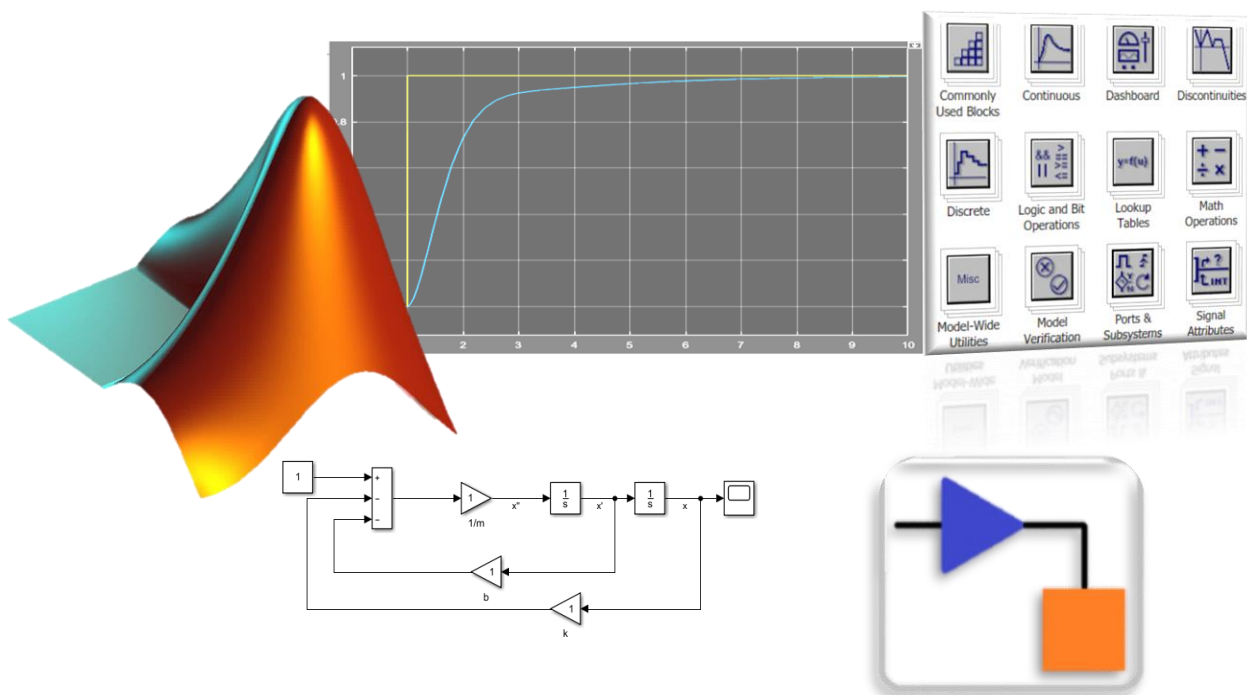




**Universidad Nacional Autónoma  
de México**

**Facultad de Estudios Superiores  
Cuautitlán**



**Manual utilización de MATLAB  
enfocado a temas de Teoría de  
Control**

# Índice

	Página
- Introducción a MATLAB y Simulink.....	4
- Diagramas de bloques.....	12
Ejemplo 1.....	12
Ejemplo 2.....	19
Ejemplo 3.....	21
- Modelado de sistemas en Matlab Simulink.....	24
Ejemplo 1.....	24
Ejemplo 2.....	26
Ejemplo 3.....	28
- Transformada de Laplace.....	33
Ejemplo 1.....	33
Ejemplo 2.....	33
Ejemplo 3.....	34
- Anti transformada de Laplace.....	35
Ejemplo 1.....	35
Ejemplo 2.....	35
Ejemplo 3.....	36
- Fracciones parciales.....	37
Ejemplo 1.....	37
Ejemplo 2.....	38
Ejemplo 3.....	38
- Sistemas de primer orden.....	40
Ejemplo 1.....	40
Ejemplo 2.....	42
Ejemplo 3.....	43
- Sistemas de segundo orden.....	45
Ejemplo 1.....	45
Ejemplo 2.....	47
Ejemplo 3.....	49
- Controlador P.....	52
Ejemplo 1.....	52
Ejemplo 2.....	54
Ejemplo 3.....	57
- Controlador PI.....	60
Ejemplo 1.....	60
Ejemplo 2.....	62
Ejemplo 3.....	64
- Controlador PID.....	67
Ejemplo 1.....	67
Ejemplo 2.....	68
Ejemplo 3.....	70
- Diagramas de Magnitud y Fase.....	72
Ejemplo 1.....	72
Ejemplo 2.....	75
Ejemplo 3.....	76
Ejemplo 4.....	77

- Estabilidad.....	78
Ejemplo 1.....	80
Ejemplo 2.....	81
Ejemplo 3.....	81
Ejemplo 4.....	82
Ejemplo 5.....	83
- Lugar Geométrico de la Raíz.....	84
Ejemplo 1.....	84
Ejemplo 2.....	87
Ejemplo 3.....	88
- Bibliografía.....	91

## Introducción a MATLAB

El software debe estar previamente instalado. Una vez ejecutado el Software de MATLAB, aparecerá una ventana principal.

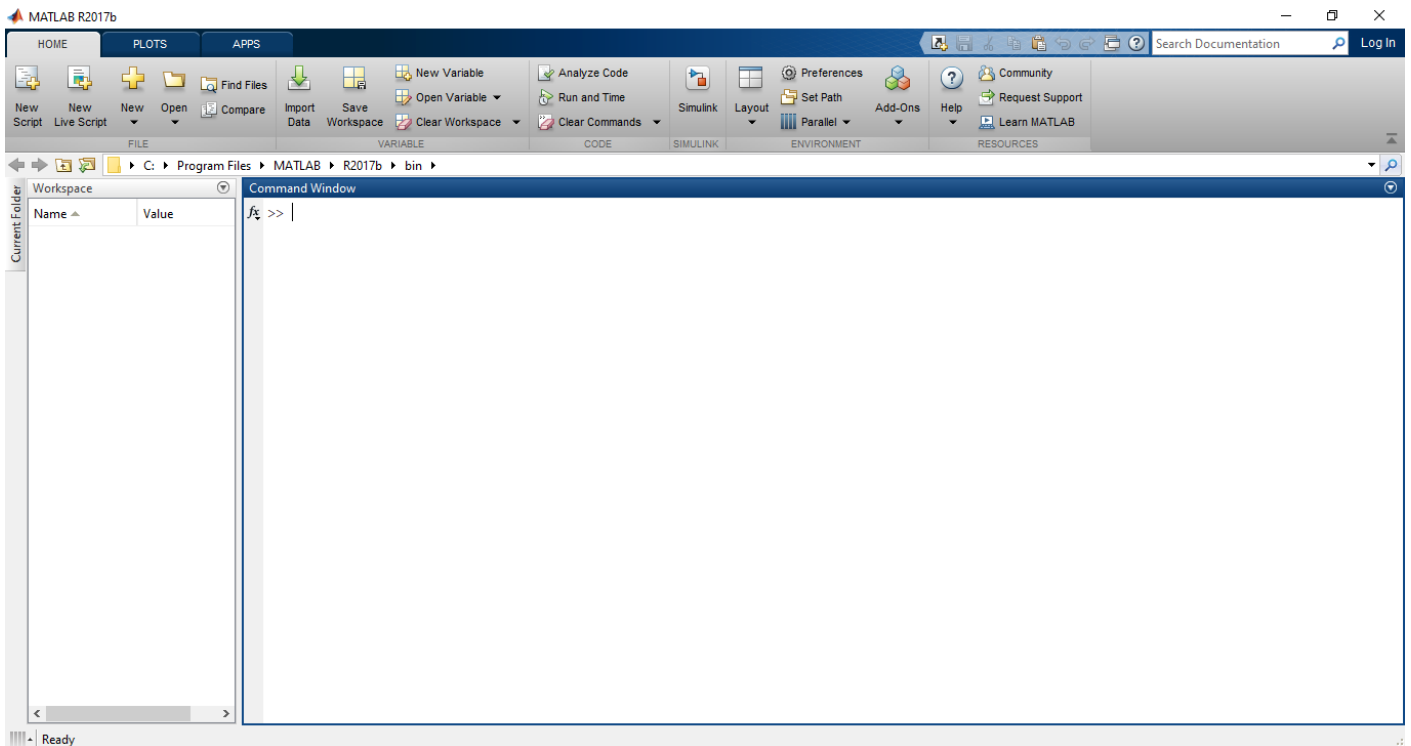


Figura 1. Ventana principal de MATLAB.

En la pestaña de nombre HOME, se muestran distintos menús.

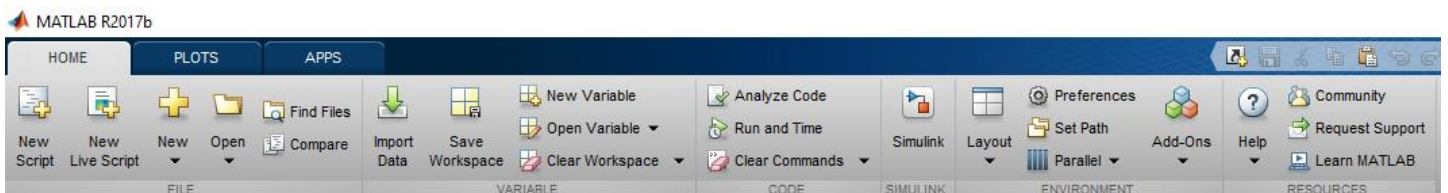
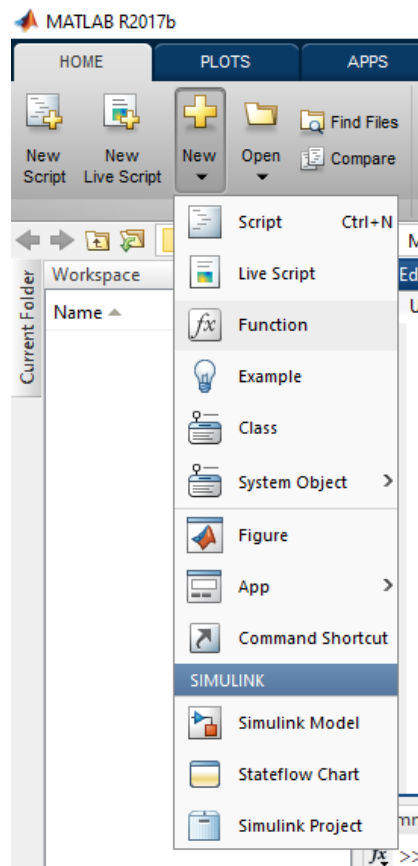


Figura 2 Pestaña HOME.

En el primer menú de nombre FILE, se nos permiten realizar acciones básicas como abrir un archivo existente, buscar, comparar, pero también en el icono de "New", encontramos diversas opciones de acciones como se muestra en la Figura 3.



>>Figura 3. Herramientas ofrecidas.

Dentro de estas opciones aparece el icono de Simulink, esta es una herramienta de diseño cuya finalidad es variar los parámetros y diseño de un sistema observando sus respuestas para decidir antes de ensamblar o llevar a cabo el proyecto.

Al dar clic en “*New Script*” la ventana principal se verá de la siguiente manera:

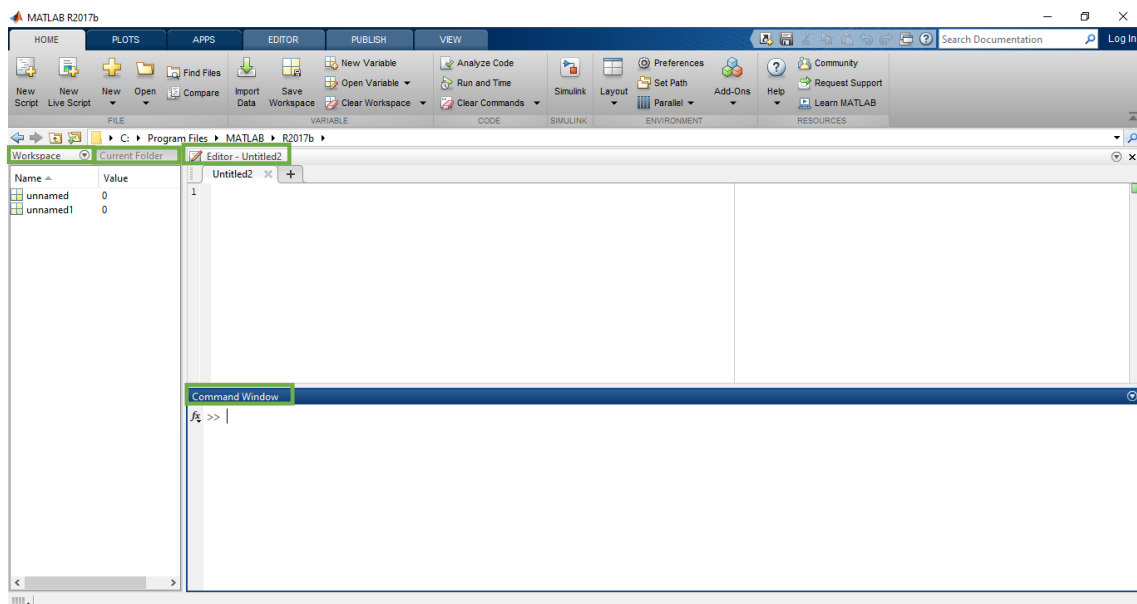


Figura 4. Entorno común de MATLAB.

- Workspace: se muestran las variables y objetos utilizados.
- Current Folder: muestra el contenido de la carpeta de trabajo.
- Editor (Command History): Su función es ejecutar líneas de comando secuencial en vez de individualmente y permite observar los últimos comandos ejecutados.
- Command Window: Es el espacio en el que se escribe el código o comandos para realizar acciones específicas.

En el siguiente menú de nombre VARIABLE encontraremos comandos para guardar e importar datos, y para utilizar variables ya sea guardadas previamente o nuevas, también nos ofrece la opción de limpiar la ventana de comandos de todo el código que hayamos escrito con anterioridad.



Figura 5. Menú VARIABLE.

Al dar clic en la opción “*New Variable*” obtendremos una ventana como la siguiente:

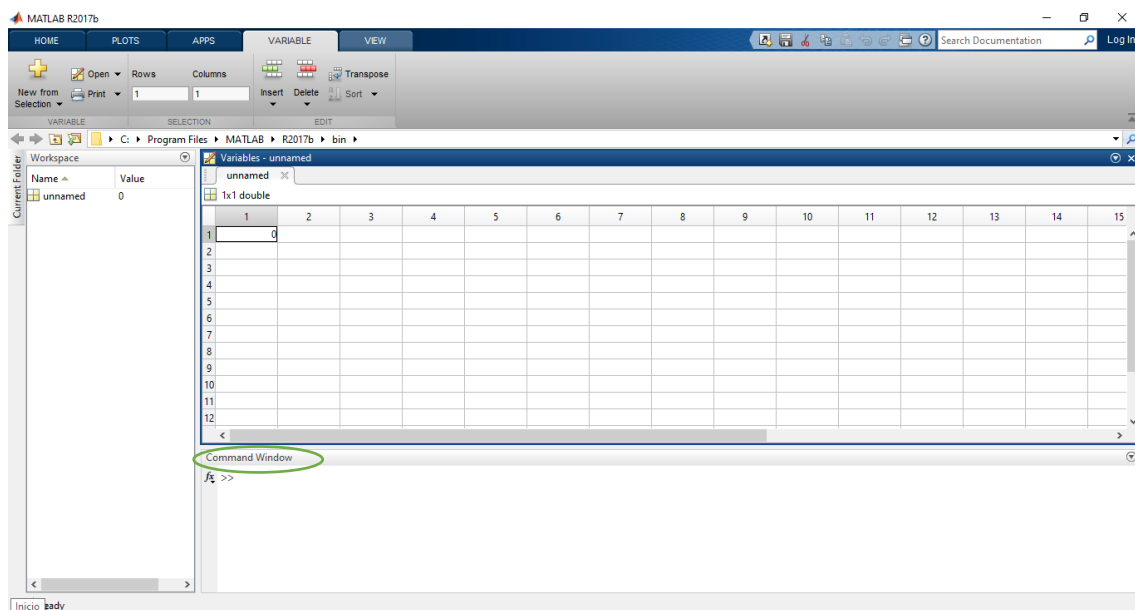


Figura 6. Edición de Variables.

En esta podremos editar las variables que necesitemos, tanto su nombre como sus valores, es importante al momento de escribir el código utilizar el mismo nombre asignado a las variables. El código deberá escribirse en la ventana que se encuentra abajo llamada “*Command Window*”.

En esta ventana podremos realizar operaciones aritméticas básicas, operaciones con funciones trigonométricas, operaciones indefinidas, declaración de variables y operaciones con estas, operaciones con vectores, creación de matrices, además de poder cambiar el formato de la respuesta a la operación.

```

Command Window
>> a= sin(pi)
a =
    1.2246e-16
>> format long, a
a =
    1.224646799147353e-16
>> 1/0
ans =
    Inf
>> b=0:5:10
b =
     0     5    10
>> A=[44;44]
A =
    44
    44
fx >> |

```

Figura 7. Ejemplo de algunas operaciones y comandos.

En la *Figura 6* se muestra una asignación de valor a la variable a que a su vez contiene una operación con una función trigonométrica y el valor de Pi, al final de cada línea de comando se debe dar ENTER para que este se ejecute. En la siguiente línea se puede observar el cambio de formato a largo, este comando puede acortar el formato, ponerlo en radianes, entre otras; se debe poner coma y la variable o de lo contrario marcará error. También se muestra una operación indefinida y la declaración de un vector.

Al final se muestra la manera correcta de declarar una matriz, se deben de cuidar los signos a utilizar en los comandos y la disposición de estos para que el código funcione correctamente.

La notación del resultado también puede cambiarse desde la pestaña VIEW.

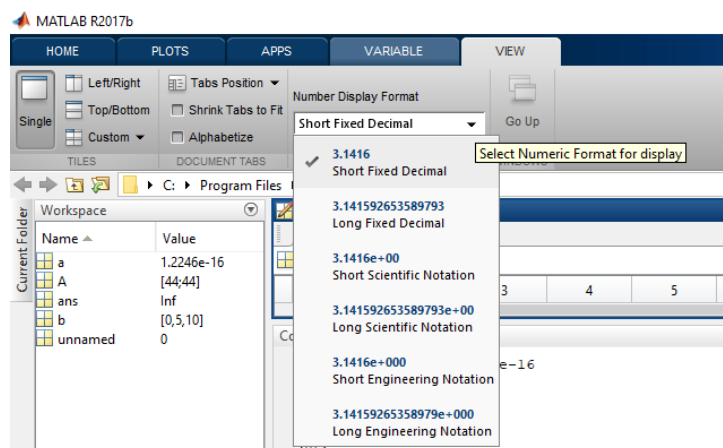


Figura 8. Pestaña VIEW.

Hay otras cosas que podemos editar en esa pestaña tales como el espacio de trabajo y la ubicación del nombre de las variables. *Figura 8.*

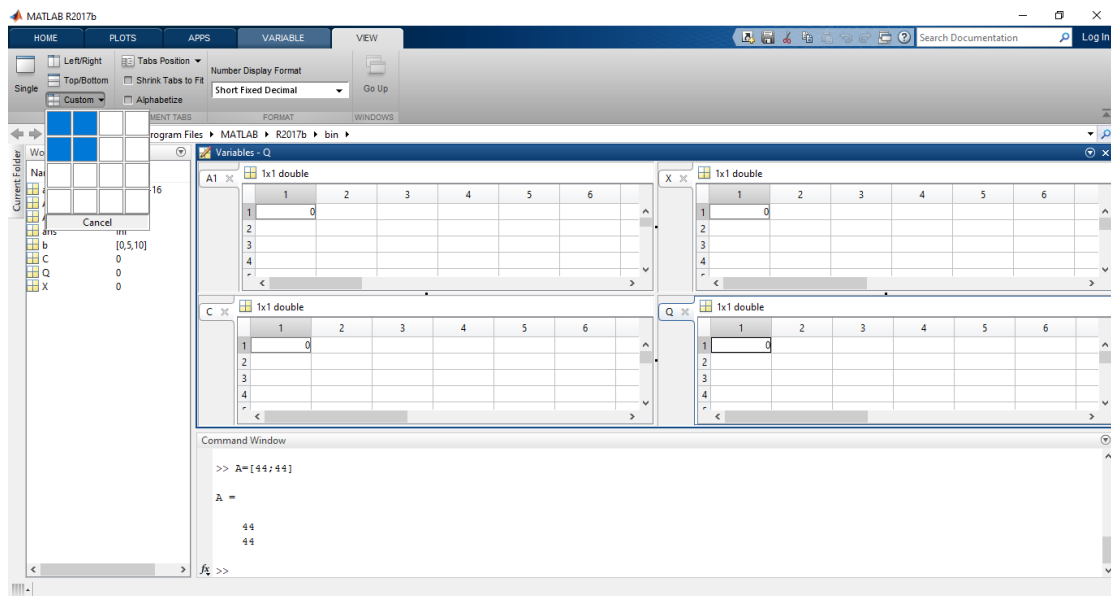


Figura 9. Personalización del área de trabajo.

Con los valores introducidos en esas variables podemos crear gráficas con el comando “*plot*”, con el nombre de las variables a graficar escritas entre paréntesis y separadas por comas.



Figura 10. Menús restantes.

En la *Figura 9* observamos los menús restantes, en CODE se muestran opciones para analizar y ejecutar el código previamente escrito; en la barra de SIMULINK se ejecuta esta herramienta, en la barra ENVIRONMENT podemos personalizar ciertas características de nuestro espacio de trabajo y por último en RESOURCES se ofrece ayuda incluso con videos para aprender a utilizar MATLAB.

Los menús contenidos en las pestañas restantes se muestran a continuación:

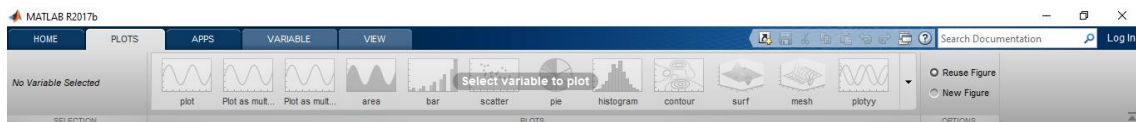


Figura 11. Personalización de gráficas.

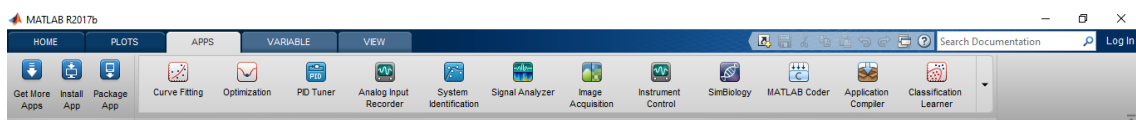


Figura 12. Aplicaciones auxiliares de MATLAB.



## Simulink

Esta herramienta se puede ejecutar dando clic en el ícono que se encuentra en la pestaña HOME o seleccionándolo de la barra File, en el icono *New* y posteriormente Simulink Model.

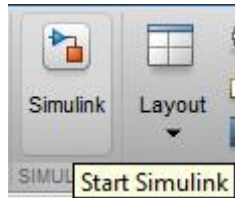


Figura 13. Icono para iniciar Simulink.

Aparecerá una ventana como la mostrada en la Figura 13:

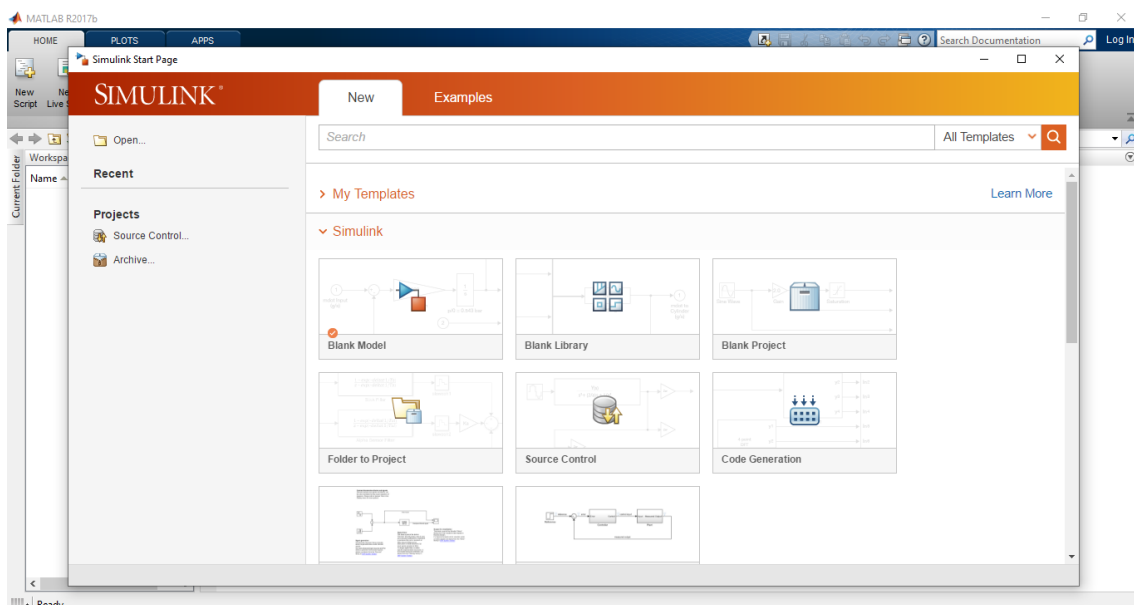


Figura 14. Ventana principal de Simulink.

Debemos seleccionar la primera opción “Blank Model”, una vez realizado esto se abrirá una ventana (Figura 14):

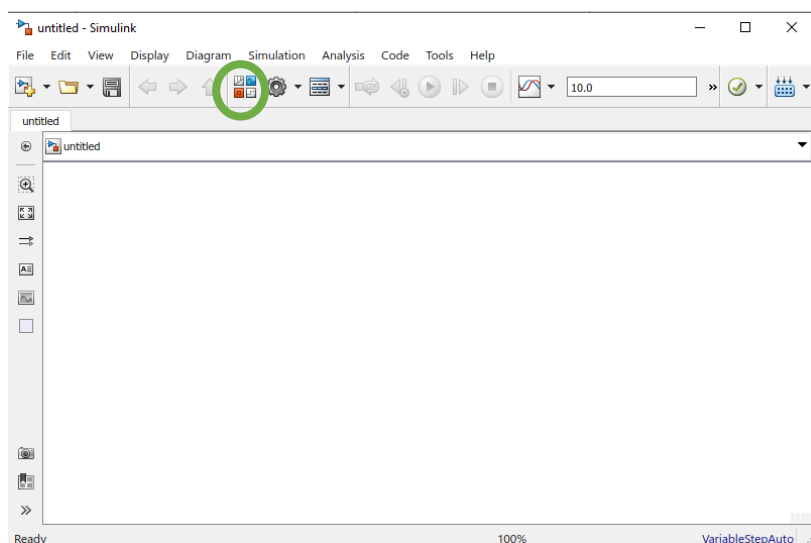


Figura 15. Ventana Simulink

De la cual debemos dar clic en el icono mostrado a continuación:



Figura 16. Icono para mostrar librerías.

Estas son las opciones que nos ofrece Simulink:

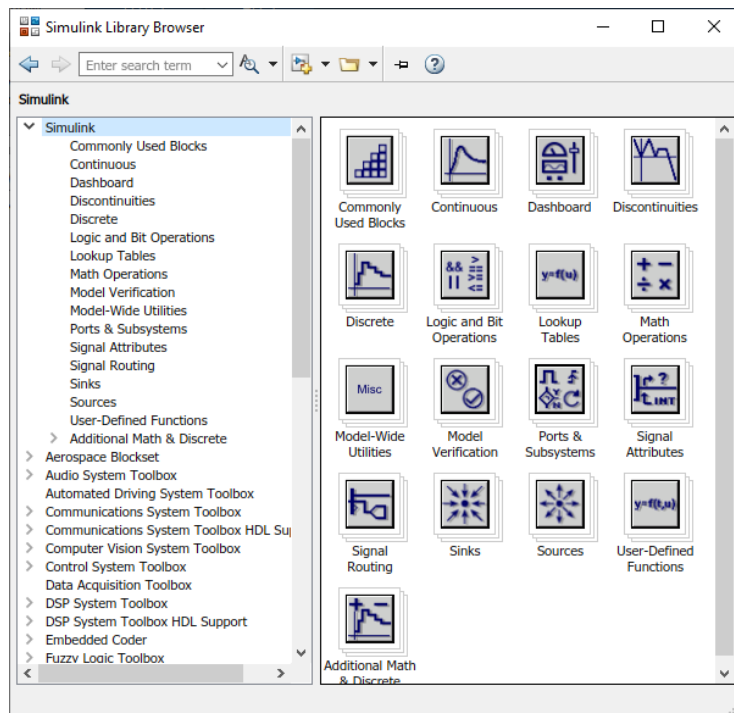


Figura 17. Librerías de Simulink.

Simulink ofrece una gran variedad de elementos a utilizar para modelar un sistema, hay tanto operadores matemáticos como elementos para sistemas de comunicaciones, simulación aeronáutica, elementos para analizar diversos fenómenos físicos, entre otros.

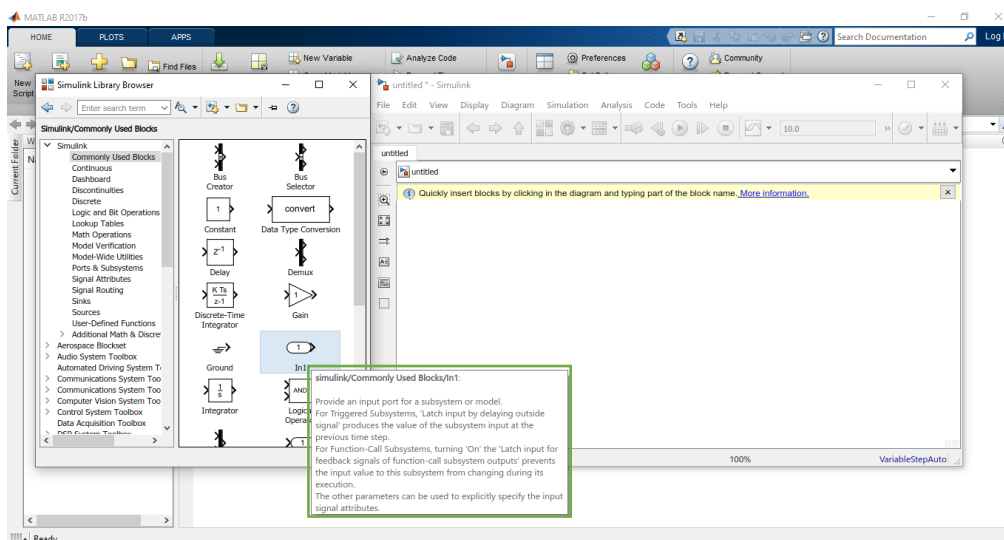


Figura 18. Información de los componentes.

Si acercamos el puntero sin dar clic nos mostrará información sobre cada componente.

Es importante tener las dos ventanas juntas ya que los componentes a utilizarse deben arrastrarse a la ventana de Simulink.

Para insertar algún componente debemos dar clic y sin soltar arrastrarlo hacia nuestro espacio de trabajo, una vez que se encuentre en el sitio deseado dejar de dar clic.

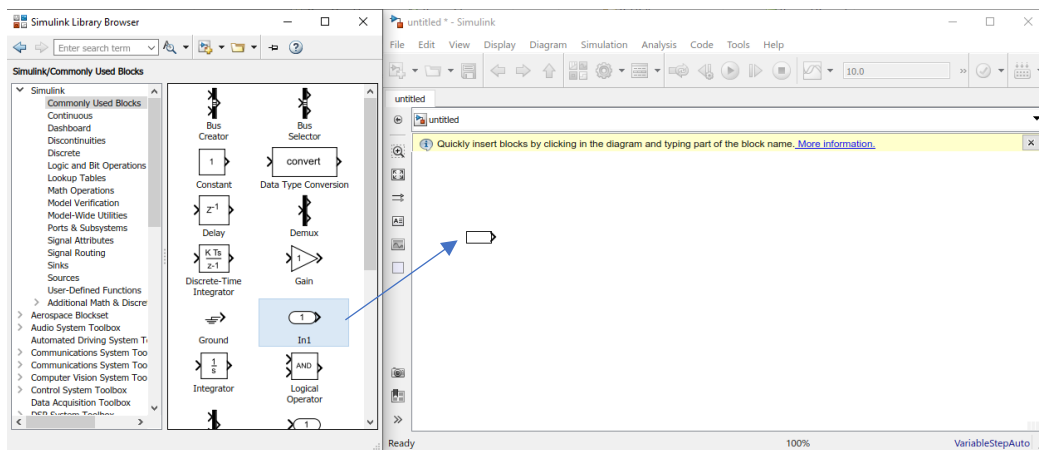


Figura 19. Inserción de componentes.

Existen diversas acciones realizables a cada componente, están las básicas como copiar, cortar y pegar, y hay opciones de cambio de formato, cambio de posición, requerimientos, propiedades, opciones de ayuda, entre otros.

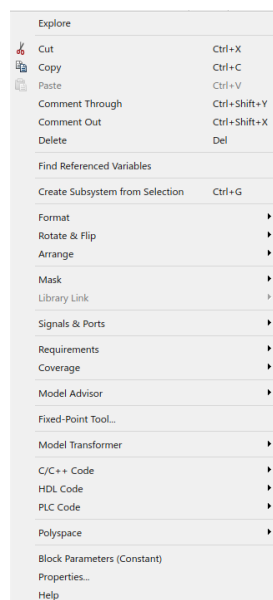


Figura 20. Opciones para cada componente.

A través de los temas se explicarán con detalle las acciones a realizar en distintos ejemplos.

