UNIVERSIDAD NACIONAL DE COLOMBIA
SEDE BOGOTÁ

Análisis estructural básico (2015941-1) : Curso
Ma y Ju; 9:00-11:00 : Horario
406-229 : Lugar

Martin Estrada : Profesor
mestradam@unal.edu.co : Correo

APROXIMACIÓN A LA IMPLEMENTACIÓN DEL MÉTODO DE LA RIGIDEZ

A continuación se presenta una forma de implementar en computador una versión simplificada del método de la rigidez para hacer más eficiente la solución de ejercicios. Las rutinas o funciones que se presentan están escritas en archivos `.m`, los cuales se pueden utilizar directamente en programas de cálculo matemático como `MatLab`, `Octave` o `FreeMat`.

Archivos de entrada

La inserción de datos en los programas de cálculo es una tarea, por lo general, tediosa, larga y complicada. Pues cualquier error de digitación puede causar problemas graves o indetectables en la solución de un problema. Los datos necesarios para resolver los ejercicios en esta implementación particular se asignarán a través de un único archivo `<nombre>.m`, como se describe a continuación. En este momento los grados de libertad de los nudos no necesitan tener una etiqueta (número) asignada, pero se debe tener en cuenta el orden de esos GL en cada nudo. Por defecto el orden de los grados de libertad en un nudo tridimensional es $[\delta_x\, \delta_y\, \delta_z\, \theta_x\, \theta_y\, \theta_z]$ y ese orden se mantiene siempre. En los nudos más simples (con menos GL) se descartan los GL innecesarios, pero el orden se mantiene. Por ejemplo los GL asociados a un nudo de elementos tipo cercha son $[\delta_x\ \delta_y]$, los de un nudo de elementos tipo pórtico plano son $[\delta_x\ \delta_y\ \theta_z]$, etc.

Coordenadas de los nudos: Esta variable contiene las coordenadas de todos los nudos del dominio en forma de tabla. Cada fila de la tabla representa un nudo (fila 1 = nudo 1, fila 2 = nudo 2, etc.) y las columnas son las coordenadas x, y, y z.

Por ejemplo, si se tiene un dominio con 4 nudos en un espacio tridimensional con las coordenadas dadas en la tabla 1,

Tabla 1: Coordenadas de los nudos

Nudo	x	y	z
1	4	0	3
2	0	0	3
3	4	0	0
4	4	5	3

el archivo de entrada deberá tener la variable `coord` de la siguiente forma:

```
coord = [
4 0 3 ;
0 0 3 ;
4 0 0 ;
4 5 3 ]
```

Conectividades de los elementos: En esta tabla se especifican los dos nudos que conecta cada elemento y su orden indica la dirección del elemento. Cada fila indica el número de elemento y las columnas representan el nudo inicial r y el nudo final s, respectivamente.

Por ejemplo, si se tiene un dominio con 3 elementos conectados en 4 nudos de la siguiente forma:

Tabla 2: Conectividades de los elementos

Elem.	r	s
1	2	1
2	1	4
3	3	1

el archivo de entrada deberá tener la variable `conec` de la siguiente forma:

```
conec = [
2 1 ;
1 4 ;
3 1 ]
```

Restricciones: En este archivo se especifican las restricciones de movimiento que tiene cada nudo. El archivo puede verse como la unión de dos tablas: la mitad izquierda indica para cada GL si tiene (1) o no tiene (0) restricción de movimiento, y la parte derecha indica el valor o magnitud de esa restricción. A los GL que no tienen restricción, se les debe asignar un valor cualquiera en el lado derecho y por defecto puede ser cero (0). Cada fila indica un nudo.

Por ejemplo, si el nudo 1 de una estructura está libre de movimiento, el nudo 2 está empotrado sin moverse, el nudo 3 está empotrado con un desplazamiento de -10 cm en la dirección z y el nudo 4 puede girar libremente alrededor del eje x, la variable de restricciones debe ser

```
restr = [
0 0 0 0 0 0 0 0  0   0 0 0 ;
1 1 1 1 1 1 0 0  0   0 0 0 ;
1 1 1 1 1 1 0 0 -0.1 0 0 0 ;
1 1 1 0 1 1 0 0  0   0 0 0 ]
```

Propiedades de los elementos: En esta variable se especifican las propiedades necesarias de los elementos, entre las cuales se encuentran propiedades geométricas locales y propiedades mecánicas del material. El orden sugerido en que se escriben las propiedades de los elementos en este documento no sigue ninguna razón lógica, de manera que cada quién podrá hacerlo de una forma diferente. El tipo del elemento estará determinado por la tabla 3 y el corte determina si se tiene en cuenta (1) o no se tiene en cuenta (0) el aporte a la deflexión por los esfuerzos cortantes.

