Laboratorio de Transferencia de Tecnología CAU:

IDENTIFICACIÓN DINÁMICA LINEAL DE SISTEMAS MECÁNICOS VÍA CONTROLADORES ALGEBRAICOS UNIVERSALES

Prof. Dr. Fredy Vides
Scientific Computing Innovation Center, UNAH &
Centre for Analysis of Data-Driven Systems
E-mail: fredyvides@caddslab.com

ÍNDICE

OBJETIVOS

1. Clasificar modelos dinámicos estructurales planares en términos de sus caracteríticas mecánicas elementales.
2. Identificar el modelo dinámico lineal planar que mejor describe la deformación dinámica de un conjunto de nodos en una estructura dada, con a lo más dos grados de libertad por nodo.

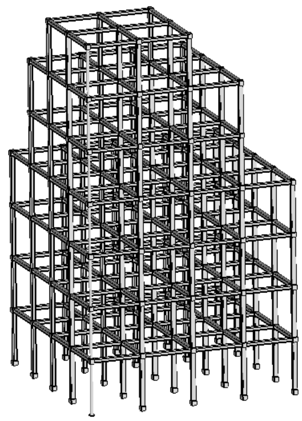


FIGURA 1.1. Estructura de acero de 7 niveles sujeta a cargas externas.

3. Calcular numéricamente de forma eficiente la deformación dinámica aproximada de un conjunto de nodos en una estructura dada, con a lo más dos grados de libertad por nodo.

1. DINÁMICA ESTRUCTURAL EN CONJUNTOS DE NODOS CON DOS GRADOS DE LIBERTAD

1.1. Identificación de sistemas dinámicos estructurales. Considerando una estructura como la mostrada en la figura 1.1. El problema de identificación dinámica estructural lineal **IDEL** es un problema de ingeniería inversa, que consiste en calcular el modelo matricial dinámico óptimo en el sentido de los mínimos cuadrados, que describe el comportamiento dinámico de la estructura en la figura 1.1, con base en secuencias de datos correspondientes al comportamiento dinámico observado de la estructura, medido a través de sensores localizados en puntos de interés específicos en la estructura.
En particular, el problema problema de identificación dinámica estructural puede interpretarse como la construcción de un dispositivo descriptor-predictor electrónico/digital $\mathfrak{S}$, que permite calcular/predecir el estado futuro x_{t+1} del sistema mecánico en estudio, en términos del estado presente x_t, de acuerdo al diagrama (1.1).

$$x_t \longrightarrow \boxed{\quad \mathfrak{S} \quad} \longrightarrow x_{t+1} \tag{1.1}$$

Los modelos dinámicos que estudiaremos en este laboratorio son de la forma.

$$\begin{cases} x_{t+1} = Ax_t + c \\ y_t = Sx_t \\ x_t = \arg\min \|y - X_t\|_2 \end{cases}, \quad t \in \mathbb{Z}^+, X_t \in X([0,T]), T > 0 \tag{1.2}$$

donde $\{X_t\}_{t\geq 1}^m$ es una muestra de tamaño m en una serie de tiempo $X([0,T]), T > 0$, de datos generados por una colección de sensores localizados en puntos de interés en la estructura como la de la figura 1.1.

Los modelos de matriciales de la forma (1.2) serán calculados aplicando los algoritmos y técnicas desarrollados por F. Vides en [3].

2. Modelos Dinámicos Planares Lineales Basados en Señales Mecánicas Lineales

2.1. Modelos mecánicos dinámicos lineales. Considerando el sistema mecánico determinado por la deflexión de un nodo de interés en la estructura de la figura 1.1, cuya dinámica es aproximadamente descrita por la ecuación diferencial:

$$m\frac{d^2x}{dt^2}+\delta\frac{dx}{dt}+\omega^2 x=0 \tag{2.1}$$

donde se asume que $m,\omega,\delta\in\mathbb{R}$, $m>0$ y $4m\omega^2-\delta^2>0$
Es posible aplicar reducción de orden para verificar que el sistema (2.1) es equivalente al sistema de ecuaciones diferenciales de primer orden.

$$\frac{d}{dt}\begin{bmatrix}x\\y\end{bmatrix}=\begin{bmatrix}0 & 1\\ -\frac{\omega^2}{m} & -\frac{\delta}{m}\end{bmatrix}\begin{bmatrix}x\\y\end{bmatrix} \tag{2.2}$$

El sistema (2.2) puede estudiarse utilizando el siguiente sistema genérico.