## Diagrama de bloques

Para realizar diagramas de bloques en MATLAB, nos apoyaremos de la herramienta Simulink.

### Ejemplo 1:

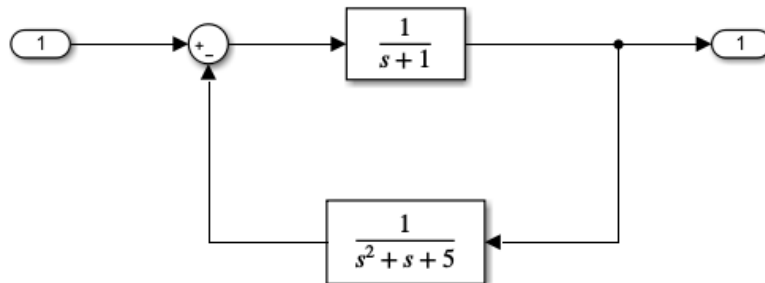


Figura 21. Diagrama de bloques de un sistema retroalimentado.

### 1.- Abrir la herramienta Simulink.

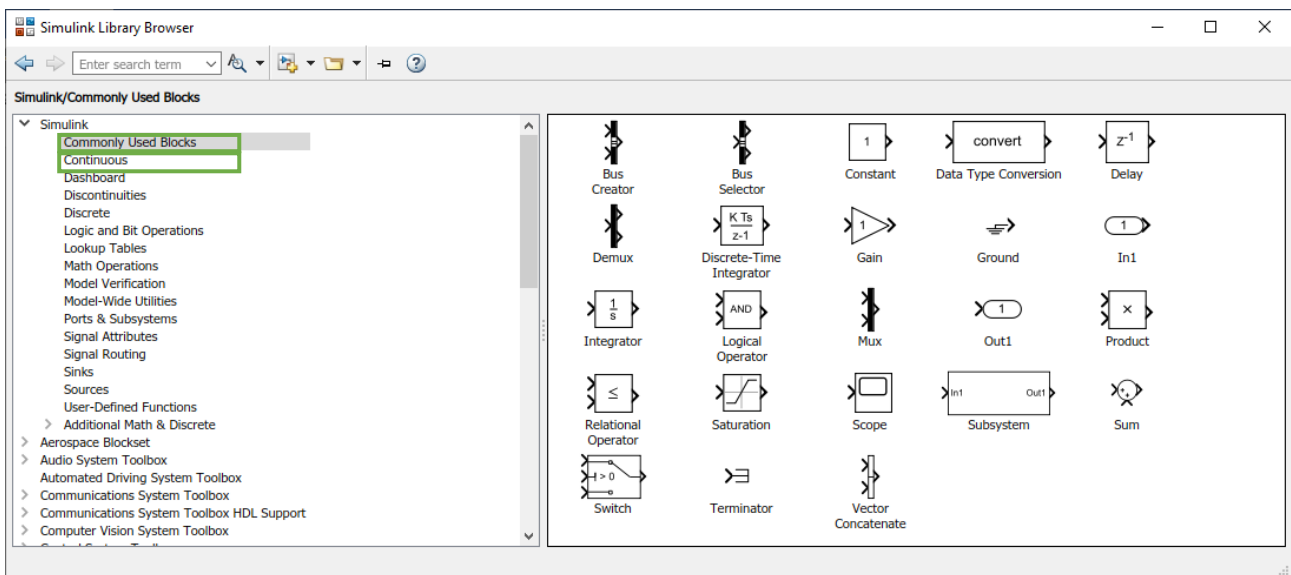


Figura 22. Componentes en la pestaña Commonly Used Blocks.

Utilizaremos las opciones que se encuentran en Commonly Used Blocks, también necesitaremos los elementos que se encuentran en la opción de Continuous para agregar algunas funciones.

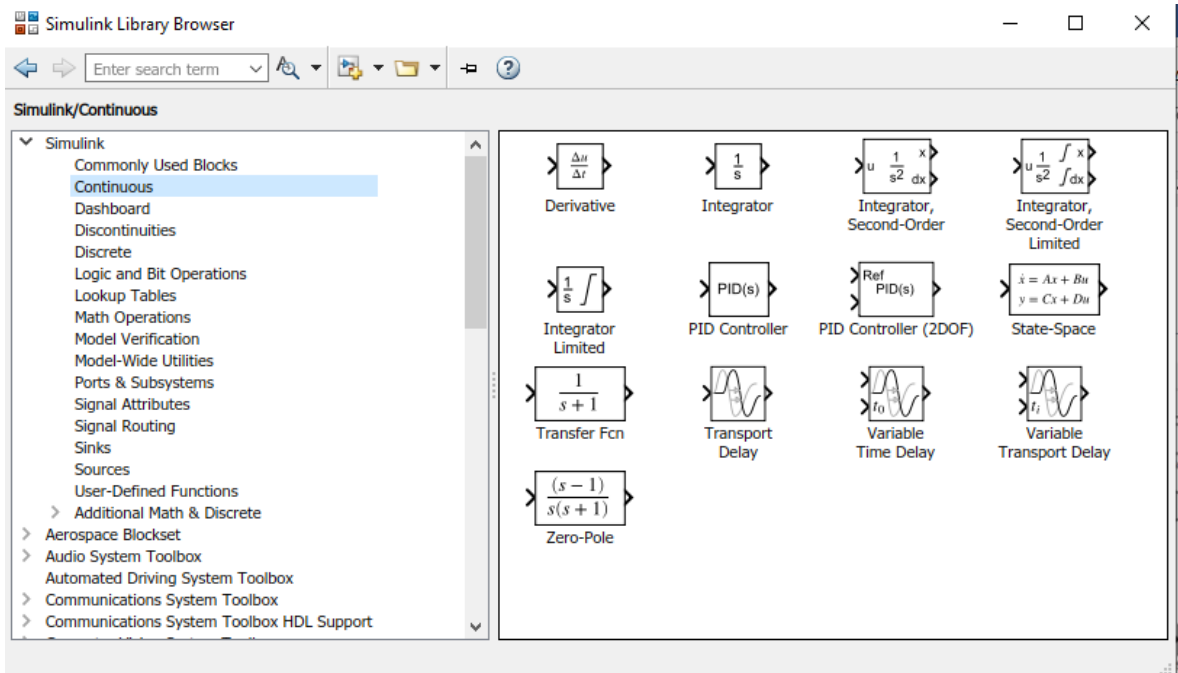


Figura 23. Opciones en la pestaña Continuous.

Lo primero que debemos hacer es insertar nuestras entradas y salidas (In/Out) y un sumador, que se encuentra en la primera pestaña.

Para insertar cualquier componente damos clic en él y sin soltar lo arrastramos a la ventana de trabajo de Simulink.

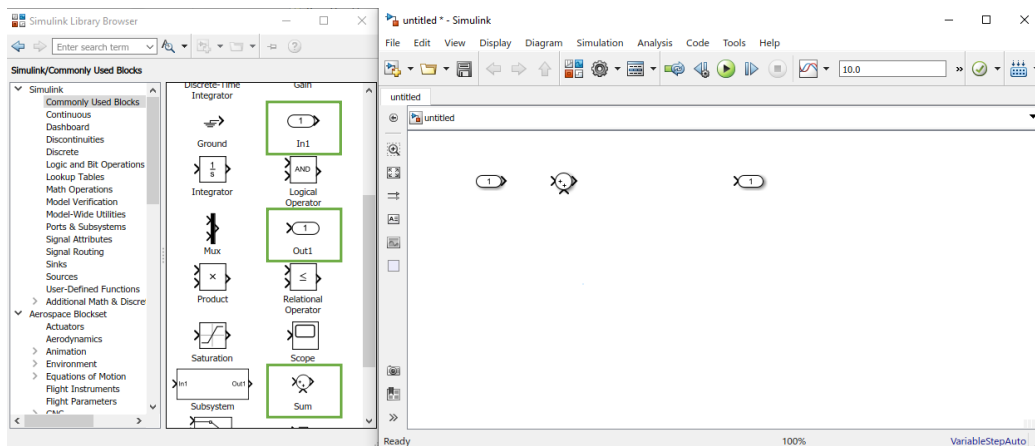


Figura 24. Elementos colocados en la ventana de trabajo.

A continuación, de la pestaña Continuous seleccionaremos la opción de "Transfer Form", e insertaremos dos funciones de transferencia.

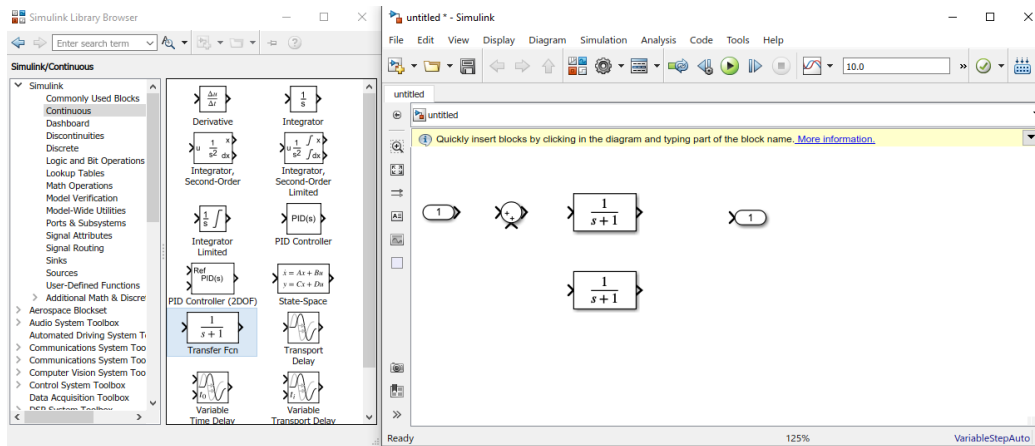


Figura 25. Inserción de los componentes que tendrá nuestro sistema.

Para unir los componentes del diagrama acercamos el cursor al componente que queremos unir, el cursor se pondrá en forma de cruz, dar clic y sin soltar dirigirse al siguiente componente, se creará una flecha color rojo.

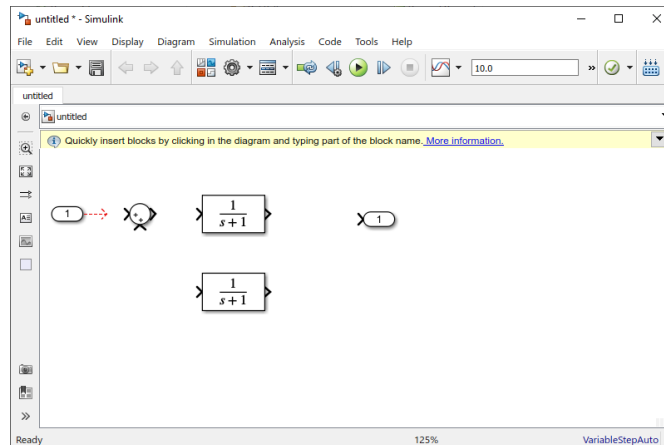


Figura 26. Flecha que indica la unión de componentes.

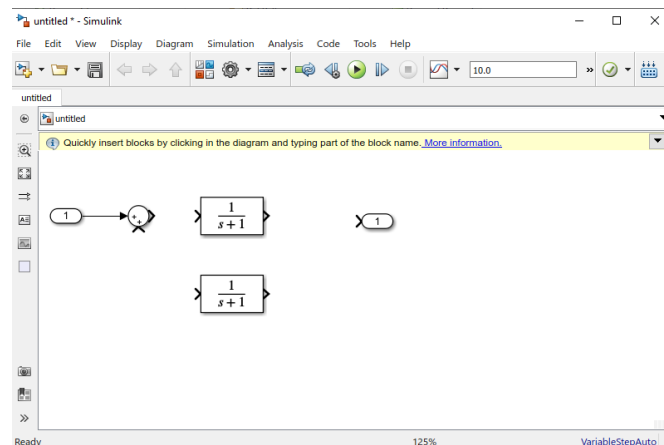


Figura 27. Componentes unidos.

Así se unen la entrada, el sumador, la primera función de transferencia y la salida. Para conectar la segunda función de transferencia, debemos dar clic derecho sobre ella, aparecerá un menú con varias opciones de las cuales seleccionaremos Rotate & Flip y después en el submenú, Flip Block para invertir el sentido del componente, una vez realizado esto ubicamos el cursor y lo

subimos a la línea que une la primera función de transferencia con la salida, también se puede girar el componente utilizando el comando **ctrl+R**.

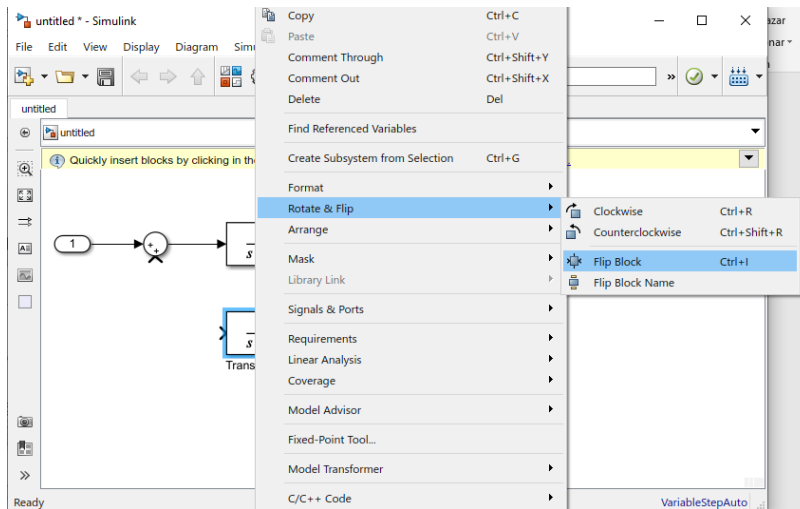


Figura 28. Inversión de sentido del componente.

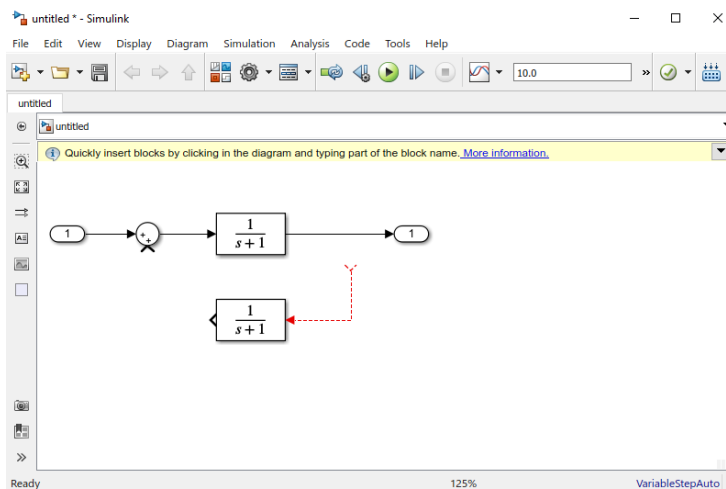


Figura 29. Indicador previo a la unión de componente y línea.

Si se necesita cambiar de lugar o recorrer un componente se debe dar clic y mover hacia el lugar deseado sin soltar hasta que esté ubicado correctamente.

Se pueden cambiar los signos encontrados en el sumador y su orden, para esto debemos dar doble clic sobre el sumador, aparecerá una ventana como la que se muestra a continuación:

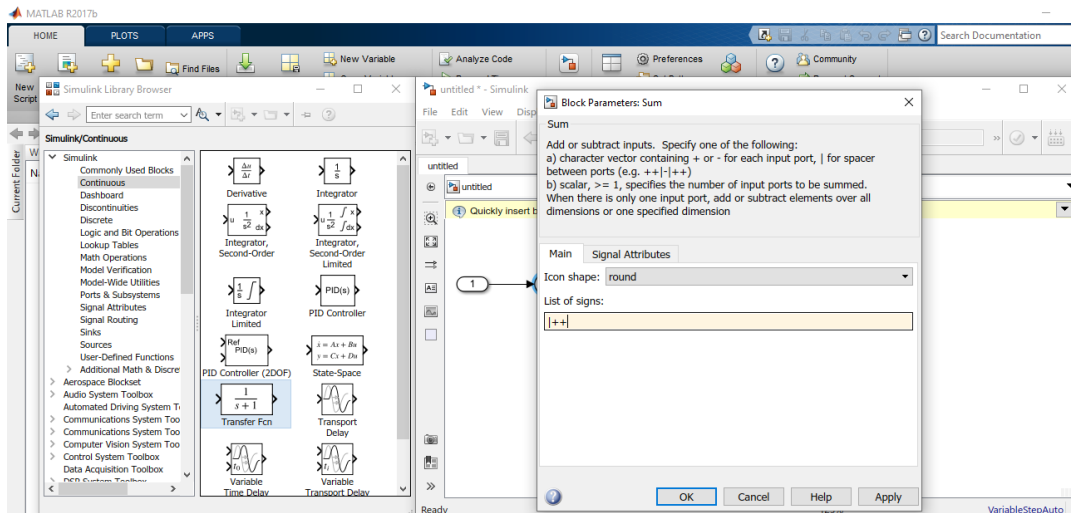


Figura 30. Menú para cambio de signos del sumador.

En este ejemplo se utilizará un más y un menos. Una vez abierto el menú, se debe cambiar el segundo signo que se encuentra escrito por un signo de menos y dar clic en *Ok*.

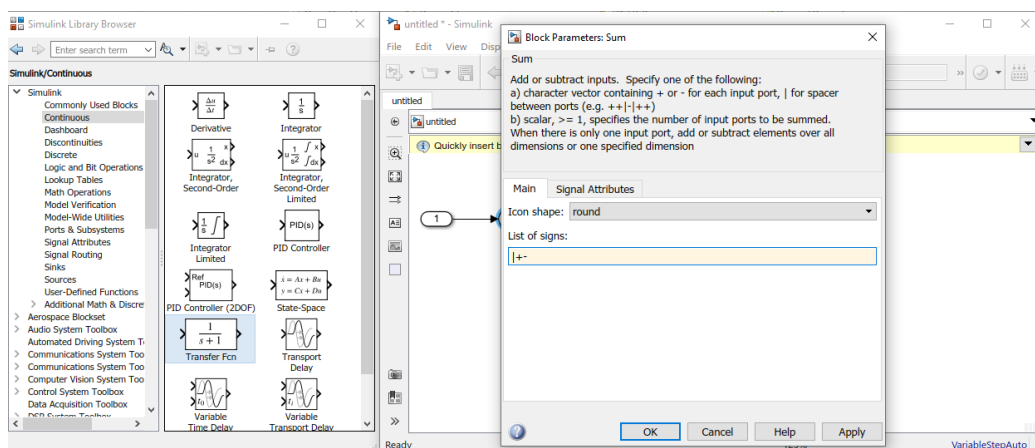


Figura 31. Cambio de signos en el sumador.

Para modificar la función, dar doble clic, aparecerá una ventana como esta:

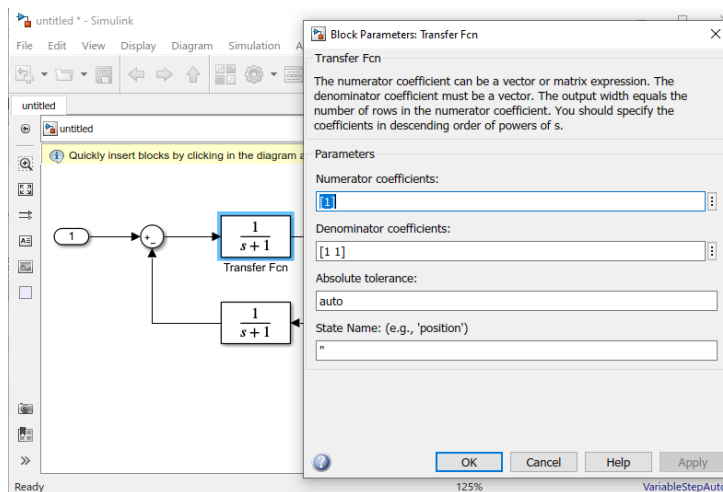


Figura 32. Edición de la función de transferencia.



Los 1 representan a las “s” y dependiendo el número de unos que aparezcan dentro de los corchetes será el grado de la función. Si hay uno es  $s^0=1$ , si hay dos  $s^1=s$ , pero la función es  $s+1$ , si es 2 tenemos  $s^2= s^2$  y así sucesivamente.

Para nuestro ejemplo modificaremos la función encargada de la retroalimentación para volverla cuadrática aumentando un uno a los corchetes del denominador y cambiando el valor del tercer uno a 5.

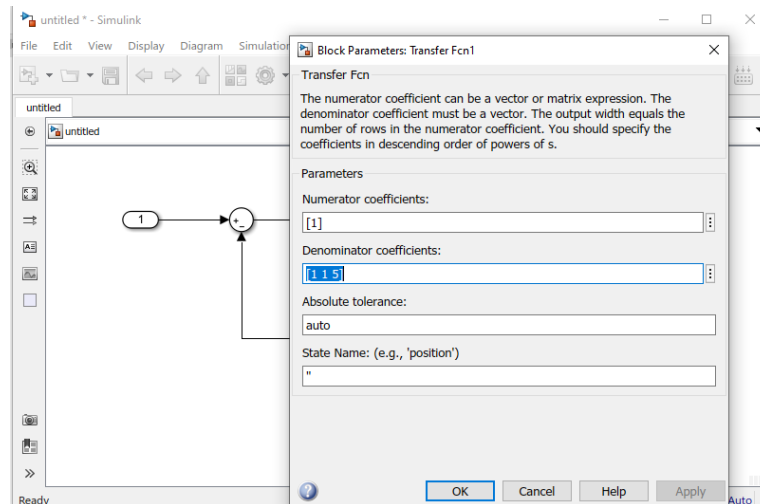


Figura 33. Edición de la función de transferencia.

Puede aparecer algo similar a (Figura 33), esto sólo quiere decir que el espacio es insuficiente y debemos agrandar el componente, esto se logra dando clic en el componente y posteriormente en los pequeños cuadros que aparecen alrededor para agrandararlo.

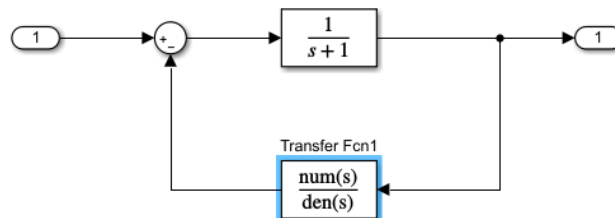


Figura 34. Espacio insuficiente.

El diagrama resultante debe verse como el mostrado en la Figura 34.

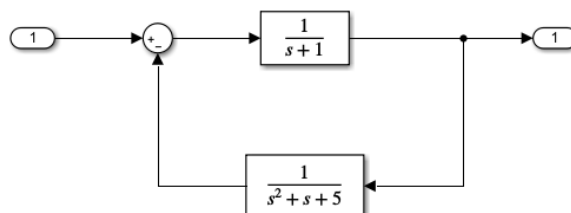


Figura 35. Diagrama de bloques finalizado.

Una vez que nuestro diagrama en Simulink esté realizado, debemos guardar el archivo dando clic en la pestaña File y seleccionando la opción de Save As, posteriormente debemos asignar un nombre y una ubicación al archivo.

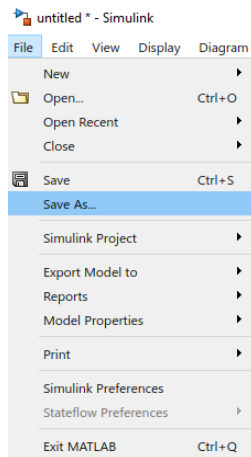


Figura 36. Menú File.

Debemos de mantener abierta la ventana de Simulink después de realizar este paso.

A continuación, regresaremos a MATLAB y en la ventana de comandos debemos escribir el comando **[num,den]=linmod(' ')** y entre las comas escribir el nombre del archivo de Simulink, una vez escrito esto dar Enter, se mostrará el resultado de la operación en el diagrama de bloques.

Para un mejor y más entendido acomodo, podemos utilizar el comando **G=tf(num,den)**.

```

Command Window
>> [num,den]=linmod('DB1')

num =

    0    1.0000    1.0000    5.0000

den =

    1.0000    2.0000    6.0000    6.0000

>> G=tf(num,den)

G =

      s^2 + s + 5
-----
    s^3 + 2 s^2 + 6 s + 6

Continuous-time transfer function.

fx >> |
  
```

Figura 37. Función resultante.

Una vez realizado este cambio habremos completado el primer ejercicio.

## Ejercicio 2:

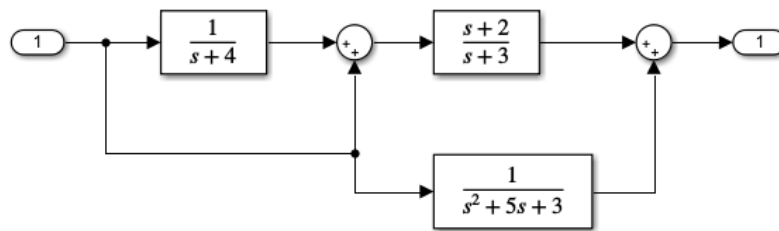


Figura 38. Sistema retroalimentado.

Para este ejemplo utilizaremos tres funciones de transferencia, dos operadores, una entrada y una salida que acomodaremos similar a la siguiente Figura:

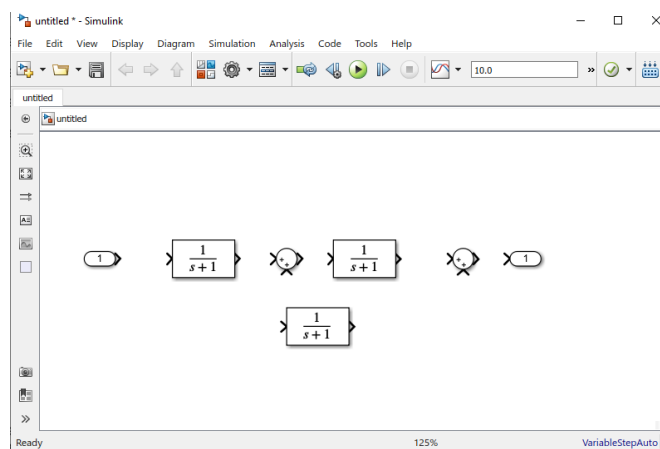


Figura 39. Componentes a utilizar.

Uniremos la línea principal de componentes desde la entrada hasta la salida.

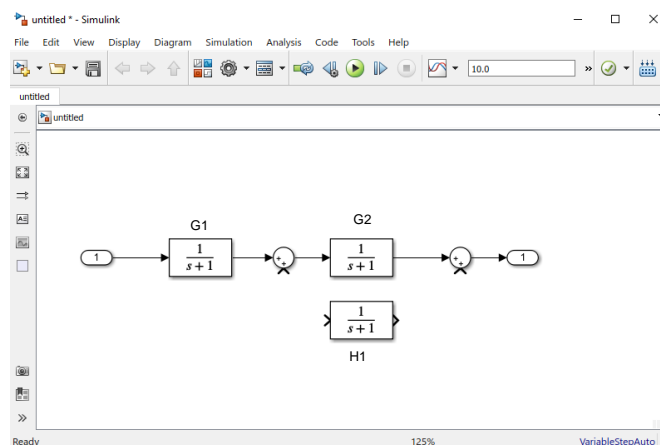


Figura 40. Unión y asignación de nombre a las funciones de transferencia.

A continuación, uniremos el primer operador a la línea que va de la entrada a G1, posteriormente uniremos H1 a la línea que acabamos de realizar, debe crearse un punto que marque la unión de ambas líneas y para finalizar nuestro diagrama uniremos H1 al segundo operador.

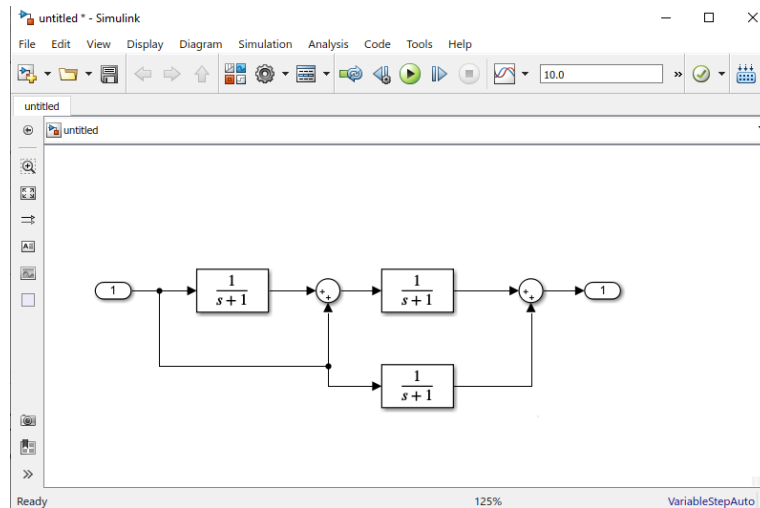


Figura 40 Unión del sistema.

El siguiente paso es modificar las funciones de transferencia:

- G1 Coeficientes denominador: [1 4]
- G2 Coeficientes numerador: [1 2], Coeficientes denominador: [1 3],
- H1 Coeficientes denominador: [1 5 3]

Quedando nuestro diagrama de bloques de la siguiente manera:

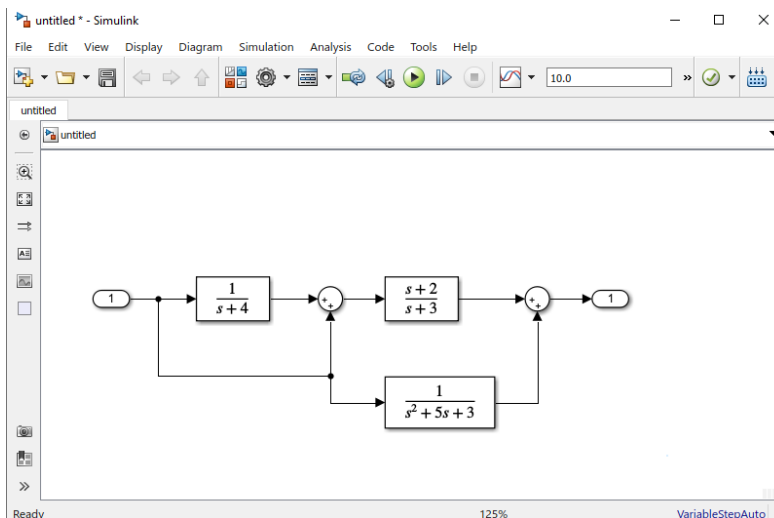


Figura 41. Diagrama de bloques finalizado.

Ahora guardamos el archivo de Simulink y sin cerrarlo volvemos a MATLAB.

En Command Window escribiremos el comando **[num,den]=linmod(' ')** , entre las comas colocamos el nombre del archivo de Simulink, y oprimimos Enter.

A continuación escribimos el comando **G=tf(num,den)** y obtenemos la función de transferencia resultante.

```

Command Window

>> [num,den]=linmod('DB2')

num =

    1.0000    12.0000    49.0000    78.0000    42.0000

den =

    1.0000    12.0000    50.0000    81.0000    36.0000

>> G=tf(num,den)

G =

      s^4 + 12 s^3 + 49 s^2 + 78 s + 42
      -----
      s^4 + 12 s^3 + 50 s^2 + 81 s + 36

Continuous-time transfer function.

fx >> |

```

Figura 42. Función resultante.

### Ejemplo 3:

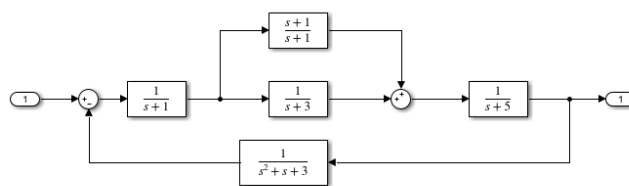


Figura 43. Sistema retroalimentado con dos operadores.

Para este sistema seleccionaremos dos sumadores y cinco funciones de transferencia sin olvidar agregar la entrada y la salida.

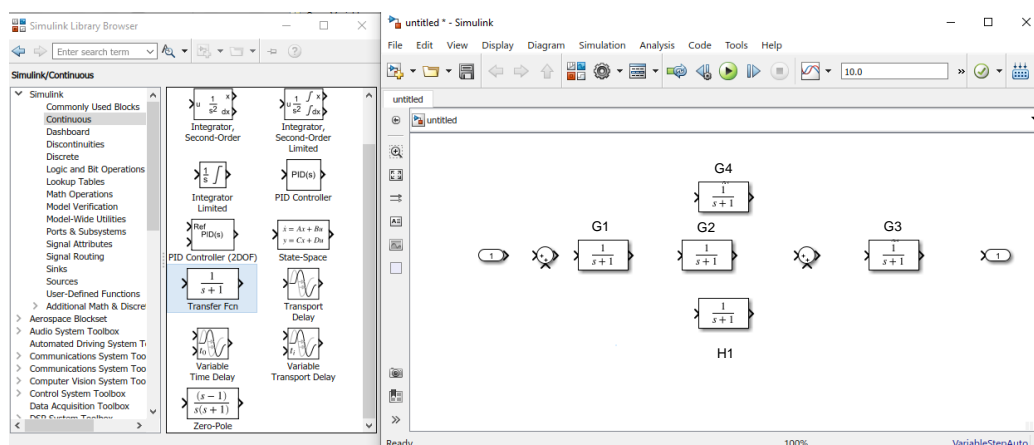


Figura 44. Componentes a utilizar.

En la *Figura 44* se asignó un nombre para cada función de transferencia con la finalidad de un mejor entendimiento a la hora de explicar las conexiones.

Lo primero que haremos es unir la línea de componentes desde la entrada hasta la salida.