Tabla 3: Tipo de elemento

Tipo de elemento	Etiqueta
Cercha	1
Pórtico plano	2
Pórtico espacial orientado en x	3
Pórtico espacial orientado en y	4
Pórtico espacial orientado en z	5

Por ejemplo, para un ejercicio de 3 elementos con las propiedades registradas en la tabla 4,

el archivo de entrada deberá tener la variable `props` de la siguiente forma:

Tabla 4: Propiedades de los elementos

Elem.	tipo	$E(\,\mathrm{kN/m^2})$	$G(\,\mathrm{kN/m^2})$	$A(\,\mathrm{m^2})$
1	3	$2{,}2 \times 10^7$	$8{,}5 \times 10^6$	0,12
2	4	$2{,}2 \times 10^7$	$8{,}5 \times 10^6$	0,12
3	5	$2{,}2 \times 10^7$	$8{,}5 \times 10^6$	0,10

Elem.	$I_x(\,\mathrm{m^4})$	$I_y(\,\mathrm{m^4})$	$I_z(\,\mathrm{m^4})$	$J(\,\mathrm{m^4})$	α	corte
1	0	$9{,}0 \times 10^{-4}$	$1{,}6 \times 10^{-3}$	$1{,}9439 \times 10^{-3}$	1.2	1
2	$9{,}0 \times 10^{-4}$	0	$1{,}6 \times 10^{-3}$	$1{,}9439 \times 10^{-3}$	1.2	1
3	$5{,}2083 \times 10^{-4}$	$1{,}3333 \times 10^{-3}$	0	$1{,}2735 \times 10^{-3}$	1.2	1

```
props = [
3 2.2e+07 8.5e+06 0.12 0          9.0e-04    1.6e-03 1.9439e-3 1.2 1 ;
4 2.2e+07 8.5e+06 0.12 9.0e-04    0          1.6e-03 1.9439e-3 1.2 1 ;
5 2.2e+07 8.5e+06 0.10 5.2083e-04 1.3333e-03 0       1.2735e-3 1.2 1 ]
```

La longitud suele considerarse como una propiedad de los elementos, pero es recomendable calcularla a partir de las coordenadas de los nudos. De esta manera no se corre el riesgo de que a un elemento se le asigne una longitud mayor o menor que la real.

Fuerzas nodales: En esta variable se ingresan las acciones que están concentradas en nudo específico, en el orden apropiado: $[F_x\ F_y\ F_z\ M_x\ M_y\ M_z]$. El primer caracter de cada fila indica el número del nudo al cual van asociadas las fuerzas.

Por ejemplo el nudo 3 de una estructura tienen las fuerzas nodales dadas en la tabla 5.

Tabla 5: Fuerzas de nodales

Nudo	F_x	F_y	F_z	M_x	M_y	M_z
3	10,0	0,0	5,5	0,0	$-2{,}75$	0,0

de manera que el archivo de entrada contiene las siguientes fuerzas nodales:

```
fnod = [
3 10 0 5.5 0 -2.75 0 ]
```

Si en la estructura no existen fuerzas nodales externas, se recomienda que de todas formas exista la variable `fnod`, pero conteniendo únicamente un cero (0).

```
fnod = 0
```

Fuerzas de empotramiento: En esta variable se ingresan las fuerzas de empotramiento o fuerzas de la condición de desplazamiento 0. En esta implementación se pide que las fuerzas de empotramiento sean calculadas previamente y registradas en el archivo en el orden apropiado para cada elemento: $[F_x\ F_y\ F_z\ M_x\ M_y\ M_z]$ del nudo r y $[F_x\ F_y\ F_z\ M_x\ M_y\ M_z]$ del nudo s en una fila. El primer caracter de cada fila indica el número del elemento al cual van asociadas las fuerzas.

Por ejemplo los elementos 1 y 2 de una estructura tienen las fuerzas de empotramiento dadas en la tabla 6.

de manera que el archivo de entrada las fuerzas de empotramiento deberían ser:

Tabla 6: Fuerzas de empotramiento (condición de desplazamiento 0)

Elem.	F_x^r	F_y^r	F_z^r	M_x^r	M_y^r	M_z^r	F_x^s	F_y^s	F_z^s	M_x^s	M_y^s	M_z^s
1	0,0	6,0	0,0	0,0	0,0	5,0	0,0	6,0	0,0	0,0	0,0	−5,0
3	0,0	5,25	0,0	2,625	0,0	0,0	0,0	5,25	0,0	−2,625	0,0	0,0

```
femp = [
0 6 0 0 0 5 0 6 0 0 0 -5 ;
0 5.25 0 2.625 0 0 0 5.25 0 -2.625 0 0 ]
```

De la misma manera que con las fuerzas nodales, si en la estructura no existen fuerzas de empotramiento, la variable `femp.txt` debe existir y contener un cero (0).

```
femp = 0
```

Funciones para el cálculo de las matrices de rigidez

Las matrices de rigidez de diferentes tipos de elementos estructurales son distintas. A continuación se presentan las matrices de los elementos típicos en el análisis de estructuras civiles.

