Ceduvirt

MATLAB APLICADO A INGENIERÍA

CONTENIDO

INTRODUCCIÓN

La integración de las Tecnologías de Información y comunicación (TIC) en las asignaturas de un currículo puede realizarse de varias formas. Una de ellas es el uso de las simulaciones. Estas se han convertido en una excelente herramienta para mejorar la compresión y el aprendizaje en áreas como las matemáticas, física, estadística, finanzas, etc. La simulación permite probar, analizar y descubrir cómo funciona o cómo se comporta un fenómeno.

Matlab es un programa interactivo de cálculo numérico y de visualización de datos basado en software de matrices, en un entorno de desarrollo totalmente integrado y orientado a proyectos que requieren un elevado cálculo numérico y visualización gráfica.

En las universidades Matlab se ha convertido en una herramienta básica tanto para estudiantes, como para docentes e investigadores por su amplio abanico de programas especializados llamados Toolboxes que cubren casi todas las áreas del conocimiento. Dispone de un programa SIMULINK que es un entorno gráfico interactivo con el que se puede analizar, modelar y simular sistemas.

1. VARIABLES Y FUNCIONES

1.1 OPERADORES

Una variable se crea por asignación. Los operadores básicos son:

x +y	Suma
x – y	Diferencia
x * y	Producto
x / y	División
x ^y	Potencia

Ejemplos:

En la ventana de comandos de Matlab, ejecutar:

```
>> v = 3
>> x = v + 6
>> y = v ^5 / 4
>> x = 2*3^5 + (5-3)* 8
```

1.2 VECTORES

Un *vector* *fila* de n elementos se puede representar de dos formas:

V = [v1,v2,v3,…..vn]   % con coma entre ellos, o

V = [v1 v2 v3 …..vn]   % con espacios entre ellos

Ejemplo:

Vector = [1 1.2 3.4 4/5 2.25]

Un vector se puede representar sin necesidad de explicitar todos los elementos, así:

EXPRESIÓN MATLAB	SIGNIFICADO
Vector = [a : b]	a y b son el primero y último elemento. Los elementos intermedios se diferencian en una unidad
Vector = [a : s : b]	a y b son el primero y último elemento. Los elementos intermedios se diferencian en la cantidad s
Vector = linespace[a,b,n]	a y b son el primero y último elemento. Hay n elementos uniformemente espaciados entre sí
Vector = logspace[a,b,n]	a y b son el primero y último elemento. Hay n elementos logarítmicamente espaciados entre sí

Ejemplos:

>>Vector1 = [5:5:30]   % elementos de 5 a 30 en pasos de 5

Vector1 = 5 10 15 20 25 30

```
>>Vector2 = [5:10]
Vector2 = 5  6  7  8  9  10  % elementos de 5 a 10 en pasos de 1 (por defecto)
```

Un *vector columna* se representa con sus elementos separados por punto y coma.

Ejemplo:

```
>>Vector = [2; 3; 2.5; 4.5; 8]

Vector =
        2
        3
        2.5
        4.5
        8
```

1.3 MATRICES

Las matrices se representan en Matlab introduciendo entre corchetes los vectores fila separados por punto y coma.

Ejemplo:

```
>>A = [1 3 5; 4 7 9; 4 2 10]
A =

        1   3   5
        4   7   9
        4   2   10
```

Algunas definiciones de variables matriciales:

A(m,n)	Define el elemento (m,n) de la matriz A
B = A’	Define la transpuesta de A
A(a:b,c:d)	Define una submatriz formada por las filas que hay entre la a-ésima y la b-ésima y por las columnas que hay entre la c-ésima y la d-ésima
A(:,c:d)	Submatriz formada por las filas de A y las columnas que hay entre la c-ésima y d-ésima
A(a:b,:)	Submatriz formada por las columnas de A y las filas que hay entre la a-ésima y b-ésima
size(A)	Devuelve el tamño u orden de la matriz A

Ejemplos:

```
>> A(2,3)
        ans =
                9
>> B = A'

B =

    1    4    4
    3    7    2
    5    9   10
```

Ceduvirt

```
>> eye(3)

ans =

     1     0     0
     0     1     0
     0     0     1

>> C=B(:,2:3)

C =

     4     4
     7     2
     9    10

>> D = B(1:2,:)

D =

     1     4     4
     3     7     2

>> size(D)



ans =

     2     3
```

1.4 FUNCIONES

FUNCIONES TRIGONOMÉTRICAS	
Directas	**Inversas**
sin(x)	**asin(x)**
cos(x)	**acos(x)**
tan(x)	**atan(x)**
csc(x)	**acsc(x)**
sec(x)	**asec(x)**
cot(x)	**acot(x)**
FUNCIONES HIPERBÓLICAS	
sinh(x)	**asinh(x)**
cosh(x)	**acosh(x)**
tanh(x)	**atanh(x)**
csch(x)	**acsch(x)**
sech(x)	**asech(x)**
coth(x)	**acoth(x)**
FUNCIONES EXPONENCIALES Y LOGARÍTMICAS	
exp(x)	**Función exponencial base e**
log10(x)	**Logaritmo decimal**
log(x)	**Logaritmo natural**
sqrt(x)	**Raíz cuadrada**
abs(x)	**Valor absoluto**
NÚMEROS COMPLEJOS	
abs(z)	**Módulo del complejo z**
angle(z)	**Argumento del complejo z**
conj(z)	**Conjugado del complejo z**
real(z)	**Parte real del complejo z**
imag(z)	**Parte imaginaria del complejo z**
factorial(n)	**n! = n(n-1)(n-2)(n-3)…..3.2.1**

Ejemplos:

Calcular las siguientes expresiones en Matlab

a) $y = e^{\sqrt{x^2+2x-5}}$ para x = 2.5

```
>> y = exp(sqrt(x^2+2*x-5))
```

b) y = 2sen(5x) + 3cos(2x) para x = 30º

```
>> x = 30*pi/180
>> y = 2*sin(5*x) + 3*cos(2*x)
```

c) y = $\log \sqrt[3]{x+5}$ + ln(x^2)

```
>> y = log10(x + 5)^(1/3) + log(x^2)
```

d) Para el número complejo z = 4.5 + j 5.6 hallar el módulo y argumento

```
>> z = 4.5 + 5.6i
>> mag = abs(z)          % módulo
>> ang = angle(z)        % argumento
>> ang = ang*180/pi      % argumento en grados
>> Parte_Imag=imag(z)
>> Parte_Real=real(z)
>> Conjugado=conj(z)
```

2. POLINOMIOS

Los comandos usados por Matlab para trabajar con polinomios son:

p = poly(r)	Da los coeficientes del polinomio P cuyas raíces son el vector r
y = polyval(p,x)	Evalúa el polinomio p en el valor de x
r = roots(c)	Encuentra las raíces del polinomio c
p = polyfit(x,y,n)	Polinomio de orden n que ajusta los puntos (x,y)
s = solve('ecuacion1','ecuacion2')	Resuelve las ecuaciones
d = det(A)	Calcula determinante de A

Ejemplos:

a) >> p=poly([2 3 4])

```
p =
    1    -9    26   -24
```

% El polinomio es $x^3 – 9x^2 +26x – 24$

b) Para x = 2.5 calcular $y = x^4 – 3x^2 + 5x -2.8$

```
>> x = 2.5;
>> p = [1 0 -3 5 -2.8];
>> y = polyval(p,x)


y =

   4.0750
```

c) Encontrar las raíces de: $x^5 – 3x^3 + x^2 -5x + 2$

```
>> c = [1 0 -3 1 -5  2];
>> r = roots(c)

r =
  -2.1716
   1.8905
  -0.0575 + 1.1076i
  -0.0575 - 1.1076i
   0.3960
```

d) Calcular el polinomio interpolador de segundo orden que pasa por los puntos (-1,4), (0,2) y (1,6)

```
>> x = [-1,0,1]; y = [4,2,6];
>> p = polyfit(x,y,2)

p =
    3.0000    1.0000    2.0000
```

El polinomio interpolador que más se ajusta es $3x^2 + x + 2$

e) Calcular $\sqrt{1-x} + \sqrt{1+x} = 4$

```
>> s = solve('sqrt(1-x)+sqrt(1+x)=4')

s =
   4*i*3^(1/2)
 -4*i*3^(1/2)

>> eval(s)
```

s(1)=6.9281i s(2)=-6.9281

f) Resolver el sistema de ecuaciones:

2x + 3y =5

x – 2y = -2

```
>> [x,y] = solve('2*x + 3*y = 5','x - 2*y = -2')

% x = 4/7, y = 9/7
```

g) Calcular el determinante de la matriz: $\begin{bmatrix} 2 & 4 & -1 \\ 3 & -2 & 5 \\ -1 & 3 & 6 \end{bmatrix}$

```
>> A = [2 4 -1;3 -2 5;-1 3 6];
>> d = det(A)

% d = -153
```

3. REPRESENTACIÓN GRÁFICA

Matlab ofrece diversas formas de representación gráfica.

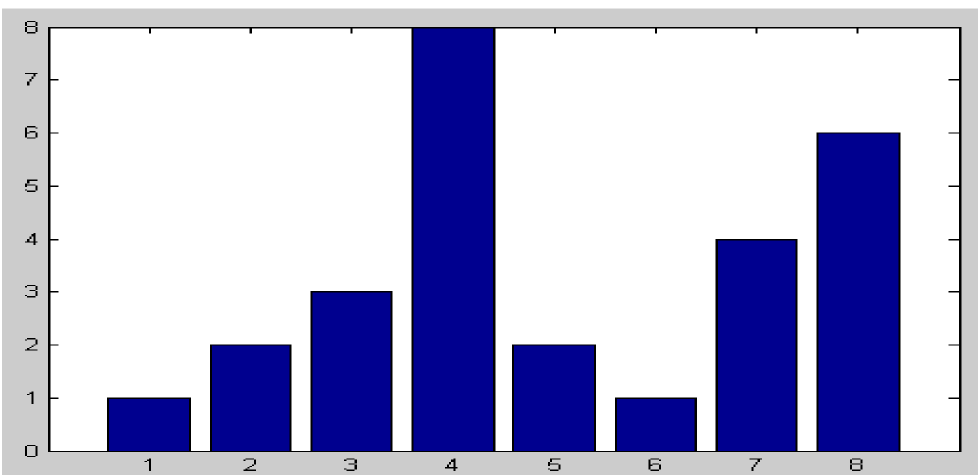
COMANDO MATLAB	DESCRIPCIÓN
bar(y)	Gráfica de barras relativo al vector y
bar(x,y)	Gráfica de barras al vector y; x define el eje x
barh(….)	Gráfica de barras horizontales
bar(….;'color')	Color = r, g, y, c, m, k
bar(y,'estilo')	Estilo=grouped (agrupado), stacked (anidado)

bar3(y,…)	Barras en tres dimensiones
plot(x,y)	Grafica y en función de x
plot(x,y,'bo')	Grafica y en función de x on color y caracter
fplot('f',[x1 x2],'y*')	Grafica función f entre x1 y x2
fplot('[f1,f2,..]',[x1 x2])	Grafica las funciones en el intervalo dado
tittle('texto')	Título de la gráfica
xlabel('texto'), ylabel('texto')	Rótulos en el eje x y en el eje y
grid	Pone rejilla en la gráfica
axis([x1 x2 y1 y2])	Define límite de los ejes
legend('rotulo1','rotulo2',….)	Coloca legenda en la gráfica
text(x,y,'texto')	Coloca texto en coordenadas (x,y)
subplot(m,n,p)	Subgráficas de m filas, n columnas