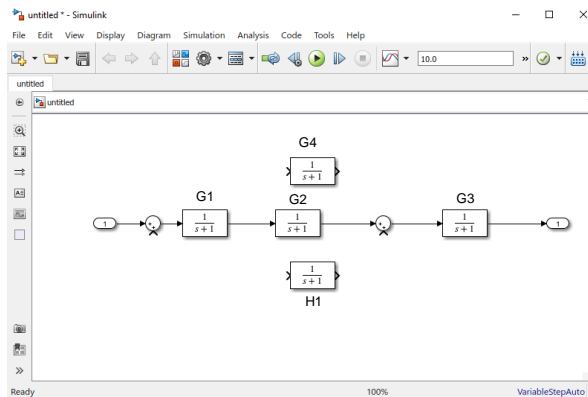


Figura 45. Unión de línea principal de componentes.

Ahora debemos cambiar la ubicación de los signos en el segundo operador, esto se logra escribiendo el símbolo de “barra vertical” (|), después de los signos a utilizar.

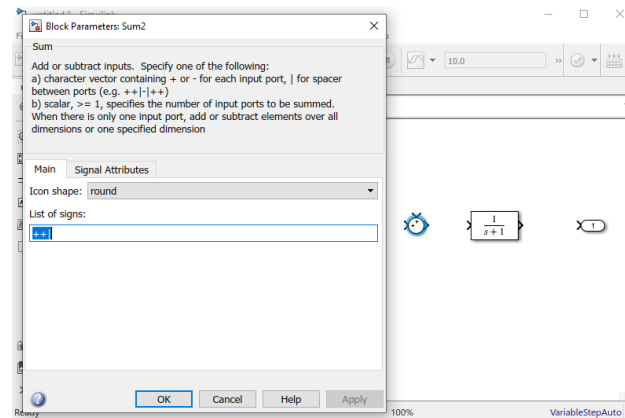


Figura 46. Cambio de posición de signos en el operador.

A continuación, partiendo de la línea de unión de G1 y G2 iremos a la función G4 y posteriormente a la suma del operador.

De la unión de G3 y la salida nos dirigiremos a H1 (invirtiendo su sentido previamente) y para cerrar el lazo nos dirigiremos al símbolo de suma del primer operador.

Por último, cambiaremos el signo del primer operador a menos.

Ahora es momento de editar las funciones de transferencia:

- G1 Coeficientes denominador: [1 1]
- G2 Coeficientes denominador: [1 3]
- G3 Coeficientes denominador: [1 5]
- G4 Coeficientes numerador: [1 1]
- H1 Coeficientes denominador: [1 1 3]

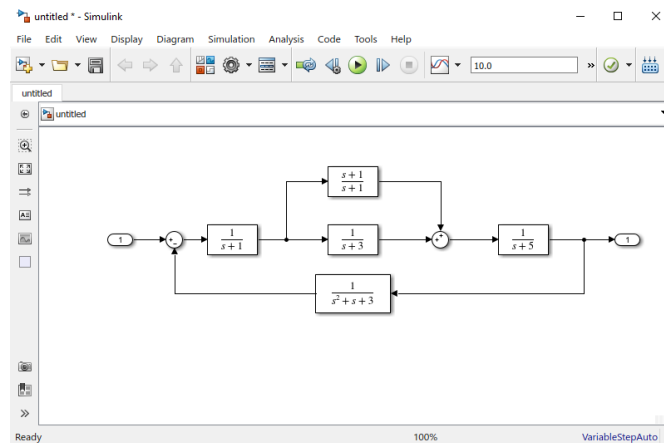


Figura 47. Diagrama finalizado.

El siguiente paso es guardar el archivo de Simulink y manteniéndolo abierto volver a MATLAB, a la ventana de comandos.

Escribiremos el comando **[num,den]=linmod(' ')** , entre las comas colocamos el nombre del archivo de Simulink, y oprimimos Enter, agregamos el comando **G=tf(num,den)** y obtenemos la función de transferencia resultante.

```

Command Window
>> [num,den]=linmod('DB3')

num =

    0         0    1.0000    6.0000   12.0000   19.0000   12.0000

den =

    1.0000   11.0000   45.0000  100.0000  150.0000  134.0000   49.0000

G=tf(num,den)

G =

          s^4 + 6 s^3 + 12 s^2 + 19 s + 12
-----
    s^6 + 11 s^5 + 45 s^4 + 100 s^3 + 150 s^2 + 134 s + 49

Continuous-time transfer function.

fx >> |

```

Figura 48. Función de transferencia resultante ordenada.

Con estos ejemplos finalizaría el tema de diagramas de bloques.

## Modelado se Sistemas en MATLAB Simulink.

En este apartado realizaremos ejemplos de modelos físicos con ayuda de Simulink.

### Ejemplo 1:

Simulación de llenado de un tanque.

$$\Delta Q = Q_i - Q_s$$
$$\frac{\Delta V}{\Delta T} = \frac{A \Delta h}{\Delta t} = A \frac{dh(t)}{dt}$$
$$A \frac{dh(t)}{dt} = Q_i - K\sqrt{h}$$

Despejando obtenemos el modelo siguiente:

$$\frac{dh(t)}{dt} = \frac{1}{A}(Q_i - K\sqrt{h(t)})$$

Considerando condiciones iniciales con valores de:

$$Q_i = 6 \text{ m}^3/\text{s}$$

$$A = 4 \text{ m}^2$$

$k = 3$  (pérdidas en el sistema).

Abrimos un nuevo documento de Simulink, utilizaremos los componentes Constant para el valor de k, Gain (2 componentes), Sum, Integrator, Scope para observar en una gráfica el comportamiento del sistema y Sqrt que se encuentra en la pestaña de Math Operations.

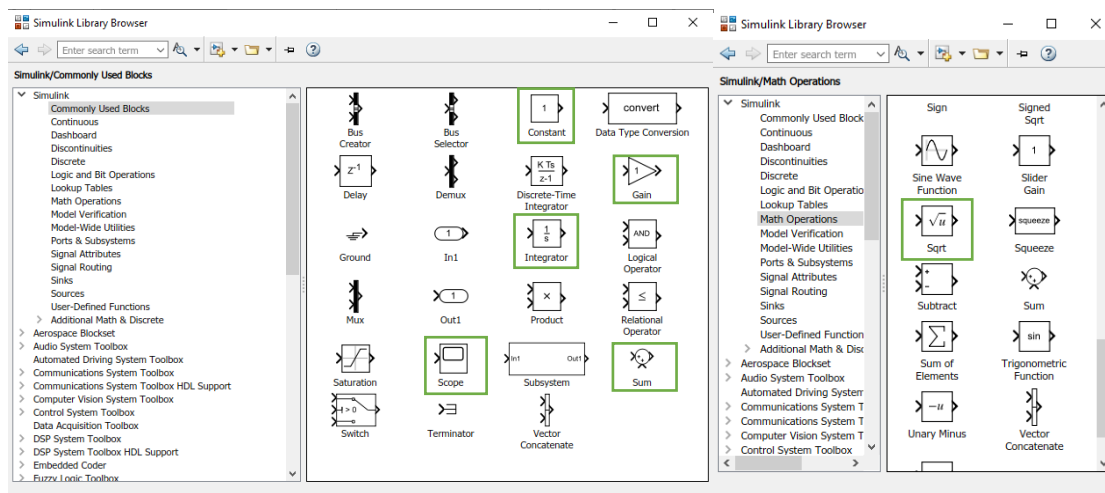


Figura 49. Componentes a utilizar.

Colocamos los componentes en un orden similar al de la *Figura 50*.



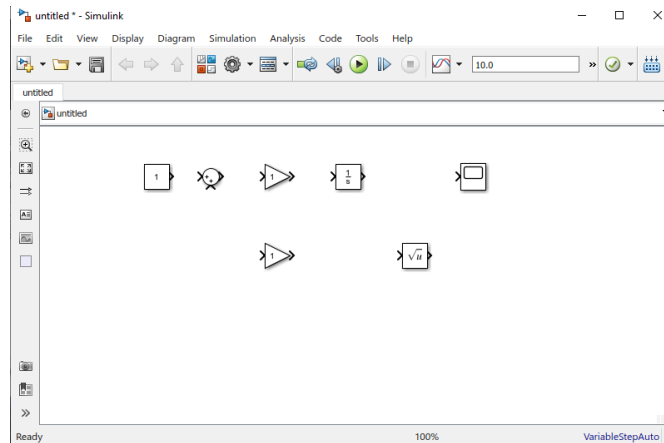


Figura 50. Componentes colocados.

Invertimos el sentido del Gain y el Sqrt. (Clic derecho, Rotate & Flip, Flip Block) y unimos todos los componentes. Por último cambiamos uno de los signos del Operador dando doble clic ( |+- ).

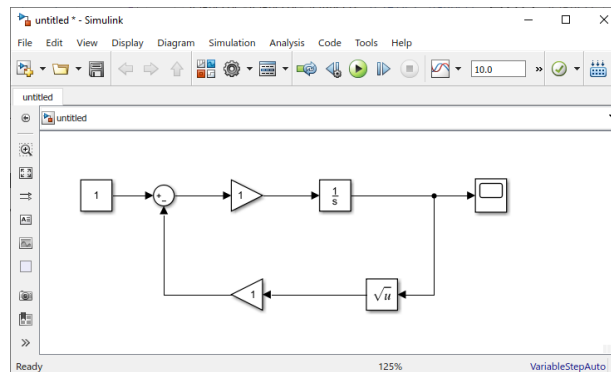


Figura 51. Sistema terminado.

Una vez que nuestro sistema esté colocado y unido correctamente, debemos cambiar los valores del caudal inicial, la constante (k) y el área.

Dando doble clic en el componente de constante cambiamos el valor por 6 (Caudal inicial) y presionamos OK.

Al primer Gain le asignamos el valor del área, como en la fórmula está debajo de un uno colocamos 1/4 y en el integrador colocamos el valor de la constante.

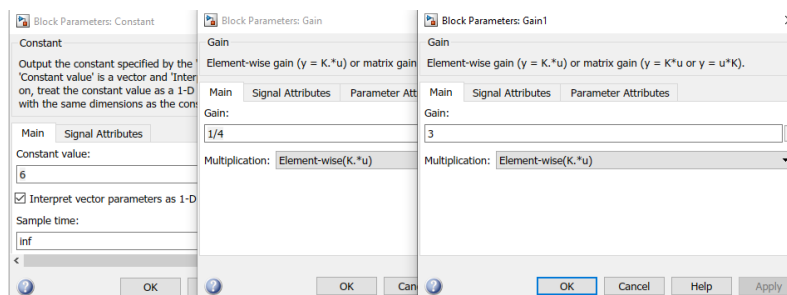


Figura 52. Asignación de valores.

Damos clic en el ícono verde de Run señalado en la Figura 53.



Figura 53. Menú para ejecutar la simulación y cambiar el tiempo.

Dar doble clic en Scope y se mostrará el comportamiento gráfico del sistema.

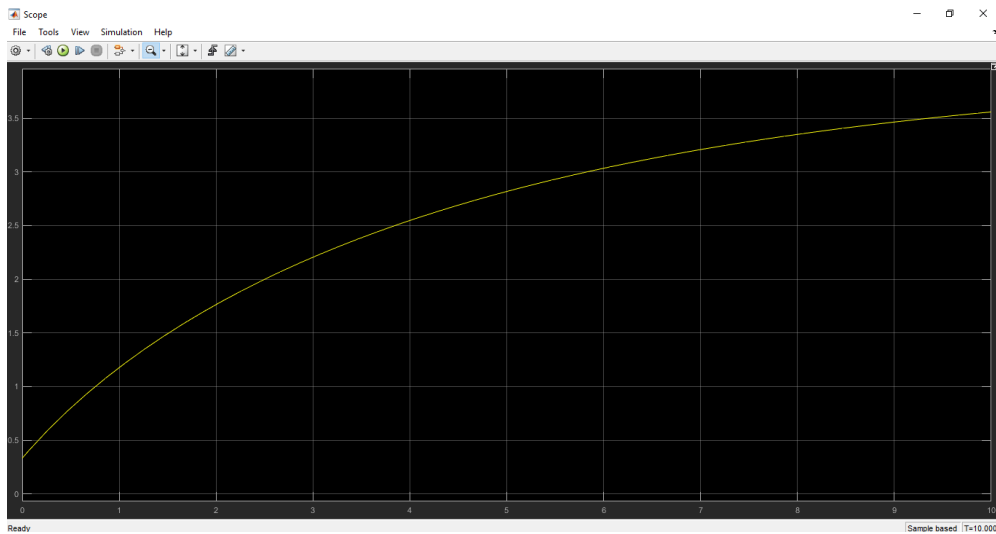


Figura 54. Comportamiento del sistema.

### Ejemplo 2:

Considerando un sistema masa fuerza amortiguamiento.

Sus ecuaciones son:

$$m x''(t) + b x'(t) + k x(t) = F(t)$$

$$x'' = 1/m * (F - b x' - k x)$$

En la pestaña de Commonly Used Blocks, seleccionaremos los componentes: Constant, Gain, Integrator (2) y Scope. De la pestaña de Math Operations seleccionaremos el componente Add.

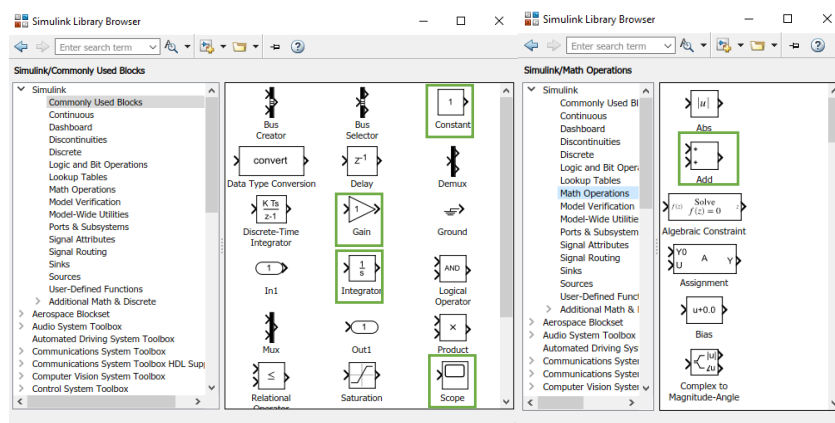


Figura 55 Componentes a utilizar señalados.

Lo primero que debemos hacer ahora es editar el componente Add, de manera que quede una terminal de suma y dos de resta, esto se logra con doble clic y cambiar por "+--". Podemos modificar la forma y tamaño con fines estéticos. Quedando de la siguiente manera:

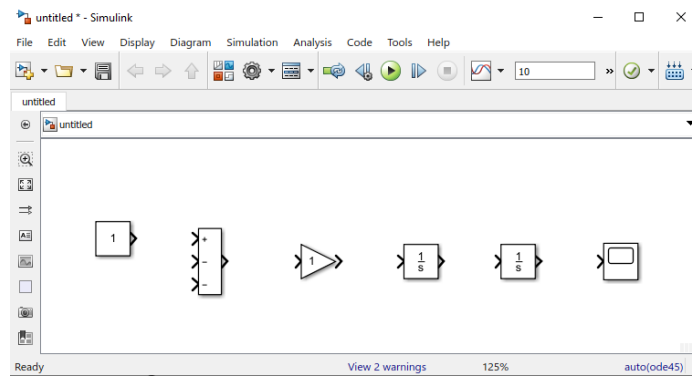


Figura 56. Componentes en Simulink.

Ahora uniremos la constante al signo de más del componente Add y de la salida de este nos dirigiremos a Gain que a su vez se conectará a los dos integradores y al Scope.

Podemos nombrar los componentes con la finalidad de comprender la analogía con las ecuaciones, esto se logra en el caso de Gain, dando clic en la palabra, Gain será  $1/m$ .

La línea que sale de Gain al integrador es la aceleración ( $x''$ ) para cambiar o asignar un nombre a la línea se debe dar doble clic, al integrar una vez la aceleración obtendremos la velocidad ( $x'$ ) y al integrar de nuevo obtendremos el desplazamiento ( $x$ ). Esto representado en Simulink queda de la siguiente manera:

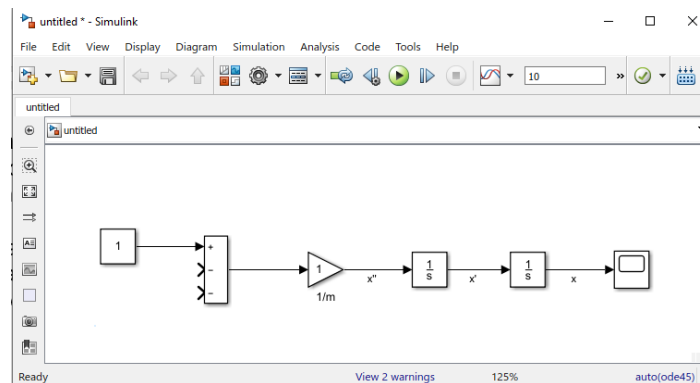


Figura 57. Primera parte de la ecuación.

De los signos restantes saldrán líneas de conexión, una hacia la velocidad y otra hacia el desplazamiento, posteriormente se colocarán dos nuevos componentes en las líneas recién creadas, estos se acomodarán automáticamente sin necesidad de rotarlos.

Las ganancias serán nuestras constantes  $k$  y  $b$ , asignamos los nombres. El sistema está ilustrado en la siguiente figura:

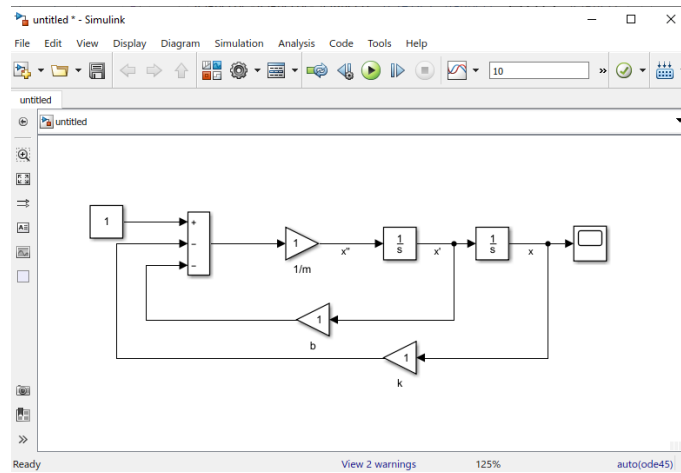


Figura 58. Modelo finalizado.

A continuación, al igual que el ejemplo anterior, damos clic en el icono de *Run*, y luego en *Scope* y podremos observar el comportamiento subamortiguado de este sistema cuando todas las constantes tienen un valor unitario.

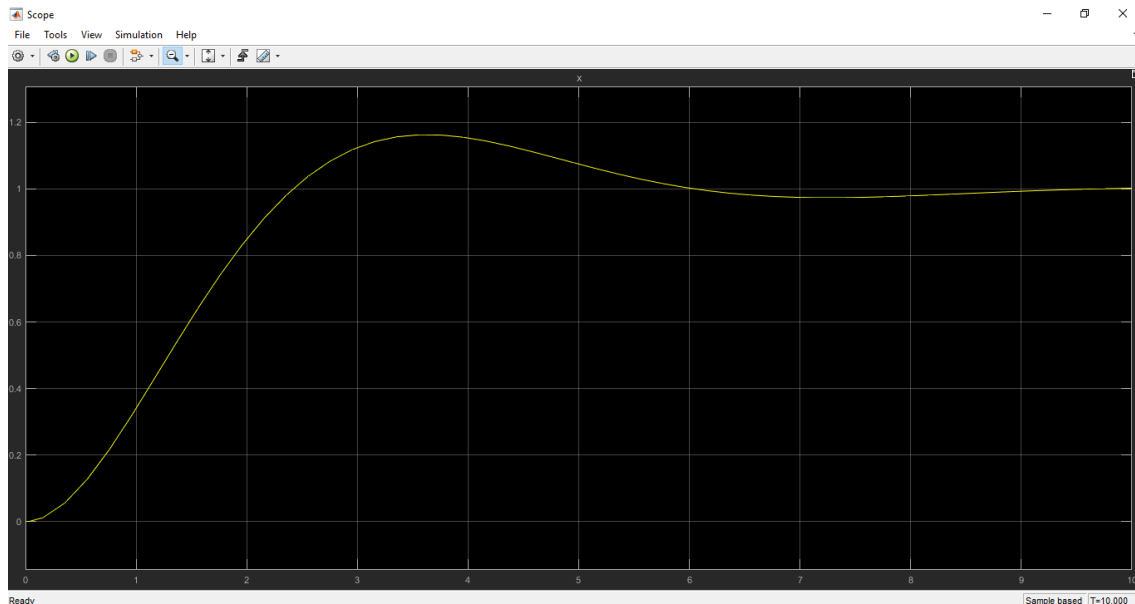


Figura 59. Comportamiento del sistema.

### Ejemplo 3:

Se pueden realizar modelos eléctricos también en Simulink. En este ejemplo utilizaremos un circuito de C.D. y calcularemos la corriente que circula en la resistencia de 5 Ohms.

Para ello necesitamos ir a la pestaña llamada Simscape en Simulink, aparecerán diversas opciones para trabajar con modelos mecánicos, eléctricos, electrónicos, hidráulicos, entre otros.

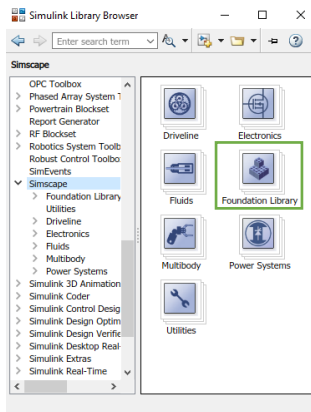


Figura 60. Elementos presentes en Simscape.

Ahora daremos clic en Foundation Library y posteriormente en la opción Electrical, aparecerán tres opciones.

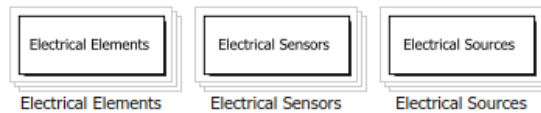


Figura 61. División de componentes eléctricos.

De la opción Electrical Sources obtendremos la fuente de alimentación de C.D., (para este circuito usaremos dos), y de la opción Electrical Elements obtendremos los componentes restantes necesarios para nuestro circuito.

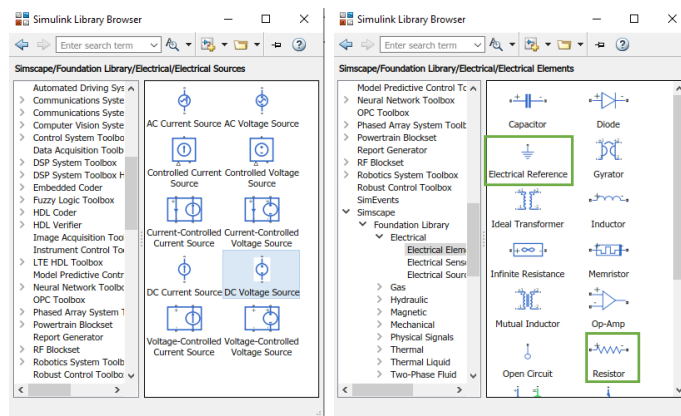


Figura 62. Elementos a ocupar.

Es necesario utilizar una referencia que es la puesta a tierra del circuito. Y se utilizarán 4 resistencias una de las cuales rotaremos 90°. A continuación, se muestran los componentes antes de conectarse.

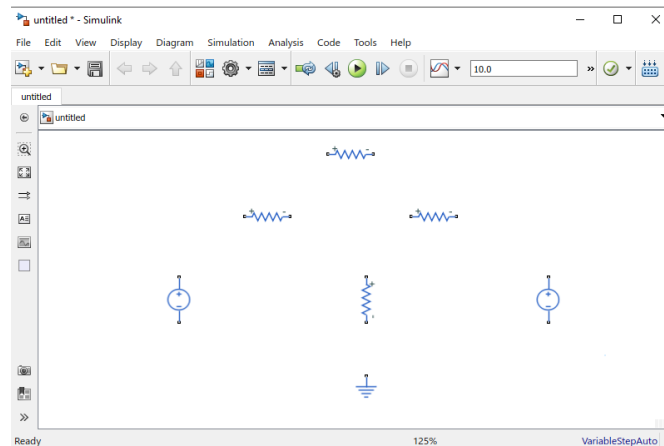


Figura 63. Componentes del circuito.

Se asignarán los valores de las resistencias y las fuentes de voltaje, eso dando doble clic sobre cada componente, de igual manera, se etiquetará cada componente con su valor, esto dando clic en el componente y posteriormente en el nombre.

NOTA: Dos componentes no pueden tener el mismo nombre, se sugiere utilizar el nombre “2 Ohms (1)” para la segunda resistencia igual.

Las resistencias tienen valores de 2 Ohms (dos de ellas), 5 Ohms y 3 Ohms, y las fuentes de 8 y 6 Volts. En la *Figura 64* se muestran estos pasos realizados.

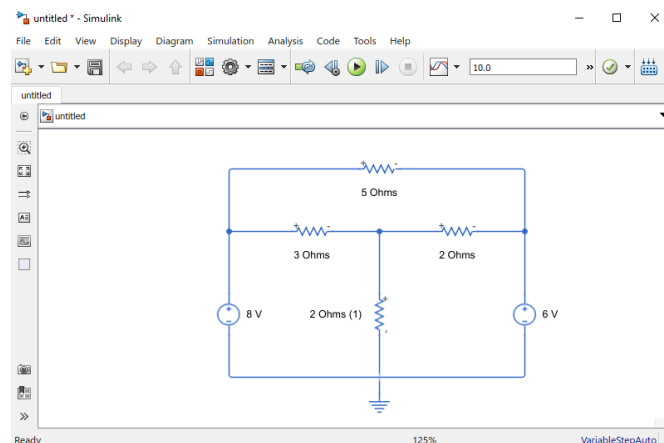


Figura 64. Circuito incluidos los valores de resistencias y fuentes.

Para que el circuito funcione la resistencia “2 Ohms” debe estar en sentido contrario por su polaridad, se modifica esto al sistema y continuamos.

Ahora debemos agregar un amperímetro que se encuentra en la opción de Electrical Sensors. Se conecta el amperímetro en serie al circuito para medir la corriente en la resistencia de 5 Ohms.

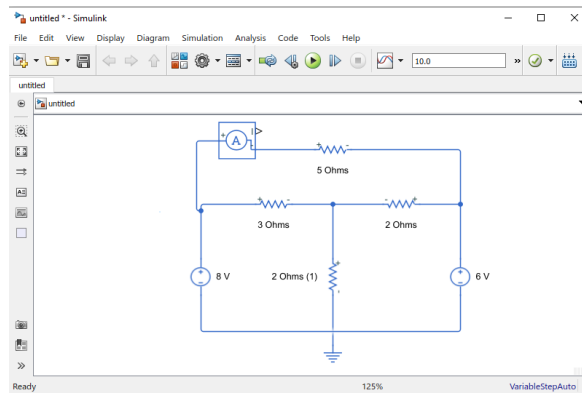


Figura 65. Circuito con el amperímetro conectado.

Para observar el valor de salida del amperímetro necesitaremos un elemento llamado Simulink Converter, que se encuentra en el menú Utilities dentro de Simscape, dentro de este menú también se encuentra el componente Solver Configuration, requisito que lo incluya nuestro sistema, también se necesita un display que se encuentra en el menú Sinks de Simulink.

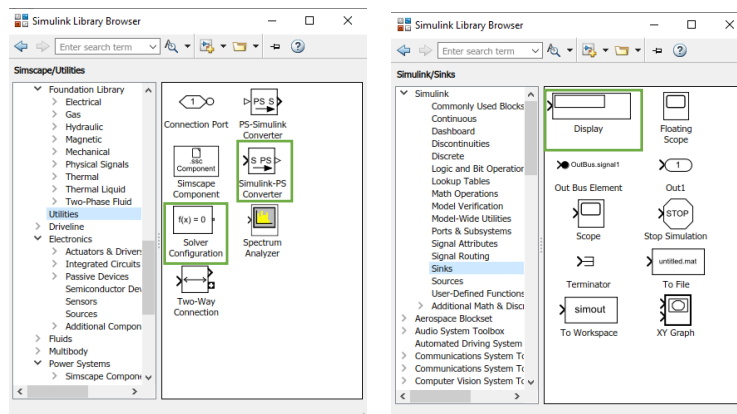


Figura 66. Componentes restantes de agregar.

El componente Solver Configuration puede conectarse a cualquier punto de nuestro circuito, sólo se debe verificar que se haya creado un nodo al conectar ya que de lo contrario no estará realmente conectado.

Por último, debemos cambiar el color del Display y del Amperímetro, esto dando clic derecho > Format > Background Color y elegimos un color a nuestro gusto.

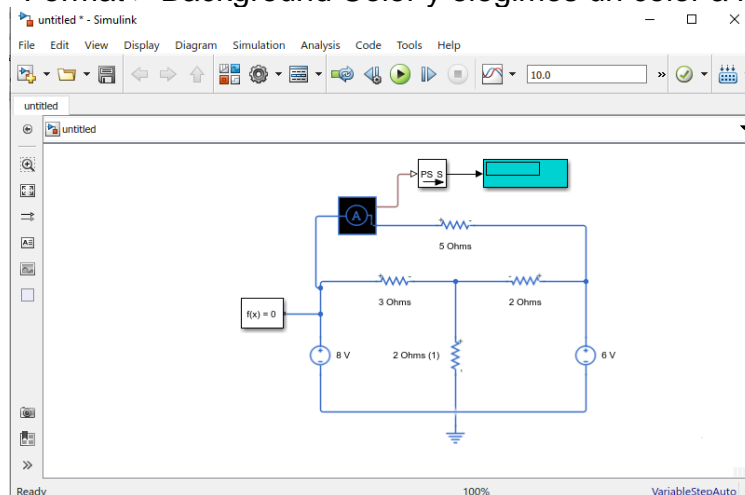


Figura 67. Circuito terminado.

Damos clic en el icono *Run* para observar el resultado en el display.

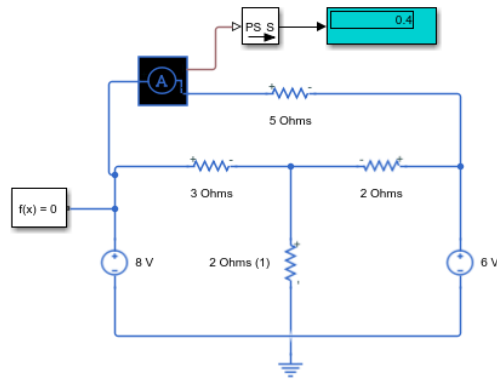


Figura 68. Respuesta obtenida del amperímetro.

Obtuvimos una corriente de 0.4 Amperes pasando por la resistencia de 5 Ohms. Así finalizaría nuestro segundo tema.



## Transformada de Laplace

### Ejemplo 1:

Para realizar una transformada de Laplace en MATLAB, podemos utilizar una serie de comandos preestablecidos.

La función a aplicar Laplace es:  $t^3 + 3$

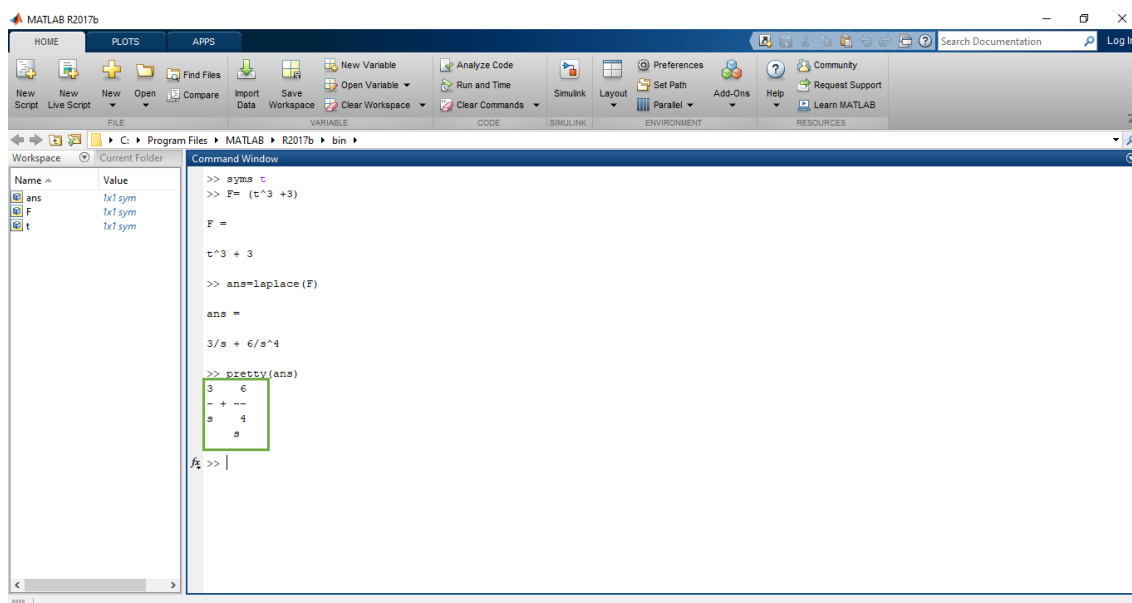
En la ventana Comand Window de MATLAB, escribiremos el código que nos permitirá realizar la transformada.

El primer paso es escribir el comando **syms t**, ya que está en el dominio del tiempo aun y dar Enter.

Lo siguiente es declarar la función, en este caso se declaró con la letra F, la manera correcta de escribirla es: **F = (t^3 + 3)**, el exponente debe escribirse con el símbolo “^”, al dar Enter MATLAB nos mostrará la función que introdujimos.

El comando necesario para realizar la transformada es **ans=laplace( )**, dentro del paréntesis escribiremos la literal que asignamos a la función, después de dar Enter aparecerá el resultado.

