

Manual de Prácticas

Biomecatrónica

Andrés Quintero Zea, PhD.

Universidad ELA

Manual de prácticas – Biomecatrónica

Andrés Quintero Zea, PhD. (andres.quintero27@eia.edu.co)

MATLAB®, Simulink® y Control Systems Toolbox™ son marcas comerciales de The MathWorks, Inc. The MathWorks no garantiza la exactitud del texto ni de los ejercicios de este libro. El uso o discusión de este libro sobre el software MATLAB®, Simulink® y Control Systems Toolbox™ o productos relacionados no constituye respaldo o patrocinio por parte de The MathWorks de un enfoque pedagógico particular o uso particular del software MATLAB®, Simulink® y Control Systems Toolbox™.

Este libro fue escrito por el autor en \LaTeX usando las fuentes tipográficas Myriad Pro para encabezados de sección, Cronos Pro para texto corrido, Minion Pro para ecuaciones matemáticas y Roboto Mono para fragmentos de código.

Copyright ©2023 Andrés Quintero Zea

Imagen de la portada creada por Inteligencia Artificial usando la herramienta [Microsoft Copilot](#)



Esta obra está bajo una Licencia Creative Commons
Atribución-NoComercial-CompartirIgual 4.0 Internacional

Resumen: <https://creativecommons.org/licenses/by-nc-sa/4.0/>

Licencia: <https://creativecommons.org/licenses/by-nc-sa/4.0/legalcode>

Prefacio

Bienvenido al “Manual de Prácticas – Biomecatrónica”. Este recurso ha sido diseñado específicamente para estudiantes de Ingeniería Biomédica de la Universidad EIA, con el fin de explorar los fundamentos del control analógico y aplicar estos conocimientos a la solución de problemas en el ámbito de la salud.

La ingeniería biomédica desempeña un papel crucial en la mejora de la atención médica mediante la aplicación de principios y técnicas de ingeniería a los desafíos en el campo de la medicina. Este manual se centra en la aplicación de conceptos de control analógico en situaciones relevantes para la ingeniería biomédica, como el diseño de sistemas de control para dispositivos médicos, monitoreo de señales fisiológicas y mejora de la precisión en instrumentación biomédica.

Este manual ha sido diseñado para proporcionar una experiencia de aprendizaje práctica y relevante para los estudiantes de ingeniería biomédica. A medida que te sumerjas en los ejercicios prácticos y estudios de caso, te alentamos a considerar el impacto positivo que la ingeniería biomédica puede tener en la salud y el bienestar de las personas.

Esperamos que este recurso sirva como un valioso compañero en tu trayectoria académica y profesional en el emocionante campo de la ingeniería biomédica. ¡Disfruta del viaje!

Andrés Quintero Zea, PhD.

Índice general

Prefacio	i
1. Introducción al <i>Control System Toolbox</i>	1
1.1. Introducción al <i>Control System Toolbox</i>	2
1.2. Creación y Manipulación de Sistemas Dinámicos	2
1.2.1. Función de Transferencia	3
1.2.2. Cero – Polo – Ganancia	5
1.2.3. Espacio de Estado	6
1.3. Análisis de Respuesta Transitoria y en Frecuencia	8
1.3.1. Respuesta en el tiempo	8
1.3.2. Respuesta en la frecuencia	11
1.4. Trabajo práctico	13
1.4.1. Prodecimiento	13
1.4.2. Evaluación	14

Índice de tablas

1.1.	Funciones para el análisis de respuesta temporal	9
1.3.	Funciones para el análisis de respuesta frecuencial	11
1.5.	Parámetros del modelo	14

Índice de figuras

1.1.	Propiedades de un objeto tipo <code>tf</code>	4
1.2.	Propiedades de un objeto tipo <code>zpk</code>	6
1.3.	Propiedades de un objeto tipo <code>ss</code>	8
1.4.	Respuestas del sistema	9
1.5.	Respuesta del sistema usando <code>lsim</code>	11
1.6.	Gráficas para análisis de respuesta en frecuencia	12
1.7.	Sistema de suspensión utilizado en el análisis	13

Práctica 1

Introducción al *Control System Toolbox*

Introducción

En esta sesión, nos embarcaremos en un emocionante viaje para aprender a utilizar las funciones poderosas que ofrece el *Control System Toolbox* de MATLAB. Este conjunto de herramientas especializado nos permitirá diseñar y analizar sistemas de control analógico de manera eficiente y precisa.

