Práctica 5
Modelos Matemáticos en Ecuaciones Diferenciales

Objetivo. El objetivo de esta práctica es dar una introducción al uso del comando `ode` y sus variantes para definir y resolver ecuaciones diferenciales mediante Matlab.

Introducción.

El modelo matemático más natural para representar el comportamiento de un sistema dinámico lo constituyen las ecuaciones diferenciales. Las ecuaciones diferenciales permiten representar sistemas dinámicos analógicos de muy diversos tipos: lineales y no lineales, variantes e invariantes en el tiempo, con parámetros concentrados y con parámetros distribuidos.

Debido a la dificultad de establecer una teoría general para tratar tanta variedad de sistemas, la teoría clásica de control se restringe solamente a los SLIT (sistemas lineales e invariantes en el tiempo) cuyo modelado se cubre perfectamente con la noción de función de transferencia que se trató en la práctica No. 4.

En general, un sistema SISO puede ser modelado mediante una ecuación diferencial, no necesariamente lineal.

Por ejemplo, consideremos el caso de una varilla rígida de masa m actuada en su extremo superior por un motor que puede imprimirle un par para hacerla girar respecto a ese mismo extremo (ver figura (5.1).

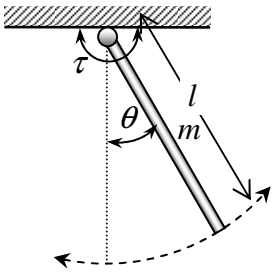

Figura 5.1.- Varilla rígida actuada por uno de sus extremos.

Aplicando la segunda ley de Newton para movimiento rotacional respecto al eje de giro:

$$\sum \tau_{ext} = J\alpha \tag{4.1}$$

Donde: $\sum \tau_{ext}$ es el par resultante de los pares externos aplicados al cuerpo en movimiento

J es el momento de inercia respecto al eje de giro del cuerpo en movimiento

$\alpha = \frac{d^2\theta}{dt^2}$ es la aceleración angular del cuerpo respecto al eje de giro

En el caso de la varilla, los pares externos aplicados son: el par del motor τ, el par debido a la

fricción τ_f que siempre se opone al movimiento de la varilla y el par debido a la fuerza de gravedad τ_g. Ver el diagrama del cuerpo libre en la figura 5.2. en el cual se supone que la masa de la varilla está concentrada en su centro.

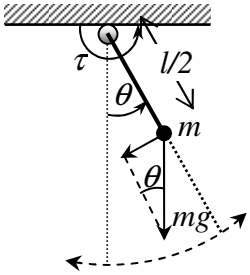

Figura 5.2.- Diagrama del cuerpo libre del centro de masa de la varilla.

El momento de inercia respecto al extremo de giro es $J = \frac{1}{3}ml^2$, por lo tanto, la ecuación (4.1) queda como sigue

$$\tau - \tau_f - \tau_g = \frac{1}{3}ml^2 \frac{d^2\theta}{dt^2} \tag{4.2}$$

Del diagrama del cuerpo libre (figura 5.2) se observa que el par debido a la fuerza de gravedad es producido por la componente del peso perpendicular a la varilla, por lo tanto, $\tau_g = mg\frac{l}{2}\sin\theta$. Además, si consideramos sólo el efecto de la fricción viscosa, el par debido a la fricción será proporcional a la velocidad tangencial, es decir, $\tau_f = \mu l \frac{d\theta}{dt}$, donde μ es el coeficiente de fricción viscosa. Sustituyendo en (4.2) se obtiene

$$\tau - \mu\frac{d\theta}{dt} - mg\frac{l}{2}\sin\theta = \frac{1}{3}ml^2\frac{d^2\theta}{dt^2} \tag{4.3}$$

o bien, en notación compacta

$$\frac{1}{3}ml^2\ddot{\theta} + \mu\dot{\theta} + mg\frac{l}{2}\sin\theta = \tau \tag{4.4}$$

Despejando la máxima derivada de θ se obtiene

$$\boxed{\ddot{\theta} = -a_2\dot{\theta} - a_1\sin\theta + b_0\tau} \tag{4.5}$$

donde: $a_2 = \frac{3\mu}{ml^2}$, $a_1 = \frac{3g}{2l}$ y $b_0 = \frac{3}{ml^2}$

La ecuación (4.5) es un modelo matemático para la varilla actuada por el motor, cuya entrada es el par del motor τ y cuya salida es el ángulo θ de la varilla respecto a la vertical, como se muestra en la figura 5.3.

