Análisis de Señales Curso 2024

Trabajo Práctico Nº2

Sistemas lineales – con Octave aplicado

Actividad 1

Determine si los siguientes sistemas verifican las propiedades de:

- Linealidad
- Causalidad
- Invariancia
- Memoria (Dinámico)
- Estabilidad

a) $y(t) = x(t-5) - x(3-t)$

b) $y[n] = 2 \cdot x[n - n_o]$; $0 < n_o$ y $n_o \in \mathbb{Z}$

c) $y[n] = [n+3] \cdot x[n-3]$

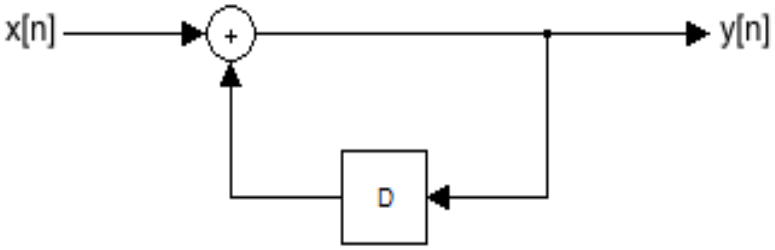
d) $y[n] = 5 \cdot n \cdot x^2[n]$

e) $y[n] = 3 \cdot x[n+3]$

f) $y(t) = \int_{-\infty}^{2t} x(\sigma)\, d\sigma$

Actividad 2

Determine que el sistema de la figura es lineal, invariante en el tiempo, dinámico e inestable:



Ejemplo 1

Queremos conocer la respuesta al impulso utilizando Octave del siguiente sistema discreto:

$$y[n] = +\frac{1}{2}y[n-1] + 2x[n] + 3x[n-1]$$

Para esto abrimos el editor seleccionando la solapa edit o escribimos editor en la línea de comandos. Allí generamos un archivo de comandos con el nombre Ej1.m, para realizar las siguientes acciones:

```
%Generar un vector con la función delta de Kronecker.
%Para esto se crea un vector de ceros con 20 valores llamado delta
%y se almacena el valor 1 en la sexta posición.
delta = zeros(1,20);
delta(6)=1;
%Se colocan los coeficientes de la señal de salida en un vector A.
A = [1 -0.5];
%Se colocan los coeficientes de la señal de entrada en un vector B.
B = [2 3];
%Se calcula la señal de salida de la ecuación lineal en diferencias
%para la señal de entrada delta.
Y = filter(B,A,delta);
%En este caso las condiciones iniciales son nulas.
%Caso contrario ver las variantes de dicho comando.
%Para realizar la gráfica de la respuesta al impulso utilizamos;
n=-5:1:14;
stem(n,Y,'filled');
grid;
title('respuesta al impulso del ejemplo1');
```

Actividad 3

Para los siguientes sistemas LIT discretos queremos conocer la respuesta al impulso utilizando Octave/Matlab/Python. Entregue las líneas de código.

