LABORATORIO DE SISTEMAS DE CONTROL AUTOMÁTICO

PRÁCTICA N°1

1. TEMA

INTRODUCCIÓN A MATLAB Y SIMULINK

2. OBJETIVOS

2.1. Familiarizar al estudiante con el uso del programa desarrollado por MathWorks y con el toolbox denominado SIMULINK para la simulación de sistemas.
2.2. Incentivar en el estudiante el uso del entorno de programación de MATLAB como herramienta de análisis, diseño, y simulación de sistemas de control.
2.3. Familiarizar al estudiante con el uso de los recursos en línea proporcionados por MathWorks.

3. INFORMACIÓN

MATLAB es un lenguaje de programación codificado en C, basado en matrices para cálculos científicos y de ingeniería. El nombre MATLAB es una abreviatura para MATrix LABoratory.

MATLAB trabaja esencialmente con matrices numéricas rectangulares (que pueden tener elementos complejos), lo cual implica el uso de vectores fila o columna. Por esta razón este paquete tiene una proyección hacia el control moderno (descrito a variables de estado) y es útil para ilustrar las relaciones existentes entre las técnicas clásicas y modernas de análisis mediante un conjunto de rutinas de propósito general que permiten modelar, analizar y simular cualquier tipo de sistema dinámico. MATLAB contiene librerías (Toolboxes) de propósito específico, así por ejemplo: Sistemas de Control, Procesamiento de Señales, Optimización, Identificación de Sistemas, Lógica Fuzzy, Redes Neuronales, etc.

MATLAB es un sistema abierto ya que el usuario puede editar sus propias funciones específicas, las mismas que son guardadas a manera de macros o programas denominados *"archivos.m"* porque son almacenados en las bibliotecas de MATLAB mediante archivos ASCII con la extensión ".m". Además, tiene un entorno de desarrollo gráfico denominado App Designer disponible desde la versión 2019A o GUI en versiones anteriores.

3.1 Comandos generales

MATLAB posee un conjunto sumamente extenso de funciones agrupadas por áreas. El comando *help* permite disponer de más información acerca de estas funciones. Cuando se desea ayuda en línea de un comando en particular se escribe: *help comando*.

Algunos de los comandos generales de MATLAB se indican a continuación:

help	ayuda
demo	demostraciones
who	muestra variables en memoria
what	lista archivos específicos de MATLAB
clear	limpia variables y funciones
exit, quit	salida de MATLAB

3.2. Manipulación de matrices

Existen diferentes formas de introducir una matriz en MATLAB. Se tiene el siguiente ejemplo:

A = [1 2 3; 4 5 6; 7 8 9] Se asigna a la variable A una matriz 3x3

Los elementos de cada columna de una matriz pueden separarse tanto por comas como por espacios en blanco. Cada fila se separa mediante punto y coma. Las operaciones más importantes que se pueden realizar con matrices son:

- (+) adición
- (-) sustracción
- (*) multiplicación
- (^) potenciación
- (‘) transpuesta
- (\) división izquierda A\B es equivalente a inv(A)*B
- (/) división derecha A/B es equivalente a A*inv(B).

Estas operaciones para matrices se aplican también a escalares (matrices 1x1). Si los tamaños de las matrices son incompatibles para la operación matricial se obtiene un mensaje de error.

Se puede además efectuar operaciones con arreglos (elemento por elemento). Para ello, las operaciones *, ^, \, y /, deben ser precedidas por un punto. Por ejemplo, tanto [1 2 3].* [1 2 3] como [1 2 3].^2 darán el mismo resultado: [1 4 9].

Para la construcción y manipulación de matrices existen diferentes funciones que están disponibles en MATLAB. Entre ellas:

eye	matriz identidad	*zeros*	matriz de ceros
ones	matriz de unos.	*inv*	inversa
eig	valores propios	*poly*	polinomio característico
expm	matriz exponencial	*rank*	rango
det	determinante	*size*	tamaño

min	mínimo elemento	*max*	máximo elemento
poly	definición de polinomios a través de sus raíces		
rand	matriz generada aleatoriamente		
tril	parte triangular inferior de una matriz		
roots	raíces del polinomio característico trace traza		

Para generar vectores y submatrices se usa la “notación de dos puntos”. Su uso adecuado hace que las instrucciones sean más simples y legibles. Por ejemplo, observe el resultado de las siguientes expresiones:

1 : 3	genera un vector fila [1 2 3]
1 : 0.5 : 3	genera un vector de 1 a 3 con pasos de 0.5, es decir [1 1.5 2 2.5 3]
A(:,2)	es la segunda columna de A
A(1:2,:)	son las dos primeras filas de A
A(2,3)	elemento de la fila 2 y columna 3 de la matriz A.