Figura 5.3.- Modelo entrada-salida de la varilla actuada por el motor.

El modelo en Espacio de Estado.

Obsérvese que la ecuación diferencial obtenida (4.5) es una ecuación no lineal, por lo tanto no puede ser expresada mediante una función de transferencia. En este caso, una alternativa de modelo más natural es la propia ecuación diferencial (4.5), o alguna variante de ésta.

La ecuación (4.5) es una ecuación de diferencial segundo orden. Esta ecuación puede ser reescrita como dos ecuaciones diferenciales de primer orden, definiendo dos ***variables de estado*** x_1, x_2 como sigue:

$$x_1 = \theta, \;\; x_2 = \dot{\theta} \tag{4.6}$$

sustituyendo estas variables en la ecuación (4.5), obtenemos

$$\boxed{\begin{aligned} \dot{x}_1 &= x_2 \\ \dot{x}_2 &= -a_2 x_2 - a_1 \sin x_1 + b_0 \tau \end{aligned}} \tag{4.7}$$

al sistema de ecuaciones diferenciales de primer orden dado por (4.7) se le llama modelo en variables de estado o en espacio de estado o simplemente ***ecuaciones de estado***.

El espacio de estado es la representación n-dimensional (n=2 en este caso) cuyos ejes coordenados son las variables de estado. En el ejemplo de la varilla actuada, como solo hay dos variables de estado, el espacio de estado es un plano de una variable de estado respecto a la otra (x_2 respecto a x_1 o viceversa).

Los comandos `ode` de Matlab.

Matlab permite obtener la respuesta de modelos en variables de estado mediante diversas versiones del comando `ode` (Ordinary Differential Equation). Las diferentes versiones disponibles dependen de los métodos numéricos que utiliza Matlab para el cálculo de las soluciones del sistema de ecuaciones de estado. En la siguiente tabla se describen las versiones disponibles.

Comando (solver)	Método numérico	Tipo de problema	Exactitud	Cuando usar
ode45	Runge-Kutta 4,5	No rígido	media	Mayoría de los casos. Debe ser primer método a intentar
ode23	Runge-Kutta 2,3	No rígido	baja	Problemas moderadamente rígidos. Tolerancia al error holgada
ode113	Adams-Bashforth-Moulton	No rígido	Baja a alta	Problemas con tolerancia estricta al error. Problemas computacionalmente intensivos.
ode15s	Fórmulas de diferenciación numérica	rígido	Baja a media	Si **ode45** es lento debido a rigidez del problema
ode23s	Fórmula de Rosenbrock	rígido	baja	Tolerancia burda al error en un problema rígido y matriz de masa constante.
ode23t	Regla trapezoidal	Moderada-mente rígido	baja	Problema moderadamente rígido si se requiere una solución sin amortiguamiento numérico.
ode23tb	Regla trapezoidal+dif numérica	rígido	baja	Tolerancia al error holgada en un problema rígido

Rigidez. Se dice que el problema numérico de resolver un sistema de ecuaciones diferenciales es rígido si es numéricamente inestable (produce errores grandes ante pequeños cambios), excepto para pasos de integración extremadamente pequeños.

De acuerdo a la tabla anterior, en ausencia de información del tipo de ecuación que se quiere resolver, el primer método a utilizarse debe ser el `ode45`. A continuación se describe la manera en que se usa el comando `ode45`, sin embargo, la descripción es válida para cualquiera de los ode descritos en la tabla anterior: `ode23`, `ode45`, `ode113`, `ode15s`, `ode23s`, `ode23t`, `ode23tb`.