(`kCerchaPlana.m`) **Elementos tipo cercha plana:** Argumentos de la función:

- ← `k` matriz de rigidez del elemento.
- → `E` módulo de elasticidad del material.
- → `A` área transversal del elemento.
- → `L` longitud del elemento.
- → `c` coseno del ángulo de inclinación del elemento.
- → `s` seno del ángulo de inclinación del elemento.

kCerchaPlana.m

```
function [k] = kCerchaPlana(E,A,L,c,s)
    % -------------------------------------------------------------------
    % Calcula la matriz de rigidez de un elemento tipo cercha plana
    %
    % MEM-2011
    % -------------------------------------------------------------------

    % Matriz en coordenadas locales
    k = [ A*E/L ,  0 ,-A*E/L ,  0 ;
          0     ,  0 , 0     ,  0 ;
         -A*E/L ,  0 , A*E/L ,  0 ;
          0     ,  0 , 0     ,  0 ];

    % Matriz de transformación
    T = [ c , s ,  0 , 0  ;
         -s , c ,  0 , 0  ;
          0 , 0 ,  c , s  ;
          0 , 0 , -s , c  ];

    % Matriz en coordenadas globales
    k = T'*k*T;
```

(`kPorticoPlano.m`) **Elementos tipo pórtico plano:** Argumentos de la función:

← `k` matriz de rigidez del elemento.
→ `E` módulo de elasticidad del material.
→ `A` área transversal del elemento.
→ `I` momento de inercia de la sección transversal.
→ `L` longitud del elemento.
→ `B` factor de deformación por cortante (beta).
→ `c` coseno del ángulo de inclinación del elemento.
→ `s` seno del ángulo de inclinación del elemento.

kPorticoPlano.m

```
function [k] = kPorticoPlano(E,A,I,L,B,c,s)
    % -------------------------------------------------------------------
    % Calcula la matriz de rigidez de un elemento tipo pórtico teniendo en
    % cuenta la contribución a la deflexión por los esfuerzos cortantes
    % "beta".
    % El argumento beta debe ser calculado por fuera de la función y es:
    % B = (6*alpha*E*I)/(G*A*L*L)
    %
    % MEM-2011
    % -------------------------------------------------------------------

    % cálculo de constantes por efecto del cortante
    c1 = 1 / (1+2*B);
    c2 = (2+B) / (2*(1+2*B));
    c3 = (1-B) / (1+2*B);

    % variables internas
    d1 = E*A/L ;
    d2 = 12*E*I*c1/L^3 ;
    d3 = 6*E*I*c1/L^2 ;
    d4 = 4*E*I*c2/L ;
    d5 = 2*E*I*c3/L ;

    % matriz en coordenadas locales
    k = [ d1 , 0  , 0  ,-d1 , 0  , 0  ;
        0  , d2 , d3 , 0  ,-d2 , d3 ;
        0  , d3 , d4 , 0  ,-d3 , d5 ;
       -d1 , 0  , 0  , d1 , 0  , 0  ;
        0  ,-d2 ,-d3 , 0  , d2 ,-d3 ;
        0  , d3 , d5 , 0  ,-d3 , d4 ];

    % matriz de transformación
    T = [ c, s, 0, 0, 0, 0 ;
       -s, c, 0, 0, 0, 0 ;
        0, 0, 1, 0, 0, 0 ;
        0, 0, 0, c, s, 0 ;
        0, 0, 0,-s, c, 0 ;
        0, 0, 0, 0, 0, 1 ];

    % matriz de rigidez en coordenadas globales
    k = T' * k * T;
    end
```

(`kPortico3d<X,Y,Z>.m`) **Elementos tipo pórtico espacial:** Los elementos tipo pórtico espacial tienen una matriz de transformación un poco compleja de calcular. Por eso, acá no se presenta una matriz general, sino matrices específicas para elementos orientados en las direcciones x, y y z del sistema

coordenado.

