

**UNIVERSIDAD NACIONAL  
AUTÓNOMA DE MÉXICO**  
**FACULTAD DE ESTUDIOS SUPERIORES**  
**CUAUTITLAN**  
**INGENIERÍA MECÁNICA ELÉCTRICA**



**TEORÍA DE CONTROL Y ROBÓTICA**

***USO DE MATLAB EN SISTEMAS DE  
CONTROL***

**ELABORADO POR**  
**LAURA ANDREA MONTOYA AYALA**

**MAYO 2020**

**Laura Andrea Montoya Ayala | IME**

1. INTRODUCCIÓN.....	4
2. MATLAB.....	5
2.1 FUNCIONES .....	5
2.2 HERRAMIENTAS ADICIONALES.....	5
3. ENTORNO DE TRABAJO DE MATLAB.....	6
4. MODELADO DE SISTEMAS EN MATLAB SIMULINK.....	7
4.1 EJEMPLO 1.....	15
4.2 EJEMPLO 2.....	15
4.3 EJEMPLO 3.....	15
5. FUNCION DE TRANSFERENCIA POR MEDIO DE DIAGRAMAS DE BLOQUES EN MATLAB.....	16
5.1 EJEMPLO 1.....	17
5.2 EJEMPLO 2.....	18
5.3 EJEMPLO 3.....	19
6. TRANSFORMADA DE LAPLACE EN MATLAB.....	20
6.1 EJEMPLO 1.....	21
6.2 EJEMPLO 2.....	22
6.3 EJEMPLO 3.....	23
7. ANTI TRANSFORMADA DE LAPLACE EN MATLAB.....	24
7.1 EJEMPLO 1.....	26
7.2 EJEMPLO 2.....	27
7.3 EJEMPLO 3.....	28
8. FRACCIONES PARCIALES EN MATLAB.....	29
8.1 EJEMPLO 1.....	32

8.2 EJEMPLO 2.....	34
8.3 EJEMPLO 3.....	36
9. SISTEMAS DE PRIMER ORDEN EN MATLAB.....	38
9.1 EJEMPLO 1.....	40
9.2 EJEMPLO 2.....	45
9.3 EJEMPLO 3.....	51
10.SISTEMAS DE SEGUNDO ORDEN EN MATLAB.....	56
10.1 EJEMPLO 1.....	58
10.2 EJEMPLO 2.....	62
10.3 EJEMPLO 3.....	66
11. CONTROLADORES EN MATLAB.....	70
11.1 CONTROLADOR PROPORCIONAL .....	70
11.1.1 EJEMPLO 1 .....	73
11.2 CONTROLADOR PROPORCIONAL INTEGRAL .....	75
11.2.1 EJEMPLO 2.....	78
11.3 CONTROLADOR PROPORCIONAL INTEGRAL DERIVATIVO .....	80
11.3.1 EJEMPLO 3.....	84
12. RESPUESTA EN FRECUENCIA EN MATLAB.....	87
12.1 DIAGRAMA DE BODE .....	87
12.1.1 EJEMPLO 1.....	89
12.1.2 EJEMPLO 2 .....	90
12.1.3 EJEMPLO 3 .....	91
12.1.4 EJERCICIO 1 .....	92
13.ESTABILIDAD DE UN SISTEMA.....	93
13.1 ESTABILIDAD POR EL MÉTODO DE ROUTH-HURWITZ...93	
13.1.1 EJEMPLO 1.....	100
13.1.2 EJEMPLO 2.....	101
13.1.3 EJEMPLO 3.....	102
13.2 ESTABILIDAD POR EL MÉTODO DE LGR.....	103
13.2.1 EJEMPLO 1.....	107
13.2.2 EJEMPLO 2 .....	109
13.2.3 EJEMPLO3.....	111
14. BIBLIOGRAFÍA.....	113

# 1. INTRODUCCIÓN

Los ingenieros de *sistemas de control* utilizan MATLAB® y Simulink® en todas las etapas de desarrollo, desde la modelización de la planta hasta el diseño y ajuste de los algoritmos de control y la lógica de supervisión, finalizando con la implementación gracias a la generación automática de código y la verificación, validación y comprobación del sistema. MATLAB y Simulink ofrecen:

- Un entorno de diagramas de bloques multidominio para modelizar la dinámica de la planta, diseñar algoritmos de control y ejecutar simulaciones de lazo cerrado.
- Modelización de plantas mediante herramientas de modelización física o identificación del sistema.
- Funciones prediseñadas y herramientas interactivas para analizar el sobreimpulso, el tiempo de subida, el margen de fase, el margen de ganancia y otras características de rendimiento y estabilidad en los dominios de la frecuencia y el tiempo.
- Lugar de raíces, diagramas de Bode, LQR, LQG, control robusto, control predictivo de modelos y otras técnicas de diseño y análisis.
- Ajuste automático de sistemas de control PID, de ganancia programada y SISO/MIMO arbitrarios.
- Modelización, diseño y simulación de la lógica de supervisión para llevar a cabo la planificación, el cambio de modo y la detección, aislamiento y recuperación de errores (FDIR).

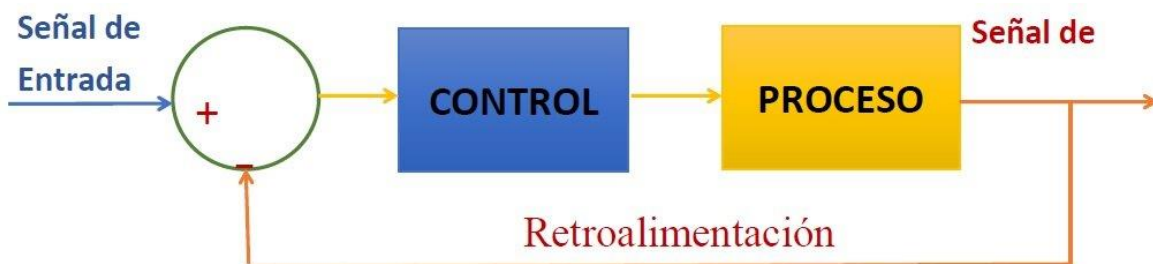


Figura 1. Sistema de control.

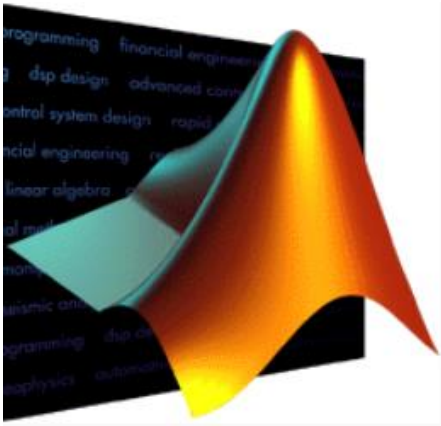


Figura 2. MATLAB.

## 2. MATLAB

Matrix Laboratory (“laboratorios de matrices”). Software matemático con entorno de desarrollo integrado (IDE) que tiene un lenguaje de programación propio (Lenguaje M) y es multiplataforma (Unix, Windows y Apple Mac Os X).

Software de un gran uso en Centros de Investigación y Desarrollo, así como en universidades.

### 2.1 FUNCIONES DE MATLAB

Dentro de sus principales funciones se encuentran:

- Manipulación de Matrices.
- La representación de datos y funciones.
- Implementación de algoritmos.
- Creación de interfaces de usuario (GUI).
- Comunicación con programas en otros lenguajes y con otros dispositivos.

### 2.2 HERRAMIENTAS ADICIONALES

MATLAB también cuenta con algunas herramientas adicionales:

- **Simulink** (plataforma de simulación multidominio).
- **GUIDE** (editor de interfaces de usuario - GUI).

Y también se pueden ampliar sus capacidades con las cajas de herramientas de MATLAB , y con los paquetes de bloques de Simulink.

## *Cajas de herramientas y paquetes de bloques*

Las más de 35 cajas de herramientas y paquetes de bloques agrupan las funcionalidades de MATLAB, estas se clasifican en las siguientes categorías.

<b>MATLAB (Cajas de herramientas)</b>	<b>Simulink</b>
Matemáticas y Optimización	Modelado de punto fijo
Estadística y Análisis de datos	Modelado basado en eventos
Diseño de sistemas de control y análisis	Modelado físico
Procesado de señal y comunicaciones	Gráficos de simulación
Procesado de imagen	Diseño de sistemas de control y análisis
Pruebas y medidas	Procesado de señal y comunicaciones
Biología computacional	Generación de código
Modelado y análisis financiero	Prototipos de control rápido y SW/HW HIL
Desarrollo de aplicaciones	Tarjetas integradas
Informes y conexión a bases de datos	Verificación, validación y comprobación
Compiler	Verificación, validación del código y desarrollo de ejecutables

*Tabla 1. Cajas de herramientas y paquetes de bloques.*

### 3. ENTORNO DE TRABAJO DE MATLAB

Matlab consiste en un entorno de ventanas con cuatro partes:

- **Command Window:** Es la ventana en la que se escriben las instrucciones que se quieren ejecutar.
- **Current Folder:** Muestra el contenido de la carpeta de trabajo. La dirección de la carpeta de trabajo se puede cambiar mediante la barra desplegable que aparece encima de las ventanas.
- **Workspace:** La ventana Workspace muestra información sobre las variables y objetos definidos.
- **Editor / Command History:** Permite ingresar una serie de comandos que se ejecutaran de manera secuencial de tal manera que nos evita ejecutar cada comando de manera individual. Esta ventana también puede mostrarlos últimos comandos (instrucciones) ejecutados.

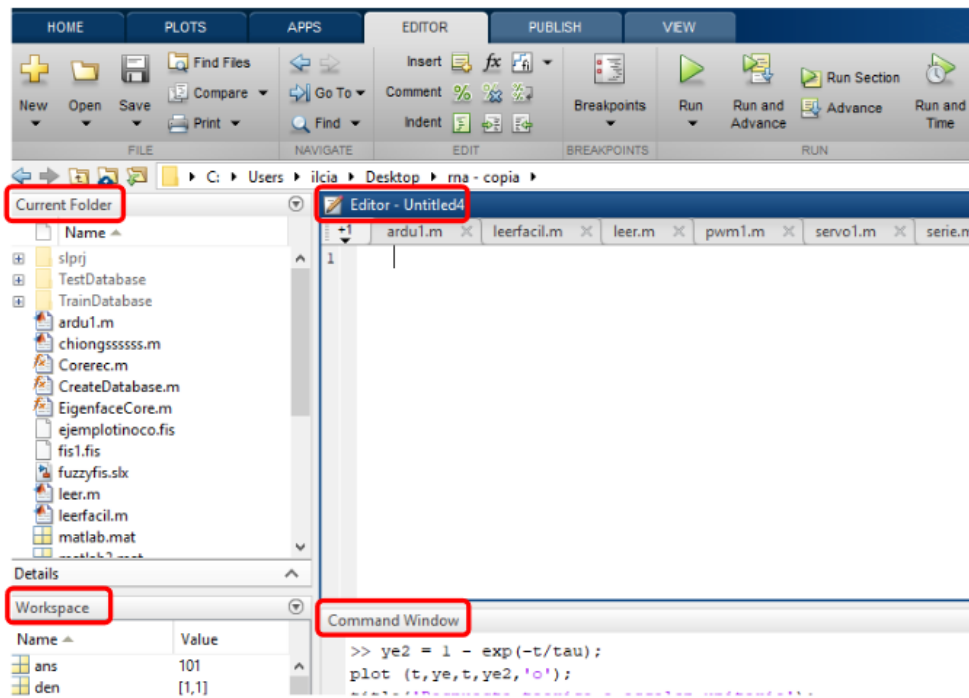


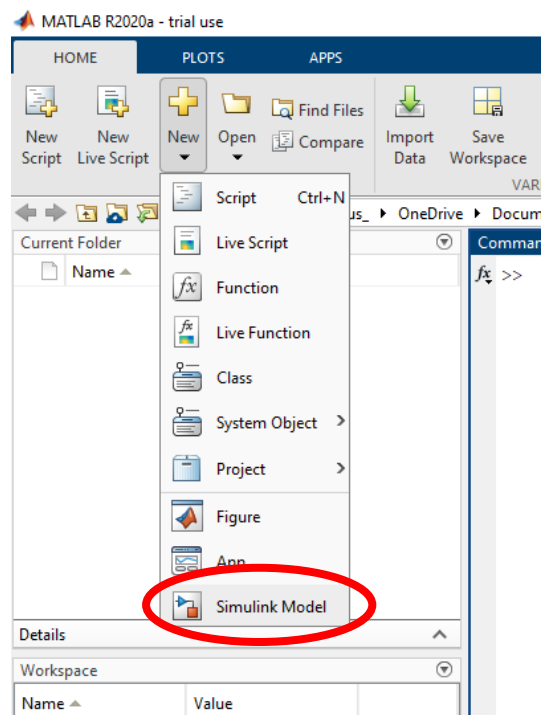
Figura 3. Entorno de trabajo de MATLAB.

## 4. MODELADO DE SISTEMAS EN MATLAB SIMULINK

**SIMULINK** es una toolbox especial de MATLAB que sirve para simular el comportamiento de los sistemas dinámicos. Puede simular sistemas lineales y no lineales, modelos en tiempo continuo y tiempo discreto y sistemas híbridos de todos los anteriores. Es un entorno gráfico en el cual el modelo a simular se construye clicando y arrastrando los diferentes bloques que lo constituyen. Los modelos SIMULINK se guardan en ficheros con extensión \*.mdl.

- *Modelado de sistemas por Diagrama de bloques.*

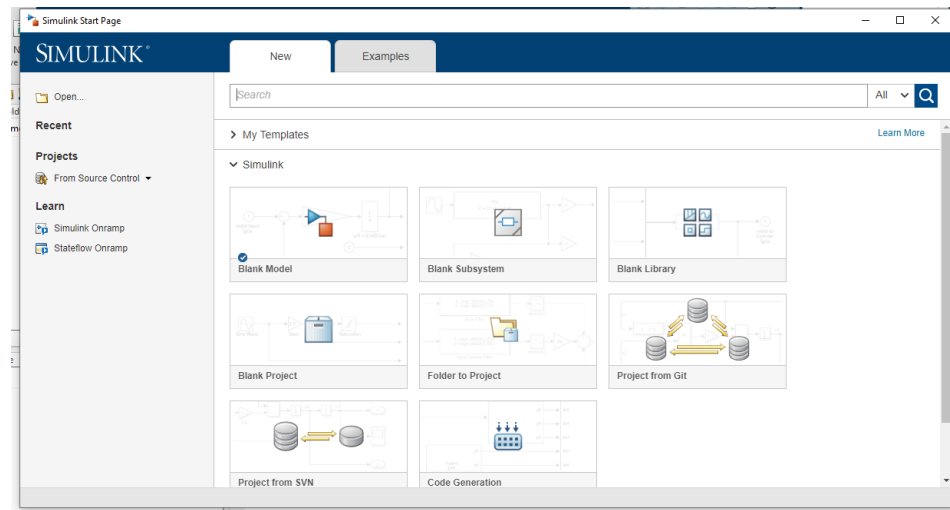
1. En el entorno de trabajo de Matlab, abrimos Simulink.



*Figura 4. Simulink Model.*

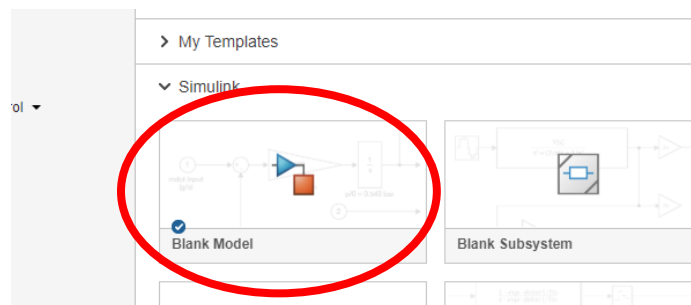


2. Tenemos el entorno de trabajo de simulink.



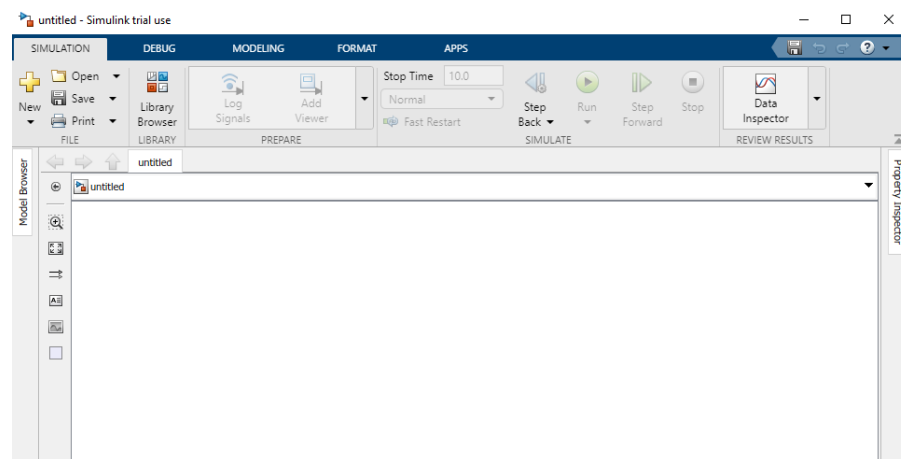
*Figura 4.1 Entorno de trabajo de Simulink Model*

3. Seleccionamos “Blank model” para empezar a modelar nuestro sistema



*Figura 4.2 Blank model.*

4. Se abrirá el entorno de trabajo de Blank Model



*Figura 4.3 Entorno de Blank model.*

5. En la parte superior izquierda se encuentra la librería.

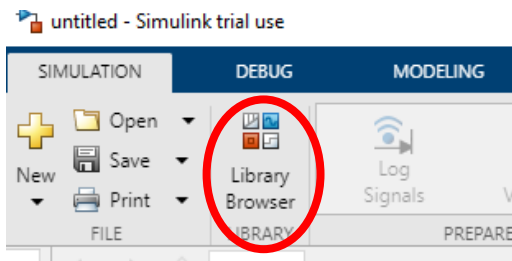


Figura 4.4 Librería de Simulink.

6. Se van a utilizar las dos librerías que aparecen al principio “Commonly Used Blocks” y “Continuous”.

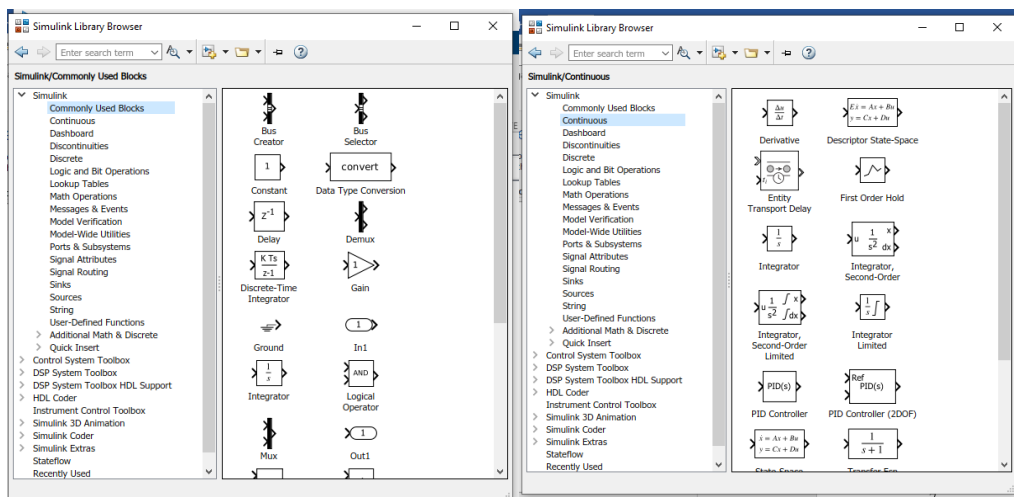


Figura 4.5 Librerías de Simulink.

7. En la librería “Commonly Used Blocks” aparecen la entrada y salida del sistema.

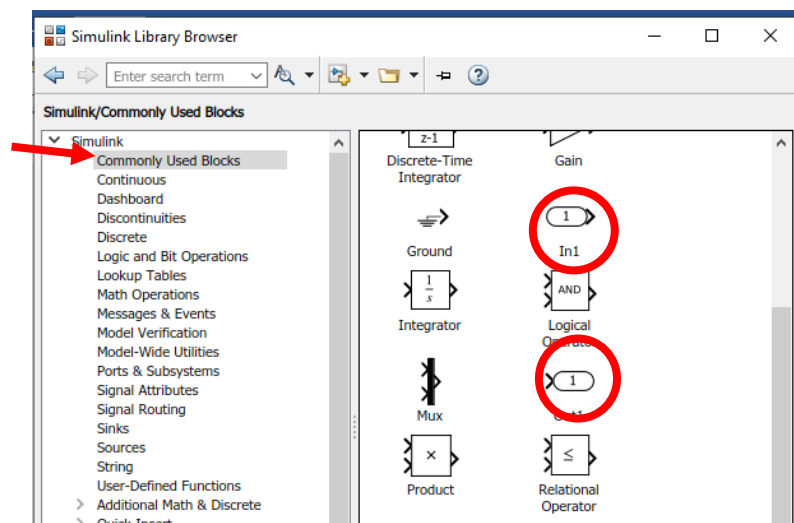


Figura 4.6 Librería Commonly Used Blocks.

8. En “Commonly Used Blocks” (Bloques de uso común), se encuentra la opción para introducir el sumador.

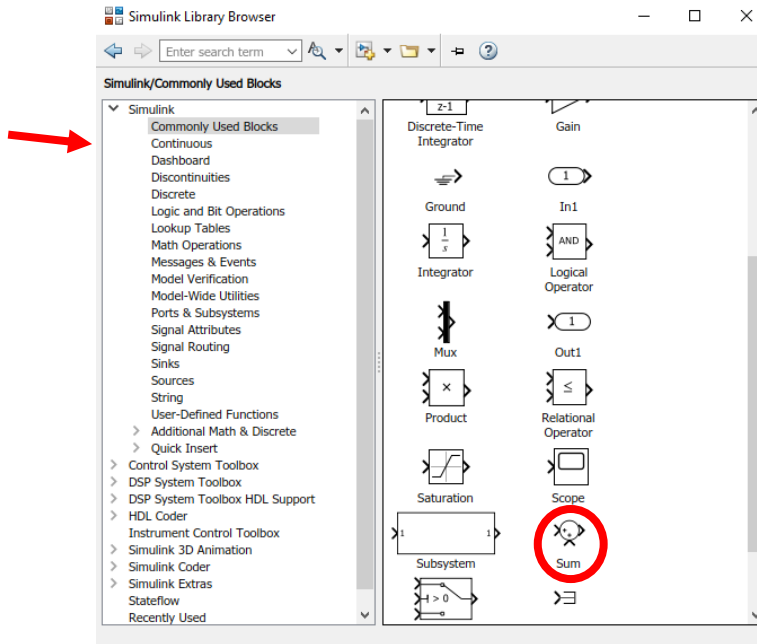


Figura 4.7 Librería Commonly Used Blocks.

9. En “Continuous” se encuentra la opción para meter la función de transferencia del sistema.

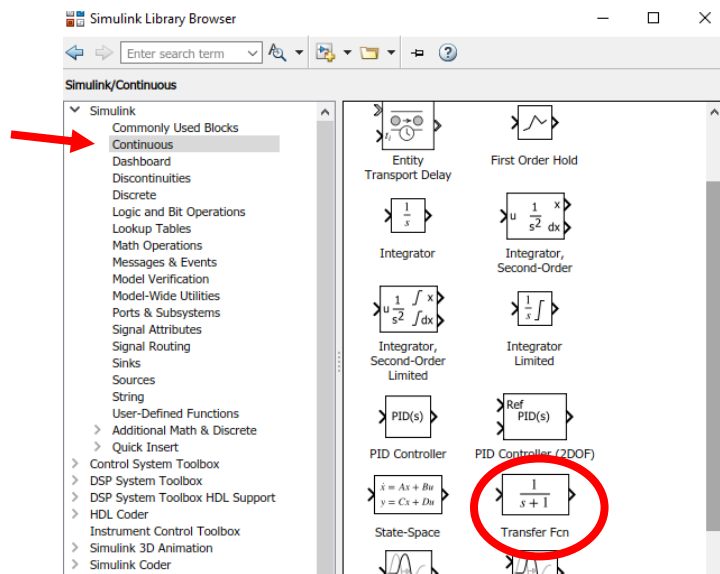


Figura 4.8 Librería Continuous.

10. Se inserta cada componente del sistema.

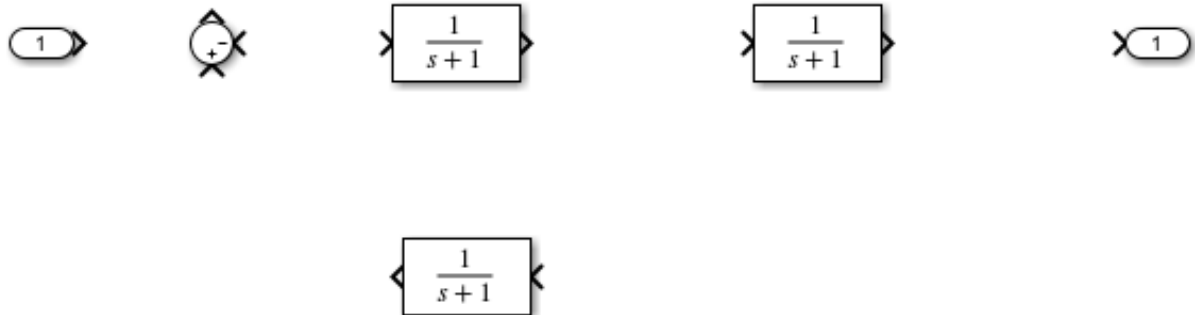


Figura 4.9 Construcción del sistema de control en simulink.

