Manejo de Software Específico MATLAB/Octave

A fin de repasar el uso del lenguaje de programación principal de esta cátedra, MATLAB/Octave, se adjunta un pequeño resumen de los comandos más importantes que usaremos en el cursado.

Vectores y Matrices

Para Matlab/Octave todos los elementos son arreglos matriciales. En particular, los escalares son matrices de dimensión 1×1, los vectores fila son matrices de dimensión $1 \times n$, y los vectores columna son matrices de dimensión $n \times 1$. Las matrices se declaran entre corchetes ([ ]), las columnas se separan con espacios o con coma (,), y para indicar el comienzo de una nueva fila se utiliza el punto y coma (;).

Hay varias funciones que permiten construir matrices con valores particulares, como, por ejemplo:

Función	Operación
eye(n)	Matriz identidad de dimensión n
zeros(n)	Matriz con elementos nulos cuadrada de dimensión n
zeros(n,m)	Matriz con elementos nulos de dimensión n x m
ones(n,m)	Matriz de unos cuadrada de dimensión n x m
diag(v,m)	Matriz cuadrada con los elementos del vector v en la matriz m

Para construir vectores fila que vayan de un valor inicial a un valor final con un paso constante, existen dos formas simples de construir estos vectores:

- especificando entre paréntesis o corchetes el valor inicial, el paso y el valor final, donde si el paso no es especificado por defecto toma el valor 1: **x=[valor inicial:paso:valor final];**
- a partir de la función linspace: **x=linspace(valor inicial,valor final,cantidad de puntos)**.

Operadores Lógicos

Los operadores lógicos operan entre dos variables lógicas y su resultado es nuevamente una variable lógica:

Símbolo	Operador
| ó ||	OR: el resultado será verdadero si alguna de las dos variable es verdadera
& ó &&	AND: el resultado será verdadero si ambas variables lo son
xor()	OR exclusivo: el resultado será verdadero si solo una de las variables lo es
! ó ~	NOT: opera sobre una sola variable lógica, da como resultado el valor contrario

Funciones

Matlab/Octave tiene definido funciones matemáticas tales como: **cos(x), sin(x), tan(x), abs(x), exp(x), sqrt(x), log(x), log2(x), log10(x), rem(x), sign(x)**, etc. Si quiere una explicación sobre la operación que realiza o sobre cómo utilizarla puede escribir el comando **help** seguido del nombre de la función en la ventana de comandos. Algo importante a tener en cuenta es que **las funciones trigonométricas operan sobre radianes y no sobre grados**.

Cuando necesitemos definir funciones propias (o funciones de usuario), la misma debe ser especificada como sigue:

```
function [argumentos de salida] = nombre (argumentos de entrada)
    declaraciones de variables locales
    sentencias
end
```

Donde **argumentos de entrada** son los valores que serán pasados desde el programa principal a la función para que pueda realizar las operaciones y devolver el resultado de la evaluación de la función en la variable nombre, así como otros resultados que puedan generarse y serán devueltos en los **argumentos de salida**. Algo muy importante a tener en cuenta es que el nombre de la función debe coincidir con el nombre del **archivo.m** en el que es guardada. Las funciones son llamadas e invocadas en el programa principal mediante:

```
[argumentos de salida] =nombre (argumentos de entrada)
```

Gráficas

Matlab/Octave cuenta con una gran variedad de funciones destinadas a realizar gráficos. La función **plot()** es la que más utilizaremos a lo largo de este curso. Para utilizarla en general necesitamos especificar entre paréntesis dos variables separadas por coma, que representen las coordenadas cartesianas x e y.

Por defecto, para graficar, el comando **plot()** utiliza líneas. Pero muchas veces es conveniente graficar utilizando símbolos ('*', 'o', '+', etc.) y para ello se especifica el mismo como un tercer argumento entre comillas. Otro parámetro que puede ser especificado es el color ('k', 'r', 'g', 'b', etc.). Para mayor detalle escriba en la ventana de comandos **help plot**.

También se recomienda revisar los siguientes comandos:

hold on, hold off, title(), xlabel(), ylabel(), grid on, legend(), subplot(), figure().

Auto-valores y Auto-vectores

La obtención los auto valores (o eigenvalores, valores propios) y autovectores (eigenvectores, vectores propios) de las matrices características de los sistemas de múltiples grados de libertad, es una de las tareas más importantes en el estudio de la Mecánica Vibratoria, por ende, es necesario conocer los comandos básicos para obtenerlas en MATLAB/Octave.