Argumentos de las funciones:

- ← `k` matriz de rigidez del elemento.
- → `E` módulo de elasticidad del material.
- → `G` módulo de cortante del material.
- → `A` área transversal del elemento.
- → `Ix` momento de inercia alrededor de x de la sección transversal.
- → `Iy` momento de inercia alrededor de y de la sección transversal.
- → `Iz` momento de inercia alrededor de z de la sección transversal.
- → `L` longitud del elemento.
- → `Bx` factor de deformación por cortante (beta) alrededor de x.
- → `By` factor de deformación por cortante (beta) alrededor de y.
- → `Bz` factor de deformación por cortante (beta) alrededor de z.

kPortico3dX.m

```
function [k] = kPortico3dZ(E,G,L,A,Ix,Iy,J,Bx,By)
    % ------------------------------------------------------------------
    % Calcula la matriz de rigidez de un elemento tipo pórtico espacial
    % con la contribución a la deflexión por los esfuerzos cortantes
    % "beta".
    % El argumento beta debe ser calculado por fuera de la función y es:
    % beta = (6*alpha*E*I)/(G*A*L*L)
    %
    % MEM-2011
    % ------------------------------------------------------------------

    % cálculo de constantes por efecto del cortante
    c1x = 1 / (1+2*Bx);
    c2x = (2+Bx) / (2*(1+2*Bx));
    c3x = (1-Bx) / (1+2*Bx);

    c1y = 1 / (1+2*By);
    c2y = (2+By) / (2*(1+2*By));
    c3y = (1-By) / (1+2*By);

    % variables internas
    d1 = E*A/L ;
    d2 = G*J/L ;

    d3x = 12*E*Ix*c1x/L^3 ;
    d4x = 6*E*Ix*c1x/L^2 ;
    d5x = 4*E*Ix*c2x/L ;
    d6x = 2*E*Ix*c3x/L ;

    d3y = 12*E*Iy*c1y/L^3 ;
    d4y = 6*E*Iy*c1y/L^2 ;
    d5y = 4*E*Iy*c2y/L ;
    d6y = 2*E*Iy*c3y/L ;

    % Matriz
    k = [ d3y , 0   , 0  , 0   , d4y , 0  ,-d3y , 0   , 0  , 0   , d4y , 0  ;
        0   , d3x , 0  ,-d4x , 0   , 0  , 0   ,-d3x , 0  ,-d4x , 0   , 0  ;
        0   , 0   , d1 , 0   , 0   , 0  , 0   , 0   ,-d1 , 0   , 0   , 0  ;
```

```
     0   ,-d4x , 0  , d5x , 0   , 0  , 0   , d4x , 0  , d6x , 0   , 0  ;
     d4y , 0   , 0  , 0   , d5y , 0  ,-d4y , 0   , 0  , 0   , d6y , 0  ;
     0   , 0   , 0  , 0   , 0   , d2 , 0   , 0   , 0  , 0   , 0   ,-d2 ;
    -d3y , 0   , 0  , 0   ,-d4y , 0  , d3y , 0   , 0  , 0   ,-d4y , 0  ;
     0   ,-d3x , 0  , d4x , 0   , 0  , 0   , d3x , 0  , d4x , 0   , 0  ;
     0   , 0   ,-d1 , 0   , 0   , 0  , 0   , 0   , d1 , 0   , 0   , 0  ;
     0   ,-d4x , 0  , d6x , 0   , 0  , 0   , d4x , 0  , d5x , 0   , 0  ;
     d4y , 0   , 0  , 0   , d6y , 0  ,-d4y , 0   , 0  , 0   , d5y , 0  ;
     0   , 0   , 0  , 0   , 0   ,-d2 , 0   , 0   , 0  , 0   , 0   , d2 ];
```

