



UNIVERSIDAD
POLITÉCNICA
DE MADRID


ANÁLISIS DE ESTRUCTURAS A BAJA FRECUENCIA Y REDUCCIÓN MODAL

ESTRUCTURAS DE USO ESPACIAL

Autor: Daniel DEL RÍO VELILLA

Profesores: Elena ROIBÁS

MADRID, 14 DE JUNIO DE 2021

Repositorio del proyecto  <https://github.com/Danieldelriovelilla/EUE.git>

Índice

Índice de figuras	I
Índice de tablas	II
1. Introducción	1
2. Obtención de las matrices de masa y rigidez	2
3. Análisis espectral sin reducción modal	3
4. Método de síntesis de componentes, método de Craig-Bampton	5
5. Comparación de modelos	9
Referencias	11
A. Bandas de tercio de octava	12
B. Código	13
B.1. Método de elementos finitos	13
B.2. Craig-Bampton	14
B.3. Otras funciones implementadas	17
B.4. Programa principal	17

Índice de figuras

1.	Esquema del sistema formado por dos vigas acopladas mediante unión puntual.	1
2.	Modelo de las vigas en Patran.	2
3.	Matrices de rigidez de la Viga 1 generadas con Patran-Nastran (51×51).	3
4.	Matrices de rigidez de la Viga 2 generadas con Patran-Nastran (61×61).	3
5.	Valor RMS de la velocidad de las vigas sin aplicar métodos de reducción de 1 a 2000 Hz.	5
6.	Cantidad de frecuencias en cada banda para las vigas sin ensamblar.	8
7.	Valor RMS de la velocidad de las vigas aplicando el método de reducción de CB analizando de 1 a 2000 Hz.	9

Índice de tablas

1.	Propiedades de las vigas.	1
2.	Comparación de resultados variando el número de modos reducidos.	10
3.	Banda de tercios de octava [Hz].	12

1. Introducción

En este documento se recoge el proceso realizado para la aplicación del método Craig-Bampton de reducción modal en un modelo estructural sencillo de elementos finitos.

Se va a partir del sistema representado en la Figura 1. Consiste en dos vigas empotradas en sus extremos opuestos y unidas por el extremo libre. Además se aplica una carga vertical unitaria a una distancia de 0,25 m de la Viga 1 y se va a considerar amortiguamiento es nulo en ambas vigas.

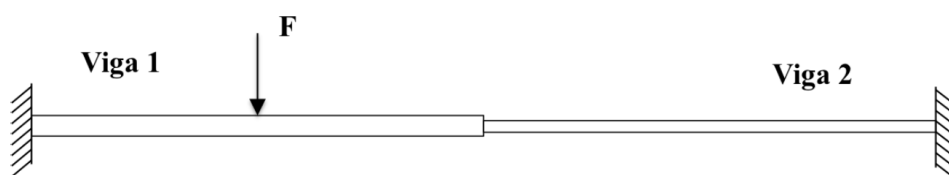


Figura 1: Esquema del sistema formado por dos vigas acopladas mediante unión puntual.

Cada una de las vigas se modelizará en Patran-Nastran utilizando las propiedades mostradas en la Tabla 1.

Tabla 1: Propiedades de las vigas.

Parámetro	Viga 1	Viga 2
Módulo elástico [Pa]	$2,4 \cdot 10^8$	$7 \cdot 10^{10}$
Coefficiente de Poisson	0,4	0,3
Densidad [kg/m ³]	$1,4 \cdot 10^3$	$1,1333 \cdot 10^4$
Longitud [m]	0.5	0.6
Radio sección [m]	$1 \cdot 10^{-3}$	$5 \cdot 10^{-4}$

Posteriormente se extraerán las matrices de masa y rigidez de cada viga con MATLAB y se aplicará el método de reducción modal. Y finalmente se compararán los resultados obtenidos de los modelos con y sin reducción modal.

Puesto que toda la parte que comprende al ensamblaje de matrices y resolución del sistema de ecuaciones sin aplicar reducción ya se ha realizado en otras asignaturas previas, no se explicará en profundidad el procedimiento empleado. Sin embargo, se indicará las funciones utilizadas para dicho propósito.

2. Obtención de las matrices de masa y rigidez

Para obtener las matrices de masa y rigidez se han modelado las vigas por separado en Patran con las propiedades de la Tabla 1. El tamaño de los elementos utilizados viene definido por el enunciado y es igual a 0.01 m, dando un total de 50 elementos (51 nodos) para la viga 1 y 60 elementos (61 nodos) para la viga 2.

La diferencia geométrica entre las dos vigas y la representación de los elementos se puede ver en la Figura 2.

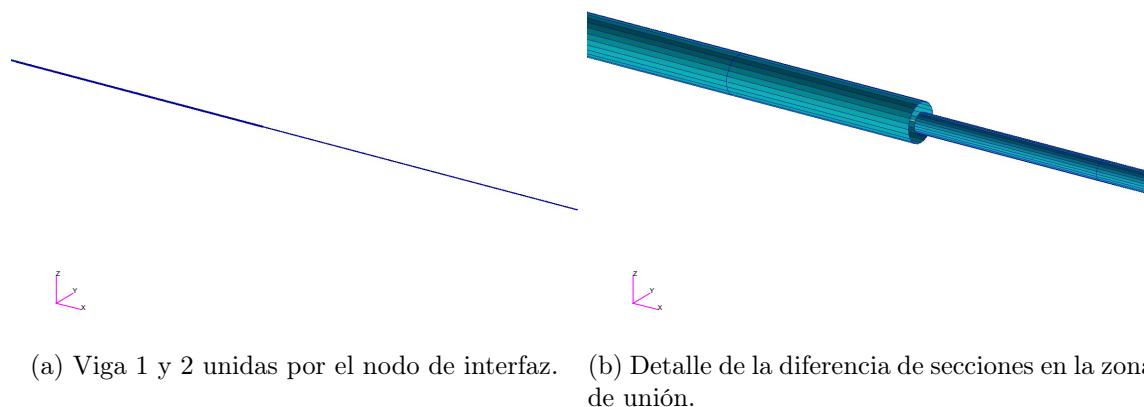


Figura 2: Modelo de las vigas en Patran.

Una vez modeladas las vigas se genera un caso de análisis modal sin imponer ninguna carga ni restricciones de desplazamiento.