En este caso, se recomienda repasar la función:

[V, D] = eig(A); matriz diagonal D son eigenvalores y matriz V cada columna es eigenvector

https://la.mathworks.com/help/matlab/ref/eig.html?s_tid=doc_ta

MECÁNICA VIBRATORIA – Ingeniería Mecatrónica	Alumno:
Trabajo Práctico N° 0 - Repaso	Legajo:
Manejo de Software Específico MATLAB/Octave	Año 2024

Solución de EDOs

Otra función importante es aquella que permite resolver EDOs con métodos numéricos de grado alto (Runge-Kutta orden 4 o 5), que nos evita la necesidad de programarlos nosotros.

Se recomienda revisar la siguiente función, que nos permite además resolver un Sistema de Ecuaciones Diferenciales Ordinarias (muy útil para sistemas de múltiples grados de libertad):

[t, Y] = ode45(odefun , tspan , y0);

Dónde **tspan=[t0 tf]**, y la función integra el sistema de ecuaciones diferenciales **y' = f(t,y)** desde **t0** a **tf** con condiciones iniciales **y0**. Cada fila de la matriz de solución (**Y**) se corresponde con un valor devuelto en el vector columna **t**.

https://la.mathworks.com/help/matlab/ref/ode45.html?s_tid=doc_ta

---

Luego del repaso de conceptos, se plantean los siguientes ejercicios.

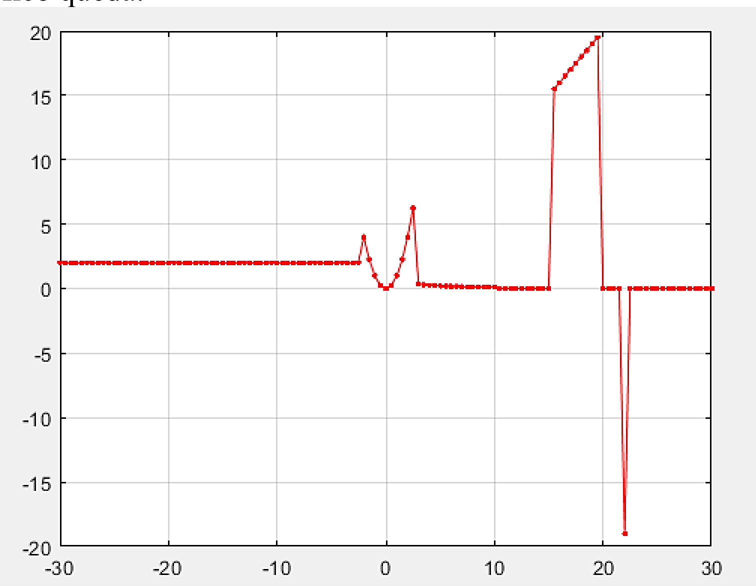
Problema N°1

Defina la siguiente función a trazos haciendo uso del comando if y de los operadores lógicos y relacionales. Implementarla como una función de usuario:

$$f(x) = \begin{cases} 2 & \text{si} \quad x < -2 \\ x^2 & \text{si} \quad -2 \leq x < 3 \\ \frac{1}{x} & \text{si} \quad 3 \leq x \leq 10 \\ x & \text{si} \quad 15 < x < 20 \\ 3 - x & \text{si} \quad x = 22 \\ 0 & \text{en} \quad \text{cc} \end{cases}$$

Evalúe f(x) en el vector x= [-30:0.5:30] y grafique el resultado haciendo uso del comando ***plot()***.

Rta.: El gráfico queda:



Problema N°2

Defina los vectores t, x e y de la siguiente manera:

```
t=[0:2*pi/50:2*pi];
x=2*cos(t);
y=5*sin(t);
```

a) Grafique x e y en función de t. Los comandos ***plot(), figure()*** y ***subplot()*** pueden serle útiles.

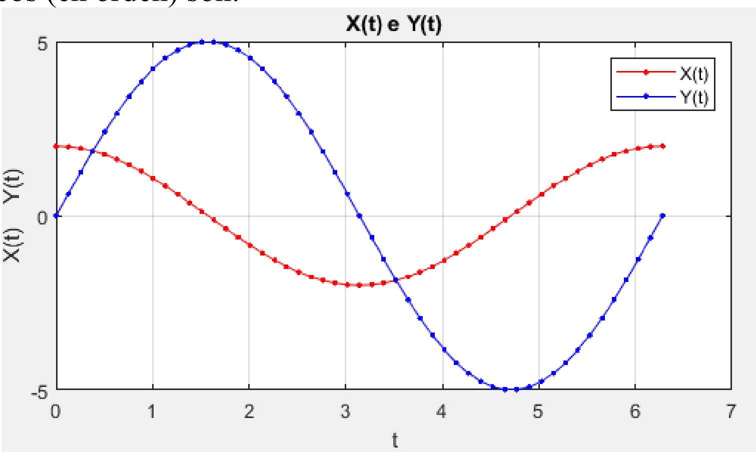
b) La curva r(t) = (x(t), y(t)) representa una elipse descripta en forma paramétrica. Para graficarla puede utilizar ***plot(x,y)***.