3.3. Manejo de Gráficos

MATLAB puede generar gráficos planos y gráficos de malla de superficies tridimensionales. Para el manejo de gráficos planos, la instrucción *plot* crea gráficos en el plano XY; si x e y son vectores de la misma longitud, realiza un gráfico plano de los elementos de x versus los elementos de y. Por ejemplo, se puede dibujar la gráfica de la función $y = \left(sen(x)\right)^2$, sobre el intervalo [-4,4] con las siguientes instrucciones:

```
>> t = -4 : 0.01 : 4;
>> y = sin(t).^2;
>> plot(t,y)
```

De lo anterior se tiene que t es un vector que inicia en - 4 y termina en 4 en pasos de 0.01.

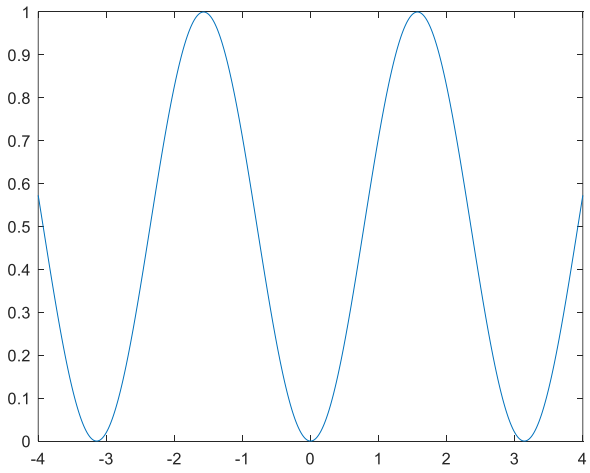

Figura 1. Señal sinusoidal

Diferentes funciones pueden dibujarse sobre una misma figura; para ello basta definir cada función con nombre diferente y ejecutar la instrucción:

plot(t, y1, t, y2).

Otra forma es manteniendo abierta la pantalla de gráficos, para ello se ejecuta la instrucción *"hold on"* como se muestra:

```
plot(t,y1)
hold on
plot(t,y2)
```

Para salir de la pantalla de gráficos se ejecuta *hold off*.

Puede ponerse título, comentarios en los ejes o en cualquier otra parte con los siguientes comandos que tienen una cadena como argumento:

title	('título del gráfico')
xlabel	('comentario en el eje x')
ylabel	('comentario en el eje y')
axis	permite escalar los ejes manualmente
grid	cuadriculado en el gráfico
legend	permite etiquetar el gráfico

La figura muestra el resultado de utilizar los comandos mencionados anteriormente.

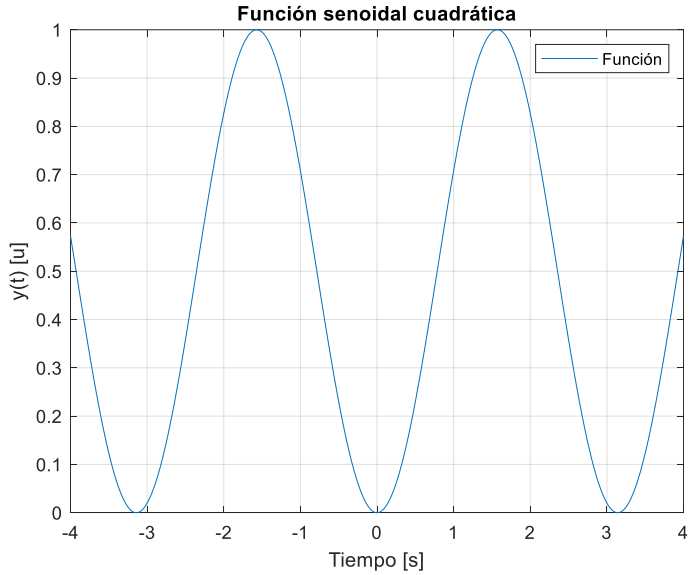

Figura 2. Señal sinusoidal con etiquetas

Si se desea poner variar gráficos en una sola ventana de gráficos distribuidos de manera ordenada, se usa el comando *tiledlayout.* Se obtendrá un resultado como este:

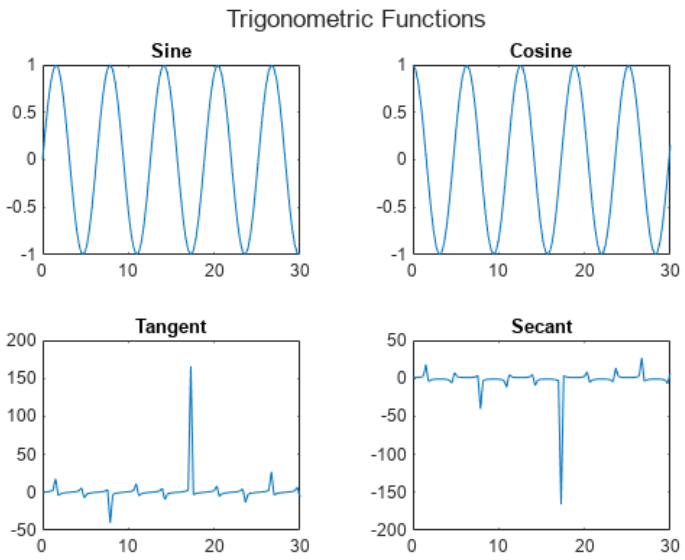

Figura 3. Grafica por medio del comando tilelayout

Para más información sobre edición de los gráficos se pueden consultar el siguiente enlace: https://www.mathworks.com/help/matlab/ref/plot.html

3.4. Script y Live Script

Matlab permite colocar órdenes en un simple archivo de texto y, a continuación, decirle a matlab que lo abra y evalúe las órdenes exactamente como si hubiesen sido escritas desde la línea de orden de Matlab. Estos archivos se llaman archivos script o archivos-M, y deben finalizar con la extensión 'm'.

Para crear un archivo .m en Matlab, se debe dirigir a la barra de herramientas y seleccionar NEW SCRIPT FILE.

A partir de la versión 2016A es posible combinar código con texto en un archivo denominado ***Live Script***, el cual permite obtener un documento dinámico en el cual se puede observar de manera más amigable los resultados de nuestro código. Los archivos de un live script finalizan con la extensión. mlx

Para crear un archivo. mlx en Matlab, se debe dirigir a la barra de herramientas y seleccionar NEW LIVE SCRIPT FILE.

3.5. Funciones y subfunciones

Las funciones permiten definir funciones enteramente análogas a las de MATLAB, con su nombre, sus argumentos y sus valores de retorno. Los ficheros *.m que definen funciones permiten extender las posibilidades de MATLAB; de hecho, existen bibliotecas de ficheros *.m que se venden (toolkits) o se distribuyen gratuitamente (a través de Internet). Las funciones definidas en ficheros *.m se caracterizan porque la primera línea (que no sea un comentario) comienza por la palabra function, seguida por los valores de retorno (entre

corchetes [ ] y separados por comas, si hay más de uno), el signo igual (=) y el nombre de la función, seguido de los argumentos (entre paréntesis y separados por comas).

function *[lista de valores de retorno] =* ***name****(lista de argumentos)*

Nota: name es el nombre de la función dentro del archivo name.m

La lista de valores de retorno y/o argumentos pueden no ir.

3.6. Simulink

Simulink es un entorno gráfico para modelación y simulación de sistemas. Las diferentes bibliotecas que posee permiten construir funciones y realizar el análisis del modelo de una manera sencilla empleando simulación. Para invocar a Simulink, basta teclear en la pantalla de comandos:

>> Simulink

A breves rasgos Simulink permite:

- Dibujar elementos y conexiones en una ventana gráfica. Las conexiones indican el recorrido de las señales de un elemento a otra.
- Los elementos se los extrae de la biblioteca del propio Simulink, e inclusive se pueden crear nuevos elementos.
- Los resultados se obtienen como salida de algunos elementos, pudiéndose almacenar, ver gráficamente, etc.
- Los datos o señales de entrada pueden obtenerse de salidas de variables especiales, del disco o de variables utilizadas en MATLAB.