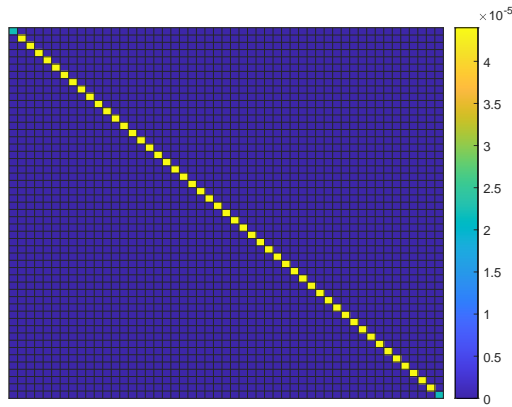
Ahora, para crear un archivo que contenga las matrices de rigidez y masa es necesario incluir en la sección *CASE CONTROL DECK* del archivo *.bdf* del modelo de cada viga el comando `EXTSEOUT(STIF, DAMP, MASS, EXTID=1, DMIGPCH)`.

Cuando se ejecute con Nastran el archivo *.bdf* una vez introducido el comando anterior este generará un fichero *punch*, *.pch*, que contendrá las matrices de interés.

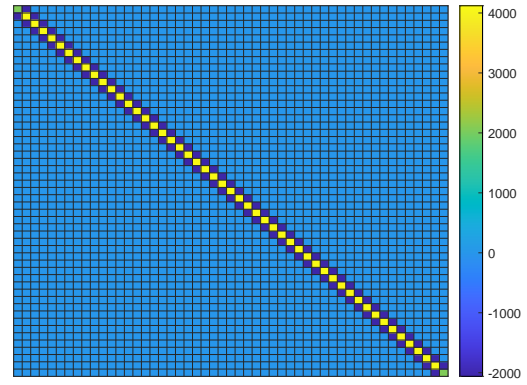
Para poder trabajar con las matrices en MATLAB es necesario extraerlas del fichero *punch*, para ello se nos ha facilitado en el Moodle de la asignatura la función `ReadPunchFile`.

Esta función tiene como argumentos de entrada la ruta del archivo *punch* del que se quieren extraer las matrices de rigidez y los grados de libertad de interés. En este caso, el único grado de libertad de interés ha sido el desplazamiento vertical, que corresponde con el número 3. Como argumentos de salida la función devuelve la matriz de masa, rigidez y un vector con la numeración de los nodos que forman el sistema.

Tras realizar todo este proceso se presentan las matrices de masa y rigidez de la viga 1 en la Figura 3 y de la viga 2 en la Figura 4. Además se ha creado un vector columna que representa las cargas nodales. Para la viga 1 tendrá la componente 26 (correspondiente con la distancia 0.25 m) de valor -1 y el resto igual a 0. En la viga 2 todas las componentes serán igual a 0.

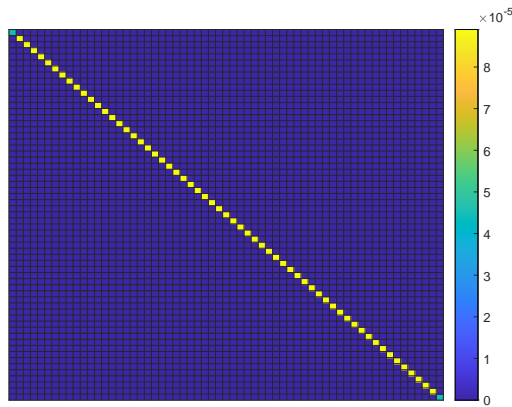


(a) Matriz de masas.

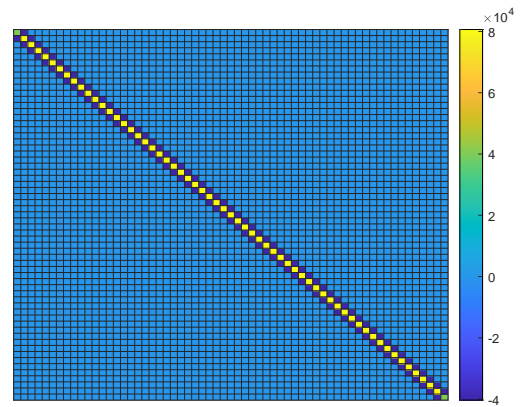


(b) Matriz de rigidez.

Figura 3: Matrices de rigidez de la Viga 1 generadas con Patran-Nastran (51×51).



(a) Matriz de masas.



(b) Matriz de rigidez.

Figura 4: Matrices de rigidez de la Viga 2 generadas con Patran-Nastran (61×61).

3. Análisis espectral sin reducción modal

En este primer análisis se va a calcular el valor de la velocidad cuadrática media en banda fina de frecuencias, de 1 a 2000 Hz con incrementos de 1 Hz.

Para poder tener una idea de la disminución tiempo de cómputo que se consigue utilizando métodos de reducción modal, se va a hacer este primer análisis con el método del elementos finitos sin aplicar ningún método de reducción. Esto consiste en el ensamblado de las dos vigas por el nodo de interfaz para construir una matrices del sistema de dos vigas y después resolver el siguiente sistema matricial para obtener la velocidad:

$$[M] \{\ddot{\vec{q}}(t)\} + [K] \{\vec{q}(t)\} = \vec{f}(t) , \quad (1)$$

donde $[M]$ es la matriz de masas, $[K]$ la matriz de rigidez, \vec{q} el vector de desplazamientos y \vec{f} el vector de fuerzas externas sobre el sistema completo.

Como la carga aplicada es oscilatoria, la respuesta también lo será, por lo que se puede hacer el siguiente cambio,

$$\begin{aligned} \vec{f} &= \vec{F} \cdot e^{i\omega t} , \\ \vec{q} &= \vec{Q} \cdot e^{i\omega t} \end{aligned}$$

que permite pasar al plano de la frecuencia, quedando el siguiente sistema a resolver,

$$-\omega^2[M] \{\vec{Q}\} + [K] \{\vec{Q}\} = (-\omega^2[M] + [K]) \{\vec{Q}\} = \vec{F} ; \quad (2)$$

cuya solución es,

$$\{\vec{Q}\} = (-\omega^2[M] + [K])^{-1} \vec{F} . \quad (3)$$

Una vez se haya obtenido el vector de desplazamientos se calcula el valor cuadrático medio de la velocidad con la siguiente expresión

$$\dot{q}_{rms} = \sqrt{\frac{1}{N} \sum_{i=1}^N (\omega \cdot q_i)^2} . \quad (4)$$

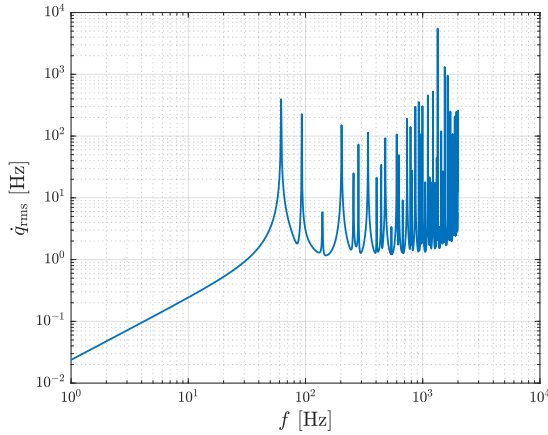
Las matrices de ambas vigas ya se tienen cargadas en MATLAB, por lo que hay que proceder a su ensamblaje para poder aplicar la ecuación (3).