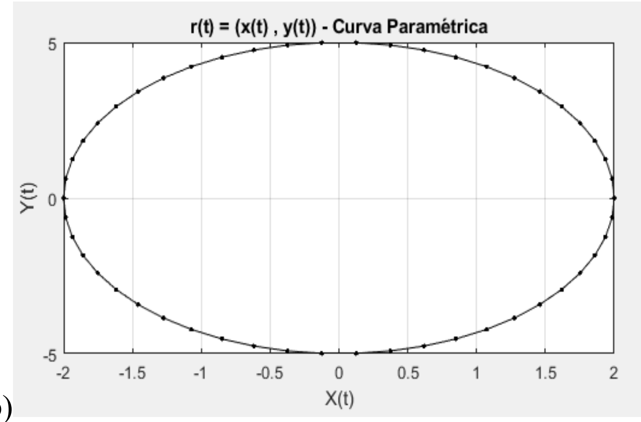
c) ¿Cual parece ser el eje mayor de la elipse? ¿Cuál es realmente el eje mayor de la elipse? ¿A qué se debe esto? La sentencia ***axis equal*** puede ser de utilidad.

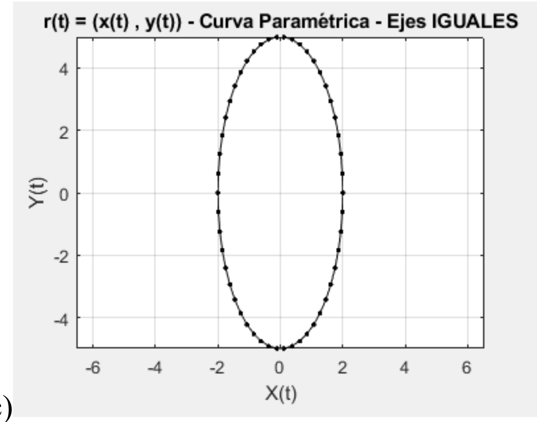
d) Modifique el paso de la variable ***t*** y/o el límite superior de la misma, y observe qué sucede con la gráfica. Luego, realice la siguiente “animación” con las siguientes líneas:

```
figure(…)
title('Animación')
axis([-5 5 -5 5]), hold on, grid on,
xlabel('X(t)'), ylabel('Y(t)')
for i=1:length(t)-1
   plot(x(i:i+1),y(i:i+1),'LineWidth',2);
   pause(1/10);
end
hold off
```

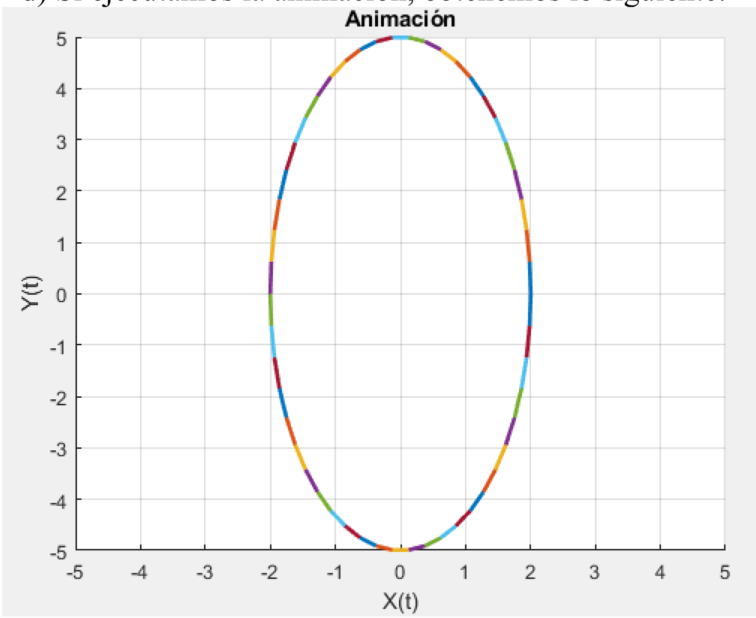
Rta.: Los gráficos (en orden) son:

a) 

b) 

c) 

d) Si ejecutamos la animación, obtenemos lo siguiente:



Problema N°3

Ante el Sistema de EDOs $\frac{dX}{dt} = A * X$; se puede encontrar la solución $x(t) = e^{t*A} * X(0)$. Dada la siguiente matriz A, calcule los auto-valores de la misma e interprete el significado de sus valores. Recordar la Fórmula de Euler.