Objetivo de la Práctica

El objetivo principal de esta práctica es familiarizarnos con las capacidades del *Control System Toolbox* y desarrollar habilidades prácticas en su aplicación. Al finalizar esta sesión, podrás:

1. Comprender las funciones clave del *Control System Toolbox*.
2. Simular sistemas de control analógico básicos utilizando herramientas específicas.
3. Ganar confianza en la manipulación de funciones y comandos para resolver problemas específicos de sistemas de control.

Contenido de la Práctica

Introducción al *Control System Toolbox*

Una visión general de las capacidades y herramientas disponibles en el *Control System Toolbox*, destacando su importancia en el diseño y análisis de sistemas de control.

Creación y Manipulación de Sistemas Dinámicos

Aprenderemos a utilizar funciones para crear modelos de sistemas dinámicos y a manipular sus características clave.

Análisis de Respuesta Transitoria y en Frecuencia

Exploraremos las funciones que nos permiten analizar la respuesta transitoria y en frecuencia de nuestros sistemas de control.

Metodología

La práctica consistirá en una serie de ejercicios prácticos guiados que te llevarán paso a paso a través de las funciones esenciales del *Control System Toolbox*. Te animamos a que experimentes con los comandos y explores cómo afectan a los sistemas de control que estamos diseñando.

Este conocimiento será fundamental para las prácticas posteriores, donde aplicaremos estas habilidades en contextos específicos.

Al final de la práctica, deberás construir un informe escrito con algunos ejercicios que desarrollarás, posteriormente, de manera independiente.

1.1 Introducción al *Control System Toolbox*

El *Control System Toolbox* es un conjunto de rutinas diseñadas para MATLAB, destinadas a aplicar las diversas herramientas derivadas de la teoría de control en sistemas lineales.

Este *toolbox* está integrado por cuatro grupos de funciones:

- Modelos de sistemas dinámicos.
- Análisis lineales.
- Diseño y ajuste de sistemas de control.
- Cálculos con matrices especiales de la teoría de control.

Cuando abordamos el análisis de un sistema específico, el primer paso implica la obtención de las ecuaciones dinámicas que describen su comportamiento, en un proceso conocido como *modelado*. En caso de que estas ecuaciones sean no lineales, se procede con un proceso conocido como *linealización* con el fin de obtener un modelo de comportamiento aproximadamente lineal en las vecindades de un punto de operación. Esto proporciona una descripción del sistema, ya sea en forma de función de transferencia o de variables de estado, que servirá como punto de partida para la *simulación* del comportamiento dinámico del sistema.

En esta práctica, comenzaremos explorando las diversas formas de ingresar modelos y luego nos adentraremos en las herramientas de análisis, examinando las respuestas en tiempo y frecuencia. Finalmente, concluiremos esta exploración con un enfoque en algunas funciones para la visualización de los resultados de simulación.

1.2 Creación y Manipulación de Sistemas Dinámicos

Todo intento de diseñar un sistema debe iniciarse con la anticipación de su rendimiento antes de que el sistema pueda ser detalladamente diseñado o construido en la realidad. Esta anticipación se fundamenta

en una *representación matemática* de las características dinámicas del sistema, conocida como *modelo matemático*. En el caso de numerosos sistemas físicos, los modelos matemáticos útiles se articulan en términos de *ecuaciones diferenciales*. Estas ecuaciones diferenciales se pueden manipular con el fin de llevarlas a formas estandarizadas. Para el caso de los sistemas de control, contamos con tres *representaciones estándar básicas*: función de transferencia, cero–polo-ganancia y variables de estado.

El *Control System Toolbox* incorpora funciones para la creación de objetos con atributos propios de cada uno de los tres tipos de representación. Es bueno aclarar que un mismo sistema puede ser representado en cualquiera de las tres formas y estas son *equivalentes* entre sí, pues representan un mismo sistema.