11. Para editar el sumador (cambiar los signos) se le da doble click derecho para o con “CTRL r” para cambiar el sentido.

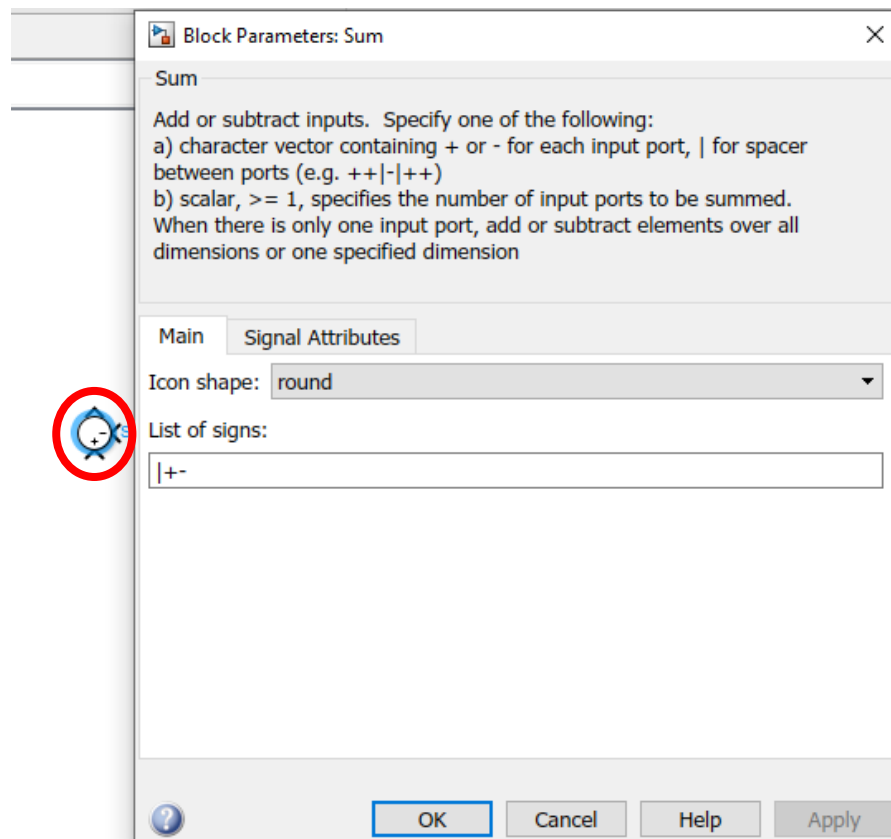


Figura 4.10 Edición del sumador.

12. Para editar la función de transferencia se le da doble clic derecho y para cambiar el sentido “ctrl r”.

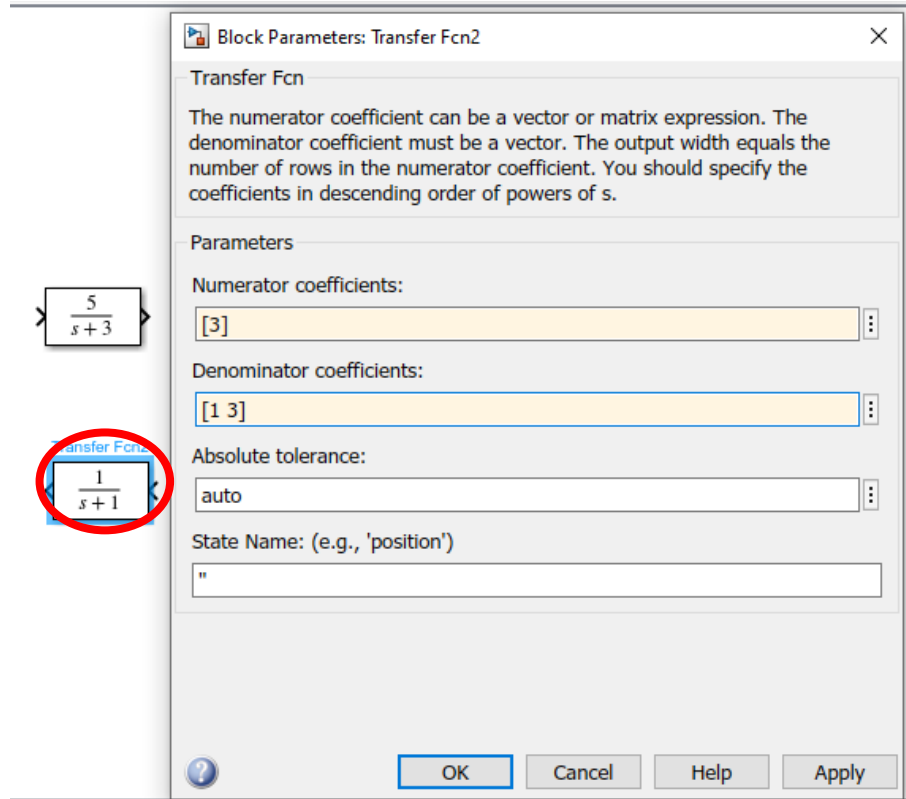


Figura 4.11 Edición de la función de transferencia.

13. Se unen los componentes del sistema.

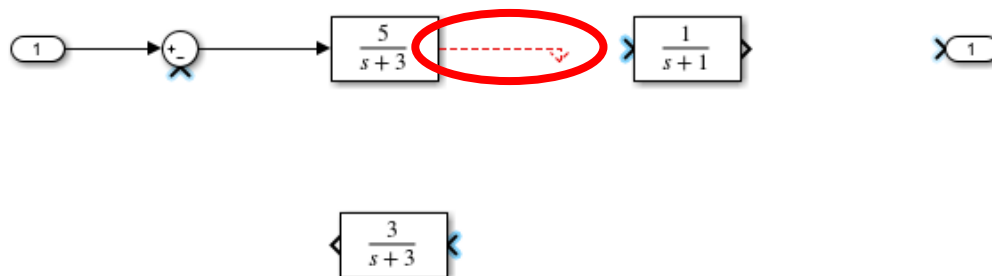
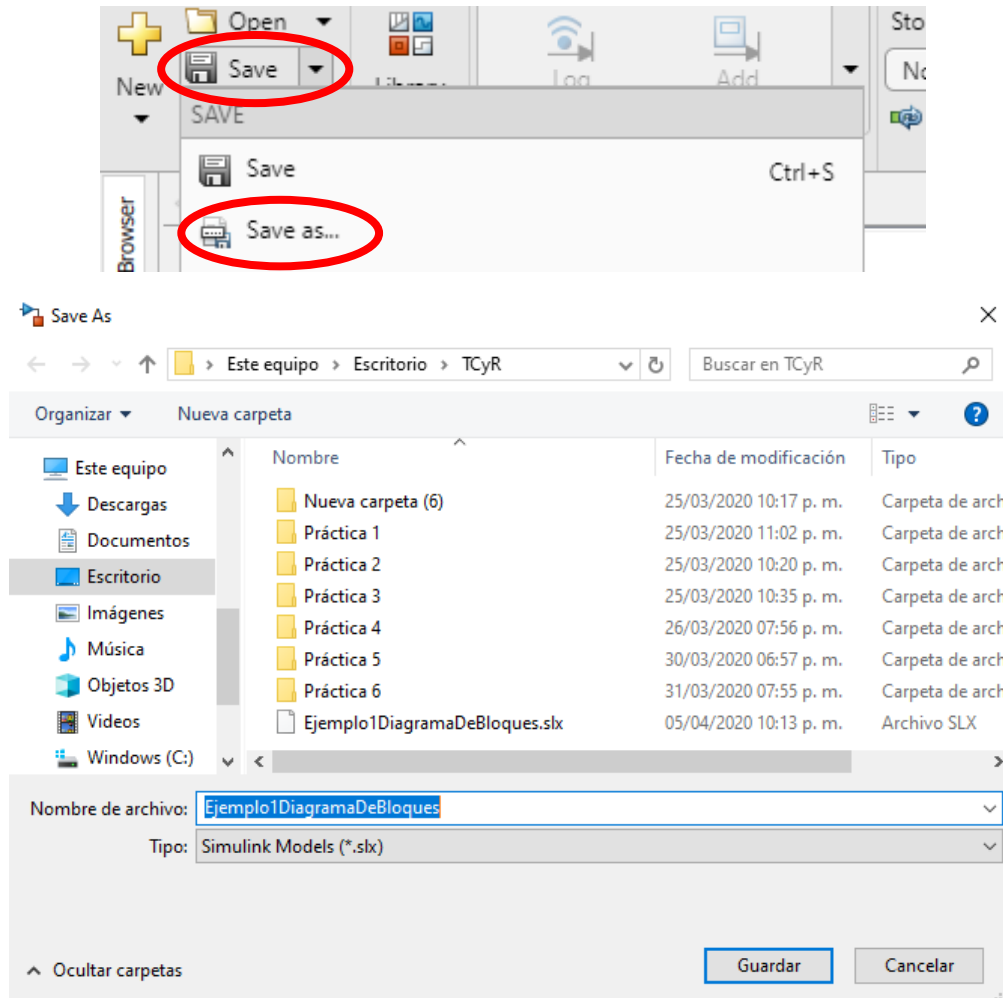


Figura 4.12 Construcción del sistema de control en Simulink.

14. Finalmente se debe de guardar el archivo para posteriormente utilizarlo en MATLAB.



*Figura 4.13 Archivo.*

## 4.1 EJEMPLO 1

“Ejemplo1DiagramaDeBloques”

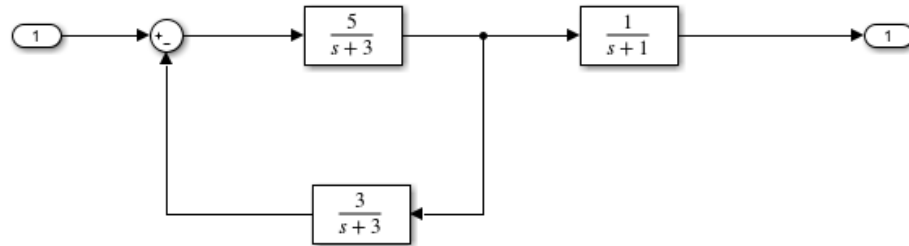


Figura 4.14 Ejemplo 1: Simulink.

## 4.2 EJEMPLO 2

“Ejemplo2DiagramaDeBloques”

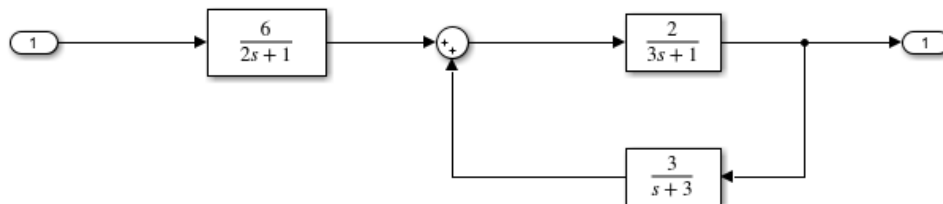


Figura 4.15 Ejemplo 2: Simulink.

## EJEMPLO 3

“Ejemplo3DiagramaDeBloques”

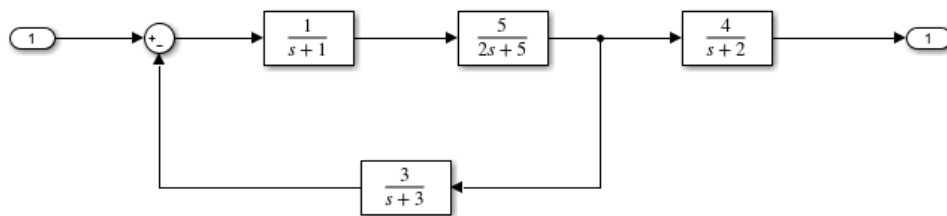


Figura 4.16 Ejemplo 3: Simulink.

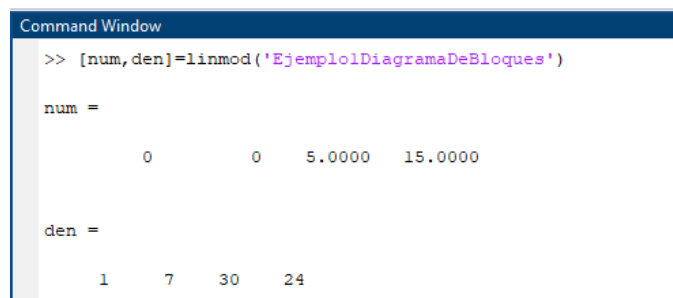
## 5. FUNCION DE TRANSFERENCIA POR MEDIO DE DIAGRAMAS DE BLOQUES EN MATLAB

Es importante el modelado de los sistemas por medio de Simulink para poder obtener la función de transferencia por medio de MATLAB.

1. En el entorno de trabajo de MATLAB en “Command Window” se va a escribir el siguiente comando, con el nombre del archivo que contiene del diagrama de bloques:

***[num,den]=linmod('Ejemplo1DiagramaDeBloques')***

Con este comando vamos a obtener los valores del numerador y del denominador de nuestra función de transferencia.

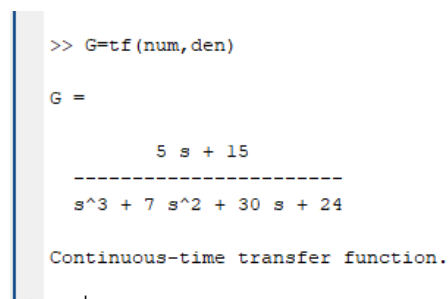


```
Command Window
>> [num,den]=linmod('Ejemplo1DiagramaDeBloques')
num =
      0      0  5.0000 15.0000
den =
      1      7  30  24
```

*Figura 5. Comandos para la función de transferencia en Matlab.*

2. Para visualizar bien la función de transferencia se va a escribir el siguiente comando:

***G=tf(num,den)***



```
>> G=tf(num,den)
G =
      5 s + 15
-----
s^3 + 7 s^2 + 30 s + 24
Continuous-time transfer function.
```

*Figura 5.1 Función de transferencia en Matlab.*



## EJEMPLO 1

“Ejemplo1DiagramaDeBloques”

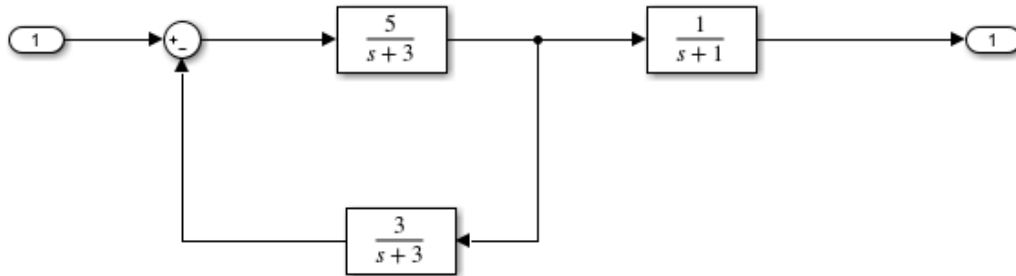


Figura 5.2 Ejemplo1: Función de transferencia por diagrama de bloques” Simulink”.

```
Command Window
>> [num,den]=linmod('Ejemplo1DiagramaDeBloques')

num =

      0      0  5.0000  15.0000

den =

      1      7     30     24

>> G=tf(num,den)

G =

      5 s + 15
-----
s^3 + 7 s^2 + 30 s + 24

Continuous-time transfer function.
```

Figura 5.3 Ejemplo1: Función de transferencia por diagrama de bloques” Matlab”.

## EJEMPLO 2

“Ejemplo2DiagramaDeBloques”

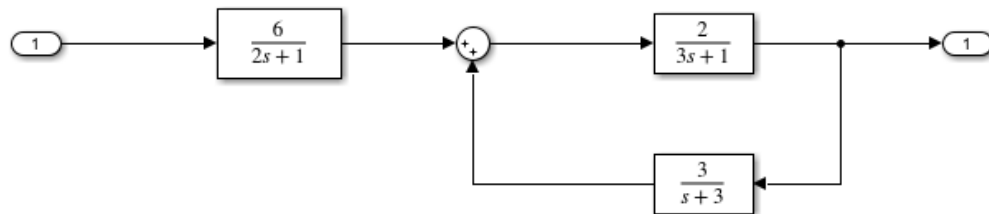


Figura 5.4 Ejemplo 2: Función de transferencia por diagrama de bloques” Simulink”.

```
Command Window
>> [num,den]=linmod('Ejemplo2DiagramaDeBloques')

num =
    0    0    2    6

den =
    1.0000    3.8333    0.6667   -0.5000

>> G=tf(num,den)

G =
          2 s + 6
-----
s^3 + 3.833 s^2 + 0.6667 s - 0.5

Continuous-time transfer function.
```

Figura 5.5 Ejemplo 2: Función de transferencia por diagrama de bloques” Matlab”.

### EJEMPLO 3

“Ejemplo3DiagramaDeBloques”

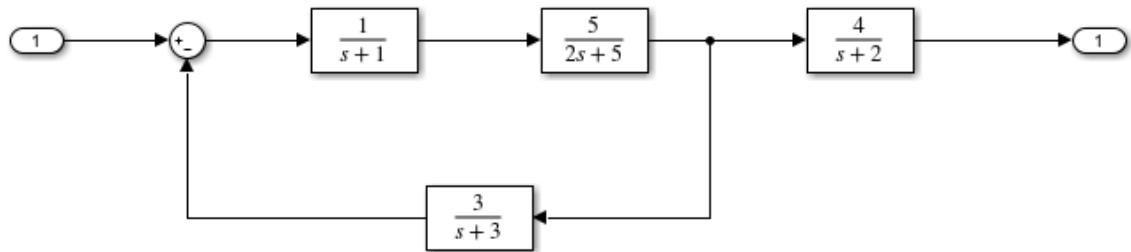


Figura 5.6 Ejemplo 3: Función de transferencia por diagrama de bloques” Simulink”.

```
Command Window
>> [num,den]=linmod('Ejemplo1DiagramaDeBloques')

num =

    0    0    5.0000    15.0000

den =

    1    7    30    24

>> G=tf(num,den)

G =

      5 s + 15
-----
s^3 + 7 s^2 + 30 s + 24

Continuous-time transfer function.

fx >>
```

Figura 5.7 Ejemplo3: Función de transferencia por diagrama de bloques” Matlab”.

## 6. TRANSFORMADA DE LAPLACE EN MATLAB

Para la resolución de Transformada de Laplace en MATLAB se van a seguir los siguientes pasos:

1. Primero se deben de declarar las variables, tanto de la función, como la de la Transformada de Laplace, con el siguiente comando:

***syms s t***

2. Posteriormente se ingresa la función:

***f=10\*exp(-5\*t)+3\*t\*exp(-t)-2\*sin(8\*t)\*exp(-t)***

```
>> syms s t
>> f=10*exp(-5*t)+3*t*exp(-t)-2*sin(8*t)*exp(-t)

f =

10*exp(-5*t) + 3*t*exp(-t) - 2*sin(8*t)*exp(-t)
```

*Figura 6. Función en Matlab.*

3. Para obtener la transformada de Laplace se utiliza el siguiente comando:

***laplace(f)***

4. Para visualizar mejor el resultado se utiliza el siguiente comando:

***Pretty(ans)***

```
>> laplace(f)

ans =

3/(s + 1)^2 + 10/(s + 5) - 16/((s + 1)^2 + 64)

>> pretty(ans)

      3          10          16
----- + ----- - -----
      2      s + 5          2
(s + 1)          (s + 1) + 64
```

*Figura 6.1 Comando Laplace.*

## 6.1 EJEMPLO 1

$$f = e^{-t} \text{sen}(2t)$$

$$f(s) = \frac{2}{(s+1)^2+4}$$

```
Command Window
>> f=exp(-t)*sin(2*t)

f =

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

>> laplace(f)

ans =

2/((s + 1)^2 + 4)

>> pretty(ans)

      2
-----
      2
(s + 1) + 4
```

Figura 6.2 Ejemplo1: Transformada de Laplace.

## EJEMPLO 2

$$f = \text{sen}\left(\frac{2t}{3}\right)$$

$$f(s) = \frac{2}{3(s^2 + \frac{4}{9})}$$

```
Command Window
>> syms st
>> f=sin(2*t/3)

f =

sin((2*t)/3)

>> pretty(f)
  / 2 t \
sin| --- |
  \ 3 /

>> laplace(f)

ans =

2/(3*(s^2 + 4/9))

>> pretty(ans)
      2
-----
  / 2  4 \
3 | s  + - |
  \      9 /
```

Figura 6.3 Ejemplo 2: Transformada de Laplace.

### EJEMPLO 3

$$f = \frac{\cos(7t)}{e^{5t}}$$

$$f(s) = \frac{s+5}{(s+5)^2+49}$$

```
Command Window

>> f=(cos(7*t))/(exp(5*t))

f =

cos(7*t)*exp(-5*t)

>> pretty(f)
cos(7 t) exp(-5 t)

>> laplace(f)

ans =

(s + 5)/((s + 5)^2 + 49)

>> pretty(ans)
      s + 5
-----
      2
(s + 5)  + 49
```

Figura 6.4 Ejemplo1: Transformada de Laplace.

## 7. ANTI TRANSFORMADA DE LA PLACE EN MATLAB

Para la resolución de Antitransformada de Laplace en MATLAB se van a seguir los siguientes pasos:

1. Primero se deben de declarar las variables, tanto de la función, como la de la Transformada de Laplace, con el siguiente comando:

***syms s t;***

2. Posteriormente se ingresa la función:

$$f = \frac{4s^2 + 2s + 14}{(s-1)(s^2+4)}$$

$$f = (4 * s^2 + 2 * s + 14) / ((s - 1)(s^2 + 4))$$

3. Para visualizar mejor el resultado se utiliza el siguiente comando:

***Pretty(f)***

```
Command Window
>> syms s t;
f = (4*s^2+2*s+14) / ((s-1)*(s^2+4))

f =

(4*s^2 + 2*s + 14) / ((s^2 + 4)*(s - 1))

>> pretty(f)
      2
  4 s  + 2 s + 14
  -----
      2
  (s  + 4) (s - 1)
```

Figura 7 Comando para ingresar una función en Matlab.



4. Para obtener la antitransformada de Laplace se utiliza el siguiente comando:

*ilaplace(f)*

```
Command Window
>> syms s t;
f = (4*s^2+2*s+14) / ((s-1)*(s^2+4))

f =

(4*s^2 + 2*s + 14) / ((s^2 + 4)*(s - 1))

>> pretty(f)
      2
  4 s  + 2 s + 14
  -----
      2
  (s  + 4) (s - 1)

>> ilaplace(f)

ans =

sin(2*t) + 4*exp(t)
```

*Figura 7.1 Comando antitransformada de Laplace.*

## 7.1 EJEMPLO 1

$$L^{-1}\left\{\frac{1}{s^2-2s+9}\right\}$$

$$f(t) = \frac{\sqrt{2} e^t \text{sen}(2\sqrt{2}t)}{4}$$

### Command Window

```
>> syms s t
>> f(s) = (1)/(s^2-2*s+9)

|
f(s) =

1/(s^2 - 2*s + 9)

>> pretty(f(s))
      1
-----
      2
s  - 2 s + 9

>> ilaplace(f(s))

ans =

(2^(1/2)*exp(t)*sin(2*2^(1/2)*t))/4

>> pretty(ans)
sqrt(2) exp(t) sin(2 sqrt(2) t)
-----
                        4
```

Figura 7.2 Ejemplo 1: Anti transformada de Laplace.

