 2*sqrt(10/7))];
    w4 = [18+sqrt(30) 18-sqrt(30)]/36;
    w5 = [322+13*sqrt(70) 322-13*sqrt(70)]/900;
    % Vectores precargados
    X = {  0
           [-sqrt(1/3) sqrt(1/3)]
           [-sqrt(3/5) 0 sqrt(3/5)]
           [-x4(2) -x4(1) x4(1) x4(2)]
           [-x5(2) -x5(1) 0 x5(1) x5(2)]};
    W = {   2
            [1 1]
            [5/9 8/9 5/9]
            [w4(2) w4(1) w4(1) w4(2)]
            [w5(2) w5(1) 128/225 w5(1) w5(2)]};
    dim = length(quad);
    pg.n = prod(quad); % Cantidad de puntos gauss
    pg.u = zeros(pg.n,dim); % Posiciones
```

```
    pg.w = zeros(pg.n,1); %Pesos

    weightcols = zeros(pg.n,dim); % Auxiliar para calcular pesos
    for d = 1:dim
        xpg = X{quad(d)};
        wpg = W{quad(d)};
        if d == dim
            pg.u(:,dim) = repmat(xpg',pg.n/length(xpg),1);
            weightcols(:,dim) = repmat(wpg',pg.n/length(xpg),1);
            break
        end
        col = [];
        colw = [];
        for k = 1:pg.n/(prod(quad(d+1:end))*length(xpg))
            for j = 1:quad(d)
                col = [col; repmat(xpg(j)', prod(quad(d+1:end)),1)];
                colw = [colw; repmat(wpg(j)', prod(quad(d+1:end)),1)];
            end
        end
        pg.u(:,d) = col;
        weightcols(:,d) = colw;
    end
    pg.w = prod(weightcols,2);
end
```

Apuntes Auxiliares (Luis)

Los elementos finitos siempre me dan soluciones más rígidas que la realidad porque estoy limitando sus nodos de movimientos mediante el uso de polinomios. En la realidad persisten funciones más complejas. Esto significa que el método de elementos finitos me aproxima la realidad por el lado más rígido, esto se tiene que tomar en cuenta!

Tablas

$n=0$						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

Tabla 8.1: El triangulo de pascal de orden $n=6$.

Expresiones útiles

Operador derivada

Factor Corrección para $\nu \approx 0,25$	
Perfil	k
Rectangular $\frac{h}{b} \gtrsim 1$	$\frac{5}{6}$
Rectangular $\frac{h}{b} = 0,5$	$0,7961$
Rectangular $\frac{h}{b} = 0,25$	$0,6308$
Circular	$\frac{9}{10}$
Tubo de paredes delgadas circular	$\frac{1}{2}$
Wide Flange doble T	$k_y \approx \frac{A_f}{1,2A}$ $k_z \approx \frac{A_w}{A}$

Tabla 8.2: A_w es el área del alma y A_f es el área del ala.