El ensamblar las marices consiste en concatenar las dos matrices de rigidez y masa y sumar sus valores en las posiciones donde compartan nodos, llegando a un sistema en el que el número de nodos totales serán igual a la suma de ambos sistemas menos el número de nodos que compartan. El ensamblaje de las matrices está implementado en la función `f_Ensambar`, B.1.

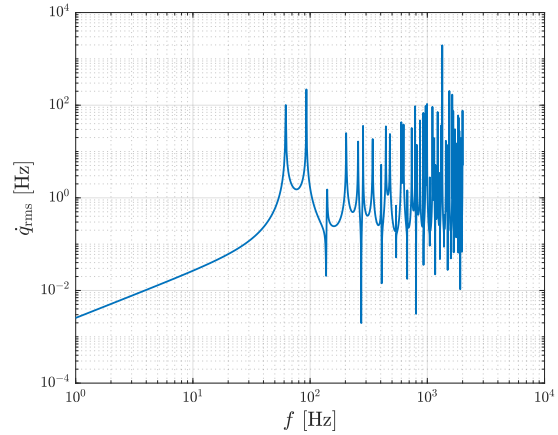
Una vez se tiene el sistema ensamblado, hay que eliminar las filas y columnas de las matrices de

masa y rigidez y las filas del vector de cargas correspondientes a las condiciones de contorno aplicadas, que en este caso es la de empotramiento. Esto lo realiza íntegramente la función `f_Solucion`, B.1.

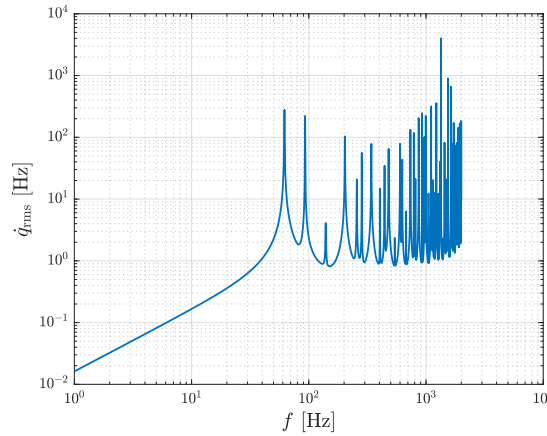
Los resultados de velocidades RMS, en escala logarítmica, para las vigas independientes y el sistema completo se pueden ver en las Figura 5.



(a) Velocidad RMS de la viga 1.



(b) Velocidad RMS de la viga 2.



(c) Velocidad RMS del sistema.

Figura 5: Valor RMS de la velocidad de las vigas sin aplicar métodos de reducción de 1 a 2000 Hz.

Resolver este sistema matricial ha llevado un tiempo medio tras 10 ejecuciones de 1.254 segundos.

4. Método de síntesis de componentes, método de Craig-Bampton

Los modelos de elementos finitos se basan en esencia en invertir matrices para resolver el sistema de la ecuación (3). Cuando las matrices son de tamaño elevado es conveniente la aplicación de algún método que permita reducir su tamaño para que no sea muy costoso desde el punto de vista computacional,

pero que mantenga la representación física del modelo. En este apartado se va a estudiar la aplicación del método de Craig-Bampton al sistema de vigas anteriormente resuelto.

Para aplicar este método es necesario reordenar los nodos de las vigas en nodos interiores y de frontera,

$$\{q\} = \{q_b q_i\} = \{q_b q_I q_i\} , \quad (5)$$

donde el subíndice b indica que es un nodo de frontera, I es nodo de interfaz, que se trata igual que los de frontera, e i nodo interior. Por lo tanto, el nodo 1 de la viga 1 y el 61 de la viga 2 serán los nodos de frontera de sus respectivas matrices, los nodos 51 y 1 serán los nodos de interfaz y el resto corresponderán a los nodos interiores.

Con este criterio se reorganizarán las matrices de masa y rigidez, quedando con la siguiente estructura,

$$\begin{bmatrix} M_{bb} & M_{bi} \\ M_{ib} & M_{ii} \end{bmatrix} \begin{Bmatrix} \ddot{q}_b \\ \ddot{q}_i \end{Bmatrix} + \begin{bmatrix} K_{bb} & K_{bi} \\ K_{ib} & K_{ii} \end{bmatrix} \begin{Bmatrix} q_b \\ q_i \end{Bmatrix} = \begin{Bmatrix} F_b \\ F_i \end{Bmatrix} \quad (6)$$

El método de Craig-Bampton propone clasificar los desplazamientos en modos estáticos y modos interiores elásticos con frontera fija, por lo que se obtendrían los desplazamientos de la siguiente forma,

$$\{q\} = [\phi_s] \{\eta_f\} + [\phi_i] \{\eta_i\} = [\phi_s, \phi_i] \begin{Bmatrix} \eta_f \\ \eta_i \end{Bmatrix} = [\psi] \{\eta\} . \quad (7)$$

- Cálculo de modos estáticos (de frontera):

Para el cálculo de estos modos se desprecian los efectos inerciales, se suponen nulas las fuerzas externas correspondientes a los nodos interiores

$$\begin{bmatrix} K_{bb} & K_{bi} \\ K_{ib} & K_{ii} \end{bmatrix} \begin{Bmatrix} q_b \\ q_i \end{Bmatrix} = \begin{Bmatrix} R_b \\ 0 \end{Bmatrix} \quad (8)$$

y finalmente se impone un desplazamiento unitario en cada grado de libertad de frontera,

$$\{q_i\} = -[K_{ii}]^{-1} [\hat{K}_{ib}] \{q_b\} \quad \text{con } \{q_b\} = [I] . \quad (9)$$

