PRÁCTICA 0: Introducción a Matlab

Sistemas de Control Inteligente

Alba Calvo Herrero
María Sanz Espeja

Grupo A3 lunes
Grado Ingeniería Informática

Índice

Introducción

El presente trabajo se centra en una pequeña introducción a la aplicación de MATLAB. Analizaremos cómo las capacidades de programación y las herramientas gráficas de MATLAB pueden contribuir significativamente a la resolución de problemas complejos y al desarrollo de soluciones efectivas en este dominio. La práctica se divide en dos partes, la primera con 7 ejercicios sobre matrices y polinomios, y la segunda con 2 ejercicios donde se usa la herramientas simulink.

Los archivos matlab a su vez se dividen en dos partes. El archivo *practica0_parteI.m* contiene todos los ejercicios de la parte 1, y para ejecutar cada ejercicio se debe ejecutar la sección de ese ejercicio.

Parte I

Ejercicio 1

El primer ejercicio nos centramos en los vectores y matrices y sus operaciones básicas.

1. Cree la siguiente matriz A y el vector v:

$$A = \begin{bmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \\ 7 & 8 \end{bmatrix} \qquad v = \begin{bmatrix} 14 \\ 16 \\ 18 \\ 20 \end{bmatrix}$$

```
%% Ejercicio 1. Matrices y vectores
clear all
close all
% 1. Cree la siguiente matriz A y el vector v
A = [1,2;3,4;5,6;7,8];
v = [14;16;18;20];
```

Al ejecutarlo se obtiene lo deseado, ya que si se separa por ";" se obtiene un salto de fila y con "," insertar otra columna en la misma fila.

```
A =
     1     2
     3     4
     5     6
     7     8
v =
    14
    16
    18
    20
```

2. Obtenga y visualice una matriz B concatenando la matriz A y el vector v.

```
% 2.Obtenga y visualice una matriz B concatenando la matriz A y el vector
v.
B = [A,v]; % Concatena la matriz A y el vector v en columnas y crea una
nueva matriz B.
disp('B = ');
disp(B); % Muestra la matriz B en la consola.
```

En este apartado se obtiene la concatenación de A y v, encontrando primero los valores de la matriz A y en una nueva columna los valores del vector v. Si se quisiera los valores del vector en la primera columna, se deberá poner de la siguiente forma: [v, A];

```
B =
    1    2   14
    3    4   16
    5    6   18
    7    8   20
```

3. Obtenga y visualice un vector fila resultado de concatenar las filas de la matriz B.

```
% 3. Obtenga y visualice un vector fila resultado de concatenar las filas
de la matriz B
nfilas= size(B,1);
V=[];
for i=1:nfilas
    fila=B(i,:)
    V=[V,fila];
end
disp("V= ");
disp(V);
```

En el anterior fragmento de código se utiliza un bucle for para obtener el mismo resultado que con el siguiente fragmento de código al realizar la traspuesta de la matriz B:

```
%Forma rápida
V2=B'; %Traspuesta de B
V2=V2(:);
disp("V2= ");
disp(V2);
```

El resultado de anteriores ejecuciones es el siguiente:

```
V=
    1    2   14    3    4   16    5    6   18    7    8   20
```

4. Obtenga y visualice un vector columna resultado de concatenar las columnas de la matriz B.

```
% 4. Obtenga y visualice un vector columna resultado de concatenar las columnas de
la matriz B
C=B(:);
disp("C= ");
disp(C);
```

Tras la ejecución se obtiene lo esperado, ya que al utilizar “(:)” después de una matriz se produce un vector columna.

```
C=
     1
     3
     5
     7
     2
     4
     6
     8
    14
    16
    18
    20
```

Ejercicio 2

Continuamos en este ejercicio con operaciones de matrices:

Realice un script de Matlab que permita desarrollar una serie de operaciones con una matriz:

```
clear all
close all

% 1. script ha de generar una matriz, cuadrada y aleatoria de tamaño indicado por
el usuario. En la línea de comandos se ha de visualizar el mensaje: "Indique el
tamaño de la matriz".

n= input("Introduzca el tamaño de la matriz: ");
A = rand(n);
```

Se solicita un número entero para generar una matriz cuadrada de dicho número rellena con números aleatorios. Por ejemplo n=5.

```
% 2. A partir de la matriz construida, el script deberá calcular y
```

```
presentar por pantalla los siguientes datos:
% a) Matriz generada
disp("matriz A: ")
disp(A)
```

La matriz obtenida:

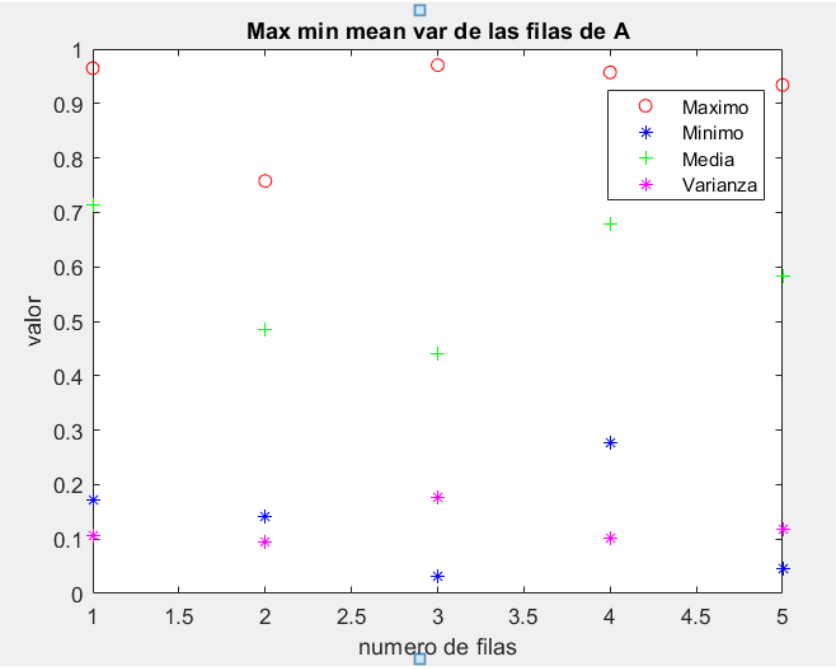
```
matriz A:
    0.9649    0.8003    0.9595    0.6787    0.1712
    0.1576    0.1419    0.6557    0.7577    0.7060
    0.9706    0.4218    0.0357    0.7431    0.0318
    0.9572    0.9157    0.8491    0.3922    0.2769
    0.4854    0.7922    0.9340    0.6555    0.0462

% b) Una segunda matriz formada por las columnas impares de la matriz
inicial
B=A(:,1:2:n);
disp('B= ');
disp(B);
```

En este caso, se crea la matriz B que cogerá los valores de las columnas, comenzando por la primera y saltando de dos en dos hasta llegar al valor n. De esta forma se cogerán los valores impares.

```
B=
    0.9649    0.9595    0.1712
    0.1576    0.6557    0.7060
    0.9706    0.0357    0.0318
    0.9572    0.8491    0.2769
    0.4854    0.9340    0.0462

% c) El valor de los elementos de la diagonal de la matriz generada
C = diag(A);
disp('C= ');
disp(C);

LA matriz C está compuesta por los valores de la diagonal de la matriz A:
C=
    0.9649
    0.1419
    0.0357
    0.3922
    0.0462
% d) Valor máximo, mínimo, medio y varianza de cada fila. Estos valores se
```

```
han de representar gráficamente, indicando en el eje de abscisas el número
de fila
vector_max=[];
vector_min=[];
vector_mean=[];
vector_var=[];
for i = 1:n
    fila=A(i,:);
    maximo=max(fila);
    minimo=min(fila);
    media=mean(fila);
    varianza=var(fila);
    vector_max=[vector_max, maximo];
    vector_min=[vector_min, minimo];
    vector_mean=[vector_mean, media];
    vector_var=[vector_var, varianza];
end

figure(1)
plot(vector_max, 'ro')
hold on;
plot(vector_min, 'b*')
plot(vector_mean, 'g+')
plot(vector_var, 'm*')
hold off;
xlabel('numero de filas')
ylabel('valor')
title('Max min mean var de las filas de A')
legend({'Maximo', 'Minimo', 'Media', 'Varianza'})
```

Tras realizar los cálculos necesarios para obtener el valor máximo, mínimo, medio y varianza, se representa en una gráfica de la forma más evidente con colores y diferentes formas en una gráfica.



Ejercicio 3

Programe un script en Matlab que permita realizar una serie de operaciones con dos matrices (A y B) que se introducirán por teclado. Para ello:

1. Solicite al usuario las dimensiones de las matrices en formato [filas cols], (si se introduce un único número, la matriz será cuadrada).

```
input_str = input('Ingrese las dimensiones de la matriz en formato [filas cols]: ',
's');
Dimensiones = str2num(input_str);
if length(Dimensiones) == 1
Dimensiones = [Dimensiones Dimensiones];
end
```

2. Genere dos matrices (A y B) de las dimensiones elegidas. Para rellenar las matrices, escriba una función en Matlab (en un fichero diferente) que reciba como parámetro las dimensiones deseadas [filas cols], y devuelva la matriz rellena.

3. La función debe pedir datos al usuario para cada posición de la matriz. En caso de que el usuario escriba 'r', la matriz se rellenará de valores aleatorios

```
% Archivo IntroducirMatriz.m
function Matriz = IntroducirMatriz(Dimensiones)
```

```
        filas = Dimensiones(1);
        cols = Dimensiones(2);
        Matriz = zeros(filas, cols);
        for i = 1:filas
                for j = 1:cols
                        fprintf('Ingrese el valor para la posición (%d, %d): ', i, j);
                        entrada = input('', 's');
                        if strcmpi(entrada, 'r')
                                % Rellenar con valores aleatorios
                                Matriz(i, j) = rand();
                        else
                                Matriz(i, j) = str2num(entrada);
                        end
                end
        end
end
```