Función de Transferencia

Una función de transferencia es una representación matemática que describe la relación entre la entrada y la salida de un SLIT. La función de transferencia se expresa como la razón de Laplace de la transformada de la salida a la transformada de la entrada, bajo la suposición de condiciones iniciales nulas. La forma general de una función de transferencia $H(s)$:

$$G(s) = \frac{C(s)}{R(s)} = \frac{\text{num}(s)}{\text{den}(s)},$$

donde $H(s)$ es la función de transferencia, $C(s)$ es la transformada de Laplace de la salida, $R(s)$ es la transformada de Laplace de la entrada y s es la variable compleja de Laplace.

En MATLAB, un sistema representado en forma de función de transferencia se crea por medio de la función `tf` que recibe como argumentos los vectores numéricos que representan los polinomios asociados al numerador (`num`) y al denominador (`den`) de la fracción racional $G(s)$. Recuerda que en MATLAB el polinomio $s^5 + 4s^3 + 2s^2 + 3s + 7$ se representa con el vector `[1 0 4 2 3 7]`.

Por ejemplo, si tenemos el modelo de un sistema representado por la función de transferencia

$$G(s) = \frac{s^4 + 17s^3 + 99s^2 + 223s + 140}{s^5 + 32s^4 + 363s^3 + 2092s^2 + 5052s + 4320}$$

en MATLAB crearemos el objeto `tf` de la siguiente forma:

```
num = [1 17 99 223 140];
den = [1 32 363 2092 5052 4320];
G = tf(num, den)
```

que producirá como salida en el *Command Window*

G =

$s^4 + 17 s^3 + 99 s^2 + 223 s + 140$

$$s^5 + 32 s^4 + 363 s^3 + 2092 s^2 + 5052 s + 4320$$

Continuous-time transfer **function**.

Una vez creado nuestro objeto `tf`, podemos acceder a las propiedades de este usando la función `tfdata`,

```
[NUM,DEN] = tfdata(G)
```

que produce la salida

NUM =

1×1 cell array

{[0 1 17 99 223 140]}

DEN =

1×1 cell array

{[1 32 363 2092 5052 4320]}

donde NUM y DEN son variables de clase `cell` que contienen los polinomios del numerador y denominador, respectivamente. El objeto `tf` tiene otras propiedades que permiten personalizarlo con los datos del sistema. En la Figura 1.1 se muestran estas propiedades, que pueden ser visualizadas dando doble click sobre el nombre de la variable del objeto `tf`.

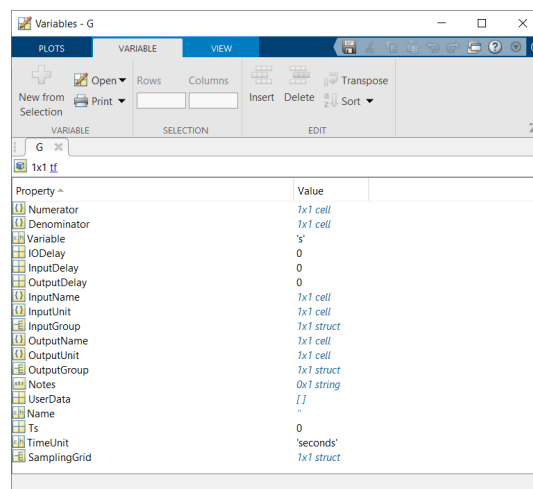


Figura 1.1: Propiedades de un objeto tipo `tf`

Cero – Polo – Ganancia

Un modelo de cero–polo–ganancia es una forma específica de representar una función de transferencia, especialmente útil en el análisis y diseño de sistemas dinámicos. En este tipo de modelo, la función de transferencia se expresa en función de los ceros y polos del sistema, junto con un término de ganancia. La forma general de una función de transferencia de cero-polo-ganancia es:

$$G(s) = K \frac{(s - z_1)(s - z_2) \dots (s - z_m)}{(s - p_1)(s - p_2) \dots (s - p_n)},$$

donde $G(s)$ es la función de transferencia, K es la ganancia del sistema, z_1, z_2, \dots, z_m son los ceros y p_1, p_2, \dots, p_n son los polos del sistema.

En este contexto:

- Los ceros (z) son los valores de s que hacen que el numerador de la función de transferencia sea cero, lo que significa que la salida es cero en esos puntos específicos.
- Los polos (p) son los valores de s que hacen que el denominador de la función de transferencia sea cero, indicando las ubicaciones en las cuales el sistema puede volverse inestable o donde la respuesta puede tener singularidades.
- La ganancia (K) es un factor de escala que afecta la amplitud de la respuesta del sistema.