$$\frac{d}{dt}\begin{bmatrix}x\\y\end{bmatrix}=\begin{bmatrix}0 & 1\\ -\alpha & -\beta\end{bmatrix}\begin{bmatrix}x\\y\end{bmatrix} \tag{2.3}$$

para dos parámetros adimensionales α,β.
Es posible aplicar GNU Octave para resolver (2.3) para las condiciones iniciales $x(0)=1$, $y(0)=x'(0)=0$. Aplicando un método numérico de cuarto orden para calcular la curva solución aproximada $\hat{X}([0,10])\approx X([0,10])$ correspondiente a la solución $X(t)$ de (2.1), con un orden de error relativo $\epsilon_r=\mathcal{O}(10^{-16})$, para las condiciones iniciales $x(0)=1, y(0)=0$, y para los valores paramétricos $m=10,\omega=\sqrt{50},\delta=0$.
A continuación se presenta el código del programa `MechanicalSystem.m` en GNU Octave, que puede aplicarse para calcular $\hat{X}([0,10])$ y generar un subconjunto del campo vectorial que controla la dinánica del sistema mecánico (2.1) con base en el modelo dinámico genérico (2.3).

```
% Copyright (C) 2020 Fredy Vides
% function [t,x,A,C,Th]=MechanicalSystem(m,k,d,T,x0)
% A mechanical system simulation

% Example: [t,x,A,C,Th]=MechanicalSystem(10,50,0,10,[1,0]);

% Author: Fredy Vides <fredy@HPCLAB>
% Scientific Computing Innovation Center
% Created: 2020-05-03
function [t,x,A,C,Th]=MechanicalSystem(m,k,d,T,x0)
  Fx=@(x,y)y;
  Fy=@(x,y)(1/m)*(-k*x+d*y);
  f=@(t,y)[y(2);(1/m)*(-k*y(1)+d*y(2))];
  opt=odeset ("RelTol", eps);
  [t,x] = ode45 (f, [0, T], x0,opt);
  cx=norm(max(abs(x)));
```

```
  [X,Y]=meshgrid(-cx:2*cx/30:cx);
  quiver(X,Y,Fx(X,Y),Fy(X,Y),'k');
  hold on;
  plot(x(:,1),x(:,2),'k','markersize',12);
  hold off;
  axis equal;
  N=size(x,1);
  h=T/N;
  C=[0 1;-k/m d/m];
  p=(h.^(4:-1:0))./(factorial(4:-1:0));
  A=polyvalm(p,C);
  Th=expm(h*C);
end
```

Es posible aplicar el programa `MechanicalSystem.m` para simular una serie de tiempo de deflexiones aproximadas $\hat{X}([0,10]) = \{x_1, \ldots, x_T\}$, para el elemento estructural representado por un nodo genérico de dos grados de libertad.
Una vez que se ha escrito el programa y se ha guardado en un directorio de ruta simple (sin dejar espacios entre los nombres de los sub-directorios), es posible utilizar el icono  para asegurarse de que el Directorio Actual/Current Directory de GNU Octave sea el directorio donde se guardó el archivo de programa `MechanicalSystem.m`.
Se puede ejecutar el programa `MechanicalSystem.m` en la ventana de comandos de GNU Octave con la siguiente secuencia de comandos:

```
>> [t,x,A,C,Th]=MechanicalSystem(10,50,0,10,[1,0]);
```

2.2. Cómputo de controladores universales para sistemas dinámicos estructurales. Ahora es posible desarrollar un método numérico que permita calcular una representación matricial de un controlador algebraico universal (**UAC**) con tres grados de libertad iniciales, basado en la muestra $\hat{X}([0,10])$ de la serie de tiempo $\hat{X}([0,T]), T > 0$, generada por el programa `MechanicalSystem.m`, en el sentido de [2], aplicando las técnicas presentadas en [2,3].
Es posible escribir un programa GNU Octave que permite calcular la representación matricial del UAC (estabilizado aproximado) del sistema. La representación matricial del UAC producirá un par (T_h, f) donde $T_h \in \mathbb{R}^{2\times 2}$ y $f \in \mathbb{R}^n$ cumplen (además de otras condiciones genéricas estudiadas en [2,3]) las restricciones:

$$\left\{ \begin{array}{l} x_{t+1} = T_h x_t + f \\ x_t = \arg\min \|y - X_t\|_2 \end{array} \right. , \quad t \in \mathbb{Z}^+, X_t \in X([0,T]), T > 0 \tag{2.4}$$

Los grados de libertad en este contexto se refieren a que en la matriz inicial $A \in \mathbb{R}^{2\times 2}$ generada por el algoritmo, aunque las cuatro entradas están restringidas por la información dinámica estructural del sistema, solo tres entradas son libres para tomar valores factibles no preestablecidos, la matriz estructurada A es luego **estabilizada** para generar la matriz T_h, de tal manera que el sistema (2.4) sea **localmente estable** en el sentido de sistemas dinámicos lineales de tiempo discreto, es decir, la deflexión de los elementos estructurales se mantiene dentro del margen de tolerancia estableciado para la estructura o sistema estructural en estudio, durante un periodo de tiempo determinado. El código del programa