Por último, con la finalidad de ver de una manera más “clara” la respuesta podemos utilizar el comando **pretty(ans)**.



```
>> syms t
>> F = (t^3 + 3)
F =
t^3 + 3
>> ans=laplace(F)
ans =
3/s + 6/s^4
>> pretty(ans)
 3   6
- + --
 s  s^4
fs >> |
```

Figura 69. Transformada de Laplace.

Como podemos observar, la función resultante es:  $F(s) = \frac{3}{s} + \frac{6}{s^4}$

### Ejemplo 2:

La función que utilizaremos es la siguiente:  $\frac{e^{-3t} \sin(2t)}{t}$

Repetiremos los pasos del ejemplo anterior, lo primero es introducir el comando **syms t**, posteriormente declaramos la función.

Para escribir la función correctamente debemos declarar el Número de Euler como “exp”, utilizar el asterisco (\*) para realizar multiplicaciones y la barra inclinada (/) para indicar una división.

Dicho esto, procederemos a escribir la función, que queda de la siguiente manera:

$$f = \exp(-3*t)*\sin(2*t)/t$$

Después de dar Enter introducimos el comando **ans=laplace( )**, y posteriormente el comando **pretty(ans)**, opcional.

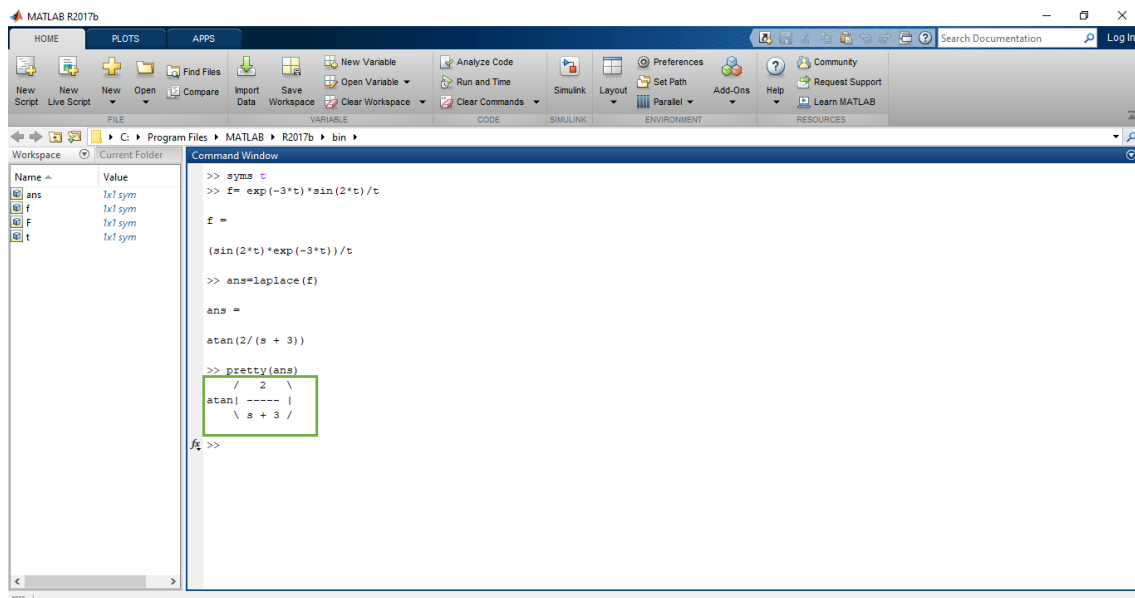


Figura 70. Segundo ejemplo transformada de Laplace.

La función resultante es:  $f(s) = \tan^{-1}\left(\frac{2}{s+3}\right)$

### Ejemplo 3:

La función a transformar será  $f(t) = \frac{1}{4} - \frac{1}{4} \cos 2t$

Escribimos el comando **syms t**, declaramos la función como:  $f = 1/4 - (1/4 * \cos(2*t))$

Ahora escribimos el comando encargado de realizar la transformada: **ans=laplace( )**, escribiendo entre paréntesis, en este caso, la letra f.

Por último, para visualizar mejor el resultado, utilizamos el comando **pretty(ans)**.

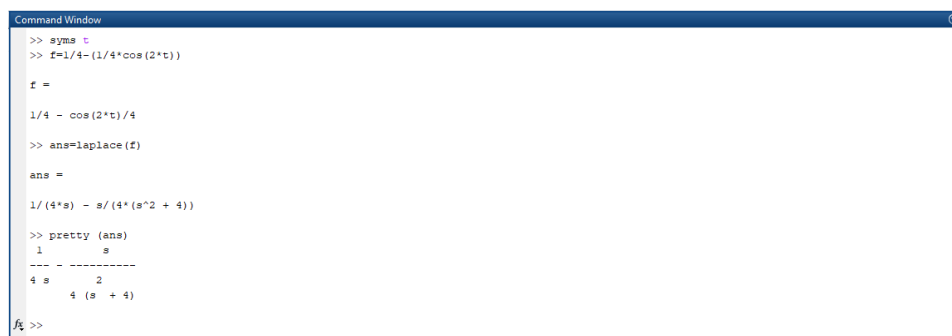


Figura 71. Resultante.

La función resultante es:  $f(s) = \frac{1}{4s} + \frac{s}{4(s^2+4)}$

## Anti transformada de Laplace

### Ejemplo 1:

La función a la que le aplicaremos Laplace inverso es:  $\frac{1}{s^2+3s+2}$

Utilizaremos el mismo comando que al aplicar la transformada, pero cambiando la t por la s. El comando es **syms s**.

Declaramos la función como  $f=1/(s^2+3*s+2)$ , damos Enter e introducimos el comando **ans=ilaplace( )**, entre paréntesis la variable asignada a la función.

En este caso no es necesario utilizar ningún otro comando.



```
Command Window
>> syms s
>> f=1/(s^2+3*s+2)

f =

1/(s^2 + 3*s + 2)

>> ans=ilaplace(f)

ans =

exp(-t) - exp(-2*t)

f/ >>
```

Figura 72. Anti transformada de Laplace de una función.

Función obtenida:  $f(s) = e^{-t} - e^{-2t}$

### Ejemplo 2:

Para este ejemplo, retomaremos el ejemplo 1 del tema de transformada de Laplace, con la finalidad de comprobar la veracidad del programa.

La función resultante del ejemplo uno fue:  $F(s) = \frac{3}{s} + \frac{6}{s^4}$

Por lo tanto, nos dirigimos a la ventana de comandos, insertamos el comando **syms s**, posteriormente agregamos la función como:  $F=(3/s)+(6/s^4)$  y escribimos el comando **ans=ilaplace(F)**.

La anti transformada resultante es  $F(t) = t^3 + 3$ , de esta manera comprobamos que los resultados son correctos.



```
Command Window
>> syms s
>> F=(3/s)+(6/s^4)

F =

3/s + 6/s^4

>> ans=ilaplace(F)

ans =

t^3 + 3

f/ >> |
```

Figura 73. Comprobación de resultados.

### Ejemplo 3:

La función a transformar será  $f(s) = \left( \frac{s+3}{s^2+6s+2} \right)$

De la misma manera que en los anteriores ejemplos, escribimos el comando **syms s** y declaramos la función como  $f = ((s+3)/(s^2+6*s+2))$

Por último escribimos el comando **ans=ilaplace( )**, insertando la variable asignada a la función y observamos el resultado.

```
Command Window
>> syms s
>> f = ((s+3)/(s^2+6*s+2))
f =
(s + 3)/(s^2 + 6*s + 2)
>> ans = ilaplace(f)
ans =
exp(-3*t)*cosh(7^(1/2)*t)
f >>
```

Figura 74. Resultante de la anti transformada de Laplace.

Función resultante:  $f(t) = e^{-3t} \cdot \cosh\left(7^{\frac{1}{2}}t\right)$

Con este ejemplo finalizaríamos los temas de Transformada y Anti transformada de Laplace.

## Fracciones Parciales

A continuación, se presentarán tres casos diferentes de fracciones parciales para resolver funciones.

### Ejemplo 1:

Para resolver fracciones parciales en Matlab debemos escribir los coeficientes del numerador y el denominador.

Si tenemos la fracción  $\frac{s+4}{s^2+7s+12}$

Debemos escribir los coeficientes del numerador entre corchetes “[ ]” y separados entre comas, de lo contrario MATLAB marcará error.

Quedará de la siguiente manera: num=[3,4]

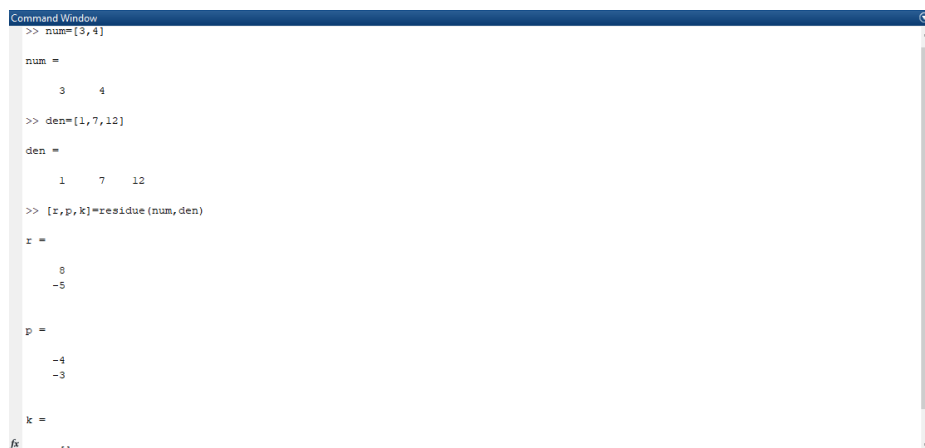
De igual manera, debemos escribir los coeficientes del denominador.

Quedará de la siguiente manera: den=[1,7,12]

Ahora utilizamos el comando **[r,p,k]=residue(num,den)**

La r significa las variables del numerador, la p significa las variables del denominador y la k son las variables independientes.

Al dar Enter, nos arrojará los resultados que acomodaremos en las fracciones.



```
Command Window
>> num=[3,4]
num =
     3     4
>> den=[1,7,12]
den =
     1     7    12
>> [r,p,k]=residue(num,den)
r =
     8
    -5
p =
    -4
    -3
k =
     0
```

Figura 75. Código para obtener resultado de fracciones parciales.

Podemos observar que no hay ningún valor en k, en p y r hay dos valores, lo que significa que tendremos dos fracciones. Así que acomodamos de la siguiente manera:

$$\frac{8}{(s+4)} + \frac{-5}{(s+3)}$$

Los valores de “p” irán acompañados de una “s” y deben cambiarse por el signo contrario al que tiene el resultado.

Se cambian porque se consideran  $s-4=0$ , entonces  $s=4$ .

### Ejemplo 2:

Considerando la fracción  $\frac{s^4-2s^2+4s+1}{s^3-s^2-s+1}$

En este caso al escribir los componentes del numerador, como no hay una  $s^3$ , se debe escribir un 0 respetando su espacio. Quedando de la siguiente manera:  
num=[1,0,-2,4,1]

En la fracción del denominador no es necesario hacer esto, el comando queda de la siguiente manera: den=[1,-1,-1,1]

Introducimos estos valores en MATLAB y tecleamos el comando **[r,p,k]=residue(num,den)**

```
Command Window
>> num=[1,0,-2,4,1]
num =
     1     0    -2     4     1
>> den=[1,-1,-1,1]
den =
     1    -1    -1     1
>> [r,p,k]=residue(num,den)
r =
     1.0000
     2.0000
    -1.0000
p =
     1.0000
     1.0000
    -1.0000
k =
     1     1
fx >>
```

Figura 76. Resultado cuando el valor de k no es cero.

En este caso, la factorización del denominador contiene factores lineales repetidos, por lo tanto, el segundo término se eleva al cuadrado. El resultado queda de la siguiente manera.

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

### Ejemplo 3:

Tenemos la fracción:  $\frac{s^3+5s^2+9s+7}{s^2+3s+2}$

Para esta escribiremos los componentes de la siguiente manera: num=[1,5,9,7] y den=[1,3,2]

Después de dar Enter a ambos comandos, agregamos el comando **[r,p,k]=residue(num,den)** y analizamos los resultados arrojados para escribir las fracciones.

```
Command Window
>> num=[1,5,9,7]
num =
    1     5     9     7
>> den=[1,3,2]
den =
    1     3     2
>> [r,p,k]=residue(num,den)
r =
   -1
    2
p =
   -2
   -1
k =
    1     2
fx >>
```

Figura 77. Valores resultantes.

Resultante:  $(x + 2) + \frac{2}{(s+2)} - \frac{1}{s+1}$

## Sistemas de Primer Orden

### Ejemplo 1:

Se explicará una de las funciones más comunes y sencillas:  $G(s) = \frac{k}{\tau s + 1}$

En esta función se aprecian claramente los componentes. Para simular su respuesta en MATLAB necesitamos ingresar sus valores de la siguiente manera:

$k=1$ ,  $\tau=1$ ,  $\text{num}=k$ ,  $\text{den}=[\tau, 1]$ . Después de cada uno se debe colocar un punto y coma “,” y dar Enter.

Necesitaremos la función “*step*” para obtener la respuesta a la entrada del escalón unitario.

Utilizaremos el comando **t=[ ]**; para ingresar el intervalo de tiempo de nuestra simulación. Se deben escribir tres parámetros, en este caso utilizaremos los siguientes:  $t=[0:0.1:15]$

Donde:

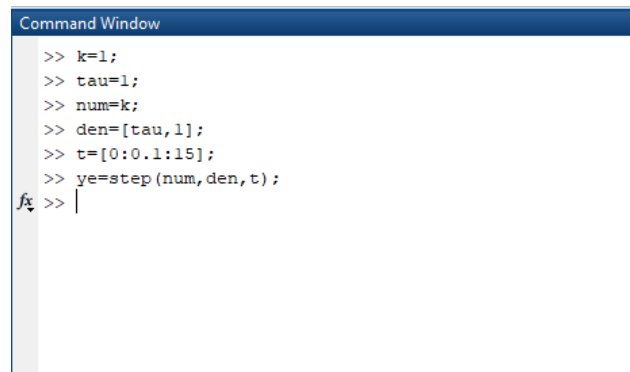
0= Valor en el que inicia la simulación.

0.1=Tiempo de respuesta de la simulación.

15= Tiempo en el que finaliza la simulación.

No debemos olvidar separar los valores por el símbolo “.” en vez de comas, ya que de lo contrario habrá un error al no ser valores reales en el vector tiempo.

Para utilizar la función *step* debemos ingresar el siguiente comando: **ye= step (num,den,t);**



```
Command Window
>> k=1;
>> tau=1;
>> num=k;
>> den=[tau,1];
>> t=[0:0.1:15];
>> ye=step(num,den,t);
fx >> |
```

Figura 78. Valores de la función declarados y comandos para la simulación.

Para visualizar la respuesta, debemos ingresar el comando **plot(t, ye)**, así se graficará respecto al tiempo.

Podemos agregar comandos para la edición de la gráfica como el comando **grid;** que agrega la cuadrícula, los comandos **xlabel(' ')** y **ylabel(' ')** para agregar un nombre a los ejes y el comando **title(' ')** utilizado para nombrar la gráfica.

Podemos nombrar nuestra gráfica como: Respuesta a una entrada escalón unitario. Quedando como se muestra:



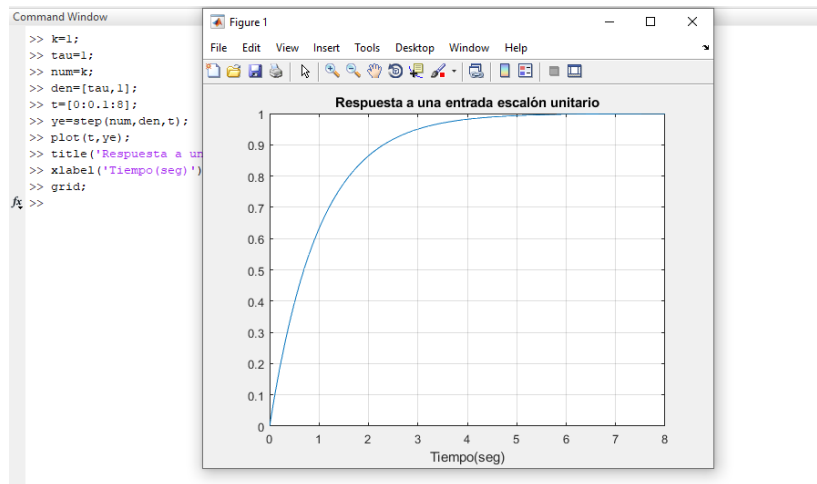


Figura 79. Gráfica de respuesta.

Recordemos que  $\tau$  es el tiempo que le toma a la señal alcanzar el 63% de la salida máxima, existen comandos para analizar el vector resultante para así comprobar la ganancia estática, para este primer ejemplo mostraremos todo el procedimiento de análisis de resultados.

Los comandos son los siguientes:

```

yRP=ye(length(ye));
k=yRP;
n=1;
while ye(n)<0.63*yRP;
n=n+1;
end;
tauEstim=0.1*(n-1);
fprintf('constante de tiempo:%f\n',tauEstim);

```

Donde se incluye una condición “while” y un valor de tau estimado conforme n aumenta. Después de ingresar los comandos anteriores obtendremos el valor del tiempo en que el sistema llega al 63%.

```

Command Window
>> k=1;
>> tau=1;
>> num=k;
>> den=[tau,1];
>> t=[0:0.1:8];
>> ye=step(num,den,t);
>> plot(t,ye);
>> title('Respuesta a una entrada escalón unitario')
>> xlabel('Tiempo(seg)')
>> grid;
>> yRP=ye(length(ye));
>> k=yRP;
>> n=1;
>> while ye(n)<0.63*yRP;
>> n=n+1;
>> end;
>> tauEstim=0.1*(n-1);
>> fprintf('constante de tiempo:%f\n',tauEstim);
constante de tiempo:1.000000
>>

```

Figura 80. Comprobación del valor de  $\tau$

Ahora se comprobará que el resultado coincide con lo teórico, la función en el dominio del tiempo es  $G(t) = 1 - e^{-t/\tau}$ , cuando  $t \geq 0$

Declaramos la función  $G(t)$ , como: `ye2=1-exp(-t/tau);`

Utilizamos el comando `plot` para graficar las dos funciones respecto al tiempo: **`plot(t, ye, t, ye2, "o")`**;

Se agrega el "o" para crear unos pequeños círculos en el gráfico.

Al igual que en la gráfica anterior podemos cambiar el nombre con **`title('')`** y agregar la cuadrícula con **`grid`**;

El gráfico obtenido es:

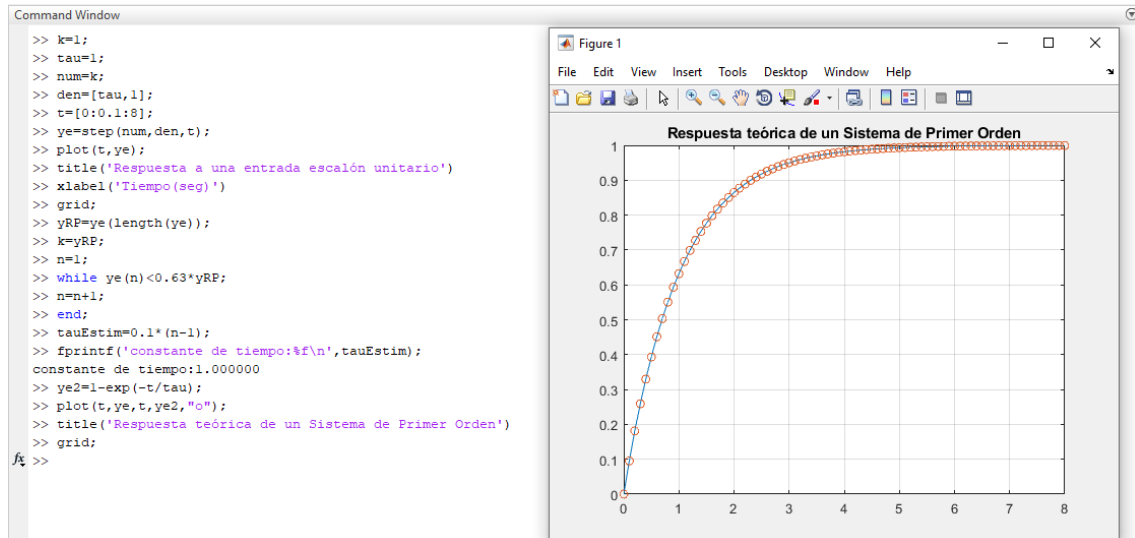


Figura 81. Respuesta teórica a escalón unitario.

Con esta gráfica podemos observar el comportamiento y cómo se superponen ambas gráficas.

### Ejemplo 2:

Tenemos la función:  $G(s) = \frac{30}{s+6}$ , el denominador debe de quedar en la forma  $\tau s + 1$ , por lo tanto, dividimos todo entre 6, quedando de la siguiente manera:

$$G(s) = \frac{5}{0.16s+1}, \text{ donde } \tau = 0.16 \text{ y } k = 5$$

Declaramos las variables en el código, agregando la información de que el numerador es  $k$  (`num=k;`) y el denominador es  $0.16s + 1$  (`den=[tau,1];`). Si escribiéramos `den=[.16tau,1]` sería incorrecto ya que el valor de  $\tau$  ya fue declarado previamente.

Declaramos los intervalos de tiempo: **t=[0:0.1:10]**; (sin olvidar agregar “;”) y ya que necesitamos una entrada escalón, utilizaremos el comando: **ye= step (num,den,t)**;

El código quedará de la siguiente manera:

```
Command Window
>> k=5;
>> tau=0.16;
>> num=k;
>> den=[tau,1];
>> t=[0:0.1:10];
>> ye= step (num,den,t);
fx >>
```

Figura 82. Declaración de valores.

Para observar la respuesta utilizamos el comando **plot(t,ye)**, modificamos en título con el comando “**title**”, la etiqueta del eje x para indicar que representa el tiempo y agregamos la cuadrícula para observar mejor el comportamiento del sistema.

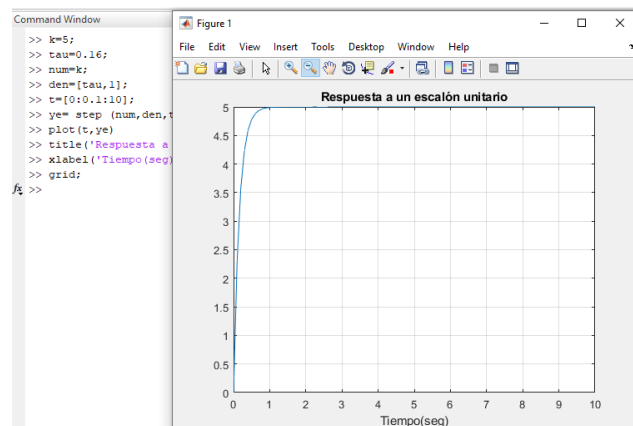


Figura 83. Respuesta del sistema graficada.

Ese es el comportamiento de nuestra función de transferencia recibiendo a la entrada una señal tipo escalón unitario.

### Ejemplo 3:

De igual manera, tenemos la función:  $G(s) = \frac{10}{0.4s+1}$ , esta función ya se encuentra en la forma  $\tau s + 1$ , procederemos a obtener los valores:  $\tau = 0.4$  y  $k = 5$

Declaramos los valores de la siguiente manera:

```
k=10;
tau=0.4;
num=k;
den=[tau,1];
```

Y agregamos el intervalo de tiempo  $t=[0:0.1:5]$ ;  
 Posteriormente ingresamos la señal de escalón unitario  $ye= \text{step}(\text{num},\text{den},t)$ ;  
 Utilizamos el comando “*plot*” para graficar la respuesta. Cambiando el título de la gráfica, el nombre del eje x y agregando la cuadrícula.

La gráfica obtenida es:

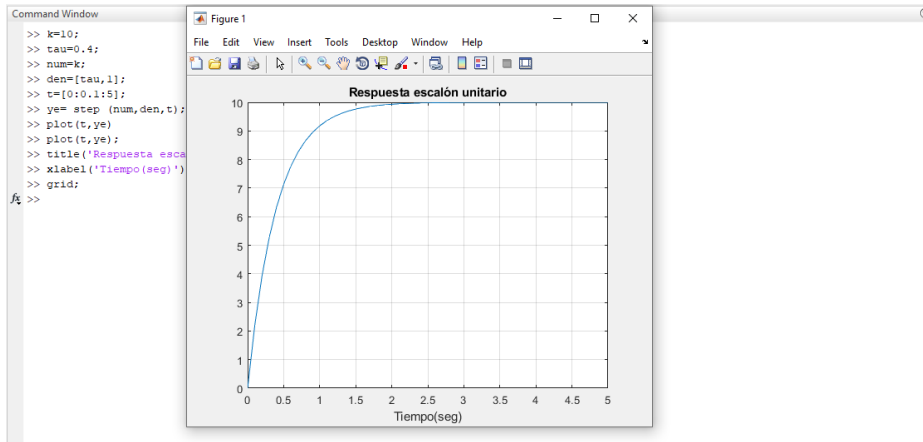


Figura 84. Respuesta de la función.

Con los comandos vistos en los temas anteriores, declaramos la función como  $G=10/((0.4*s)+1)$ , aplicamos la inversa de Laplace y posteriormente graficamos ambas funciones con respecto al tiempo obteniendo la siguiente gráfica:

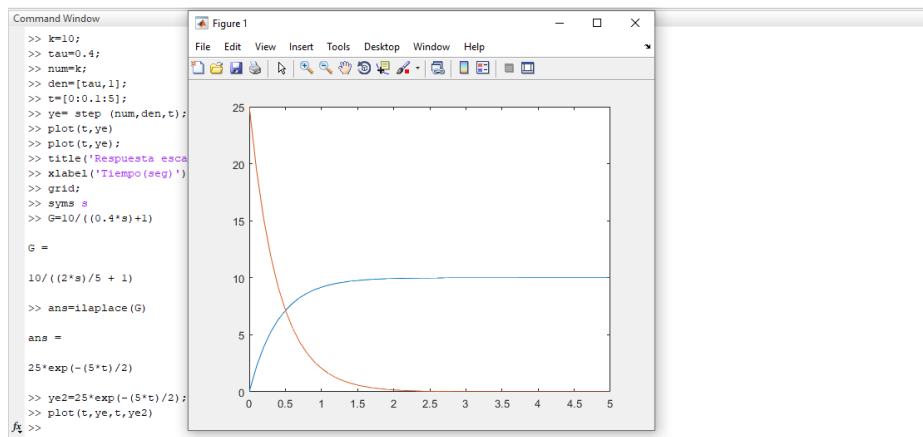


Figura 85. Respuesta teórica.

Con esto finalizamos el tema de Sistemas de Primer Orden.

## Sistemas de Segundo Orden

### Ejemplo1:

$$\text{Función a utilizar: } G2(S) = \frac{0.2s^2 + 0.3s + 1}{(s + 0.5)(s^2 + 0.4s + 1)}$$

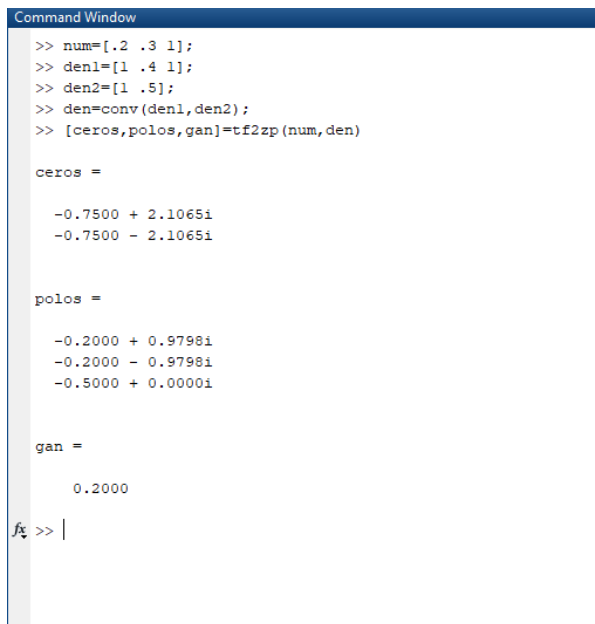
En la ventana de comandos debemos escribir los componentes del numerador y del denominador, en el caso del denominador se utilizarán “den1” y “den2” para introducir los valores, ésto entre corchetes y sin signos que los separen, posteriormente se utilizará el comando **den=conv(den1,den2)**; para unificar.

Quedarán de la siguiente manera:

```
num=[.2 .3 1];  
den1=[1 .4 1];  
den2=[1 .5];  
den=conv(den1,den2);
```

Utilizando el comando **[ceros,polos,gan]=tf2zp(num,den)** podremos obtener un vector que contiene los ceros de la función de transferencia, un vector que contiene los polos y un escalar que indica al ganancia estática.

Añadiendo este código obtendremos los siguientes valores:



```
Command Window  
>> num=[.2 .3 1];  
>> den1=[1 .4 1];  
>> den2=[1 .5];  
>> den=conv(den1,den2);  
>> [ceros,polos,gan]=tf2zp(num,den)  
  
ceros =  
  
    -0.7500 + 2.1065i  
    -0.7500 - 2.1065i  
  
polos =  
  
    -0.2000 + 0.9798i  
    -0.2000 - 0.9798i  
    -0.5000 + 0.0000i  
  
gan =  
  
    0.2000  
  
fx >> |
```

Figura 86. Valores de polos, ceros y ganancia.

Recordemos que la respuesta a la entrada escalón unitario se obtiene con la función “step”, definimos un intervalo de tiempo:

```
t2=[0:0.2:20];
```

```
Definimos la función: y2e=step(num,den,t2);
```

```
Graficamos con respecto al tiempo, plot(t2,y2e);
```

```
Cambiamos el nombre de la gráfica y del eje x y agregamos la cuadrícula.
```

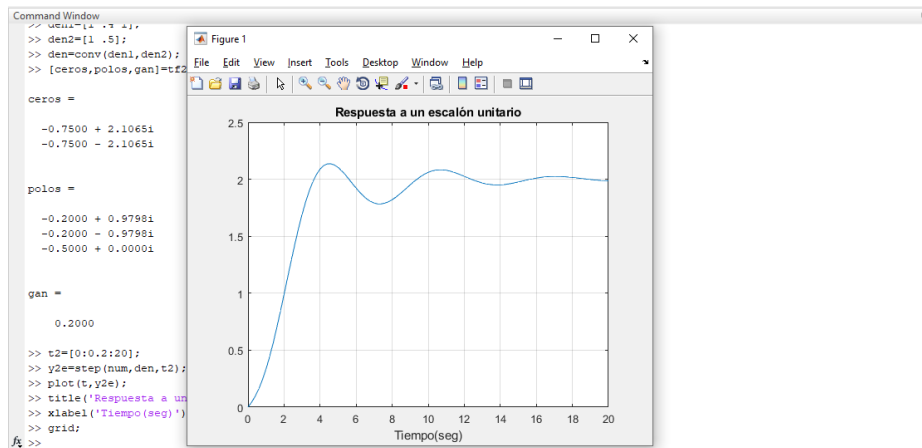


Figura 87. Respuesta de un sistema de Segundo Orden.

Al igual que en el tema de Sistemas de Primer Orden, en este tema ahondaremos en un ejemplo para visualizar nuevas funciones y comandos.

Los sistemas entregan una respuesta en frecuencia, la cual puede ser analizada mediante diagramas de Bode, para obtener estos necesitamos los siguientes comandos:

```
[mag,phase,w]=bode(num,den);
subplot(211), loglog(w,mag), title('Magnitud'),xlabel('rad/s');
subplot(212), semilogx(w,phase), title('Fase'),xlabel('rad/s');
```

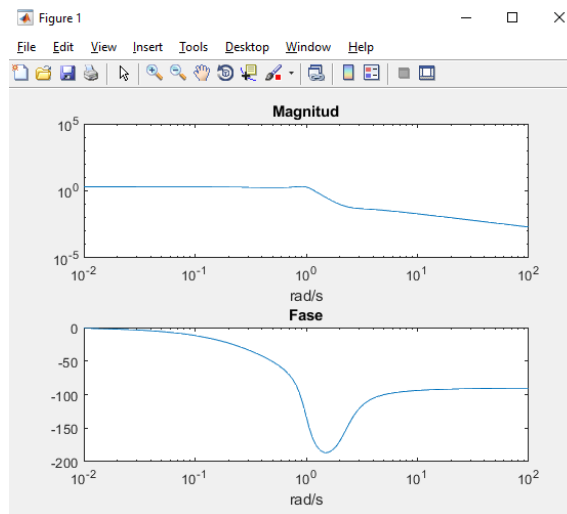


Figura 88. Diagramas de Bode.

Una función de transferencia se puede representar de manera paramétrica mediante el diagrama de Nyquist, este calcula la parte real y la imaginaria de la función  $G(j\omega)$  y sin parámetros de salida grafica estos valores. Para realizar un diagrama de Nyquist debemos utilizar los siguientes comandos:

```
[re,im]=nyquist(num,den,w);
plot(re,im,re,-im,'r');
```

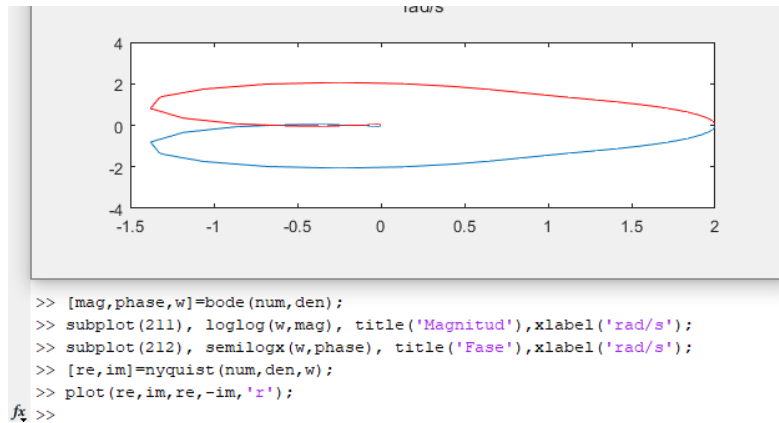


Figura 89. Diagrama de Nyquist.