Sintaxis de `ode45`.- La forma más sencilla de ejecutar el comando `ode45` es la siguiente:

```
[T,X]=ode45(odefun,rango,x0,opciones)
```

donde:

`odefun`	Es un manejador de la función definida por el usuario, en la cual se definen las ecuaciones de estado.
`rango`	Es un vector especificando el intervalo de integración. Si se especifica solo el inicio y el final: `t0=[t0,tf]` Matlab calcula la solución desde `t0` hasta `tf` calculando la solución en cada paso de integración. Si se desea calcular la solución en puntos específicos, se deberán especificar dichos puntos: `rango = [t0,t1,...,tf]`
`x0`	Vector de valores iniciales de las variables de estado.
`opciones`	Estructura de parámetros opcionales que cambia las propiedades de integración por default del método. Se pueden establecer opciones usando la función `odeset`. Las propiedades comúnmente modificadas son la tolerancia de error `RelTol` (`1e-3` por default) y el vector de tolerancias absolutas de error `AbsTol` (todos los componentes son `1e-6` por default).
`T`	Vector de instantes de tiempo en los que se calculó la solución
`X`	`X` es una matriz cuyas columnas corresponden a cada variable de estado y cuyos renglones corresponden al instante de tiempo de cada renglón de `T`.

Descripción

`[T,X]=ode45(odefun,rango,x0)` integra el sistema de ecuaciones diferenciales $\dot{x} = f(t,x)$ definido dentro de la función cuyo manejador es `odefun` con las condiciones iniciales dadas `x0`, calculando la solución en el intervalo dado por `rango`.

La función `f=odefun(t,x)`, para un escalar `t` y un vector columna `x`, debe producir el vector columna `f` correspondiente a $f(t,x)$.

Manejadores de funciones.

Matlab permite el uso de manejadores o apuntadores a funciones para poder utilizar funciones como parámetros de otras funciones, o para poder acceder a ellas fuera de su alcance normal.

Para construir un manejador de función solamente se precede el nombre de la función por el símbolo @.

Ejemplo: Construimos una función que recibe como parámetro la función que se desea graficar:

```
function y=grafica(funcion)
% Grafica la funcion especificada en el intervalo de 0 10
x=0:0.01:10;
y=funcion(x);
plot(x,y);
grid on
```

Luego podemos usar la función anterior con diferentes funciones a graficar:

```
h1=@sin;  %Manejador de la función sin
h2=@cos; %Manejador de la función cos
y=grafica(h1); %Grafica la función sin(x) en el intervalo de 0 a 10
y=grafica(h2); %Grafica la función cos(x) en el intervalo de 0 a 10
```

Solución de un sistema de ecuaciones de estado sin entrada mediante `ode45`.

Para ilustrar como se utiliza ode45 para obtener la solución de un conjunto de ecuaciones de estado a continuación definimos las ecuaciones de estado (4.7) para el ejemplo del sistema de la varilla actuada considerando los siguientes valores para las constantes involucradas

$$m = 0.1Kg\ ,\ \ \mu = 0.01\ Nt\ \ seg\ ,\ l = 0.5mt\ ,\ \ g = 9.81\ mt/seg^2$$

y se considera el movimiento natural de la varilla (con el actuador desconectado), es decir, la entrada de par es cero, $\tau = 0\ Nt$. Para ello debemos escribir la función siguiente:

```
function dx=varilla(t,x)
%% Ecuaciones de estado de una varilla rígida actuada
%% por un motor en uno de sus extremos
%%Constantes del sistema:
m=0.1;
mu=0.01;
L=0.5;
g=9.81;
a2=3*mu/m/L^2;
a1=3*g/2/L;
b0=3/m/L^2;
%% Entrada al sistema (cero)
tao=0;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
dx = zeros(2,1);    %Inicializa vector columna para dos ecuaciones
%% Ecuaciones de Estado:
dx(1) = x(2);
dx(2) = -a2*x(2)-a1*sin(x(1))+ b0*tao;
```

Para obtener la solución de las ecuaciones de estado, usamos el comando `ode45` como sigue:

```
[t,x]=ode45(@varilla,[0 10],[pi/2 0]); %Calcula la solución
%%% en el intervalo de 0 a 10 con condiciones iniciales (pi/2,0)
plot(t,x); %Grafica los dos estados respecto al  tiempo
plot(x(:,1),x(:,2)) %grafica la solución en el plano de estado
```

En la figura 5.4 se observa el comportamiento de los dos estados respecto al tiempo. Como puede verse, la dinámica corresponde a lo esperado en un péndulo simple con fricción, el cual comienza oscilando un tiempo y al final se estabiliza en la posición vertical de reposo ($\theta = 0$). El mismo comportamiento se visualiza en la trayectoria de la solución en el plano de estado de la figura 5.5.