`TMatrixID.m` de GNU Octave que calcula el par (T_h, f) que cumple las restricciones (2.4), se muestra a continuación:

```
% Copyright (C) 2020 Fredy Vides
% function [A,Ap,f]=TMatrixID(x,m)
% A universal algebraic controller
% comptutation
% This method is based on the UAC method
% presented by F. Vides in the article:
% "Universal algebraic controllers and
% system identification"

% See also: MechanicalSystemID.
% Author: Fredy Vides <fredy@HPCLAB>
% Scientific Computing Innovation Center
% Created: 2020-05-03
function [A,Ap,f]=TMatrixID(x,m)
  N=size(x,1);
  m=min([m N-1]);
  x0=x(1:m,1);
  x1=x(2:(m+1),1);
  y0=x(1:m,2);
  y1=x(2:(m+1),2);
  E=ones(m,1);
  c1=[y0 E]\(x1-x0);
  c2=[x0 y0 E]\y1;
  A=[1 c1(1);c2(1:2).'];
  f=[c1(2);c2(3)];
  [v,a]=eig(A);
  a=diag(a);
  a=diag(a./abs(a));
  Ap=real(v*a/v);
end
```

Aplicando el programa `TMatrixID.m`, calcular el par UAC (T_h, f) para el sistema mecánico, basado en la muestra $\hat{X}([0, 10])$, para un tamaño de submuestreo $m = 120$, ejecutando el programa `TMatrixID.m` en la ventana de comandos de GNU Octave utilizando la siguiente secuencia de comandos.

```
>> [A,Th,f]=TMatrixID(x,120)
```

2.3. Identificación lineal de dinámica estructural basada en señales mecánicas lineales. Ahora se puede desarrollar un algoritmo de identificación que permite calcular el modelo lineal UAC de tres grados de libertad que mejor aproxima la dinámica del sistema mecánico en el sentido de mínimos cuadrados (con respecto a la métrica Euclidiana d_2 en $\mathbb{R}^2$), con base en la muestra $\hat{X}([0, 10])$ de la serie de tiempo $\hat{X}([0, T]), T > 0$. Es posible escribir un programa en GNU Octave que permite implementar un algoritmo identificación para sistemas basados en señales mecánicas, con base en modelos genéricos de la forma (2.3). El código del programa `MechanicalSystemID.m` para la identificación de sistemas mecánicos estructurales lineales se muestra a continuación.

```
% Copyright (C) 2020 Fredy Vides
% function [Th,Thp,C]=MechanicalSystemID(m)
% A sytem identificacion computational
% implementation

% Example: [Th,Thp,C]=MechanicalSystemID(120)

% Author: Fredy Vides <fredy@HPCLAB>
% Scientific Computing Innovation Center
% Created: 2020-05-03
function [Th,Thp,C]=MechanicalSystemID(m)
[t,x,A,C,Th]=MechanicalSystem(10,50,0,10,[1,0]);
hold on;
[Ap,Thp,f]=TMatrixID(x,m);
N=size(x,1);
Yp=[1;0];
Y=Yp;
for k=1:3*N
    Y=[Y Th*Y(:,k)];
    Yp=[Yp Thp*Yp(:,k)+f];
end
plot(x(:,1),x(:,2),'k');
plot(Y(1,:),Y(2,:),'b.-','markersize',12);
plot(Yp(1,:),Yp(2,:),'r.-','markersize',12);
axis equal;
hold off;
end
```

Se puede aplicar el programa `MechanicalSystemID.m` para calcular y visualizar diversas identificaciones aproximadas de sistemas mecánicos estructurales de la forma (2.3), correspondientes a diferentes valores de submuestreo m. En particular para $m = 120$, se pueden utilizar las siguientes secuencias de comandos para calcular la identificación del sistema correspondiente.

```
>> [Th,Thp,C,T,Tp]=MechanicalSystemID(120);
```

La salida gráfica producida por el programa se muestra en la figura 2.1.

3. Modelos Dinámicos Planares Lineales Basados en Señales Mecánicas No Lineales

3.1. Modelos mecánicos dinámicos no lineales. Considerando el sistema mecánico determinado por la deflexión de un nodo de interés en la estructura de la figura 1.1, cuya dinámica es aproximadamente descrita por el sistema genérico (3.1),

$$\frac{d}{dt}\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} y & \\ -\alpha x & -\beta x^3 \end{bmatrix} \tag{3.1}$$

para dos parámetros adimensionales α, β.

Es posible aplicar GNU Octave para resolver (2.3) para las condiciones iniciales $x(0) = 1$, $y(0) = x'(0) = 0$. Aplicando un método numérico de cuarto orden para calcular la curva

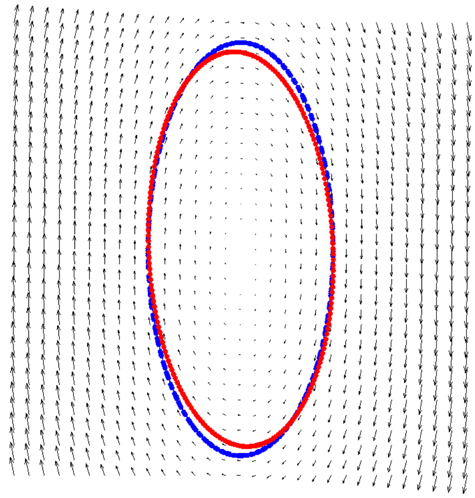


FIGURA 2.1. Salida gráfica producida por el algoritmo de identificación para $m = 120$.