Este tipo de representación es especialmente útil para comprender cómo los ceros, polos y la ganancia afectan el comportamiento del sistema en el dominio de Laplace. Facilita el análisis y diseño de sistemas lineales al proporcionar una descripción estructurada y modular de la función de transferencia.

Si el modelo de nuestro sistema se encuentra expresado en función de los ceros, los polos y la ganancia, en MATLAB utilizaremos la función `zpk`, la cual crea un objeto `zpk`. Por ejemplo, si tenemos el siguiente modelo:

$$G(s) = \frac{20(s + 2)(s + 3)(s + 6)(s + 8)}{s^2(s + 7)(s + 9)(s + 10)(s + 15)}$$

ingresaremos en MATLAB el siguiente código:

```
Z = [-2, -3, -6, -8];  
P = [0, 0, -7, -9, -10, -15];  
K = 20;  
G = zpk(Z, P, K)
```

que tiene como salida

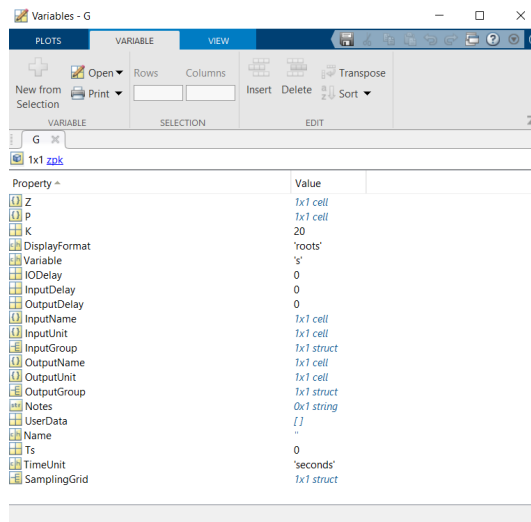
G =

$$\frac{20 (s+2) (s+3) (s+6) (s+8)}{s^2 (s+7) (s+9) (s+10) (s+15)}$$

Continuous-time zero/pole/gain model.

Es importante notar que los vectores de polos y ceros contienen las *ubicaciones* de los polos y ceros, respectivamente, sobre el plano complejo, por lo que es muy importante conservar su signo, es decir, cada polo asociado con un término $s + a$ se debe ingresar como $-a$. Además, en este ejemplo en concreto, debido al término s^2 en el numerador, fue necesario ingresar dos veces el 0 para representar el polo doble en el origen.

Las propiedades del objeto tipo zpk se muestran en la Figura 1.2



Property	Value
Z	1x1 cell
P	1x1 cell
K	20
DisplayFormat	'roots'
Variable	's'
IODelay	0
InputDelay	0
OutputDelay	0
InputName	1x1 cell
InputUnit	1x1 cell
InputGroup	1x1 struct
OutputName	1x1 cell
OutputUnit	1x1 cell
OutputGroup	1x1 struct
Notes	0x1 string
UserData	[]
Name	''
Ts	0
TimeUnit	'seconds'
SamplingGrid	1x1 struct

Figura 1.2: Propiedades de un objeto tipo zpk

Espacio de Estado

La representación en espacio de estado es otra forma de describir sistemas dinámicos, pero a diferencia de la representación en función de transferencia y cero–polo–ganancia, se enfoca en describir las ecuaciones diferenciales de primer orden que modelan el comportamiento del sistema. Esta representación es particularmente útil cuando se trabaja con sistemas de múltiples entradas y múltiples salidas.

En el contexto de un SLIT, la representación en espacio de estado se expresa mediante un conjunto de ecuaciones diferenciales ordinarias, conocidas como ecuaciones de estado, en la siguiente forma:

$$\begin{aligned}\dot{x}(t) &= Ax(t) + Bu(t) \\ y(t) &= Cx(t) + Du(t)\end{aligned}$$

donde $x(t)$ es el vector de estado que describe la condición interna del sistema, $\dot{x}(t)$ es la derivada temporal de $x(t)$, $u(t)$ es el vector de entrada, $y(t)$ es el vector de salida y A , B , C , y D son matrices constantes que caracterizan al sistema y definen cómo las variables de estado, entrada y salida se relacionan entre sí.