Ejemplos:

a) >> y=[1 2 3 8 2 1 4 6];

>> bar(y)

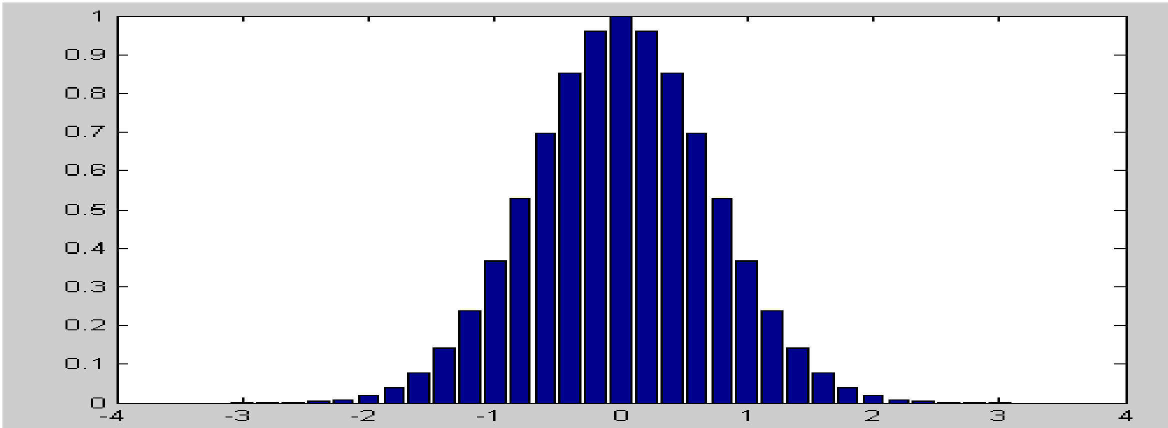


b) gráfico de barras para la función y = e^{-x*x} cuando x varía de -3 a 3

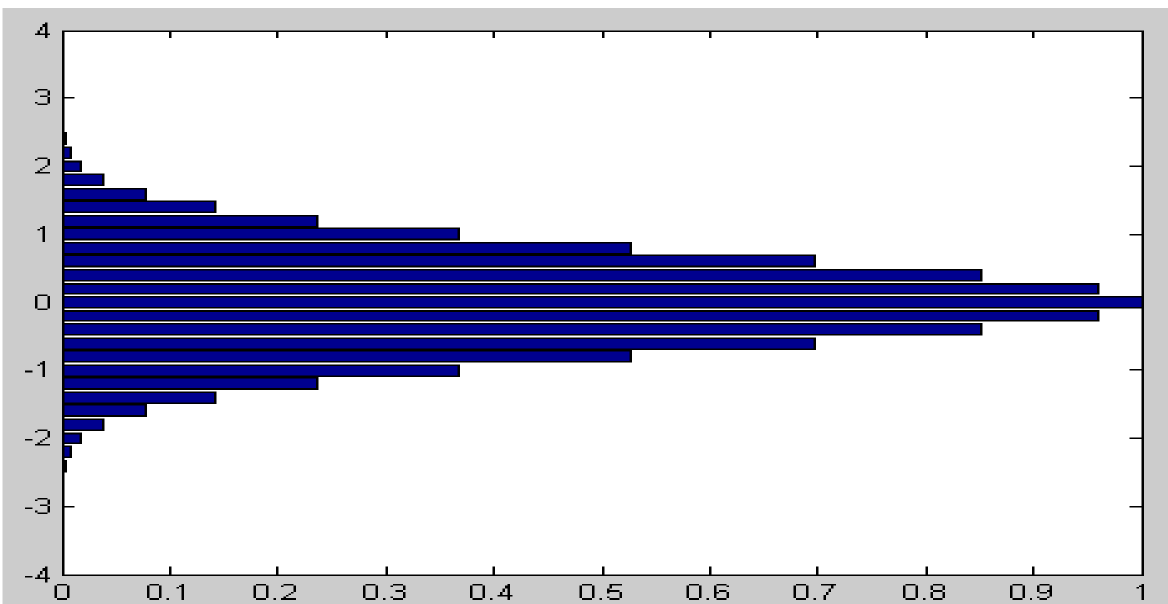
>> x = -3:0.2:3;

>> y = exp(-x.*x);

>> bar(x,y)



c) barh(x,y)



d) Ejecutar

```
>> bar(x,y,'g')
```

e) y =

```
   10    8    6
    2    5    8
```

6	0	9
5	8	7
9	4	2

Ejecutar >> bar(y,'grouped')

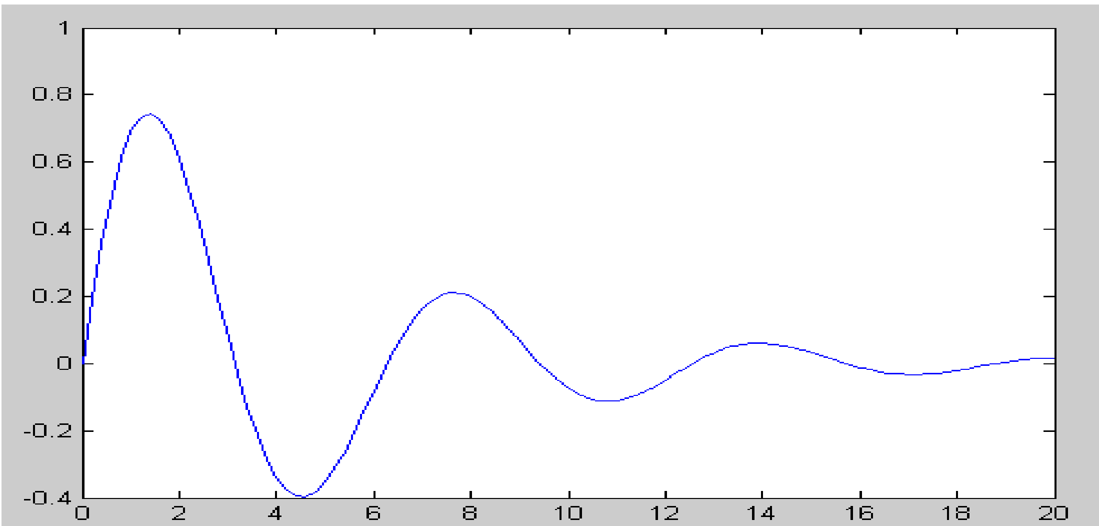
>> bar(y,'stacked')

>> bar3(y,'stacked')

f) Ejecutar >> x = 0:0.2:20;

>> y = sin(x).*exp(-0.2*x);

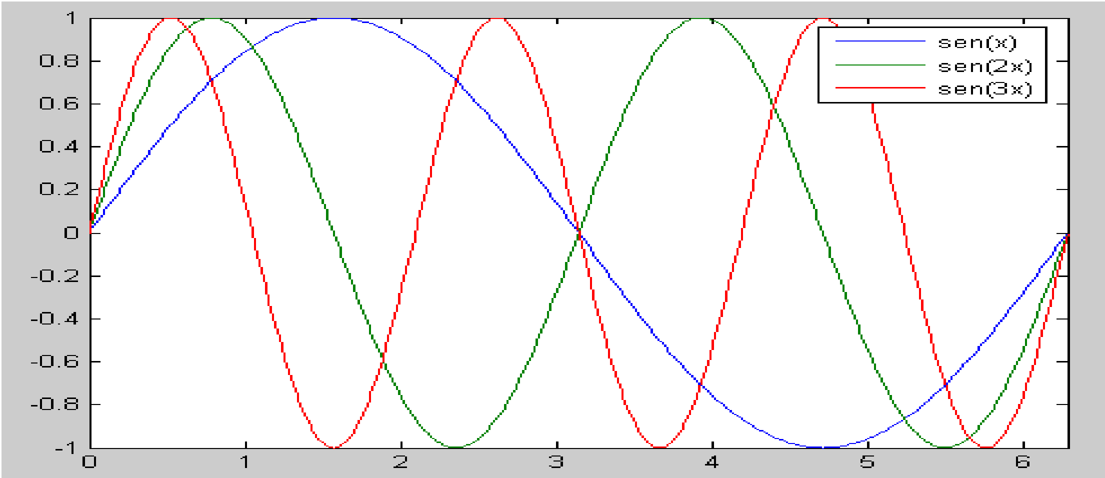
>> plot(x,y)



>> plot(x,y,’r*’)

g) Ejecutar >> fplot('[sin(x), sin(2*x), sin(3*x)]',[0,2*pi])

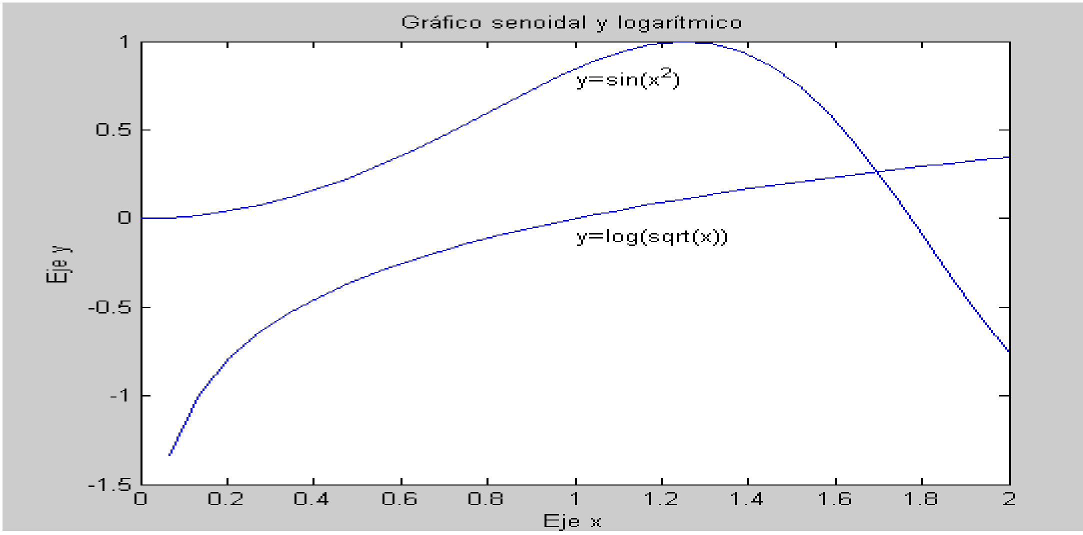
>> legend('sen(x)','sen(2x)','sen(3x)')



h) Ejecutar en la ventana de edición el programa:

```
x=linspace(0,2,30);
y=sin(x.^2);
plot(x,y)
text(1,0.8,'y=sin(x^2)');
hold on
z=log(sqrt(x));
plot(x,z)
text(1,-0.1,'y=log(sqrt(x))')
xlabel('Eje x')
ylabel('Eje y')
title('Gráfico senoidal y logarítmico')
```