solución aproximada $\hat{X}([0,10]) \approx X([0,10])$ correspondiente a la solución $X(t)$ de (2.1), con un orden de error relativo $\epsilon_r = \mathcal{O}(10^{-16})$, para las condiciones iniciales $x(0) = 1, y(0) = 0$, y para los valores paramétricos $m = 10, \omega = \sqrt{50}, \delta = 0$.
A continuación se presenta el código del programa `NLMechanicalSystem.m` en GNU Octave, que puede aplicarse para calcular $\hat{X}([0,10])$ y generar un subconjunto del campo vectorial que controla la dinánica del sistema mecánico (3.1).

```
% Copyright (C) 2020 Fredy Vides
% function [t,x]=NLMechanicalSystem(m,k,d,T,x0)
% A mechanical system simulation

% Example: [t,x]=NLMechanicalSystem(10,50,0,10,[1,0]);

% Author: Fredy Vides <fredy@HPCLAB>
% Scientific Computing Innovation Center
% Created: 2020-05-03
function [t,x]=NLMechanicalSystem(m,k,d,T,x0)
  Fx=@(x,y)y;
  Fy=@(x,y)(1/m)*(-k*x+d*x.^3);
  f=@(t,y)[y(2);(1/m)*(-k*y(1)+d*y(1).^3)];
  opt=odeset ("RelTol", eps);
  [t,x] = ode45 (f, [0, T], x0,opt);
  cx=norm(max(abs(x)));
  [X,Y]=meshgrid(-cx:2*cx/30:cx);
  quiver(X,Y,Fx(X,Y),Fy(X,Y),'k');
  hold on;
  plot(x(:,1),x(:,2),'k','markersize',12);
  hold off;
  axis equal;
```

```
end
```

Es posible aplicar el programa `NLMechanicalSystem.m` para simular una serie de tiempo de deflexiones aproximadas $\hat{X}([0,10]) = \{x_1, \ldots, x_T\}$, para el elemento estructural representado por un nodo genérico de dos grados de libertad.
Una vez que se ha escrito el programa y se ha guardado en un directorio de ruta simple (sin dejar espacios entre los nombres de los sub-directorios), es posible utilizar el icono  para asegurarse de que el Directorio Actual/Current Directory de GNU Octave sea el directorio donde se guardó el archivo de programa `MechanicalSystem.m`.
Se puede ejecutar el programa `NLMechanicalSystem.m` en la ventana de comandos de GNU Octave con la siguiente secuencia de comandos:

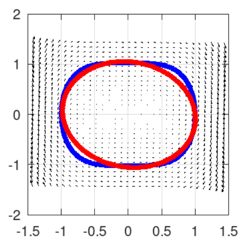
```
>> [t,x]=NLMechanicalSystem(10,50,0,10,[1,0]);
```

3.2. Identificación lineal de dinámica estructural basada en señales mecánicas no lineales. Aplicando el programa Octave `TMatrixID.m` desarrollado en §2.2, se puede desarrollar un algoritmo de identificación que permite calcular el modelo lineal UAC de tres grados de libertad que mejor aproxima la dinámica del sistema mecánico en el sentido óptimo de mínimos cuadrados, con base en la muestra $\hat{X}([0,10])$ de la serie de tiempo $\hat{X}([0,T]), T > 0$. Es posible escribir un programa en GNU Octave que permite implementar un algoritmo identificación para sistemas basados en señales mecánicas, con base en modelos genéricos de la forma (3.1). El código del programa `NLMechanicalSystemID.m` para la identificación de sistemas mecánicos estructurales lineales se muestra a continuación.

```
% Copyright (C) 2020 Fredy Vides
% function [Thp,f,t,x,Y]=NLMechanicalSystemID(m)
% A sytem identificacion computational
% implementation

% Example: [Thp,f,t,x,Y]=NLMechanicalSystemID(180)

% Author: Fredy Vides <fredy@HPCLAB>
% Scientific Computing Innovation Center
% Created: 2020-05-03
function [Thp,f,t,x,Yp]=NLMechanicalSystemID(m)
subplot(211);
[t,x]=NLMechanicalSystem(2,.1,-4,30,[1,0]);
hold on;
[Ap,Thp,f]=TMatrixID(x,m);
N=size(x,1);
Yp=[1;0];
for k=1:(N-1)
    Yp=[Yp Thp*Yp(:,k)+f];
end
plot(x(:,1),x(:,2),'b.-','markersize',12);
plot(Yp(1,:),Yp(2,:),'r.-','markersize',12);
axis square;
grid on;
```