Por último, obtendremos los márgenes de fase y de ganancia con el comando:  
**[mg,mf,wmg,wmf]=margin(num,den)**



Figura 90. Margen de ganancia y fase.

### Ejemplo 2:

Utilizando la función:  $G2(S) = \frac{10s^2+4s+1}{4s^2+2s+1}$

Obtenemos los componentes del numerador y el denominador: **num=[10 4 1];**  
**den=[4 2 1];**

E ingresamos el comando: **[ceros,polos,gan]=tf2zp(num,den)** para obtener los ceros, polos y la ganancia del sistema.

A continuación declaramos los intervalos de tiempo, utilizaremos la variable **t2=[0:0.2:20];**

Utilizaremos una entrada escalón unitario, con el comando "step", por lo tanto, definimos la función: **y2e=step(num,den,t2);**

Agregamos el comando "plot" para graficar con respecto al tiempo: **plot(t2,y2e);**  
Cambiamos el nombre de la gráfica, del eje x con el comando **xlabel('Tiempo(seg)');** y agregamos la cuadrícula.

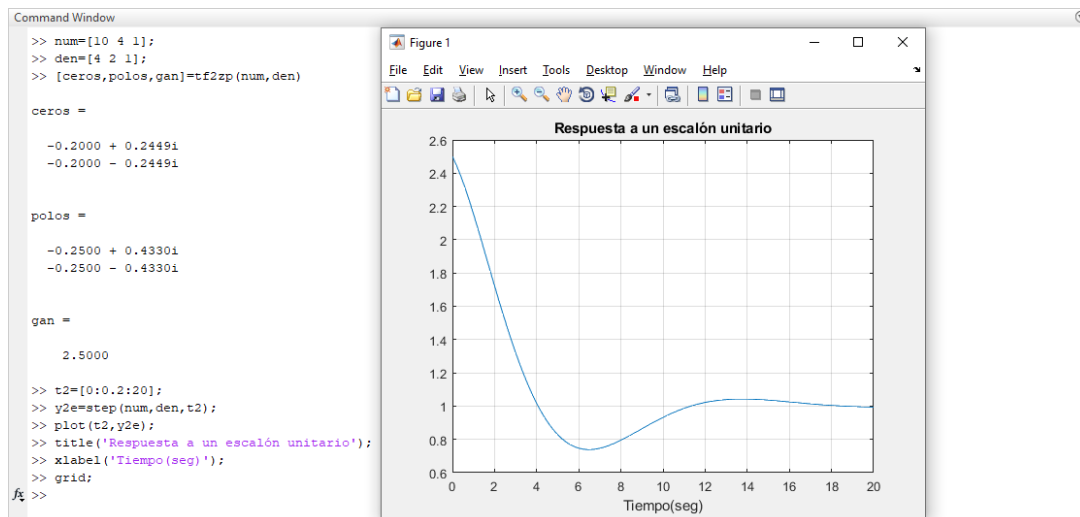


Figura 91. Ceros, polos y ganancia del sistema con su gráfica de respuesta.

Al igual que en primer ejemplo, obtendremos los diagramas de Bode con los siguientes comandos:

```

[mag,phase,w]=bode(num,den);
subplot(211), loglog(w,mag), title('Magnitud'),xlabel('rad/s');
subplot(212), semilogx(w,phase), title('Fase'),xlabel('rad/s');
  
```

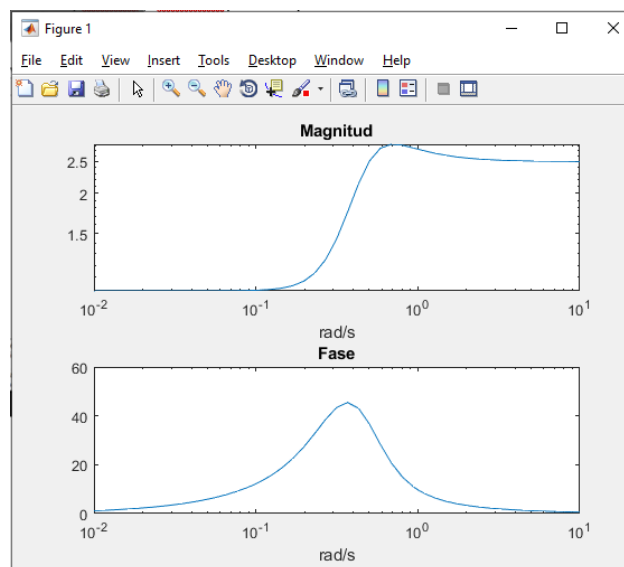


Figura 92. Diagramas Magnitud y Fase.

De igual manera utilizamos los siguientes comandos para obtener el diagrama de Nyquist:

```

[re,im]=nyquist(num,den,w);
plot(re,im,re,-im,'r');
title('Diagrama Nyquist') , este solamente para cambiar el nombre.
  
```



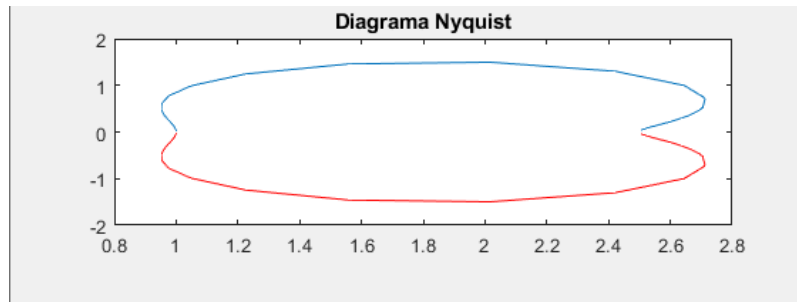


Figura 93. Diagrama Nyquist.

Para finalizar este ejemplo, obtendremos los márgenes de fase y de ganancia con el comando: **[mg,mf,wmg,wmf]=margin(num,den)**

```

Command Window
>> [mag,phase,w]=bode(num,den);
>> subplot(211), loglog(w,mag), title('Magnitud'),xlabel('rad/s');
>> subplot(212), semilogx(w,phase), title('Fase'),xlabel('rad/s');
>> [re,im]=nyquist(num,den,w);
>> plot(re,im,re,-im,'r');
>> title('Diagrama Nyquist')
>> [mg,mf,wmg,wmf]=margin(num,den)

mg =
    Inf

mf =
   -180

wmg =
    NaN

wmf =
     0

fc >>
  
```

Figura 94. Fase y ganancia del sistema.

### Ejemplo 3:

Considerando la función de transferencia obtenida de un sistema resorte-masa-amortiguador:

$$\frac{X(s)}{F(s)} = \frac{1}{Ms^2 + Ds + K}$$

Datos:

D= 0.25 Ns/m

K=1/M

M=0.0625 kg.

f(t)=1 N

Dividiendo entre M los términos de la función de transferencia y sustituyendo valores:

$$\frac{\frac{1}{M}}{\frac{M}{M}s^2 + \frac{D}{M}s + \frac{K}{M}} = \frac{\frac{1}{0.0625}}{s^2 + \frac{0.25}{0.0625}s + \frac{1}{0.0625}} = \frac{16}{s^2 + 4s + 16}$$

A continuación, declararemos los valores del numerador y el denominador como se muestra:

**num=[0 0 16]; y den=[1 4 16];**

Debemos ingresar el comando: **[ceros, polos, gan]=tf2zp(num, den)** para obtener los valores mencionados.

Declararemos los intervalos de tiempo como: **t2=[0:0.1:10];**

Para ingresar una entrada escalón unitario ingresamos el comando: **y2e=step(num, den, t2);**

Graficamos con respecto al tiempo: **plot(t2, y2e);** y al igual que en los ejemplos anteriores cambiamos el nombre de la gráfica, del eje x y agregamos la cuadrícula.

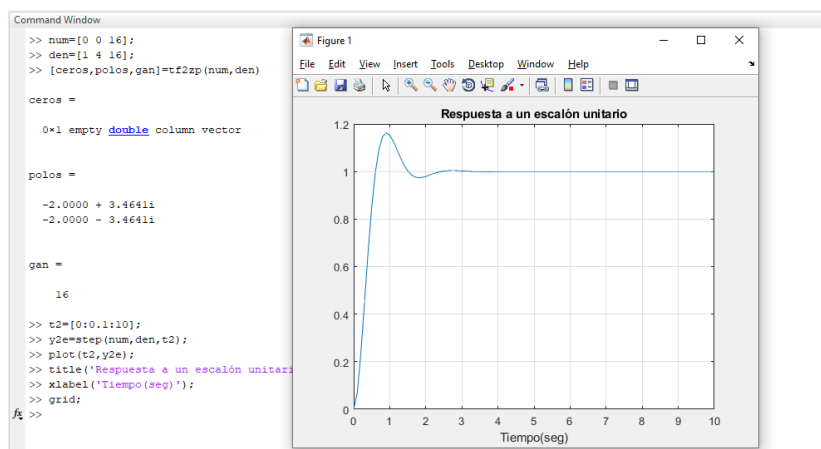


Figura 95. Respuesta del sistema a una entrada escalón unitario.

Obtenemos los diagramas de Bode con los comandos:

**[mag, phase, w]=bode(num, den);**

**subplot(211), loglog(w, mag), title('Magnitud'), xlabel('rad/s');**

**subplot(212), semilogx(w, phase), title('Fase'), xlabel('rad/s');**

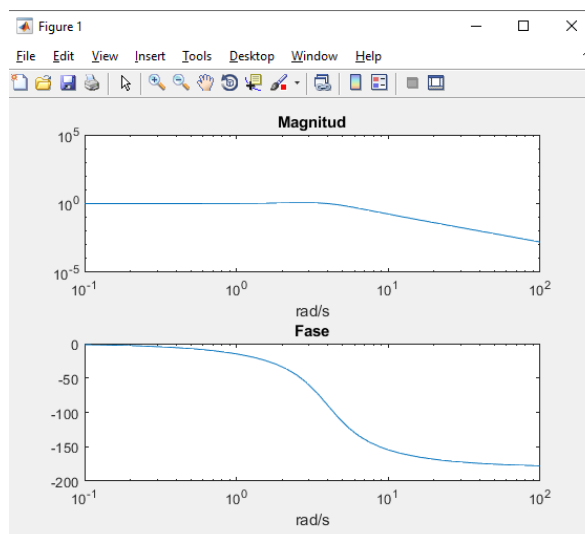


Figura 96. Diagramas de Bode resultantes.

Resumiendo, en un solo paso, obtendremos el diagrama de Nyquist y la ganancia y fase del sistema con los siguientes comandos:

```
[re,im]=nyquist(num,den,w);  
plot(re,im,re,-im,'r');  
title('Diagrama Nyquist')  
[mg,mf,wmg,wmf]=margin(num,den)
```

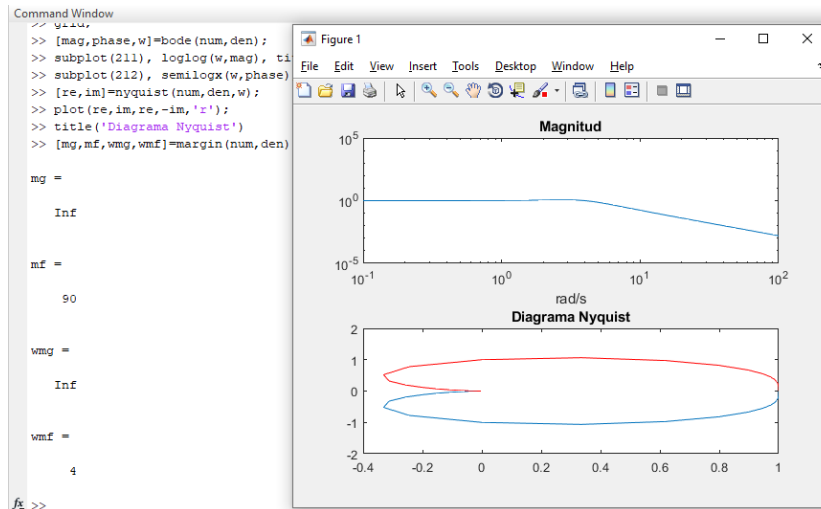


Figura 97. Diagrama de Nyquist y valores de ganancia y fase.

## Controlador P

El control proporcional, algunas veces llamado ganancia o sensibilidad, es una acción de control que reproduce cambios de la entrada con cambios en la salida. La acción proporcional del controlador responde a los cambios presentes en la entrada y generara inmediatamente y proporcionalmente cambios en la salida.

### Ejemplo 1:

Podemos observar la respuesta a un controlador utilizando la ventana de comandos de MATLAB. Suponiendo una función de transferencia:

$$H(s)=\frac{2}{0.4s^2+2}$$

Dividiendo entre 0.4 obtenemos  $H(s)=\frac{5}{s^2+5}$  , estos serán los valores que ingresaremos.

Suponiendo un valor de tau=0.3 procedemos a escribir nuestros comandos.

Para agregar la función utilizamos: **H= tf (5, [1 5]);**

Posteriormente escribimos **tau=0.1;**

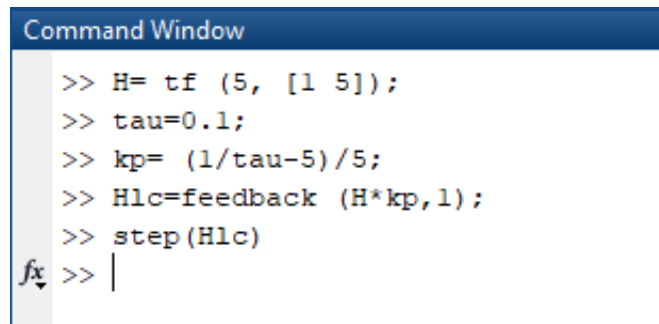
Y escribimos **kp= (1/tau-5)/5;**

Para obtener la respuesta en Lazo cerrado utilizamos el comando:

**Hlc=feedback (H\*kp,1);**

Para graficar esta respuesta utilizamos el comando “**step( )**”, que introducirá una señal escalón, dentro del paréntesis se colocará la variable “**Hlc**”.

Los comandos quedarán de la siguiente manera:



```
Command Window
>> H= tf (5, [1 5]);
>> tau=0.1;
>> kp= (1/tau-5)/5;
>> Hlc=feedback (H*kp,1);
>> step(Hlc)
fx >> |
```

Figura 98. Comandos Controlador P.

Y la gráfica será la siguiente:

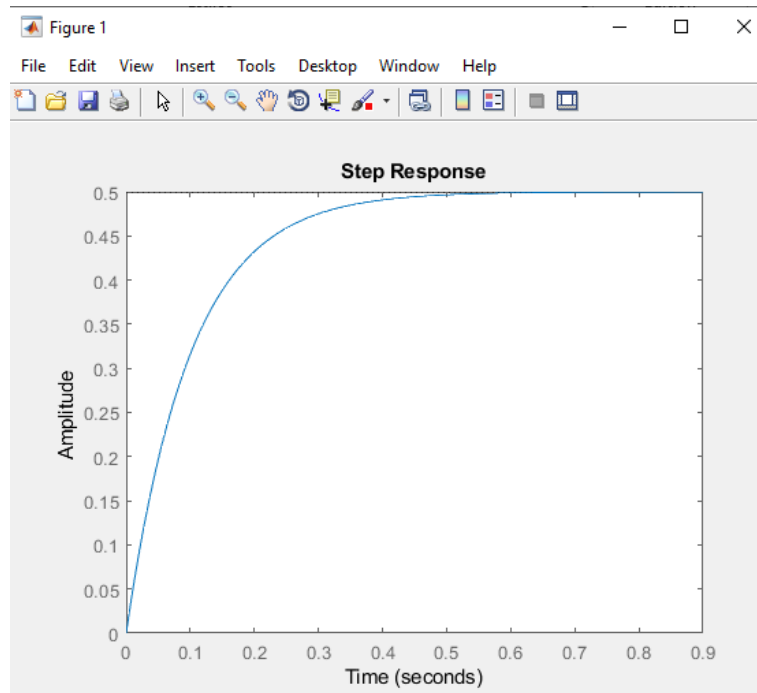


Figura 99. Respuesta control P.

A la cual, con clic derecho podemos agregar la cuadrícula y algunas funciones más, en este caso seleccionaremos el tiempo de respuesta, observando lo siguiente:

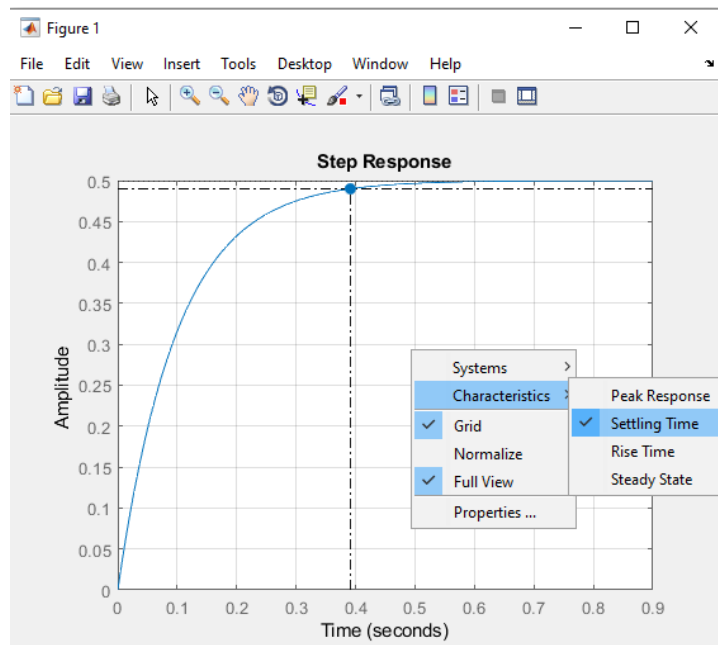


Figura 100. Tiempo de respuesta del sistema.

## Ejemplo 2:

Para este ejemplo utilizaremos la herramienta Simulink y comprobaremos que se obtiene la misma señal de salida utilizando los valores del ejemplo 1.

Una vez que se haya ejecutado Simulink y se haya elegido la opción de “New Blank Model”, los componentes a utilizar serán: Step, Sum, Gain, Transfer Function y Scope, que se encuentran en las siguientes pestañas:

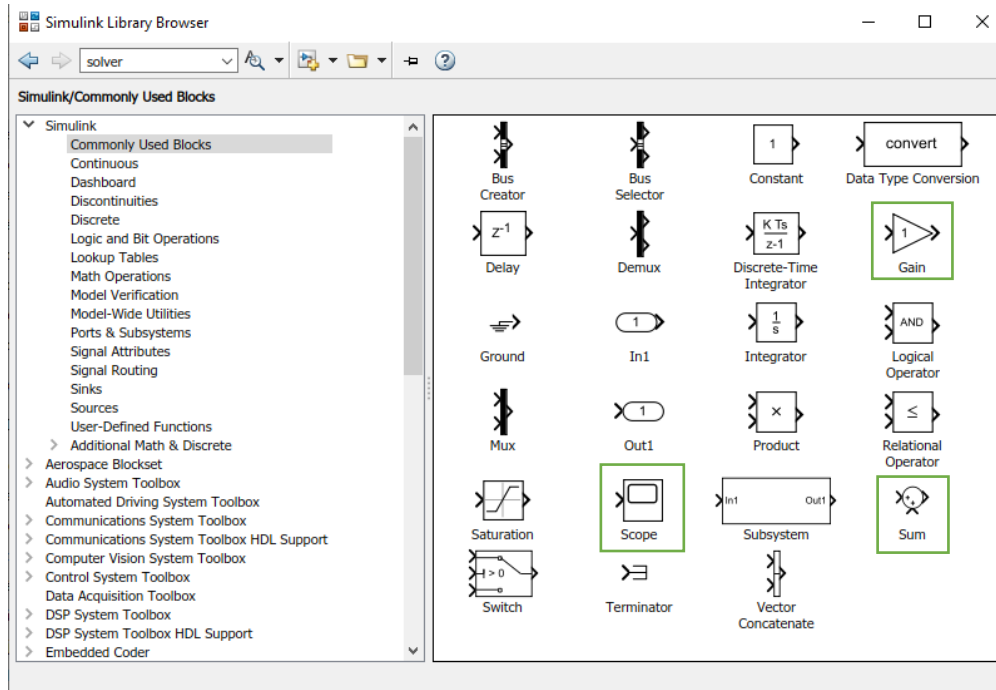


Figura 101. Ubicación de los componentes.

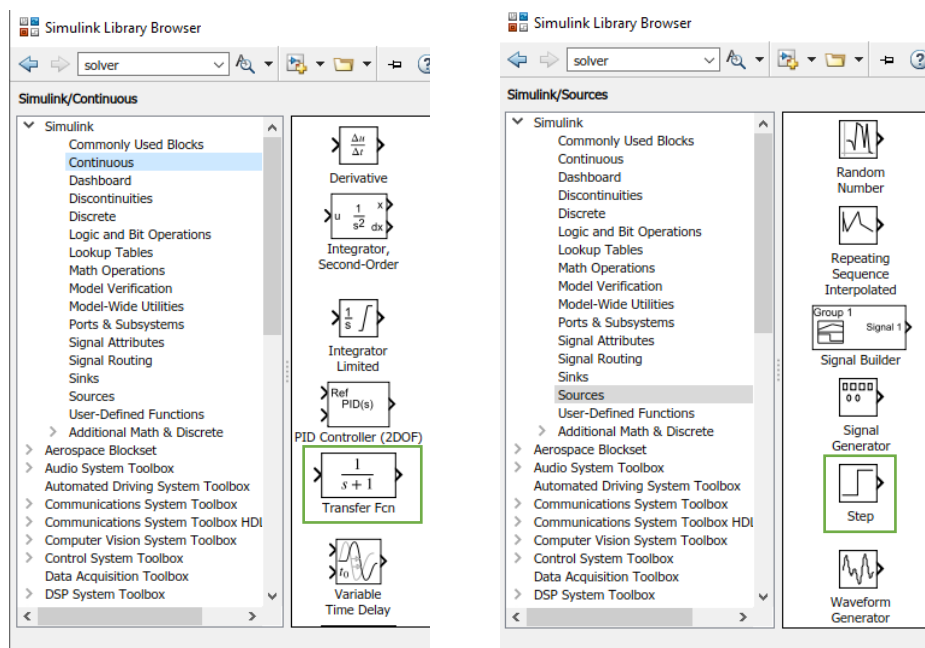


Figura 102. Ubicación de los componentes.

Se colocarán de la siguiente manera:

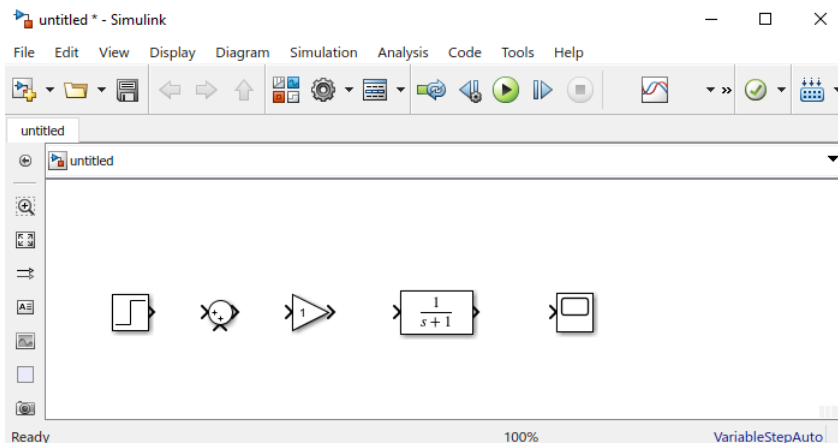


Figura 103. Orden de los componentes.

A continuación, se unirán los componentes.

Se revisan los valores de la entrada escalón, ya que el valor inicial debe ser cero al igual que el “Step time”.

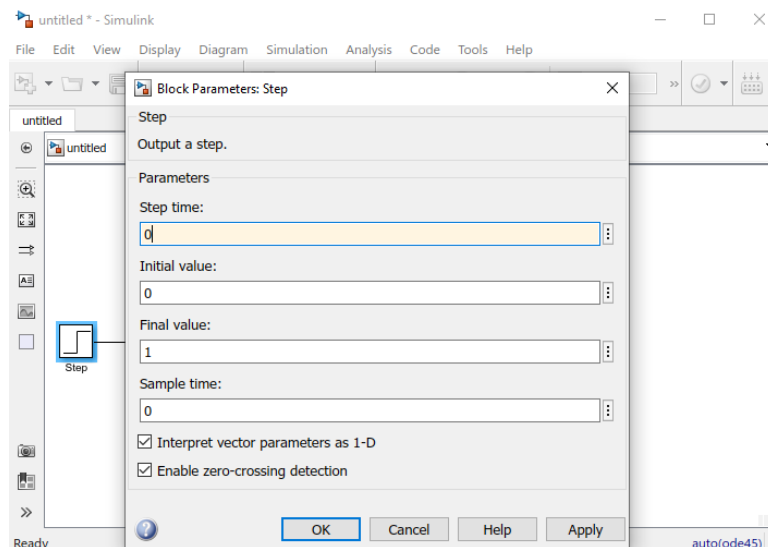


Figura 104. Parámetros de la función Step.

También se cambiará un signo del sumador, ya que la retroalimentación debe ser negativa.

Quedando: “|+-”

Ahora se asignarán los valores de la ganancia (kp) y de la función de transferencia.

Realizando la operación:  $k_p = \frac{(\frac{1}{0.1} - 5)}{5} = 1$

La función es:  $H(s) = \frac{5}{s^2 + 5}$

Se conecta la retroalimentación del cable antes del “Scope” al signo menos del sumador. Como pudimos observar en la gráfica del ejemplo 1, el tiempo de respuesta es de aproximadamente 4 segundos, por lo que cambiaremos el valor como se muestra en la siguiente figura:

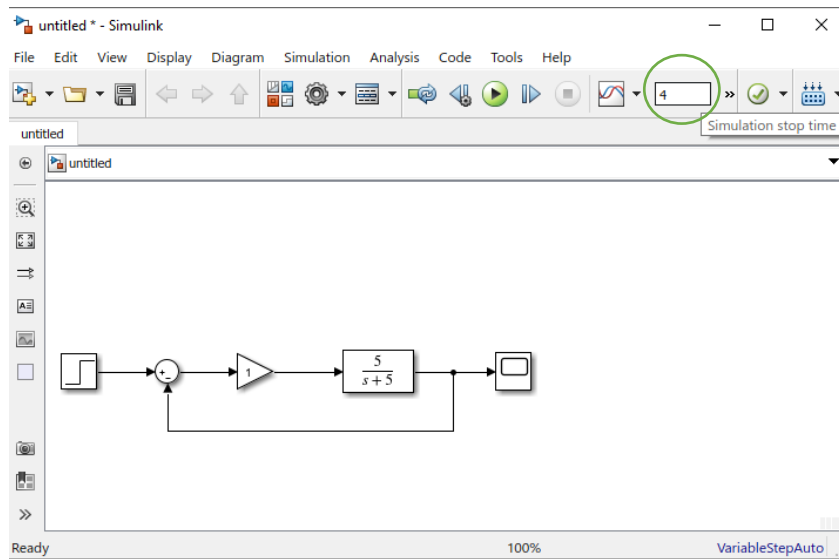


Figura 105. Sistema en Simulink.

Damos clic al botón Run, esperamos unos segundos y damos clic en el osciloscopio, obteniendo la gráfica:

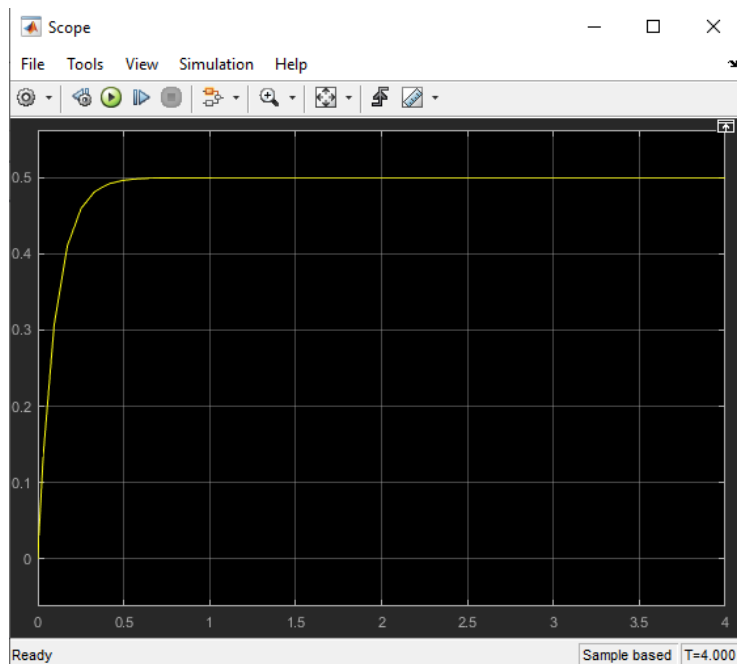


Figura 106. Respuesta del sistema.



### Ejemplo 3:

Para este ejemplo utilizaremos los componentes: Step (2), Sum (2), PID Controller, Transfer Function (4), Mux que sirve para combinar las entradas entregando un solo valor de salida y Scope.

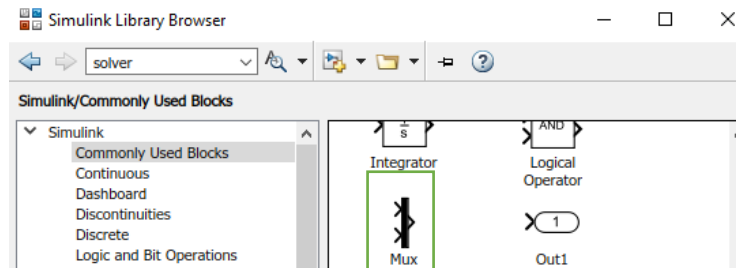


Figura 107. Ubicación del componente Mux.

Se colocarán los componentes de la siguiente manera:

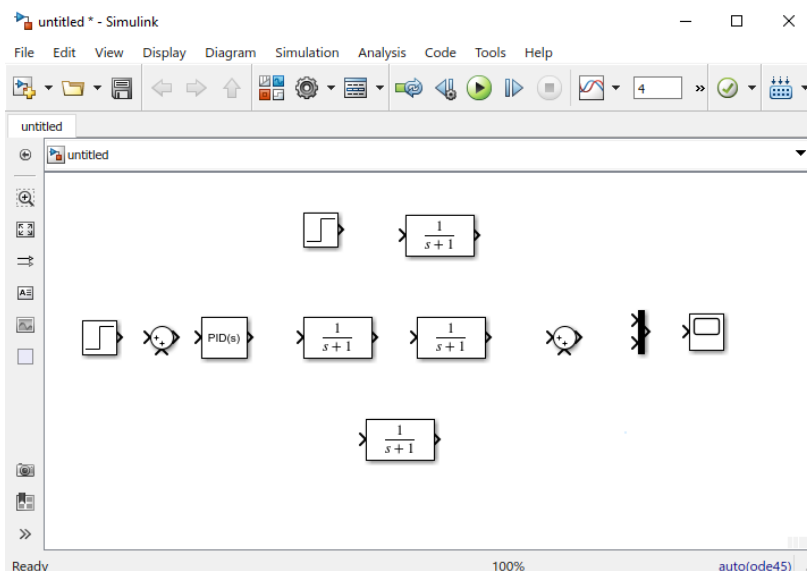


Figura 108. Acomodo de componentes.

Se cambia el signo del primer sumador escribiendo “|+” y en el segundo sumador se escribe “++”. Ahora se une la línea principal de componentes, se invierte el componente de función de transferencia colocado en la parte de abajo y se une de la línea después del segundo sumador al primer sumador.

En la parte de arriba la entrada Step se une a la función de transferencia y esta a su vez a la entrada del segundo sumador

Por último, conectaremos en un punto de la línea después de la entrada Step principal hasta el componente Mux, quedando como se muestra.

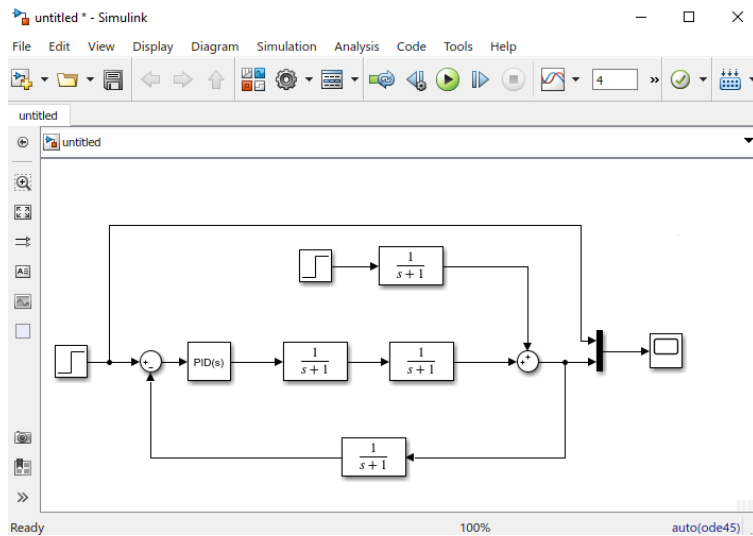


Figura 109. unión de componentes.

Damos doble clic al PID Controller y modificamos para que sea Sólo un Controlador Proporcional.