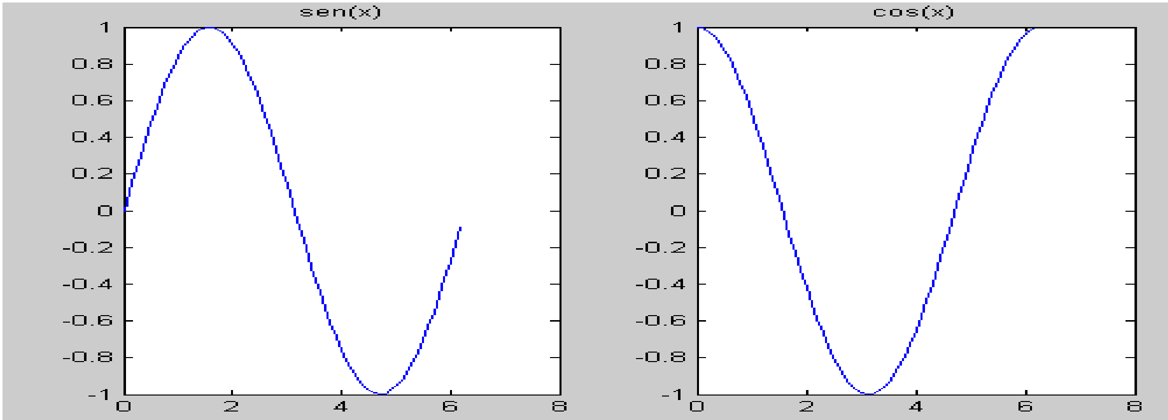
Copie y pegue en la ventana de comando y ejecute, se obtendrá la gráfica:



Ejemplos:

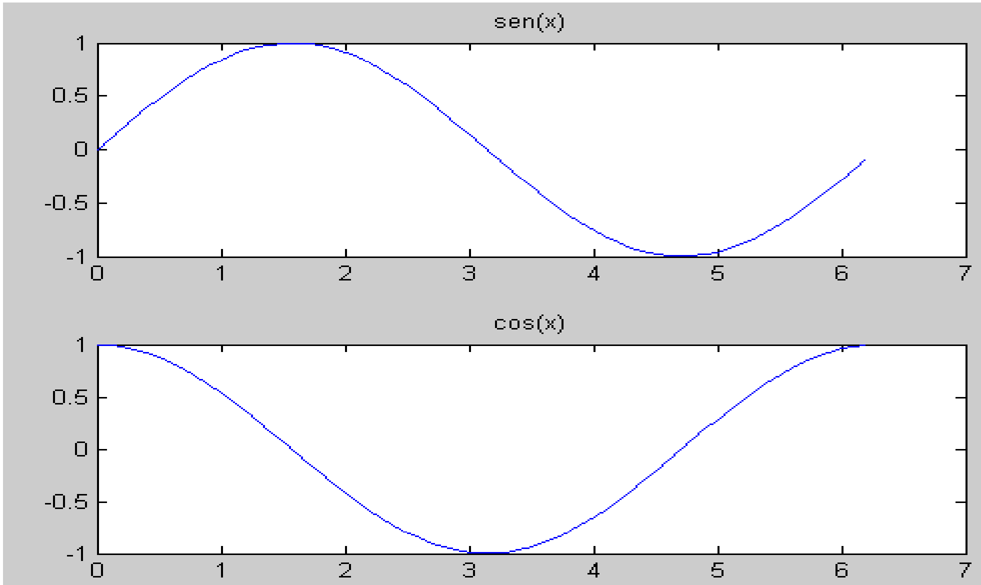
a) Graficar en dos subgráficas una fila y dos columnas:

```
x = [0:0.1:2*pi];
y = sin(x);
z = cos(x);
subplot(121);
plot(x,y)
title('sen(x)')
subplot(122);
plot(x,z)
title('cos(x)')
```



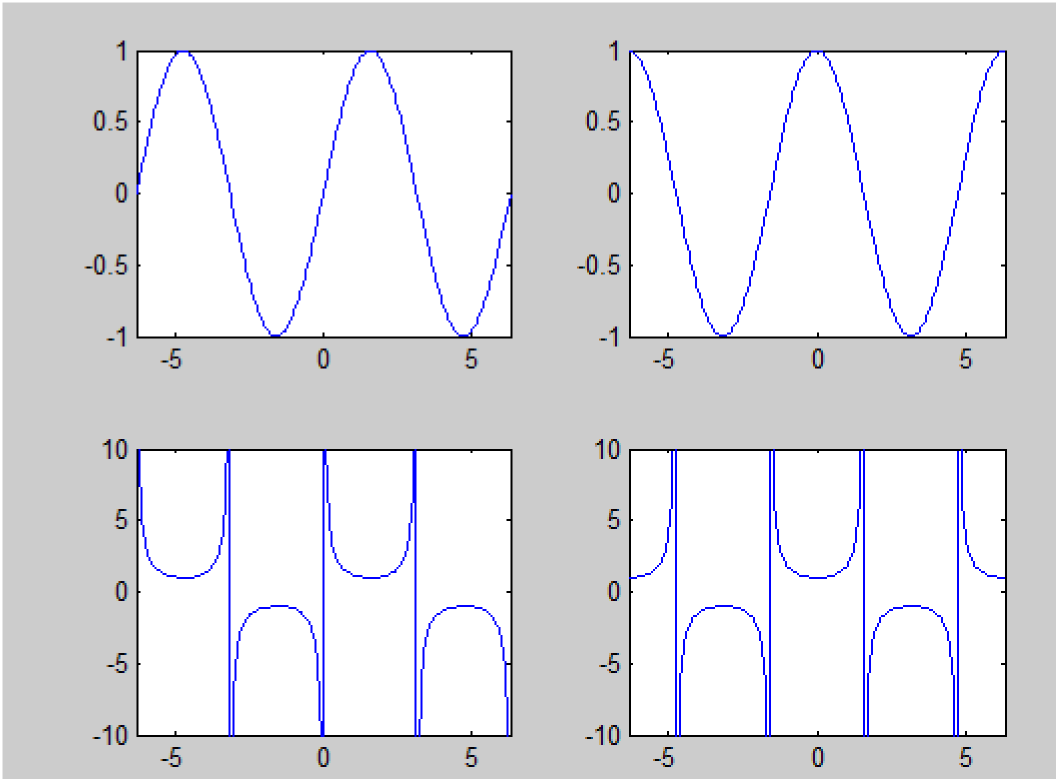
b) Graficar en dos subgráficas dos fila y una columna:

```
x = [0:0.1:2*pi];
y = sin(x);
z = cos(x);
subplot(211);
plot(x,y)
title('sen(x)')
hold on
subplot(212);
plot(x,z)
title('cos(x)')
```



c) Graficar en cuatro subgráficas dos filas y dos columnas:

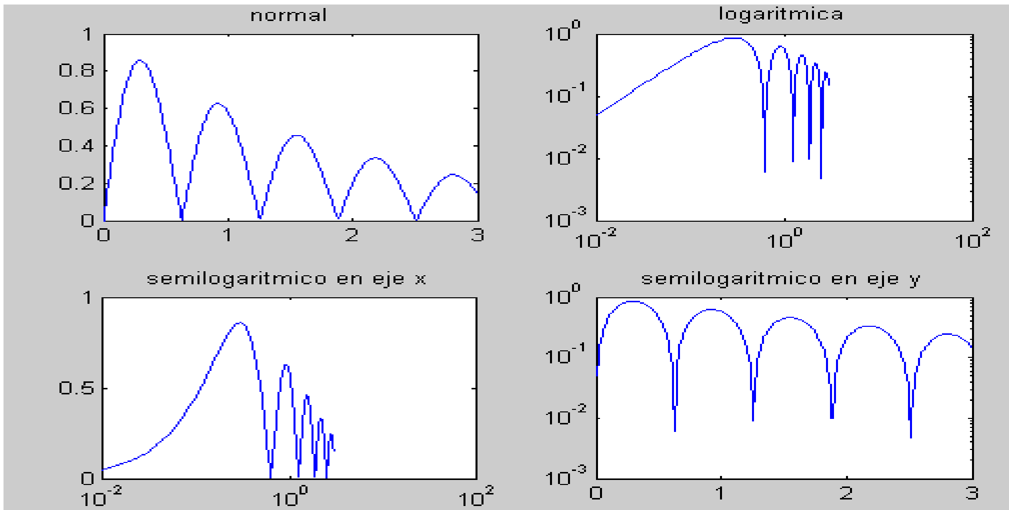
```
subplot (221);
fplot('sin(x)',[-2*pi 2*pi]);
subplot (222);
fplot('cos(x)',[-2*pi 2*pi]);
subplot (223);
fplot('csc(x)',[-2*pi 2*pi -10 10]);
subplot (224);
fplot('sec(x)',[-2*pi 2*pi -10 10]);
```



d) Graficar en diferentes escalas

```
x = 0:0.01:3;
y = abs(exp(-0.5*x).*sin(5*x));
subplot(221);
plot(x,y)
title('normal')
hold on
subplot(222)
loglog(x,y)
title('logaritmica')
subplot(223)
semilogx(x,y)
title('semilogaritmico en eje x')
subplot(224)
semilogy(x,y)
```

title(‘semilogaritmico en eje y’)



4. CÁLCULO NUMÉRICO

4.1 LÍMITES

OPERACIÓN MATEMÁTICA	COMANDO MATLAB
$\lim_{x \to 0} f(x)$	limit(f,x,0)
$\lim_{x \to a} f(x)$	limit (f, x, a) o limit (f, a)
$\lim_{x \to a^-} f(x)$	limit (f,x,a, ‘left’)
$\lim_{x \to a^+} f(x)$	limit (f,x,a, ‘right’)

Ceduvirt

Ejemplos:

a) Hallar f ´(x) = $\lim_{h \to 0} \frac{f(x+h) - f(x)}{h}$ si f(x) = cos(x)

syms h n x

limit $((\cos(x+h) - \cos(x))/h, h, 0)$

ans =

-sin(x)

b) Hallar el límite de la sucesión: $\lim_{n \to \infty} \left(\frac{-3+2n}{-7+3n} \right)^4$

>> limit (((2*n-3)/(3*n-7))^4, inf)

ans =

16/81

c) Hallar $\lim_{x \to 0^-} \frac{x}{|x|}$

>> limit $(x/abs(x), x, 0, `left`)$

ans = -1

d) Hallar $\lim_{x \to 0^+} \frac{x}{|x|}$

>> limit $(x/abs(x), x, 0, `right`)$

ans = 1

e) >> limit $\left(x/abs(x),x,0\right)$

ans =

NaN (not number)

(no existe)

Ejemplos:

Hallar el límite de las funciones:

a) $\lim_{x \to 2} \dfrac{x-\sqrt{2+x}}{-3+\sqrt{1+4x}}$,

b) $\lim_{x \to 0} \dfrac{sen[(ax)]^2}{x^2}$

```
>> syms x a
>> limit((x-(2+x)^(1/2))/(-3+(1+4*x)^(1/2)),2)
```

ans = 9/8

```
>> limit(sin(a*x)^2/x^2,x,0)
```

ans = a^2

4.2 DERIVADAS

OPERACIÓN MATEMÁTICA	COMANDO MATLAB
$\frac{\partial f}{\partial x}$	diff(x) o diff(f,x)
$\frac{\partial f}{\partial t}$	diff(f,t)
$\frac{\partial^n f}{\partial b^n}$	diff(f,b,n)

Ejemplos:

a) Hallar la derivada con respecto a x de f(x) = sen(5x)

>> syms x

>> f = sin$(5*x)$

>> diff (f)

ans = 5$*$cos $(5*x)$

b) $g(x)=e^x\cos(x)$

>> g = exp(x) $*$ cos(x)

>> diff (g)

ans = exp(x)*cos(x)-exp(x)*sin(x)

En estos ejemplos, Matlab simplifica, en otros casos, se debe usar el comando: *simplify*

Para una constante también se debe definir como simbólica:

Ejemplo: diff (5)

ans = [ ]

c = sym('5')

diff(c)

ans = 0

Ejemplos:

a) Hallar la derivada de la función f(t) = sen(st): $\frac{\partial f}{\partial t}$

>> syms s t

>> f = sin(s*t)

>> diff(f,t)

ans = cos(s*t)*s

b) Hallar la derivada con respecto a s: $\frac{\partial f}{\partial s}$

>> diff(f,s)

ans = cos(s*t)*t

c) Hallar la segunda derivada de f con respecto a t: $\frac{\partial^2 f}{\partial t^2}$

>> diff(f,t,2)

ans = -sin(s*t)*s^2

d) Hallar la derivada con respecto a x de: $f = x^n$

>> f = x ^ n

>> F = diff(f)

F = x ^ n * n / x

>> simplifity (F)

= x ^ (n-1) $*$ n

e) f(x) = log(sen(2x))

>> syms x

>> diff(log(sin(2*x)))

ans = 2*cos(2*x)/sin(2*x)

Ejemplos:

f(x,y) = sen(xy)+cos(xy^2)

Calcular:

a) $\frac{\partial f}{\partial x}$

>> syms x y

>> f = sin(x*y)+cos(x*y^2)

>> diff(f,x)

ans = cos(x*y)*y-sin(x*y^2)*y^2

b) $\frac{\partial f}{\partial y}$

>> diff(f,y)

ans = cos(x*y)*x-2*sin(x*y^2)*x*y

c) $\frac{\partial^2 f}{\partial x^2}$

>> diff(diff(f,x),x)

ans = -sin(x*y)*y^2-cos(x*y^2)*y^4

d) $\frac{\partial^2 f}{\partial y^2}$

>> diff(diff(f,y),y)

Ans = -sin(x*y)*x^2-4*cos(x*y^2)*x^2*y^2-2*sin(x*y^2)*x

e) $\frac{\partial^2 f}{\partial x \partial y}$

>> diff(diff(f,x),y)

Ans = -sin(x*y)*x*y+cos(x*y)-2*cos(x*y^2)*x*y^3-2*sin(x*y^2)*y

4.3 INTEGRALES

OPERACIÓN MATEMÁTICA	**COMANDO MATLAB**
$\int f \; dx$	int (f) integral indefinida o int (f,x)
$\int_a^b f(x)dx$	int (f,x,a,b) integral definida o int (f,a,b)

$\iint f(x)dx$	Int(int(f,x)) Integral doble
$\iint f(x,y)dxdy$	Int(int(f(x,y),x),y)
$\int_a^b \int_c^d f(x,y)dxdy$	Int(int(f(x,y),x,a,b),y,c,d))

Ejemplos:

a) Hallar la integral de $\int x^n dx$

>> int (x^n)

ans = x^(n+1)/(n+1)

b) >> int(y ^(-1))

ans = log(y)

c) >> int(1/(a+u^2))

ans = 1/a^(1/2)*atan(u/a^(1/2))

d) >> f = sin(a*teta+b)

>> int(f)

ans = -1/a∗cos(a∗teta + b)

e) $\int_0^\infty \exp(-x\wedge 2)dx \Rightarrow$

>> int (exp(-x^2), x , 0, inf)

ans =1/2∗pi^(1/2)

Ceduvirt

f) $\int_{-\infty}^{\infty} e^{-ax2} dx$

>> syms a positive

>> syms x

>> f = exp (-a $*$ x^2);

>> int (f,x,-inf,inf)

ans = 1/a^(1/2) $*$ pi ^ (1/2)

Ejemplos:

a) $\int a \ln(bx) dx$

>> syms a b x

>> int(a*log(b*x),x)

Ans =   a*x*log(b*x)-a*x

b) $\iint a \ln(xy) dxdy$

>> int(int(a*log(x*y),x),y)

ans = a*y*x*log(x*y)-2*a*x*y

c) $\int_0^1 a \ln(xy) dx$

>> int(a*log(x*y),x,0,1)

Ans = a*log(y)-a

d) $\int_0^1 \int_2^3 a \ln(xy)dxdy$

```
>> int(int(a*log(x*y),x,2,3),y,0,1)
Ans = -2*a*log(2)+3*a*log(3)-2*a
```

5. DINÁMICA DE SISTEMAS

5.1. SISTEMAS

Un sistema es una combinación de componentes que actúan conjuntamente para alcanzar un objetivo específico. Un sistema es dinámico cuando la salida presente depende de las entradas pasadas y es estático cuando la salida presente depende solamente de las entradas presentes**.**

Los componentes básicos de un sistema son:

a) Elementos que son las partes del sistema
b) Estructura. Se refiere a las interrelaciones y procesos entre las partes del sistema.
c) Ambiente. Relaciona el sistema con el todo. Es su entorno
d) Entradas. Son las fuentes de energía, recursos e información que necesita el sistema para su funcionamiento y que importa del ambiente
e) Salidas. Son los productos o resultados que se construye a través de la estructura y los procesos internos.

Los sistemas pueden clasificarse de las siguientes maneras:

a) Sistemas de lazo abierto y
b) Sistemas en lazo cerrado, que son los que realimentan parte de su salida a la entrada.

Ejemplo de un sistema hidráulico en lazo abierto y cerrado son los siguientes:

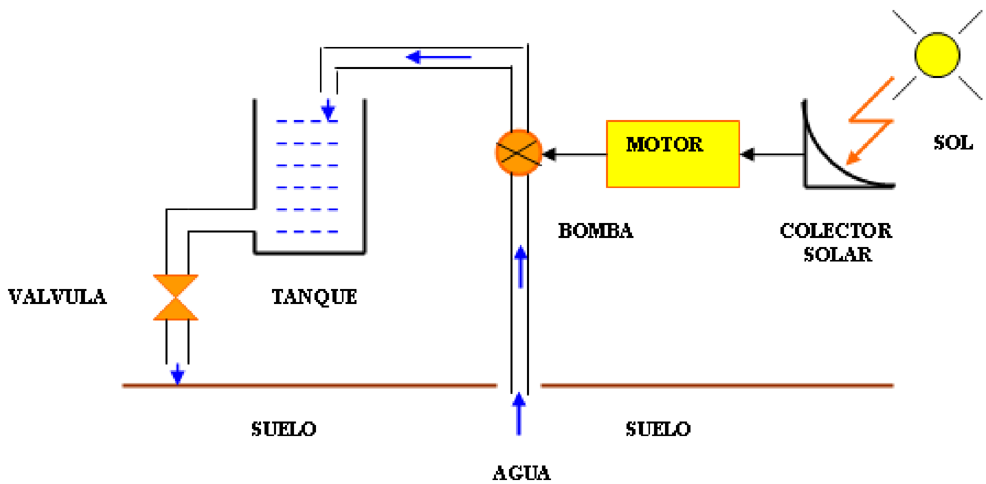


FIGURA: SISTEMA EN LAZO ABIERTO

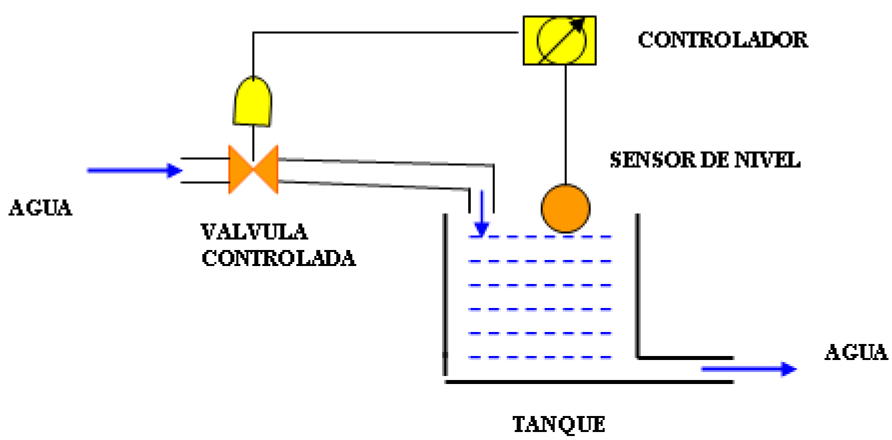


FIGURA: SISTEMA DE LAZO CERRADO

5.2 MODELO MATEMÁTICO

Es la descripción matemática que predice el funcionamiento del sistema. En los sistemas físicos el modelo matemático se describe por ecuaciones diferenciales. Los sistemas lineales se modelan con ecuaciones diferenciales lineales y son aquellos que se les aplica el principio de superposición, esto es, la respuesta de un sistema a varias entradas simultáneas es la suma de las respuestas individuales.

Para elaborar un modelo:

a) Se debe dibujar un diagrama esquemático del sistema y definir las variables.
b) Escribir las ecuaciones utilizando las leyes físicas de cada componente, combinándolos de acuerdo al diagrama y obtener el modelo matemático
c) Verificar la validez del modelo comparando la predicción de las ecuaciones del modelo con los resultados experimentales. El modelo se debe ajustar hasta que haya una buena concordancia entre lo teórico y lo práctico.

En general, la construcción de modelos se basa en la observación del sistema. Existen algunos caminos básicos para obtener un modelo:

Modelamiento de Sistemas: Esta estrategia consiste en descomponer (abstractamente) el sistema en subsistemas más simples, cuyos modelos sean factibles de obtener gracias a la experiencia previa. Una vez obtenidos estos submodelos, se buscan las relaciones que existen entre ellos, para interconectarlos y obtener el modelo del sistema original.