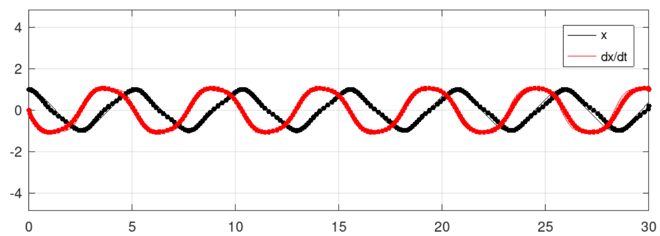


FIGURA 3.1. Salida gráfica producida por el algoritmo de identificación para un tamaño de submuestreo de la serie de tiempo del sistema de $m = 140$. La sub-figura superior muestra el diagrama de fase aproximado del sistema mecánico. La sub-figura inferior muestra las gráficas de posición y velocidad correspondientes.

```
hold off;
subplot(212);
plot(t,x(:,1),'k',t,x(:,2),'r');
hold on;
plot(t,Yp(1,:),'k.-','markersize',12);
plot(t,Yp(2,:),'r.-','markersize',12);
legend('x','dx/dt')
axis equal;
grid on;
end
```

Se puede aplicar el programa `NLMechanicalSystemID.m` para calcular y visualizar diversas identificaciones aproximadas de sistemas mecánicos estructurales de la forma (3.1), correspondientes a diferentes valores de submuestreo m. En particular para $m = 140$, se pueden utilizar las siguientes secuencias de comandos para calcular la identificación del sistema correspondiente.

```
>> [Thp,Tp,t,x,Y]=NLMechanicalSystemID(140);
```

La salida gráfica producida por el programa se muestra en la figura 3.1.

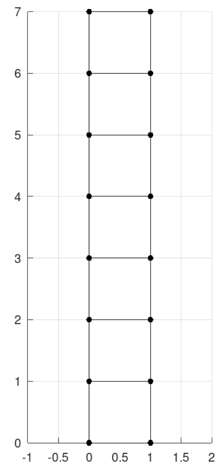


FIGURA 4.1. Subestructura $\mathcal{E}$ de orden reducido basada en una configuración de sensores predeterminada.

4. MODELOS DINÁMICOS PLANARES LINEALES MULTIVARIADOS DE DOS GRADOS DE LIBERTAD

Consideremos una subestructura $\mathcal{E}$ de orden reducido de una estructura como la de la figura 1.1, la cual está basada en una configuración de sensores que puede ser representada por la figura 4.1

Si la dinámica de $\mathcal{E}$ está aproximadamente determinada por un modelo matricial de la forma:

$$\mathbf{M}\frac{d^2\mathbf{x}}{dt^2} = \mathbf{K}\mathbf{x} + \mathbf{D}\frac{d\mathbf{x}}{dt}, \ \ \mathbf{x}(0) = \mathbf{x}_0, \ \ \frac{d\mathbf{x}(0)}{dt} = \mathbf{v}_0 \tag{4.1}$$

donde $\mathbf{M}$ es una matriz de masa simétrica positiva definida. Es posible reducir el modelo (4.1) a una expresión de la forma.

$$\frac{d\mathbf{y}}{dt} = \mathbf{H}\mathbf{y}, \ \ \mathbf{y}(0) = \mathbf{y}_0 \tag{4.2}$$

donde $\mathbf{H}, \mathbf{y}, \mathbf{y}_0$ están definidos por las siguientes expresiones.

$$\mathbf{y} = \begin{bmatrix} \mathbf{x} \\ \frac{d\mathbf{x}}{dt} \end{bmatrix}$$

$$\mathbf{y_0} = \begin{bmatrix} \mathbf{x}_0 \\ \mathbf{v}_0 \end{bmatrix}$$

$$\mathbf{H} = \begin{bmatrix} 0 & I \\ -\mathbf{M}^{-1}\mathbf{K} & -\mathbf{M}^{-1}\mathbf{D} \end{bmatrix}$$

La solución de (4.2) está determinada por la expresión.

$$\mathbf{y}(t) = e^{t\mathbf{H}}\mathbf{y}_0 \tag{4.3}$$

En esta sección, en lugar de buscar una solución analítica (aproximada) directa para simular la dinámica de $\mathcal{E}$, aplicaremos la tecnología de controlarores algebraicos universales (UAC) desarrollada en [3] para calcular un modelo computacional de la forma (1.1),(1.2).

Observación. *La estructura matricial del modelo* (4.2) *produce algunas restricciones estructurales en los modelos dinámicos de la forma* (4.3). *En particular, si* $\mathbf{D} = 0$*, las matrices* $e^{t\mathbf{H}}$ *cumplen con las siguientes restricciones.*

$$\left(e^{t\mathbf{H}}\right)^\top Z e^{t\mathbf{H}} = Z,\ \ t \geq 0 \tag{4.4}$$