## 7.2 EJEMPLO 2

$$L^{-1}\left\{\frac{2s+3}{s^2+6s+13}\right\}$$

$$f(t) = e^{-3t} \left( \cos(2t) - \frac{3\sin(2t)}{4} \right)$$

```
Command Window
>> syms s t;
>> f(s)=(2*s+3)/(s^2+6*s+13)

f(s) =

(2*s + 3)/(s^2 + 6*s + 13)

>> pretty(f(s))
  2 s + 3
-----
  2
s  + 6 s + 13

>> ilaplace(f(s))

ans =

2*exp(-3*t)*(cos(2*t) - (3*sin(2*t))/4)

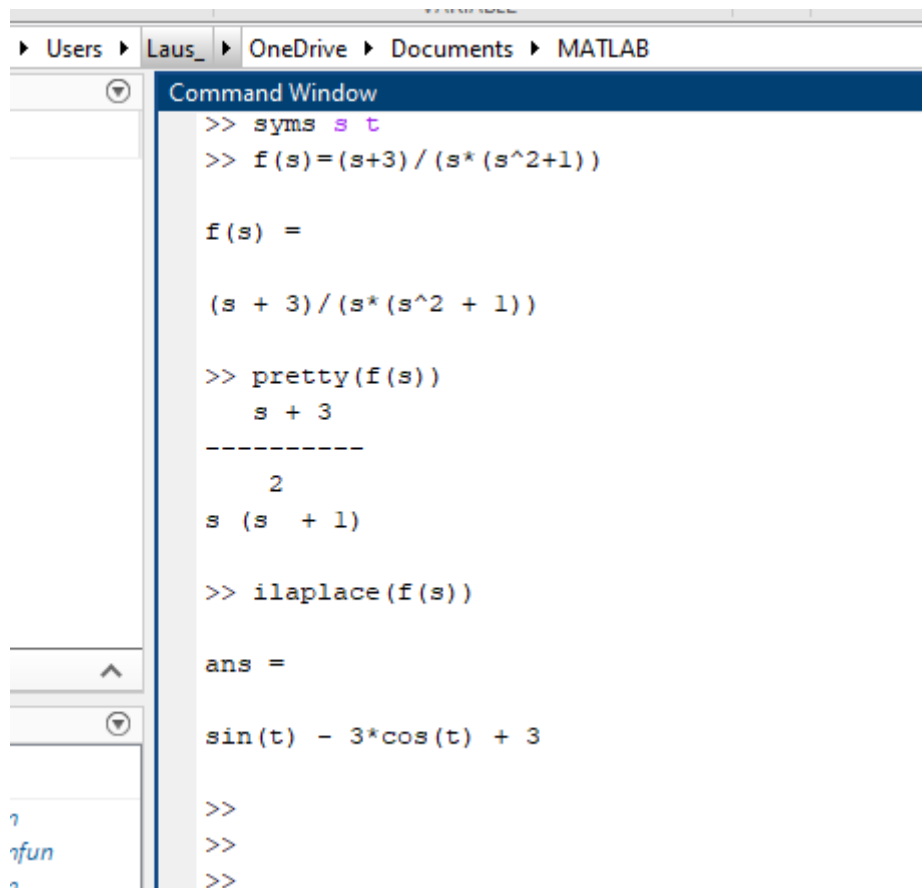
>> pretty(ans)
      /          sin(2 t) 3 \
exp(-3 t) | cos(2 t) - ----- | 2
          \          4      /
```

Figura 7.3 Ejemplo 2: Anti transformada de Laplace.

### 7.3 EJEMPLO 3

$$L^{-1}\left\{\frac{s+3}{s(s^2+1)}\right\}$$

$$f(t) = \sin(t) - 3\cos(t) + 3$$



```
Users \ Laus_ \ OneDrive \ Documents \ MATLAB
Command Window
>> syms s t
>> f(s)=(s+3)/(s*(s^2+1))

f(s) =

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

>> pretty(f(s))
      s + 3
-----
      2
s (s + 1)

>> ilaplace(f(s))

ans =

sin(t) - 3*cos(t) + 3

>>
>>
>>
```

Figura 7.4 Ejemplo 3: Anti transformada de Laplace.

## 8. FRACCIONES PARCIALES EN MATLAB

Existen diferentes métodos para la resolución de fracciones parciales por medio de MATLAB, a continuación, se desarrollará uno de ellos.

1. Se debe de declarar la variable que se estará utilizando, en este caso “x”, con el siguiente comando:

***syms x;***

2. Posteriormente se ingresa la función en la cual vamos a trabajar:

$$\frac{7}{x^2 + 3x - 10}$$

***f = (7) / (x^2+3\*x-10)***

```
>> syms x
f=(7)/(x^2+3*x-10)
f =
7/(x^2 + 3*x - 10)
```

*Figura 8. Función de transferencia.*

3. Después de ingresar la función vamos a declarar las variables que queremos encontrar del numerador, con el siguiente comando:

***syms A B***

4. Para factorizar el denominador de la función se introduce el siguiente comando, nombrando una nueva variable:

***c = factor(x^2+3\*x-10)***

```
>> syms A B
>> c=factor(x^2+3*x-10)
c =
[ x + 5, x - 2]
```

*Figura 8.1 Factorización del denominador.*

5. A las funciones del resultado anterior se van a nombrar con nuevas variables:

$$d = x + 5$$

$$e = x - 2$$

```
>> d=x+5
d =
x + 5
>> e=x-2
e =
x - 2
```

Figura 8.2 Variables de las funciones anteriores.

6. Después realizaremos la multiplicación de nuestras nuevas variables (d,e) por las variables que queremos conocer (A,B).

$$f = (d*e*A) / d + (d*e*B)/e$$

```
>> f=((d*e*A)/d)+((d*e*B)/e)
f =
A*(x - 2) + B*(x + 5)
```

Figura 8.3 Multiplicación de variables.

7. Con estos resultados podemos obtener nuestro sistema de ecuaciones.

$$\begin{aligned} 7 &= A(x - 2) + B(x + 5) \\ 7 &= Ax - 2A + Bx + 5B \\ 7 &= x(A + B) + (-2A + 5B) \end{aligned}$$

$$A + B = 0$$

$$-2A + 5B = 7$$

8. Con nuestro sistema de ecuaciones vamos a formar una matriz y la vamos a introducir a MATLAB con los siguientes comandos:

$$g = [1 \ 1; -2 \ 5]$$

$$h = [0; 7]$$

```
g =  
    1    1  
   -2    5  
  
>> h=[0;7]  
  
h =  
  
    0  
    7
```

Figura 8.4 Matriz en Matlab.

9. Posteriormente vamos a obtener la inversa de la matriz g, con el siguiente comando:

$$i = \text{inv}(g)$$

```
>> i=inv(g)  
  
i =  
  
    0.7143   -0.1429  
    0.2857    0.1429
```

Figura 8.4 Matriz inversa en Matlab.

10. Finalmente, para conocer los valores de A y B multiplicaremos i por h, con el siguiente comando:

$$j = i * h$$

```
>> j=i*h  
  
j =  
  
   -1.0000  
    1.0000
```

Figura 8.5 Valores de A y B.

## 8.1 EJEMPLO 1

$$\frac{7}{x^2+3x-10} = \frac{-1}{(x+5)} + \frac{1}{(x-2)}$$

```
Command Window
>> syms x
f=(7)/(x^2+3*x-10)

f =

7/(x^2 + 3*x - 10)

>>

syms A B
c=factor(x^2+3*x-10)

c =

[ x + 5, x - 2]

>> d=x+5

d =

x + 5

>> e=x-2

e =

x - 2
```

Figura 8.6. Ejemplo 1: Fracciones parciales.



```
>> f=( (d*e*A)/d)+( (d*e*B)/e)
f =
A*(x - 2) + B*(x + 5)
```

*Figura 8.7 Ejemplo 1: Fracciones parciales.*

$$7 = A(x - 2) + B(x + 5)$$

$$7 = Ax - 2A + Bx + 5B$$

$$7 = x(A + B) + (-2A + 5B)$$

$$A + B = 0$$

$$-2A + 5B = 7$$

```
>> g=[1 1;-2 5]
g =
     1     1
    -2     5

>> h=[0;7]
h =
     0
     7

>> i=inv(g)
i =
    0.7143   -0.1429
    0.2857    0.1429

>> j=i*h
j =
   -1.0000
    1.0000
```

*Figura 8.8 Ejemplo 1: Fracciones parciales.*

## 8.2 EJEMPLO 2

$$\frac{2x-7}{(6x^2-5x+1)} = \frac{5}{3x-1} - \frac{4}{2x-1}$$

```
Command Window
>> syms x
num=2*x-7
num =
2*x - 7
>> den=6*x^2-5*x+1
den =
6*x^2 - 5*x + 1
>> f=(num/den)
f =
(2*x - 7)/(6*x^2 - 5*x + 1)
>> syms A B
>> c=factor(den)
c =
[ 3*x - 1, 2*x - 1]
>> d=3*x-1
d =
3*x - 1
>> e=2*x-1
e =
2*x - 1
>> f=(d*e*A)/d +(d*e*B)/e
f =
A*(2*x - 1) + B*(3*x - 1)
```

Figura 8.9 Ejemplo 2: Fracciones parciales.

$$2x - 7 = 2Ax - A + 3Bx - B$$

$$2x - 7 = x(2A + 3B) + (-A - B)$$

$$2A + 3B = 2$$

$$A + B = 7$$

```

Command Window

>> g=[2 3;-1 -1]

g =

     2     3
    -1    -1

>> h=[2;]

h =

     2

>> h=[-2;-1]

h =

    -2
    -1

>> i=inv(g)

i =

    -1    -3
     1     2

>> j=i*h

j =

     5
    -4

```

Figura 8.10 Ejemplo 2: Fracciones parciales.

### 8.3 EJEMPLO 3

$$\frac{5}{x^2 - 6x + 5} = \frac{-1.25}{x - 1} + \frac{1.25}{x - 5}$$

```
Command Window
>> syms x
f=(5)/(x^2-6*x+5)

f =

5/(x^2 - 6*x + 5)

>> syms A B
>> c=factor(x^2-6*x+5)

c =

[ x - 1, x - 5]

>> d=x-1

d =

x - 1

>> e=x-5

e =

x - 5

>> f=((d*e*A)/d)+((d*e*B)/e)

f =

A*(x - 5) + B*(x - 1)
```

Figura 8.11 Ejemplo 3: Fracciones parciales

$$5 = A(x - 5) + B(x - 1)$$

$$5 = Ax - 5A + Bx - B$$

$$5 = x(A + B) + (-5A - B)$$

$$A + B = 0$$

$$-5A - B = 5$$

```

Command Window

>> g=[1 1;-5 -1]

g =

     1     1
    -5    -1

>> h=[0;5]

h =

     0
     5

>> i=inv(g)

i =

   -0.2500   -0.2500
    1.2500    0.2500

>> j=i*h

j =

   -1.2500
    1.2500
fx

```

Figura 8.12 Ejemplo 3: Fracciones parciales.

## 9. SISTEMAS DE PRIMER ORDEN EN MATLAB

La representación en forma de función de transferencia viene dada de manera general como:

$$\frac{k}{ts + 1}$$

1. Se debe de declarar las variables que se estará utilizando, en este caso “s, t”, con el siguiente comando:

```
syms s t
```

2. Posteriormente se agregan los valores del numerador y del denominador, con los siguientes comandos.

```
K = 1;  
tau = 1;  
num = K;  
den = [tau,1]
```

La respuesta a la entrada escalón unitario de entrada se obtiene con la función step. Para ello, se debe de definir un intervalo de simulación utilice los siguientes comandos:

```
t = [0:0.1:10];  
ye = step(num,den,t);
```

3. Para visualizar de manera gráfica la simulación se deben de ingresar los siguientes comandos:

```
plot(t,ye);  
title ('Respuesta a un escalon unitario');  
xlabel ('tiempo(seg)');  
grid;
```

4. Las dos características fundamentales de un sistema de primer orden son su ganancia estática K y su constante de tiempo  $\tau$ . La constante de tiempo es el tiempo que le toma a la señal alcanzar el 63% de la salida máxima.

La ganancia estática es el cociente entre la amplitud de salida y la de entrada en el régimen permanente. Estos valores se pueden comprobar directamente en la gráfica o analizando el vector de datos resultante. Para ello se deben escribir los siguientes comandos.

```
yRP = ye(length(ye));  
k = yRP  
n = 1;  
while ye(n) < 0.63*yRP  
n = n + 1;  
end  
tauEstim = 0.1*(n-1)
```

5. Con la siguiente instrucción, Matlab responde imprimiendo en la ventana de comandos el tiempo que le toma al sistema alcanzar el 63%.

```
fprintf("constante de tiempo:%f\n",tauEstim)
```

## 9.1 EJEMPLO 1

$$G(s) = \frac{1}{s+1}$$

```
Command Window
>> syms s t
>> K=1

K =

    1

>> tau=1

tau =

    1

>> num=K

num =

    1

>> den=[tau,1]

den =

    1    1

>> t=[0:0.1:10]

t =

Columns 1 through 5
    0    0.1000    0.2000    0.3000    0.4000

Columns 6 through 10
    0.5000    0.6000    0.7000    0.8000    0.9000
```

*Figura 9. Ejemplo 1: Sistemas de primer orden.*



Columns 11 through 15				
1.0000	1.1000	1.2000	1.3000	1.4000
Columns 16 through 20				
1.5000	1.6000	1.7000	1.8000	1.9000
Columns 21 through 25				
2.0000	2.1000	2.2000	2.3000	2.4000
Columns 26 through 30				
2.5000	2.6000	2.7000	2.8000	2.9000
Columns 31 through 35				
3.0000	3.1000	3.2000	3.3000	3.4000
Columns 36 through 40				
3.5000	3.6000	3.7000	3.8000	3.9000
Columns 41 through 45				
4.0000	4.1000	4.2000	4.3000	4.4000
Columns 46 through 50				
4.5000	4.6000	4.7000	4.8000	4.9000
Columns 51 through 55				
5.0000	5.1000	5.2000	5.3000	5.4000
Columns 56 through 60				
5.5000	5.6000	5.7000	5.8000	5.9000
Columns 61 through 65				
6.0000	6.1000	6.2000	6.3000	6.4000

*Figura 9.1 Ejemplo 1: Sistemas de primer orden.*

```

Columns 66 through 70
    6.5000    6.6000    6.7000    6.8000    6.9000

Columns 71 through 75
    7.0000    7.1000    7.2000    7.3000    7.4000

Columns 76 through 80
    7.5000    7.6000    7.7000    7.8000    7.9000

Columns 81 through 85
    8.0000    8.1000    8.2000    8.3000    8.4000

Columns 86 through 90
    8.5000    8.6000    8.7000    8.8000    8.9000

Columns 91 through 95
    9.0000    9.1000    9.2000    9.3000    9.4000

Columns 96 through 100
    9.5000    9.6000    9.7000    9.8000    9.9000

Column 101
    10.0000

>> ye=step(num,den,t)

```

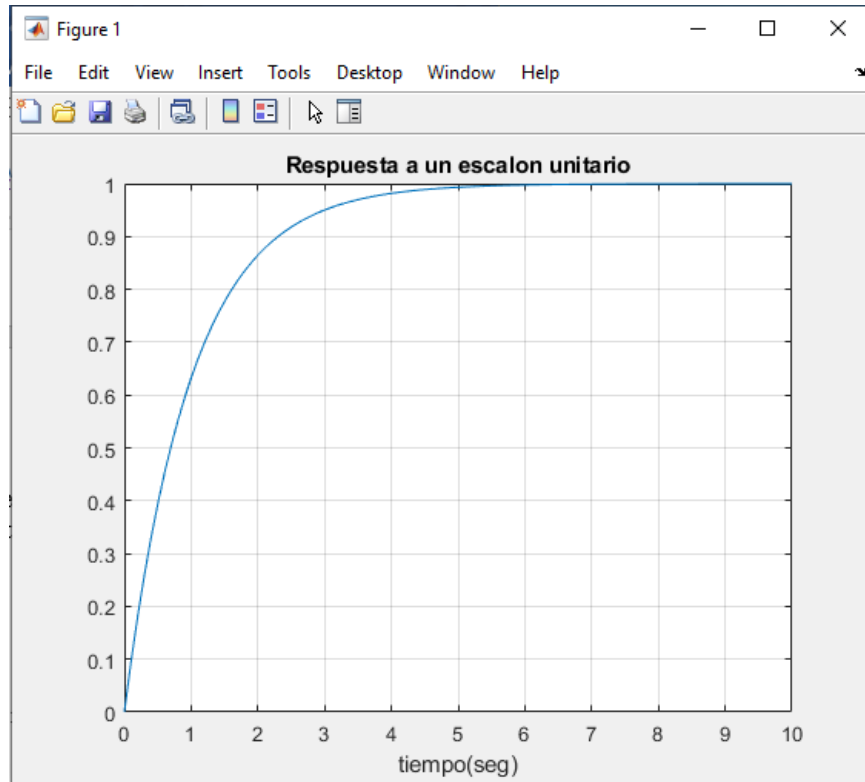
*Figura 9.2 Ejemplo 1: Sistemas de primer orden.*

Command Window	Command Window	Command Window	Command Window
ye =	0.9179	0.9955	0.9998
0	0.9257	0.9959	0.9998
0.0952	0.9328	0.9963	0.9998
0.1813	0.9392	0.9967	0.9998
0.2592	0.9450	0.9970	0.9998
0.3297	0.9502	0.9973	0.9998
0.3935	0.9550	0.9975	0.9998
0.4512	0.9592	0.9978	0.9998
0.5034	0.9631	0.9980	0.9998
0.5507	0.9666	0.9982	0.9998
0.5934	0.9698	0.9983	0.9998
0.6321	0.9727	0.9985	0.9998
0.6671	0.9753	0.9986	0.9998
0.6988	0.9776	0.9988	0.9998
0.7275	0.9798	0.9988	0.9998
0.7534	0.9817	0.9989	0.9998
0.7769	0.9834	0.9990	0.9999
0.7981	0.9850	0.9991	0.9999
0.8173	0.9864	0.9992	0.9999
0.8347	0.9877	0.9993	0.9999
0.8504	0.9889	0.9993	0.9999
0.8647	0.9899	0.9994	0.9999
0.8775	0.9909	0.9994	0.9999
0.8892	0.9918	0.9995	0.9999
0.8997	0.9926	0.9995	0.9999
0.9093	0.9933	0.9996	0.9999
fx	0.9939	0.9996	0.9999
	0.9945	0.9997	1.0000
	fx	fx	
	0.9950	0.9997	

Figura 9.3 Ejemplo 1: Sistemas de primer orden.

```
>> plot(t,ye)
>> title ('Respuesta a un escalon unitario');
>> xlabel ('tiempo(seg) ');
>> grid;
```

Figura 9.4 Ejemplo 1: Sistemas de primer orden.



*Figura 9.5 Ejemplo 1: Gráfica sistemas de primer orden.*

```

>>
>> yRP= ye(length(ye));
>> k=yRP

k =

    1.0000

>> n = 1;
>> while ye(n) < 0.63*yRP
n=n+1;
end
>> tauEstim = 0.1*(n-1)

tauEstim =

    1

>> fprintf("constante de tiempo:%f\n",tauEstim)
constante de tiempo:1.000000

```

*Figura 9.6 Ejemplo 1: Sistemas de primer orden.*

## 9.2 EJEMPLO 2

$$G(s) = \frac{3}{2s+3}$$

```
Command Window
>> syms s t
>> K=3

K =

    3

>> tau=2

tau =

    2

>> num=K

num =

    3

>> den=[tau,3]

den =

    2    3

>> t=[0:0.1:10]

t =

Columns 1 through 5

    0    0.1000    0.2000    0.3000    0.4000

Columns 6 through 10

    0.5000    0.6000    0.7000    0.8000    0.9000
```

Figura 9.7 Ejemplo 2: Sistemas de primer orden.

Columns 11 through 15	1.0000	1.1000	1.2000	1.3000	1.4000
Columns 16 through 20	1.5000	1.6000	1.7000	1.8000	1.9000
Columns 21 through 25	2.0000	2.1000	2.2000	2.3000	2.4000
Columns 26 through 30	2.5000	2.6000	2.7000	2.8000	2.9000
Columns 31 through 35	3.0000	3.1000	3.2000	3.3000	3.4000
Columns 36 through 40	3.5000	3.6000	3.7000	3.8000	3.9000
Columns 41 through 45	4.0000	4.1000	4.2000	4.3000	4.4000
Columns 46 through 50	4.5000	4.6000	4.7000	4.8000	4.9000
Columns 51 through 55	5.0000	5.1000	5.2000	5.3000	5.4000
Columns 56 through 60	5.5000	5.6000	5.7000	5.8000	5.9000
Columns 61 through 65	6.0000	6.1000	6.2000	6.3000	6.4000
Columns 66 through 70					

*Figura 9.8 Ejemplo 2: Sistemas de primer orden.*

```
6.5000    6.6000    6.7000    6.8000    6.9000
Columns 71 through 75
7.0000    7.1000    7.2000    7.3000    7.4000
Columns 76 through 80
7.5000    7.6000    7.7000    7.8000    7.9000
Columns 81 through 85
8.0000    8.1000    8.2000    8.3000    8.4000
Columns 86 through 90
8.5000    8.6000    8.7000    8.8000    8.9000
Columns 91 through 95
9.0000    9.1000    9.2000    9.3000    9.4000
Columns 96 through 100
9.5000    9.6000    9.7000    9.8000    9.9000
Column 101
10.0000
>> ye=step(num,den,t)
```

*Figura 9.9 Ejemplo 2: Sistemas de primer orden.*

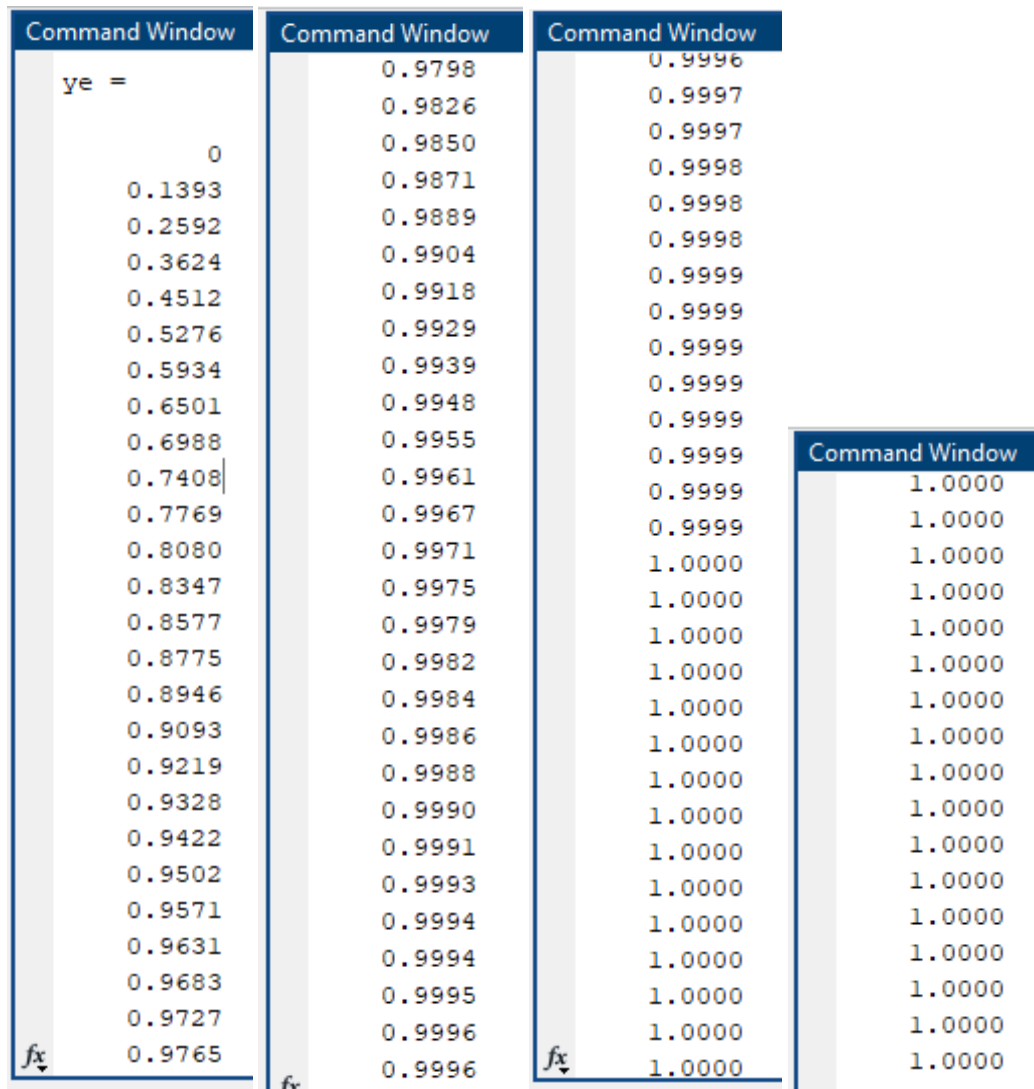


Figura 9.10 Ejemplo 2: Sistemas de primer orden.

```
>> plot(t, ye);
>> title ('Respuesta a un escalon unitario');
>> xlabel ('tiempo(seg) ');
>> grid;
>>
>>
```

Figura 9.11 Ejemplo 2: Sistemas de primer orden.



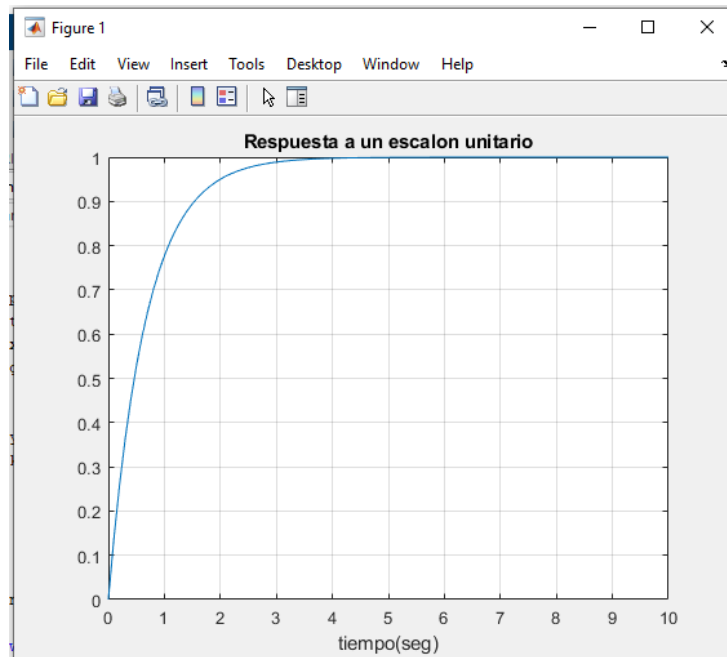


Figura 9.12 Ejemplo 2: Gráfica sistemas de primer orden.

```

Command Window
//
>> yRP= ye(length(ye));
>> k=yRP

k =

    1.0000

>> n=1;
>>
>> while ye(n) < 0.63*yRP
n=n+1
end

n =

    2

n =

    3

n =

    4

```

Figura 9.13 Ejemplo 2: Sistemas de primer orden.

```
n =  
5  
  
n =  
6  
  
n =  
7  
  
n =  
8
```

```
>> tauEstim = 0.1*(n-1)  
  
tauEstim =  
  
0.7000  
  
>> fprintf("constante de tiempo:%f\n",tauEstim)  
constante de tiempo:0.700000  
>>
```

*Figura 9.14 Ejemplo 2: Sistemas de primer orden.*

### 9.3 EJEMPLO 3

$$G(s) = \frac{1}{s+5}$$

```
Command Window
>> syms s t
>> K=1
K =
    1
>> tau=1
tau =
    1
>> num=K
num =
    1
>> den=[tau,5]
den =
    1    5
>> t=[0:0.1:10]
t =
Columns 1 through 5
    0    0.1000    0.2000    0.3000    0.4000
Columns 6 through 10
    0.5000    0.6000    0.7000    0.8000    0.9000
```

Figura 9.15 Ejemplo 3: Sistemas de primer orden.