Al abrir el toolbox Simulink se abre la pantalla de bloques que usa para su funcionamiento, entre ellos tenemos:

Continuos: integradores, derivadores, función de transferencia, retardo de transporte, memorias, etc.

Discretos: funciones de transferencia discreta, filtros digitales, ZOH, espacio de estado discreto, etc.

Matemática: sumadores, ganancias, funciones trigonométricas, matrices, etc.
Fuentes: escalón unitario, seno, ruido blanco, variables desde un archivo. mat, generadores de señales, etc.

No-lineales: switches, relees, etc.

Señales y Sistemas: entradas y salidas; multiplexores y demultiplexores para varias entradas y/o salidas y para vectores.

Salidas: displays, osciloscopios, salidas a archivos. mat, o al espacio de trabajo.

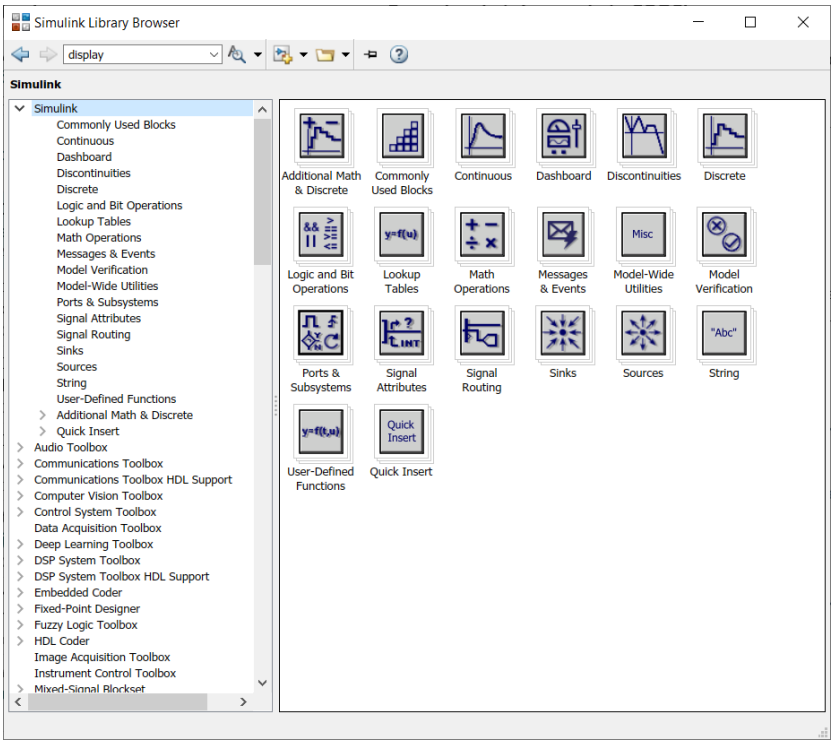

Figura 4. Librerías de Simulink

Para analizar un modelo de cualquier tipo, se empieza creando un archivo nuevo tipo "model" (elegir opción desde File...New), y se abren los distintos grupos de funciones a utilizar simplemente arrastrando con el mouse, al área de trabajo, los bloques deseados. Luego se procede a conectarlos por medio de un click sostenido uniendo sus entradas y salidas.

Después se configuran los parámetros de cada bloque según el modelo y los parámetros con los que se trabajará en el menú de simulación, como: el tiempo de inicio, tiempo de finalización; tipo de algoritmo de integración, etc. Finalmente, se inicia (start) la simulación.

El progreso de la simulación se puede observar en la pantalla mientras ésta corre y al final, los resultados se pueden guardar en el espacio de trabajo de MATLAB creando archivos .mat de la misma manera para guardar la información y posteriormente analizarla o imprimirla.