Donde Z *es la matriz determinada por la expresión.*

$$Z = \begin{bmatrix} 0 & I \\ -I & 0 \end{bmatrix} \tag{4.5}$$

4.1. Identificación de sistemas dinámicos con base en señales mecánicas externas. Considerando nuevamente la subestructura $\mathcal{E}$ en el medio continuo $\mathbb{R}^2$ en la figura 4.1. Es posible simular le dinámica aproximada/esperada original de $\mathcal{E}$ con el programa de GNU Octave `StructuralDynamicalSystem.m` cuyo código se muestra a continuación.

```
% Copyright (C) 2020 Fredy Vides
% function [t,x,A]=StructuralDynamicalSystem(N,m,d,T,x0,v0)
% A mechanical system simulation

% Example: [t,x,A]=StructuralDynamicalSystem(4,100,0,5,...
% [0,0,0,.1,0,0,0,.1],[0,0,0,0,0,0,0,0]);

% Author: Fredy Vides <fredy@HPCLAB>
% Scientific Computing Innovation Center
% Created: 2020-05-03
function [t,x,A]=StructuralDynamicalSystem(N,m,d,T,x0,v0)
  ht=T/m;
  t=0:ht:T;
  x=[x0,v0];
  A=spdiags(ones(N,1)*[10 -20 10],-1:1,N,N);
  A(N,N)=A(N,N)/2;
  Ce=abs(max(eig(A)));
  E=eye(N);
  A=[-A,-Ce*E;-Ce*E,-A];
  E=eye(2*N);
  Z=zeros(2*N);
  Z=[Z E;-E Z];
  [v,a]=eig(full(A));
  lap=diag((d+sqrt(d^2-4*diag(a)))/2);
  lam=diag((d-sqrt(d^2-4*diag(a)))/2);
  z=[v v;v*lap v*lam];
  lap=diag(lap);
  lam=diag(lam);
  A=real(z*diag([exp(ht*lap);exp(ht*lam)])/z);
  if d==0
```

```
A=(A-Z*inv(A).'*Z)/2;
  end
  E=A.';
  for k=1:m
    x=[x;x(k,:)*E];
  end
end
```

4.1.1. Sistemas dinámicos estructurales disipativos. Podemos aplicar el programa GNU Octave `StructuralDynamicalSystem.m` para calcular una muestra de señales mecánicas correspondientes a deflexiones y velocidades de deflexión del sistema estructural $\mathcal{E}(t)$ bajo condiciones disipativas, para una secuencia $\{t_k\}_{k=1}^{N}$ de tiempos de muestreo. Para generar la muestra de señales mecánicas antes mecionada y almacenarla en memoria como un archivo en formato `*.csv`, podemos aplicar la siguiente secuencia de comandos en GNU Octave.

```
>> L=7;
>> [t,x,A]=StructuralDynamicalSystem(L,200,-1,20,...
.1*[1 zeros(1,L-1) 1 zeros(1,L-1)],zeros(1,2*L));
>> csvwrite ('TXdata.csv', [t.',x]);
```

Con base en señales mecánicas almacenadas en formato `*.csv`, es posible identificar la dinámica de los sistemas mecánicos correspondientes a estas señales aplicando programas basados en algoritmos UAC como el programa `BuildingDynamicsA.m` cuyo código se muestra a continuación.

```
% Examples:
% L=7;[t,x,A]=StructuralDynamicalSystem(L,200,-1,...
% 20,.1*[1 zeros(1,L-1) 1 zeros(1,L-1)],zeros(1,2*L));
% csvwrite ('TXdata.csv', [t.' x]);
% [t,x,xp,Ap,Aps]=BuildingDynamicsA('TXdata',3,...
% 20,0,0);
% L=7;[t,x,A]=StructuralDynamicalSystem(L,200,0,...
% 20,.1*[1 zeros(1,L-1) 1 zeros(1,L-1)],zeros(1,2*L));
% csvwrite ('TXdata.csv', [t.' x]);
% [t,x,xp,Ap,Aps]=BuildingDynamicsA('TXdata',3,...
% 20,0,1);
% L=7;[t,x,A]=StructuralDynamicalSystem(L,200,0,...
% 20,.1*[1 zeros(1,L-1) 1 zeros(1,L-1)],zeros(1,2*L));
% csvwrite ('TXdata.csv', [t.' x]);
% [t,x,xp,Ap,Aps]=BuildingDynamicsA('TXdata',3,...
% 20,1,1);
function [t,x,xp,Ap,Aps]=BuildingDynamicsA(fname,sm,ss,sp,smat)
   TXdata=csvread([fname,'.csv']);
   [N,n]=size(TXdata);
   t=TXdata(:,1);
   x=TXdata(:,2:n);
   L=(n-1)/4;
   if sp==0
    [Ap,Aps]=LSDITMatrixID(x,ss);
```

```
    else
     [Ap,Aps]=HLSDITMatrixID(x,ss);
    end
    xp = TXdata(1,2:n);
    if smat==1
     Tm=Aps.';
    else
Tm=Ap.';
    end
    for k=1:(N-1)
      xp=[xp;xp(k,:)*Tm];
      subplot(121);
      [dx,dy]=BuildingDynamics(1,L-1,x(k,:));
      title('Original system behavior')
      subplot(122);
      [dx,dy]=BuildingDynamics(1,L-1,xp(k,:));
      title('Identified system behavior')
      pause(.1);
    end
    figure;
    subplot(211);plot(t,x(:,sm),'k.-','markersize',15);
    hold on;
    plot(t,xp(:,sm),'b.-','markersize',15);
    hold off;
    grid on;
    legend('original signal','identified signal');
    subplot(212);plot(t,x(:,sm+L),'k.-','markersize',15);
    hold on;
    plot(t,xp(:,sm+L),'b.-','markersize',15);
    hold off;
    grid on;
    legend('original signal','identified signal');
end
```