A = [0 -6 -1
6 2 -16
-5 20 -10]

Rta.: Tenemos los autovectores en V(columnas) y autovalores en L (no ordenados):

V = [-0.8326 + 0.0000i 0.2003 - 0.1394i 0.2003 + 0.1394i
-0.3553 + 0.0000i -0.2110 - 0.6447i -0.2110 + 0.6447i
-0.4248 + 0.0000i -0.6930 + 0.0000i -0.6930 + 0.0000i]

L = [-3.0710 + 0.0000i
-2.4645 +17.6008i
-2.4645 -17.6008i]

La parte real de los valores propios es negativa, y por eso $e^{\lambda t}$ tiende a cero a medida que aumenta ***t***. La parte imaginaria no nula de los valores propios, ±***ω***, aporta el componente oscilatorio, ***sin(ωt)***, a la solución de la ecuación diferencial.

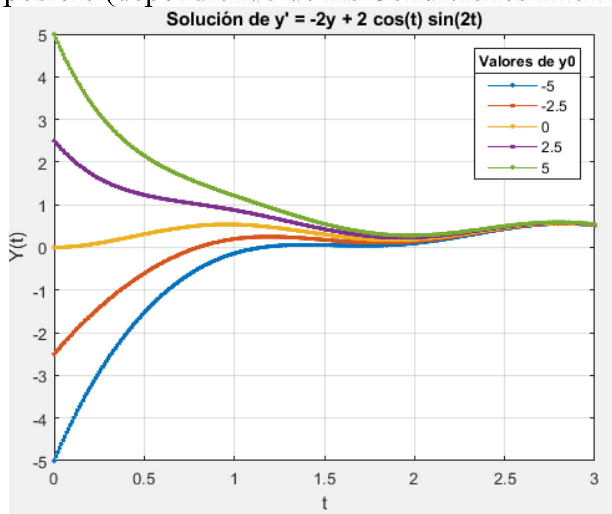
Problema N°4

Para sistemas de EDO con una ecuación, puede especificar y0 como un vector que contenga varias condiciones iniciales. Esto crea un sistema de ecuaciones y ***ode45*** resuelve el sistema para generar resultados para cada valor inicial.

Cree una función ***anónima*** para representar la ecuación $\mathbf{f(t,y) = -2 * y + 2 * cos(t) * sin(2t)}$. La misma debe aceptar dos variables (***t*** e ***y***). También cree un vector de condiciones iniciales (distintas) ***y0***, de longitud 5, en el rango [-5, 5]. Resuelva la EDO para un **t=[0, 3]** con ***ode45*** y grafique.

Función ***anónima***: https://la.mathworks.com/help/matlab/matlab_prog/anonymous-functions.html

Rta.: Un gráfico posible (dependiendo de las Condiciones Iniciales) es:



Problema N°5

Como vimos, para hacer uso de la función "ode45" o cualquiera de las que resuelven EDOs, es necesario ingresar ecuaciones diferenciales de primer orden. Si deseamos resolver ecuaciones de orden superior (por ejemplo, de segundo orden, muy comunes en la materia), debemos realizar una Reducción de Orden.

Resuelva la siguiente ecuación diferencial con la función ode45, realizando la reducción de orden necesaria (revisar el apunte de cátedra "ec_diferencial_2do_orden_a_primer_orden.pdf"). Luego grafique su resultado, y el de su derivada primera.

$$m * \ddot{x}(t) + c * \dot{x}(t) + k * x(t) = F(t)$$

Con: m = 10 kg; k = 500 N/m; c = 15 N.s/m; F(t) = 100*sen(3*t); $x(0) = 1m$; $\dot{x}(0) = 0\frac{m}{s}$

Rta.: De acuerdo a la reducción de orden que plantea el apunte de cátedra, realizamos un cambio de variable y obtenemos:

$$x_1(t) = x(t) \quad ; \quad x_2(t) = \dot{x}(t) = \frac{dx(t)}{dt} \equiv \dot{x}_1(t) \quad ; \quad \dot{x}_2(t) = \frac{d\dot{x}(t)}{dt} = \ddot{x}(t)$$

Reemplazando y despejando:

$$\dot{x_1}(t) = x_2(t) \qquad \dot{x_2}(t) = -\frac{c}{m} * x_2(t) - \frac{k}{m} * x_1(t) + \frac{1}{m} * F(t)$$