Esta estrategia busca una descripción desde adentro del sistema, generalmente basada en el conocimiento de las leyes que rigen los sistemas simples. El modelo así obtenido se conoce como Modelo de Caja Blanca, o Modelo Interno

Identificación de Sistemas: Esta estrategia consiste en acumular un número suficiente de observaciones sobre las señales de entrada y salida del sistema, con el propósito de emplearlas para construir un modelo del mismo. No se centra en lo que existe al interior del sistema, sino en su comportamiento respecto al entorno. El modelo así obtenido se conoce como Modelo de Caja Negra, o Modelo Entrada – Salida

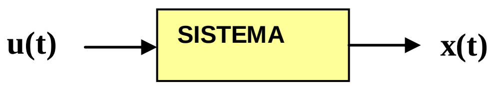
Estrategia híbrida: Existe una tercera estrategia, que realmente es una combinación de las anteriores: Al igual que en la estrategia de modelamiento, se emplea el conocimiento que esté a la mano acerca de la estructura interna del sistema y las leyes que rigen su comportamiento, y se emplean observaciones para determinar la información que haga falta. El modelo así obtenido se conoce como Modelo de Caja Gris

Para un sistema continuo de una única entrada y una única salida, el modelo empleado corresponde a una ecuación diferencial ordinaria de coeficientes constantes:

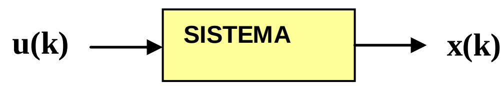
$$a_n \frac{d^n x}{dt^n} + \ldots + a_1 \frac{dx}{dt} + a_0 x(t) = b_m \frac{d^m u}{dt^m} + \ldots + b_1 \frac{du}{dt} + b_0 u(t)$$

Por su parte, un sistema discreto de una única entrada y una única salida, tendrá por modelo una ecuación de diferencias finitas ordinaria de coeficientes constantes:

$$a_n x(k+n) + \ldots + a_1 x(k+1) + a_0 x(k) = b_m u(k+n) + \ldots + b_1 u(k+1) + b_0 u(k)$$



SISTEMA CONTINUO

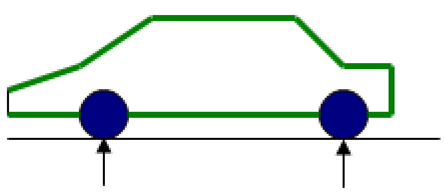


SISTEMA DISCRETO

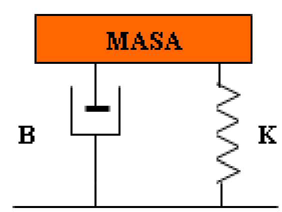
Otro tipo de ecuaciones diferenciales que se emplearán relacionan vectores de variables mediante matrices.

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} \begin{bmatrix} x_1(t) \\ x_2(t) \\ x_3(t) \end{bmatrix} \qquad \begin{bmatrix} x_1(k+1) \\ x_2(k+1) \\ x_3(k+1) \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} \begin{bmatrix} x_1(k) \\ x_2(k) \\ x_3(k) \end{bmatrix}$$

5.3 SISTEMA MECÁNICO



SISTEMA DE AMORTIGUACIÓN



MODELO DEL SISTEMA

Es el sistema de amortiguación de un automóvil, donde:

B es el coeficiente de amortiguación que depende de la velocidad (*v*)

K es la constante del resorte que depende de la elongación (x)

x_0 = Posición inicial

x_i = Posición final

Leyes de Modelo:

$$F_K = Kx, \qquad F_B = B\frac{dx}{dt}, \qquad F = ma = m\frac{d^2x}{dt^2}$$

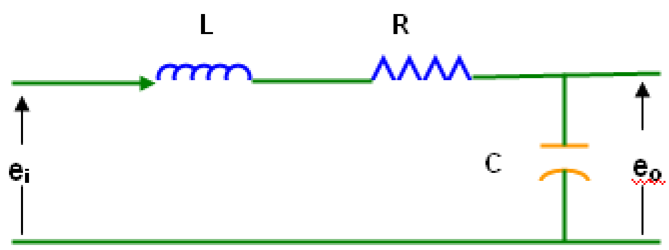
Ecuación dinámica:

$$m\frac{d^2x}{dt^2} + B(\frac{dx_o}{dt} - \frac{dx_i}{dt}) + K(x_0 - x_i) = 0$$

Ecuación diferencial:

$$m\frac{d^2x_0}{dt^2} + B\frac{dx_o}{dt} + Kx_0 = B\frac{dx_i}{dt} + Kx_i$$

5.4 SISTEMA ELÉCTRICO



Las variables del sistema son:

e_i = Voltaje de entrada

e_o = Voltaje de salida

R = Resistencia

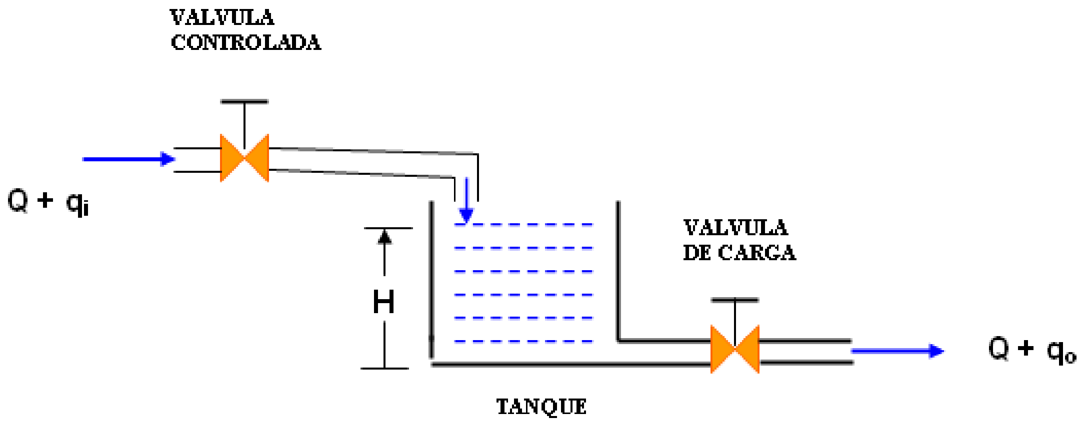
L = Inductancia

C = Capacitancia

Ecuación dinámica (Ecuación diferencial):

$$(1)\quad L\frac{di}{dt}+Ri+\frac{1}{C}\int idt=e_i,\qquad (2)\quad \frac{1}{C}\int idt=e_o$$

5.5 SISTEMA HIDRÁULICO



Variables del sistema:

Q = Caudal estable

q_i = Variación del caudal de entrada

q_o = Variación del caudal de salida

H = Altura estable

h = Variación de la altura

C = Capacitancia del tanque

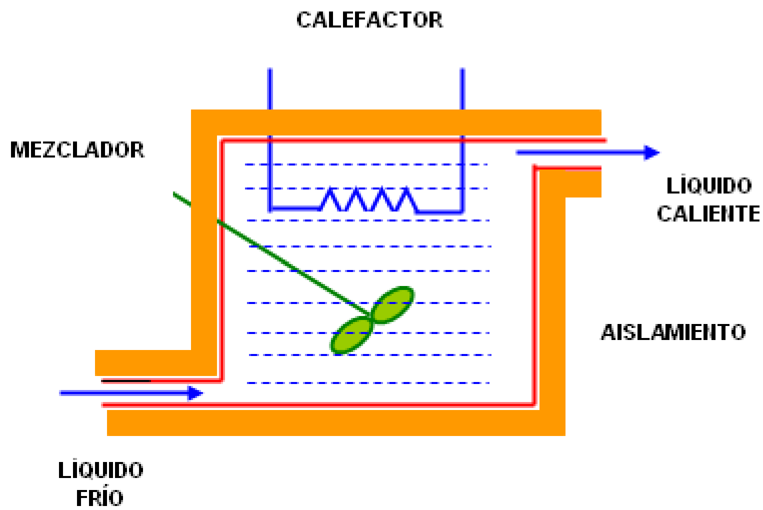
R = Resistencia hidráulica

Ceduvirt

Ecuaciones dinámicas (Ecuación diferencial):

$$(1)\quad C\frac{dh}{dt} = q_i - q_o, \qquad (2)\quad RC\frac{dh}{dt} + h = Rq_i, \qquad q_o = \frac{h}{R}$$

5.6 SISTEMA TÉRMICO



Variables del sistema:

H = Entrada de calor en estado estable

h = Cambio de calor

θ = Temperatura

Ecuación dinámica (Ecuación diferencial):

$$(1)\quad C\frac{d\theta}{dt} = h_i - h_o, \qquad (2)\quad RC\frac{d\theta}{dt} + \theta = Rh_i, \qquad h_o = \frac{\theta}{R}$$

6. TRANSFORMADA DE LAPLACE

Es un método operativo que resuelve ecuaciones diferenciales convirtiéndolas en ecuaciones algebraicas. Permite predecir el comportamiento de un sistema sin necesidad de resolver las ecuaciones diferenciales del sistema.

Sea f(t) un función en el tiempo,
La Transformada de Laplace de f(t) es:

$$\mathcal{L}\,[f(t)]=\int_0^\infty f(t)e^{-st}dt$$

6.1 FUNCIÓN ESCALÓN

f(t) = A, para t > 0 y f(t) = 0, para t < 0

$$\mathcal{L}\,[f(t)] = \int_0^\infty Ae^{-st}dt = \left.\frac{-Ae^{-st}}{s}\right|_0^\infty = \frac{A}{s} = F(s)$$

6.2 FUNCIÓN RAMPA

f(t) = At, para t > 0 y f(t) = 0, para t < 0

$$\mathcal{L}\,[f(t)] = \int_0^\infty Ate^{-st}dt = \frac{A}{s}\int_0^\infty e^{-st} = \frac{A}{s^2} = F(s)$$

6.3 FUNCIÓN EXPONENCIAL

f(t) = Ae^{-at}, para t > 0 y f(t) = 0, para t < 0

$$\mathfrak{L}\,[f(t)] = \int_0^\infty Ae^{-at}e^{-st}dt = A\int_0^\infty e^{-(a+s)t} = \frac{A}{s+a} = F(s)$$

6.4 FUNCIÓN SENOIDAL

f(t) = Asen(wt), para t > 0 y f(t) = 0, para t < 0

$$\mathfrak{L}\,[f(t)] = \int_0^\infty Asen(wt)e^{-st}dt = A\int_0^\infty sen(wt)e^{-st} = \frac{Aw}{s^2+w^2} = F(s)$$

6.5 DERIVACIÓN

$$\mathfrak{L}\left[\frac{d}{dt}f(t)\right] = sF(s) - f(0)$$

$$\mathfrak{L}\left[\frac{d^2}{dt^2}f(t)\right] = s^2F(s) - sf(0) - f^{'}(0)$$

Donde f(0) = f(t) cuando t = 0, y $f^{'}(0) = \frac{d}{dt}f(t) \quad cuando \quad t = 0$

6.6 INTEGRACIÓN

$$\mathcal{L}\left[\int_0^t f(t)dt\right] = \frac{F(s)}{s}$$

6.7 COMANDO MATLAB

Para calcular la Transformada de Laplace de una función se usa el comando

>> laplace(f(t))

Ejemplos:

Calcular la transformada de Laplace de:

a) f(t) = cos(wt)+sen(wt)

>> syms w t
>> laplace(cos(w*t)+sin(w*t))

ans =

s/(s^2+w^2)+w/(s^2+w^2)

b) f(t) = 3t + $2t^2$

>> laplace((3*t)+2*t^2)

ans =

3/s^2+4/s^3

c) f(t) = $3e^{-2t} - 2e^{5t}$

```
>> laplace(3*exp(-2*t)- 2*exp(5*t))

ans =

3/(s+2)-2/(s-5)
```

6.8 LAPLACE INVERSA

Dada la Transformada de Laplace F(s), la Transformada Inversa de Laplace es f(t). La forma más general de encontrar la transformada inversa es descomponer la función F(s) en fracciones parciales y luego aplicar su inversa a cada término.

$$F(s) = \frac{Q(s)}{P(s)} = \frac{r1}{s-p1} + \frac{r2}{s-p2} + ... + K$$

Se utiliza el comando Matlab **residue:**

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

Donde **num** es un vector compuesto por los coeficientes del polinomio del denominador y **den** es un vector compuesto por los coeficientes del polinomio del numerador.