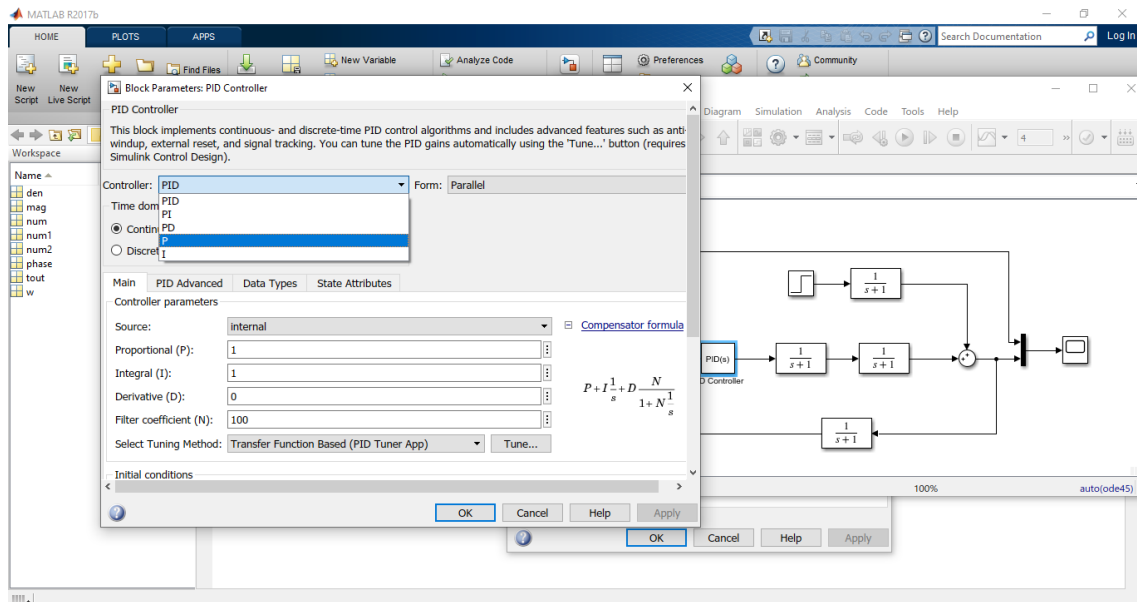


Figura 110. Cambio al controlador P.

A continuación, modificaremos parámetros e insertaremos el valor de las funciones de transferencia. Insertamos cuatro funciones en nuestro sistema que tendrán los siguientes valores:

Función de transferencia Proceso:  $\frac{5}{2s^2+6}$

Función de transferencia Final:  $\frac{1}{1}$

Función de transferencia Media:  $\frac{1}{1}$

Función de transferencia Perturbación:  $\frac{5}{2s^2+6}$

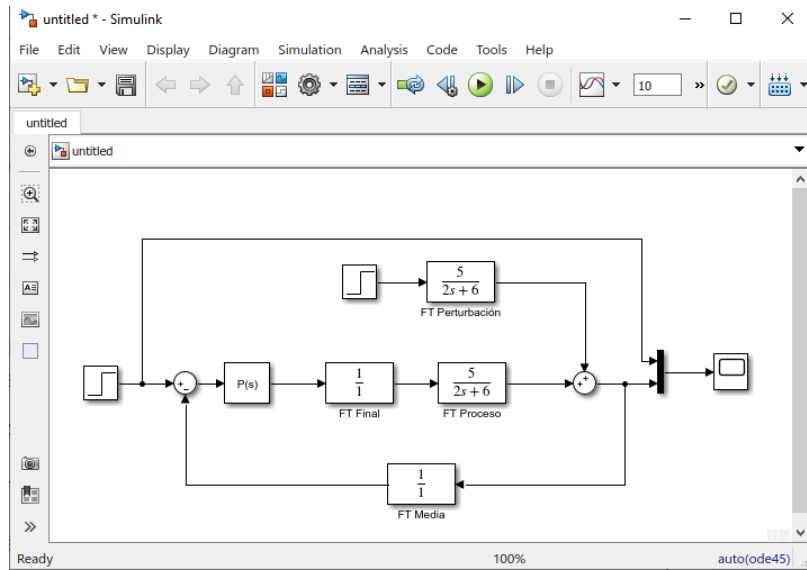


Figura 111. Sistema terminado.

Obteniendo la siguiente respuesta:

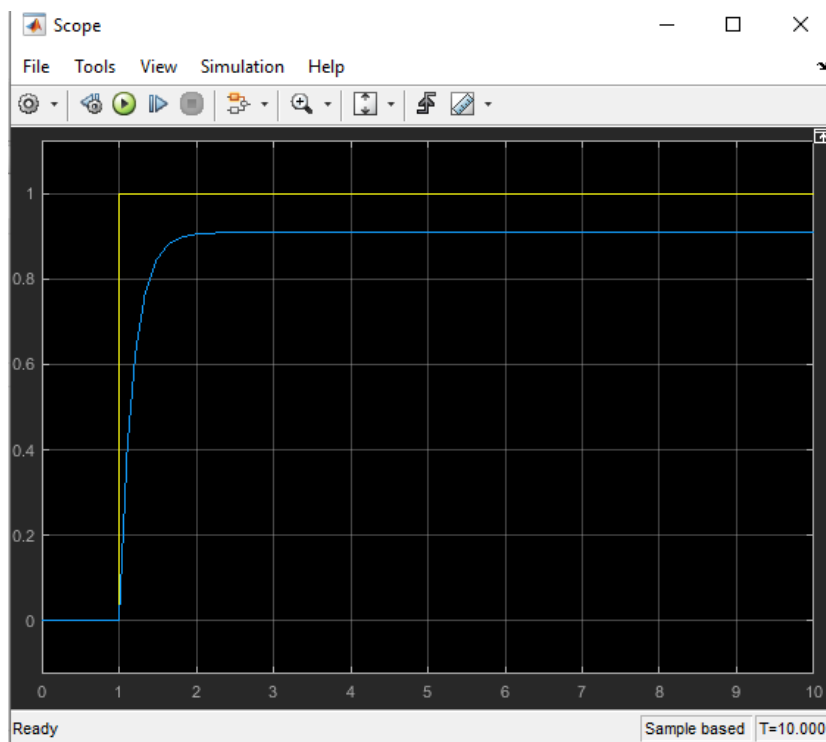


Figura 112. Respuesta del controlador.

## Controlador PI

El control integral, algunas veces llamado reset o control flotante, es una acción de control que provoca un cambio en la señal de salida respecto del tiempo a una razón proporcional de la cantidad de error. La acción integral del controlador responde a un error acumulado en el tiempo, cambiando la señal de salida tanto como se necesite para eliminar completamente el error. Si la acción proporcional (P) le dice a la salida tanto desplazarse cuando un error aparece, la acción integral (I) le dice a la salida que tan rápido moverse cuando un error aparece. La acción proporcional (P) actúa en el presente y la acción integral (I) actúa en el pasado.

### Ejemplo 1:

Utilizaremos la función de transferencia de primer orden:  $\frac{0.5}{s+5}$

Los componentes a utilizar en Simulink son: Step (2), Sum (3), Gain (2), Transfer Function (2), Integrator, Mux y Scope.

Colocamos los componentes de la siguiente manera:

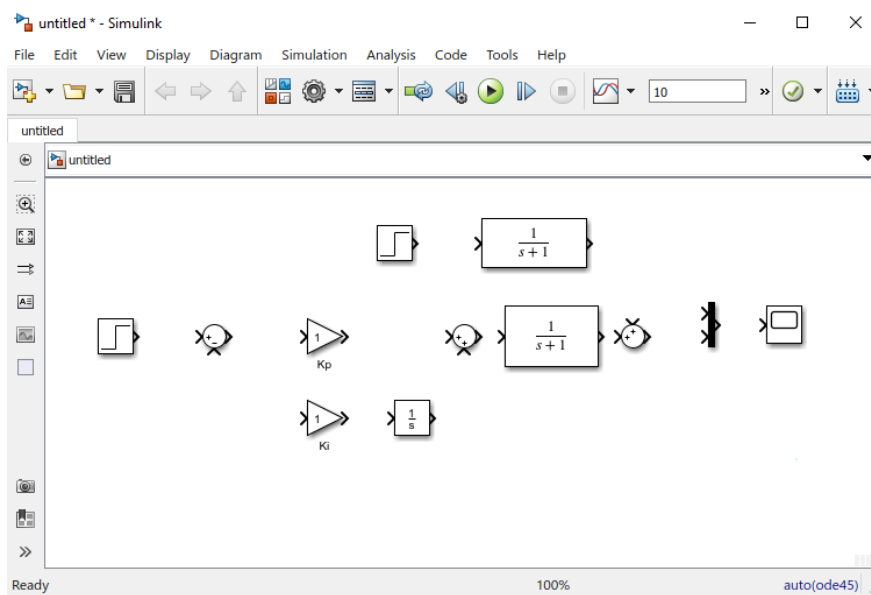


Figura 113. Componentes a utilizar.

Cambiamos un signo en el primer sumador escribiendo “|+” y en el tercer sumador para cambiar la disposición de las entradas colocamos “++”.

Para insertar los valores de la función debemos escribir como coeficiente del numerador [0.5] y como coeficiente del denominador [1 5].

El componente Gain que se encuentra en la línea principal será nuestro Kp y tendrá un valor de 80, el segundo componente Gain será nuestro Ki con un valor de 15.

Debemos unir la línea principal de componentes, se conecta  $K_i$  a la línea principal, al integrador y esto va a al segundo punto de suma, también debemos conectar la retroalimentación (de la línea principal al signo negativo del primer sumador), unir al componente Mux la señal de entrada y este a su vez al osciloscopio. El circuito queda de la siguiente manera:

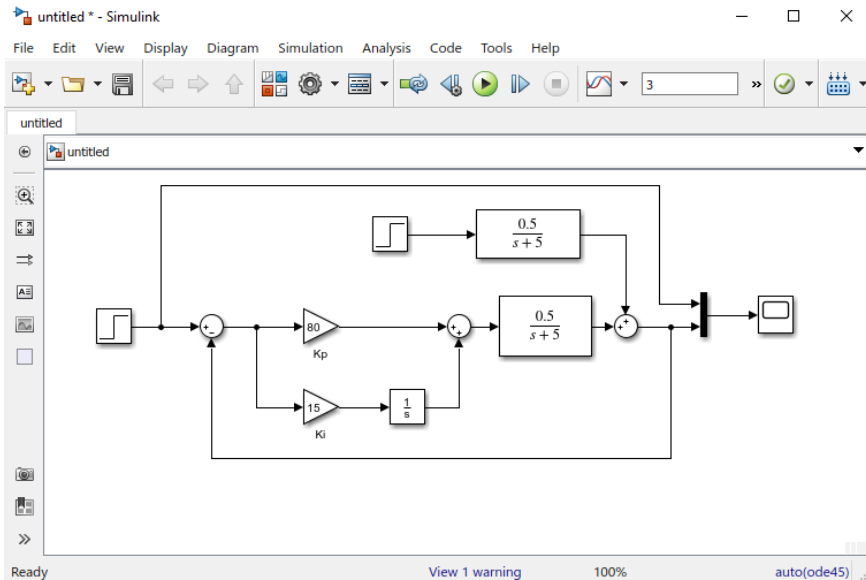


Figura 114. Circuito terminado.

Cambiamos el tiempo de simulación a tres segundos y damos clic en el botón verde Run, después de que compile abrimos el osciloscopio observando las señales siguientes:

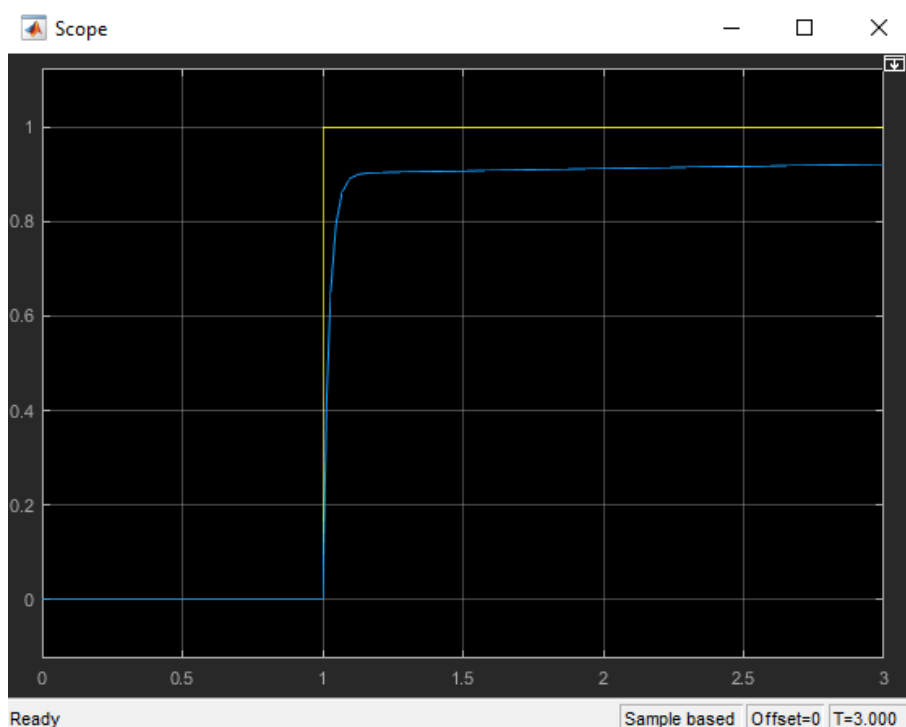


Figura 115. Respuesta del sistema.

## Ejemplo 2:

La función a utilizar será  $\frac{0.5}{s+2}$

Los componentes que utilizaremos son: Step, Sum, PID Controller, Transfer Function, Mux y Scope.

Se colocarán de la siguiente manera:

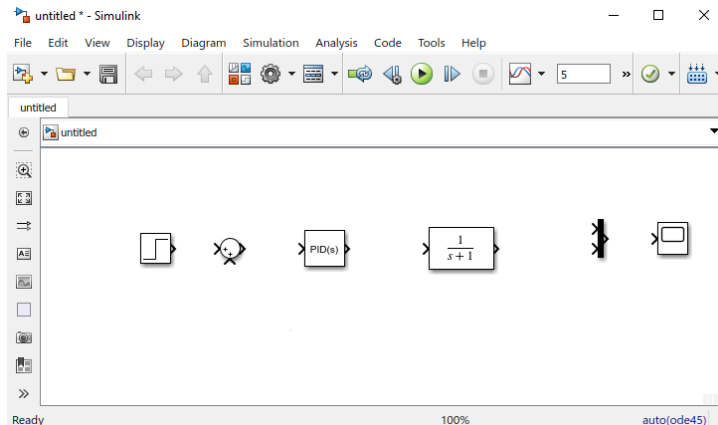


Figura 116. Componentes.

Debemos dar doble clic en el controlador y aparecerá una ventana en la que seleccionaremos el tipo de controlador que deseamos ocupar, en este caso PI.

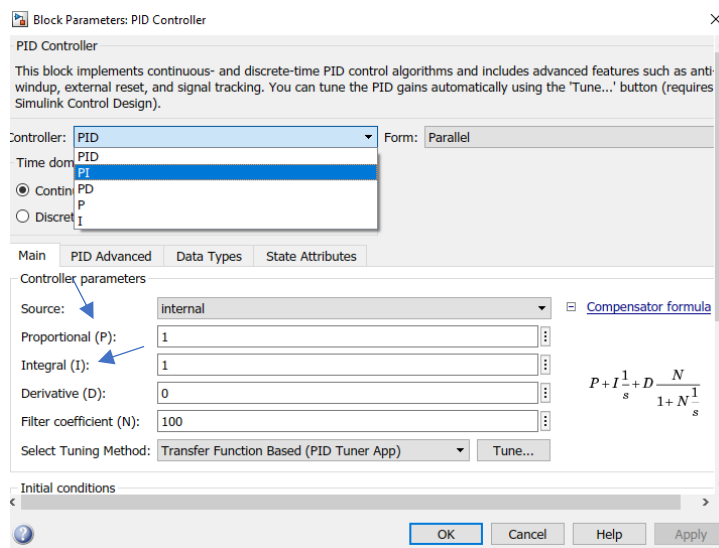


Figura 117. Selección de controlador PI.

Se señaló con flechas el lugar en el que asignaremos el valor de Kp y de Ki. Para este ejemplo utilizaremos Kp=10 y Ki=1.

Uniremos la línea principal de componentes, la retroalimentación al punto de suma y la entrada al componente Mux para observar ambas señales. El tiempo a utilizar será de cinco segundos. Se observará un sistema similar al de la figura:

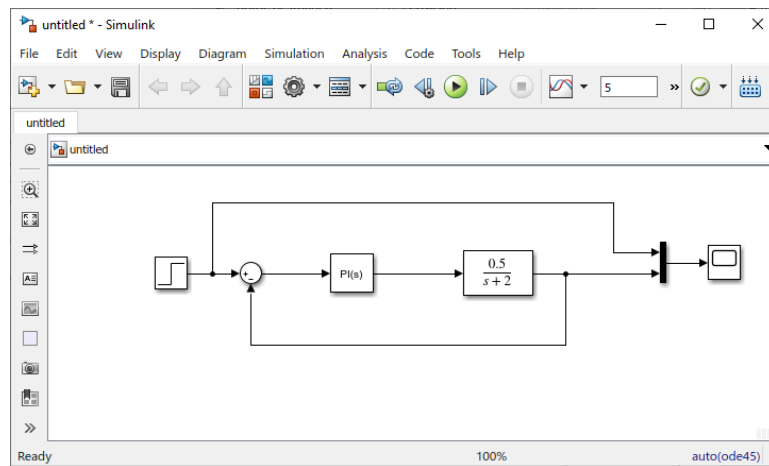


Figura 118. Sistema conectado.

Por último, abrimos el osciloscopio y observamos.

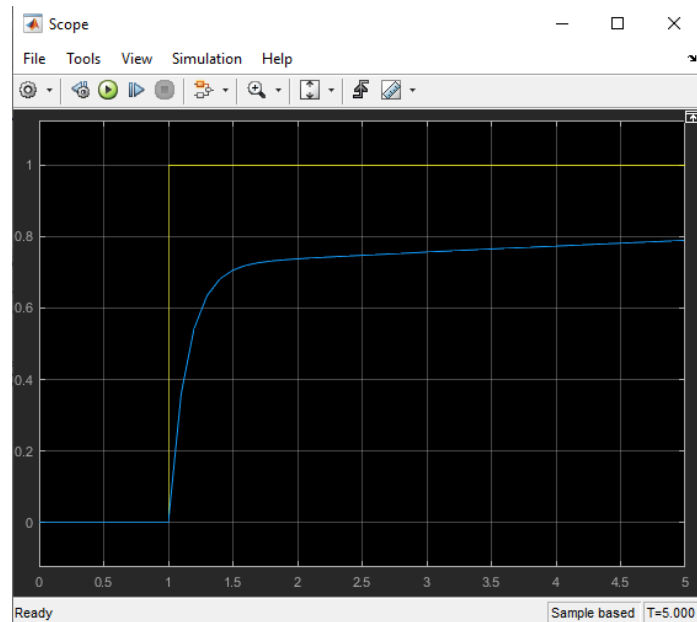


Figura 119. Respuesta del sistema.

### Ejemplo 3:

Utilizaremos una función de transferencia de segundo orden:  $\frac{10}{2s^2+s+6}$

Se necesitan los mismos componentes del ejemplo anterior y se hace la misma conexión, sólo cambiarán los valores de la función de transferencia de Kp y de Ki.

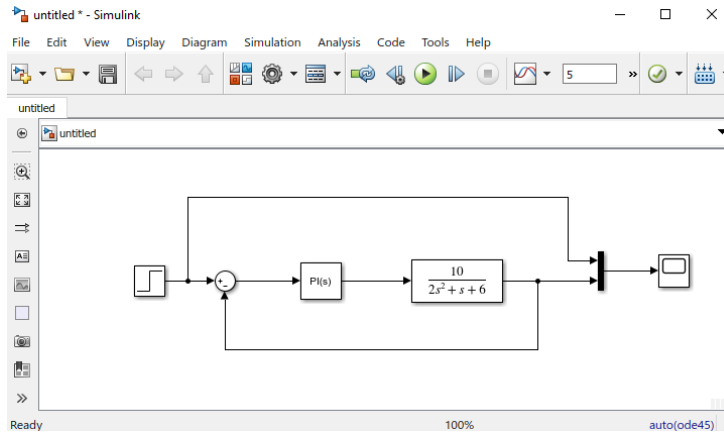


Figura 120. Conexión del sistema.

Después de ingresar la función, podemos obtener los parámetros necesarios de Kp y Ki con ayuda del controlador. Se da doble clic y posteriormente se selecciona la opción Tune.

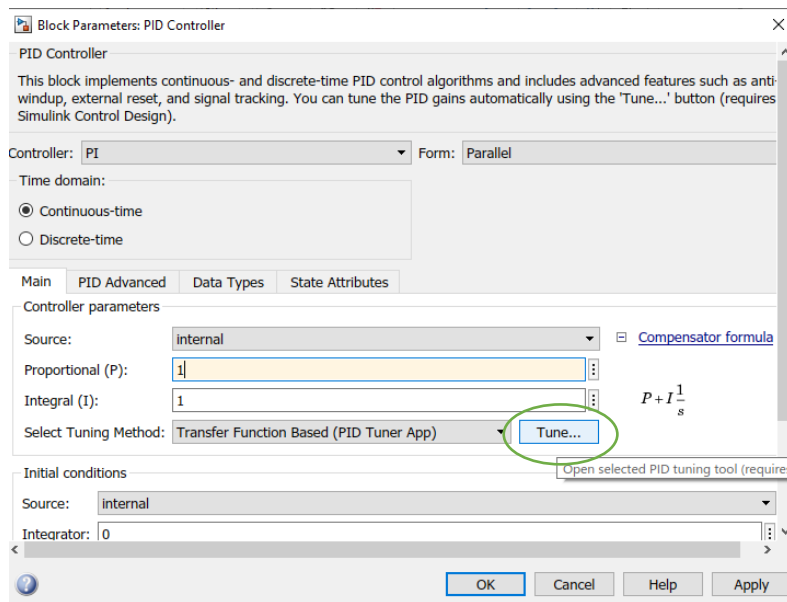


Figura 121. Menú controlador.



Aparece una ventana similar a la de la figura:

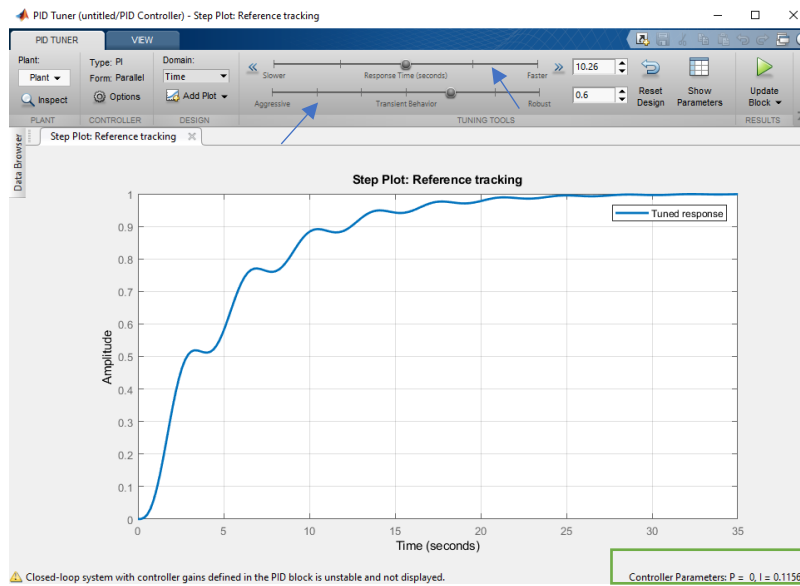


Figura 122. Menú Tune.

Se indica con flechas el lugar en el que se podrá modificar hasta obtener la señal deseada, pudiendo observar los parámetros necesarios para obtenerla en la parte de abajo, señalada con un rectángulo. Para este ejemplo se modificaron hasta obtener la señal:

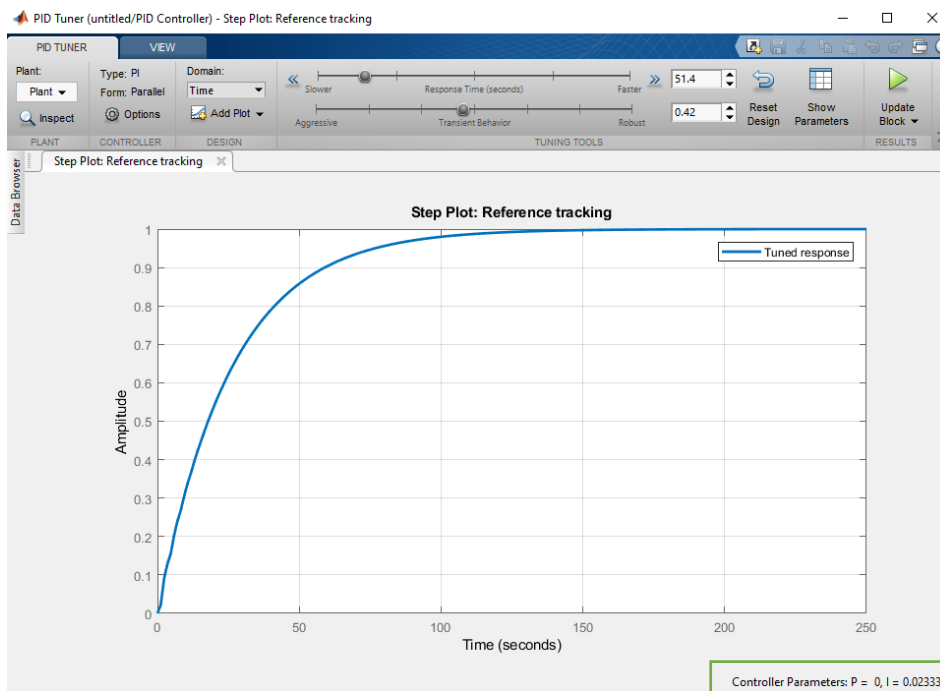


Figura 123. Señal deseada.

Como se puede observar, los parámetros que debemos ingresar son  $K_p=0.1$  y  $K_i=0.02333$ , los ingresamos en la ventana de edición del controlador y aplicamos los cambios. Se asigna un tiempo de 3 segundos, se corre y se observa la siguiente señal:

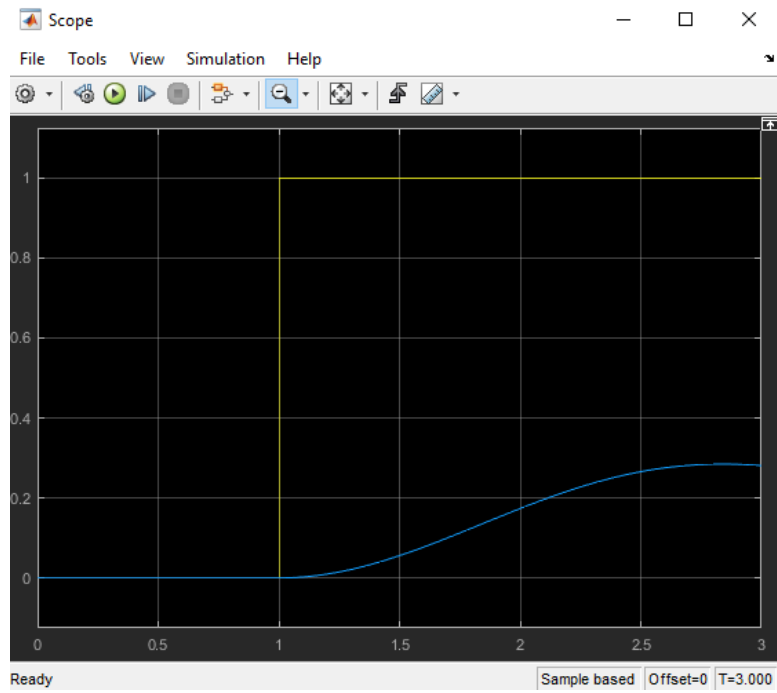


Figura 124. Señal corregida.

## Controlador PID

El control derivativo, algunas veces llamado rate(razón) o pre-act, es una acción de control que realiza un desplazamiento en la señal de salida proporcional a la tasa a la cual cambia la entrada. La acción derivativa del controlador reacciona a que tan rápido cambia la entrada respecto al tiempo, alterando la señal de salida en proporción con la tasa de cambio de entrada. La acción derivativa (D) le dice a la salida que tan lejos ir cuando la entrada cambia.

La acción derivativa (D) actúa en el futuro: eficazmente “anticipa” los overshoot (sobre impulso) intentando una respuesta de salida acorde a que tan rápido la variable de proceso está creciendo o cayendo.

### Ejemplo 1:

Utilizaremos una función de segundo orden:  $\frac{2}{0.5s^2+s+2}$

Debemos declarar los valores del numerador y el denominador con los comandos: **num=[2]** y **den=[0.5 1 2]**. La Función de planta se declarará como “Gp”, el comando para la función de transferencia es: **Gp= tf(num,den);**

Con una función de transferencia de retroalimentación: **H=1;**

A continuación utilizaremos la variable M para obtener la retroalimentación con el comando **M=feedback(Gp,H);** y graficaremos el sistema con una entrada escalón usando “**step(M)**”

Para mantener la gráfica de la señal para observar el cambio al agregar los controladores utilizaremos el comando “**hold on**”

Los valores a utilizar serán: **kp=7**, **ki=13** y **kd=4**, antes de dar Enter se coloca el signo “;”.

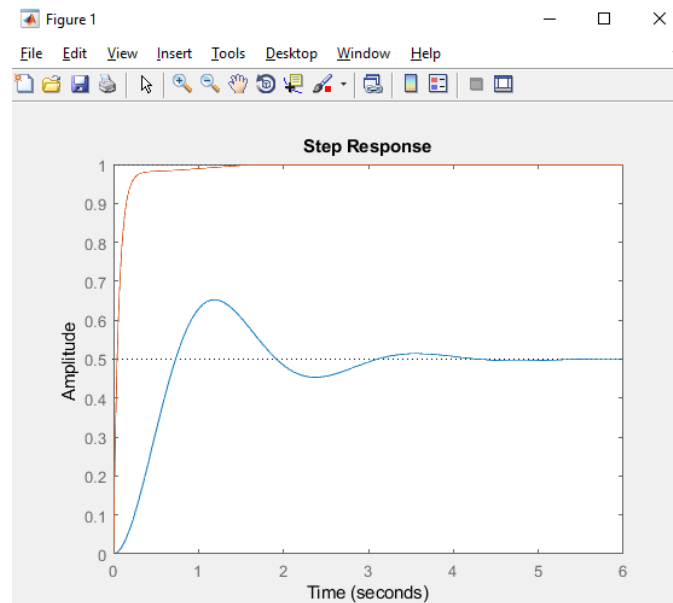
Se utilizará la variable “**Gc**” para indicar la función al aplicarse el valor de los controladores, este comando se declarará de la siguiente manera: **Gc=pid(kp,ki,kd).**

Y obtendremos de nuevo la función de retroalimentación con el comando: **Mc=feedback(Gp\*Gc,H).** Por último, ingresamos le entrada **step(Mc).** El código obtenido es el siguiente:

```
Command Window
>> num=[2];
>> den=[0.5 1 2];
>> Gp= tf(num,den);
>> H=1;
>> M=feedback(Gp,H);
>> step(M)
>> hold on
>> kp= 7;
>> ki=13;
>> kd=4;
>> Gc=pid(kp,ki,kd);
>> Mc=feedback(Gp*Gc,H);
>> step(Mc)
fx >>
```

Señal 125. Código para controlador PID.

La respuesta obtenida es:



Señal 126. Respuesta del sistema.

### Ejemplo 2:

Función:  $\frac{1}{s^2+3s+1}$

Los componentes a utilizar son: Step, Sum, PID Controller, Transfer Function, Mux y Scope.

Se colocarán de la siguiente manera:

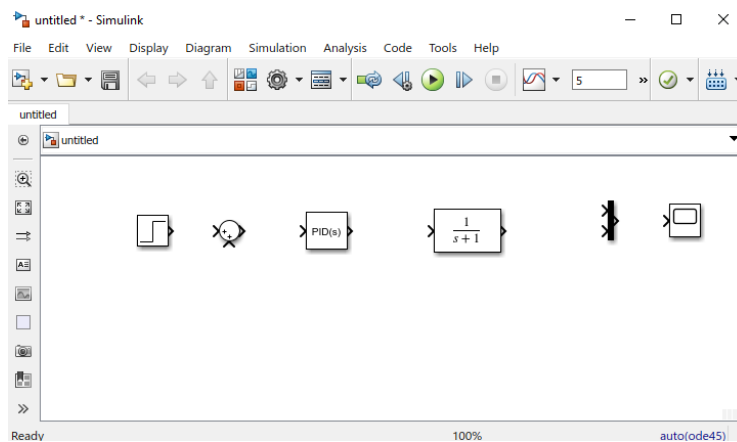


Figura 127. Componentes.

El controlador PID no se modificará, sólo sus valores y la Función de Transferencia.

Dando doble clic en el controlador, ingresaremos los siguientes valores:  $K_p= 24$ ,  $K_i= 1$ ,  $K_d= 8$  y aplicamos los cambios.