Las ecuaciones anteriores forman un Sistema de Ecuaciones Diferenciales de Primer Orden, susceptibles de ser resueltas por "ode45":

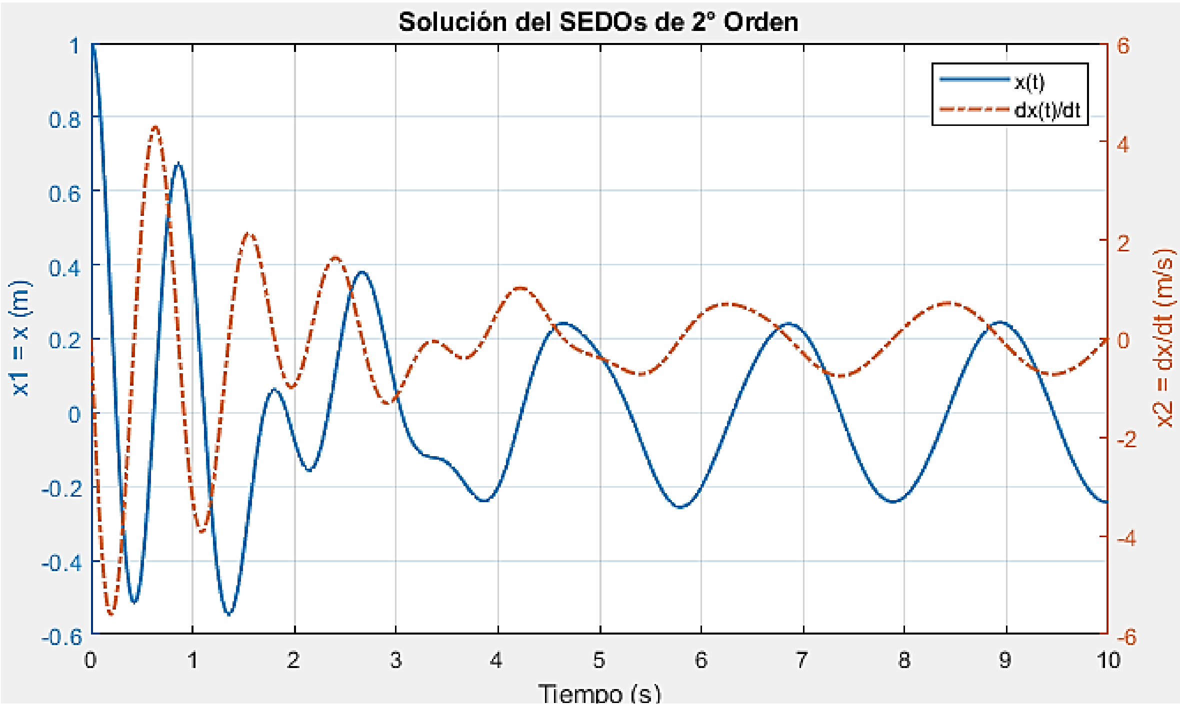
$$\dot{X}(t) = \begin{bmatrix} \dot{x_1}(t) \\ \dot{x_2}(t) \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -\frac{k}{m} & -\frac{c}{m} \end{bmatrix} * \begin{bmatrix} x_1(t) \\ x_2(t) \end{bmatrix} + \begin{bmatrix} 0 \\ \frac{1}{m} * F(t) \end{bmatrix} = \begin{bmatrix} x_2(t) \\ -\frac{c}{m} * x_2(t) - \frac{k}{m} * x_1(t) + \frac{1}{m} * F(t) \end{bmatrix}$$

Se ingresan al código con el siguiente comando (por ejemplo):

```
f =@(t,X) [X(2);    -c/m*X(2) - k/m*X(1) + 1/m*5*sin(3*t)];
```

El resultado de "ode45" será una matriz de tantas filas como instantes de tiempo definamos, y 2 columnas. La primera corresponde a $x_1(t)$ y la segunda a $x_2(t)$.

A continuación, el gráfico del resultado:



De esta forma, podemos resolver EDOs de cualquier orden, solo debemos reducirlo a un SEDOs de primer orden, que tendrá tantas ecuaciones como sea el orden de la ecuación original. Por supuesto, tendremos también un vector de coordenadas iniciales, en lugar de un valor escalar.

Para saber más sobre la posibilidad de generar "dos ejes y", como en el gráfico anterior, revisar "https://www.mathworks.com/help/matlab/creating_plots/plotting-with-two-y-axes.html".