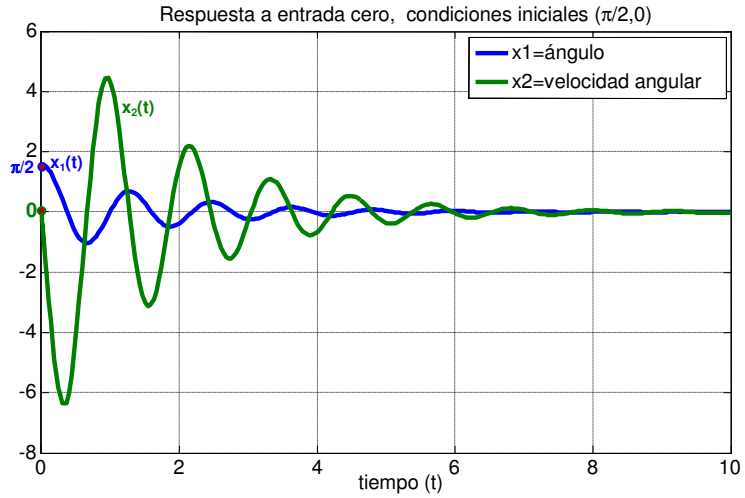

Figura 5.4.- Respuesta de la varilla con el actuador desconectado.

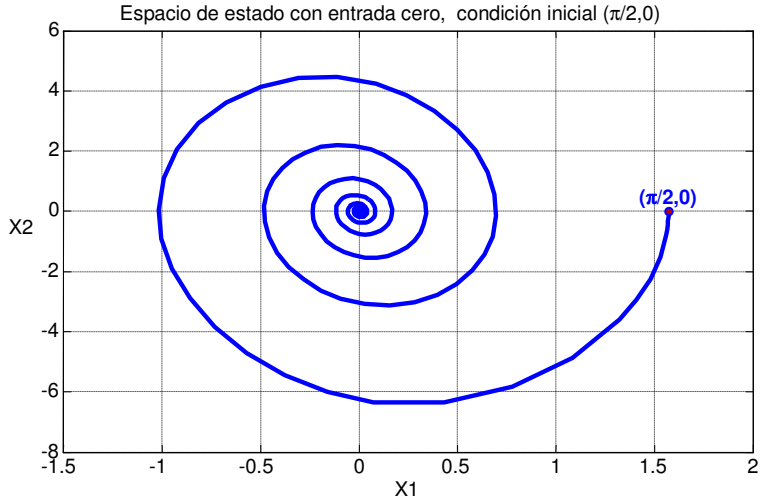

Figura 5.5.- Respuesta de la varilla sin actuador en el plano de estado.

Se puede mejorar la exactitud del método de integración, o disminuir la exactitud para reducir el tiempo de cálculo cambiando los parámetros mediante `odeset` como sigue

```
>> opciones=odeset('RelTol',1e-6,'AbsTol',[1e-6 1e-7]);
>> [t,x]=ode45(@varilla,[0 10],[pi/2 0],opciones);
```

Solución de un sistema de ecuaciones de estado con entrada mediante `ode45`.

Una manera sencilla de considerar funciones del tiempo como entradas a un sistema definido en una función `odefun` es calcularlas dentro del código de la misma función `odefun`, sin embargo, esto no es lo más adecuado, pues se supone que una entrada debe provenir de fuera del sistema.

Por ejemplo, para considerar una entrada sinusoidal de frecuencia 1 hertz, en el ejemplo de la varilla, simplemente reemplazamos la línea donde se define la entrada `tao=0` por:

```
tao=sin(2*pi*t); % Entrada sinusoidal de 1 hertz
```

La única manera de que la función `odefun` en la cual se define el sistema de ecuaciones de estado acepte una entrada variable desde afuera de la función es enviársela como un parámetro vectorial adicional.