a) $y[n] = 5x[n-1] - x[n-2]$

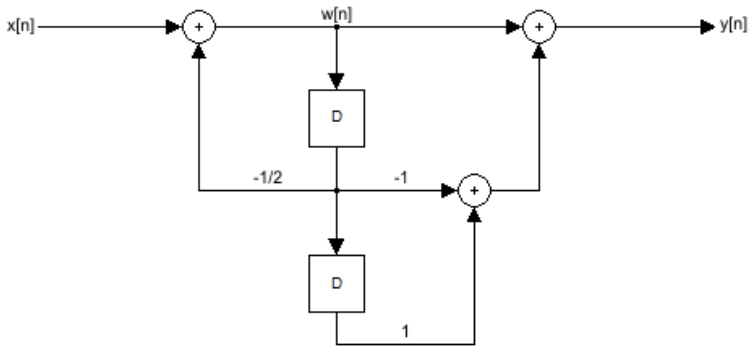
b) $2y[n] + 6y[n-2] = x[n] + x[n-2]$

c) $y[n] - 0{,}4y[n-1] = x[n] - x[n-2]$

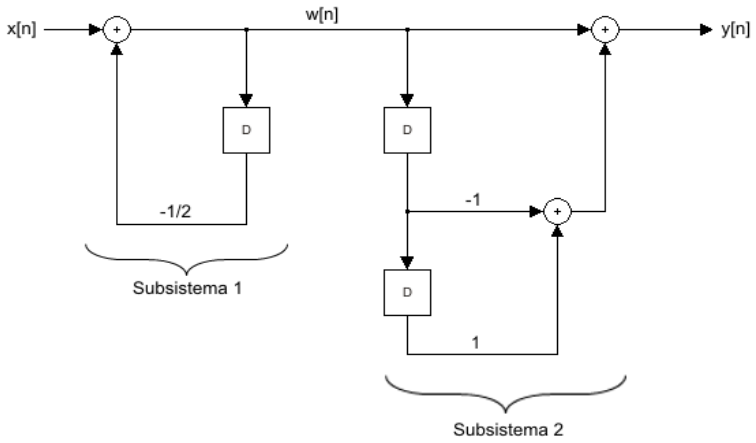
Trabajo Práctico Nº2
Sistemas lineales – con Octave aplicado

Ejemplo 2

Para el siguiente sistema LIT discreto calcule la respuesta al impulso dividiendo al sistema en dos subsistemas.



En el primer subsistema la salida w[n] en un momento presente *n* depende o recurre a valores previos de la salida w[n − k], por lo tanto se le llama ***recursivo***. La característica del sistema recursivo es que su respuesta al impulso suele tener una longitud infinita, por lo tanto, también se denomina de respuesta de impulso infinito o IIR.



En cambio, en el segundo subsistema la salida y[n] no depende de los valores previos de la salida, sino solo de las entradas ponderadas y desplazadas w[n − m], por lo cual el sistema se llama ***no recursivo***. Una característica que poseen estos sistemas es que la respuesta al impulso de los sistemas no recursivos es de longitud finita; denominada respuesta de impulso finito o FIR.

En Octave se copia el siguiente programa en un archivo, por ejemplo Ej2.m, y se ejecuta. Así obtendremos la respuesta al impulso de cada subsistema y la respuesta al impulso de todo el sistema.

```
%Vector de variable independiente discreta.
n=-20:1:20;
%Vector de ceros con 41 valores llamado delta
%Se almacena el valor 1 en la posición 21.
delta=zeros (1,41);
delta(21)=1;
%Subsistema h1
%Se colocan los coeficientes de la señal de salida en un vector A
A = [1 0.5];
%Se colocan los coeficientes de la señal de entrada en un vector B
B = [1];
%Calcula la señal de salida de la ecuación lineal en diferencias h1
%para la señal de entrada delta.
h1 = filter(B,A,delta);
%En este caso las condiciones iniciales son nulas.
%Caso contrario ver las variantes de dicho comando.
```

```
%Subsistema h2
%Se colocan los coeficientes de la señal de salida en un vector C.
C = [1];
%Se colocan los coeficientes de la señal de entrada en un vector D.
D = [1 -1 +1];
%Calcula la señal de salida de la ecuación lineal en diferencias h2
%para la señal de entrada delta.
h2=filter(D,C,delta);
%En este caso las condiciones iniciales son nulas.
%Caso contrario ver las variantes de dicho comando.
%Grafica la respuesta al impulso;
subplot (3,1,1)
stem(n,h1,'filled');
title('respuesta al impulso del sistema h1');
subplot (3,1,2)
stem(n,h2,'filled');
title('respuesta al impulso del sistema h2');
%Vector de variable independiente discreta nc para la respuesta al
%impulso del sistema total. Dicho vector debe tener como mínimo una
%longitud igual a la suma de las longitudes de las señales discretas
%menos uno.
nc=n(1)+n(1):n(41)+n(41);
%Convolución de las respuestas al impulso de los dos %subsistemas
ht=conv(h1,h2);
%Grafica de las respuestas al impulso del sistema total
subplot(3,1,3);
stem(nc,ht,'filled');
title('respuesta al impulso del sistema total');
```

Actividad 4

Para el sistema LIT anterior aplique la propiedad conmutativa de la convolución, intercambiando el orden de los subsistemas.

a) Obtenga la expresión de la ecuación en diferencias de cada subsistema.
b) Utilizando Octave/Matlab/Python grafique la respuesta al impulso y entregue las líneas de código.