La representación en espacio de estado ofrece una visión completa del comportamiento dinámico del sistema y es particularmente útil para el análisis y diseño de sistemas complejos. Puede capturar información sobre la estructura interna y la interconexión de las variables del sistema, lo que facilita la comprensión de su comportamiento en diferentes condiciones. Además, esta representación es fundamental en el diseño de controladores y en el análisis de la estabilidad del sistema.

Si tenemos un sistema representado en espacio de estado con las siguientes matrices:

$$A = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & -3 & 3 \end{pmatrix} \quad B = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}$$

$$C = \begin{pmatrix} 1 & 0 & 0 \end{pmatrix} \quad D = 0$$

Ingresaremos el siguiente código para la creación del objeto `ss` que lo representa:

```
A = [0, 1, 0; 0, 0, 1; 1, -3, 3];
B = [1; 1; 1];
C = [1, 0, 0];
D = 0;
sys = ss(A,B,C,D)
```

que produce la salida

`sys =`

```
A =
      x1  x2  x3
x1      0   1   0
x2      0   0   1
x3      1  -3   3
```

```
B =
      u1
x1      1
x2      1
x3      1
```

```
C =
      x1  x2  x3
y1      1   0   0
```

$$D = \begin{bmatrix} & u1 \\ y1 & 0 \end{bmatrix}$$

Continuous-time state-space model.

El objeto ss creado tiene las propiedades mostradas en la Figura 1.3

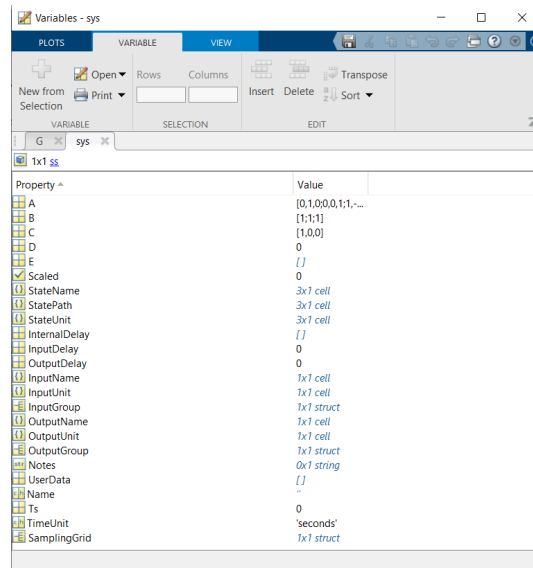


Figura 1.3: Propiedades de un objeto tipo ss

1.3 Análisis de Respuesta Transitoria y en Frecuencia

Una vez que hemos comprendido cómo ingresar sistemas en MATLAB, podemos avanzar hacia la exploración de las funciones especializadas en el análisis.

Respuesta en el tiempo

En lo que respecta al cálculo de la respuesta temporal de un SLIT, el *Control Systems Toolbox* dispone de funciones específicas, las cuales están detalladas en la Tabla 1.1.

Las funciones `step` e `impz` se aplican de la misma forma. Por ejemplo, dado el sistema

```
num = [1 17 99 223 140];
den = [1 32 363 2092 5052 4320];
G = tf(num,den);
```

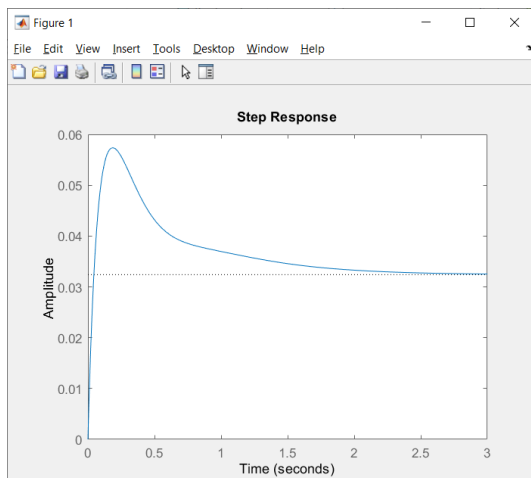
Tabla 1.1: Funciones para el análisis de respuesta temporal

Función	Descripción
step	Respuesta al escalón
impulse	Respuesta al impulso
lsim	Respuesta ante una entrada arbitraria
gensig	Genera señales de entrada