El nuevo parámetro deberá contener los valores de la entrada evaluada en un rango de instantes de tiempo que difícilmente coincidirá con los instantes de tiempo que calcule el método de integración, ya que estos últimos se calculan con un paso de integración variable que el método va adaptando para disminuir el error de integración, por esta razón será necesario enviar también como parámetro el vector de instantes de tiempo en que la entrada se está proporcionando.

Estos dos parámetros serán utilizados por la función `odefun` para calcular los valores de la entrada en los instantes de tiempo de integración mediante ***interpolación***, usando el comando `interp1`.

Ejemplo: La función varilla se reescribirá para aceptar como parámetros los vectores de tiempo y de valores de la entrada como sigue

```
function dx=varilla1(t,x,t1,entrada)
%% entrada es el vector de valores de entrada en los instantes
%% de tiempo dados en t1
%% Constantes del sistema:
m=0.1;
mu=0.01;
L=0.5;
g=9.81;
a2=3*mu/m/L^2;
a1=3*g/2/L;
b0=3/m/L^2;
%% Entrada al sistema (vector de valores de entrada en cada instante t1)
tao= interp1(t1,entrada,t); % Interpola los puntos (t1,entrada) en cada
%instante de tiempo de integración t
%%%%%%%%%%%%%%
dx = zeros(2,1);    %Vector columna para dos ecuaciones
%% Ecuaciones de Estado:
dx(1) = x(2);
dx(2) = -a2*x(2)-a1*sin(x(1))+ b0*tao;
```

Para obtener la respuesta a una función de entrada primero se generará la entrada deseada, por ejemplo a continuación se considera una entrada pulso de amplitud 1 Nt y de duración 0.5 seg que aparece en el instante t=0.5 seg

```
>> tu=0:0.01:10;  % Vector de instantes de tiempo de la entrada
>> u=(tu>0.5 & tu<1); %Entrada pulso de amplitud 1 y duración 0.5
```

Usando estos vectores de tiempo y valores de entrada como parámetros, invocamos la función `varilla1` como sigue, con intervalo de integración de 0 a 10 seg y con condiciones iniciales cero.

```
>> opciones=odeset('RelTol',1e-3,'AbsTol',[1e-6 1e-6]);
>> [t,x]=ode45(@(t,x)varilla1(t,x,tu,u),[0 10],[0 0],opciones);
```

En la figura 5.6 se muestran las gráficas de la respuesta de los dos estados del sistema así como la entrada de par aplicada.

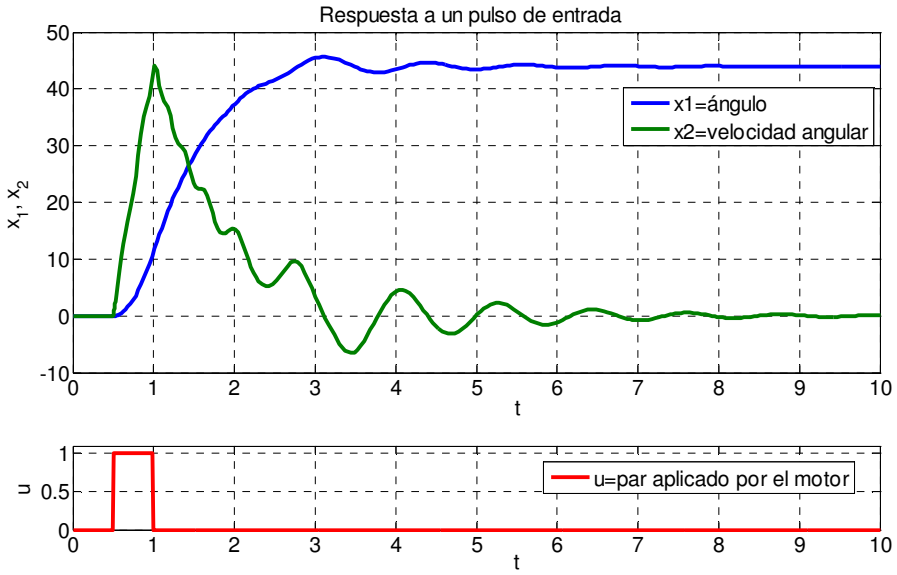

Figura 5.6.- Respuesta de la varilla a un pulso de par aplicado por el motor.

