MATLAB

Expresiones aritméticas

- ^ Potencia
- * Multiplicación
- / división
- + suma
- - resta

Variables

Se pueden definir variables para utilizarlas en otras expresiones

```
>> a = 3^2
a= 9
>> b = 3
b = 3
>> a/b
ans = 3
```

Matrices

Una forma de crear una matriz es introduciendo sus elementos entre corchetes, poniendo cada fila en una línea distinta y separando los elementos de una misma fila por espacios en blanco. También se puede utilizar el punto y coma para separar filas y la coma para separar elementos de una fila.

```
>>A=[a b
  c 1]
A =
        9.0000                  6.0000
        0 + 2.0000i     1.0000
>> B=[1 2;3 4;5 6]
B =
1 2
3 4
5 6
```

A una matriz de una sola fila o de una sola columna también se le llama vector. Una forma rápida de crear vectores fila de elementos equiespaciados es utilizando el operador ‘dos puntos’.

```
>> v1=1:3
v1 =
        1 2 3
>> v2=1:0.5:3
v2 =
        1.0         1.5000 2.0000 2.5000 3.0000
```

Graficos

La función plot crea gráficos en el plano XY; si x e y son vectores de la misma longitud, la orden plot(x,y) accede a la pantalla gráfica y realiza un gráfico plano de los elementos de x contra los elementos de y. Por ejemplo, podemos dibujar la gráfica de la función seno sobre el intervalo [-4,4] con las instrucciones siguientes:

```
>> x = -4:.01:4;
>> y = sin(x);
>>plot(x,y)
```

Sistemas

Dada una función de transferencia H(s) el sistema queda definido, para MATLAB, con la introducción del numerador y el denominador de la función. Dado que estos son polinomios en potencias de s será suficiente con introducir los coeficientes de estos polinomios, lo cual haremos en forma de vectores fila, ordenando los coeficientes por orden de potencias descendentes.
Ejemplo: Sea el sistema

$$H(s) = \frac{3s^2 + 2}{3s^3 + 5s^2 + 2s + 1}$$

Entraremos este sistema a MATLAB con las instrucciones

```
>> num=[3 0 2];
>> den=[3 5 2 1];
>> H = tf(num,den)
```

Desarrollo en fracciones simples

Para realizar el desarrollo en fracciones simples utilizaremos la expresión:

```
>> [r, p, k]  = residue (num, den)
```

Donde r son los residuos, p los polos y k los términos directos.
La instrucción inversa es

```
>> [num, den] = residue (r,p,k)
```

Ceros y polos
Para obtener los ceros y los polos de una función de transferencia utilizaremos la expresión:

```
>>  [z,p,K] = tf2zp(num,den)
```

Donde z son los zeros, p los polos y K la ganancia

En caso de tener los ceros, polos y la ganancia del sistema obtendremos la función de transferencia de la siguiente manera:

```
>> [num, den] = zp2tf(z,p,K)
>> H= tf(num,den)
```

Funciones de transferencia serie, paralelo y realimentadas

Para calcular la función de transferencia de un sistema en serie, paralelo o realimentado utilizaremos las siguientes expresiones.
Supongamos a función G1(s) = num1/den1 y G2(s) = num2/den2.

Para el sistema serie tendremos:

```
>> [num,den]  = series(num1, den1, num2, den2)
```

Para el sistema paralelo tendremos:

```
>> [num,den]  = parallel (num1, den1, num2, den2)
```

Para el sistema realimentado tendremos:

```
>> [num,den]  = feedback (num1, den1, num2, den2)
```

Transformación de función de transferencia al espacio de estados

Para transformar el modelo del sistema de función de transferencia al espacio de estados en sistemas de una entrada y una salida utilizaremos la siguiente expresión:

```
>> [A, B, C, D] = tf2ss(num,den)
```

Y viceversa

```
>> [num,den] = ss2tf(A, B, C, D)
```

En el caso de varias salidas, la instrucción a utilizar es

```
>> [num,den] = ss2tf(A, B, C, D, iu),
```

Donde iu es un entero en el intervalo [1, n] con n el número de salidas.

Respuesta al impulso

Para obtener la respuesta temporal de una función de transferencia ante una entrada impulso se emplea el comando *impulse*, definiendo los vectores de los polinomios como en el apartado anterior.

```
>> num = [3]
>> den=[1 5]
>> G=tf(num,den)
>>impulse(G)
```

O también

```
>> num = [3]
>> den=[1 5]
>>impulse(num,den)
```

Respuesta al escalón

Para obtener la respuesta ante una entrada escalón unitario se emplea el comando step.

```
>> step(G)
```

Si quisiéramos obtener la respuesta ante un escalón de valor 5 y queremos ver la salida hasta un tiempo de 5 segundos, en pasos de 0.1, la instrucción es la siguiente.

```
>>step(5*num,den,0:0.1:5)
```

Respuesta a la rampa y otras señales
Es necesario hacer uso de la instrucción
```
>> [y, x] = lsim(num, den, u, t)
```

Donde u es la señal de entrada y t el tiempo

Suponiendo la función de transferencia del ejemplo anterior, para el intervalo de 0 a 10 segundos, podemos calcular la respuesta a la rampa de la siguiente manera.

```
>> t=0:0.1:10
>> u=t;
>> num = [3]
>> den=[1 5]
>>[y,x]=lsim(num,den,u,t);
>> plot(t,y,t,u)
```

Sistemas de segundo orden
Si tenemos el coeficiente de amortiguamiento y la frecuencia natural no amortiguada, podemos definir el sistema de la siguiente manera:

```
>> wn = 5
>> xi = 0.4
>>[num0,den] = ord2(wn,xi)
>>num = wn^2*num0
>>printsys(num,den)
```

Diagrama de Bode

El comando *bode* dibuja el diagrama de Bode de una función de transferencia.

```
>>bode(G) o  bode(num,den)
```

Lugar de raíces
El comando *rlocus* dibuja el lugar de raíces de un sistema.

```
>>rlocus(num,den) o rlocus(G)
```

Diagrama de nyquist
El comando *nyquist* calcula el diagrama de Nyquist de un sistema.

```
>> nyquist(num, den) o nyquist(G)
```