La transformación estática queda de la siguiente forma:

$$\{q\} = \begin{Bmatrix} q_b \\ q_i \end{Bmatrix} = \begin{bmatrix} I \\ -[K_{ii}]^{-1} [K_{ib}] \end{bmatrix} \{q_b\} = [\phi_s] \{q_b\} \quad (10)$$

- Cálculo de los modos elásticos interiores:

Se calculan las formas modales suponiendo fijos los grados de libertad de frontera,

$$\begin{aligned} | [K_{ii}] - \omega^2 [M_{ii}] | &= \{0\} \\ ([K_{ii}] - \omega^2 [M_{ii}]) [\phi_{ii}] &= \{0\} \end{aligned} \quad , \quad (11)$$

quedando así la transformación modal de los nodos interiores,

$$\{q\} = \begin{Bmatrix} q_b \\ q_i \end{Bmatrix} = \begin{bmatrix} 0 \\ [\phi_{ii}] \end{bmatrix} \{ \eta_i \} = [\phi_i] \{ \eta_i \} . \quad (12)$$

Finalmente, uniendo las matrices anteriores se obtiene la matriz de transformación al espacio de Craig-Bampton,

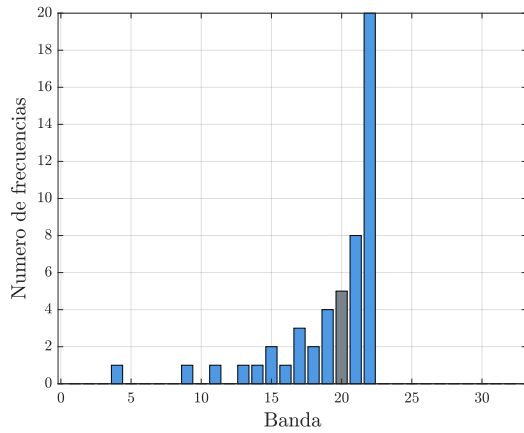
$$[\psi] = [\phi_s, \phi_i] = \begin{bmatrix} [I] & [0] \\ -[K_{ii}]^{-1} [K_{ib}] & [\phi_{ii}] \end{bmatrix} \quad (13)$$

donde las matrices de masa y rigidez en el espacio de Craig-Bampton serán

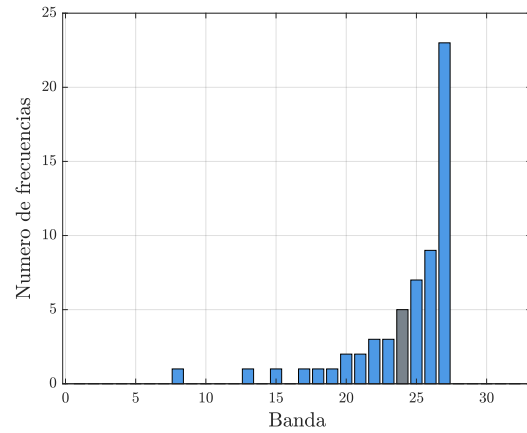
$$[M_{CB}] = [\psi]^t [M] [\psi] , \quad [K_{CB}] = [\psi]^t [K] [\psi] . \quad (14)$$

Este proceso se ha implementado en la función `f_CB B.2`.

Una vez que se tienen las matrices en el espacio de Craig-Bampton se ha calculado la cantidad de modos a reducir. El enunciado fija que es necesario retener hasta el valor máximo de la primera banda de tercios de octava, 3, en la que aparezcan al menos 5 modos. Así que, utilizando la función `f_Num_Reducir`, B.2, se obtienen la Figura 6 que muestra la última banda a retener.



(a) Última banda para la viga 1.



(b) Última banda para la viga 2.

Figura 6: Cantidad de frecuencias en cada banda para las vigas sin ensamblar.

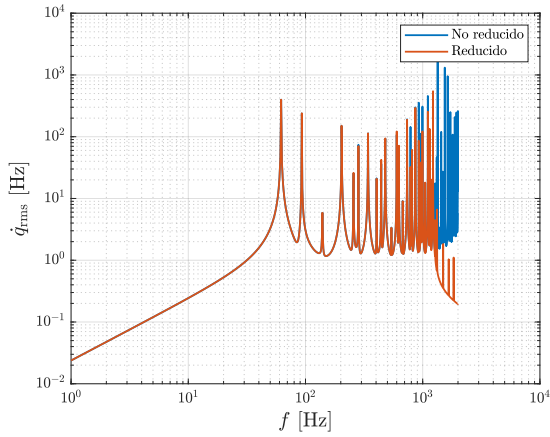
Por lo tanto, para la viga 1 se reducirán un total de 28 modos y para la viga 2 serán 39. Cuanto más rígida, mayor número de modos se reducen.

El ensamblaje de las matrices es igual al realizado en el apartado anterior, pero manteniendo el orden de nodos fijado para el espacio de Craig-Bampton. Esta tarea la realiza la función `f_Ensamblar_CB` B.2.

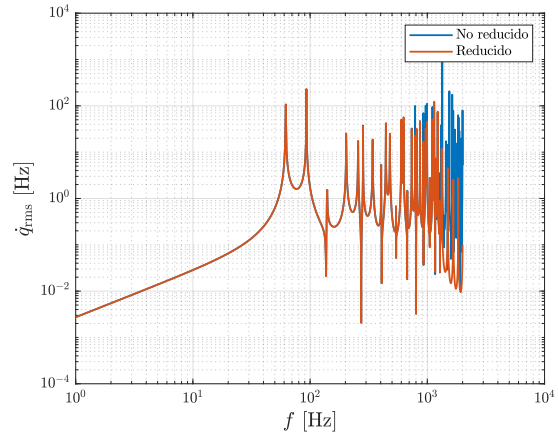
Habiendo ensamblado las matrices se procede a reducir el sistema, que consiste en eliminar las filas y columnas del sistema ensamblado correspondientes a los modos que se van a reducir. En el caso de la matriz $[\psi]$ se eliminan solo las columnas, para volver del espacio de Craig-Bampton conservando los grados de libertad totales. Una vez reducido el sistema se resuelve igual que se hizo en el método sin reducir. Todo este proceso se ha implementado en la función B.2.

Los resultados de velocidad cuadrática media de cada una de las vigas obtenidas con el método de Craig-Bampton se pueden ver en la Figura 7.