Observarción. *El programa* `BuildingDynamicsA.m` *requiere los programas GNU Octave:* `BuildingDynamics.m`*,* `LSDITMatrixID.m` *y* `HLSDITMatrixID.m`*. Para información sobre el procedimiento necsario para obtener copias de estos programas, el lector es referido a la sección §5.*

Podemos aplicar el programa GNU Octave `BuildingDynamicsA.m` para identificar y aproximar la dinámica del sistema correspondiente a la muestra de señales mecánicas almacenada en el archivo `TXdata.csv`, la secuencia de comandos recomendada se muestra a continuación.

```
>> [t,x,xp,Ap,Aps]=BuildingDynamicsA('TXdata',3,20,0,0);
```

Las salidas gráficas generadas por el programa se muestran en la figura 4.2.

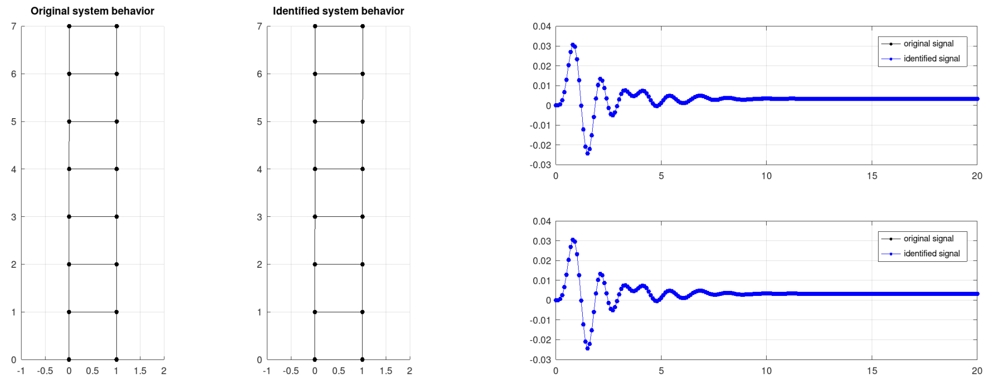


FIGURA 4.2. Salidas gráficas generadas por el algoritmo de identificación UAC: Identificación de comportamiento dinámico (izquierda). Identificación de señales mecánicas correspondientes (derecha).

Observación: *La dinámica identificada en la figura 4.2 del ejemplo anterior, corresponde a un sistema disipativo donde la matriz estructural* $\mathbf{D}$ *en simétrica negativa definida.*

4.1.2. Sistemas dinámicos estructurales conservativos. Podemos aplicar el programa GNU Octave `StructuralDynamicalSystem.m` para calcular una muestra de señales mecánicas correspondientes a deflexiones y velocidades de deflexión del sistema estructural $\mathcal{E}(t)$ bajo condiciones conservativas, para una secuencia $\{t_k\}_{k=1}^{N}$ de tiempos de muestreo. Para generar la muestra de señales mecánicas antes mecionada y almacenarla en memoria como un archivo en formato `*.csv`, podemos aplicar la siguiente secuencia de comandos en GNU Octave.

```
>> L=7;
>> [t,x,A]=StructuralDynamicalSystem(L,200,0,20,...
.1*[1 zeros(1,L-1) 1 zeros(1,L-1)],zeros(1,2*L));
>> csvwrite ('TXdata.csv', [t.',x]);
```

Podemos aplicar el programa GNU Octave `BuildingDynamicsA.m` para identificar y aproximar la dinámica del sistema correspondiente a la muestra de señales mecánicas almacenada en el archivo `TXdata.csv`, la secuencia de comandos recomendada se muestra a continuación.

```
>> [t,x,xp,Ap,Aps]=BuildingDynamicsA('TXdata',3,20,1,1);
```

Las salidas gráficas generadas por el programa se muestran en la figura 4.3.

Observación: *La dinámica identificada en la figura 4.2 del ejemplo anterior, corresponde a un sistema conservativo donde* $\mathbf{D} = 0$.