- Barra: $\partial\, x$
- Viga: $\frac{\partial^2 v}{\partial x^2}$
- 2D: $\begin{bmatrix} \partial/\partial x & 0 \\ 0 & \partial/\partial y \\ \partial/\partial y & \partial/\partial x \end{bmatrix}$

$$\sigma_v = \sqrt{\sigma_x^2 + \sigma_y^2 + \sigma_z^2 - \sigma_x\sigma_y - \sigma_x\sigma_z - \sigma_y\sigma_z + 3(\sigma_{xy}^2 + \sigma_{xz}^2 + \sigma_{yz}^2)} \tag{8.1}$$

$$\sigma_v = \sqrt{\tfrac{1}{2}[(\sigma_{11}-\sigma_{22})^2 + (\sigma_{11}-\sigma_{33})^2 + (\sigma_{22}-\sigma_{33})^2] + 3(\sigma_{12}^2 + \sigma_{13}^2 + \sigma_{23}^2)} \tag{8.2}$$

$$\lambda = \frac{E\,\nu}{(1+\nu)(1-2\nu)} \qquad \mu = G = \frac{E}{2(1+\nu)} \tag{8.3}$$

$$\begin{bmatrix} \sigma_{xx} \\ \sigma_{yy} \\ \sigma_{xy} \end{bmatrix} = \frac{E}{1-\nu^2} \begin{bmatrix} 1 & \nu & 0 \\ \nu & 1 & 0 \\ 0 & 0 & \frac{1-\nu}{2} \end{bmatrix} \cdot \begin{bmatrix} \varepsilon_{xx} \\ \varepsilon_{yy} \\ 2\varepsilon_{xy} \end{bmatrix} \tag{8.4}$$

$$\begin{bmatrix} \sigma_{11} \\ \sigma_{22} \\ \sigma_{12} \end{bmatrix} = \frac{E}{(1+\nu)(1-2\nu)} \begin{bmatrix} 1-\nu & \nu & 0 \\ \nu & 1-\nu & 0 \\ 0 & 0 & \frac{1-2\nu}{2} \end{bmatrix} \cdot \begin{bmatrix} \varepsilon_{11} \\ \varepsilon_{22} \\ 2\varepsilon_{12} \end{bmatrix} \tag{8.5}$$

$$\sigma_n = \frac{\sigma_{xx} + \sigma_{yy}}{2} + \left(\frac{\sigma_{xx} - \sigma_{yy}}{2}\right)\cos 2\theta + \tau_{xy}\sin 2\theta \tag{8.6}$$

$$\tau_n = -\left(\frac{\sigma_{xx} - \sigma_{yy}}{2}\right)\sin 2\theta + \tau_{xy}\cos 2\theta \tag{8.7}$$

$$I = \int_{-1}^{1} \phi(\xi)\mathrm{d}\xi \approx \phi(\xi_1)W_1 + \phi(\xi_2)W_2 \ldots \phi(\xi_n)W_n \tag{8.8}$$

$$I = \int_{-1}^{1}\int_{-1}^{1} \phi(\xi,\eta)\mathrm{d}\xi\mathrm{d}\eta \approx \sum_i \sum_j W_i W_j \phi(\xi,\eta) \tag{8.9}$$

Se suele requerir que $\eta \leq 0,05$
Formulación de elemento rígido RBE2:

$$g_u = u_s - u_m + L_e\,\hat{v}_y\left(\frac{\theta_m \sin\left(\sqrt{\vartheta}\right)}{\sqrt{\vartheta}} + \frac{\varphi_m\,\psi_m\left(\cos\left(\sqrt{\vartheta}\right)-1\right)}{\vartheta}\right) - L_e\,\hat{v}_z\left(\frac{\psi_m \sin\left(\sqrt{\vartheta}\right)}{\sqrt{\vartheta}} - \frac{\varphi_m\,\theta_m\left(\cos\left(\sqrt{\vartheta}\right)-1\right)}{\vartheta}\right) - \frac{L_e\,\hat{v}_x\left({\psi_m}^2 + {\theta_m}^2\right)\left(\cos\left(\sqrt{\vartheta}\right)-1\right)}{\vartheta}$$