Ejemplo 3

El siguiente programa realizar la convolución entre la respuesta al impulso del sistema y una señal entrante al mismo dada por:

$$g[n] = 0.5^n u[n] * 0{,}8^n u[n-2]$$

Guárdelo en un archivo denominado por ejemplo Ej3 y ejecútelo en el entorno Octave.

```
%Convolucion discreta ejemplo 3
clear
n=-9:1:10;
b1=0.5*ones(1, 20);
u1=[zeros(1, 9), ones(1, 11)];
f1=(b1.^n).*u1;
b2=0.8*ones(1, 20);
u2=[zeros(1, 11), ones(1, 9)];
f2=(b2.^n).*u2;
nc=-18:1:20;
%Convolucion de las funciones
ht=conv(f1,f2);
%Grafica de las respuestas al impulso del sistema total
stem(nc,ht,'filled');
title('convolucion Ej3');
```

Actividad 5

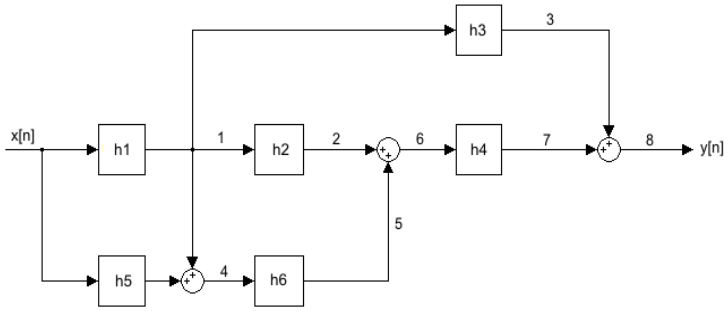
Grafique utilizando Octave/Matlab/Python las siguientes convoluciones y entregue las líneas de código.

a) $g[n] = 2rect_4[n] * (7/8)^n u[n]$
b) $g[n] = 0{,}2^n\, u[n] * (\delta[n-1] + 3\delta[n-2] + 2\delta[n-6])$

Trabajo Práctico Nº2
Sistemas lineales – con Octave aplicado

Actividad 6

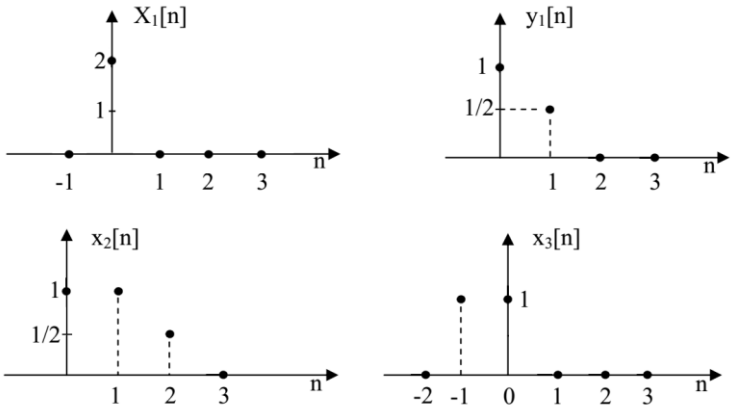
Aplicando las propiedades de interconexión de sistemas, determine la respuesta al impulso del diagrama en bloques siguiente.



Dónde: $h_1[n]=3\delta[n]$; $h_2[n]=\delta[n]$; $h_3[n]=u[n]$; $h_4[n]=2^n u[n]$; $h_5[n]=0$; $h_6[n]=\delta[n]$

Actividad 7

Sea un sistema LIT cuya respuesta a la señal $x_1[n]$ es $y_1[n]$. Halle las respuestas a las excitaciones $x_2[n]$ y $x_3[n]$.



Ejemplo 4

Dado el siguiente sistema LIT continuo, obtener la respuesta al escalón y mediante la derivada de esta encontrar la respuesta al impulso.