En la siguiente imagen se muestran los valores en la ventana correspondiente:

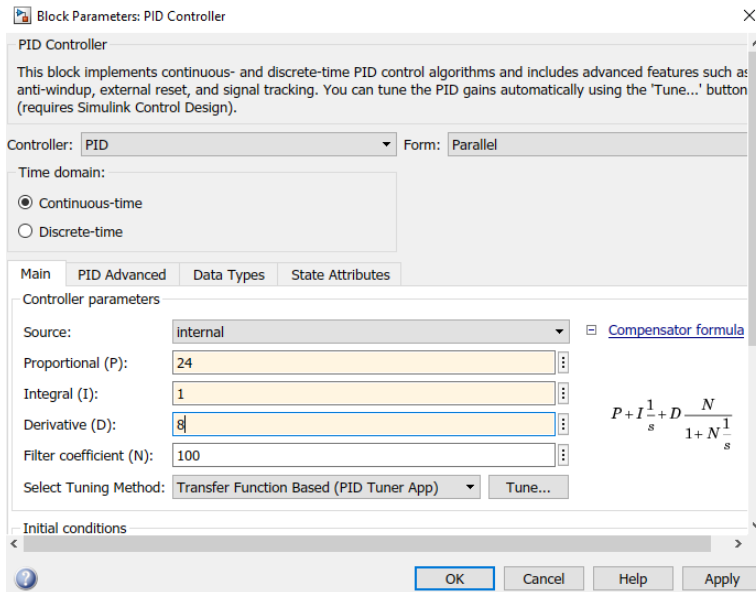


Figura 128. Valores controlador PID.

Por último, ingresamos la función de transferencia, cambiamos el tiempo a tres y compilamos.

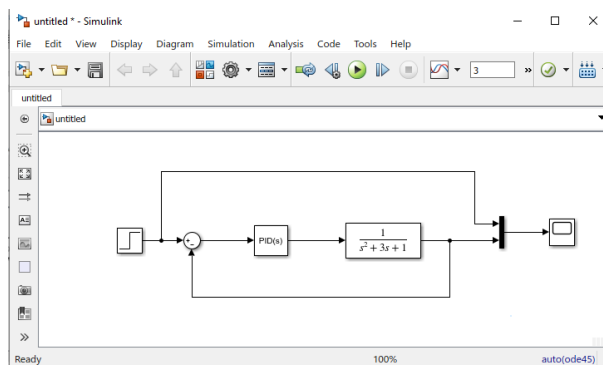


Figura 129. Sistema.

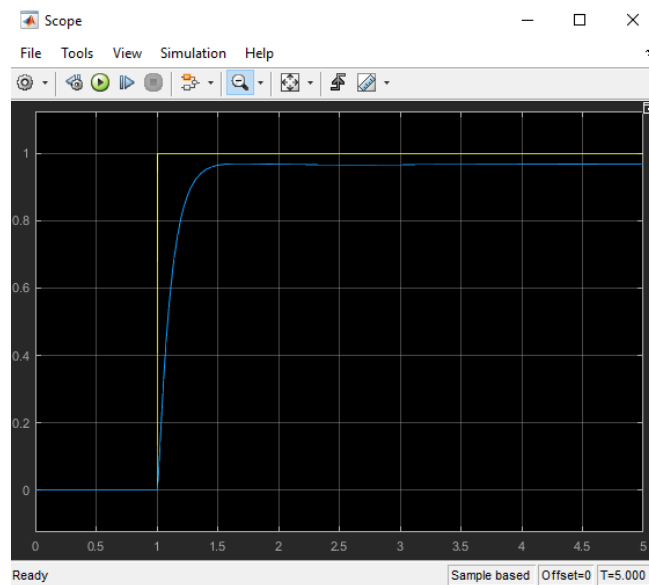


Figura 130. Respuesta obtenida.

### Ejemplo 3:

Utilizaremos la función de transferencia de segundo orden:  $\frac{1}{9s^2+26s+24}$

Los componentes a utilizar en Simulink son: Step, Sum (2), Gain (3), Transfer Function, Integrator, Derivative, Mux y Scope.

Se colocan los componentes de la siguiente manera:

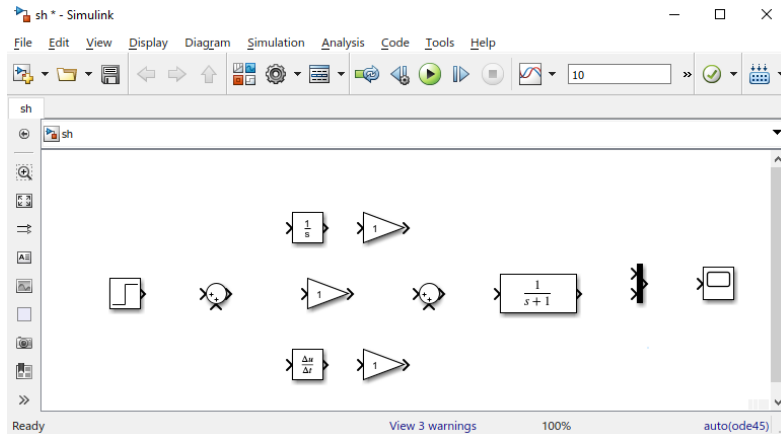


Figura 131. Componentes.

A continuación, se cambiará el signo del primer sumador con “|-” t al segundo sumador se le agregará una entrada utilizando “+++”. Se añadirá la función de transferencia con [1] para le numerador y [9 26 24] para el denominador.

Se unirán de la siguiente manera los componentes:

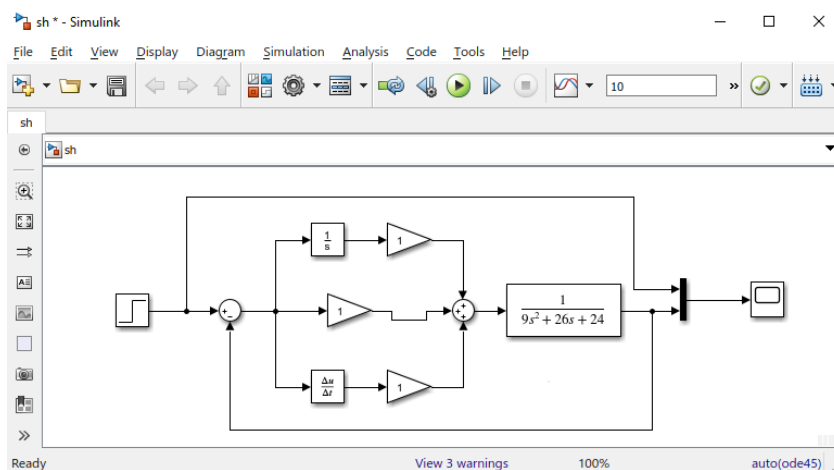


Figura 132. Unión de componentes.

Los valores a ingresar son:

Kp=50, Ki= 25 y Kd=14, en este caso usaremos los 10 segundos de respuesta. Damos clic en el botón Run.

Se cambió el nombre de los componentes indicando el lugar de cada valor.

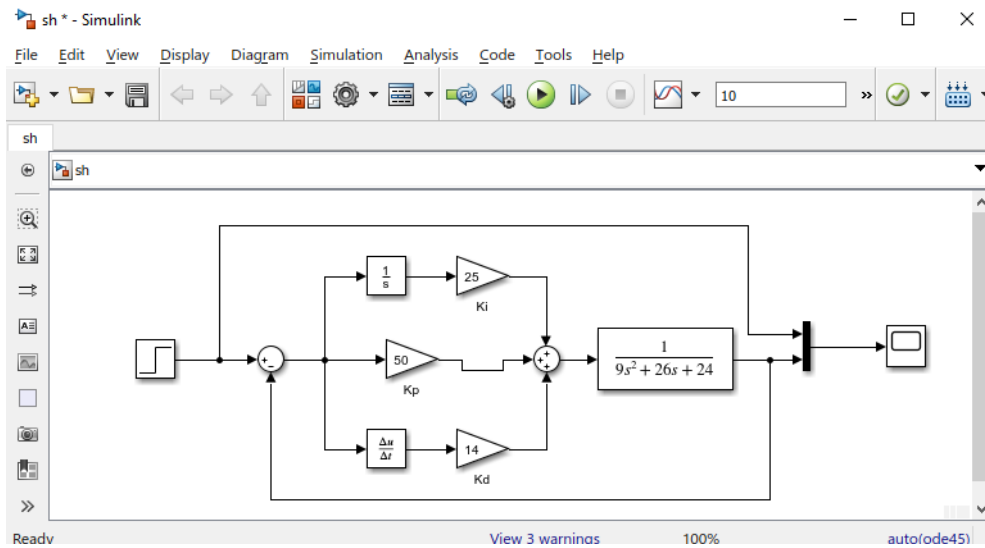


Figura 133. Sistema terminado.

Obtenemos la siguiente gráfica:

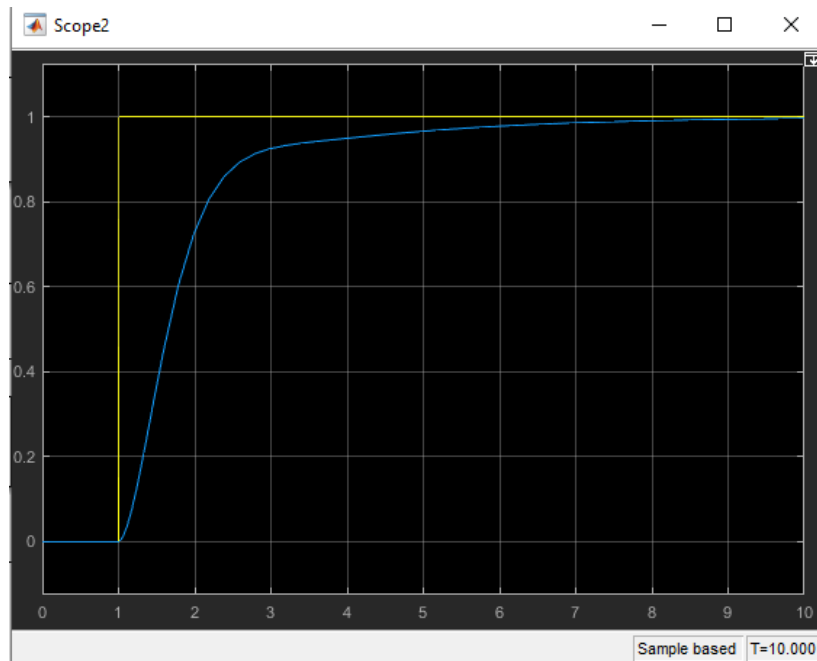


Figura 134. Respuesta al sistema.

## Diagramas de Magnitud y Fase

Se conoce por respuesta en frecuencia, a la respuesta de un sistema, en régimen permanente, cuando se utiliza como señal de entrada una excitación senoidal de amplitud constante y de frecuencia variable desde cero hasta infinito. La respuesta de un sistema LTI ante este tipo de excitación, es otra senoidal de la misma frecuencia que la entrada, pero que difiere en amplitud y fase.

Las dos ventajas principales que presentan este método son: la facilidad experimental de realización y que la FDT en el dominio frecuencia se obtiene reemplazando las del dominio complejo de las Transformadas de Laplace por  $j\omega$ . La nueva función,  $G(j\omega)$ , es una función de variable compleja, cuya representación en módulo y argumento expresará, la amplificación o atenuación del equipo y el desfase introducido a una determinada frecuencia.

Un diagrama de Bode es una representación gráfica que sirve para caracterizar la respuesta en frecuencia de un sistema. Normalmente consta de dos gráficas separadas, una que corresponde con la magnitud de dicha función y otra que corresponde con la fase.

### Ejemplo 1:

Utilizando la función:  $H(S) = \frac{S(S+2)}{(S+4)}$ , compararemos sus diagramas de Bode de magnitud y fase usando MATLAB.

Lo primero que debemos hacer es ingresar los valores de la función, para esto utilizaremos las variables  $\text{num1}=[1 \ 0]$  y  $\text{num2}=[1 \ 2]$ , recordando que el orden para colocar los números es de derecha a izquierda: constante,  $s$ ,  $s^2$ ..., etc.

Después unificaremos los valores con el comando:  $\text{num}=\text{conv}(\text{num1}, \text{num2});$

Y declaramos el denominador como:  $\text{den}=[1 \ 4];$

Para obtener los diagramas de bode utilizaremos tal cual los siguientes comandos:

```
[mag,phase,w]=bode(num,den);
```

```
subplot(211), loglog(w,mag), title('Magnitud'),xlabel('rad/s');
```

```
subplot(212), semilogx(w,phase), title('Fase'),xlabel('rad/s');
```

Por último, agregamos las cuadrículas con el comando “grid” o en la ventana de las gráficas

Se muestran los diagramas realizados manualmente y posteriormente los resultados obtenidos con el programa MATLAB.



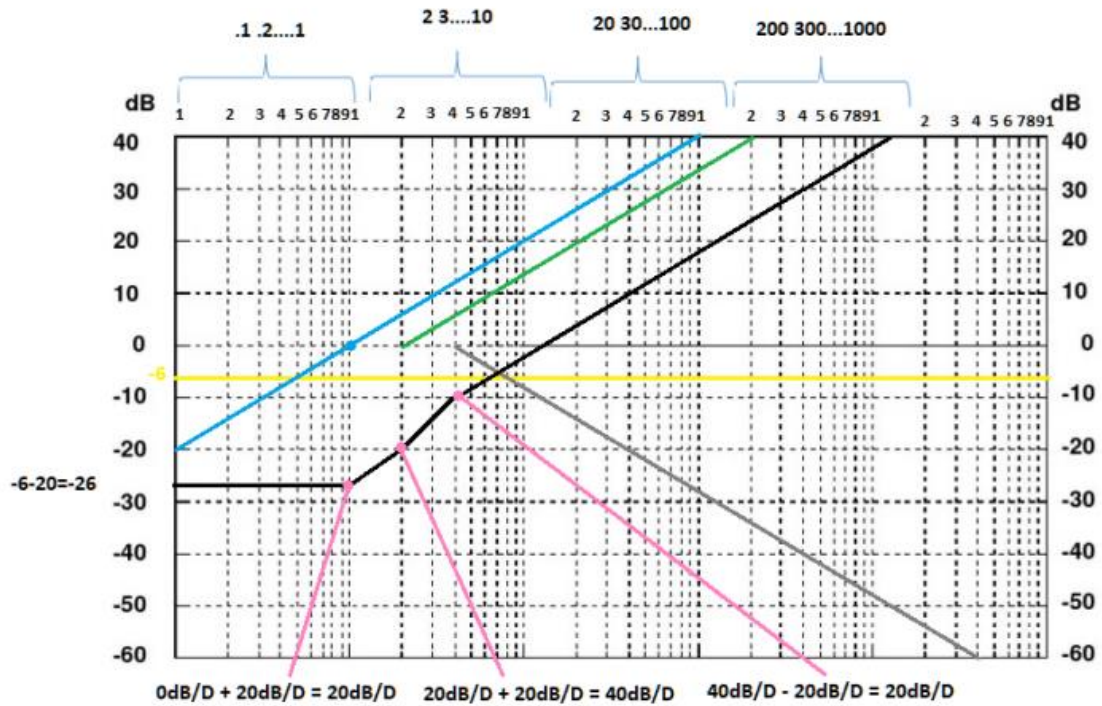


Figura 135. Diagrama Magnitud.

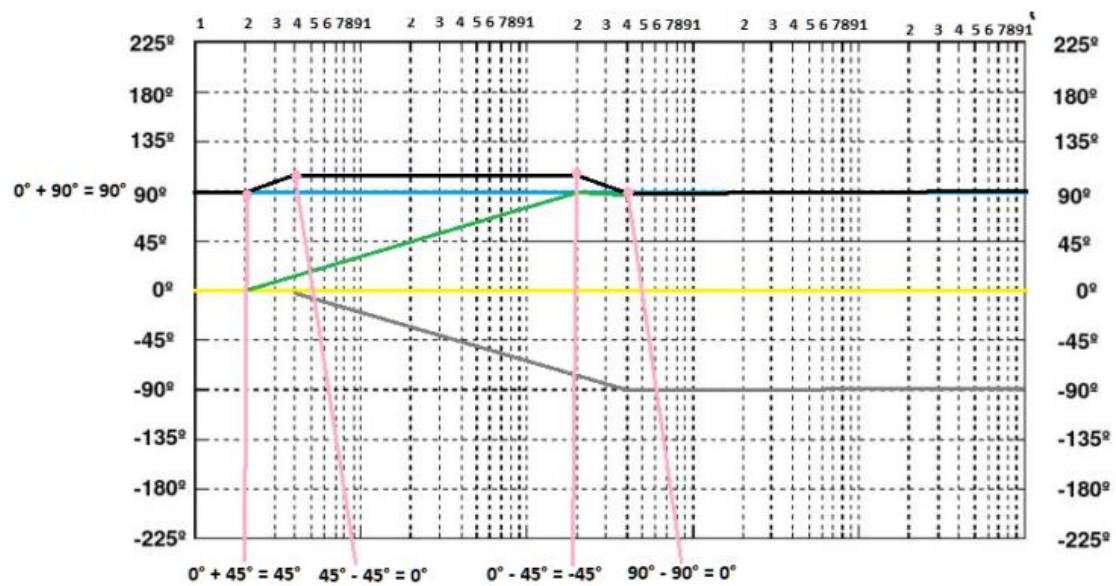


Figura 136. Diagrama Fase.

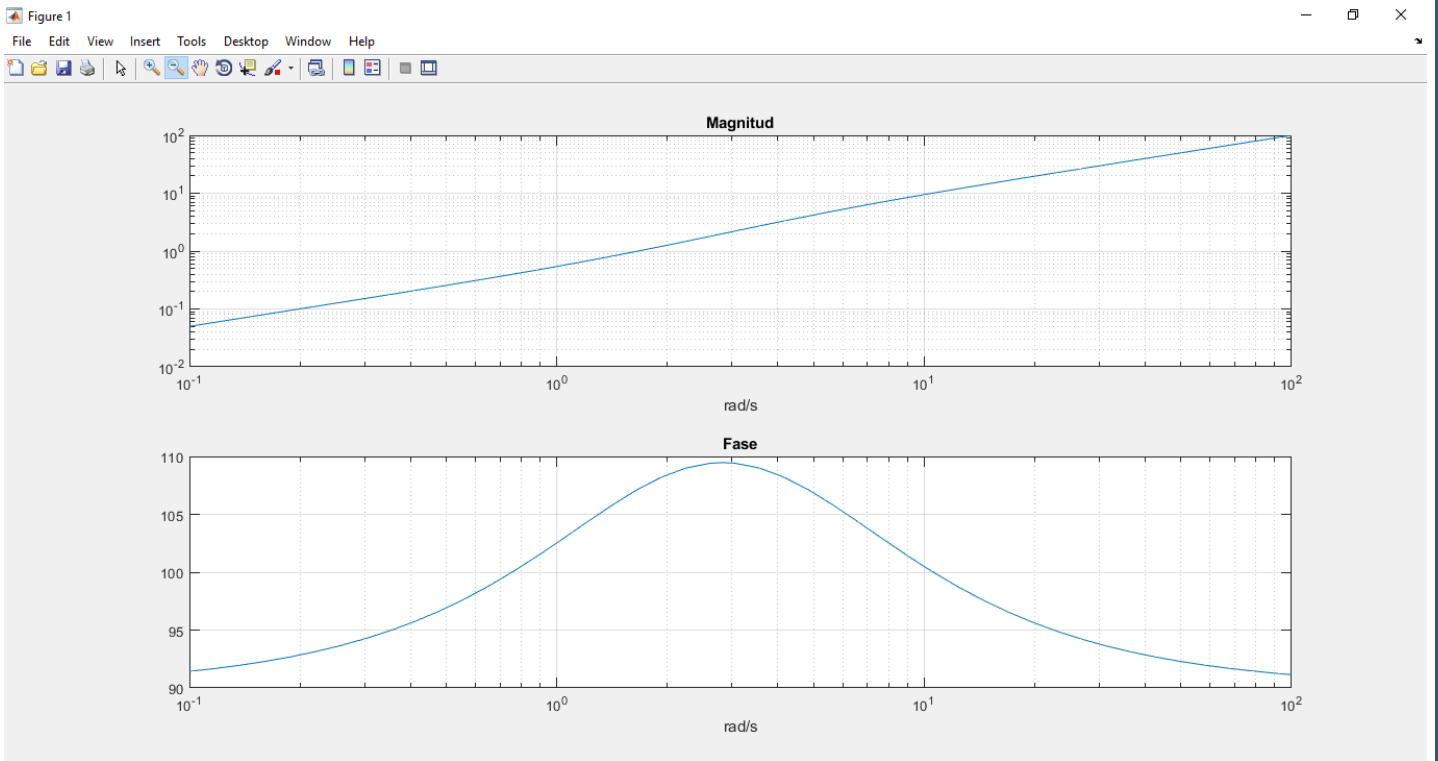


Figura 137. Diagramas de Magnitud y Fase obtenidos en MATLAB

## Ejemplo 2:

Para la función  $H(S) = \frac{S(S+5)}{(S+6)}$

Utilizando los siguientes comandos:

```
>> num1=[1 0];
```

```
>> num2=[1 5];
```

```
>> num=conv(num1,num2);
```

```
>> den=[1 6];
```

```
>> [mag,phase,w]=bode(num,den);
```

```
>> subplot(211), loglog(w,mag), title('Magnitud'),xlabel('rad/s');
```

```
>> grid;
```

```
>> subplot(212), semilogx(w,phase), title('Fase'),xlabel('rad/s');
```

```
>> grid;
```

Obtenemos:

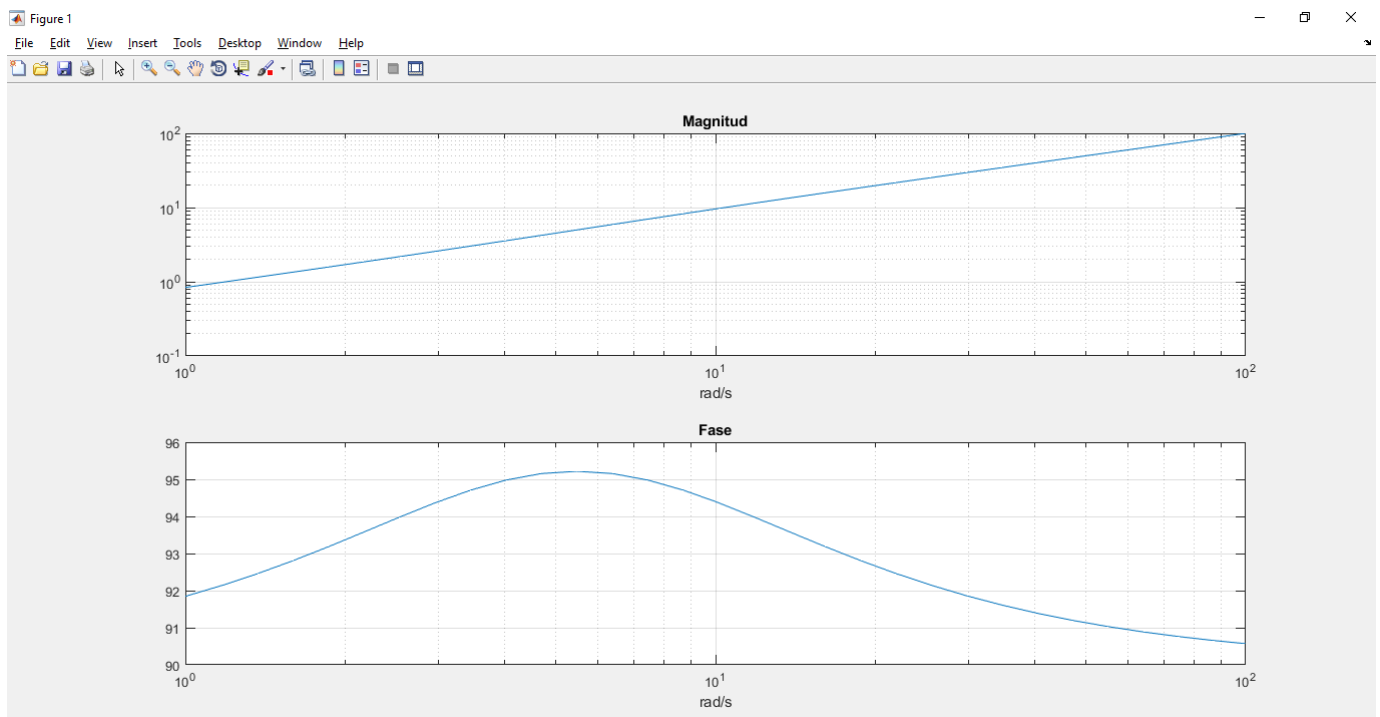


Figura 138. Diagramas de Magnitud y Fase obtenidos en MATLAB

### Ejemplo 3:

Para la función  $H(S) = \frac{S + 1}{s^2 + 3s + 1}$

Utilizando los siguientes comandos:

```
>> num=[1 1];
```

```
>> den=[1 3 1];
```

```
>> [mag,phase,w]=bode(num,den);
```

```
>> subplot(211), loglog(w,mag), title('Magnitud'),xlabel('rad/s');
```

```
>>grid;
```

```
>> subplot(212), semilogx(w,phase), title('Fase'),xlabel('rad/s');
```

```
>>grid;
```

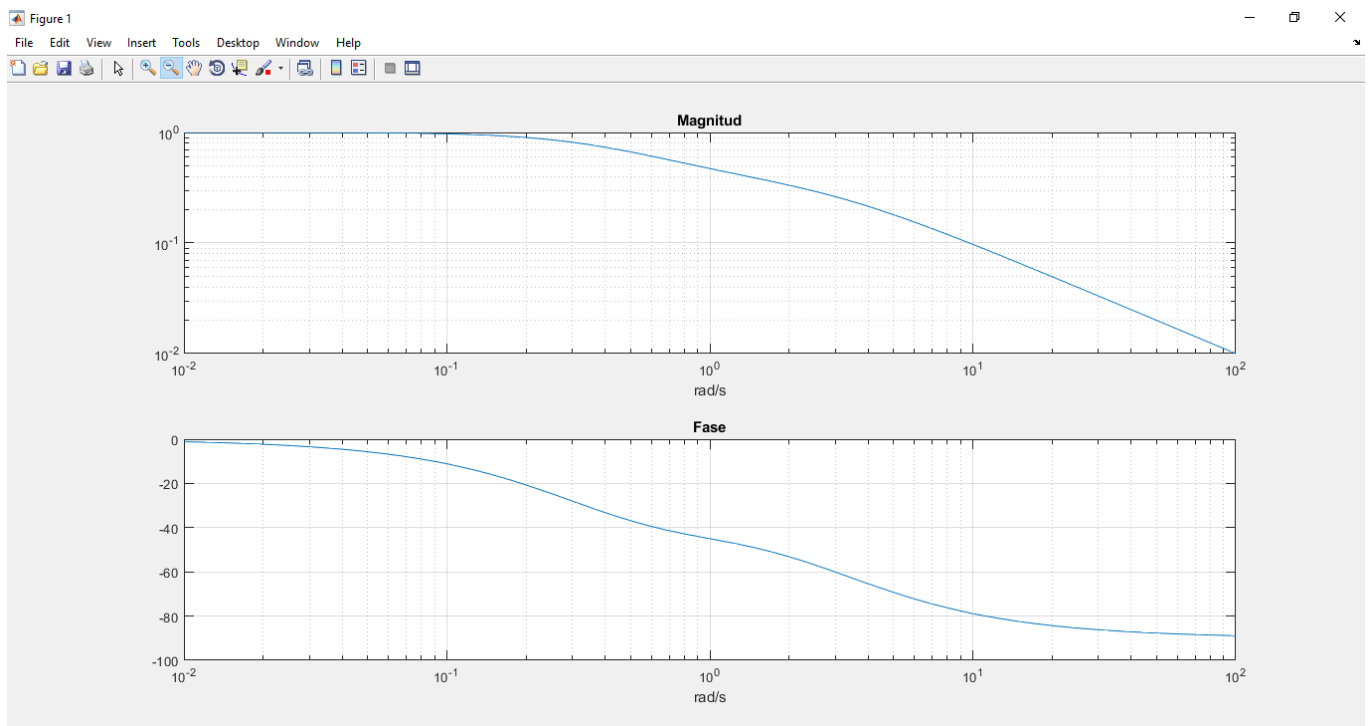


Figura 139. Diagramas de Magnitud y Fase obtenidos en MATLAB

#### Ejemplo 4:

Para la función  $H(S) = \frac{1}{9s^2 + 26s + 24}$

Utilizando el código:

```
>> num=[1];
```

```
>> den=[9 26 24];
```

```
>> [mag,phase,w]=bode(num,den);
```

```
>> subplot(211), loglog(w,mag), title('Magnitud'),xlabel('rad/s');
```

```
>> grid;
```

```
>> subplot(212), semilogx(w,phase), title('Fase'),xlabel('rad/s');
```

```
>> grid;
```

Obtenemos las gráficas:

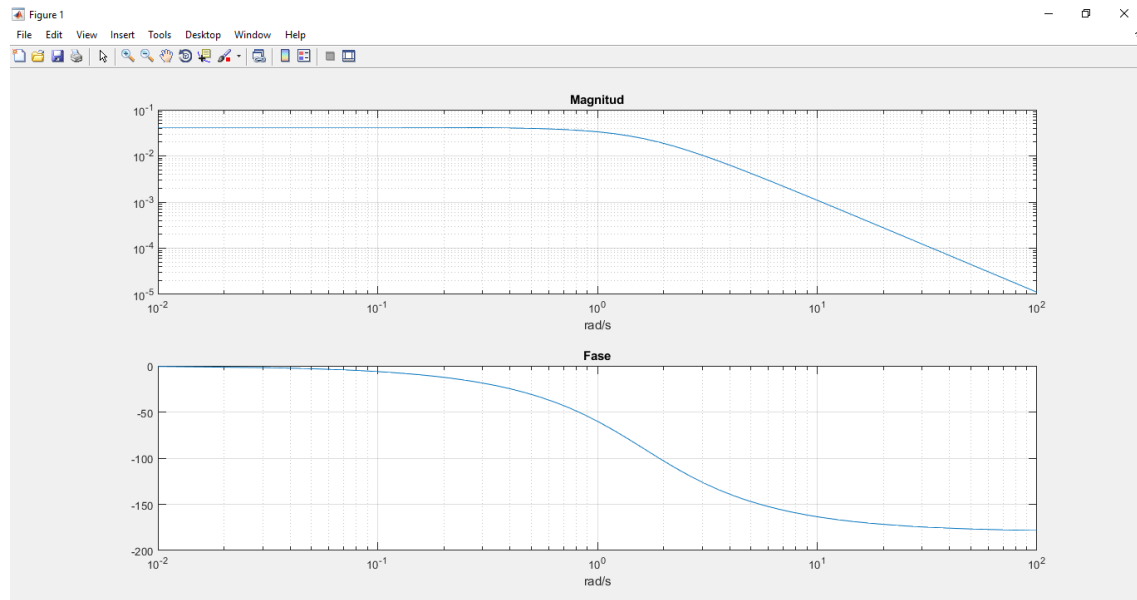


Figura 140. Diagramas de Magnitud y Fase.

## Estabilidad

Una manera de saber si nuestro sistema es estable o inestable es utilizando el método de Routh-Hurwitz, para utilizar este criterio en MATLAB no existe un comando específico, más bien se puede utilizar un código o script ejecutable para obtener la tabla y observar si el sistema presenta cambios de signo para así determinar la estabilidad o inestabilidad del sistema.

En la página web de MATLAB, se pueden encontrar diferentes scripts desarrollados, a continuación se muestra el link de donde se obtuvo el código: <https://la.mathworks.com/matlabcentral/fileexchange/17483-routh-hurwitz-stability-criterion>

Una vez obtenido el código, realicé algunas modificaciones para que se mostraran las indicaciones en español, quedando de la manera siguiente:

```
clc
clear
r=input('Ingresa los valores de los coeficientes del polinomio: ');
m=length(r);
n=round(m/2);
q=1;
k=0;
for p = 1:length(r)
    if rem(p,2)==0
        c_even(k)=r(p);
    else
        c_odd(q)=r(p);

        k=k+1;
        q=q+1;
    end
end
a=zeros(m,n);

if m/2 ~= round(m/2)
    c_even(n)=0;
end
a(1,:)=c_odd;
a(2,:)=c_even;
if a(2,1)==0
    a(2,1)=0.01;
end
for i=3:m
    for j=1:n-1
        x=a(i-1,1);
        if x==0
            x=0.01;
        end

        a(i,j)=((a(i-1,1)*a(i-2,j+1))-(a(i-2,1)*a(i-1,j+1)))/x;
    end
    if a(i,')==0
        order=(m-i+1);
        c=0;
        d=1;
        for j=1:n-1
            a(i,j)=(order-c)*(a(i-1,d));
            d=d+1;
        end
    end
end
```

```

        c=c+2;
    end
end
if a(i,1)==0
    a(i,1)=0.01;
end
end
Right_poles=0;
for i=1:m-1
    if sign(a(i,1))*sign(a(i+1,1))==-1
        Right_poles=Right_poles+1;
    end
end
fprintf('\n Tabla de Routh-Hurwitz:\n')
a
fprintf('\n Número de polos =%2.0f\n',Right_poles)

reply = input('¿Necesitas las raices del sistema? S/N ', 's');
if reply=='s' || reply=='S'
    Raices=roots(r);
    fprintf('\n Raices de los coeficientes polinomiales:\n')
    Raices
else
end
end

```

Para realizar los ejemplos debemos copiar el código y pegar en la ventana de comandos de MATLAB, nos mostrará esto en pantalla:

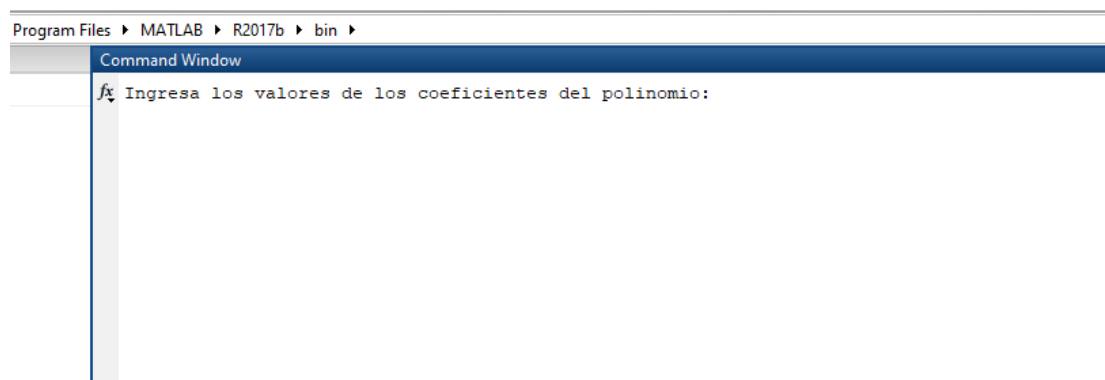


Figura 141. Mensaje al ejecutar el código.