Columns 11 through 15				
1.0000	1.1000	1.2000	1.3000	1.4000
Columns 16 through 20				
1.5000	1.6000	1.7000	1.8000	1.9000
Columns 21 through 25				
2.0000	2.1000	2.2000	2.3000	2.4000
Columns 26 through 30				
2.5000	2.6000	2.7000	2.8000	2.9000
Columns 31 through 35				
3.0000	3.1000	3.2000	3.3000	3.4000
Columns 36 through 40				
3.5000	3.6000	3.7000	3.8000	3.9000
Columns 41 through 45				
4.0000	4.1000	4.2000	4.3000	4.4000
Columns 46 through 50				
4.5000	4.6000	4.7000	4.8000	4.9000
Columns 51 through 55				
5.0000	5.1000	5.2000	5.3000	5.4000
Columns 56 through 60				
5.5000	5.6000	5.7000	5.8000	5.9000
Columns 61 through 65				
6.0000	6.1000	6.2000	6.3000	6.4000

*Figura 9.16 Ejemplo 3: Sistemas de primer orden.*

```

Columns 66 through 70
    6.5000    6.6000    6.7000    6.8000    6.9000

Columns 71 through 75
    7.0000    7.1000    7.2000    7.3000    7.4000

Columns 76 through 80
    7.5000    7.6000    7.7000    7.8000    7.9000

Columns 81 through 85
    8.0000    8.1000    8.2000    8.3000    8.4000

Columns 86 through 90
    8.5000    8.6000    8.7000    8.8000    8.9000

Columns 91 through 95
    9.0000    9.1000    9.2000    9.3000    9.4000

Columns 96 through 100
    9.5000    9.6000    9.7000    9.8000    9.9000
fv
Column 101
    10.0000

>> ye=step(num,den,t)

```

*Figura 9.17 Ejemplo 3: Sistemas de primer orden.*

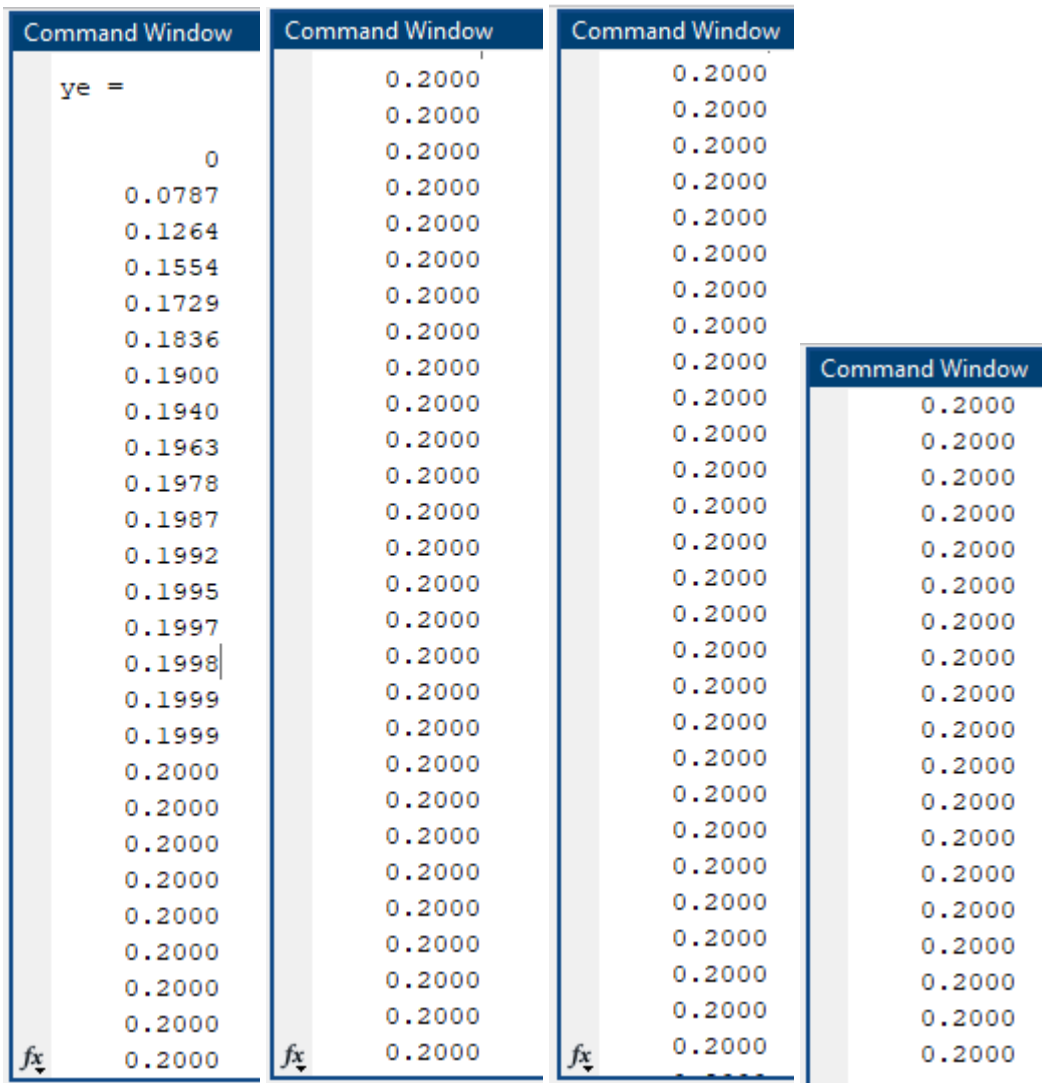


Figura 9.18 Ejemplo 3: Sistemas de primer orden

```
>> plot(t, ye);
>> title('Respuesta a un escalon unitario');
>> xlabel('tiempo(seg)');
>> grid;
>>
```

Figura 9.19 Ejemplo 3: Sistemas de primer orden.

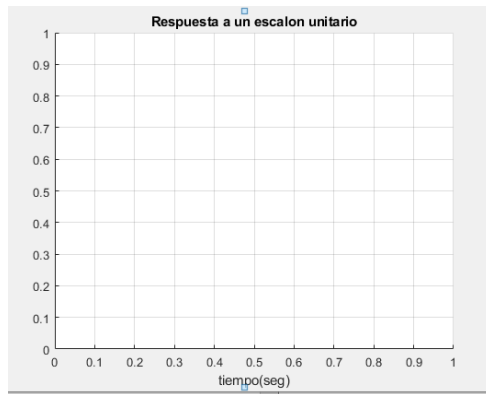


Figura 9.20 Ejemplo 3: Gráfica de sistemas de primer orden.

```

Command Window
>> |
>> yRP= ye (length(ye));
k=yRP
n=1

k =

    0.2000

n =

    1

>> while ye(n) < 0.63*yRP
n=n+1
end

n =

    2

n =

    3

fx >> tauEstim = 0.1*(n-1)

```

Figura 9.21 Ejemplo 3: Sistemas de primer orden.

```

tauEstim =

    0.2000

>> fprintf("constante de tiempo:%f\n",tauEstim)
constante de tiempo:0.200000

```

Figura 9.22 Ejemplo 3: Sistemas de primer orden.

## 10. SISTEMAS DE SEGUNDO ORDEN EN MATLAB

La representación en forma de función de transferencia viene dada de manera general como:

$$G(s) = \frac{k\omega_n^2}{s^2 + 2\zeta\omega_n s + \omega_n^2}$$

O también:

$$G_2(s) = \frac{0.2s^2 + 0.3s + 1}{(s + 0.5)(s^2 + 0.4s + 1)}$$

1. Se debe de declarar las variables que se estará utilizando, en este caso “s, t”, con el siguiente comando:

***syms s t***

2. Posteriormente se agregan los valores del numerador y del denominador, con los siguientes comandos.

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

3. Para visualizar los ceros, polos y la ganancia se ingresa el siguiente comando:

***[ceros,polos,gan] = tf2zp (num,den)***

4. Posteriormente para encontrar la respuesta a la entrada escalón unitario se obtiene con la función step. Para ello se debe de definir un intervalo de simulación, empleando los siguientes comandos.



```

t2= [0:0.3:15];
y2e = step(num,den,t2);
plot(t2,y2e);
title ('Respuesta a un escalón unitario');
xlabel ('tiempo(seg)');
grid

```

5. Para obtener la respuesta en frecuencia de los sistemas se puede obtener usando las funciones de Bode. Para obtener la gráfica de la respuesta en frecuencia escriba 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');

```

6. El comando Nyquist calcula tanto la parte real como la imaginaria de  $G(j\omega)$  y realiza la representación si no se le indican parámetros de salida. Para obtener la representación gráfica solo hay que graficar la parte real contra la imaginaria. El resultado obtenido mediante el ejemplo anterior puede verse al escribir los siguientes comandos:

```

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

```

7. Y finalmente para los márgenes de estabilidad son el margen de fase y el margen de ganancia se calculan usando el comando margin. Escriba el siguiente comando para obtener el margen de ganancia y el margen de fase de la función de transferencia de segundo orden además de sus frecuencias correspondientes:

```

[mg,mf,wmg,wmf] = margin (num,den)

```

## 10.1 EJEMPLO 1

$$G(S) = \frac{.3s^2+0.2s+1}{(s+0.5)(s^2+0.5s+1)}$$

### Command Window

```
>> syms s t
>> num = [.3 .2 1];
>> den1=[1 .5 1];
>> den2 = [1 .5];
>> den = conv(den1,den2);
>>
>>
>> [ceros,polos,gan] = tf2zp (num,den)
```

Figura 10 Ejemplo 1: Sistemas de segundo orden.

### *Raíces del sistema (polos y ceros)*

```
ceros =

-0.3333 + 1.7951i
-0.3333 - 1.7951i

polos =

-0.2500 + 0.9682i
-0.2500 - 0.9682i
-0.5000 + 0.0000i
```

Figura 10.1 Ejemplo 1 Sistemas de segundo orden: polos y ceros.

## Ganancia estática del sistema

```
gan =  
fx 0.3000
```

Figura 10.2 Ejemplo 1 Sistemas de segundo orden: ganancia del sistema.

## Respuesta temporal a entrada escalón unitario

```
>>  
>> t2= [0:0.3:15];  
>> y2e = step(num,den,t2);  
plot(t2,y2e);  
title ('Respuesta a un escalón unitario');  
xlabel ('tiempo(seg)');  
>> grid;
```

Figura 10.3 Ejemplo 1 Sistemas de segundo orden: respuesta temporal.

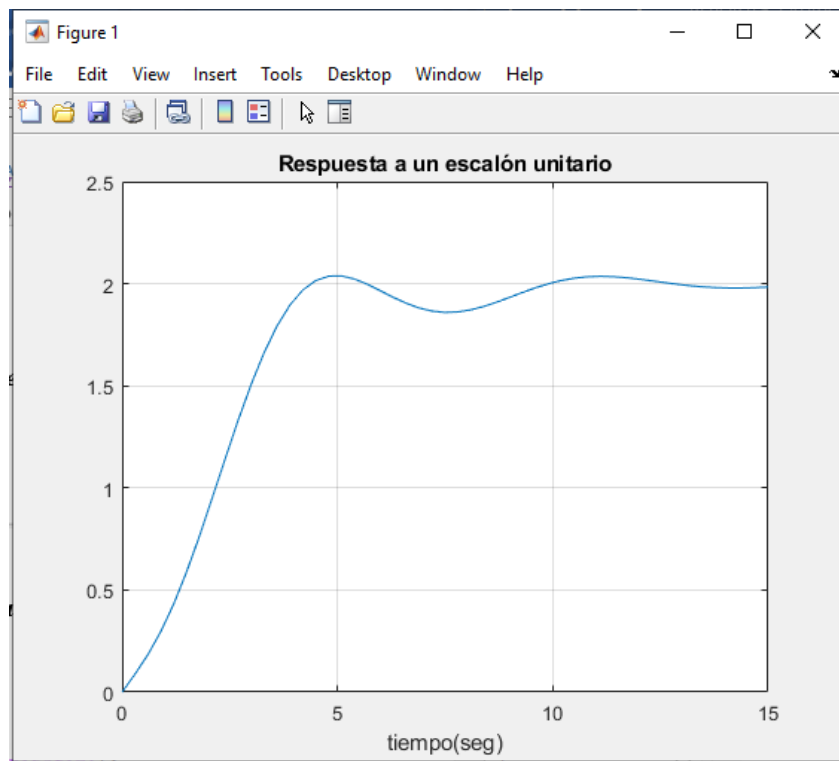


Figura 10.4 Ejemplo 1 Sistemas de segundo orden: grafica respuesta a un escalón unitario.

## Respuesta frecuencial en magnitud y fase

```
|  
>> [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');  
fx >> |
```

Figura 10.5 Ejemplo 1 Sistemas de segundo orden: respuesta frecuencial.

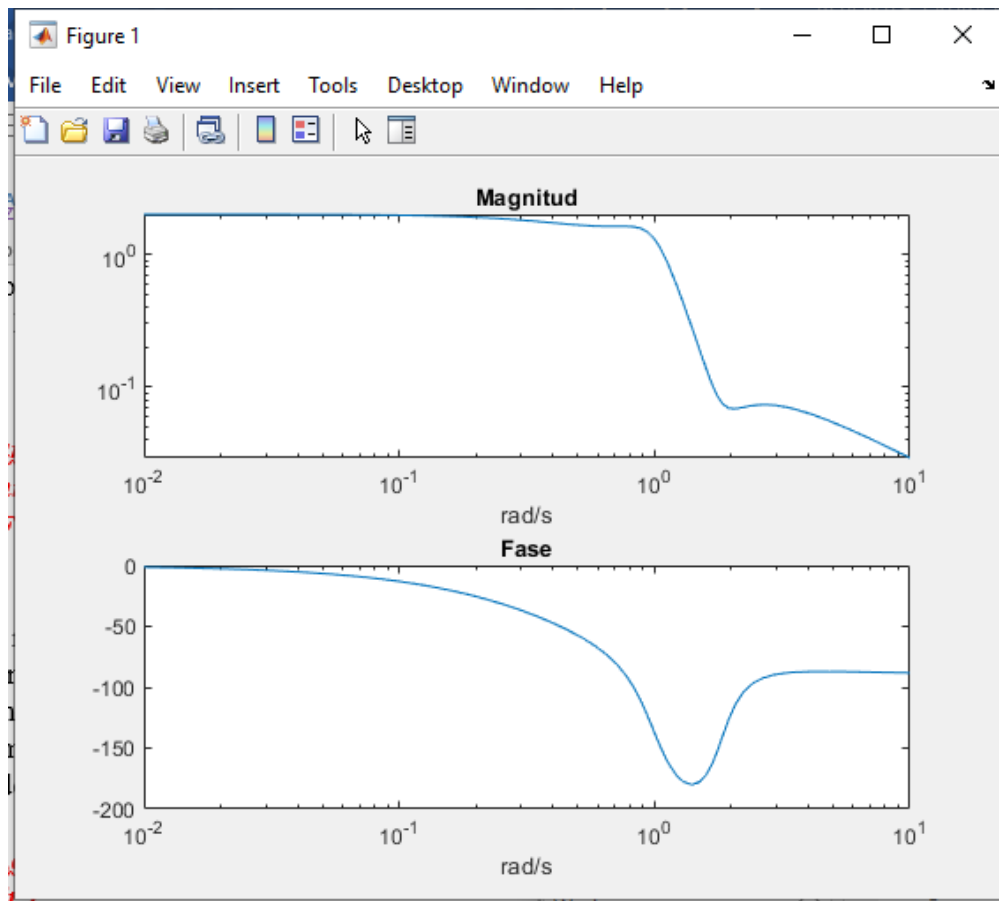


Figura 10.6 Ejemplo 1 Sistemas de segundo orden: grafica respuesta frecuencial.

## Las partes real e imaginaria de $G(jw)$

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

Figura 10.7 Ejemplo 1 Sistemas de segundo orden: partes real e imaginaria.

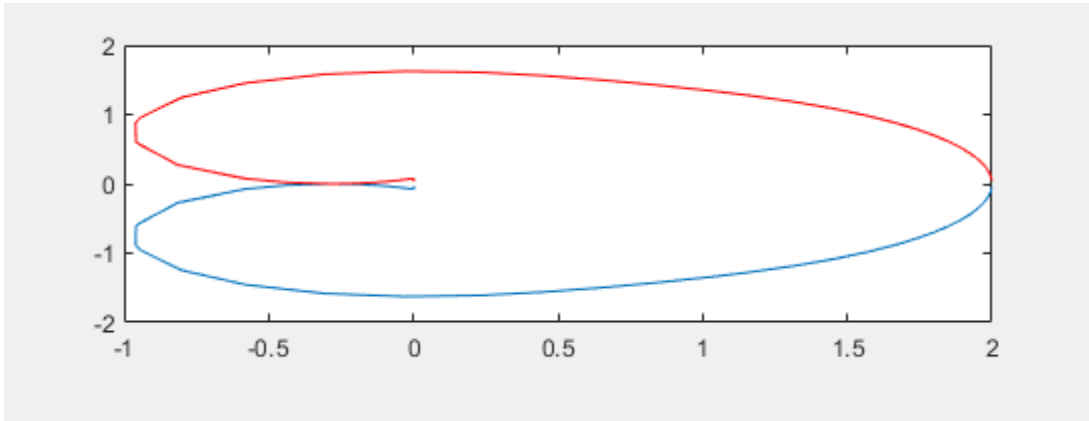


Figura 10.8 Ejemplo 1 Sistemas de segundo orden: grafica parte real e imaginaria.

### Los márgenes de ganancia

```

%%
>> [mg,mf,wmg,wmf] = margin (num,den)

mg =

    3.3334

mf =

   25.1677

wmg =

    1.3844

wmf =

    1.0852

```

Figura 10.9 Ejemplo 1 Sistemas de segundo orden : márgenes de ganancia.

## EJEMPLO 2

$$G(S) = \frac{s^2 + 0.2s + 1}{(2s + 0.2)(s^2 + 0.1s + 1)}$$

```
Command Window
>> syms s t
>> num=[1 .2 1]

num =

    1.0000    0.2000    1.0000

>> den1=[1 .1 1]

den1 =

    1.0000    0.1000    1.0000

>> den2=[2 .2]

den2 =

    2.0000    0.2000

>> den=conv(den1,den2);
>>
>> [ceros,polos,gan]=tf2zp(num,den)
```

Figura 10.10 Ejemplo 2: Sistemas de segundo orden.

### *Raíces del sistema (polos y ceros)*

```
ceros =

    -0.1000 + 0.9950i
    -0.1000 - 0.9950i

polos =

    -0.0500 + 0.9987i
    -0.0500 - 0.9987i
    -0.1000 + 0.0000i
```

Figura 10.11 Ejemplo 2: Sistemas de segundo orden: polos y ceros.

## Ganancia estática del sistema

```
gan =  
0.5000
```

Figura 10.12 Ejemplo 2: Sistemas de segundo orden.

## Respuesta temporal a entrada escalón unitario

```
>> t2= [0:0.3:15];  
>> y2e = step(num,den,t2);  
>> plot(t2,y2e);  
>> title ('Respuesta a un escalón unitario');  
>> xlabel ('tiempo(seg)');  
>> grid
```

Figura 10.13 Ejemplo 2: Sistemas de segundo orden: respuesta temporal

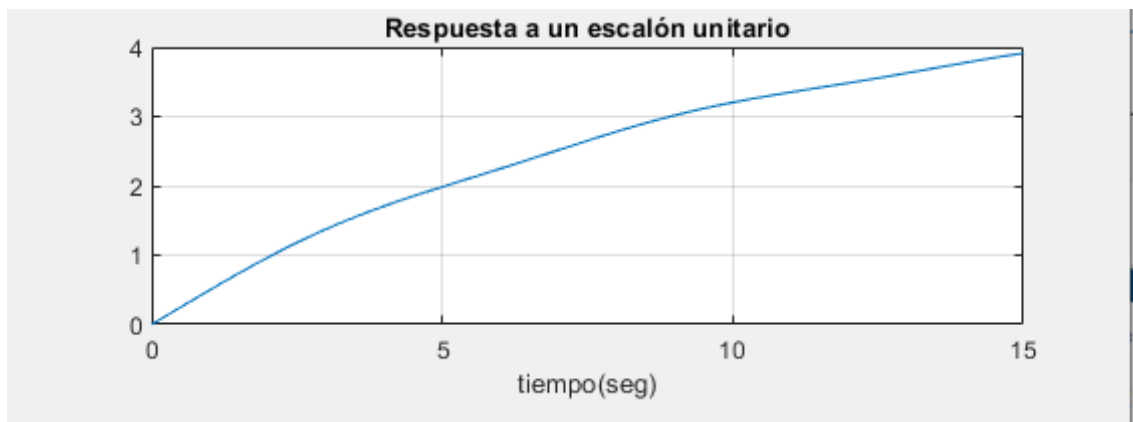


Figura 10.14 Ejemplo 2: Sistemas de segundo orden: Gráfica de respuesta a un escalon unitario

## Respuesta frecuencial en magnitud y fase

```
%%  
[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 10.15 Ejemplo 2: Sistemas de segundo orden: Respuesta frecuencial

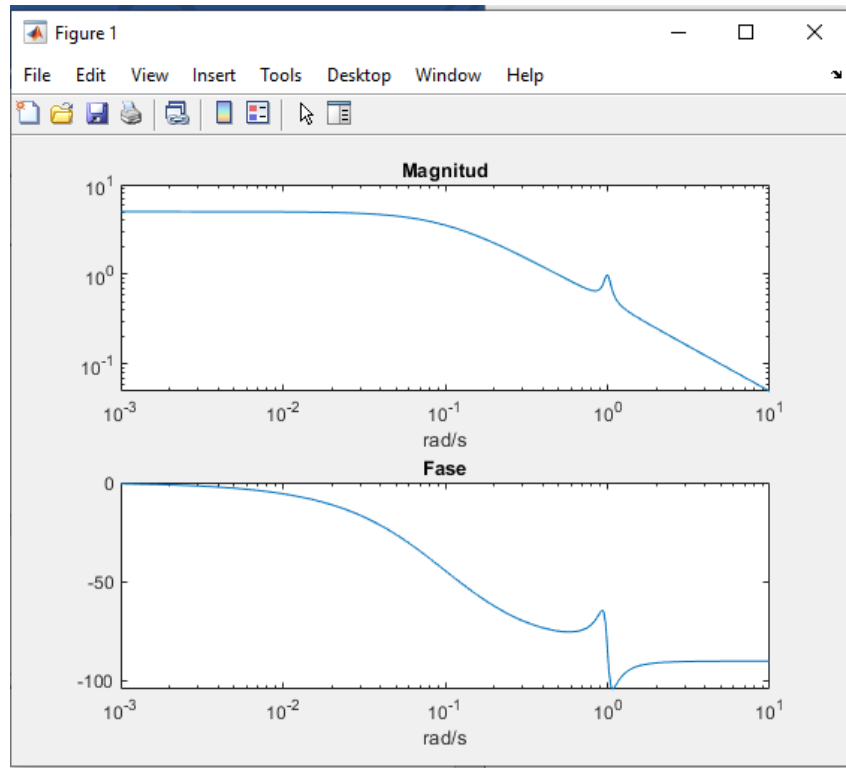


Figura 10.16 Ejemplo 2: Sistemas de segundo orden: Grafica respuesta frecuencial

### Las partes real e imaginaria de $G(j\omega)$

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

Figura 10.17 Ejemplo 2: Sistemas de segundo orden: Parte real e imaginaria.

