Matlab para Análisis Dinámico de Sistemas

Análisis Dinámico de Sistemas, curso 2006-07

7 de noviembre de 2006

1. Introducción

Para usar las funciones aquí mencionadas se necesita Matlab con el paquete de Control de Sistemas (*Control System Toolbox*). Para obtener un listado de todas las funciones disponibles en esta *toolbox*, basta ejecutar en línea de comandos de Matlab:

```
help control
```

Asimismo, para obtener una descripción breve del modo de uso de una función cualquiera, basta ejecutar en línea de comandos:

```
help nombre_de_la_funcion
```

2. Definición de un sistema por su función de transferencia

Para definir un sistema por su función de transferencia, se puede usar la función `tf` de la siguiente manera:

```
G=tf(num,den)
```

donde `G` será la variable que contendrá el objeto "sistema" (que además se mostrará por pantalla al realizar la asignación), y `num` y `den` son respectivamente los polinomios del numerador y del denominador de la función de transferencia en el formato de representación de polinomios de Matlab. Este formato consiste en un vector que contiene los coeficientes del polinomio en orden de grado decreciente, siendo el de más a la derecha siempre el término independiente. Por ejemplo, para definir el polinomio del denominador $s^3 + 5s + 10$, se escribiría:

```
den=[ 1 0 5 10]
```

Obsérvese que el segundo elemento del vector es un cero que corresponde al término de grado 2.

Otra función útil para definir los polinomios de numerador y denominador a partir de los ceros y los polos del sistema es `poly`, que crea un polinomio (con representación Matlab) a partir de un vector conteniendo sus raíces, por ejemplo:

```
den=poly( [ polo1 polo2 ] )
```

Si en ese caso fuéramos a definir un sistema con dos polos complejos conjugados, habríamos escrito antes de la línea anterior:

```
polo1=-5+6*j
polo2=conj(polo1)
```

También existe una función `roots` para obtener las raíces a partir del polinomio. Por ejemplo, para obtener los polos a partir del polinomio del denominador:

```
polos=roots(den)
```

3. Respuesta a impulso y escalón unitarios

Para obtener la respuesta a impulso unitario se dispone de la función `impulse`, y para el escalón unitario `step`. Ambas tienen como único parámetro el objeto sistema. Por ejemplo, para el escalón:

```
sis=tf([1 2],[1 2 3])
step(sis)
```

generaría la gráfica de la evolución en el tiempo de la salida del sistema ante una entrada escalón unitario mostrada en la figura 1:

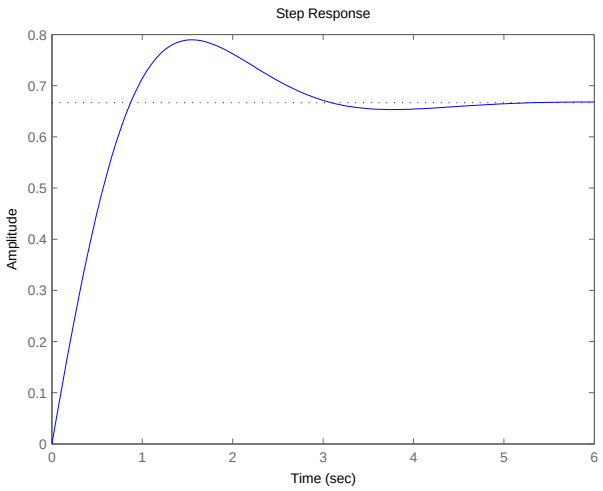

Figura 1: Respuesta a escalón unitario

4. Mapa de polos y ceros

La función `pzmap` realiza la representación gráfica de la situación en el plano complejo de los polos (cruces) y los ceros (círculos) del sistema. Por ejemplo, el siguiente código Matlab:

```
sis=tf([1 8.5],[1 10 61])
pzmap(sis)
```

sirve para representar el mapa de polos y ceros mostrado en la figura 2, correspondiente al sistema con función de transferencia:

$$G(s) = \frac{s + 8{,}5}{s^2 + 10s + 61}$$

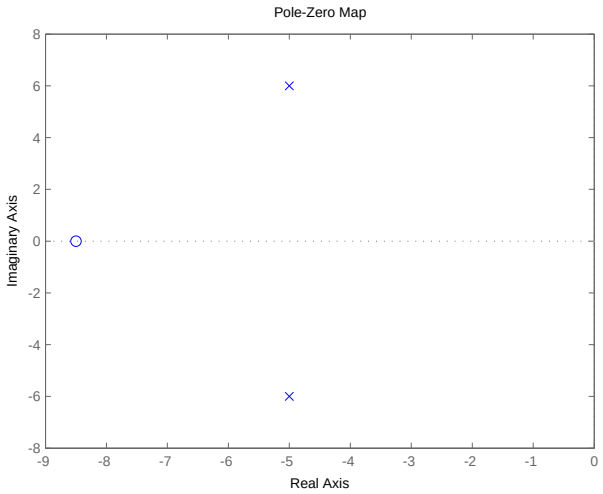

Figura 2: Mapa de polos y ceros

5. Representación gráfica de la respuesta temporal a partir de su expresión funcional

En este apartado se pretenden dar algunas ideas básicas para representar la respuesta temporal de un sistema lineal a partir de la expresión funcional obtenida como resultado de aplicar el método de Heaviside[1].