kPortico3dY.m

```
function [k] = kPortico3dY(E,G,L,A,Ix,Iz,J,Bx,Bz)
    % ------------------------------------------------------------------
    % Calcula la matriz de rigidez de un elemento tipo pórtico espacial
    % con la contribución a la deflexión por los esfuerzos cortantes
    % "beta".
    % El argumento beta debe ser calculado por fuera de la función y es:
    % beta = (6*alpha*E*I)/(G*A*L*L)
    %
    % MEM-2011
    % ------------------------------------------------------------------

    % cálculo de constantes por efecto del cortante
    c1x = 1 / (1+2*Bx);
    c2x = (2+Bx) / (2*(1+2*Bx));
    c3x = (1-Bx) / (1+2*Bx);

    c1z = 1 / (1+2*Bz);
    c2z = (2+Bz) / (2*(1+2*Bz));
    c3z = (1-Bz) / (1+2*Bz);

    % variables internas
    d1 = E*A/L ;
    d2 = G*J/L ;

    d3x = 12*E*Ix*c1x/L^3 ;
    d4x = 6*E*Ix*c1x/L^2 ;
    d5x = 4*E*Ix*c2x/L ;
    d6x = 2*E*Ix*c3x/L ;

    d3z = 12*E*Iz*c1z/L^3 ;
    d4z = 6*E*Iz*c1z/L^2 ;
    d5z = 4*E*Iz*c2z/L ;
    d6z = 2*E*Iz*c3z/L ;

    % Matriz
    k = [ d3z , 0  , 0   , 0   , 0  ,-d4z ,-d3z , 0  , 0   , 0   , 0  ,-d4z ;
        0   , d1 , 0   , 0   , 0  , 0   , 0   ,-d1 , 0   , 0   , 0  , 0   ;
        0   , 0  , d3x , d4x , 0  , 0   , 0   , 0  ,-d3x , d4x , 0  , 0   ;
        0   , 0  , d4x , d5x , 0  , 0   , 0   , 0  ,-d4x , d6x , 0  , 0   ;
        0   , 0  , 0   , 0   , d2 , 0   , 0   , 0  , 0   , 0   ,-d2 , 0   ;
       -d4z , 0  , 0   , 0   , 0  , d5z , d4z , 0  , 0   , 0   , 0  , d6z ;
       -d3z , 0  , 0   , 0   , 0  , d4z , d3z , 0  , 0   , 0   , 0  , d4z ;
        0   ,-d1 , 0   , 0   , 0  , 0   , 0   , d1 , 0   , 0   , 0  , 0   ;
        0   , 0  ,-d3x ,-d4x , 0  , 0   , 0   , 0  , d3x ,-d4x , 0  , 0   ;
```

```
     0   , 0  , d4x , d6x , 0  , 0   , 0   , 0  ,-d4x , d5x , 0  , 0   ;
     0   , 0  , 0   , 0   ,-d2 , 0   , 0   , 0  , 0   , 0   , d2 , 0   ;
    -d4z , 0  , 0   , 0   , 0  , d6z , d4z , 0  , 0   , 0   , 0  , d5z ];
```

kPortico3dZ.m

```
function [k] = kPortico3dX(E,G,L,A,Iy,Iz,J,By,Bz)
    % ------------------------------------------------------------------
    % Calcula la matriz de rigidez de un elemento tipo pórtico espacial
    % con la contribución a la deflexión por los esfuerzos cortantes
    % "beta".
    % El argumento beta debe ser calculado por fuera de la función y es:
    % beta = (6*alpha*E*I)/(G*A*L*L)
    %
    % MEM-2011
    % ------------------------------------------------------------------

    % cálculo de constantes por efecto del cortante
    c1y = 1 / (1+2*By);
    c2y = (2+By) / (2*(1+2*By));
    c3y = (1-By) / (1+2*By);

    c1z = 1 / (1+2*Bz);
    c2z = (2+Bz) / (2*(1+2*Bz));
    c3z = (1-Bz) / (1+2*Bz);

    % variables internas
    d1 = E*A/L ;
    d2 = G*J/L ;

    d3y = 12*E*Iy*c1y/L^3 ;
    d4y = 6*E*Iy*c1y/L^2 ;
    d5y = 4*E*Iy*c2y/L ;
    d6y = 2*E*Iy*c3y/L ;

    d3z = 12*E*Iz*c1z/L^3 ;
    d4z = 6*E*Iz*c1z/L^2 ;
    d5z = 4*E*Iz*c2z/L ;
    d6z = 2*E*Iz*c3z/L ;

    % Matriz
    k = [ d1 , 0   , 0   , 0  , 0   , 0   ,-d1 , 0   , 0   , 0  , 0   , 0   ;
        0  , d3z , 0   , 0  , 0   , d4z , 0  ,-d3z , 0   , 0  , 0   , d4z ;
        0  , 0   , d3y , 0  ,-d4y , 0   , 0  , 0   ,-d3y , 0  ,-d4y , 0   ;
        0  , 0   , 0   , d2 , 0   , 0   , 0  , 0   , 0   ,-d2 , 0   , 0   ;
        0  , 0   ,-d4y , 0  , d5y , 0   , 0  , 0   , d4y , 0  , d6y , 0   ;
        0  , d4z , 0   , 0  , 0   , d5z , 0  ,-d4z , 0   , 0  , 0   , d6z ;
       -d1 , 0   , 0   , 0  , 0   , 0   , d1 , 0   , 0   , 0  , 0   , 0   ;
        0  ,-d3z , 0   , 0  , 0   ,-d4z , 0  , d3z , 0   , 0  , 0   ,-d4z ;
        0  , 0   ,-d3y , 0  , d4y , 0   , 0  , 0   , d3y , 0  , d4y , 0   ;
        0  , 0   , 0   ,-d2 , 0   , 0   , 0  , 0   , 0   , d2 , 0   , 0   ;
        0  , 0   ,-d4y , 0  , d6y , 0   , 0  , 0   , d4y , 0  , d5y , 0   ;
        0  , d4z , 0   , 0  , 0   , d6z , 0  ,-d4z , 0   , 0  , 0   , d5z ];
```