$$y'(t) + 5y(t) = x(t)$$

El siguiente archivo de Octave resuelve simbólicamente y grafica la ecuación diferencial para un escalón a la entrada. Luego se deriva dicha respuesta para obtener la respuesta al impulso de dicho sistema comparando ambas gráficas. Para poder trabajar simbólicamente, primero se debe cargar el módulo simbólico mediante el comando: **`pkg load symbolic`**.

```
%Calculo y grafica de la respuesta al escalón de un
%sistema LIT continuo.
%Derivando dicha respuesta obtenemos la respuesta
%al impulso y su gráfica.
clear;
% definición de una función simbólica
syms y(x);
%escritura del sistema LIT
eq = diff(y,x,1)+5*y == 1;
cond = [y(0)==0];
%Resolución simbólica del sistema al escalón
S = dsolve(eq,cond);
```

```
%Derivada de la función anterior
dS=diff(S)
%Conversión a función anonima
h=function_handle(S);
dh=function_handle(dS);
%Definición de variable independiente numérica
c=0:.1:5;
%Evaluación y gráfica de ambas funciones
ph=h(c);
plot(c,ph);
hold on;
pdh=dh(c);
plot(c,pdh);
```

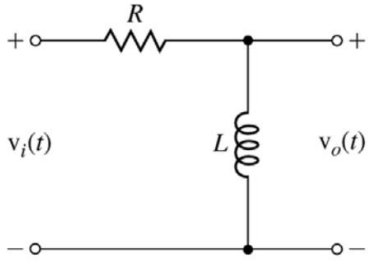
Actividad 8

Determinar la respuesta de los siguientes sistemas continuos a la delta de Dirac utilizando Octave/Matlab/Python. Aplique lo desarrollado en el ejemplo 4 para obtener la solución y realizar la gráfica de la respuesta de los puntos a), b) y c). Entregue las líneas de código.

a) $y''(t) + 2y'(t) + 4y(t) = x(t)$
b) $y''(t) + 6y'(t) + 5y(t) = x(t)$
c) $y''(t) - 2y'(t) + 3y(t) = x(t)$

Actividad 9

El circuito de la figura inferior muestra un filtro RL con los valores R=100Ω y L=10mH. El voltaje de la señal de entrada es $V_i(t)$ y el voltaje de la señal de salida es $V_o(t)$.



a) Encuentre la ecuación diferencial del circuito.
b) Determine y grafique la respuesta al escalón unitario utilizando Octave/Matlab/Python.
c) Determine y grafique la respuesta al impulso utilizando Octave/Matlab/Python.

Ejemplo 5

Utilizando Octave el siguiente programa grafica las funciones intervinientes en la convolución, para luego calcular la convolución y mostrar la gráfica.

$$g(t) = rect(t) * rect(t)$$

Para realizar la convolución continua, se abre el Octave y en el editor se copia el siguiente programa:

```
%Archivo del Ej5;
clear;
%Intervalo de tiempo;
ts=0.01;
%Vector de variable independiente discreta;
n=-100:99;
%Construcción de la función rectángulo con la función escalón
%heaviside;
rec=heaviside((n*ts)+.5)-heaviside((n*ts)-.5);
%Grafico de las señales a convolucionar
subplot(3,1,1);
```

Trabajo Práctico Nº2
Sistemas lineales – con Octave aplicado

```
plot(ts*n,rec);
title('senal 1 a convolucionar');
subplot(3,1,2);
plot(ts*n,rec);
title('senal 2 a convolucionar');
%Vector de variable independiente discreta de la convolución;
%Suma de la longitud de las dos señales a convolucionar;
nc=n(1)+n(1):n(end)+n(end);
%convolución y grafica de dos señales rectángulo;
y=ts*conv(rec,rec);
subplot(3,1,3);
plot(ts*nc,y);
title('convolución de dos rectángulos centrados');
```

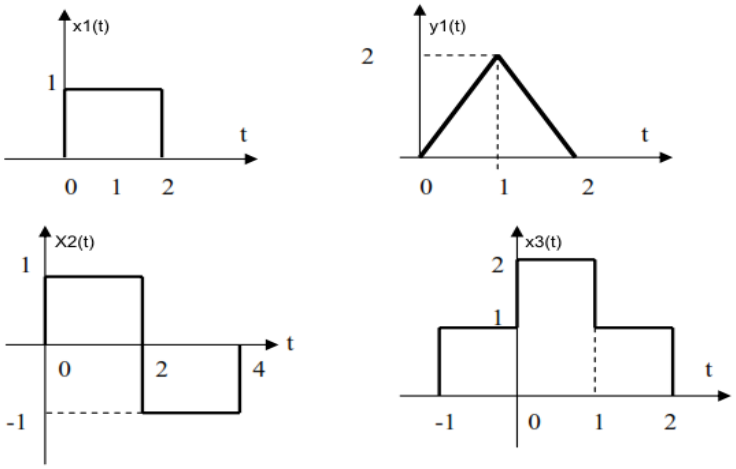
Actividad 10

Graficar las funciones, calcular la convolución y su gráfica utilizando Octave/Matlab/Python para cada inciso. Entregar las líneas de código.