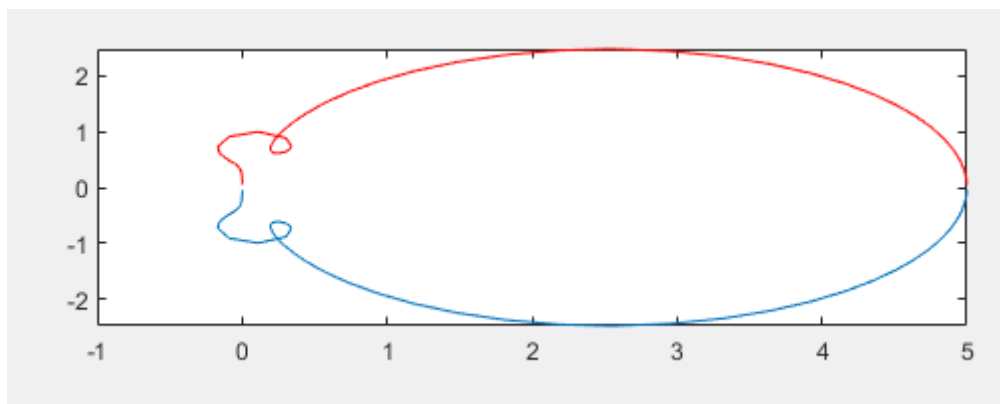


Figura 10.18 Ejemplo 2: Sistemas de segundo orden: Gráfica parte real e imaginaria.



## *Los márgenes de ganancia*

```
>> [mg,mf,wmg,wmf] = margin (num,den)

mg =

    Inf

mf =

    105.1601

wmg =

    NaN

|
wmf =

    0.4931
```

*Figura 10.19 Ejemplo 2: Sistemas de segundo orden: Márgenes de ganancia.*

### 10.3 EJEMPLO 3

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

```
Command Window
>> syms s t
num=[1 2 1]

num =

     1     2     1

>> den1=[1 1 2]

den1 =

     1     1     2

>> den2=[1 3]

den2 =

     1     3

>> den=conv(den1,den2)

den =

     1     4     5     6

fx >> [ceros, polos, gan]=tf2zp(num, den)
```

Figura 10.20 Ejemplo 3: Sistemas de segundo orden.

#### *Raíces del sistema (polos y ceros)*

```
ceros =

    -1
    -1

polos =

-3.0000 + 0.0000i
-0.5000 + 1.3229i
-0.5000 - 1.3229i
```

Figura 10.21 Ejemplo 3 Sistemas de segundo orden: Polos y ceros.

### *Ganancia estática del sistema*

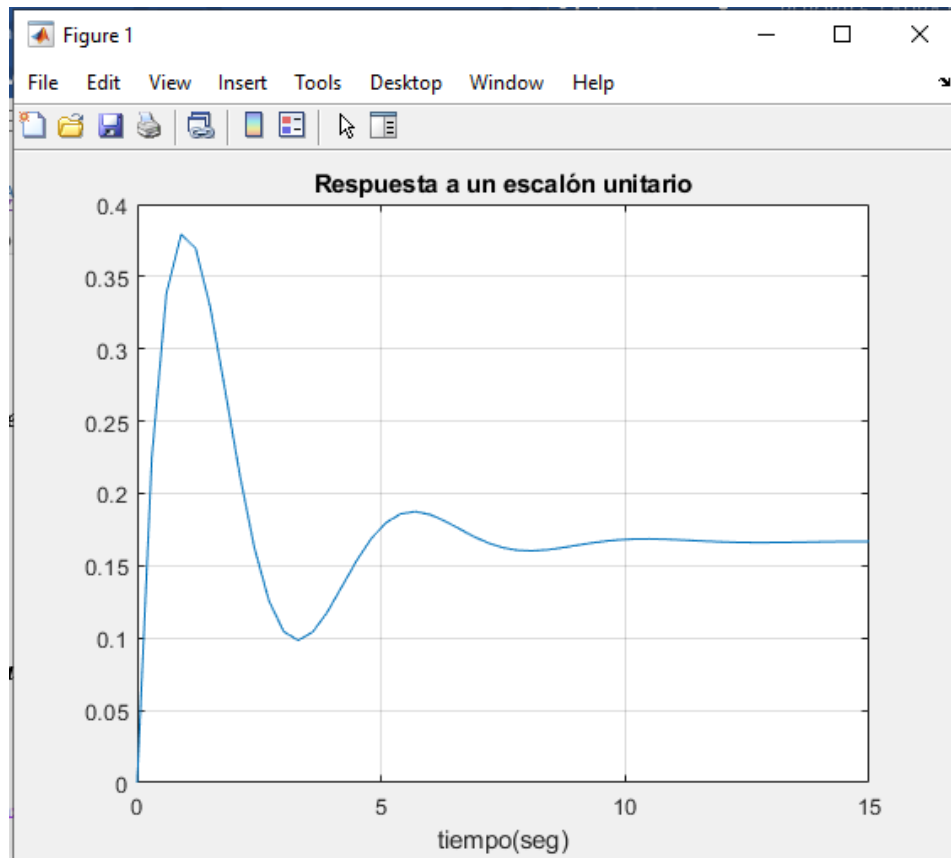
```
gan =  
1
```

*Figura 10.22 Ejemplo 3 Sistemas de segundo orden: Ganancia del sistema.*

### *Respuesta temporal a entrada escalón unitario*

```
>> t2= [0:0.3:15];  
>> y2e = step(num,den,t2);  
>> plot(t2,y2e);  
>> title ('Respuesta a un escalón unitario');  
>> xlabel ('tiempo(seg)');  
>> grid;
```

*Figura 10.23 Ejemplo 3 Sistemas de segundo orden: Respuesta a un escalón unitario.*



*Figura 10.24 Ejemplo 3 Sistemas de segundo orden: Gráfica de la respuesta a un escalón unitario.*

## Respuesta frecuencial en magnitud y fase

```
>>  
>> [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');  
>>  
fx >> |
```

Figura 10.25 Ejemplo 3 Sistemas de segundo orden: Respuesta frecuencial.

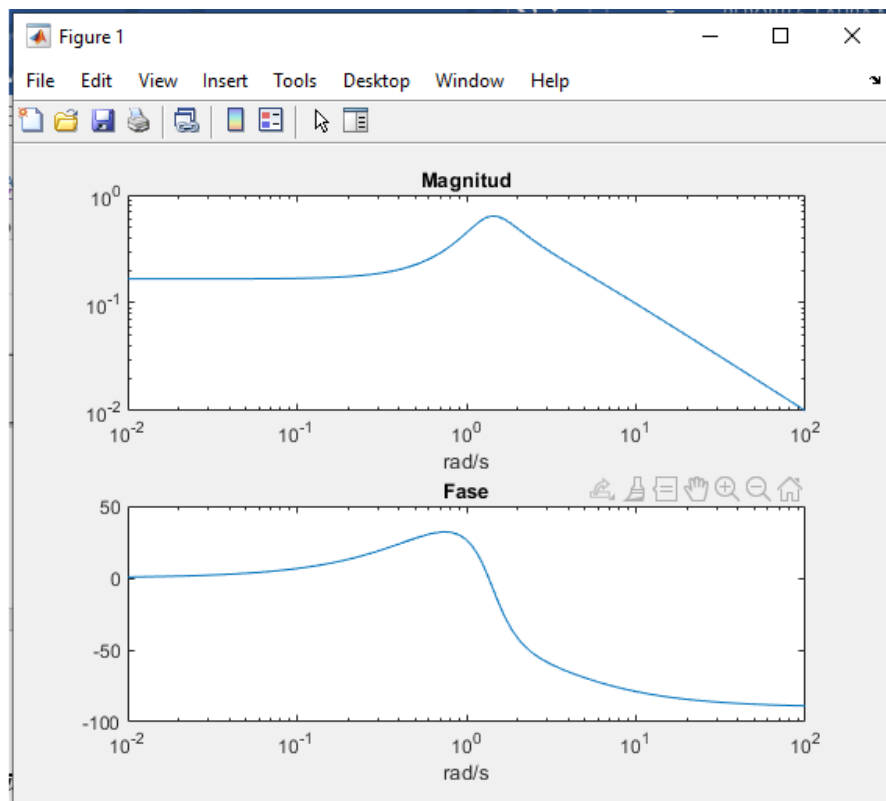


Figura 10.26 Ejemplo 3 Sistemas de segundo orden: Respuesta frecuencial.

## Las partes real e imaginaria de $G(j\omega)$

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

Figura 10.27 Ejemplo 3 Sistemas de segundo orden: Parte real e imaginaria.

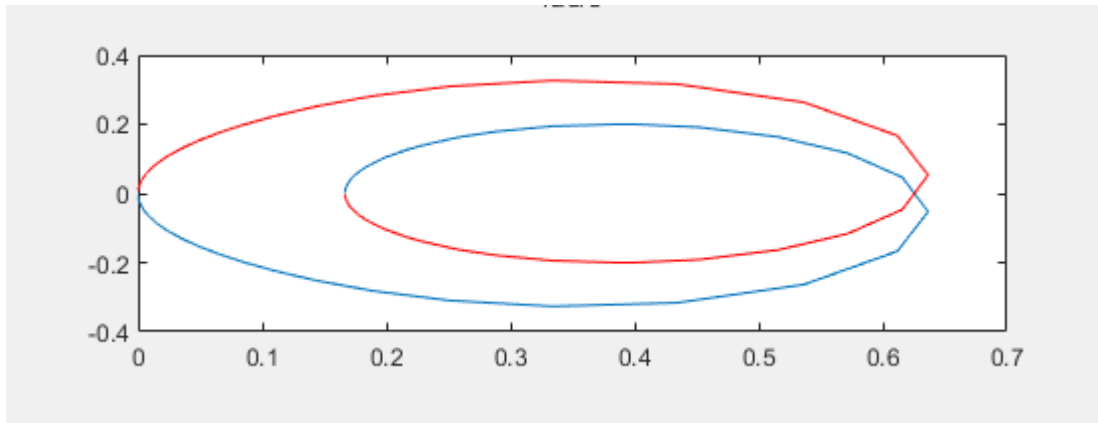


Figura 10.28 Ejemplo 3 Sistemas de segundo orden: Gráfica parte real e imaginaria.

### Los márgenes de ganancia

```
>> [mg,mf,wmg,wmf] = margin (num,den)

mg =

    Inf

mf =

    Inf
|

wmg =

    NaN

wmf =

    NaN

>> |
```

Figura 10.29 Ejemplo 3 Sistemas de segundo orden: Márgenes de ganancia.

## 11.CONTROLADORES

Los controladores nos ayudan a corregir los errores que tienen nuestros sistemas de control.

### 11.1 CONTROLADOR PROPORCIONAL

1. Se debe ingresar la función de transferencia a la cual se le va a agregar el controlador proporcional, con el siguiente comando:

$$\begin{aligned}n &= [3] \\d &= [1,9,4] \\f &= \text{tf}(n,d)\end{aligned}$$

```
Command Window
>>
n = [3]
d = [1,9,4]
f = tf(n,d)

n =
     3

d =
     1     9     4

f =

      3
-----
s^2 + 9 s + 4

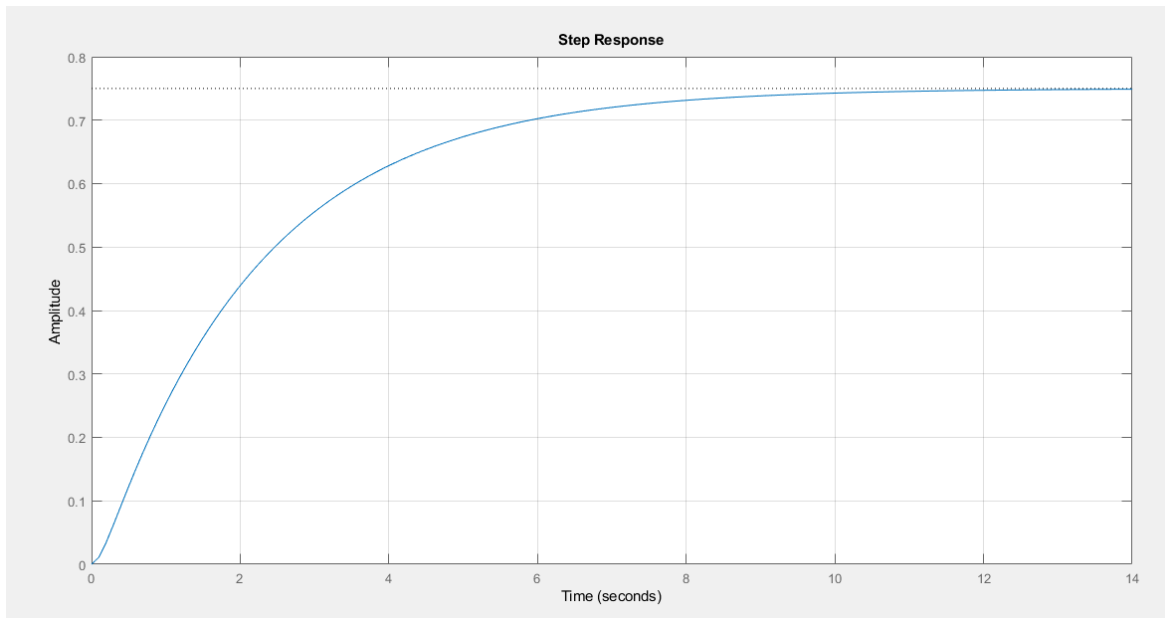
Continuous-time transfer function.
```

Figura 11. Función de transferencia.

2. Para poder visualizar la respuesta del sistema antes de ingresar el controlador proporcional, se ingresa el siguiente comando:

$$\text{step}(f)$$

Con el cual podemos visualizar el tiempo de respuesta.



*Figura 11.1 Gráfica función de transferencia.*

3. Se va a declarar una variable  $p$ , que va a representar la ganancia del controlador proporcional.

$$Kp = 4$$

4. Se ingresará una función para controladores, en el cual solo se ingresará el valor de  $p$ , por el tipo de controlador que estamos manejando:

$$c = pid(Kp,0,0)$$

5. Para realizar el enlace de control se ingresa el siguiente comando:

$$sys=feedback(c*f,1)$$

Donde:

$c$  = controlador

$f$  = función de transferencia

$1$  = retroalimentación

```
>> c = pid(kp,0,0)
sys=feedback(c*f,1)

c =

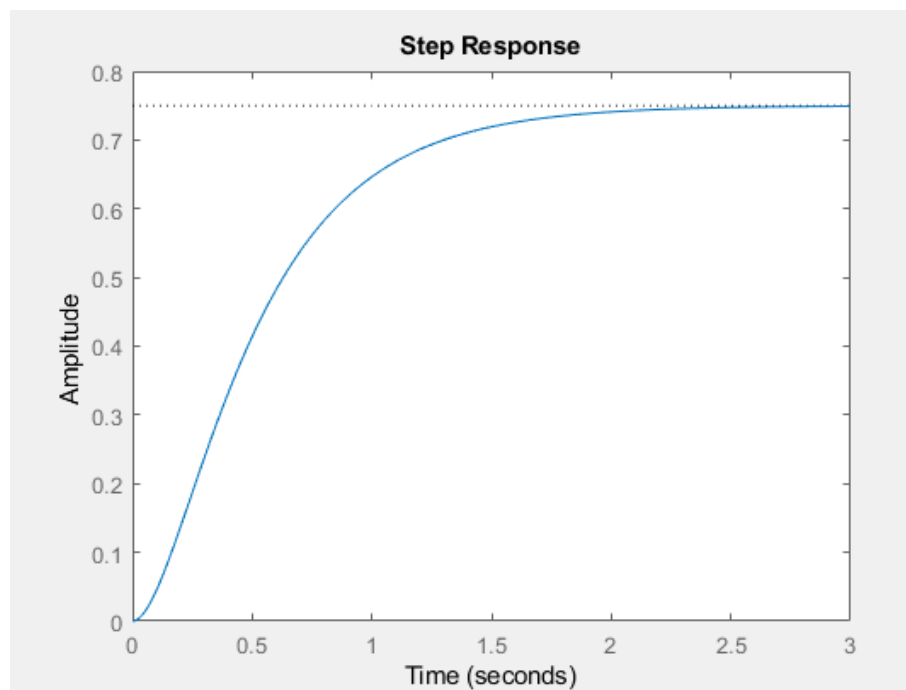
    Kp = 4

P-only controller.
```

*Figura 11.2 Enlace de control.*

6. Para poder visualizar la respuesta del sistema antes de ingresar el controlador proporcional, se ingresa el siguiente comando:

*step(sys)*



*Figura 11.3 Función de transferencia con controlador P*

Podemos observar que el tiempo de respuesta del sistema con un controlador proporcional va a disminuir notablemente.



## 11.1.1 EJEMPLO 1

```
Command Window
>> n = [1,5,0]
d = [1,1,5]
f = tf (n,d)

n =
    1    5    0

d =
    1    1    5

f =
      s^2 + 5 s
      -----
      s^2 + s + 5

Continuous-time transfer function.

>> step(f)
```

Figura 11.4 Ejemplo 1: Controlador P.

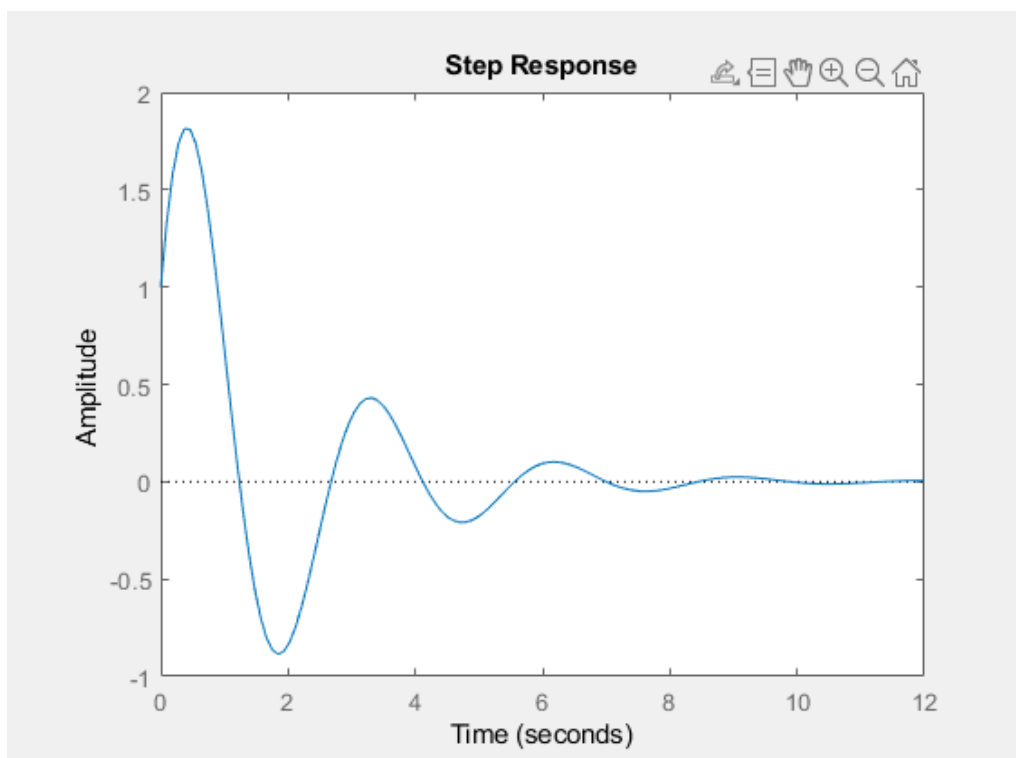


Figura 11.5 Ejemplo 1: Gráfica sin controlador P.

```
>> Kp=10

Kp =

    10

>> c = pid(Kp,0,0)

c =

    Kp = 10

P-only controller.

>> sys=feedback(c*f,1)

sys =

    10 s^2 + 50 s
-----
   11 s^2 + 51 s + 5

Continuous-time transfer function.
```

Figura 11.6 Ejemplo 1: Controlador P.

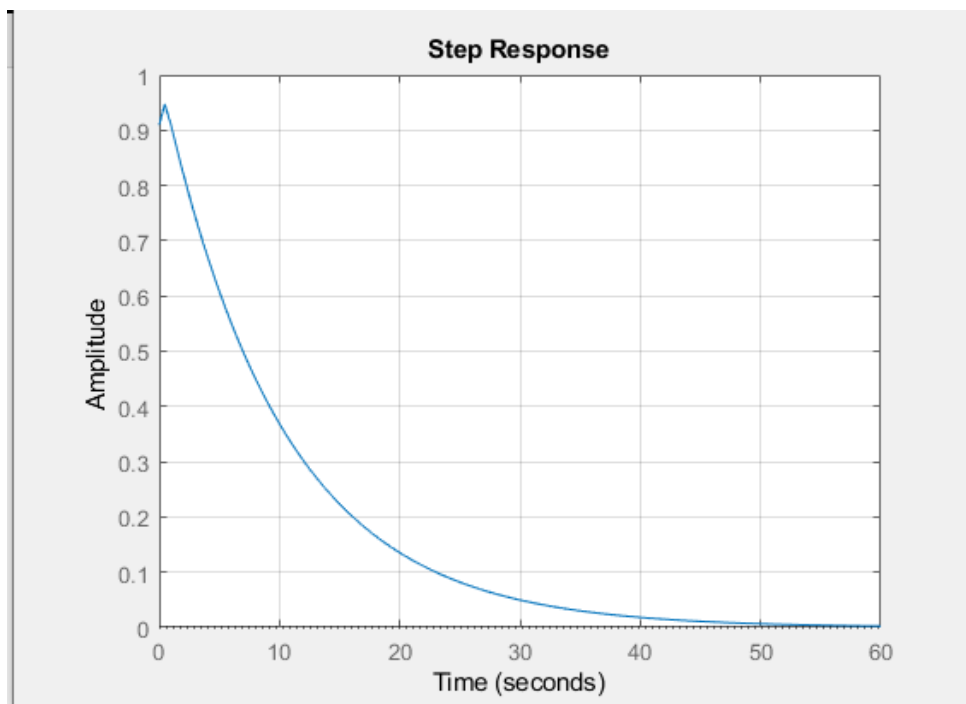


Figura 11.7 Ejemplo 1: Gráfica con controlador P.

## 11.2 CONTROLADOR PROPORCIONAL INTEGRAL

1. Primero se debe de ingresar la función de transferencia del sistema para ver el comportamiento.

$$\begin{aligned}n &= [1] \\d &= [1,9,27,10] \\G &= \text{tf}(n,d) \\&\text{step}(G)\end{aligned}$$

```
>> n=[1]
d=[1,9,27,10]
G=tf(n,d)

n =
    1

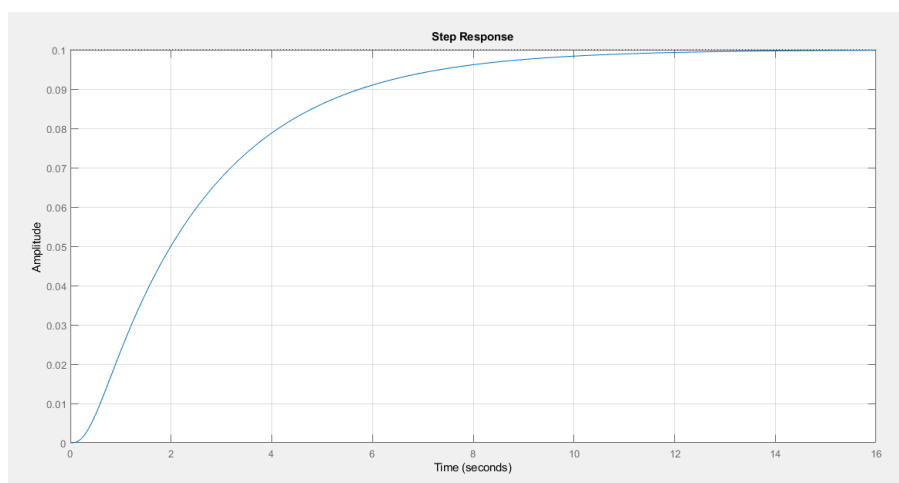
d =
    1     9    27    10

G =
      1
-----
s^3 + 9 s^2 + 27 s + 10

Continuous-time transfer function.

step(G)
```

*Figura 11.8 Función de transferencia.*



*Figura 11.9 Gráfica función de transferencia.*

2. Se debe de simular el diagrama de bloques del sistema de control junto con el controlador en simulink.

Ingresando una entrada de escalón, el controlador PI, la función de transferencia del sistema, un osciloscopio y un sumador.

Vamos a editar la función de transferencia del sistema y la función de transferencia del controlador.

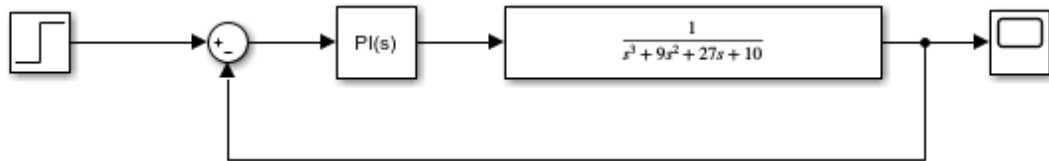


Figura 11.10. Diagrama de bloques.