Generación de tablas necesarias

Para ordenar el problema y poder ensamblar las matrices de rigidez se requiere enumerar los grados de libertad y asignarle esos grados de libertad a cada elemento. Esto se hace con dos rutinas:

(`asignarGL.m`) **Asignar grados de libertad:** Esta función le asigna un valor numérico a los grados de libertad de cada nudo de la estructura. A continuación se presenta una rutina o función que asigna la numeración de los GL ordenando los GL libres (sin restricción) de manera ascendente desde 1 en adelante y los GL restringidos de manera descendente desde el número total de grados de libertad hacia atrás.

Argumentos de la función:

←	`grlib`	tabla de grados de libertad por nudo.
→	`nnud`	número total de nudos de la estructura.
→	`glnud`	grados de libertad por nudo.
→	`ngld`	número de grados de libertad con desplazamiento desconocido
→	`restr`	tabla de restricciones.

asignarGL.m

```
function grlib = asignarGL(nnud,glnud,ngld,restr)
    % -------------------------------------------------------------------
    % Asignar grados de libertad a los nudos de la estructura en un
    % orden lógico: los grados de libertad libres (cinemáticos) se
    % asignan en orden ascendente desde 1 en adelante y los GL
    % restringidos se asignan en orden ascendente desde ngld hasta
    % el último GL de todo el dominio.
    %
    % MEM-2011
    % -------------------------------------------------------------------

    %
    cont1 = 1;
    cont2 = ngld + 1;
    for i=1:nnud
      for j=1:glnud
        if restr(i,j)==0
          grlib(i,j) = cont1;
          cont1 = cont1+1;
        else
          grlib(i,j) = cont2;
          cont2 = cont2+1;
        end
      end
    end
```

(`crearInidencias.m`) **Crear tabla de incidencias:** Esta función genera una tabla indicando los grados de libertad sobre los cuales incide cada elemento. Cada fila representa un elemento y las columnas son grados de libertad: primero los GL del nudo inicial del elemento y luego los GL del nudo final del elemento.

Argumentos de la función:

←	`incid`	tabla de incidencias.
→	`nele`	número total de elementos de la estructura.
→	`conec`	tabla de conectividades de los elementos.
→	`grlib`	tabla de grados de libertad por nudo.

crearIncidencias.m

```
function incid = crearIncidencias(nele,conec,grlib)
    % ------------------------------------------------------------------
    % Crea la tabla de incidencias de los elementos en su orden.
    % La tabla de incidencias se refiere a una tabla con los grados de
    % libertad sobre los cuales incide cada elemento. Esta tabla
    % contiene por cada fila (elemento) los GL asociados a ese elemento.
    %
    % MEM-2011
    % ------------------------------------------------------------------

    %
    for i=1:nele
      incid(i,:) = [grlib(conec(i,1),:),grlib(conec(i,2),:)];
    end
```

Ensamblaje de matriz de rigidez y vectores de fuerzas

Una vez se tienen las tablas anteriores, la matriz de rigidez de cada elemento de la estructura y los vectores de fuerzas, se debe ensamblar todo esto en un sistema global. De este procedimiento se obtiene la matriz de rigidez y los vectores de fuerzas de la estructura completa, no de cada elemento por separado.

(`ensamblarM.m`) **Ensamblar matrices:** Cada elemento de la estructura le aporta algo de rigidez a la matriz global y la localización o posición de ese aporte depende de los grados de libertad sobre los cuales incide el elemento en cuestión. La siguiente función permite sumar el aporte de rigidez de un elemento en la posición correcta de la matriz de rigidez global.