5. COMENTARIOS

Los algoritmos de identificación de sistemas presentados en este documento forman parte del sub-proyecto *Linear Dynamical System Identification for Mechanical Systems* del proyecto

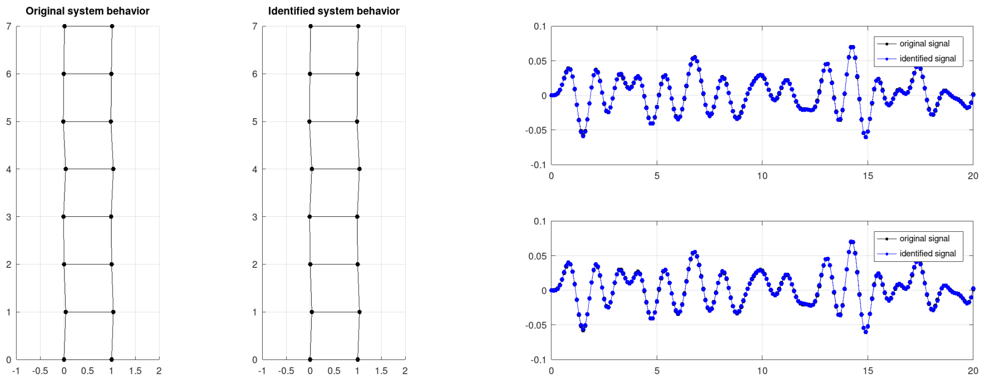


FIGURA 4.3. Salidas gráficas generadas por el algoritmo de identificación UAC: Identificación de comportamiento dinámico (izquierda). Identificación de señales mecánicas correspondientes (derecha).

Universal Algebraic Controllers and System Identification, cuyos repositorio está disponible como parte del directorio de proyectos del CADDS (Centre for Analysis of Data-Driven Systems), en la dirección:
https://github.com/cadds-lab/LDSI.
Para más información sobre los programas utilizados en este documento, o para obtener programas adicionales necesarios para realizar las prácticas de laboratorio de este manual, se refiere al lector al respositorio **LDSI** antes mencionado. En particular, el programa `BuildingDynamicsA.m` requiere los programas GNU Octave: `BuildingDynamics.m`, `LSDITMatrixID.m` y `HLSDITMatrixID.m`.

6. EJERCICIOS

1. Modificar los métodos implementados en los proyectos de laboratorio de las secciones §4.1.1 y §4.1.2 para sub-estructuras de 5 niveles, en lugar de las subestructuras de 7 niveles consideradas en §4.1.1 y §4.1.2. Manteniendo el resto de restricciones estructurales relativamente invariantes, es decir, las cargas estructurales iniciales se aplican en el mismo nivel y en las mismas condiciones, y las matrices estructurales se mantienen invariantes localmente.
2. Determinar las condiciones necesarias para que los sistemas descritos por las ecuaciones (2.3), (3.1) y (4.2) sean Hamiltonianos.
3. Probar que las soluciones de las ecuaciones (2.3) y (4.2) son continuas con respecto la topología producto (estudiada en [4, §1.9]) sobre $\mathbb{R}^2$ y $\mathbb{R}^{2n}$, respectivamente.
4. Considerando el caso particular $\beta = 0$ para la ecuación (2.3).
   (a) Calcular una función $H_\alpha \in C^2(\mathbb{R}^2, \mathbb{R})$ que verifica las restricciones:
   $$x' = y = \frac{\partial H_\alpha}{\partial y},\ \ y' = -\alpha x = -\frac{\partial H_\alpha}{\partial x},\ \ H_\alpha(0,0) = 0$$
   para cualquier (x, y) en el conjunto solución de (2.3) cuando $\beta = 0$.
   (b) Para la función $H_\alpha(x, y)$ calculada en (a). Considerar la siguiente relación en $\mathbb{R}^2$: $(x, y) \sim_\alpha (x', y') \iff H_\alpha(x, y) = H_\alpha(x, y)$. Probar que $\sim_\alpha$ es una relación de equivalencia.

(c) Considerando la topología cociente (estudiada en [4, §1.10]) en $\mathbb{R}^2$ correspondiente a la relación de equivalencia $\sim_\alpha$ definida en (b). Calcular $\pi_\alpha((1,0))$, donde π_α es el mapa de proyección canónica correspondiente a la topología cociente determinada por $\sim_\alpha$.

(d) Considerando el caso particular de la función H_1 calculada en (a) (cuando $\alpha = 1$) y la relación de equivalencia correspondiente $\sim_1$. Si $\mathbb{D}^2 = \{(x,y) \in \mathbb{R}^2 | x^2 + y^2 \leq 1\}$, probar o refutar que $\mathbb{D}^2/\sim_1 \;\simeq_H\; [0,1]$ (los conjuntos $\mathbb{D}^2/\sim_1$ y $[0,1]$ son homeomorfos) con respecto a la topología cociente correspondiente a $\sim_1$.

Referencias

[1] Edwards, C. H., Penney, D. E (2000). Ecuaciones Diferenciales. 4a Ed. Prentice Hall.

[2] Vides, F (2019). On uniform connectivity of algebraic matrix sets. Banach J. Math. Anal., 13(4):918-943, 2019.

[3] Vides, F (2020). Universal algebraic controllers and system identification. In press. 2020. Also available at https://arxiv.org/pdf/2001.11133.pdf

[4] Vides, F (2019). Introducción a la Topología y a la Teoría de Homotopía. Manual de Lecturas del Curso. Documento disponible en la dirección: https://cadds-lab.github.io/introductory_topology.pdf