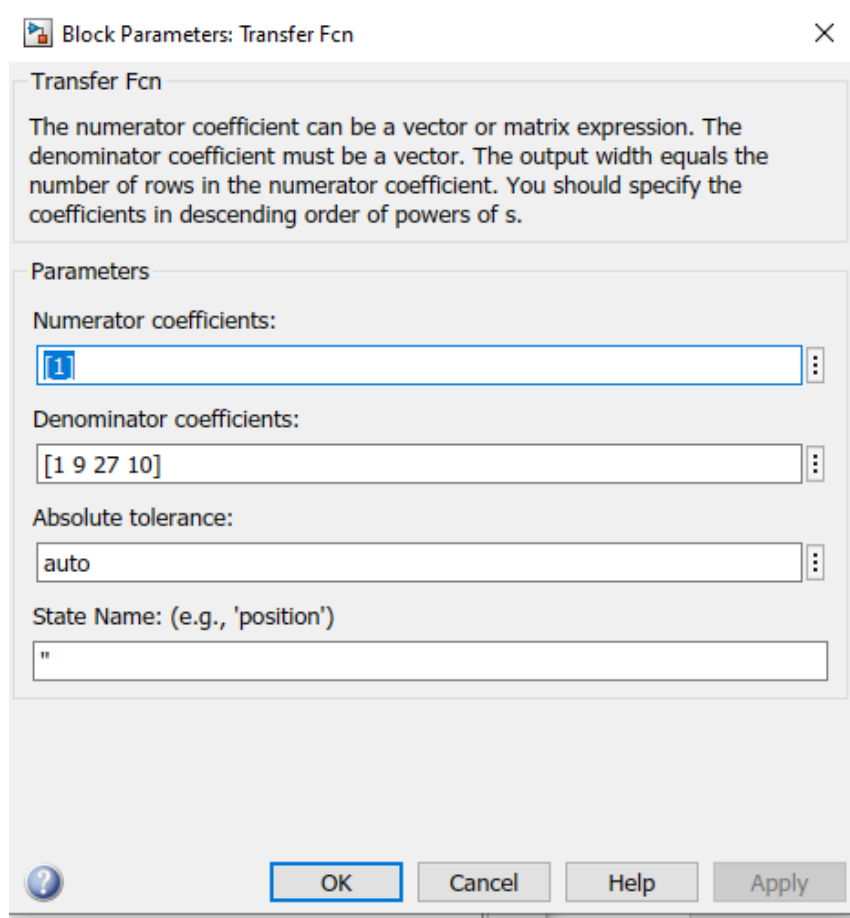
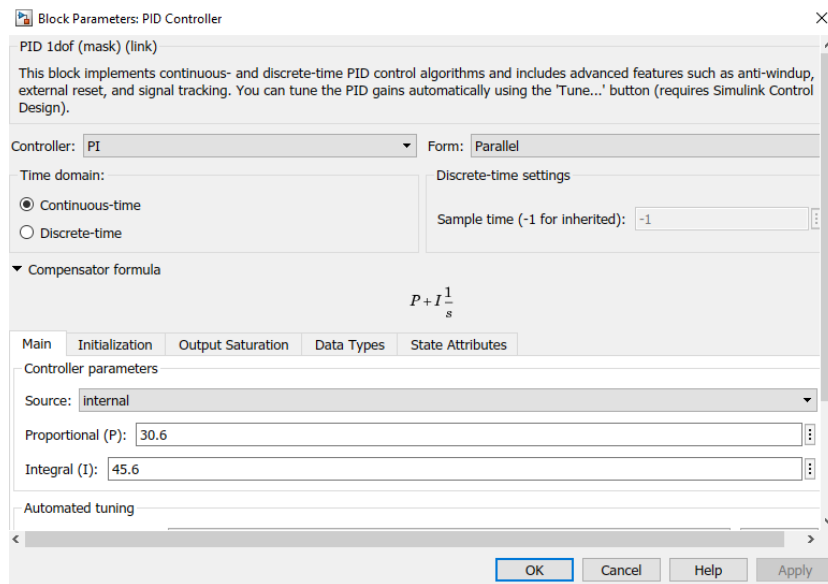
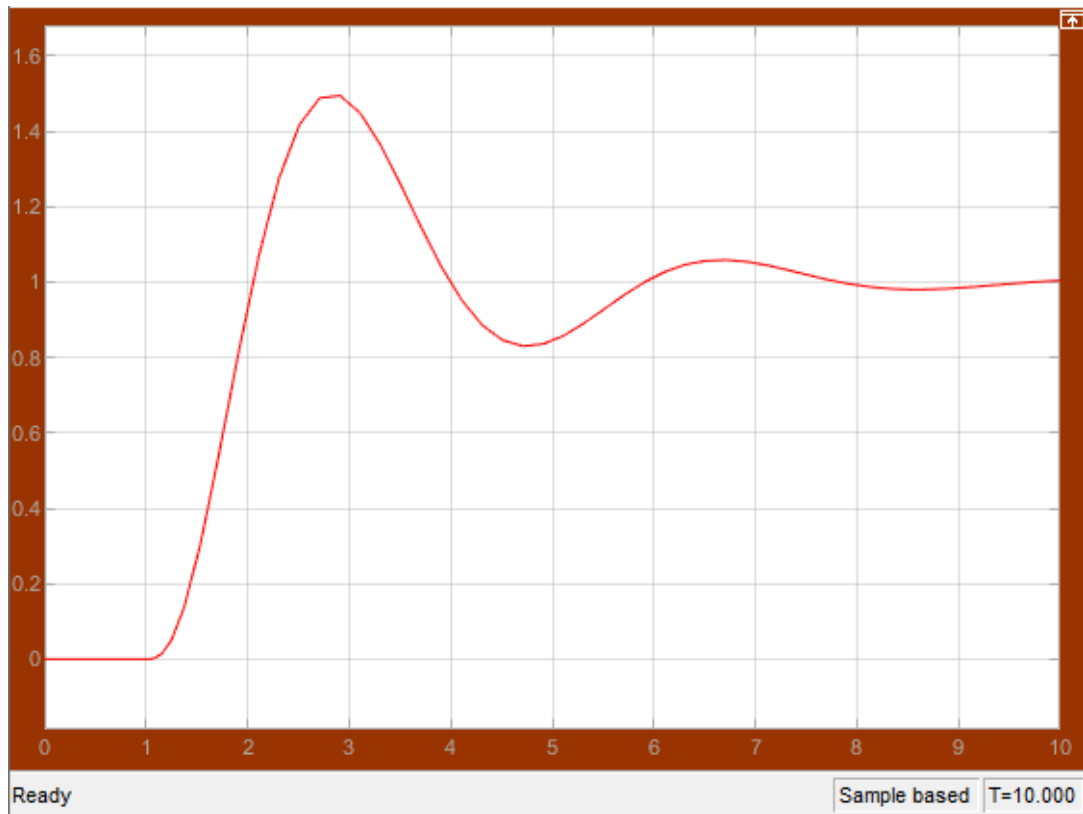


Figura 11.11 Editor del diagrama de bloques.



*Figura 11.12 Editor del diagrama del controlador.*

3. Al tener el diagrama de bloques armado en el osciloscopio vamos a generar la respuesta de salida del sistema con el controlador PI.



*Figura 11.12 Respuesta de salida con el controlador PI*

## 11.2.1 EJEMPLO 2

### Función de transferencia del sistema

$$G(S) = \frac{4}{(s^2+7s+5)}$$

### Función de transferencia del controlador

$$H(S) = 50.5. + \frac{430}{s}$$

```
>> n=[4]
d=[1 7 5]

n =
     4

d =
     1     7     5

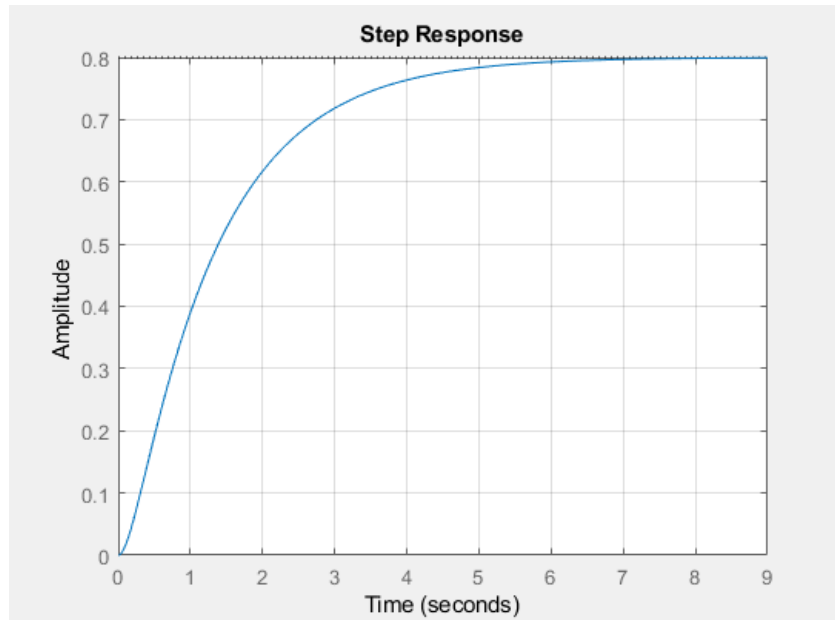
G=tf(n,d)

G =
     4
-----
s^2 + 7 s + 5

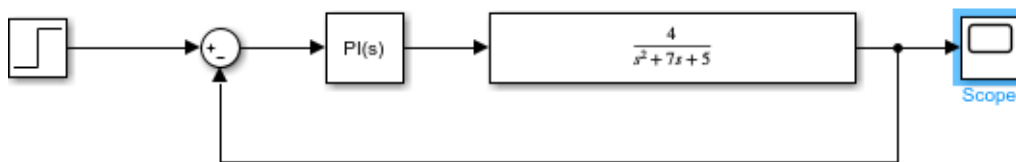
Continuous-time transfer function.

step(G)
```

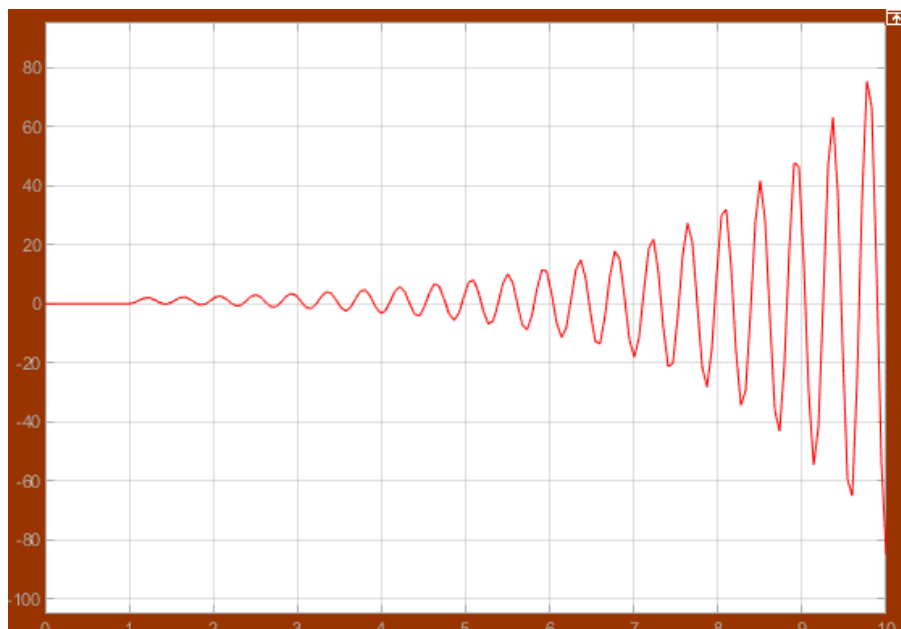
Figura 11.13 Función de transferencia.



*Figura 11.14 Gráfica de función de transferencia.*



*Figura 11.15 Diagrama de bloques con controlador PI.*



*Figura 11.16 Gráfica de respuesta de salida de con controlador PI.*

### 11.3 CONTROLADOR PROPORCIONAL INTEGRAL DERIVATIVO

1. Se debe de ingresar la función de transferencia del sistema de control a la cual se le va agregar el controlador proporcional integral, con el siguiente comando:

$$\begin{aligned}n &= [1] \\d &= [6,11,6,1] \\G &= tf(n,d)\end{aligned}$$

```
Command Window
>> n=[1]
n =
    1
>> d=[6,11,6,1]
d =
     6     11     6     1
>> G=tf(n,d)
G =
          1
-----
 6 s^3 + 11 s^2 + 6 s + 1
Continuous-time transfer function.
>> step(G)
```

Figura 11.17 Función de transferencia.

2. Para poder visualizar la respuesta del sistema antes de ingresar el controlador proporcional, se ingresa el siguiente comando:

$$\text{step}(f)$$



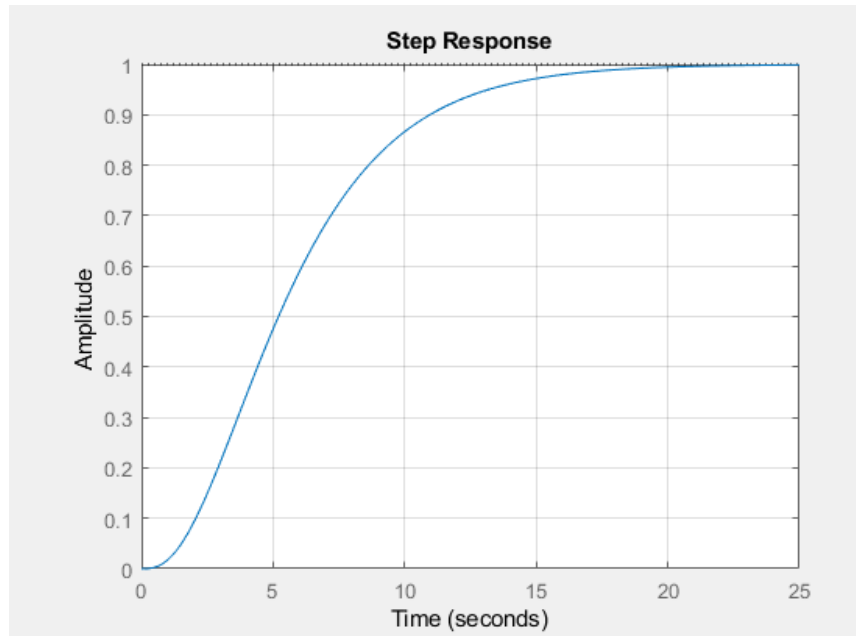


Figura 11.18 Gráfica de la función de transferencia.

Con el cual podemos visualizar el tiempo de respuesta.

3. Se va a declarar una variable p, i, d que va a representar la ganancia del controlador proporcional, integral y derivativo, respectivamente:

$$\begin{aligned}
 K_p &= 10 \\
 K_i &= K_p / 0.25 \\
 K_d &= 0.63
 \end{aligned}$$

```

>> Kp=10

Kp =

    10

>> Ki= Kp/0.25

Ki =

    40

>> Kd=0.63

Kd =

    0.6300

```

Figura 11.19 Variables del controlador.

4. Ya teniendo los valores de las constantes, ingresamos el siguiente comando para obtener la función de transferencia del controlador PID:

$$G_c = K_p + tf([K_p], [K_i \ 0]) + tf([K_p * K_d \ 0], 1)$$

```
>> Gc=Kp+tf([Kp],[Ki 0])+tf([Kp*Kd 0],1)

Gc =

      252 s^2 + 400 s + 10
      -----
              40 s

Continuous-time transfer function.
```

Figura 11.20 Función de transferencia con el controlador PID.

5. Al agregarle el controlador al sistema, vamos a obtener una nueva función de transferencia (a lazo cerrado), esto se obtiene con el comando:

$$G_{LC} = G * G_c / (1 + G * G_c)$$

```
>> G_LC=G*Gc/(1+G*Gc)

G_LC =

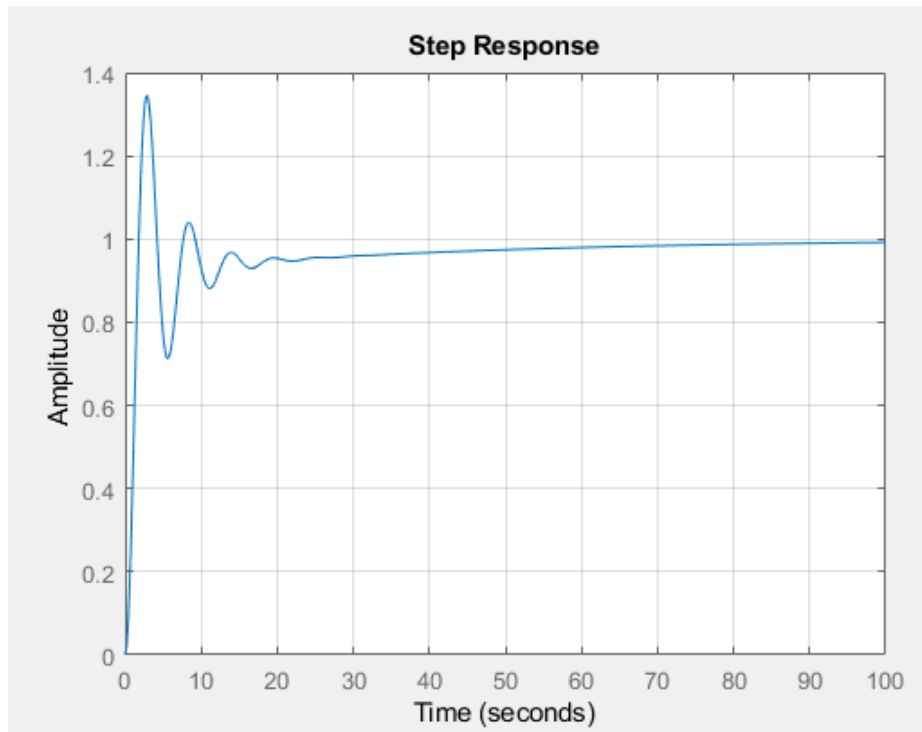
      60480 s^6 + 206880 s^5 + 238880 s^4 + 110480 s^3 + 18400 s^2
                                                    + 400 s
      -----
      57600 s^8 + 211200 s^7 + 369280 s^6 + 437280 s^5 + 331680 s^4
                                                    + 129680 s^3 + 20000 s^2 + 400 s

Continuous-time transfer function.
```

Figura 11.21 Función de transferencia a lazo cerrado.

6. Para poder visualizar la respuesta del sistema después de ingresar el controlador proporcional, se ingresa el siguiente comando:

*step(G\_LC)*



*Figura 11.21 Gráfica de la función de transferencia con el controlador PID.*

Podemos observar el tiempo de respuesta del sistema con un controlador proporcional integral derivativo.

### 11.3. 1 EJEMPLO 3

```
Command Window
>> n=[1 7]
n =
    1    7
>> d=[1 6 2]
d =
    1    6    2
>> G=tf(n,d)
G =
      s + 7
  -----
 s^2 + 6 s + 2
Continuous-time transfer function.
>> step(G)
```

Figura 11.22 Ejemplo 3 Controlador PID: Función de transferencia.

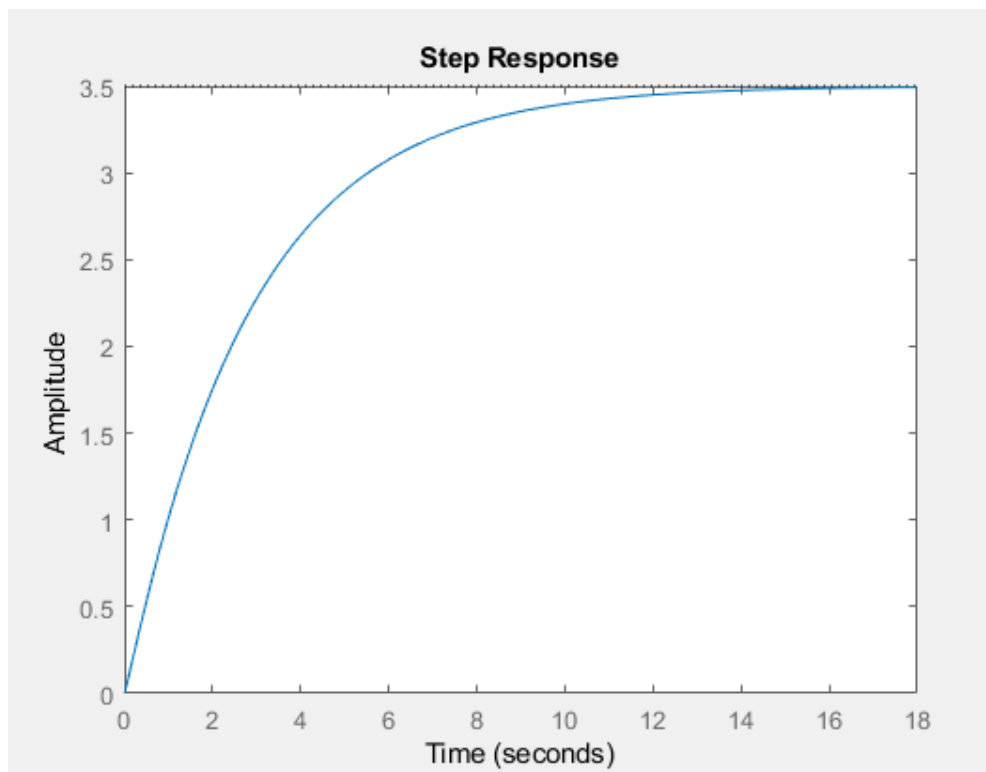


Figura 11.23 Ejemplo 3 Controlador PID: Gráfica de la función de transferencia.

```

Command Window

>> Kp=6

Kp =

     6

>> Ki= Kp/.5

Ki =

    12

>> kd=0.25

kd =

    0.2500

>> Gc=Kp+tf([Kp],[Ki 0])+tf([Kp*Kd 0],1)

Gc =

    45.36 s^2 + 72 s + 6
    -----
             12 s

Continuous-time transfer function.

```

Figura 11.24 Ejemplo 1 Controlador PID: Variables PID.

```

>> G_LC=G*Gc/(1+G*Gc)

G_LC =

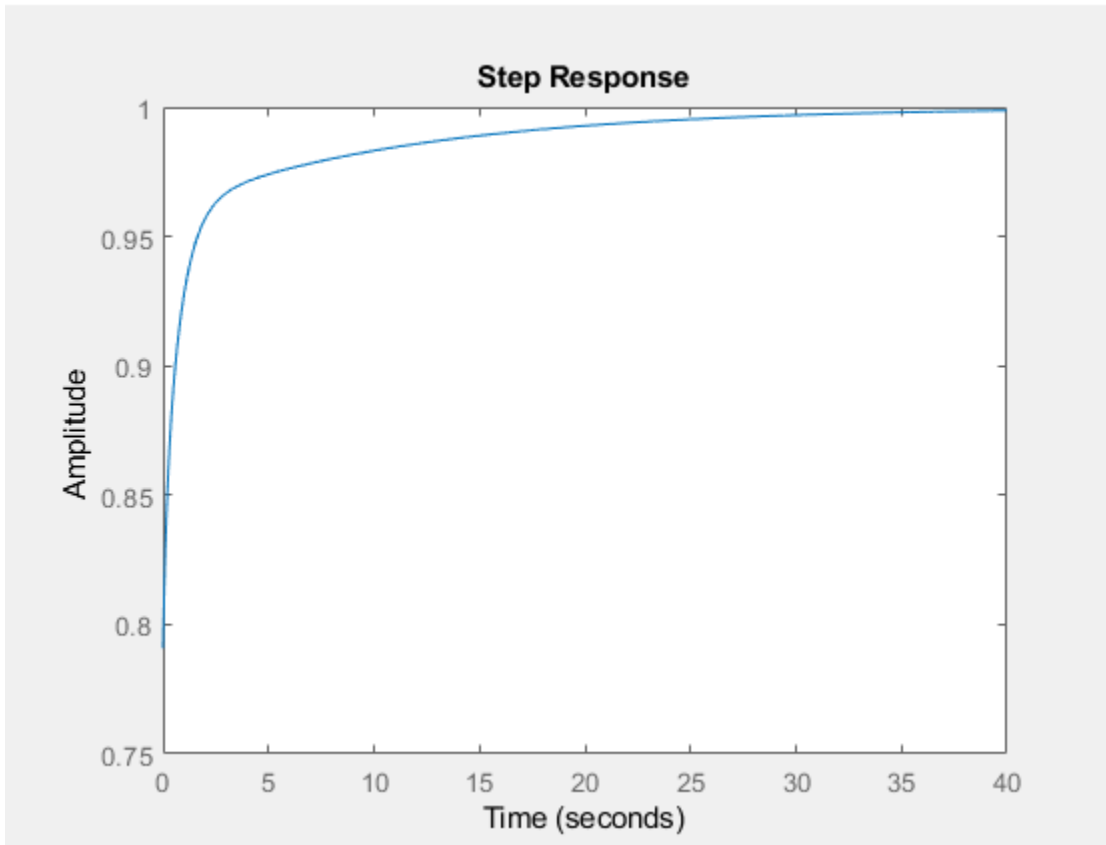
    544.3 s^6 + 7940 s^5 + 3.525e04 s^4 + 4.657e04 s^3 + 15264 s^2 + 1008 s
    -----
    688.3 s^6 + 9668 s^5 + 4.101e04 s^4 + 5.003e04 s^3 + 15840 s^2 + 1008 s

Continuous-time transfer function.

>> step(G_LC)

```

Figura 11.25 Ejemplo 1 Controlador PID: Función de transferencia a lazo cerrado.