a) $g(t) = rect(t) * rect(t/2)$
b) $g(t) = rect(t-1) * rect(t/2)$
c) $g(t) = (rampa(t) \cdot rect(t-1/2)) * rect(t)$
d) $g(t) = (rampa(t) \cdot rect(t-1/2)) * \exp(-t) \cdot u(t)$

Actividad 11

Dado un sistema LIT cuya respuesta a la señal $x_1(t)$ es la señal de salida $y_1(t)$. Determinar y dibujar la respuesta de este sistema a las señales $x_2(t)$ y $x_3(t)$.



Trabajo Práctico Nº2
Sistemas lineales – con Octave aplicado
ANEXO

Resolución numérica de Ecuaciones diferenciales ordinarias de primer orden (EDO)

Una ecuación diferencial ordinaria de primer orden es una ecuación que se puede escribir en esta forma:

$$y' = \frac{\partial y}{\partial x} = g(x, y)$$

Donde *x* es la variable independiente e *y* es una función de *x*. Una solución a una EDO de primer orden es una función:

$$y = f(x) \qquad tal\ que \qquad f'(x) = g(x, y)$$

Calcular la solución de una ecuación diferencial implica integrar para obtener *y* a partir de *y'*; por tanto, las técnicas para resolver ecuaciones diferenciales también se conocen como técnicas para integrar ecuaciones diferenciales. La solución de una ecuación diferencial generalmente es una familia de funciones, donde por lo regular se necesita una condición inicial o condición de frontera para especificar una solución inicial única.

Muchas ecuaciones diferenciales tienen soluciones analíticas complicadas o simplemente no las tienen, en estos casos se requiere una técnica numérica para resolver la ecuación diferencial. Las técnicas numéricas más comunes para resolver ecuaciones diferenciales ordinarias son el método de Euler y los métodos de Runge-Kutta. Tanto el método de Euler como los métodos de Runge-Kutta aproximan una función usando su expansión de serie de Taylor.

Recuerde que una serie de Taylor es una expansión que puede servir para aproximar una función cuyas derivadas existen en un intervalo que contiene los valores *a* y *b*. La expansión de serie de Taylor de f(b) es:

$$f(b) = f(a) + (b-a)f'(a) + \frac{(b-a)^2}{2!}f''(a) + \cdots + \frac{(b-a)^n}{n!}f^n(a) + \cdots$$

Una aproximación de serie de Taylor de primer orden usa los términos en los que interviene la función y su primera derivada. Mientras que una aproximación de segundo orden usa los términos en los que interviene la función, su primera derivada y su segunda derivada. Cuantos más términos de la serie de Taylor se usen para aproximar una función, más exacta será la aproximación.

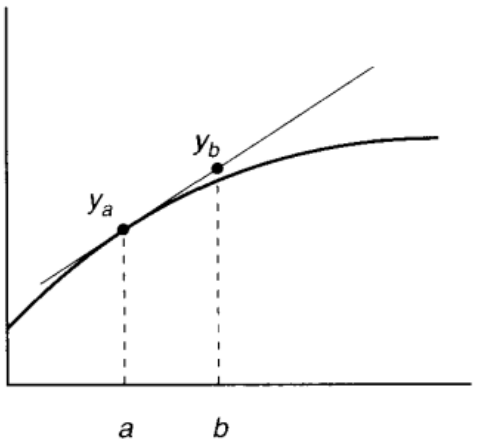


Figura 1

Métodos de Runge-Kutta

Los métodos más utilizados para integrar una ecuación diferencial de primer orden son los métodos de Runge-Kutta. Estos métodos se basan en aproximar una función usando su expansión de serie de Taylor; así, un método de Runge-Kutta de primer orden usa una expansión de serie de Taylor de primer orden, un método de RungeKutta de segundo orden usa una expansión de serie de Taylor de segundo orden, etc. (El método de Euler es equivalente a un método de Runge-Kutta de primer orden.) Las funciones de Octave/Matlab que veremos usan aproximaciones de orden 2, 3, 4 y 5 para estimar valores de una función desconocida f(x) usando una ecuación diferencial.