Argumentos de la función:

- ↔ `k` matriz de rigidez global de la estructura sin el aporte del elemento, que será reemplazada por la matriz con el aporte del elemento
- → `kel` matriz de rigidez del elemento.
- → `incid` tabla de incidencias.

ensamblarM.m

```
function k = ensamblarM(k,kel,incid)
    % ------------------------------------------------------------------
    % Ensambla la matriz global de la estructura con base en la matriz
    % de rigidez del elemento y la tabla de incidencias. Cabe
    % recordar que la tabla de incidencias contiene los grados de
    % libertad asociados a cada elemento, con lo cual resulta fácil
    % saber en cuál posición de la matriz global se debe sumar el
    % aporte de rigidez del elemento en cuestión.
    %
    % Esta rutina sólo ensambla la matriz de un elemento y no de todos
    % los del dominio. Es necesario hacer un ciclo para ejecutar esta
    % función tantas veces como elementos existan en el problema.
    %
    % MEM-2011
    % ------------------------------------------------------------------

    %
    [fil,col] = size(kel);
    for i=1:1:fil
      for j=1:1:col
```

```
        k(incid(1,i),incid(1,j))=k(incid(1,i),incid(1,j))+kel(i,j);
      end
    end
```

Como el procedimiento anterior sólo ensambla la matriz de rigidez de un elemento, se requiere un ciclo en la rutina principal de análisis que permita ensamblar la matriz de la estructura con el aporte de todos los elementos.

```
k = zeros(ngl,ngl);
for i=1:nele
  % matriz de rigidez de ese tipo de elemento específico
  ke = kPorticoPlano(A(i),E(i),I(i),L(i),m(i),n(i))
  % ensamblaje de la matriz anterior en la global
  k = ensamblar(k,ke,incid(i,:));
end
```

(`ensamblarV.m`) **Ensamblar vectores:** Los vectores de fuerzas o de desplazamientos asociados a elementos o nudos particulares también se pueden ensamblar de una forma muy similar a las matrices de rigidez. Los grados de libertad sobre los cuales incide cada componente de fuerza o desplazamiento determinará su lugar en el vector global.

Argumentos de la función:

- ↔ `f` vector de fuerza global de la estructura sin el aporte del elemento. Será reemplazado por el vector con el aporte del elemento
- → `fel` vector de fuerzas del elemento o el nudo.
- → `incid` tabla de incidencias del elemento o el nudo.

ensamblarV.m

```
function f = ensamblarV(f,fel,incid)
    % -------------------------------------------------------------------
    % Ensambla un vector de fuerzas con base en el vector de incidencias.
    % Si la fuerza es nodal el vector de incidencias es el vector de
    % grados de libertad del nudo.
    %
    % El vector de fuerzas debe ser un vector columna.
    %
    % MEM-2011
    % -------------------------------------------------------------------

    %
    [fil,col] = size(fel);
    for i=1:1:fil
      f(incid(1,i),1)=f(incid(1,i),1)+fel(i,1);
    end
```

Esta función sólo ensambla un vector y no todos los existentes. Además la tabla de incidencias de los elementos no es la misma que para los nudos, la cual se conoce más comúnmente con el nombre de tabla de grados de libertad (`grlib`). El procedimiento necesario para ensamblar los vectores de fuerzas nodales y de empotramiento, utilizando las tablas anteriores, son los siguientes:

```
if fnod == 0
  fa = zeros(ngld,1);
else
  [fil,col] = size(fnod);
  fa = zeros(ngld,1);
  for i=1:fil
    fa = ensamblarV(fa,fnod(i,2:col)',grlib(fnod(i,1),:));
  end
end

% vector de fuerzas de empotramiento
if femp == 0
  f0 = zeros(ngl,1);
else
  [fil,col] = size(femp);
  f0 = zeros(ngl,1);
  for i=1:fil
    f0 = ensamblarV(f0,femp(i,2:col)',incid(femp(i,1),:));
  end
end
```

Procedimiento global

Con la intención de generar un procedimiento sistemático, es posible utilizar todo lo anterior dentro de una rutina que se encargue de hacer todo lo necesario para resolver un ejercicio. Una posibilidad es la siguiente.