Ejemplo:

$$F(s)=\frac{s+2}{s^2+2s+2}$$

```
>> num = [1 2] ;
>> den = [1 2 2];
>> [r,p,k]=residue(num,den)

r =

   0.5000 - 0.5000i
   0.5000 + 0.5000i


p =

  -1.0000 + 1.0000i
  -1.0000 - 1.0000i


k =

     [ ]
```

Las fracciones parciales son:

$$F(s)=\frac{0.5-0.5i}{s+1-i}+\frac{0.5+0.5i}{s+1+i}$$

Si el denominador es de la forma $P(s)^n$, entonces:

$$F(s)=\frac{Q(s)}{P(s)^n}=\frac{r1}{(s-p1)^n}+\frac{r2}{(s-p2)^{n-1}}+.....+\frac{rn}{(s-pn)}+K$$

Ejemplo:

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

```
>> num = [1 2] ;
>> den = [1 2 1];
>> [r,p,k]=residue(num,den)

%  r = [1 1 ], p = [-1 -1],  k = [ ]
```

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

6.9 SOLUCIÓN DE ECUACIONES DIFERENCIALES

Para resolver o solucionar una ecuación diferencial:

a) Se aplica transformada de Laplace a la ecuación
b) Se despeja F(s)
c) Se halla la transformada inversa

Ejemplo:

Resolver la ecuación diferencial:

$$\frac{d^2y}{dt^2}+3\frac{dy}{dt}+2y(t)=5, \quad \text{condiciones iniciales} \quad y(0)=-1, y^{'}(o)=2$$

a) Aplicamos transformada de Laplace:

$$s^2Y(s)-sy(0)-y^{'}(0)+3sY(s)-3y(o)+2Y(s)=5/s,$$

Reemplazando los valores de y(0) y $y^{'}$ (0), se tiene:

b) Despejando Y(s),

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

c) Aplicando fracciones parciales para hallar la inversa,

```
>> num = [-1 -1 5] ;
>> den = [1 3 2 0];
>> [r,p,k]=residue(num,den)
```

r = [1.5 -5 2.5], p = [-2 -1 0], k = [ ]

$$Y(s)=\frac{1.5}{s+2}-\frac{5}{s+1}+\frac{2.5}{s}$$

La transformada inversa es:

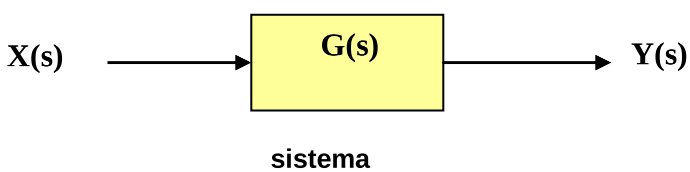
$$y(t)=1.5e^{-2t}-5e^{-t}+2.5$$

7. FUNCIÓN DE TRANSFERENCIA

La función de transferencia de un sistema descrito mediante una ecuación diferencial, se define como el cociente de la Transformada de Laplace de la salida y la Transformada de Laplace de la entrada.

El método para encontrar la función de transferencia de un sistema es el siguiente:

a) Escribir la ecuación diferencial del sistema
b) Aplicar la Transformada de Laplace de la ecuación diferencial con condiciones iniciales cero
c) Sacar el cociente entre la variable de salida y la entrada



$$G(s) = \frac{Y(s)}{X(s)} = \frac{b_0 + b_1 s + ... + b_m s^m}{a_0 + a_1 s + ... a_n s^n}$$

<u>En Matlab:</u>

```
>> num = [bo  b1  b2 ....  bm];
>>  den = [ao  a1  a2 ....  an];
>> Gs = tf(num,den)
```

Ejemplo:

Para el torque que produce un motor eléctrico (parte mecánica):

T= torque (variable de entrada)

θ = ángulo de giro (variable de salida)

J = Inercia

Ecuación diferencial: $T(t) = J\dfrac{d^2\theta}{dt^2}$

Transformada de Laplace: $T(s) = Js^2\theta(s)$

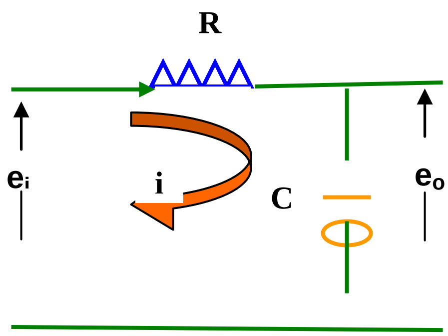
Función de transferencia: $G(s) = \dfrac{\theta(s)}{T(s)} = \dfrac{1}{Js^2}$

7.1 DIAGRAMA EN BLOQUES

La estructura de un sistema se representa por un diagrama en bloques según el siguiente procedimiento:

a) Definir la entrada y la salida
b) Escribir las ecuaciones que describa el comportamiento de cada elemento del sistema
c) Aplicar transformada de Laplace a cada elemento
d) Integrar los elementos en un diagrama completo (estructura)

Ejemplo: Circuito serie RC



a) Entrada: e_i,   salida: e_o

b) Para la resistencia

$$e_R = Ri, \quad i = \frac{e_i - e_o}{R}$$

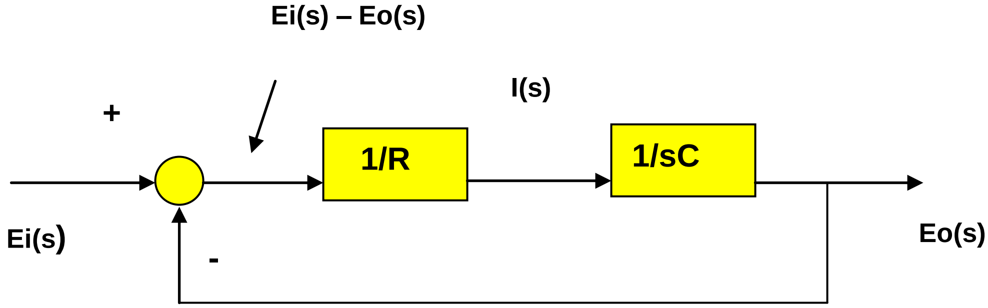
para el condensador

$$e_o = \frac{1}{C}\int i dt$$

c)Transformada de Laplace

$$I(s) = \frac{E_i(s) - E_o(s)}{R},$$
$$E_o(s) = \frac{1}{sC} I(s)$$

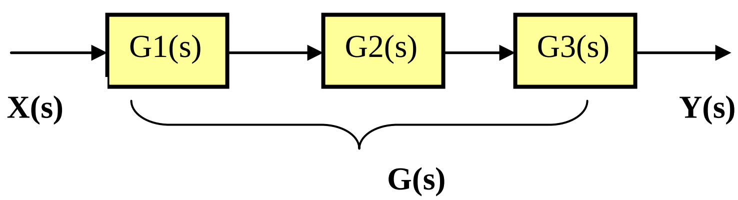
d) Estructura



7.2 TIPOS DE ESTRUCTURAS

Los diagramas de bloques mediante los cuales se estructura un sistema son complejos y son generalmente combinaciones de los siguientes tipos de estructuras:

7.2.1 ESTRUCTURA SERIE



$$\boxed{G(s) = G1(s)*G2(s)*G3(s)}$$

Matlab

G1 = tf (num1,den1);

G2 = tf (num2,den2);

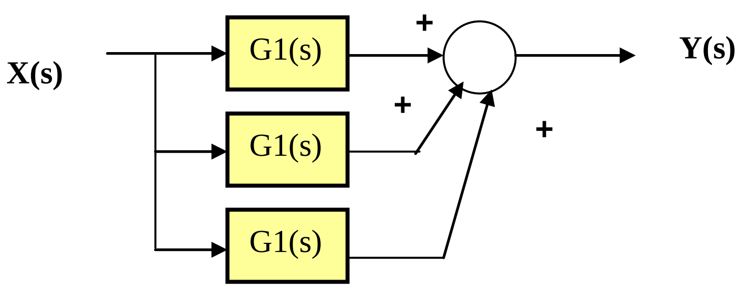
G3 = tf (num3,den3);

Gs = G1*G2*G3

también puede ser:

Gs = series(G1,G2,G3)

7.2.2 ESTRUCTURA PARALELA



G(s) = G1(s)+G2(s)+G3(s)

Matlab

G1 = tf (num1,den1);

G2 = tf (num2,den2);

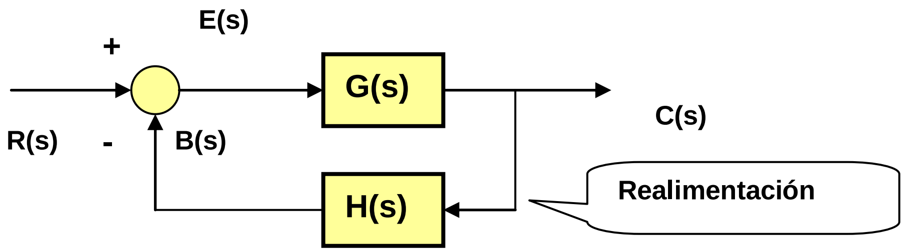
G3 = tf (num3,den3);

Gs = G1+G2+G3

también puede ser :

Gs = parallel(G1,G2,G3)

7.2.3 ESTRUCTURA REALIMENTADA



a) Función de transferencia en lazo abierto: Gla

$$Gla = \frac{B(s)}{E(s)} = G(s) * H(s)$$

b) Función de transferencia en lazo cerrado: Glc

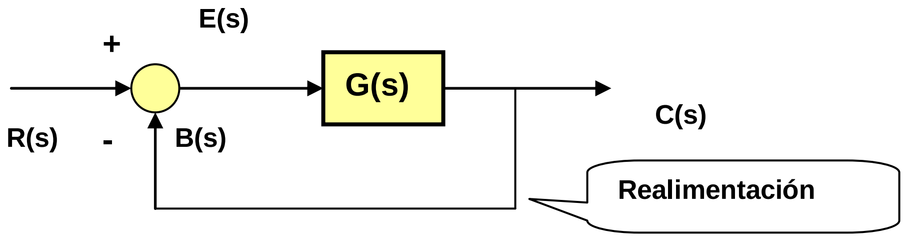
$$Glc = \frac{C(s)}{R(s)} = \frac{G(s) * E(s)}{R(s)}, \; donde, \; \mathrm{E(s) = R(s) - B(s) = R(s) - C(s) * H(s)}$$