3.6.1 Parámetros de simulación (Configuration Parameter)

Existen diversos parámetros de la simulación que se pueden modificar, el primero de ellos es el tiempo de simulación (por defecto 10 seg.) se entra en el menú simulación en el submenú de parámetros de configuración (Model Parameters Configruation) o simplemente ingresando el comando Ctrl+E. Con ello aparece una ventana como la mostrada en la figura 5. Otras opciones a conocer son:

- Tiempo inicial (por omisión 0 seg.)
- Tiempo final (por omisión 10 seg.)
- Opciones de la integración numérica
- Tipo: puede ser de paso fijo o variable

- Método: varía desde el más sencillo Euler (paso fijo) a otros más sofisticados como Dormand-Price.

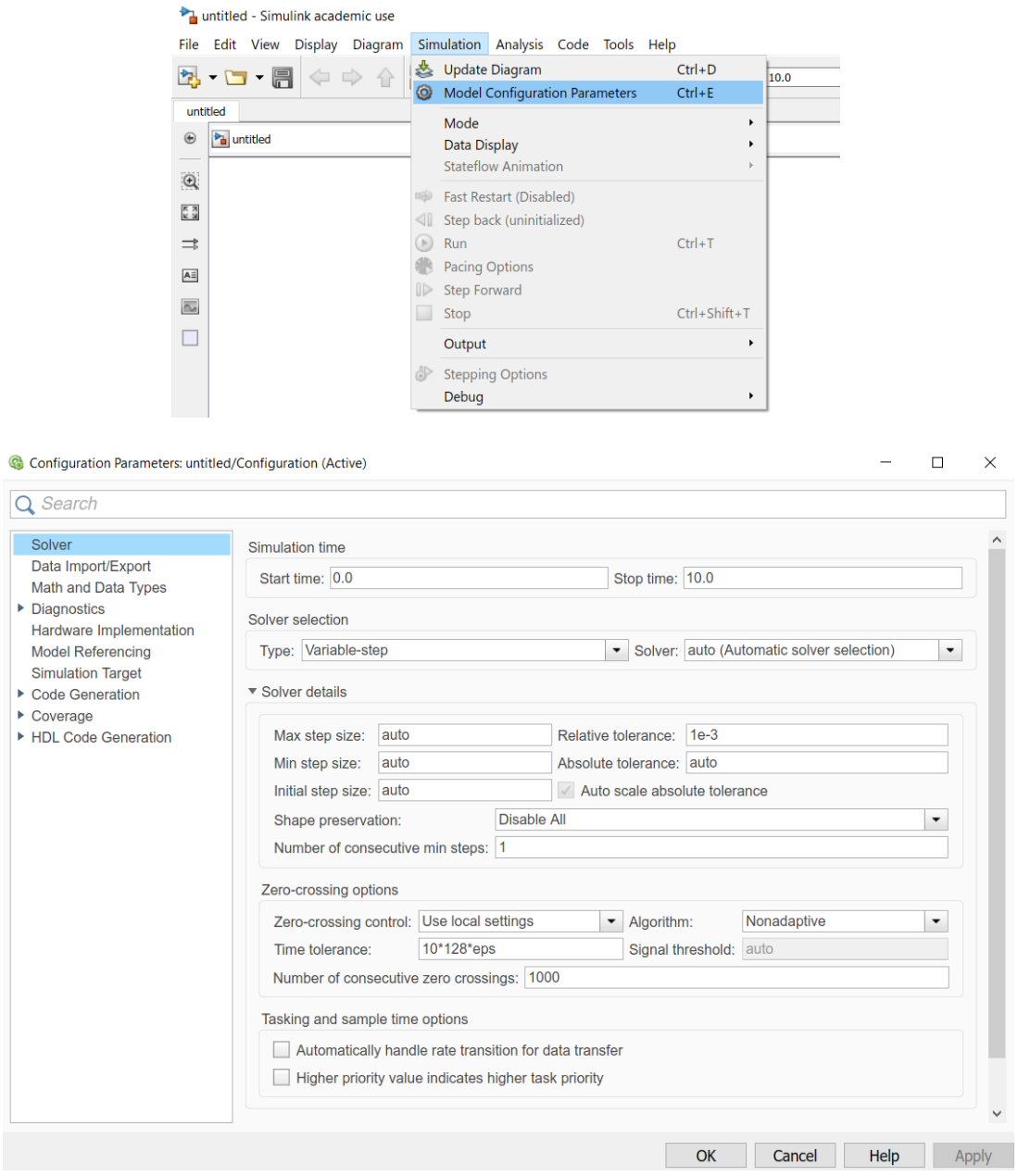

Figura 5. Parámetros de Simulación

3.6.2 Relación entre SIMULINK y Workspace de Matlab

Es posible utilizar los datos generados en SIMULINK en el workspace de MATLAB, para ellos existen bloques específicos:

FromWorkspace % Para traer datos

To Workspace % Para enviar datos

Cuando se requiere obtener datos a partir del Workspace, siempre es necesario definir una matriz de dos dimensiones, en donde estarán el tiempo y los datos asociados.

Otra de las formas y es configurando el elemento del cual se quiere obtener datos y guardar estos como un vector o un arreglo, al realizar la simulación, estas variables se guardaran directamente en el Workspace para su posterior uso.

3.7 Cuenta de MathWorks

Mathwoks es una corporación estadounidense, la cual desarrollo Matlab y Simulink, para usar todas sus herramientas es necesario disponer de una cuenta de Mathworks. Entre los recursos accesibles están:

- Matlab Online: versión en línea de Matlab.
- Matlab Mobile App: versión para dispositivos móviles de Matlab.
- Matlab Drive: servicio de almacenamiento de archivos en la nube.

Tanto Matlab Online como Matlab Mobile App usa Matlab Drive para almacenar los archivos los archivos y datos que el usuario requiera.

Matlab al igual que otros servicios de almacenamiento en la nube permite compartir los archivos entre usuarios. Con una cuenta de Mathworks asociada a una licencia de Matlab se tiene 5 GB de almacenamiento.

4. TRABAJO PREPARATORIO

4.1. Leer la información proporcionada en la hoja guía. *El instructor podrá tomar una pequeña prueba sobre esta información.*

4.2. Consultar:

4.2.1. ¿Qué es una función, un script y un live script en Matlab? ¿Cuál es la diferencia?

4.2.2. Como obtener los datos generados Simulink en el Workspace de Matlab y viceversa.

4.2.3. Consulte sobre la librería Dashboard en Simulink.

4.3. Considere la siguiente matriz:

$$A = \begin{bmatrix} 24 & 17 & 6 \\ 16 & -21 & -4 \\ -8 & 11 & 8 \end{bmatrix}$$

Utilizando Matlab, indicar el resultado de las siguientes operaciones y comentar que significa cada una de ellas:

a. A(:,1)	b. det(A)
c. A(:,2:3)	d. poly(eig(A))
e. B=[A,[ones(1,2);eye(2)]]	f. A(:,:)
g. diag(A)	h. ones(4,3)
i. eye(3)	j. A(2:4)
k. zeros(size(A))	l. rand(size(A))
m. magic(length(A))	n. rank(A)

4.4. Graficar las siguientes funciones sobre un mismo gráfico.

$$x(t) = \frac{1 - sen^2(2t)}{2t}$$

$$y(t) = \frac{2t + 1}{t^2 + 2}$$

Para $-10 \leq t \leq 10$ y considerar que los vectores deben tener al menos 300 puntos. Incluir títulos y etiquetas en los ejes.

5. EQUIPOS Y MATERIAL

- Computador con conexión a internet y Software de Control y Simulación Matlab.

6. DESARROLLO

6.1 El instructor realizará una introducción al entorno de Matlab: uso de comandos generales, generación de funciones y scripts.

6.2 Empleando Matlab graficar la órbita la luna alrededor del sol durante un año. La órbita de la luna viene dada por las siguientes ecuaciones dados en [km]:

$$x(t) = R\,cos(\varphi t) + r\,cos\,(\omega t)$$

$$y(t) = R\,sen(\varphi t) + r\,sen\,(\omega t)$$

Donde R es el radio de órbita de la tierra respecto al sol, r es el radio de órbita de la luna respecto a la tierra, φ es la velocidad angular la de la tierra alrededor del sol y ω es la velocidad de la luna alrededor de la tierra.