## Ejercicio 1:

Analizaremos el polinomio:  $8s^5 + 3s^4 + 4s^3 + s^2 + 7s + 7 = 0$

Los valores deben ingresarse dentro de corchetes “[ ]”, separados por un espacio, una vez realizado esto dar “Enter” y se mostrará la tabla.

```
Command Window
Ingresa los valores de los coeficientes del polinomio: [8 3 4 1 7 7]

Tabla de Routh-Hurwitz:

a =

    8.0000    4.0000    7.0000
    3.0000    1.0000    7.0000
    1.3333   -11.6667     0
    27.2500    7.0000     0
   -12.0092     0         0
    7.0000     0         0

Número de polos = 2
fx ¿Necesitas las raíces del sistema? S/N |
```

Figura 142. Tabla de Routh-Hurwitz.

En este ejemplo existe un cambio de signo en los valores de la primera columna en el valor de 27.25, después se encuentra -12.0092 y vuelve a cambiar a 7 positivo, por lo tanto, el sistema es **inestable**.

Este código utilizado ofrece las raíces del sistema, se elegirá esta opción y se obtiene:

```
Número de polos = 2
¿Necesitas las raíces del sistema? S/N s

Raíces de los coeficientes polinomiales:

Raíces =

    0.6926 + 0.7715i
    0.6926 - 0.7715i
   -0.5135 + 0.9201i
   -0.5135 - 0.9201i
   -0.7332 + 0.0000i
fx >> |
```

Figura 143. Raíces del Sistema.



## Ejemplo 2:

Para este utilizaremos el polinomio:  $2s^5 + 5s^4 - 5s^3 - 4s^2 - s - 7 = 0$   
Con el fin de observar cómo pueden influir los signos negativos en la estabilidad.  
Después de copiar el código ingresamos los valores [2 5 -5 -4 -1 -7]

```
Command Window
Ingresar los valores de los coeficientes del polinomio: [2 5 -5 -4 -1 -7]

Tabla de Routh-Hurwitz:

a =

    2.0000    -5.0000    -1.0000
    5.0000    -4.0000    -7.0000
   -3.4000     1.8000         0
   -1.3529    -7.0000         0
   19.3913         0         0
   -7.0000         0         0

Número de polos = 3
¿Necesitas las raíces del sistema? S/N n
fx >> |
```

Figura 144. Tabla de Routh-Hurwitz.

Se observa el cambio de signo de 5 a -3.4, posteriormente de -1.35 a 19.39 y, por último, de 19.39 a -7. El sistema es **inestable** bajo este criterio.

## Ejemplo 3:

Utilizaremos el polinomio:  $-4s^5 - 2s^4 - 1s^3 - 4s^2 - 2s - 3 = 0$   
Ingresando: [-4 -2 -1 -4 -2 -3]

```
Command Window
Ingresar los valores de los coeficientes del polinomio: [-4 -2 -1 -4 -2 -3]

Tabla de Routh-Hurwitz:

a =

   -4.0000   -1.0000   -2.0000
   -2.0000   -4.0000   -3.0000
    7.0000    4.0000         0
   -2.8571   -3.0000         0
   -3.3500         0         0
   -3.0000         0         0

Número de polos = 2
¿Necesitas las raíces del sistema? S/N n
fx >> |
```

Figura 145. Tabla de Routh-Hurwitz.

Con este ejemplo podemos observar que hay **inestabilidad** en el sistema al haber cambio de signo de -2 a 7 y de 7 a -2.85, se podría creer que por ser todos los signos de los coeficientes del polinomio negativos no se observaría el cambio de signo que determina la inestabilidad, pero no es así.

#### Ejemplo 4:

El polinomio a utilizar será:  $s^4 + 3s^3 + 3s^2 + 2s + 1 = 0$

```

Command Window
Ingresar los valores de los coeficientes del polinomio: [1 3 3 2 1]

Tabla de Routh-Hurwitz:

a =

    1.0000    3.0000    1.0000
    3.0000    2.0000         0
    2.3333    1.0000         0
    0.7143         0         0
    1.0000         0         0

Número de polos = 0
¿Necesitas las raíces del sistema? S/N s

Raíces de los coeficientes polinomiales:

Raíces =

-1.7549 + 0.0000i
-1.0000 + 0.0000i
-0.1226 + 0.7449i
-0.1226 - 0.7449i

fx >> |
    
```

Figura 146. Tabla de Routh-Hurwitz y valor de las raíces.

Se puede observar en la primera columna que no hay un cambio de signo en ninguno de los valores obtenidos, por lo tanto, se concluye que existe **estabilidad** en el sistema. Se muestran las raíces.

### Ejemplo 5:

Esta vez utilizaremos:  $2s^6 + 4s^5 + 3s^3 + 1 = 0$

Obteniendo los siguientes valores:

```
Command Window
Ingresar los valores de los coeficientes del polinomio: [2 4 0 3 0 0 1]

Tabla de Routh-Hurwitz:

a =

    2.0000    0    0    1.0000
    4.0000    3.0000    0    0
   -1.5000    0    1.0000    0
    3.0000    2.6667    0    0
    1.3333    1.0000    0    0
    0.4167    0    0    0
    1.0000    0    0    0

Número de polos = 2
¿Necesitas las raíces del sistema? S/N s

Raíces de los coeficientes polinomiales:

Raíces =

-2.2804 + 0.0000i
-0.0117 + 0.8822i
-0.0117 - 0.8822i
-0.6260 + 0.0000i
 0.4649 + 0.4836i
 0.4649 - 0.4836i
```

Figura 147. Tabla de Routh-Hurwitz y valor de las raíces.

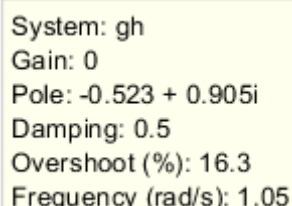
Este sistema se vuelve **inestable** al cambiar de 4 a -1.5 y de -1.5 a 3 en la columna principal.

Al contar con múltiples ceros podría creerse que es más fácil obtener que el sistema es estable, sin embargo, en este ejemplo no lo es, si bien se reducen las veces que los signos cambian no se evita.

## Lugar Geométrico de la Raíz

Para obtener el Lugar Geométrico de la Raíz en MATLAB, declararemos la función de transferencia de la cual queremos la respuesta y posteriormente mediante un comando podemos observarlo gráficamente.

Al dar clic en algún punto de las gráficas obtenidas, aparecerá un cuadro como el que se muestra a continuación:



```
System: gh
Gain: 0
Pole: -0.523 + 0.905i
Damping: 0.5
Overshoot (%): 16.3
Frequency (rad/s): 1.05
```

Figura 148. Datos de un punto en la gráfica.

Los datos mostrados son:

- System: La variable que fue graficada, en este caso, la función de transferencia “gh”.
- Gain: La ganancia del sistema.
- Pole: Las coordenadas del polo en ese punto seleccionado, mostrando el valor real y el valor imaginario (si es que lo hay).
- Damping: El coeficiente de amortiguamiento, que será menor a 1.
- Overshoot: El valor de sobre oscilación en el sistema.
- Frequency: La velocidad angular del sistema (rad/s).

Estos valores nos ayudan a interpretar la respuesta obtenida, recordemos que:

- Al encontrarse los Polos en el lado negativo del plano, el sistema será estable.
- Si se obtenemos un Zero positivo, significaría observar una respuesta inversa ante una entrada escalón.
- Los Polos cuya respuesta es imaginaria son oscilantes.
- El punto más alejado del origen, será el que tenga la respuesta más rápida ante una entrada escalón.

### Ejemplo 1:

Utilizaremos la función: 
$$\frac{2s^5+5s^4+3s^3+s^2+8s+3}{10s^5+7s^4+4s^3+s^2+5s+9}$$

Y declararemos de la siguiente manera: **gh=tf([2 5 3 1 8 3],[10 7 4 1 5 9])**

MATLAB nos mostrará la función acomodada y posteriormente utilizaremos el comando “**rlocus( )**”, dentro de los paréntesis se colocará el nombre asignado a la función, es decir, “gh”. Con clic derecho en la gráfica se agrega la cuadrícula “**Grid**” para observar mejor las posiciones de los polos y ceros.

Obteniendo la gráfica siguiente:

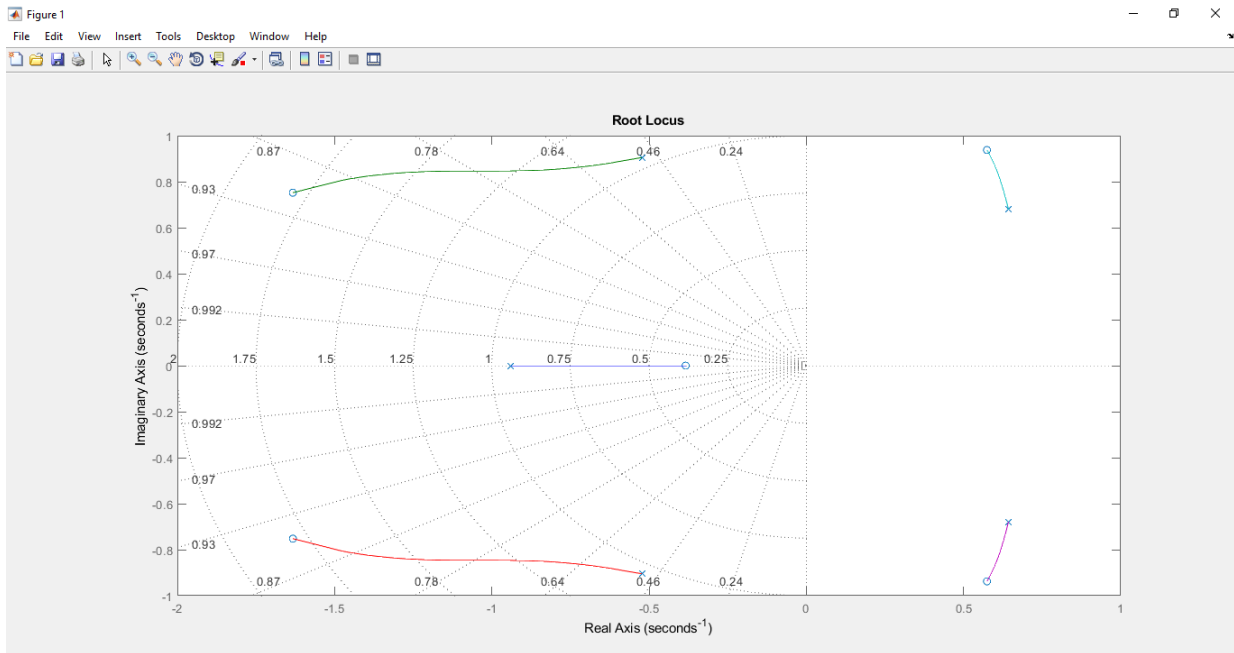


Figura 149. Gráfica del sistema.

A continuación, se seleccionarán algunos puntos en la gráfica y se interpretarán.

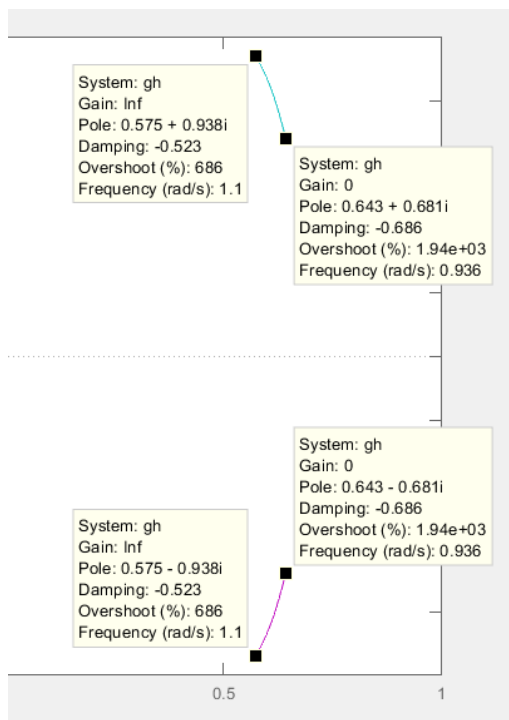


Figura 150. Información.

Recordemos que los polos se representan con una cruz y los ceros por un círculo.

Podemos observar que la ganancia de los ceros tiende a infinito, mientras que el valor de la ganancia de los polos es 0.

Como sabemos, los polos y ceros de un sistema deben ubicarse simétricamente respecto al eje Real, esto se puede comprobar observando las coordenadas de ubicación, sólo varía el signo cuando se encuentran en la parte negativa.

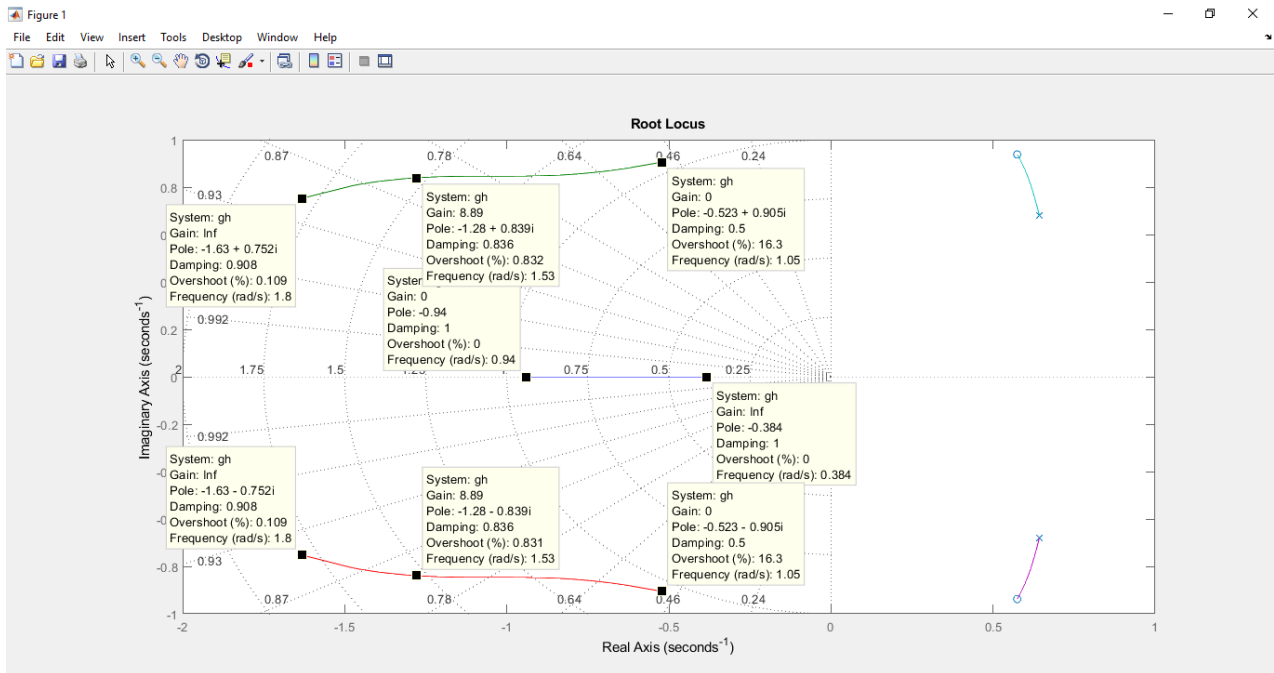


Figura 151. Gráfica de LGR.

Se puede observar que la simetría se conserva en la representación, los polos se encuentran ubicados en la coordenada  $-0.523 + 0.905i$  positiva y negativa y en  $-0.94$ , este polo no tiene componente imaginaria; todos los polos tienen un valor de ganancia 0.

Los ceros tienden a infinito, se encuentran en la coordenada  $-1.63 + 0.752i$  positiva y negativa, el otro cero se encuentra en el eje Real en  $-0.384$

Utilizando los comandos: **zeros=roots(gh.num{1})** y **polos=roots(gh.den{1})** podemos corroborar que están graficados correctamente, a continuación los valores obtenidos con los comandos:

```

Command Window

gh =

    2 s^5 + 5 s^4 + 3 s^3 + s^2 + 8 s + 3
-----
   10 s^5 + 7 s^4 + 4 s^3 + s^2 + 5 s + 9

Continuous-time transfer function.

>> zeros=roots(gh.num{1})

zeros =

   -1.6328 + 0.7521i
   -1.6328 - 0.7521i
    0.5747 + 0.9378i
    0.5747 - 0.9378i
   -0.3837 + 0.0000i

>> polos=roots(gh.den{1})

polos =

   -0.9396 + 0.0000i
   -0.5229 + 0.9050i
   -0.5229 - 0.9050i
    0.6427 + 0.6810i
    0.6427 - 0.6810i
  
```

Figura 152. Polos y ceros obtenidos con comando.

## Ejemplo 2:

Utilizaremos la función: 
$$\frac{2s^5 + s^4 + 5s + 7}{3s^5 + s^4 + 2s^2 + 5s + 8}$$

Se declarará de la siguiente manera: **gh=tf([2 1 0 0 5 7],[3 1 0 2 5 8])** y se utilizarán los comandos: **zeros=roots(gh.num{1})** y **polos=roots(gh.den{1})** para observar los valores de sus polos y ceros antes de representarlos gráficamente.

```
Command Window
gh =

      2 s^5 + s^4 + 5 s + 7
-----
      3 s^5 + s^4 + 2 s^2 + 5 s + 8

Continuous-time transfer function.

>> zeros=roots(gh.num{1})

zeros =

    1.0200 + 0.9310i
    1.0200 - 0.9310i
   -0.7292 + 1.0794i
   -0.7292 - 1.0794i
   -1.0816 + 0.0000i

>> polos=roots(gh.den{1})

polos =

    0.9591 + 0.9324i
    0.9591 - 0.9324i
   -1.1783 + 0.0000i
   -0.5366 + 0.9885i
   -0.5366 - 0.9885i
```

Figura 153. Polos y ceros de la función.

A continuación, mediante el comando **rlocus(gh)** obtendremos la respuesta gráfica y se agregará la cuadrícula.

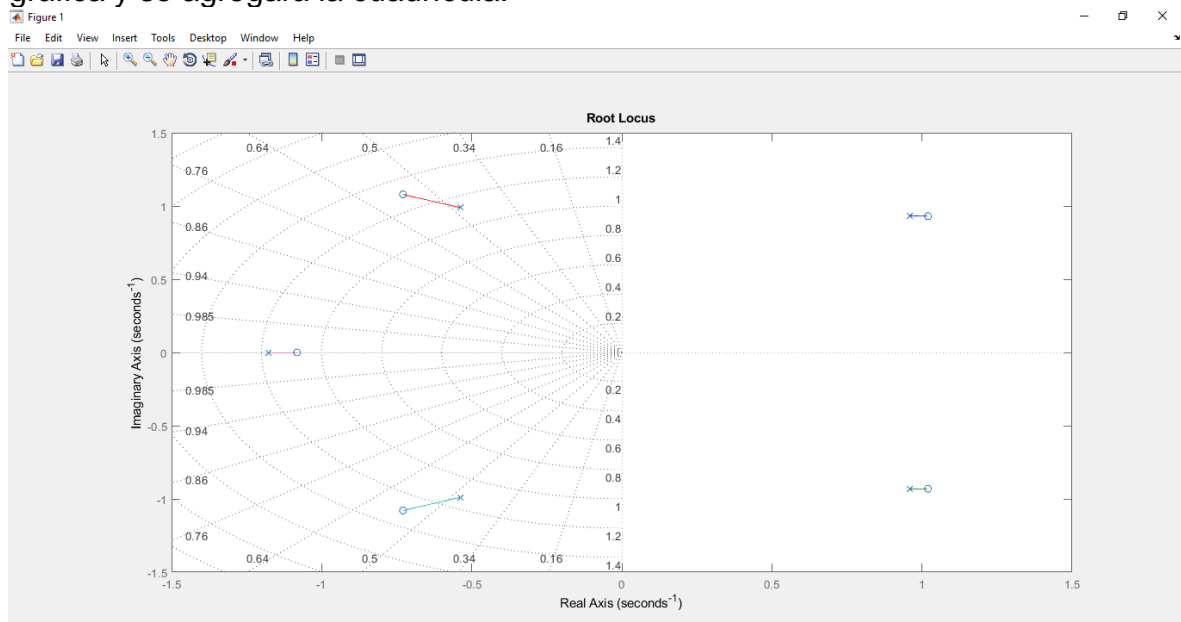


Figura 154. Representación.

Colocando el cursor en cada polo y cero, obtenemos los siguientes datos:

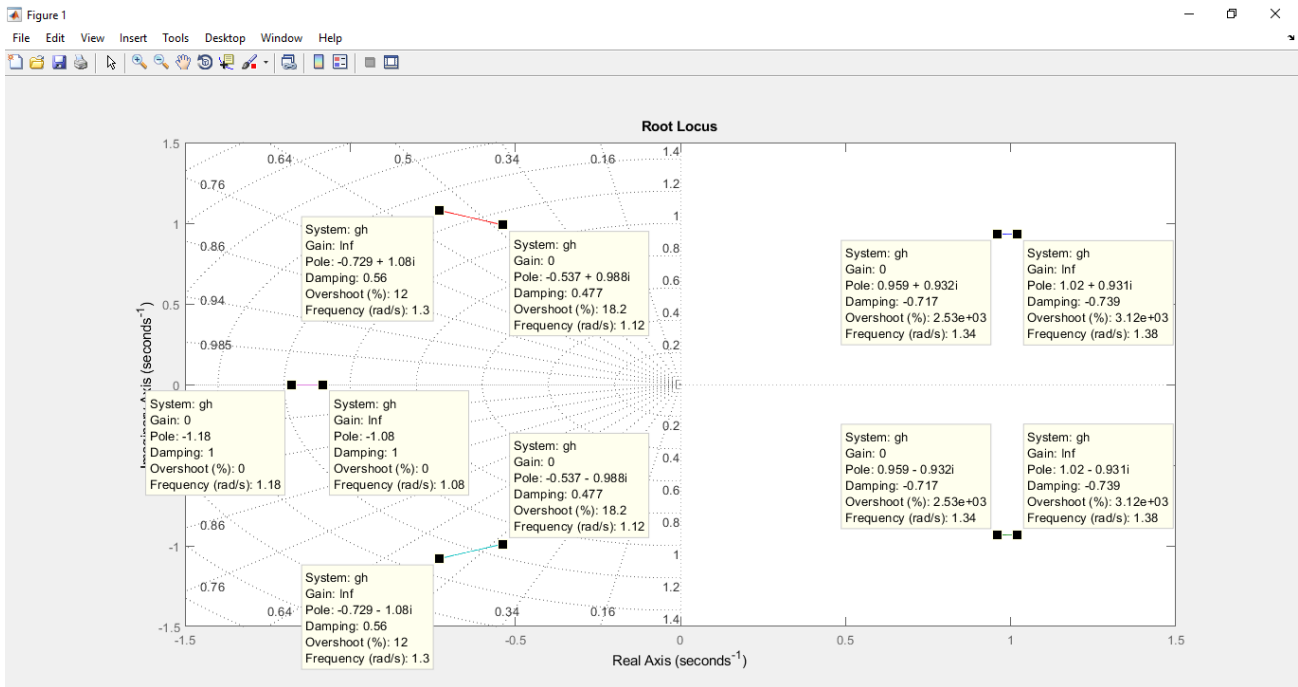


Figura 155. Datos de polos y ceros.

Como se puede observar en la imagen anterior, todos los ceros tienden a una ganancia infinita y los polos a una ganancia igual a cero. Se puede observar cómo existe simetría en las líneas obtenidas. Existen un cero y un polo localizados en el eje real en el valor de -1.18.

### Ejemplo 3:

Para la función de transferencia:  $\frac{s^4+3s^3+3s^2+2s+1}{s^4+3s^3+3s^2+3s+1}$

Se debe ingresar de la siguiente manera: **gh=tf([1 3 3 2 1],[1 3 3 3 1])** y con el comando **rlocus(gh)** se obtendrá la ubicación de los polos y ceros. Se activará la cuadrícula "Grid".



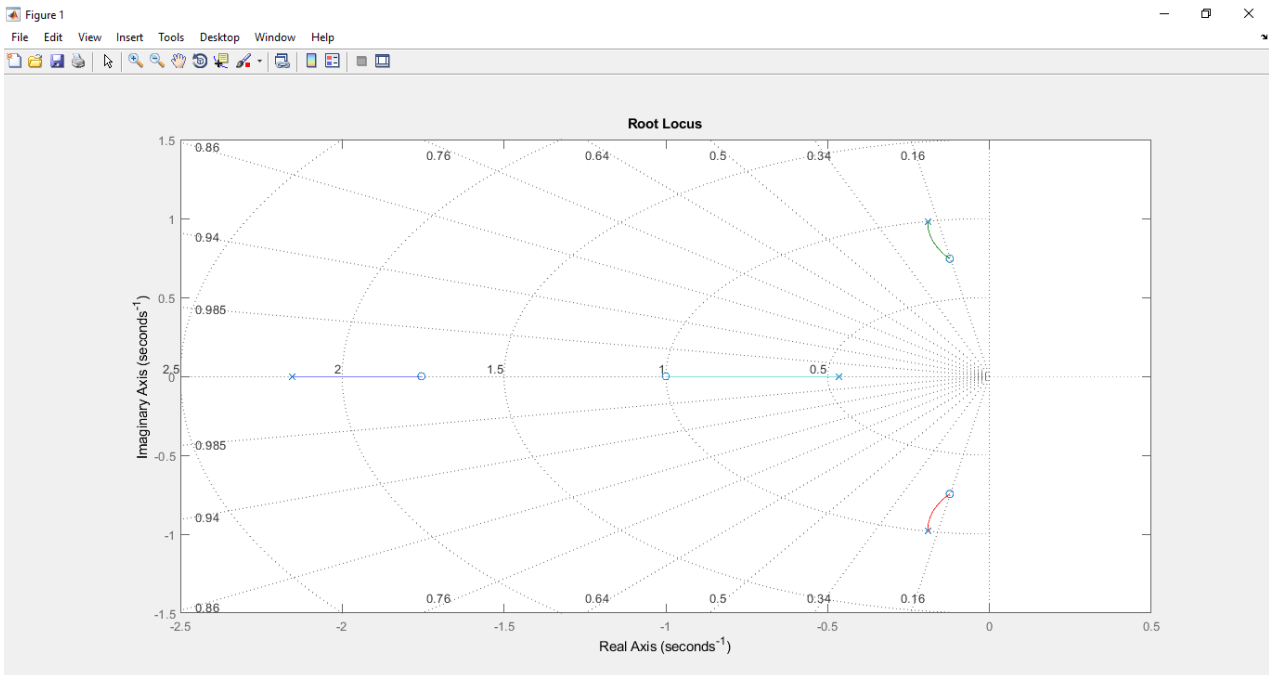


Figura 156. Gráfica de polos y ceros.

A continuación, seleccionaremos los polos y ceros graficados para conocer su ubicación y sus valores.

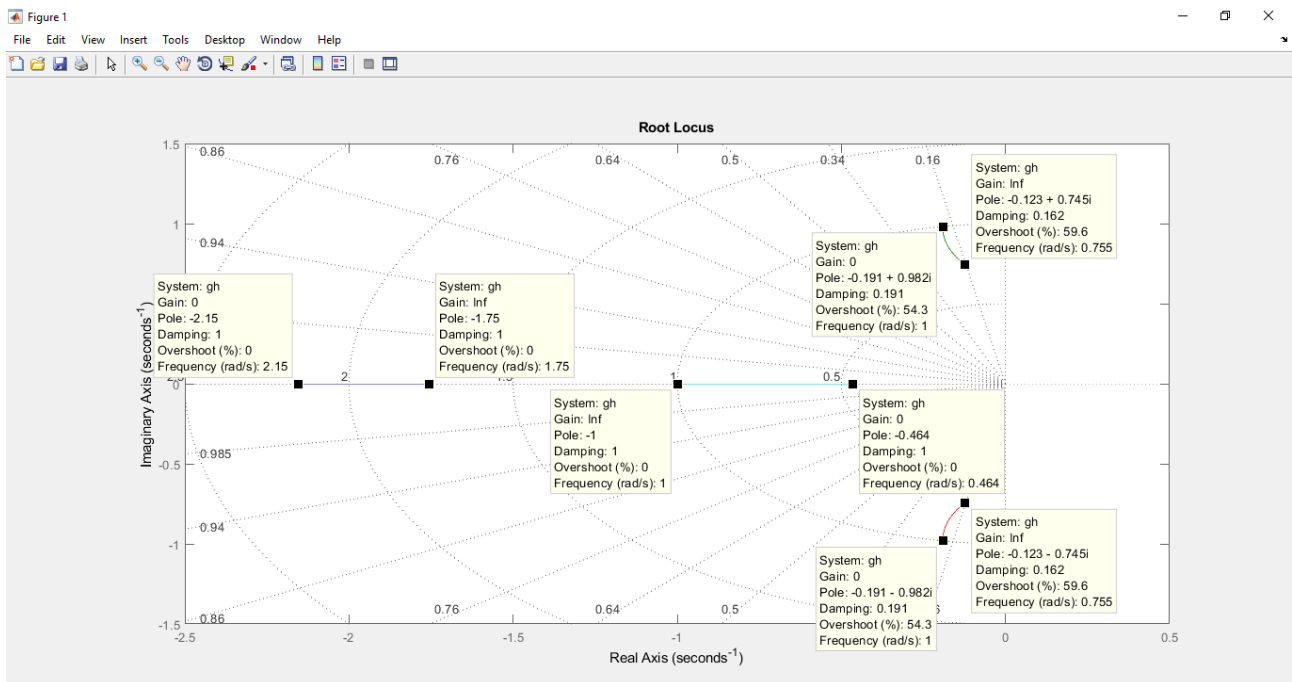


Figura 157. Datos de polos y ceros.

En este caso sólo un par de líneas no se encuentran en el eje real, conservan la simetría de la que se ha estado hablando.

Podemos notar que todos los polos se encuentran del lado negativo, esto significaría tener un sistema estable. Mediante el método de Routh-Hurwitz se obtiene el siguiente resultado:

```

Command Window

input vector of your system coefficients:
i.e. [an an-1 an-2 ... a0] = [1 3 3 3 1]

Routh-Hurwitz Table:

rhTable =

    1.0000    3.0000    1.0000
    3.0000    3.0000         0
    2.0000    1.0000         0
    1.5000         0         0
    1.0000         0         0

~~~~~> it is a stable system! <~~~~~

Number of right hand side poles = 0
fx Do you want roots of system be shown? Y/N |

```

Figura 158. Comprobación de la estabilidad el sistema.

Al no haber cambios de signo en la línea principal se concluye que el sistema es estable.

Finalmente se utilizarán los comandos: **zeros=roots(gh.num{1})** y **polos=roots(gh.den{1})** para observar los valores de sus polos y ceros y corroborar que están bien ubicados en la gráfica.

```

Command Window

gh =

    s^4 + 3 s^3 + 3 s^2 + 2 s + 1
    -----
    s^4 + 3 s^3 + 3 s^2 + 3 s + 1

Continuous-time transfer function.

>> rlocus(gh)
>> zeros=roots(gh.num{1})

zeros =

    -1.7549 + 0.0000i
    -1.0000 + 0.0000i
    -0.1226 + 0.7449i
    -0.1226 - 0.7449i

>> polos=roots(gh.den{1})

polos =

    -2.1537 + 0.0000i
    -0.1910 + 0.9816i
    -0.1910 - 0.9816i
    -0.4643 + 0.0000i

fx >>

```

Figura 159. Polos y ceros de la función.

## Bibliografía

- <https://es.mathworks.com/products/simulink.html>
- <https://la.mathworks.com/videos/introduction-to-simulink-100416.html>
- <https://la.mathworks.com/matlabcentral/answers/53100-not-enough-input-arguments>
- <https://la.mathworks.com/videos/modeling-simulation-and-testing-in-simulink-1499677822722.html>
- <https://catedra.ing.unlp.edu.ar/electrotecnia/cys/pdf/matlab2.pdf>
- [http://isa.uniovi.es/~arobles/AyC/S1\\_simulink.pdf](http://isa.uniovi.es/~arobles/AyC/S1_simulink.pdf)
- [http://olimpia.cuautitlan2.unam.mx/pagina\\_ingenieria/electronica/prac/practicas/2/M\\_Teoria\\_Control\\_Robotica\\_2020-2.pdf](http://olimpia.cuautitlan2.unam.mx/pagina_ingenieria/electronica/prac/practicas/2/M_Teoria_Control_Robotica_2020-2.pdf)
- <https://instrumentacionycontrol.net/resumen-p-i-d-lo-justo-y-necesario-que-debes-saber-y-que-nunca-entendiste/>
- <https://la.mathworks.com/discovery/pid-control.html>
- <https://www.mathworks.com/videos/control-design-made-easy-93051.html>
- <https://la.mathworks.com/matlabcentral/fileexchange/17483-routh-hurwitz-stability-criterion>
- [http://prof.usb.ve/ahoyo/Documentos/Ejercicios\\_Analisis\\_Estabilidad.pdf](http://prof.usb.ve/ahoyo/Documentos/Ejercicios_Analisis_Estabilidad.pdf)