Reemplazando,

$$C(s) = G(s) * [R(s) - C(s)H(s)] = \mathrm{G(s) * R(s) - G(s) * H(s)C(s)}$$

$$Glc = \frac{C(s)}{R(s)} = \frac{G(s)}{1 + G(s)H(s)},$$

Caso especial:



$$Gla = G(s), \qquad Glc = \frac{C(s)}{R(s)} = \frac{G(s)}{1+G(s)}, \ \mathrm{H(s)} = 1$$

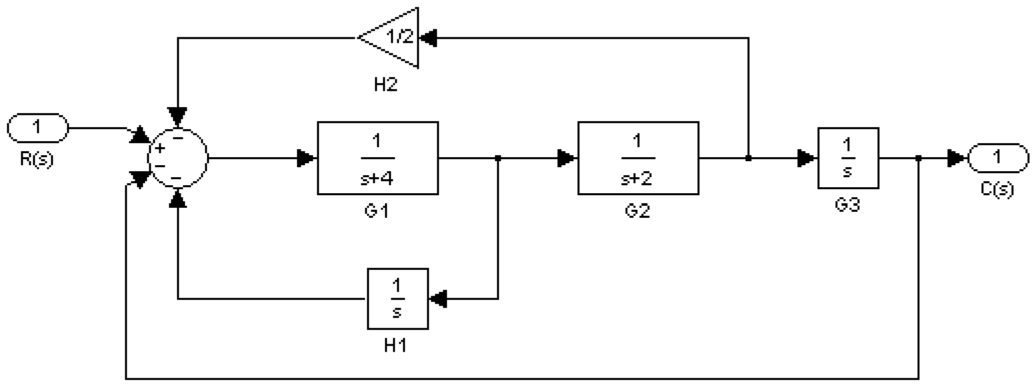
Matlab:

```
>> Gla = Gs*Hs
>> Glc = feedback(Gs,Hs)
```

Ejemplo:

Obtener la función de transferencia del sistema por Matlab:

$$G(s) = \frac{C(s)}{R(s)}$$



% Este es un programa realizado en Matlab

% para obtener la función de transferencia del sistema

G1=tf(1,[1 4])

H1=tf(1,[1 0])

% Realimentación de G1 y H1

Glc1=feedback(G1,H1)

Respuesta: $Glc1 = \dfrac{s}{s^2 + 4s + 1}$

% Estructura serie

G2=tf(1,[1 2])

G4=Glc1*G2

Respuesta: $G4 = \dfrac{s}{s^3 + 6s^2 + 9s + 2}$

% Segunda realimentación

H2=1/2

Glc2=feedback(G4,H2)

Respuesta: $Glc2 = \dfrac{s}{s^3 + 6s^2 + 9.5s + 2}$

% Segunda estructura serie

```
G3=tf(1,[1 0])
G5=Glc2*G3
```

Respuesta: $$G5=\frac{s}{s^4+6s^3+9.5s^2+2s}$$

% Tercera realimentación

```
Glc=feedback(G5,1)
```

Respuesta: $$Glc=\frac{s}{s^4+6s^3+9.5s^2+3s}$$

7.3 ESTABILIDAD

7.3.1 ECUACIÓN CARACTERÍSTICA

La ecuación característica de un sistema es el denominador de la función de transferencia del sistema igualado a cero.

Para el ejemplo anterior, la ecuación característica es igual a:

$$s^4+6s^3+9.5s^2+3s=0$$

7.3.2 POLOS Y CEROS DE UN SISTEMA

Los *polos* de un sistema son las raíces de la ecuación característica del sistema, esto es, las raíces del denominador de la función de transferencia del sistema.

Los *ceros* de un sistema son las raíces del numerador de la función de transferencia del sistema.

Para la función de transferencia:

$$G(s) = \frac{s^2 + 3s + 2}{s^3 + 6s^2 + 5s + 1}$$

Los polos y ceros son:

Matlab:

```
>> num=[1 3 2];
>> ceros=roots(num)
% ceros = -2,  -1

>> den=[1 6 5 1];
>> polos=roots(den)
% polos = -5.0489, -0.6431, -0.3080
```

En Matlab existen dos comandos más utilizados para el cálculo de los polos y ceros de un sistema que se obtienen directamente de la función de transferencia:

```
ceros = zero(Gs)
polos = pole(Gs)
```

Esto es,

```
>> num=[1 3 2];
>> den=[1 6 5 1];
>> Gs = tf(num,den);
```

```
>> ceros = zero(Gs)
>> polos = pole(Gs)
```

7.3.3 ESTABILIDAD

Con base en la gráfica de polos y ceros (eje x los reales, eje y los imaginarios) de la función de transferencia en lazo cerrado:

a) El sistema es estable cuando los polos están en el semiplano izquierdo
b) el sistema es inestable si por lo menos un polo está en el semiplano derecho
c) Es críticamente estable cuando los polos están en el eje imaginario
d) Los ceros no intervienen en la estabilidad y por tanto no importa su ubicación

Ejemplo:

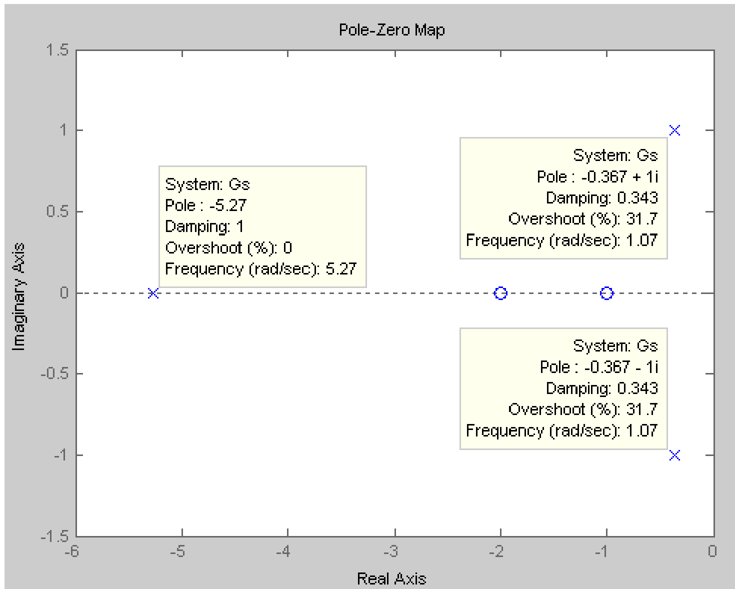
La función de transferencia en lazo cerrado de un sistema es:

$$Glc(s) = \frac{s^2 + 3s + 2}{s^3 + 6s^2 + 5s + 6}$$

Matlab:

```
>> num=[1 3 2];
>> den=[1 6 5 6];
>> Glc=tf(num,den)
>> polos=pole(Glc)
% polos =  -5.2670,  -0.3665 + 1.0024i,   -0.3665 - 1.0024i
% Para graficar los polos y ceros se usa el comando pzmap
>> pzmap(Glc)
```

% el programa presenta la siguiente figura,



El sistema tiene tres polos: un polo real y dos polos complejos conjugados. Como todos los polos están en el semiplano izquierdo, el *sistema es estable*.

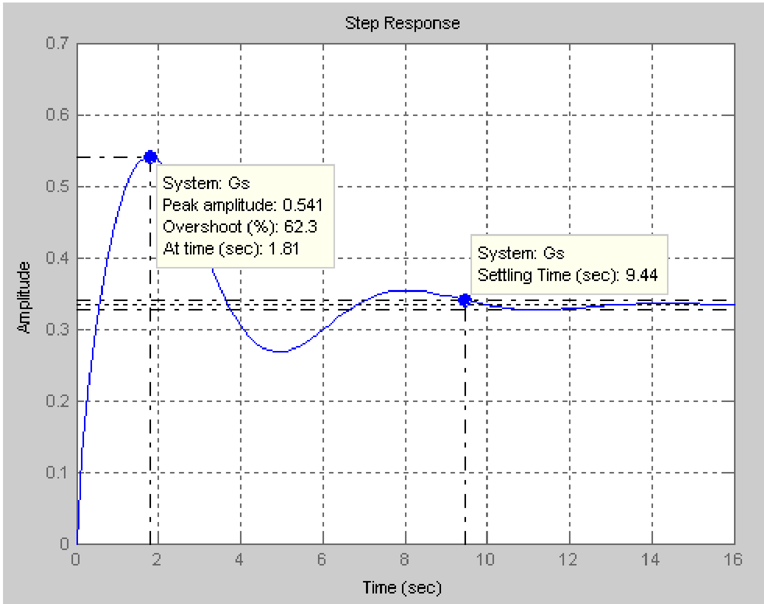
7.3.4 RESPUESTA DE UN SISTEMA

Generalmente se conoce como respuesta de un sistema la salida en el dominio del tiempo que tiene el sistema cuando a su entrada se le aplica una función escalón unitaria. También se conoce como respuesta al paso unitario.

Para el ejemplo anterior la respuesta al paso unitario se obtiene adicionando la instrucción:

```
>> step(Glc)
```

Esta respuesta tiene como característica importante la amplitud de pico, el sobreimpulso (overshoot) y el tiempo de establecimiento (setting time).



7.3.5 ERROR DE ESTADO ESTACIONARIO

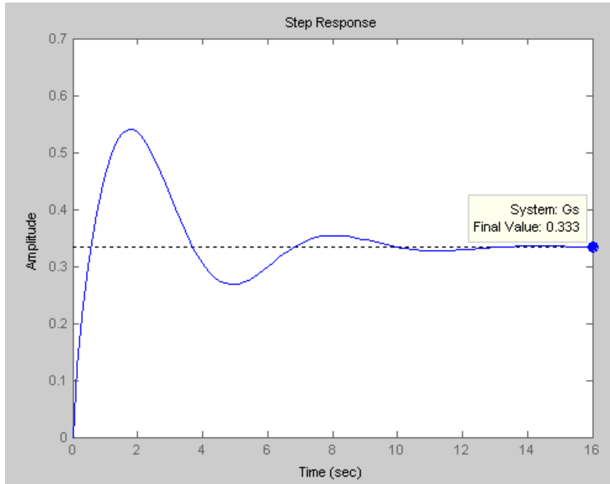
El error de estado estacionario o estado estable. es igual a:

$$Ess = 1 - \lim_{s \to 0} Glc = 1 - valor\ \mathrm{final}$$

Ejemplo:

$$Ess = 1 - \lim_{s \to 0} \frac{s^2 + 3s + 2}{s^3 + 6s^2 + 5s + 6} = 1 - 0.333 = 0.666$$

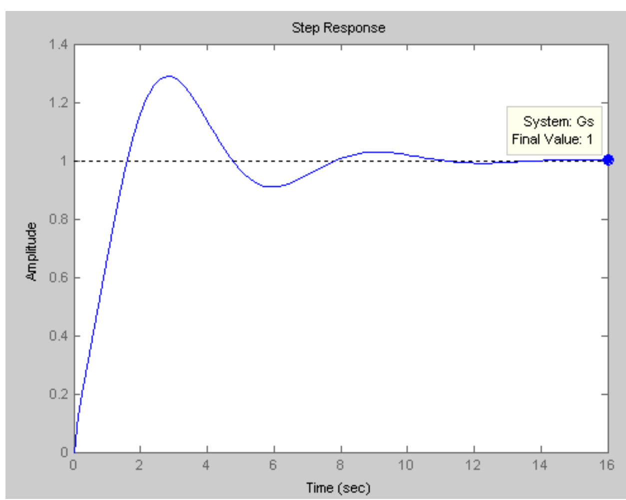
Ceduvirt



Ejemplo:

Para: $$Glc = \frac{s^2 + 3s + 6}{s^3 + 6s^2 + 5s + 6}$$

$$Ess = 1 - \lim_{s \to 0} \frac{s^2 + 3s + 6}{s^3 + 6s^2 + 5s + 6} = 1 - 1 = 0$$



8. FUNCIONES Y BUCLES

Matlab permite la ejecución de conjuntos de comandos escritos secuencialmente en la ventana de edición y que son almacenados en un archivo ***nombre.m***. Para ejecutar un archivo basta con teclear su nombre (sin extensión) en modo interactivo en la ventana de comando y pulsar ***enter***. En el archivo .m se pueden introducir textos explicativos comenzando la línea con el símbolo **%**.