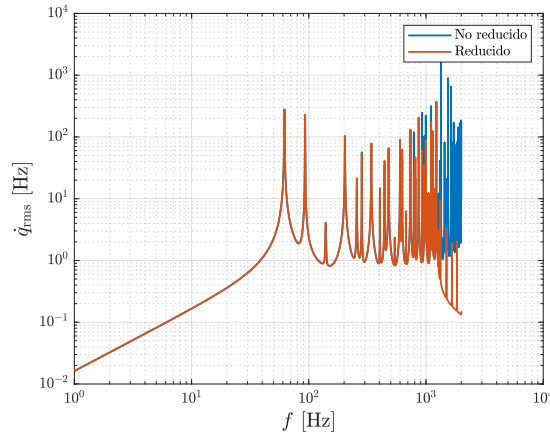
El tiempo medio de cálculo consumido para realizar tanto la reducción, como la resolución del sistema en 10 ejecuciones ha sido de 0.39 segundos.



(a) Velocidad RMS de la viga 1.



(b) Velocidad RMS de la viga 2.



(c) Velocidad RMS del sistema completo.

Figura 7: Valor RMS de la velocidad de las vigas aplicando el método de reducción de CB analizando de 1 a 2000 Hz.

5. Comparación de modelos

En la Figura 7 se puede ver de forma clara como el modelo reducido no es capaz de reproducir correctamente la respuesta de las vigas en la banda de frecuencias estudiada.

Para terminar con el estudio ha realizado una comparación entre distintas combinaciones de cantidad de modos reducidos. Esta comparación se muestra en la Tabla 2.

Tabla 2: Comparación de resultados variando el número de modos reducidos.

Modos reducidos		RMSE($\log_{10}(\dot{q})$)	Tiempo [s]
Viga 1	Viga 2		
0	0	0	1,08
10	0	0.0376	0.87
20	0	0.4464	0.70
30	0	0.7974	0.65
40	0	1.1296	0.93
0	10	0.0012	0.95
0	20	0.0133	0.71
0	30	0.0216	0.68
0	40	0.0844	0.68
10	40	0.0762	0.57
20	40	0.4524	0.53

Como era de esperar, la viga que necesita retener más modos es la menos rígida, la viga 1. El error cuadrático medio del logaritmo de la velocidad (misma representación que en las gráficas mostradas) aumenta conforme aumenta el número de modos eliminados, pero lo hace de forma más paulatina en la viga rígida. Como los tiempos de ejecución disminuyen de forma similar, esto indica a que en sistemas complejos es conveniente reducir los modos de los subsistemas más rígidos, ya que no afecta en gran medida al resultado, pero si al tiempo de cómputo total.

Además importante destacar que aún sin reducir el sistema, las matrices en el sistema de Craig-Bampton son mas sencillas de invertir que en el sistema convencional ya que el tiempo de computación necesario es inferior.

Referencias

- [1] M. Chimeno, Vibraciones y aeroacústica (2020).
- [2] E. Roibás, Ensayos y verificación del diseño (2020).
- [3] E. Roibás, Vibro-acústica en estructuras espaciales (2020).

A. Bandas de tercio de octava

Tabla 3: Banda de tercios de octava [Hz].

Banda	Lower	Central	Upper
1	14.1	16	17.8
2	17.8	20	22.4
3	22.4	25	28.2
4	28.2	31.5	35.5
5	35.5	40	44.7
6	44.7	50	56.2
7	56.2	63	70.8
8	70.8	80	89.1
9	89.1	100	112
10	112	125	141
11	141	160	178
12	178	200	224
13	224	250	282
14	282	315	355
15	355	400	447
16	447	500	562
17	562	630	708
18	708	800	891
19	891	1000	1122
20	1122	1250	1413
21	1413	1600	1778
22	1778	2000	2239
23	2239	2500	2818
24	2818	3150	3548
25	3548	4000	4467
26	4467	5000	5623
27	5623	6300	7079
28	7079	8000	8913
29	8913	10000	11220
30	11220	12220	14130
31	14130	16000	17780
32	17780	20000	22390

B. Código

B.1. Método de elementos finitos

- Función implementada para ensamblar matrices.

```
1 function [M, K, F] = f_Ensamblar(M1, K1, F1, M2, K2, F2, Ni)
2
3     gdl_1 = length(M1);
4     gdl_2 = length(M2);
5     % Inicializar matrices
6     M = zeros(gdl_1 + gdl_2 - 1);
7     K = zeros(gdl_1 + gdl_2 - 1);
8     F = zeros(gdl_1 + gdl_2 - 1, 1);
9
10    % Construir vector de posiciones
11    pos_1 = 1:gdl_1;
12    pos_2 = gdl_1:length(M);
13    pos_2(Ni(2)) = Ni(1);
14
15    % Ensamblar matrices
16    M(pos_1,pos_1) = M(pos_1,pos_1) + M1;
17    M(pos_2,pos_2) = M(pos_2,pos_2) + M2;
18
19    K(pos_1,pos_1) = K(gdl_1,gdl_1) + K1;
20    K(pos_2,pos_2) = K(pos_2,pos_2) + K2;
21
22    F(pos_1) = F(pos_1) + F1;
23    F(pos_2) = F(pos_2) + F2;
24
25 end
```

- Función que obtiene los modos propios de un sistema sin amortiguamiento.

```
1 function [mod_prop, frec_prop] = f_Modos(M, K, cc)
2
3     % Condiciones de contorno
4     M(cc,:) = []; M(:,cc) = [];
5     K(cc,:) = []; K(:,cc) = [];
6
7     % Obtencion de los modos y frecuencias propias^2
8     [mod_prop, frec_matrix] = eig( M\K );
9     frec_matrix = sqrt(frec_matrix);
10
```

```
11     frec_matrix = frec_matrix(sub2ind(size(frec_matrix),...  
12         1:size(frec_matrix,1),1:size(frec_matrix,2)));  
13  
14     [frec_prop,idx] = sort(frec_matrix);  
15     frec_prop = frec_prop/(2*pi);  
16     mod_prop = mod_prop(:,idx);  
17  
18 end
```

- Función que resuelve un modelo FEM sin amortiguamiento.

```
1 function [z, rms_z] = f_Solucion(M, K, F, cc, f)  
2  
3     % Condiciones de contorno  
4     M(cc(1:end),:) = []; M(:,cc(1:end)) = [];  
5     K(cc(1:end),:) = []; K(:,cc(1:end)) = [];  
6     F(cc(1:end)) = [];  
7  
8     % Resolver sistema  
9     w = 2*pi*f;  
10  
11     % Respuesta permanente  
12     z = (K - w^2*M)\F;  
13     z = [zeros(1,1); z; zeros(1,1)];  
14  
15     rms_z = [rms(z), rms(w*z), rms(w^2*z)];  
16  
17 end
```