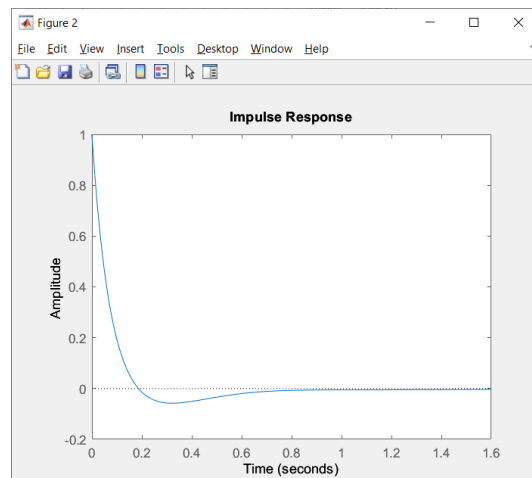
podemos obtener la respuesta al escalón y al impulso de la siguiente forma:

```
figure(1)
step(G)
figure(2)
impulse(G)
```

cuyo resultado son las dos figuras mostradas en la Figura 1.4.



(a) Respuesta al escalón (step)



(b) Respuesta al impulso (impulse)

Figura 1.4: Respuestas del sistema

A las funciones step e impulse se les puede variar el tiempo de simulación mediante un argumento adicional, como se muestra a continuación:

```
tsim = linspace(0,5,1500);
figure(1)
step(G,tsim)
figure(2)
impulse(G,tsim)
```

Además, permiten la simulación simultanea de varios sistemas, mediante el siguiente código:

```
num = [1 17 99 223 140];  
den = [1 32 363 2092 5052 4320];  
G = tf(num,den);  
G1 = tf([1 7],[1 10 216]);  
  
tsim = linspace(0,4,1500);  
figure(1)  
step(G,G1,tsim)  
figure(2)  
impulse(G,G1,tsim)
```

En los casos que necesitemos la respuesta temporal ante otras entradas podemos utilizar la función `lsim`. El llamado a esta función tiene la siguiente forma:

```
lsim(sys,u,t,xo)
```

donde `sys` es el sistema, `u` el vector de entrada, `t` el vector de tiempo y `xo` las condiciones iniciales (solo para sistemas en variables de estado). El vector `u` debe tener tantas filas como elementos tiene `t` y tantas columnas con entradas tenga el sistema.

Es posible usar la función `gensig` para generar el vector de entradas, `u`. Esta tiene la siguiente sintaxis:

```
[u,t] = gensig(tipo,tau,Tf,Ts);
```

donde `tipo` puede ser `'square'` para onda cuadrada, `'sin'` para onda senoidal y `'pulse'` para onda de pulsos. El argumento `tau` indica el periodo, `Tf` el tiempo final y `Ts` el tiempo de muestreo.

```
G = tf([1 0.5],2,[1 4 3]);  
[u1,t] = gensig('sin',1,10,0.01);  
u2 = [zeros(100,1);ones(901,1)];  
u = [0.1*u1,u2];  
lsim(G,u,t)
```

El resultado lo podemos ver en la Figura 1.5

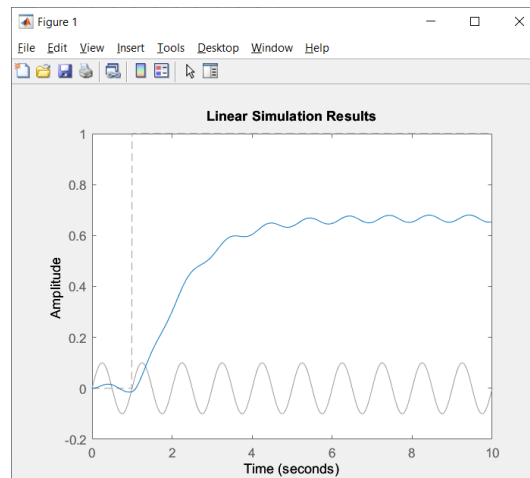


Figura 1.5: Respuesta del sistema usando lsim

Respuesta en la frecuencia

En la Tabla 1.3 se enumeran algunas de las funciones que dispone el *Control Systems Toolbox* para obtener la respuesta en frecuencia de un sistema.