*Figura 11.26 Ejemplo 1 Controlador PID: Respuesta de salida.*

## 12. RESPUESTA EN FRECUENCIA

Una forma de obtener una respuesta en frecuencia en un sistema es utilizando una señal de entrada senoidal, al ingresar una señal senoidal puede ocurrir dos situaciones:

1. La señal se atenúe
2. La señal se desfase

### 12.1 DIAGRAMA DE BODE

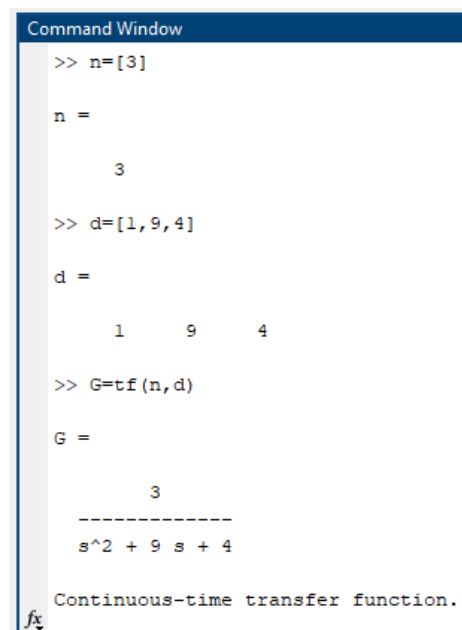
Para poder calcular la atenuación o desfase se utilizan los Diagramas de Bode.

1. Se debe de declarar la variable que se estará utilizando, en este caso “s”, con el siguiente comando:

*syms s*

2. Posteriormente se agregará la función de transferencia con los siguientes comandos.

*n = [3]*  
*d = [1,9,4]*  
*G = tf(n,d)*

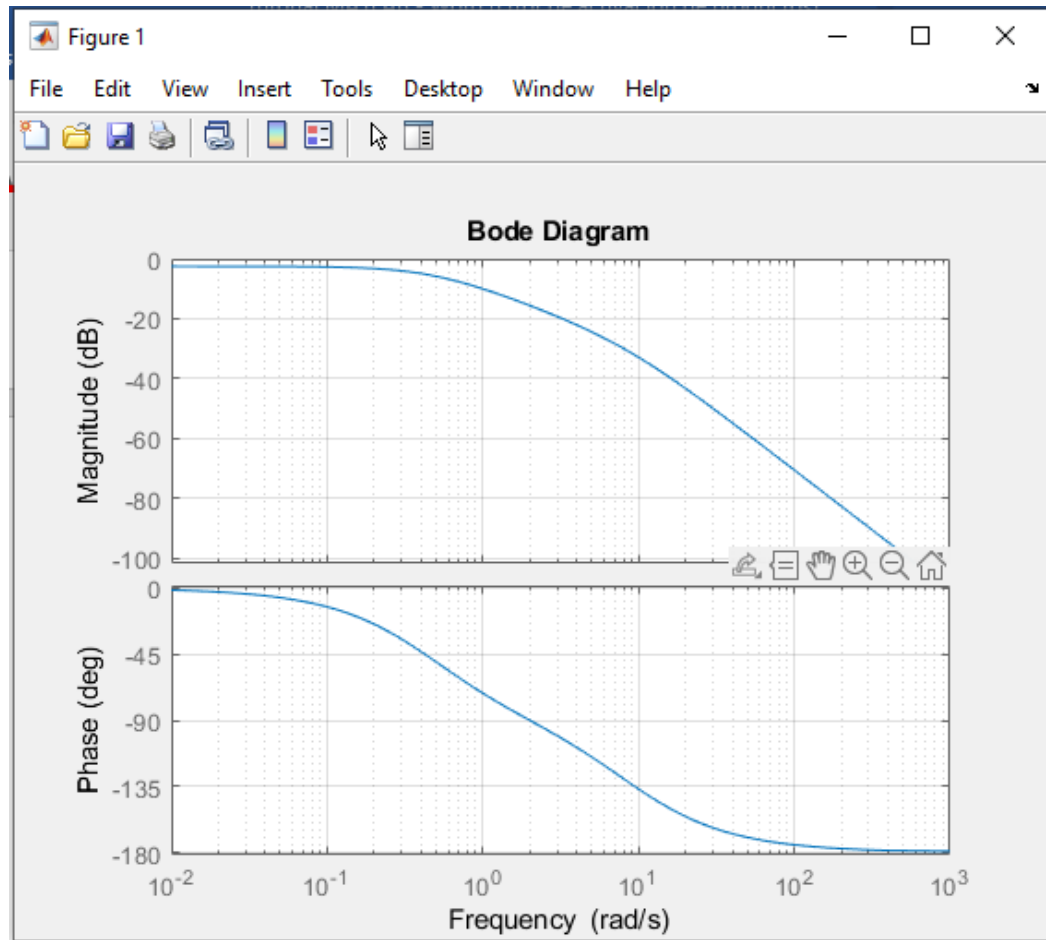


```
Command Window
>> n=[3]
n =
    3
>> d=[1,9,4]
d =
    1    9    4
>> G=tf(n,d)
G =
    3
-----
s^2 + 9 s + 4
Continuous-time transfer function.
```

Figura 12. Función de transferencia.

3. Para obtener nuestros diagramas de bode, se ingresa un comando muy sencillo:

***Bode (G)***



*Figura 12.1 Diagrama de Bode de la función de transferencia.*



## 12.1.1 EJEMPLO 1

$$G(S) = \frac{s^2+5s+2}{(s+2)}$$

```
Command Window
n =
     1     5     2
>> d=[1,2]
d =
     1     2
>> G=tf(n,d)
G =
|
s^2 + 5 s + 2
-----
s + 2
Continuous-time transfer function.
>> bode(G)
```

Figura 12.2 Ejemplo 1: Diagrama de Bode.

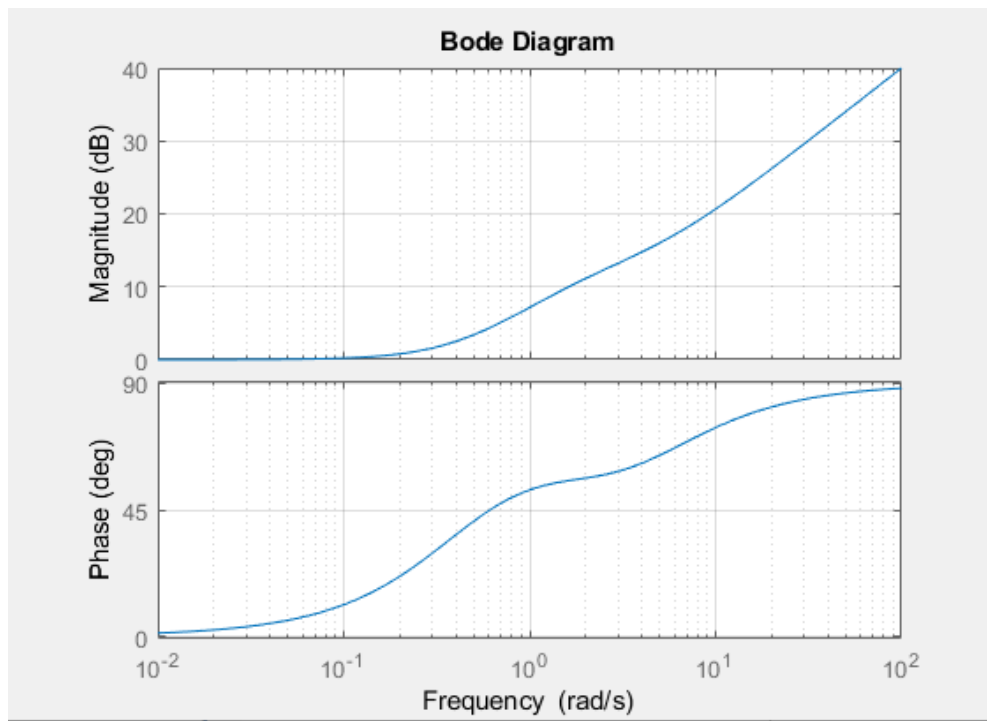


Figura 12.3 Ejemplo 1: Gráfica diagrama de Bode.

## 12.1.2 EJEMPLO 2

$$G(S) = \frac{s^2 + 7s}{(s+7)}$$

```
>> n=[1,7,0]
n =
     1     7     0
>> d=[1,7]
d =
     1     7
>> G=tf(n,d)
G =
      s^2 + 7 s
      -----
       s + 7
Continuous-time transfer function.
>> bode(G)
```

Figura 12.4 Ejemplo 2: Diagrama de Bode.

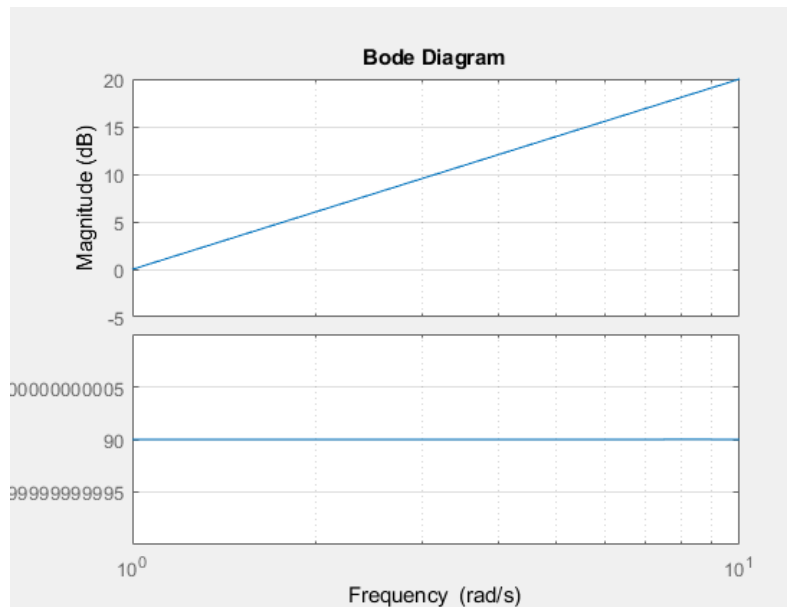


Figura 12.5 Ejemplo 2: Gráfica diagrama de Bode.

### 12.1.3 EJEMPLO 3

$$G(S) = \frac{s+9}{(2s^2+s)}$$

```
Command Window
>> n=[1,9]
n =
     1     9
>> d=[2,1,0]
d =
     2     1     0
>> G=tf(n,d)
G =
      s + 9
      -----
    2 s^2 + s
Continuous-time transfer function.
>> bode(G)
```

Figura 12.6 Ejemplo 3: Diagrama de Bode.

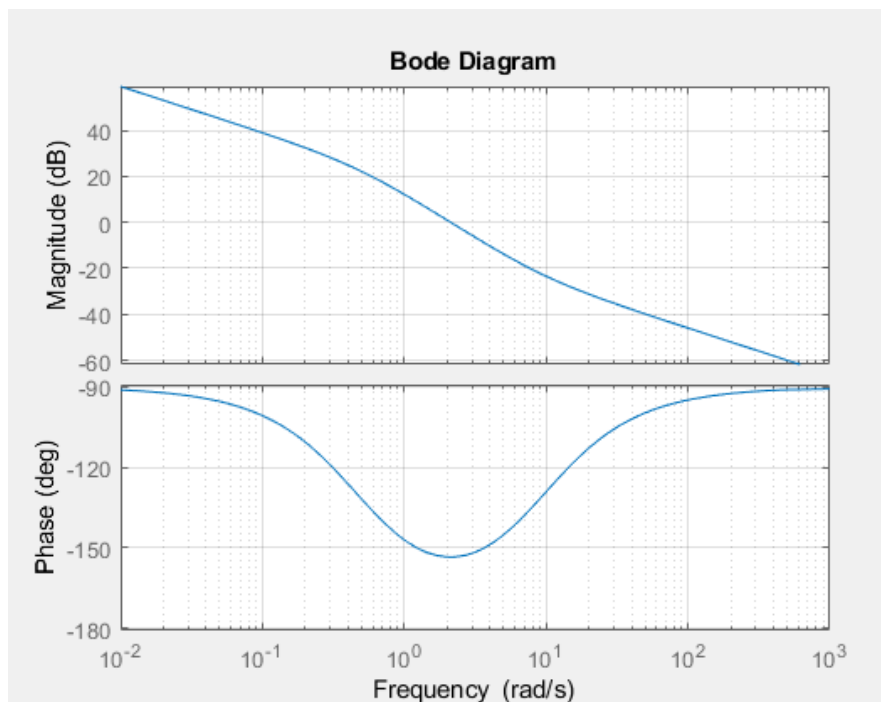


Figura 12.7 Ejemplo 3: Gráfica diagrama de Bode.

## 12.1.4 EJERCICIO 1

$$H(S) = \frac{s(s+5)}{(s+6)}$$

```
Command Window
>> n=[1,5,0]
n =
    1    5    0
>> d=[1,6]
d =
    1    6
>> G=tf(n,d)
G =
    s^2 + 5 s
    -----
    s + 6
Continuous-time transfer function.
>> bode(G)
```

Figura 12.8 Ejercicios 1: Diagrama de Bode.

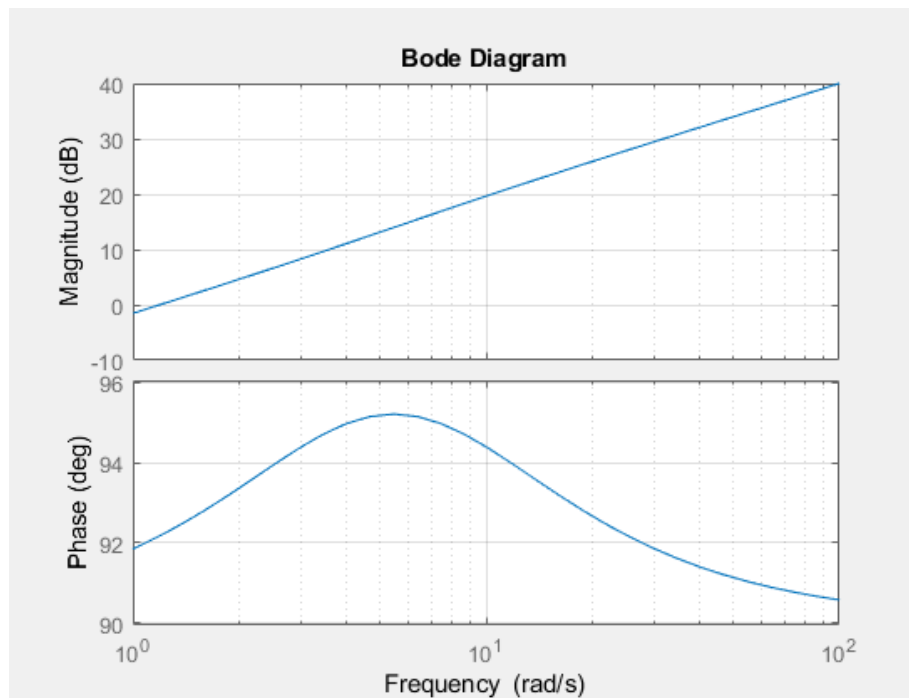


Figura 12.9 Ejemplo 1: Gráfica diagrama de Bode.

## 13. ESTABILIDAD DE UN SISTEMA

### 13.1 ESTABILIDAD POR EL MÉTODO DE ROUTH-HURWITZ

Teniendo el polinomio característico (polinomio que tenemos en el denominador de la función de transferencia), de la siguiente forma:

$$G(s) = a_n s^n + a_{n-1} s^{n-1} + a_{n-2} s^{n-2} + \dots + a_1 s^1 + a_0 = 0$$

Donde  $G(s)$  es la ecuación característica de un sistema.

Sabiendo que el plano cartesiano el lado derecho es la región inestable, mientras que el lado izquierdo es la región estable.

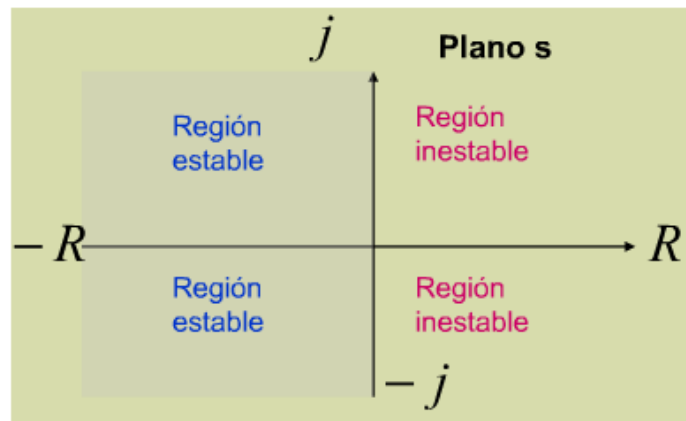


Figura 13. Plano cartesiano de estabilidad o inestabilidad.

La condición necesaria, pero no suficiente, para la estabilidad es que todos los **coeficientes** de la ecuación estén presentes y **tengan signo positivo**, es decir, si en la aplicación del teorema, existe un cambio de signo o aparece un número negativo, se dice que el sistema es inestable.

Saber si un sistema es estable o no utilizando el teorema de Routh-Hurwitz por medio de Matlab se puede realizar de diversos métodos.

1. Para declarar los coeficientes del polinomio característico se ingresa el siguiente comando.

```
coeffVector = input('input vector of your system coefficients: \n i.e. [an  
an-1 an-2 ... a0] = ');
```

2. Con los siguientes comandos se van a organizar las primeras dos filas para después proceder a armar la tabla del criterio de Routh-Hurwitz.

```
coeffLength=length(coeffVector);
```

```
rhTableColumn=round(coeffLength/2);
```

3. Posteriormente se agregará un nuevo comando para agregar una matriz de ceros vacía (Routh-Hurwitz).

```
rhTable = zeros(coeffLength,rhTableColumn);
```

4. Para poder agregar a la matriz los valores de la primera fila se ingresa el siguiente comando:

```
rhTable(1,:) = coeffVector(1,1:2:coeffLength);
```

5. Para verificar los coeficientes del polinomio son pares o impares se agrega el siguiente comando:

```
if (rem(coeffLength,2) ~= 0);
```

6. Teniendo el comando anterior, sabiendo si los coeficientes son pares o impares, los impares se irán a la segunda fila, con los siguientes comandos:

```
rhTable(2,1:rhTableColumn - 1) = coeffVector(1,2:2:coeffLength);
```

```
else
```

```
rhTable(2,:) = coeffVector(1,2:2:coeffLength);
```

```
end
```

```

coeffVector = input('input vector of your system coefficients: \n i.e. [an an-1 an-2 ... a0] = ');
coeffLength = length(coeffVector);
rhTableColumn = round(coeffLength/2);

rhTable = zeros(coeffLength,rhTableColumn);

rhTable(1,:) = coeffVector(1,1:2:coeffLength);

if (rem(coeffLength,2) ~= 0)
    rhTable(2,1:rhTableColumn - 1) = coeffVector(1,2:2:coeffLength);
else
    rhTable(2,:) = coeffVector(1,2:2:coeffLength);
end

```

*Figura 13.1 Construcción de la Tabla de Routh-Huwitz.*

7. Para calcular los siguientes coeficientes de las siguientes filas se utilizan los siguientes comandos:

```

    epss = 0.01;
    for i = 3:coeffLength

```

8. Para la fila de los ceros, utilizamos los siguientes comandos:

```

        if rhTable(i-1,:) == 0
            order = (coeffLength - i);
            cnt1 = 0;
            cnt2 = 1;
            for j = 1:rhTableColumn - 1
                rhTable(i-1,j) = (order - cnt1) * rhTable(i-2,cnt2);
                cnt2 = cnt2 + 1;
                cnt1 = cnt1 + 2;
            end
        end
    end
    for j = 1:rhTableColumn - 1

```

9. Para calcular el primer elemento de la fila superior se utiliza el siguiente comando:

```

        firstElemUpperRow = rhTable(i-1,1);

```

10. Para cada elemento de la tabla:

$$rhTable(i,j) = ((rhTable(i-1,1) * rhTable(i-2,j+1)) - \dots \\ (rhTable(i-2,1) * rhTable(i-1,j+1))) / firstElemUpperRow; \\ \text{End}$$

11. Para los ceros de la primera fila:

```
if rhTable(i,1) == 0  
rhTable(i,1) = epss;  
end  
end
```

```
epss = 0.01;  
  
] for i = 3:ceoffLength  
  
    if rhTable(i-1,:) == 0  
        order = (ceoffLength - i);  
        cnt1 = 0;  
        cnt2 = 1;  
    ]  
    for j = 1:rhTableColumn - 1  
        rhTable(i-1,j) = (order - cnt1) * rhTable(i-2,cnt2);  
        cnt2 = cnt2 + 1;  
        cnt1 = cnt1 + 2;  
    end  
end  
  
] for j = 1:rhTableColumn - 1  
  
    firstElemUpperRow = rhTable(i-1,1);  
  
    rhTable(i,j) = ((rhTable(i-1,1) * rhTable(i-2,j+1)) - ....  
        (rhTable(i-2,1) * rhTable(i-1,j+1))) / firstElemUpperRow;  
end  
  
    if rhTable(i,1) == 0  
        rhTable(i,1) = epss;  
    end  
end
```

Figura 13.2 Construcción de la Tabla de Routh-Hwitz: Coeficientes.



12. Posteriormente se calcularán los polos inestables.

```
unstablePoles = 0;
```

13. Se ingresa el siguiente comando para comprobar si existe un cambio de signos,

```
for i = 1:ceoffLength - 1  
if sign(rhTable(i,1)) * sign(rhTable(i+1,1)) == -1  
unstablePoles = unstablePoles + 1;  
end  
end
```

14. Ya calculados todos los datos anteriores, se ingresa el siguiente comando para que nuestra tabla aparezca en la pantalla.

```
fprintf('\n Routh-Hurwitz Table:\n')  
rhTable
```

```
unstablePoles = 0;  
  
for i = 1:ceoffLength - 1  
    if sign(rhTable(i,1)) * sign(rhTable(i+1,1)) == -1  
        unstablePoles = unstablePoles + 1;  
    end  
end  
  
fprintf('\n Routh-Hurwitz Table:\n')  
rhTable
```

Figura 13.3 Comandos para que aparezca en pantalla de tala de R-H.

15. Finalmente, para saber si nuestro sistema es estable o inestable, ingresaremos los siguientes comandos:

```
if unstablePoles == 0  
fprintf('~~~~~> it is a stable system! <~~~~~\n')  
else
```

```

        fprintf('~~~~~> it is an unstable system! <~~~~~\n')
        end
fprintf('\n Number of right hand side poles =%2.0f\n',unstablePoles)
reply = input('Do you want roots of system be shown? Y/N ', 's');
        if reply == 'y' || reply == 'Y'
            sysRoots = roots(coeffVector);
            fprintf('\n Given polynomial coefficients roots :\n')
            sysRoots
            end
if unstablePoles == 0
    fprintf('~~~~~> it is a stable system! <~~~~~\n')
else
    fprintf('~~~~~> it is an unstable system! <~~~~~\n')
end

fprintf('\n Number of right hand side poles =%2.0f\n',unstablePoles)

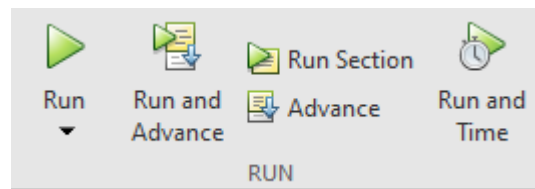
reply = input('Do you want roots of system be shown? Y/N ', 's');
if reply == 'y' || reply == 'Y'
    sysRoots = roots(coeffVector);
    fprintf('\n Given polynomial coefficients roots :\n')
    sysRoots
end

```

---

*Figura 13.4 Comandos para indicar estabilidad o inestabilidad.*

16. Ya teniendo nuestro programa listo, se le da “Run” al programa para poderlo correr, ingresando los coeficientes de nuestro polinomio.



*Figura 13.5 Run para el programa.*

```
Command Window
input vector of your system coefficients:
i.e. [an an-1 an-2 ... a0] = [1 3 7 2 8 1 2 2]

Routh-Hurwitz Table:

rhTable =

    1.0000    7.0000    8.0000    2.0000
    3.0000    2.0000    1.0000    2.0000
    6.3333    7.6667    1.3333         0
   -1.6316    0.3684    2.0000         0
    9.0968    9.0968         0         0
    2.0000    2.0000         0         0
   -0.0000         0         0         0
    2.0000         0         0         0

~~~~~> it is an unstable system! <~~~~~

Number of right hand side poles = 4
```

*Figura 13.6 Programa en ejecución.*

### 13.1.1 EJEMPLO 1

Teniendo el siguiente polinomio característico de una función de transferencia, calcular si el sistema es estable o inestable utilizando el teorema de Routh-Hurwitz:

$$9s^6 + 5s^5 + 2s^4 + 8s^3 + 9s^2 + s^1 + 5 = 0$$

```
Command Window

input vector of your system coefficients:
i.e. [an an-1 an-2 ... a0] = [9 5 2 8 9 1 5]

Routh-Hurwitz Table:

rhTable =

    9.0000    2.0000    9.0000    5.0000
    5.0000    8.0000    1.0000         0
   -12.4000    7.2000    5.0000         0
   10.9032    3.0161         0         0
   10.6302    5.0000         0         0
    -2.1123         0         0         0
    5.0000         0         0         0

~~~~~> it is an unstable system! <~~~~~

Number of right hand side poles = 4
fx Do you want roots of system be shown? Y/N |
```

Figura 13.7 Ejemplo 1. Estabilidad criterio de Routh-Hurwitz.