B.2. Craig-Bampton

- Aplicación de Craig-Bampton.

```
1 function [M_CB, K_CB, F_CB, psi] = f_CB(M, K, F, Nb, NI)  
2  
3     % Orden de los gdl  
4     Ni = 1:length(M);  
5     Ni([Nb, NI]) = [];  
6     gdl_CB = [Nb, NI, Ni];  
7  
8     % Matrices de masa y rigidez ordenadas  
9     M = M(gdl_CB,gdl_CB);  
10    Mff = M(1:2,1:2);
```

```

11     Mii = M(3:end,3:end);
12
13     K = K(gdl_CB,gdl_CB);
14     Kif = K(3:end,1:2);
15     Kii = K(3:end,3:end);
16
17     F = F(gdl_CB);
18
19     % Matriz de transformacion CB
20     phi_s = [eye(size(Mff));...
21             -inv(Kii)*Kif];
22
23     cc = [];
24     [mod_prop, ~] = f_Modos(Mii, Kii, cc);
25     phi_i = [zeros(length(Mff),size(mod_prop,1)); mod_prop];
26
27     psi = [phi_s, phi_i];
28
29     M_CB = psi'*M*psi;
30     K_CB = psi'*K*psi;
31     F_CB = psi'*F;
32 end

```

- Cálculo del número de modos a reducir.

```

1 function red = f_Num_Reducir(frecs, Bandas, Nmin)
2
3     n = zeros(size(Bandas));
4     for b = 1:length(Bandas)
5         n(b) = length(frecs(frecs>Bandas(b).lower & frecs<Bandas(b).upper));
6     end
7
8     N = find(n>=Nmin);
9     N = N(1)+1;
10
11     red = length((find(frecs>Bandas(N).lower)));
12     h = figure();
13     b = bar(n, 'FaceColor', [77, 153, 230]/255);
14     b.FaceColor = 'flat';
15     b.CData(N-1,:) = [121, 131, 140]/255;
16     grid on
17     box on
18     xlabel('Banda', 'Interpreter', 'Latex')
19     ylabel('Numero de frecuencias', 'Interpreter', 'Latex')
20     Save_as_PDF(h, ['Figures/Bandas_' num2str(red)], '')

```

```
21 end
```

- Función que ensambla matrices con el orden requerido para Craig-Bampton.

```
1 function [M, K, F, psi] = f_Ensamblar_CB(M1, K1, F1, psi1, M2, K2, F2, psi2)
2
3     gdl_1 = length(M1);
4     gdl_2 = length(M2);
5     % Inicializar matrices
6     M = zeros(gdl_1 + gdl_2 - 1);
7     K = zeros(gdl_1 + gdl_2 - 1);
8     psi = zeros(gdl_1 + gdl_2 - 1);
9     F = zeros(gdl_1 + gdl_2 - 1, 1);
10
11     % Construir vector de posiciones
12     pos_1 = [1, 3:(gdl_1+1)];
13     pos_2 = [2:3, (pos_1(end)+1):length(M)];
14
15
16     % Ensamblar matrices
17     M(pos_1,pos_1) = M(pos_1,pos_1) + M1;
18     M(pos_2,pos_2) = M(pos_2,pos_2) + M2;
19
20     K(pos_1,pos_1) = K(gdl_1,gdl_1) + K1;
21     K(pos_2,pos_2) = K(pos_2,pos_2) + K2;
22
23     F(pos_1) = F(pos_1) + F1;
24     F(pos_2) = F(pos_2) + F2;
25
26     psi(pos_1,pos_1) = psi(gdl_1,gdl_1) + psi1;
27     psi(pos_2,pos_2) = psi(pos_2,pos_2) + psi2;
28
29 end
```

- Resolver el sistema con la reducción deseada de modos por Craig-Bampton.

```
1 function [z, z1, z2] = f_Solucion_CB(M, K, F, psi, cc, f, red_1, red_2)
2
3     % Condiciones de contorno
4     M(cc,:) = []; M(:,cc) = [];
5     K(cc,:) = []; K(:,cc) = [];
6     F(cc) = [];
7     psi(cc(1:end),:) = []; psi(:,cc(1:end)) = [];
8
```

```
9      % Reduccion modos
10     if not(red_1) && not(red_2)
11         red = [];
12     elseif not(red_1>1) && not(red_2)
13         red = (50-red_1):50;
14     elseif not(red_1) && not(red_2>1)
15         red = (109-red_2):109;
16     else
17         red = [(50-red_1):50 (109-red_2):109];
18     end
19
20     M(red,:) = []; M(:,red) = [];
21     K(red,:) = []; K(:,red) = [];
22     F(red) = [];
23     psi(:,red) = [];
24
25     % Resolver sistema
26     w = 2*pi*f;
27
28     % Respuesta permanente
29     z = [0; 0; psi*((K - w^2*M)\F)];
30
31     z1 = z([1 3:52]);
32     z2 = z([2:3 53:111]);
33
34 end
```

B.3. Otras funciones implementadas

- Función que calcular valores RMS de desplazamiento, velocidad y aceleración.

```
1 function [rms_z] = f_RMS(z, f)
2
3     % Resolver sistema
4     w = 2*pi*f;
5
6     % Valor RMS de posicion, velocidad y aceleracion
7     rms_z = [rms(z), rms(w*z), rms(w^2*z)];
8
9 end
```