Para representar, por ejemplo, la siguiente respuesta impulsional:

$$g(t) = 2\mathrm{e}^{-t} - 2\mathrm{e}^{-2t} - t\mathrm{e}^{-2t}$$

correspondiente al sistema con función de transferencia:

$$G(s) = \frac{s + 3}{(s + 1)(s + 2)^2}$$

primero debe generarse la secuencia de instantes de tiempo en los que se va a evaluar la función $g(t)$ para su representación:

```
t=linspace(0,3,200);
```

expresión que genera un vector fila `t`, que contiene 200 valores equiespaciados de tiempo desde $t = 0$ hasta $t = 3$. El valor final se obtiene por prueba y error,

[1]Para entender mejor las expresiones aquí mostradas, deberá consultarse el documento "Introducción a Matlab"

pero se puede comenzar con tres veces el inverso del menor valor[2] de entre los a de todos los términos e^{-at}. En este caso es $a = 1$, y por lo tanto $t_{max} = 3/a = 3$.

Una vez obtenido el vector de tiempos,la evaluación de la función $g(t)$ para todos los instantes de tiempo contenidos en `t` es inmediato:

```
g=2*exp(-t)-2*exp(-2*t)-t.*exp(-2*t)
```

Obsérvese la diferencia de uso entre el operador `*`, que en este caso corresponde siempre a producto de escalar por vector, y el operador `.*`, que corresponde al producto elemento a elemento de dos vectores. La expresión `t*exp(-2*t)`, además de no ser lo que se pretende, generaría un error porque es el producto de dos matrices que no cumplen los requisitos dimensionales para poder llevar a cabo dicho producto: número de columnas de la primera igual al número de filas de la segunda.

Por último, con el comando `plot` se representa gráficamente el resultado de evaluar la función:

```
plot(t,g);
```

representado en la figura 3, donde las etiquetas de los ejes se han colocado con las funciones `xlabel` e `ylabel`.

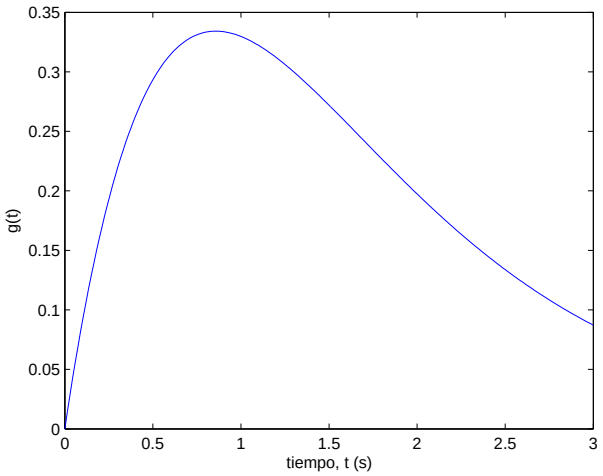

Figura 3: Respuesta a impulso representada a partir de su expresión funcional

En caso de tener polos complejos conjugados (sea uno de ellos `p`, con coeficiente de Heaviside `C`), podemos tener términos del tipo:

```
sig=real(p);
w=imag(p);
yi=2*abs(C)*exp(sig*t).*cos(w*t+angle(C))
```

donde `abs` es la función para obtener el módulo de un complejo, `angle` la función para obtener su argumento o ángulo, `real` su parte real e `imag` su parte imaginaria.

[2]Siempre positivo: no consideramos aquí sistemas inestables

6. Problema propuesto

Dado el sistema lineal definido por la siguiente ecuación diferencial:

$$\frac{d^3y(t)}{dt^3} + 5\frac{d^2y(t)}{dt^2} + 8{,}25\frac{dy(t)}{dt} + 17y(t) = 3u(t) + \frac{du(t)}{dt}$$

donde $y(t)$ es la salida y $u(t)$ es la entrada, se pide:

1. Deducir la forma aproximada de la respuesta a impulso unitario del sistema dado, a partir de la distribución de sus polos en el plano complejo.
2. Obtener la expresión matemática exacta en función del tiempo de dicha respuesta.
3. Dibujar con Matlab la respuesta impulsional obtenida en el apartado anterior y compararla con la que se obtiene por medio de la función `impulse`.