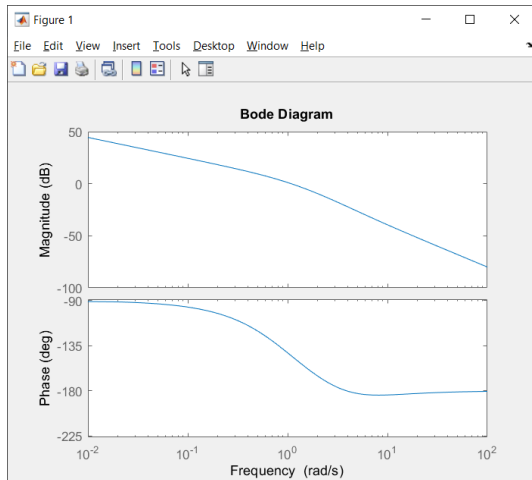
Tabla 1.3: Funciones para el análisis de respuesta frecuencial

Función	Descripción
bode	Diagrama de Bode
rlocus	Lugar de raíces
pzmap	Diagrama de polos y ceros
margin	Márgenes de fase y ganancia

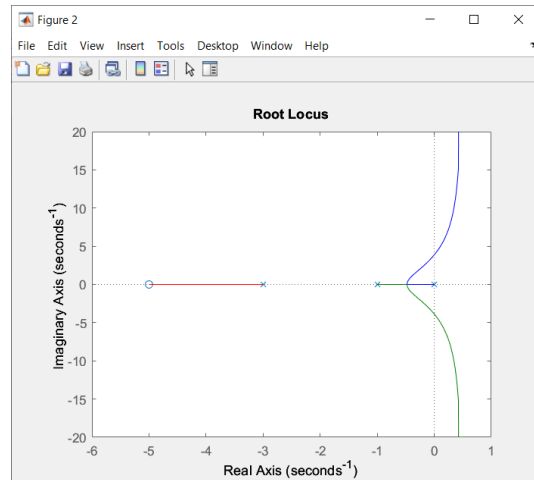
El uso de tales funciones se muestra a continuación:

```
G = zpk([-5],[0;-1;-3],1);
figure(1)
bode(G)
figure(2)
rlocus(G)
figure(3)
pzmap(G)
figure(4)
margin(G)
```

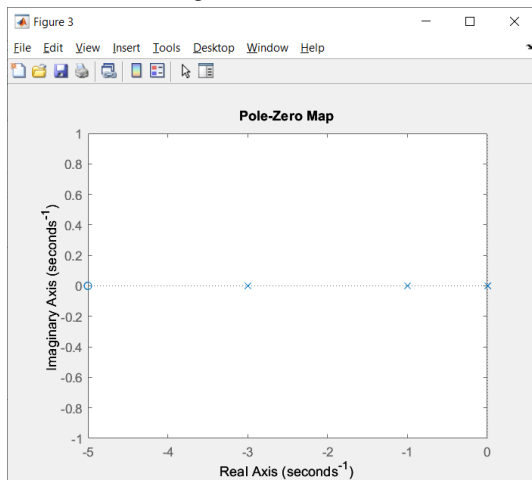
y sus resultados se pueden ver en la Figura 1.6



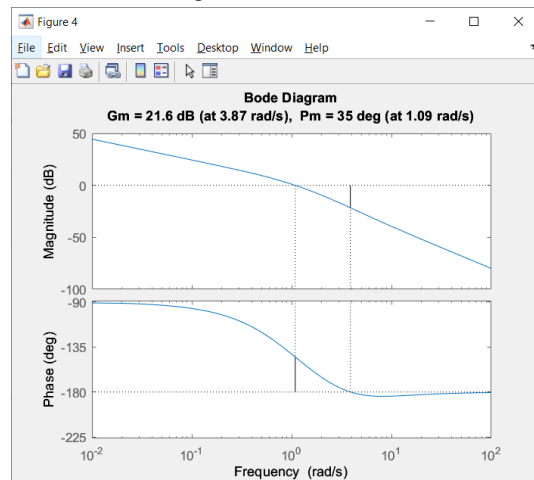
(a) Diagrama de Bode (bode)



(b) Lugar de raíces (rlocus)



(c) Diagrama de polos y ceros (step)



(d) Márgenes de fase y ganancia (impulse)

Figura 1.6: Gráficas para análisis de respuesta en frecuencia

Observaciones finales

El *Control Systems Toolbox* cuenta con la función `ltiview` que constituye un resumen de las funciones que hemos visto hasta aquí para obtener las respuestas temporal y frecuencial. al usarlo, se abre una ventana con menús que permiten importar a esta ventana cualquiera de los sistemas definidos en el *Workspace* y seleccionar el tipo de gráfico.