B.4. Programa principal

```
1 clc
2 clear
3 close all
4
5 %% EXTRAER LAS MATRICES DE MASA Y RIGIDEZ
6
7 try
8     % Cargas las matrices pre-extraídas
9     load('Data/Vigas')
10    load('Data/Bandas')
11 catch
12    load('Data/Bandas')
13    % Path de los archivos .pch
14    path_v1 = "../Viga_1/Analisis/viga_1.pch";
15    path_v2 = "../Viga_2/Analisis/viga_2.pch";
16    path_v3 = "../Vigas/Analisis/parte_1.pch";
17
18    % Extraer las matrices de la viga 1
19    [M1,K1,N1] = ReadPunchFile_nD(path_v1,[3]);
20    gdl1 = 1:length(N1);
21
22    % Struct de la viga 1
23    viga_1.M = M1;
24    viga_1.K = K1;
25    viga_1.nodes = N1;
26    viga_1.gdl = gdl1;
27    viga_1.F = zeros(length(viga_1.gdl),1);
28    viga_1.F(26) = -1;
29
30    % Extraer las matrices de la viga 2
31    [M2,K2,N2] = ReadPunchFile_nD(path_v2,[3]);
32    gdl2 = 1:length(N2);
33
34    % Struct de la viga 2
35    viga_2.M = M2;
36    viga_2.K = K2;
37    viga_2.nodes = N2;
38    viga_2.gdl = gdl2;
39    viga_2.F = zeros(length(viga_2.gdl),1);
40
41    % Extraer las matrices de la viga 2
42    [M3,K3,N3] = ReadPunchFile_nD(path_v3,[3]);
43    gdl3 = 1:length(N3);
44
45    % Struct de la viga 3
46    viga_3.M = M3;
```

```
47     viga_3.K = K3;
48     viga_3.nodes = N3;
49     viga_3.gdl = gdl3;
50     viga_3.F = zeros(length(viga_3.gdl),1);
51     viga_3.F(26) = -1;
52
53     % Save data
54     save('Data/Vigas', 'viga_1', 'viga_2', 'viga_3')
55 end
56
57
58 %% VIGAS SIN CB
59
60 % Ensamblaje de matrices
61 Ni = [viga_1.gdl(end), viga_2.gdl(1)];
62 [vigas.M, vigas.K, vigas.F] = ...
63     f_Ensamblar(viga_1.M, viga_1.K, viga_1.F, viga_2.M, viga_2.K, viga_2.F, Ni);
64
65 % Matrices de Masa
66 h = figure();
67     heatmap(vigas.M);
68     snapnow
69     colormap default
70     Ax = gca;
71     Ax.XDisplayLabels = nan(size(Ax.XDisplayData));
72     Ax.YDisplayLabels = nan(size(Ax.YDisplayData));
73     set(h, 'Units', 'Inches');
74     pos = get(h, 'Position');
75     set(h, 'PaperPositionMode', 'Auto', 'PaperUnits', 'Inches', 'PaperSize', [pos(3), ...
76         pos(4)])
76     print(h, 'Figures/Vigas_M', '-dpdf', '-r0', '-painters')
77
78 % Matriz de Rigidez
79 h = figure();
80     heatmap(vigas.K);
81     snapnow
82     colormap default
83     Ax = gca;
84     Ax.XDisplayLabels = nan(size(Ax.XDisplayData));
85     Ax.YDisplayLabels = nan(size(Ax.YDisplayData));
86     set(h, 'Units', 'Inches');
87     pos = get(h, 'Position');
88     set(h, 'PaperPositionMode', 'Auto', 'PaperUnits', 'Inches', 'PaperSize', [pos(3), ...
89         pos(4)])
89     print(h, 'Figures/Vigas_K', '-dpdf', '-r0', '-painters')
90
91
```

```

92 % Solucion
93 cc = [1, length(vigas.M)];
94 [vigas.modos, vigas.frecuencias] = f_Modos(vigas.M, vigas.K, cc);
95
96 f = 1:2000;
97 len = 0:0.01:1.1;
98
99 % Inicializar variables
100 z = zeros(length(len), length(f));
101 rms_z = zeros(length(f), 3);
102 rms_z1 = zeros(51, 3);
103 rms_z2 = zeros(61, 3);
104
105 tic
106 for i = 1:length(f)
107     [z(:,i)] = f_Solucion(vigas.M, vigas.K, vigas.F, [1,length(vigas.F)], f(i));
108     z_1 = z(1:51,i);
109     z_2 = z(51:end,i);
110     [rms_z(i,:)] = f_RMS(z(:,i), f(i));
111     [rms_z1(i,:)] = f_RMS(z_1, f(i));
112     [rms_z2(i,:)] = f_RMS(z_2, f(i));
113 end
114 toc
115
116 h = figure();
117 loglog(f,rms_z(:,2), 'LineWidth', 1.)
118 xlabel('$f$ [Hz]', 'Interpreter', 'Latex')
119 ylabel('$\dot{q}_{\mathrm{rms}}$ [Hz]', 'Interpreter', 'Latex')
120 grid on; box on
121 Save_as_PDF(h, 'Figures/Clasico_vigas','')
122
123 h = figure();
124 loglog(f,rms_z1(:,2), 'LineWidth', 1.)
125 xlabel('$f$ [Hz]', 'Interpreter', 'Latex')
126 ylabel('$\dot{q}_{\mathrm{rms}}$ [Hz]', 'Interpreter', 'Latex')
127 grid on; box on
128 Save_as_PDF(h, 'Figures/Clasico_viga_1','')
129
130 h = figure();
131 loglog(f,rms_z2(:,2), 'LineWidth', 1.)
132 xlabel('$f$ [Hz]', 'Interpreter', 'Latex')
133 ylabel('$\dot{q}_{\mathrm{rms}}$ [Hz]', 'Interpreter', 'Latex')
134 grid on; box on
135 Save_as_PDF(h, 'Figures/Clasico_viga_2','')
136
137
138 %% MODOS DE VIGAS INDEPENDIENTES