8.1 FUNCTION

El comando ***function*** permite la definición de funciones con la siguiente sintaxis:

```
function  parámetros_salida = nombre_función(parámetros_entrada)
        cuerpo de la función
```

Una vez definida la función se guarda en un archivo *nombre_función.m* para su posterior utilización. Cuando los parámetros de salida son más que uno se sitúan entre corchetes separados por comas. Si los parámetros de entrada son más que uno se separan por comas.

Ejemplo:

Definir la función fun1(x) = x^3 -2x+cos(x)

En la ventana de edición:

```
function  p = fun1(x)
% Definición de una función simple
p = x^3 – 2*x+cos(x)
```

La función se guarda en un archivo fun1.m

Podemos luego utilizar esta función, por ejemplo,

```
>> fun1(3*pi/2)
ans=
        95.2214
```

```
>> help fun1
Definición de una función simple
```

Ejemplo:

Solución de una ecuación de segundo grado

```
function [x1,x2] = cuadratica(a,b,c)
%Esta funcion calcula las raices de una ecuacion cuadratica
%la sintaxis es [x1,x2]=cuadratica(a,b,c)

radical = sqrt(b^2-4*a*c);
x1= (-b+radical) / (2*a);
x2= (-b-radical) / (2*a);
```

el archivo es cuadratica.m

```
>> help cuadratica

Esta funcion calcula las raices de una ecuacion cuadratica
 la sintaxis es [x1,x2]=cuadratica(a,b,c)

>> [x1,x2]=cuadratica(1,2,3)
```

x1 =

  -1.0000 + 1.4142i

x2 =

  -1.0000 - 1.4142i

FVAL:

La evaluación de una función en sus argumentos, también puede realizarse con el comando ***feval*** que tiene la siguiente sintaxis:

feval('F',arg1,arg2,…)

Evalúa la función F (archivo F.m) en los argumentos especificados arg1, arg2, …..

Ejemplo:

```
function  [x1,x2] = ecuacion2(a,b,c)
% Solución de la ecuación de segundo orden
d = b^2 – 4*a*c;
x1 = (-b + sqrt(d))/(2*a);
x2 = (-b - sqrt(d))/(2*a);
```

Para resolver la ecuación $x^2 + 2x +3 = 0$

```
>> [x1, x2] = feval('ecuacion2',1,2,3)
```

o también:

```
>> [x1, x2] = ecuacion2(1,2,3)
```

8.2 GLOBAL

Normalmente cada función de Matlab definida como un archivo–M contiene sus variables como variables locales, esto es, su efecto es al interior de este archivo independientemente de otros archivos. Es posible definir otras variables que tenga efecto en otros archivos-M con variables globales usando el comando ***global*** con la siguiente sintaxis:

global x y z ...   define las variables x, y, z,....como globales

8.3 FOR

Permite ejecutar de forma repetitiva un comando o grupo de comandos varias veces. Tiene la siguiente sintaxis:

caso1:

```
for  i= 1:n
        comandos
end
```

caso2:

```
for i=n:-0.2:1
        comandos
end
```

caso 3:

```
for i=1:m
        for i=n
        comandos
        end
end
```

Ejemplo:

Sumar los enteros pares de 1 a 100

```
suma=0;
for i=1:2:100
   suma=suma+i;
end
disp('El resultado es:  ')
display(suma)
```

Ejemplo:

Calcular la suma de los elementos de una matriz

```
M=[1 2 -2 4; 0 -3 1 0; 2 -1 4 3];
% numero de filas
m=length(M(:,1));
% numero de columnas
```

```
n=length(M(1,:));
suma=0;

for i=1:m
   for j=1:n
      suma=M(i,j)+suma;
   end
end

display(suma)
```

8.4 WHILE

Mientras se ejecuta una condición se ejecutan los comandos o sentencias.

```
while  condición
        sentencias
end
```

Ejemplo:

Calcular los volúmenes de las esferas para radio igual a: 1,2,3,4,5

```
r=0;
while r<5
   r=r+1;
   vol=(4/3)*pi*r^3;
   fprintf('El radio es =%g y el volumen es =%g \n',r,vol)
end
```

El radio es =1 y el volumen es =4.18879

El radio es =2 y el volumen es =33.5103

El radio es =3 y el volumen es =113.097

El radio es =4 y el volumen es =268.083

El radio es =5 y el volumen es =523.599

Ejemplo:

```
Gz=tf(1,[0.4  0.3]);
Gzc=zpk(1,[-1  0.2],0.5);
SIGA=1;

while SIGA ==1
clc
disp('LA FUNCION DE TRANSF. EN LAZO ABIERTO  ES: ');
Gla = Gz*Gzc
disp(' ');
disp('LA FUNCION DE TRANSF. EN LAZO CERRADO  ES: ');
Glc = feedback(Gla,1)
disp('PARA SEGUIR OPRIMA ENTER');
pause

clc
disp(' ');
disp('LOS POLOS DEL SISTEMA SON : ');
Polos = pole(Glc)
disp('QUE TIENEN MAGNITUDES Y ANGULOS DE :');
Mag = abs(Polos)
Ang1 = angle(Polos);
Ang = Ang1*180/pi
if (Mag(1)<1)&(Mag(2)<1)
   disp('EL SISTEMA ES ESTABLE');
```

```
else
   disp('EL SISTEMA ES INESTABLE');
end
disp(' ');
disp('PARA SEGUIR OPRIMA ENTER');
pause

SIGA = input (' PRESIONE  1 PARA SEGUIR ');
end
```

Ejemplo:

Genere una tabla que suministre los inversos, cuadrados y raíces cuadradas del 1 al 5

```
i=0;
while i<5
   i=i+1;
   A(i)=i;
   B(i)=1/i;
   C(i)=i^2;
   D(i)=sqrt(i);
end
E=[A',B',C',D']
E =

   1.0000    1.0000    1.0000    1.0000
   2.0000    0.5000    4.0000    1.4142
   3.0000    0.3333    9.0000    1.7321
   4.0000    0.2500   16.0000    2.0000
   5.0000    0.2000   25.0000    2.2361
```

8.5 IF

caso 1:

```
if condición
        sentencias
end
```

caso 2:

```
if condición1
        bloque1
elseif condción2
        bloque2
elseif condición3
        bloque3
else    % sino cumple condiciones anteriores
        bloque4
end
```

Operadores de relación:

>	Mayor que
>=	Mayor o igual que
<	Menor que
<=	Menor o igual que
==	Igual
~=	No es igual a

Operadores lógicos:

&	AND
|	OR
~	NOT
xor	XOR

Ejemplo:

```
calif = input('Dame la calificacion:  ');
if  calif >= 3.0
   disp(' ')
   disp('Aprobado')
end
if calif < 3.0
   disp(' ')
   disp('Desaprobado')
end

% Otra forma

calif = input('Dame la calificacion:  ');
if  calif >= 3.0
   disp(' ')
   disp('Aprobado')
else
   disp(' ')
   disp('Desaprobado')
end
```

Ejemplo:

El precio del vino está condicionado a la cantidad requerida. Hasta 5 botellas el precio unitario es de $10.000, de 6 a 12 botellas el precio es de $12.000, y a partir de 13 a $15.000. Elaborar un programa, que pegunta cuántas botellas se desean, indique el precio unitario y el total del gasto.

```
c=input('¿Cuántas botellas quiere? ');
if c<5
   Pu=10000;
   Pt=Pu*c;
elseif c<=12
   Pu=12000;
   Pt=Pu*c;
else
   Pu=15000;
   Pt=Pu*c;
end
disp('Precio unitario: ')
Pu
disp('Precio total: ')
Pt
```

8.6 SWITCH

Se evalúa una expresión y se compara con las expresiones en *case*. Se ejecuta el bloque que corresponda con ese resultado. Si ninguno es igual se ejecuta el bloque de *otherwise.*

```
switch expresión
        case expresión1
```

```
            bloque1
        case expresión2
            bloque2
        .........................
otherwise
        bloque3
```

Ejemplo:

```
disp('SELECCIONE PRESIONANDO :');
disp(' 1: PARA FUNCION DE LA PLANTA EN TF ');
disp(' 2: PARA FUNCION DE LA PLANTA EN ZPK ');
n=input('SELECCIONE LA OPCION :  ');
disp(' ');

switch n
case 1
num = input('ENTRE NUMERADOR DE LA PLANTA : num = ');
den = input('ENTRE DENOMINADOR DE LA PLANTA : den = ');
disp('LA FUNCION DE TRANSFERENCIA DE LA PLANTA ES : Gp(s) = ');
Gp = tf(num,den)
disp(' ');

case 2
Z = input('Entre vector de ceros : Z = ');
P = input('Entre vector de polos : P = ');
K = input('Ganancia es igual a : K = ');
disp('LA FUNCION DE TRANSFERENCIA DE LA PLANTA ES : Gp(s) = ');
Gp = zpk(Z,P,K)

end
```

8.7 INPUT

Permite imprimir un mensaje y recuperar como valor el resultado de una expresión tecleada por el usuario.

```
n = input ('Teclee el polinomio')
```

8.8 DISP

Permite imprimir un mensaje en pantalla.

```
disp('Universidad de Colombia')
```

8.9 FIND

Para el ejemplo del movimiento del péndulo, calcular por programación el ángulo para un tiempo dado.

```
%Programa en Matlab
g=9.8;
w=2;
L=0.6;
B=0.08;
m=w/g;
%Para correr el simulink tiene que en K>>movpendulo y luego correrlo
disp('Realice la simulacion y de return para ejecutar')
keyboard
a=length(t);
tf=t(a);
```

```
delta=tf/a;
%calcular el angulo para un tiempo T dado
T=input('Entre el tiempo de T: ');
i=find(t <(T+delta) & t >(T-delta));
disp('El desplazamiento angular es de: ')
AngR=teta(i(2))
AngG=AngR*180/pi
```