En la página web <https://la.mathworks.com/products/control.html> podrás expandir tus conocimientos acerca del *Control System Toolbox*, encontrar ejemplos de uso y explorar todas las funciones que vienen en él.

1.4 Trabajo práctico

Procedimiento

La Figura 1.7 muestra el sistema de suspensión de un automóvil con el que se trabajará en esta práctica. La variable de interés es la posición central del eje que sostiene la llanta, $x(t)$. La entrada al sistema es la superficie de la carretera $y(t)$, que puede variar de forma independiente. Suponemos que la masa de la carrocería no se mueve con respecto al eje del auto. Esto elimina el movimiento vertical de la carrocería, lo que simplifica el análisis. Puede que esta no sea una suposición realista, pero proporciona una primera aproximación del rendimiento de la suspensión.

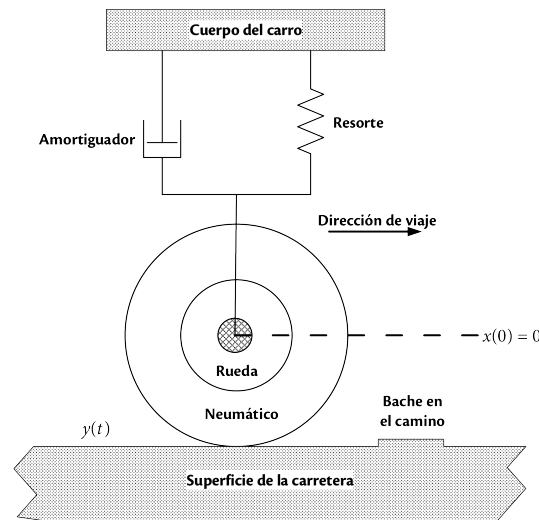


Figura 1.7: Sistema de suspensión utilizado en el análisis

El modelo matemático del sistema viene dado por la siguiente ecuación diferencial:

$$\frac{M}{K_1} \frac{d^2 x(t)}{dt^2} + \frac{B}{K_1} \frac{dx(t)}{dt} + \frac{K + K_1}{K_1} x(t) = y(t)$$

- Deberá crear un archivo en MATLAB para simular la respuesta temporal del sistema de suspensión ante diferentes entradas:
 - Escalón de amplitud 5 cm (debe buscar cómo se modifican los parámetros de la función step para simular escalones de amplitud diferente a la unidad)
 - Rampa saturada en 5 cm y pendiente 0.2 cm/ms
 - Función sigmoide saturada en 5 cm en 25 ms

Para todos los casos, use un vector de tiempo de 500 ms con pasos de 100 μ s y los parámetros del modelo mostrados en la Tabla 1.5.

Tabla 1.5: Parámetros del modelo

Parámetro	Valor
K	$1.751 \times 10^5 \text{ N/m}$
K_1	$3.5 \times 10^4 \text{ N/m}$
M	14 kg
B	536 N s/m

2. Repita las simulaciones dejando los valores de K , K_1 y M fijos y variando el valor de B tal que sea 400, 750, 1500 y 3000 N s/m. Grafique todas las respuestas en un mismo eje.
3. Obtenga *a)* el diagrama de márgenes de fase y ganancia y *b)* el mapa de polos y ceros del sistema original y de las variaciones del inciso anterior.

Evaluación

Envíe un documento con la respuesta a los siguientes elementos, junto con el archivo de MATLAB y las gráficas obtenidas.

1. Para todos los casos, establezca inicialmente la función de transferencia.
2. Describa los cambios en la respuesta temporal de la suspensión a medida que varía el amortiguamiento (tiempo de estabilización, máximo sobreimpulso, etc.).
3. Describa los cambios en los márgenes de fase y ganancia, relacionados con la variación del amortiguamiento.
4. Describa el efecto del amortiguamiento sobre la ubicación de los polos del sistema.