Caso lineal.

Cuando un sistema SISO de entrada $u(t)$ y salida $y(t)$ es un SLIT, se puede representar por un modelo lineal como la ecuación diferencial lineal de orden n siguiente,

$$a_n \overset{(n)}{y} + a_{n-1} \overset{(n-1)}{y} + \ldots + a_1 \dot{y} + a_0 y = b_m \overset{(m)}{u} + b_{m-1} \overset{(m-1)}{u} + \ldots + b_1 \dot{u} + b_0 u \tag{4.8}$$

Aunque no siempre es sencillo, es posible elegir un conjunto de variables de estado para transformar (4.8) en n ecuaciones diferenciales lineales de primer orden de la forma

$$\begin{aligned} \dot{x}_1 &= a_{11}x_1 + a_{12}x_2 + \ldots + a_{1n}x_n + b_1 u \\ \dot{x}_2 &= a_{21}x_1 + a_{22}x_2 + \ldots + a_{2n}x_n + b_2 u \\ &\ldots \\ \dot{x}_n &= a_{n1}x_1 + a_{n2}x_2 + \ldots + a_{nn}x_n + b_n u \end{aligned} \tag{4.9}$$

Además, la salida del sistema normalmente es una combinación de estados y entrada

$$y = c_1 x_1 + c_2 x_2 + \ldots . c_n x_n + du \tag{4.10}$$

Es decir, en forma matricial

$$\begin{aligned} \dot{x} &= Ax + bu \\ y &= cx + du \end{aligned} \tag{4.11}$$

Donde x es el vector de estados, A es una matriz cuadrada $n \times n$ de coeficientes constantes, b es

un vector columna de coeficientes constantes, c es un vector renglón de coeficientes constantes y d es un escalar constante (en la mayoría de los casos $d = 0$), es decir,

$$x = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix},\ \dot{x} = \begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \vdots \\ \dot{x}_n \end{bmatrix},\ A = \begin{bmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \dots & a_{nn} \end{bmatrix},\ b = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix},\ c = \begin{bmatrix} c_1 & c_2 & \dots & c_n \end{bmatrix}$$

En este caso las ecuaciones de estado y la ecuación de salida (4.11) se pueden transformar de manera directa a su Función de Transferencia equivalente aplicando Transformada de Laplace suponiendo condiciones iniciales cero, obtenemos

$$\begin{aligned} sX(s) &= AX(s) + bU(s) \\ Y(s) &= cX(s) + dU(s) \end{aligned} \tag{4.12}$$

despejando el vector de estado

$$X(s) = (sI - A)^{-1} bU(s) \tag{4.13}$$

sustituyendo en la ecuación de salida

$$Y(s) = \left[c(sI - A)^{-1} b + d \right] U(s) \tag{4.14}$$

Es decir, la función de transferencia correspondiente a (4.11) es

$$G(s) = \frac{Y(s)}{U(s)} = c(sI - A)^{-1} b + d \tag{4.15}$$

Comandos de Matlab para manejar modelos lineales en variables de estado.

Matlab provee los comandos explicados en la siguiente tabla para manejar modelos en el espacio de estado en el ***caso lineal***:

Comando	**explicación**
`sist=SS(A,B,C,D)`	Crea un sistema tipo `SS` (Espacio de Estado), es decir, `sist` contiene el modelo de espacio de estado: dx/dt = Ax(t) + Bu(t) y(t) = Cx(t) + Du(t)
`[num,den]=ss2tf(A,B,C,D,iu)`	Convierte el modelo de espacio de estado dado por las matrices A,B,C,D a la matriz de transferencia respecto a la entrada `iu`. El vector `den` contiene los coeficientes del denominador en potencias descendentes de 's'. Los coeficientes de los numeradores son regresados en la matriz `num`, la cual contiene un renglón por cada salida del sistema. Es decir, evalúa la ecuación (4.15) para el caso MIMO.
`[Z,P,K]=ss2zp(A,B,C,D,iu)`	Convierte el modelo de espacio de estado dado por las matrices A,B,C,D a la matriz de transferencia respecto a la entrada `iu` en su forma factorizada. El vector P contiene los polos del denominador. Los ceros de los numeradores son regresados en la matriz Z, la cual contiene una columna por cada salida del sistema. Las ganancias de cada numerador son regresadas en K.
`[A,B,C,D]=tf2ss(num,den)`	Convierte una función de transferencia en su forma de división de polinomios en potencias descendentes de 's' a espacio de estado.
`[A,B,C,D]= zp2ss(Z,P,K)`	Convierte una función de transferencia en su forma factorizada a espacio de estado.
`minreal(G)`	Obtiene el sistema de orden mínimo equivalente al sistema `G` dado.