Trabajo Práctico Nº2
Sistemas lineales – con Octave aplicado

Si tomamos la serie de Taylor para evaluar f(b) simplificando la notación de la derivada y considerando que el término (b - a) representa un tamaño de paso h, podemos reescribir la serie de Taylor en esta forma:

$$y_b = y_a + hy_a{}' + \frac{h^2}{2!}y_a{}'' + \cdots + \frac{h^n}{n!}y_a{}^n + \cdots$$

Una ecuación de integración Runge-Kutta de primer orden es la siguiente:

$$y_b = y_a + hy_a{}'$$

Esta ecuación estima el valor de la función y_b, usando una línea recta que es tangente a la función en y_a, como se muestra en la figura1. Para calcular el valor de y_b (que se supone está en la línea tangente) usamos un tamaño de paso h igual a (b - a) y un punto de partida y_a; en donde se usa la ecuación diferencial para calcular el valor de y_a'. Una vez determinado el valor de y_b, podemos estimar el siguiente valor de la función y_c usando lo siguiente:

$$y_c = y_b + hy_b{}'$$

Esta ecuación usa la línea tangente en y_b, para estimar y_c, como se muestra en la figura 2. Puesto que se necesita un valor inicial o de frontera para iniciar el proceso de estimación de otros puntos de la función f(x), los métodos de Runge-Kutta (y el de Euler) también se denominan **soluciones de valor inicial** o **soluciones de valor de frontera.**

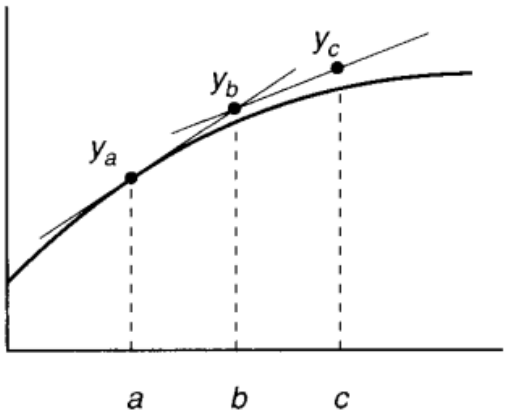


Figura 2

La ecuación de integración Runge-Kutta de primer orden es sencilla de aplicar, pero dado que aproxima la función con una serie de segmentos de recta cortos, podría no ser muy exacta si el tamaño de paso es grande o si la pendiente de la función cambia con rapidez. Es por esto que con frecuencia se usan ecuaciones de integración Runge-Kutta de orden superior para aproximar la función desconocida. Estas técnicas de orden superior promedian varias aproximaciones de tangente a la función, obteniendo así resultados más exactos. Por ejemplo, una ecuación de integración Runge-Kutta de cuarto orden usa términos de la expansión de serie de Taylor que incluyen la primera, segunda, tercera y cuarta derivadas, y calcula la estimación de la función usando cuatro estimaciones de tangente.

Funciones ODE

Octave y Matlab contienen dos funciones comunes para calcular soluciones numéricas de ecuaciones diferenciales ordinarias: **ode23** y **ode45.** A continuación describimos los argumentos y luego presentamos algunos ejemplos.

- **[x , y]=ode23('nombre_func', a, b, inicial)**
  Devuelve un conjunto de coordenadas x e *y* que representan la función y=f(x) y se calculan usando métodos de Runge-Kutta de segundo y tercer orden.

El **'nombre_func'** define una función f que devuelve valores de la ecuación diferencial y'= g(x,y) cuando recibe valores de x e y. Los valores a y b especifican los extremos del intervalo en el cual deseamos evaluar la función y=f(x). El valor de **initial** especifica el valor de la función en el extremo izquierdo del intervalo **[a,b].**

- **[x , y]=ode45('nombre_func', a, b, inicial)**
  Devuelve un conjunto de coordenadas **x** e **y** que representan la función y =f(x) y se calculan usando métodos de Runge-Kutta de cuarto y quinto orden.
  El '**nombre_func'** define una función f que devuelve valores de la ecuación diferencial y' = g(x,y) cuando recibe valores de x e y. Los valores a y b especifican los extremos del intervalo en el cual deseamos evaluar la función y=f(x). El valor de **initial** especifica el valor de la función en el extremo izquierdo del intervalo **[a,b]**.