```



```
139
140 cc = [1];
141 [viga_1.modos, viga_1.frecuencias] = f_Modos(viga_1.M, viga_1.K, cc);
142
143 cc = [61];
144 [viga_2.modos, viga_2.frecuencias] = f_Modos(viga_2.M, viga_2.K, cc);
145
146
147 %% CRAIIG-BAMPTON
148
149 % Viga 1
150 Nb_1 = 1;
151 NI_1 = viga_1.gdl(end);
152
153 [viga_1.M_CB, viga_1.K_CB, viga_1.F_CB, viga_1.psi] = ...
154     f_CB(viga_1.M, viga_1.K, viga_1.F, Nb_1, NI_1);
155
156 % Viga 2
157 Nb_2 = viga_2.gdl(end);
158 NI_2 = 1;
159
160 [viga_2.M_CB, viga_2.K_CB, viga_2.F_CB, viga_2.psi] = ...
161     f_CB(viga_2.M, viga_2.K, viga_2.F, Nb_2, NI_2);
162
163
164 % Ensamblar matrices en el espacio CB
165 Ni = [2, 2];
166 [vigas_CB.M, vigas_CB.K, vigas_CB.F, vigas_CB.psi] = ...
167     f_Ensamblar_CB(viga_1.M_CB, viga_1.K_CB, viga_1.F_CB, viga_1.psi, ...
168     viga_2.M_CB, viga_2.K_CB, viga_2.F_CB, viga_2.psi);
169
170
171 %% TODOS GDL
172
173 cc = [1,2];
174 rms_z_f = zeros(length(f), 3);
175 rms_z1_f = zeros(length(f), 3);
176 rms_z2_f = zeros(length(f), 3);
177 red_1 = 0;
178 red_2 = 0;
179
180 tic
181 for i = 1:length(f)
182     [z, z1, z2] = f_Solucion_CB(vigas_CB.M, vigas_CB.K, vigas_CB.F, vigas_CB.psi, ...
183         cc, f(i), red_1, red_2);
184     [rms_z_f(i,:)] = f_RMS(z, f(i));
185     [rms_z1_f(i,:)] = f_RMS(z1, f(i));
```

```

185     [rms_z2_f(i,:)] = f_RMS(z2, f(i));
186 end
187 toc
188
189 %% CG REDUCIDO
190 close all
191
192 cc = [1,2];
193 rms_z_r = zeros(length(f), 3);
194 rms_z1_r = zeros(length(f), 3);
195 rms_z2_r = zeros(length(f), 3);
196 red_1 = f_Num_Reducir(viga_1.frecuencias, Bandas, 5);
197 red_2 = f_Num_Reducir(viga_2.frecuencias, Bandas, 5);
198
199 tic
200 for i = 1:length(f)
201     [z, z1, z2] = f_Solucion_CB(vigas_CB.M, vigas_CB.K, vigas_CB.F, vigas_CB.psi, ...
202         cc, f(i), red_1, red_2);
203     [rms_z_r(i,:)] = f_RMS(z, f(i));
204     [rms_z1_r(i,:)] = f_RMS(z1, f(i));
205     [rms_z2_r(i,:)] = f_RMS(z2, f(i));
206 end
207 toc
208
209 h = figure();
210 loglog(f, rms_z_f(:,2), 'LineWidth', 1., 'DisplayName', 'No reducido')
211 hold on
212 loglog(f, rms_z_r(:,2), 'LineWidth', 1., 'DisplayName', 'Reducido')
213 grid on; box on;
214 legend('Interpreter', 'Latex', 'Location', 'NorthWest')
215 xlabel('$f$ [Hz]', 'Interpreter', 'Latex')
216 ylabel('$\dot{q}_{\mathrm{rms}}$ [Hz]', 'Interpreter', 'Latex')
217 grid on; box on
218 Save_as_PDF(h, 'Figures/CB_vigas_25', '')
219
220 h = figure();
221 loglog(f, rms_z1_f(:,2), 'LineWidth', 1., 'DisplayName', 'No reducido')
222 hold on
223 loglog(f, rms_z1_r(:,2), 'LineWidth', 1., 'DisplayName', 'Reducido')
224 grid on; box on;
225 legend('Interpreter', 'Latex', 'Location', 'NorthWest')
226 xlabel('$f$ [Hz]', 'Interpreter', 'Latex')
227 ylabel('$\dot{q}_{\mathrm{rms}}$ [Hz]', 'Interpreter', 'Latex')
228 grid on; box on
229 Save_as_PDF(h, 'Figures/CB_viga_1_25', '')
230

```

```

231 h = figure();
232 loglog(f,rms_z2_f(:,2), 'LineWidth', 1., 'DisplayName','No reducido')
233 hold on
234 loglog(f, rms_z2_r(:,2), 'LineWidth', 1., 'DisplayName','Reducido')
235 grid on; box on;
236 legend('Interpreter', 'Latex', 'Location', 'NorthWest')
237 xlabel('$f$ [Hz]', 'Interpreter', 'Latex')
238 ylabel('$\dot{q}_{\mathrm{rms}}$ [Hz]', 'Interpreter', 'Latex')
239 grid on; box on
240 Save_as_PDF(h, 'Figures/CB_viga_2_25', '')
241
242 rms(log10(rms_z_r(:,2))-log10(rms_z_f(:,2)))
243
244 %% REPRESENTACION MATRICES
245
246 % Masa viga 1
247 h = figure();
248 heatmap(viga_1.M);
249 snapnow
250 colormap default
251 Ax = gca;
252 Ax.XDisplayLabels = nan(size(Ax.XDisplayData));
253 Ax.YDisplayLabels = nan(size(Ax.YDisplayData));
254 set(h, 'Units', 'Inches');
255 pos = get(h, 'Position');
256 set(h, 'PaperPositionMode', 'Auto', 'PaperUnits', 'Inches', 'PaperSize', [pos(3), ...
    pos(4)])
257 print(h, 'Figures/VIga_1_M', '-dpdf', '-r0', '-painters')
258
259 % Rigidez viga 1
260 h = figure();
261 heatmap(viga_1.K);
262 snapnow
263 colormap default
264 Ax = gca;
265 Ax.XDisplayLabels = nan(size(Ax.XDisplayData));
266 Ax.YDisplayLabels = nan(size(Ax.YDisplayData));
267 % save figure as PDF
268 set(h, 'Units', 'Inches');
269 pos = get(h, 'Position');
270 set(h, 'PaperPositionMode', 'Auto', 'PaperUnits', 'Inches', 'PaperSize', [pos(3), ...
    pos(4)])
271 print(h, 'Figures/VIga_1_K', '-dpdf', '-r0', '-painters')
272
273 % Masa viga 2
274 h = figure();
275 heatmap(viga_2.M);

```

```
276     snapnow
277     colormap default
278     Ax = gca;
279     Ax.XDisplayLabels = nan(size(Ax.XDisplayData));
280     Ax.YDisplayLabels = nan(size(Ax.YDisplayData));
281     set(h, 'Units', 'Inches');
282     pos = get(h, 'Position');
283     set(h, 'PaperPositionMode', 'Auto', 'PaperUnits', 'Inches', 'PaperSize', [pos(3), ...
        pos(4)])
284     print(h, 'Figures/VIga_2_M', '-dpdf', '-r0', '-painters')
285
286 % Rigidez viga 1
287 h = figure();
288     heatmap(viga_2.K);
289     snapnow
290     colormap default
291     Ax = gca;
292     Ax.XDisplayLabels = nan(size(Ax.XDisplayData));
293     Ax.YDisplayLabels = nan(size(Ax.YDisplayData));
294     set(h, 'Units', 'Inches');
295     pos = get(h, 'Position');
296     set(h, 'PaperPositionMode', 'Auto', 'PaperUnits', 'Inches', 'PaperSize', [pos(3), ...
        pos(4)])
297     print(h, 'Figures/VIga_2_K', '-dpdf', '-r0', '-painters')
```