Ejemplo.

```
>> num=[1 1];
>> den=[1 5 6];
>> G=tf(num,den) % Define la F.T. G=(s+1)/(s^2+5*s+6)
Transfer function:
    s + 1
-------------
s^2 + 5 s + 6
>> [A,B,C,D]=tf2ss(num,den)% Obtiene las matrices del modelo
% equivalente en espacio de estado
A =
    -5    -6
     1     0
B =
     1
     0
C =
     1     1
D =
     0
>>s=tf('s');
>> G1=C*inv(s*eye(2)-A)*B+D % Evalúa la fórmula (4.15)
Transfer function:
     s^3 + 6 s^2 + 11 s + 6
--------------------------------
s^4 + 10 s^3 + 37 s^2 + 60 s + 36
```

Obsérvese que la F.T. obtenida es de cuarto orden, significa que no es la mínima posible, ya que se esperaba segundo orden pues el sistema era de dos estados (n=2), por esta razón a continuación se usa el comando `minreal`:

```
>> minreal(G1)
Transfer function:
    s + 1
-------------
s^2 + 5 s + 6
```

Ejemplo: Modelo Lineal del sistema de la varilla actuada.

Como se mencionó arriba, el ejemplo de la varilla actuada por un extremo nos condujo al modelo en ecuación diferencial no lineal (4.5), el cual se pudo escribir como un modelo en ecuaciones de estado no lineales (4.7). Este modelo se puede *aproximar* por un modelo lineal si observamos que el único término no lineal en las ecuaciones de estado (4.7) es la función $\sin(x_1)$.

Si expresamos la función $\sin(x_1)$ por su fórmula de Taylor en las cercanías de $x_1 = 0$, es decir, para movimientos de la varilla cercanos a su punto de reposo, obtenemos

$$\sin(x_1) = x_1 - \frac{1}{3!}x_1^3 + \frac{1}{5!}x_1^5 - \frac{1}{7!}x_1^7 + \ldots \qquad (4.16)$$

Si truncamos los términos no lineales, dejando solo la parte lineal

$$\sin(x_1) \approx x_1 \qquad (4.17)$$

Sustituyendo esta aproximación en el modelo de estado (4.7), obtenemos el modelo aproximado en ecuaciones de estado

$$\begin{aligned}\dot{x}_1 &= x_2 \\ \dot{x}_2 &= -a_2 x_2 - a_1 x_1 + b_0 \tau\end{aligned} \tag{4.18}$$

y con la ecuación de salida

$$y = \theta = x_1 \tag{4.19}$$

las ecuaciones (4.18) y (4.19) se pueden escribir en la forma matricial (4.11), donde

$$A = \begin{bmatrix} 0 & 1 \\ -a_1 & -a_2 \end{bmatrix},\ b = \begin{bmatrix} 0 \\ b_0 \end{bmatrix},\ c = \begin{bmatrix} 1 & 0 \end{bmatrix},\ d = 0$$

O bien, podemos obtener la función de transferencia correspondiente, mediante la expresión (4.15)

$$G(s) = \frac{Y(s)}{U(s)} = c(sI - A)^{-1} b = \begin{bmatrix} 1 & 0 \end{bmatrix} \begin{bmatrix} s & -1 \\ a_1 & s + a_2 \end{bmatrix}^{-1} \begin{bmatrix} 0 \\ b_0 \end{bmatrix}$$

calculando la matriz inversa obtenemos

$$G(s) = \frac{\begin{bmatrix} 1 & 0 \end{bmatrix}}{s^2 + a_2 s + a_1} \begin{bmatrix} s + a_2 & 1 \\ -a_1 & s \end{bmatrix} \begin{bmatrix} 0 \\ b_0 \end{bmatrix}$$