Las funciones **ode23** y **ode45** también pueden llevar dos parámetros adicionales. Se puede usar un quinto parámetro para especificar una tolerancia relacionada con el tamaño de paso; las tolerancias por omisión son 0.001 para **ode23** y 0.000001 para **ode45.**

Ejemplo de ecuación diferencial de primer orden:
Tomemos la ecuación diferencial de primer orden:

$$y' = 3 \cdot y + e^{2x}$$

El archivo de la función g1.m sería:

```
function dy = g1(x,y)
%Esta función evalúa una EDO de primer grado.
dy = 3*y+exp(2*x);
endfunction
```

Las siguientes instrucciones resuelven g1(x,y) dentro del intervalo [0,3], suponiendo que la condición inicial y =f(0)es igual a 3. Así el archivo del programa prog1.m queda:

```
%Determinar la solución de la EDO 1.
[x,y]= ode23('g1',[0 3],3);
plot(x,y);
title('So1ución de la ecuación ');
xlabel('x');
ylabel('y=f(x)');
grid on;
```

Ecuaciones diferenciales de orden superior
Una ecuación diferencial de orden superior puede escribirse como un sistema de ecuaciones diferenciales de primer orden acopladas usando un cambio de variables. Por ejemplo, considere la siguiente ecuación diferencial de orden n:

$$y^n = g(x, y, y', y'', \cdots, y^{n-1})$$

Entonces primero, definimos *n* funciones nuevas desconocidas con estas ecuaciones:

$$u_1(x) = y^{n-1}$$
$$u_2(x) = y^{n-2}$$
$$\cdots\cdots$$
$$u_{n-2}(x) = y''$$
$$u_{n-1}(x) = y'$$
$$u_n(x) = y$$

Entonces, el siguiente sistema de *n* ecuaciones de primer orden equivale a la ecuación diferencial de orden *n* anterior:

$$u'_1 = y^n = g(x, u_n, u_{n-1}, u_{n-2}, \cdots u_1)$$
$$u'_2 = u_1$$
$$\cdots\cdots$$
$$u'_{n-2} = u_{n-3}$$
$$u'_{n-1} = u_{n-2}$$

Ejemplo de ecuación diferencial de orden superior

Para demostrar este proceso utilizamos la ecuación de Van Der Pol, dicho oscilador no conservativo con amortiguamiento no lineal se desarrolla en el tiempo de acuerdo con la siguiente ecuación diferencial de 2º orden:

$$y'' = g(x, y, y') = y'(1 - y^2) - y$$

Primero definimos dos nuevas funciones:

$$u_1(x) = y'$$
$$u_2(x) = y$$

Luego obtenemos este sistema de ecuaciones diferenciales de primer orden acopladas:

$$y'' = g(x, u_2, u_1) = u'_1 = u_1(1 - {u_2}^2) - u_2$$
$$u'_2 = u_1$$

Podemos resolver un sistema de ecuaciones diferenciales de primer orden usando las funciones ODE de Octave/Matlab. Sin embargo, la función que se use para evaluar la ecuación diferencial deberá calcular los valores de las ecuaciones diferenciales de primer orden acopladas en un vector. La condición inicial también deberá ser un vector que contenga una condición inicial para:

$$y^{n-1}, y^{n-2}, \cdots y', y$$

Las funciones ODE devuelven soluciones para cada una de las ecuaciones diferenciales de primer orden.

Para resolver el conjunto de dos ecuaciones acopladas que planteamos en el ejemplo, primero definimos una función que calcule valores de las ecuaciones diferenciales de primer orden:

```
function du= eqns2(x,u)
%Esta función calcula valores para dos ecuaciones acopladas
du(1)=u(1)*(1-u(2)^2)-u(2);
du(2)=u(1);
endfunction
```

Entonces, para resolver el sistema de ecuaciones diferenciales de primer orden en el intervalo [0,20] usando las condiciones iniciales y'(0)=0.0 y y(0)=0.25, utilizamos estas instrucciones:

```
inicial=[0 0.25];
[x,y] = ode23 ('eqns2',[0,20],inicial);
subplot(2,1,1);
plot(x,y(:,1));
title('Primera derivada de y');
grid on;
subplot(2,1,2);
plot(x,y(:,2));
title ('función y');
xlabel('x');
grid on;
```