$$g_v = v_s - v_m - L_e\,\hat{v}_x\left(\frac{\theta_m\,\sin\left(\sqrt{\vartheta}\right)}{\sqrt{\vartheta}} - \frac{\varphi_m\,\psi_m\left(\cos\left(\sqrt{\vartheta}\right)-1\right)}{\vartheta}\right) + L_e\,\hat{v}_z\left(\frac{\varphi_m\,\sin\left(\sqrt{\vartheta}\right)}{\sqrt{\vartheta}} + \frac{\psi_m\,\theta_m\left(\cos\left(\sqrt{\vartheta}\right)-1\right)}{\vartheta}\right) - \frac{L_e\,\hat{v}_y\left({\varphi_m}^2 + {\theta_m}^2\right)\left(\cos\left(\sqrt{\vartheta}\right)-1\right)}{\vartheta}$$

$$g_w = w_s - w_m + L_e\,\hat{v}_x\left(\frac{\psi_m\,\sin\left(\sqrt{\vartheta}\right)}{\sqrt{\vartheta}} + \frac{\varphi_m\,\theta_m\left(\cos\left(\sqrt{\vartheta}\right)-1\right)}{\vartheta}\right) - L_e\,\hat{v}_y\left(\frac{\varphi_m\,\sin\left(\sqrt{\vartheta}\right)}{\sqrt{\vartheta}} - \frac{\psi_m\,\theta_m\left(\cos\left(\sqrt{\vartheta}\right)-1\right)}{\vartheta}\right) - \frac{L_e\,\hat{v}_z\left(\cos\left(\sqrt{\vartheta}\right)-1\right)\left({\varphi_m}^2 + {\psi_m}^2\right)}{\vartheta}$$

donde $\vartheta = \varphi_m^2 + \psi_m^2 + \theta_m^2$

$$g_\varphi = \varphi_s - \varphi_m\,, \qquad g_\psi = \psi_s - \psi_m\,, \qquad g_\theta = \theta_s - \theta_m$$

En 2D

Figuras

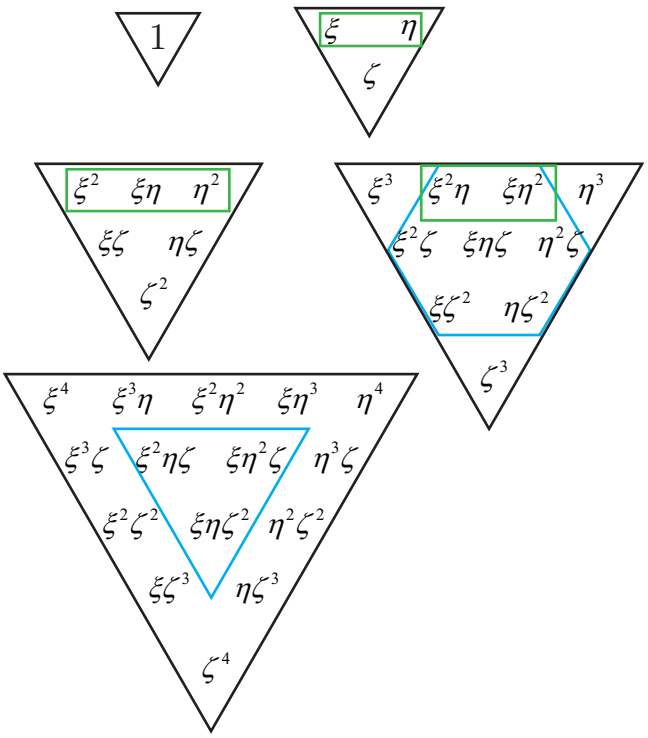


Figura 8.3: El tetraedro de Pascal. Los términos de los elementos *serendipidad* están encuadrados.

Bibliografía

Robert D Cook et al. *Concepts and applications of finite element analysis*. John Wiley & Sons, 2007.

Jack Chessa. Programing the finite element method with matlab. *Northwestern University*, 20(02), 2002.

Uday S Dixit. Finite element method: an introduction. *Indian Institute of Technology Guwahati-781*, 39, 2007.

Yunhua Luo. An efficient 3d timoshenko beam element with consistent shape functions. *Adv. Theor. Appl. Mech*, 1(3):95–106, 2008.

Stanley B Dong, Can Alpdogan, and Ertugrul Taciroglu. Much ado about shear correction factors in timoshenko beam theory. *International Journal of Solids and Structures*, 47(13):1651–1665, 2010.

Edgar Matas Hidalgo. Study of optimization for vibration absorbing devices applied on airplane structural elements. 2014.