Haciendo las operaciones

$$G(s) = \frac{\begin{bmatrix} 1 & 0 \end{bmatrix}}{s^2 + a_2 s + a_1} \begin{bmatrix} b_0 \\ s b_0 \end{bmatrix} = \frac{b_0}{s^2 + a_2 s + a_1} \tag{4.20}$$

Para validar el modelo lineal aproximado (4.20) a continuación escribimos el siguiente código para comparar los resultados del modelo lineal contra el modelo exacto no lineal (4.7).

```
>> m=0.1; mu=0.01; L=0.5; g=9.81; %%% Constantes del sistema
>> a2=3*mu/m/L^2;a1=3*g/2/L;
>> b0=3/m/L^2;
>> G=tf(b0,[1 a2 a1]) %%% F.T. del sistema, de acuerdo a (4.20)
Transfer function:
       120
-------------------
s^2 + 1.2 s + 29.43

>> tu=0:0.01:10;
>> u=(tu>0.5 & tu<0.6); %%% Define entrada pulso, amplitud 1, ancho 0.1
>> opciones=odeset('RelTol',1e-6,'AbsTol',1e-7); %% Define tolerancias
%% Obtiene la respuesta del modelo no lineal a la entrada pulso:
>> [t,x]=ode45(@(t,x)varilla1(t,x,tu,u),tu,[0 0],opciones);
%% Obtiene la respuesta del modelo lineal a la entrada pulso:
>> y=lsim(G,u,tu);
>> plot(t,x); %% Grafica los dos estados del modelo no lineal
>> grid on
>> hold on
>> plot(tu,y,'r--'); %% Grafica la salida de la F.T.
```

En la figura 5.7 se muestra la respuesta del código anterior, ligeramente arreglada para una mejor presentación.

Figura 5.7.- Respuesta de la varilla a un pulso de ancho 0.1 y amplitud 1, calculada mediante el modelo no lineal exacto (4.7) y el modelo lineal aproximado (4.20).

Ejercicios:

1) Generar las figuras 5.6 y 5.7.
2) Modificar la figura 5.7 para que en la gráfica superior solo muestre la salida exacta y la salida aproximada.
3) Después del punto anterior, modifica la entrada para que sea un pulso de amplitud 1 y duración 0.4 seg. que empiece desde el instante inicial y vuelve a generar la figura correspondiente.

Desarrollo de la Práctica.

1. Probar todos los ejemplos propuestos por el profesor conforme los va explicando.
2. Realizar todos los ejercicios propuestos.
3. Contestar el cuestionario de evaluación de la práctica.

Reportar:

1) El código utilizado para resolver el ejercicio (3) con comentarios adecuados.
2) Describe con tus propias palabras el movimiento que se observaría en la varilla si el experimento del ejercicio (3) se realizara en forma real. ¿Cuántas vueltas completas daría la varilla?

3) Para el circuito de la figura 5.8 cuya F.T. $\frac{V_C(s)}{V_i(s)}$ se obtuvo en el reporte de la práctica No. 4. Obtén las ecuaciones de estado en términos de los valores de R,L y C, eligiendo los estados siguientes: $x_1(t) = V_c(t)$, $x_2(t) = \dot{V}_c(t)$.

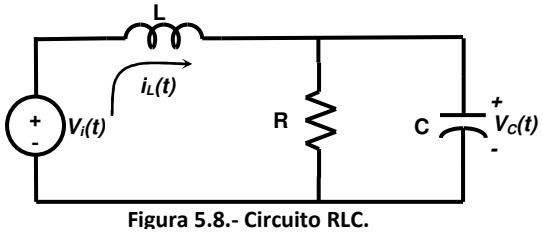

Figura 5.8.- Circuito RLC.

4) Escribe el código necesario para simular la respuesta del circuito de la figura 5.8 mediante el comando **ode45**, con entrada cero ($V_i(t) = 0$), bajo condiciones iniciales $V_c(0) = 10\ volts$ y grafica dicha respuesta en el plano $V_c(t)$ contra el tiempo y en el plano de fase $V_c(t)$ contra $\dot{V}_c(t)$.