## 13.1.2 EJEMPLO 2

Teniendo el siguiente polinomio característico de una función de transferencia, calcular si el sistema es estable o inestable utilizando el teorema de Routh-Hurwitz:

$$s^4 + 5s^3 + 7s^2 + 2s^1 + 9 = 0$$

```
Command Window

input vector of your system coefficients:
i.e. [an an-1 an-2 ... a0] = [1 5 7 2 9]

Routh-Hurwitz Table:

rhTable =

    1.0000    7.0000    9.0000
    5.0000    2.0000         0
    6.6000    9.0000         0
   -4.8182         0         0
    9.0000         0         0

~~~~~> it is an unstable system! <~~~~~

Number of right hand side poles = 2
fx Do you want roots of system be shown? Y/N |
```

Figura 13.8 Ejemplo 2. Estabilidad criterio de Routh-Hurwitz.

### 13.1.3 EJEMPLO 3

Teniendo el siguiente polinomio característico de una función de transferencia, calcular si el sistema es estable o inestable utilizando el teorema de Routh-Hurwitz:

$$s^6 + 5s^5 + 7s^4 + 10s^3 + s^2 + 3s^1 + 4 = 0$$

```
Command Window
input vector of your system coefficients:
i.e. [an an-1 an-2 ... a0] = [1 5 7 10 1 3 4]

Routh-Hurwitz Table:

rhTable =

    1.0000    7.0000    1.0000    4.0000
    5.0000   10.0000    3.0000         0
    5.0000    0.4000    4.0000         0
    9.6000   -1.0000         0         0
    0.9208    4.0000         0         0
   -42.7014         0         0         0
    4.0000         0         0         0

~~~~~> it is an unstable system! <~~~~~

Number of right hand side poles = 2
fx Do you want roots of system be shown? Y/N |
```

Figura 13.9 Ejemplo 3. Estabilidad criterio de Routh-Hurwitz.

## 13.2 ESTABILIDAD POR EL MÉTODO DE LGR

Lugar Geométrico de las Raíces (LGR) es otro método para poder conocer la estabilidad de un sistema. Consiste en obtener el comportamiento de los polos y ceros de una manera gráfica. Nos muestra que sucede con las variaciones de  $k$ , y en que puntos el sistema es estable o inestable.

Podemos obtener las gráficas con las asíntotas de los polos y ceros por medio de Matlab.

1. Primero debemos ingresar la función de transferencia de la cual queremos conocer su estabilidad, con los siguientes comandos:

**$num = [1\ 4\ 2\ 6\ 3]$**

**$den = [1\ 4\ 2\ 8\ 4\ 1]$**

**$G=tf(num,den)$**

```
>> num = [1 4 2 6 3]
den = [1 4 2 8 4 1]
G=tf(num,den)

num =

    1    4    2    6    3

den =

    1    4    2    8    4    1

G =

      s^4 + 4 s^3 + 2 s^2 + 6 s + 3
-----
      s^5 + 4 s^4 + 2 s^3 + 8 s^2 + 4 s + 1

Continuous-time transfer function.
```

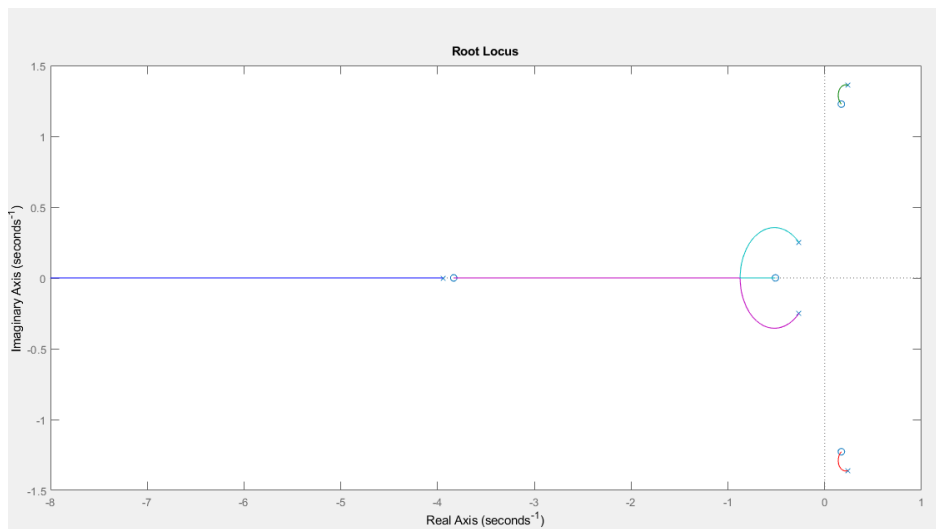
Figura 13.10 Función de transferencia.

2. Para obtener la gráfica del Lugar Geométrico de las Raíces, ingresamos el siguiente comando:

*rlocus(G)*

```
>> rlocus(G)
fx >> |
```

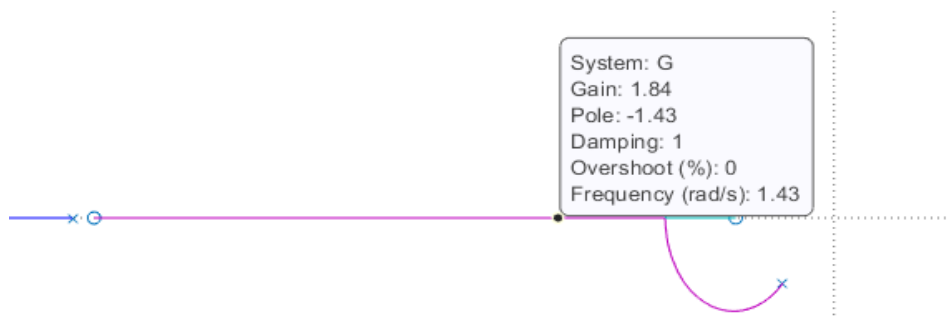
*Figura 13.10 Comando para obtener las gráficas LGR.*



*Figura 13.11 Gráfica LGR.*

*Con la gráfica podemos observar que existe inestabilidad en el sistema, se observan los polos señalados con “x”, mientras que los ceros con “•”.*

3. Se puede interactuar con la gráfica para conocer información sobre las asíntotas, esto se realiza dando click izquierdo a algún punto de la asíntota.



*Figura 13.12 Información de las asíntotas.*

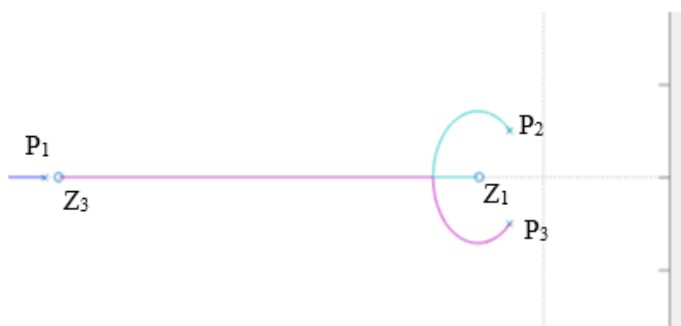


La información que nos arroja la asíntota es:

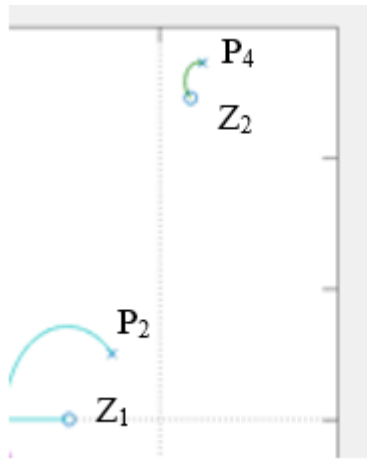
- **System:** Sistema que esta representando
- **Gain:** Valor de la ganancia.
- **Pole:** Nos indica las coordenadas (parte real e imaginaria) asociadas a este polo.
- **Damping:** Factor de amortiguamiento correspondiente al polo ubicado en la posición del cursor.
- **Overshoot:** Indica el nivel de sobreoscilación que tendría.
- **Frequency:** Frecuencia correspondiente al polo (coincide con la distancia del mismo al origen del plano complejo).

### *Interpretación de la gráfica*

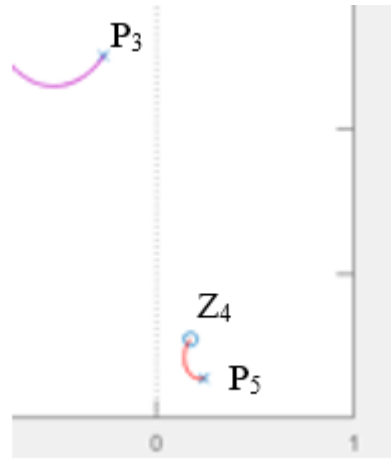
Ceros	Polos
$Z_1 = -3.8333 + 0.0000 i$	$P_1 = -3.9460 + 0.0000 i$
$Z_2 = 0.1714 + 1.2274 i$	$P_2 = -0.2655 + 0.2497 i$
$Z_3 = -0.5096 + 0.0000 i$	$P_3 = -0.2655 - 0.2497 i$
$Z_4 = 0.1714 - 1.2274 i$	$P_4 = 0.2385 + 1.3603 i$
	$P_5 = 0.2385 - 1.3603 i$



*Figura 13.13 El polo  $P_2$  tiende a cero  $Z_1$ , el polo  $P_3$  tiende a cero  $Z_3$ , mientras que el polo  $P_1$  tiende a cero en el infinito.*



*Figura 13.14 El polo  $P_4$  tiende a cero  $Z_2$ .*



*Figura 13.15 El polo  $P_2$  tiende a cero  $Z_4$ .*

### 13.2.1 EJEMPLO 1

Teniendo la siguiente función de transferencia, calcular si el sistema es estable o inestable utilizando el método de Lugar Geométrico de las Raíces y darle la interpretación a la gráfica.

$$G(s) = \frac{s^5 + 4s^4 + 10s^3 + 2s^2 + 5s + 3}{s^6 + 7s^5 + 2s^4 + 10s^3 + 4s^2 + s + 4}$$

```
>> num=[1 4 10 2 5 3]
num =
     1     4    10     2     5     3
>> den=[1 7 2 10 4 1 4]
den =
     1     7     2    10     4     1     4
>> G=tf(num,den)
G =
      s^5 + 4 s^4 + 10 s^3 + 2 s^2 + 5 s + 3
-----
      s^6 + 7 s^5 + 2 s^4 + 10 s^3 + 4 s^2 + s + 4
Continuous-time transfer function.
>> rlocus(G)
```

Figura 13.16 Ejemplo 1: LGR

### Interpretación de la gráfica

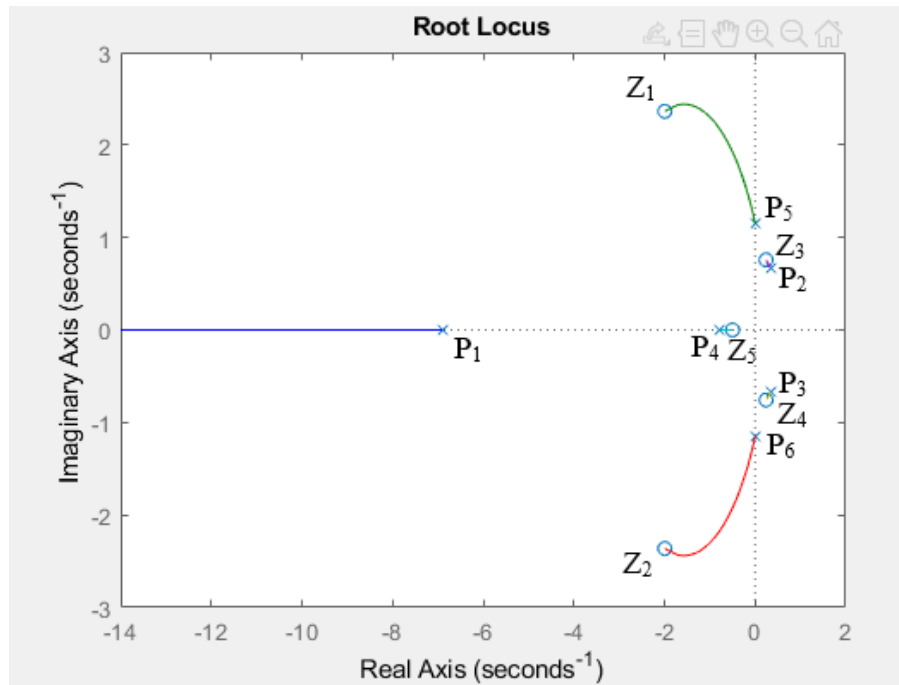


Figura 13.17 El polo  $P_1$  tiende a cero en el infinito, el polo  $P_2$  tiende a cero  $Z_3$ , el polo  $P_3$  tiende a cero  $Z_4$ , el polo  $P_4$  tiende a cero  $Z_5$ ,  $P_5$  tiende a cero  $Z_1$ , mientras que el polo  $P_6$  tiende a cero  $Z_2$ .

Ceros	Polos
$Z_1 = -1.9936 + 2.3627i$	$P_1 = -6.9081 + 0.0000i$
$Z_2 = -1.9936 - 2.3627i$	$P_2 = 0.0037 + 1.1476i$
$Z_3 = 0.2418 + 0.7575i$	$P_3 = 0.0037 - 1.1476i$
$Z_4 = 0.2418 - 0.7575i$	$P_4 = -0.7727 + 0.0000i$
$Z_5 = -0.4964 + 0.0000i$	$P_5 = 0.3366 + 0.6750i$
	$P_6 = 0.3366 - 0.6750i$

*Nota: Los polos y ceros se obtiene de la información que nos arroja las asíntotas de la gráfica.*

## 13.2.2 EJEMPLO 2

Teniendo la siguiente función de transferencia, calcular si el sistema es estable o inestable utilizando el método de Lugar Geométrico de las Raíces y darle la interpretación a la gráfica.

$$G(s) = \frac{s^5 + 2s^4 + 8s^3 + 7s^2 + s + 4}{2s^5 + 6s^4 + 2s^3 + 7s^2 + 2s + 2}$$

```
>> num=[1 2 8 7 1 4]

num =

     1     2     8     7     1     4

>> den=[2 6 2 7 2 2]

den =

     2     6     2     7     2     2

>> G=tf(num,den)

G =

      s^5 + 2 s^4 + 8 s^3 + 7 s^2 + s + 4
-----
    2 s^5 + 6 s^4 + 2 s^3 + 7 s^2 + 2 s + 2

Continuous-time transfer function.

>> rlocus(G)
```

Figura 13.18 Ejemplo 2: LGR

### Interpretación de la gráfica

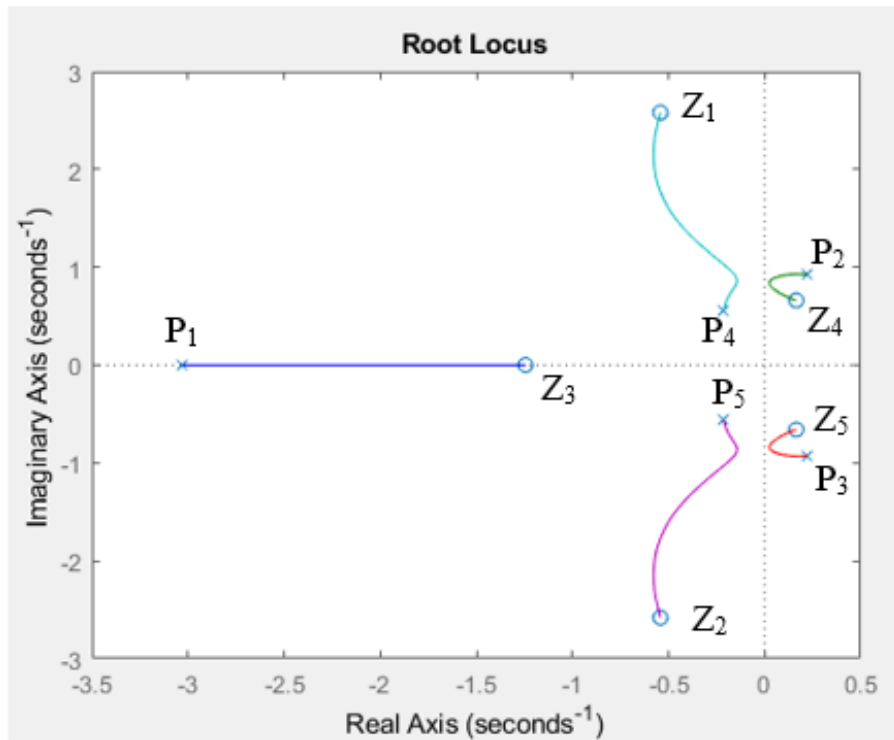


Figura 13.19 El polo  $P_1$  tiende a cero  $Z_3$ , el polo  $P_2$  tiende a cero  $Z_4$ , el polo  $P_3$  tiende a cero  $Z_5$ , el polo  $P_4$  tiende a cero  $Z_1$ , mientras que el polo  $P_5$  tiende a cero  $Z_2$ .

Ceros	Polos
$Z_1 = -0.5422 + 2.5805i$	$P_1 = -3.0274 + 0.0000i$
$Z_2 = -0.5422 - 2.5805i$	$P_2 = 0.2249 + 0.9271i$
$Z_3 = -1.2439 + 0.0000i$	$P_3 = 0.2249 - 0.9271i$
$Z_4 = 0.1642 + 0.6600i$	$P_4 = -0.2112 + 0.5642i$
$Z_5 = 0.1642 - 0.6600i$	$P_5 = -0.2112 - 0.5642i$

*Nota: Los polos y ceros se obtiene de la información que nos arroja las asíntotas de la gráfica.*

### 13.2.3 EJEMPLO 3

Teniendo la siguiente función de transferencia, calcular si el sistema es estable o inestable utilizando el método de Lugar Geométrico de las Raíces y darle la interpretación a la gráfica.

$$G(s) = \frac{s^5 + 4s^4 + 3s^3 + 2s^2 + s + 3}{s^5 + 3s^4 + 9s^3 + s^2 + 2s + 4}$$

```
>> num=[1 4 3 2 1 3]

num =

     1     4     3     2     1     3

>> den=[1 3 9 1 2 4]

den =

     1     3     9     1     2     4

>> G=tf(num,den)

G =

      s^5 + 4 s^4 + 3 s^3 + 2 s^2 + s + 3
      -----
      s^5 + 3 s^4 + 9 s^3 + s^2 + 2 s + 4

Continuous-time transfer function.

>> rlocus(G)
```

Figura 13.20 Ejemplo 3: LGR

### Interpretación de la gráfica

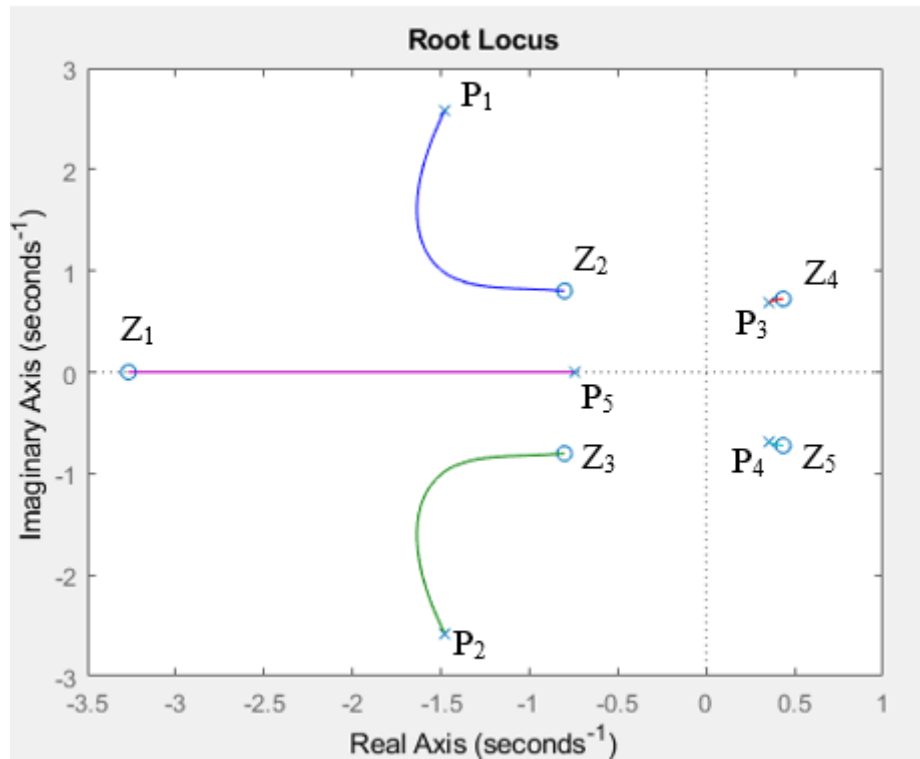


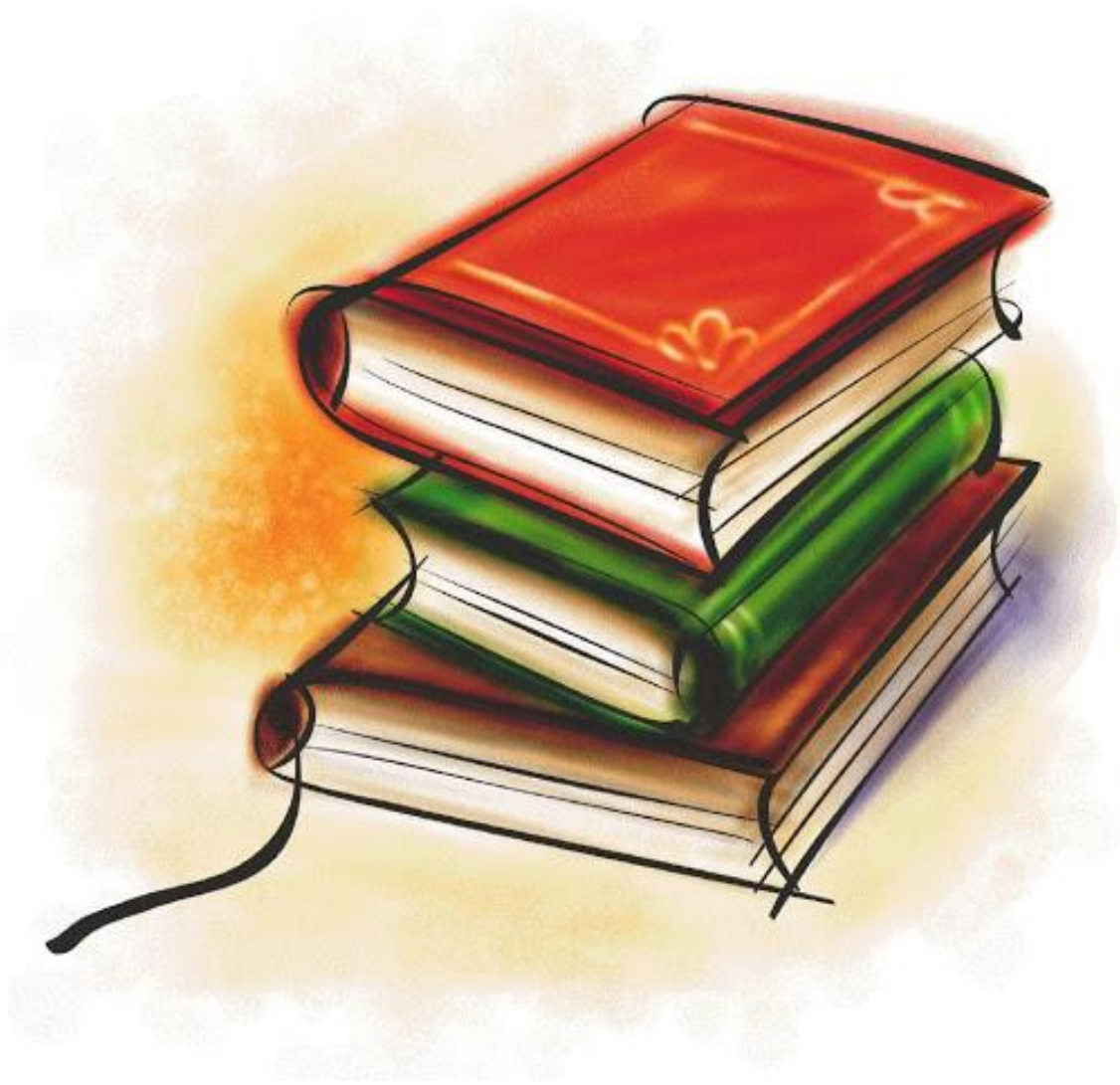
Figura 13.21 El polo  $P_1$  tiende a cero  $Z_3$ , el polo  $P_2$  tiende a cero  $Z_4$ , el polo  $P_3$  tiende a cero  $Z_5$ , el polo  $P_4$  tiende a cero  $Z_1$ , mientras que el polo  $P_5$  tiende a cero  $Z_2$ .

Ceros	Polos
$Z_1 = -3.2667 + 0.0000i$	$P_1 = -1.4815 + 2.5732i$
$Z_2 = -0.8018 + 0.8022i$	$P_2 = -1.4815 - 2.5732i$
$Z_3 = -0.8018 - 0.8022i$	$P_3 = 0.3556 + 0.6928i$
$Z_4 = 0.4352 + 0.7242i$	$P_4 = 0.3556 - 0.6928i$
$Z_5 = 0.4352 - 0.7242i$	$P_5 = -0.7482 + 0.0000i$

*Nota: Los polos y ceros se obtiene de la información que nos arroja las asíntotas de la gráfica.*



## BIBLIOGRAFÍA



Apuntes de Teoría de control y Robótica, del Ing. David Tinoco.

MathWorks.