rigidez.m

```
function rigidez
    % ------------------------------------------------------------------
    % Rutina general para solucionar problemas de análisis estructural
    % con el métododе la rigidez.
    %
    % MEM-2011
    % ------------------------------------------------------------------

    %
    clear all
    clc
    addpath("./funciones")
    diary solucion.txt

    fprintf('\n\n')
    fprintf('=========================================================\n')
    fprintf('\n')

    % Lectura de los datos de entrada
    % -------------------------------
    eval(<inputFile>.m)

    % Cálculos preliminares
    % ---------------------

    % número de elementos
```

```
[fil,col] = size(props);
nele = fil;

% número de nudos
[fil,col] = size(coord);
nnud = fil;

% número de grados de libertad por nudo
[fil,col] = size(restr);
glnud = (col)/2;

% número total de grados de libertad
ngl = nnud*glnud;

% número de grados de libertad con desplazamiento desconocido
dummy = sum(restr);
nglc = sum(dummy(1:glnud));
ngld = ngl-nglc;

% asignar propiedades a variables independientes
tipo = props(:,1);
E = props(:,2);
G = props(:,3);
A = props(:,4);
Ix = props(:,5);
Iy = props(:,6);
Iz = props(:,7);
J = props(:,8);
alpha = props(:,9);
corte = props(:,10);

% longitud y orientación de los elementos
for i=1:nele
  L(i) = sqrt((coord(conec(i,2),1)-coord(conec(i,1),1))^2 + ...
              (coord(conec(i,2),2)-coord(conec(i,1),2))^2 + ...
              (coord(conec(i,2),3)-coord(conec(i,1),3))^2 );
  c(i) = (coord(conec(i,2),1)-coord(conec(i,1),1))/L(i);
  s(i) = (coord(conec(i,2),2)-coord(conec(i,1),2))/L(i);
end

% Generación de tablas y variables de apoyo
% ------------------------------------------

% asignar numeración de grados de libertad
grlib = asignarGL(nnud,glnud,ngld,restr)

% tabla de incidencias
incid = crearIncidencias(nele,conec,grlib)

% calcular constantes beta
for i=1:nele
  if corte(i) == 1
    Bx(i) = (6*alpha(i)*E(i)*Ix(i))/(G(i)*A(i)*L(i)*L(i));
    By(i) = (6*alpha(i)*E(i)*Iy(i))/(G(i)*A(i)*L(i)*L(i));
    Bz(i) = (6*alpha(i)*E(i)*Iz(i))/(G(i)*A(i)*L(i)*L(i));
  else
    Bx(i) = 0;
```

```
    By(i) = 0;
    Bz(i) = 0;
  end
end

% Matriz de rigidez
% -----------------

% matriz de rigidez elementales y estructural (global)
k = zeros(ngl,ngl);
for i=1:nele
  fprintf('Matriz k del elemento %i \n',i)
  % matriz de rigidez del elemento i
  if tipo(i) == 1
    kel = kCerchaPlana(E(i),A(i),L(i),c(i),s(i))
  elseif tipo(i) == 2
    kel = kPorticoPlano(E(i),A(i),Iz(i),L(i),Bz(i),c(i),s(i))
  elseif tipo(i) == 3
    kel = kPortico3dX(E(i),G(i),L(i),A(i),Iy(i),Iz(i),J(i),By(i),Bz(i))
  elseif tipo(i) == 4
    kel = kPortico3dY(E(i),G(i),L(i),A(i),Ix(i),Iz(i),J(i),Bx(i),Bz(i))
  elseif tipo(i) == 5
    kel = kPortico3dZ(E(i),G(i),L(i),A(i),Ix(i),Iy(i),J(i),Bx(i),By(i))
  end
  % ensamblaje de la matriz anterior en la global
  k = ensamblarM(k,kel,incid(i,:));
end

% separar matriz en submatrices
fprintf('Submatrices de rigidez\n',i)
kaa = k(1:ngld,1:ngld)
kab = k(1:ngld,ngld+1:ngl)
kba = k(ngld+1:ngl,1:ngld)
kbb = k(ngld+1:ngl,ngld+1:ngl)

% Vectores de fuerzas y desplazamientos conocidos
% -----------------------------------------------

% vector de fuerzas nodales
if fnod == 0
  f = zeros(ngl,1);
else
  [fil,col] = size(fnod);
  f = zeros(ngl,1);
  for i=1:fil
    f = ensamblarV(f,fnod(i,2:col)',grlib(fnod(i,1),:));
  end
end
fa = f(1:ngld,1);

% vector de fuerzas de empotramiento
if femp == 0
  f0 = zeros(ngl,1);
else
  [fil,col] = size(femp);
  f0 = zeros(ngl,1);
  for i=1:fil
```

```
      f0 = ensamblarV(f0,femp(i,2:col)',incid(femp(i,1),:));
    end
  end
  fa0 = f0(1:ngld,1);
  fb0 = f0(ngld+1:ngl,1);

  % vector de desplazamientos conocidos
  for i=1:nnud
    for j=1:glnud
      if restr(i,j) == 1
        db(grlib(i,j)-ngld,1) = restr(i,glnud+j);
      end
    end
  end

  % Solución del sistema de ecuaciones
  % ----------------------------------

  da = kaa^(-1) * (fa-fa0-kab*db)
  fb = fb0 + kba*da + kbb*db

  % fin del análisis
```