$R = 1.496x10^8\ [km]$
$r = 3.845x10^5\ [km]$
$\varphi = 2\pi/365\ \ [dias^{-1}]$
$\omega = 2\pi/27.3\ \ [dias^{-1}]$

Generar tanto *x(t)* e *y(t)* como funciones de Matlab y la variable independiente *t.* Incluir etiquetas en los ejes, título de los gráficos, etc.

6.3 En la figura 6 se muestra un sistema de 2 vehículos cuyas masas m1 y m2 están sujetas por un amortiguador P y un resorte de constante k2 y además la masas m1

sujeta a la pared con un resorte de constante k1. La posición de cada uno de los vehículos (x_1, x_2) cambia cuando se aplica al sistema una fuerza $f_A(t)$

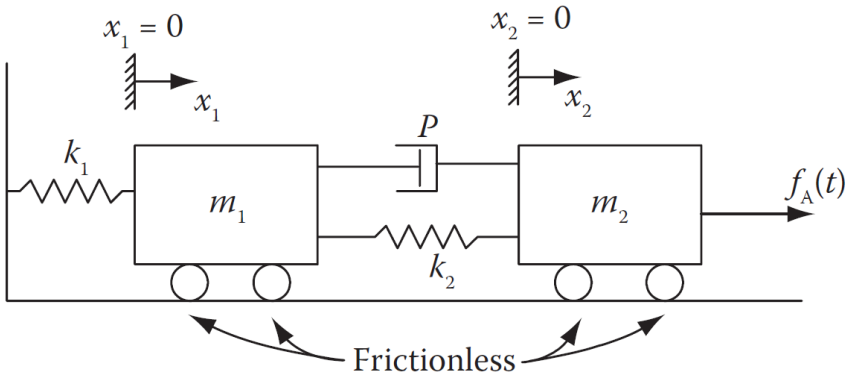

Figura 6. Sistema de dos vehículos

Donde el sistema está representado por las siguientes ecuaciones:

$$m_1 \frac{d^2x_1}{d^2t} = -k_1x_1 - k_2(x_1 - x_2) - P\left(\frac{dx_1}{dt} - \frac{dx_2}{dt}\right)$$

$$m_2 \frac{d^2x_2}{d^2t} = f_A(t) - k_2(x_2 - x_1) - P\left(\frac{dx_2}{dt} - \frac{dx_1}{dt}\right)$$

Asumiendo las condiciones iniciales siguientes:

$$\frac{dx_1}{dt}(0) = \frac{dx_2}{dt}(0) = 0$$

$$x_1(0) = x_2(0) = 0$$

6.3.1 Implementar en Simulink las ecuaciones diferenciales del modelo anterior y mostrar el comportamiento tanto de las posiciones como de las velocidades de los vehículos.

6.3.2 Ejecutar el modelo de Simulink o desde el live script de Matlab considerando los siguientes valores:

$m_1 = 3\,Kg$ $m_2 = 5\,Kg$ $k_1 = k_2 = 60\frac{N}{m}$ $P = 5\,N\frac{s}{m}$ $f_A = 1\,N$

Presentar las gráficas en el script o live script de Matlab con los datos obtenidos en Simulink.

7 INFORME

7.1 Presentar los resultados obtenidos en el desarrollo y realizar un análisis y comentarios de estos.

7.2 Empleando Matlab graficar la órbita de Mercurio (x vs. y) alrededor de la tierra. La órbita de Mercurio está dada por las siguientes ecuaciones:

$$x(t) = 93cos\, t + 36cos\, 4.15t$$

$$y(t) = 93sen\, t + 36sen(4.15t)$$

Generar tanto *x(t)* e *y(t)* como funciones de Matlab y la variable independiente *t* desde 0 hasta $44\pi/3$ en intervalos de $\pi/360$. Incluir etiquetas en los ejes, título de los gráficos, etc.

7.3 Conclusiones y Recomendaciones.

8 REFERENCIAS

Pinto, E; Matía, F. “Fundamentos de control con Matlab”; PRETICE HALL; Edición 1.

Hernández R., “Introducción a los sistemas de control: Conceptos, aplicaciones y simulación con MATLAB”, Pearson Education, 1ra. Edición, 2010.

Elaborado por: Ing. Ronald Pillajo.

Revisado por: Ing. Yadira Bravo, MSc.

Dr. Paulo Leica