4. Calcule y muestre por pantalla:

- Las matrices generadas A y B

```
A = IntroducirMatriz(Dimensiones);
B = IntroducirMatriz(Dimensiones);
disp('A= ')
disp(A)
disp('B= ')
disp(B)
```

- La transpuesta e inversa de cada una de las matrices

```
% Calcular la transpuesta de A y B
if isequal(size(A), size(B))
        A_transpuesta = A';
        B_transpuesta = B';
else
        fprintf('No se puede calcular la transpuesta porque las dimensiones de A
        y B son diferentes.\n');
end
% Calcular la inversa de A y B
if isequal(size(A), size(B)) && size(A, 1) == size(A, 2)
        A_inversa = inv(A);
        B_inversa = inv(B);
else
        fprintf('No se puede calcular la inversa porque una de las matrices no
        es invertible (determinante igual a cero).\n');
end
```

• El valor del determinante y el rango de cada una de las matrices

```
% Calcular el determinante de A y B
if isequal(size(A), size(B)) && size(A, 1) == size(A, 2)
	determinante_A = det(A);
	determinante_B = det(B);
else
	fprintf('No se puede calcular el determinante porque las dimensiones de
	A y B son diferentes.\n');
end
% Calcular el rango de A y B

rango_A = rank(A);

rango_B = rank(B);
```

• El valor del producto de A y B (matricial y elemento a elemento)

```
% Calcular el producto matricial de A y B
if size(A, 2) == size(B, 1)
	producto_matricial = A * B;
else
	fprintf('No se puede calcular el producto matricial porque las
	dimensiones de A y B no son compatibles.\n');
end
% Calcular el producto elemento a elemento de A y B
if isequal(size(A), size(B))
	producto_elemento_a_elemento = A .* B;
else
	fprintf('No se puede calcular el producto elemento a elemento porque las
	dimensiones de A y B son diferentes.\n');
end
```

• Un vector fila obtenido concatenando la primera fila de cada una de las matrices

```
% Calcular el producto elemento a elemento de A y B
if isequal(size(A), size(B))
	producto_elemento_a_elemento = A .* B;
else
	fprintf('No se puede calcular el producto elemento a elemento porque las
	dimensiones de A y B son diferentes.\n');
end
```

• Un vector fila obtenido concatenando la primera fila de cada una de las matrices

```
% Verificar si es posible concatenar las filas y columnas
if isequal(size(A, 1), size(B, 1))
	% Concatenar filas
	vector_fila = [A(1, :), B(1, :)];
else
```

```
        fprintf('No se pueden concatenar las filas porque el número de filas en
        A y B es diferente.\n');
end
```

• Un vector columna obtenido concatenando la primera columna de cada una de las matrices En caso de que no sea posible realizar alguno de los cálculos solicitados, indíquelo por pantalla.

```
if isequal(size(A, 2), size(B, 2))
        % Concatenar columnas
        vector_columna = [A(:, 1); B(:, 1)];
else
        fprintf('No se pueden concatenar las columnas porque el número de
        columnas en A y B es diferente.\n');
end
```

En el código, se han tenido en cuenta casos en los que no es posible realizar ciertos cálculos, como cuando las dimensiones de las matrices no son compatibles para la multiplicación matricial, cuando las matrices no son cuadradas y no se puede calcular la inversa o el determinante, o cuando las dimensiones de las filas y columnas no coinciden para la concatenación.

Para una matriz 2x2:

Ingrese las dimensiones de la matriz 1 en formato [filas cols]: 2 2
Ingrese las dimensiones de la matriz 2 en formato [filas cols]: 2 2
Ingrese el valor para la posición (1, 1): r
Ingrese el valor para la posición (1, 2): r
Ingrese el valor para la posición (2, 1): r
Ingrese el valor para la posición (2, 2): r
Ingrese el valor para la posición (1, 1): r
Ingrese el valor para la posición (1, 2): r
Ingrese el valor para la posición (2, 1): r
Ingrese el valor para la posición (2, 2): r
A=

0.5797	0.5499
0.1450	0.8530

B=

0.6221	0.3510
0.5132	0.4018

Transpuesta de A:

0.5797	0.1450
0.5499	0.8530

Transpuesta de B:

0.6221	0.5132

0.3510 0.4018

Inversa de A:
2.0565 -1.3256
-0.3495 1.3975

Inversa de B:
5.7549 -5.0265
-7.3510 8.9093

Determinante de A:
0.4148

Determinante de B:
0.0698

Rango de A:
2

Rango de B:
2

Producto matricial de A y B:
0.6428 0.4244
0.5280 0.3936

Producto elemento a elemento de A y B:
0.3606 0.1930
0.0744 0.3428

Vector fila concatenado:
0.5797 0.5499 0.6221 0.3510

Vector columna concatenado:
0.5797
0.1450
0.6221
0.5132

Para una matriz 2x3 y 2x2:
Ingrese las dimensiones de la matriz 1 en formato [filas cols]: 2 3
Ingrese las dimensiones de la matriz 2 en formato [filas cols]: 2 2

Ingrese el valor para la posición (1, 1): r
Ingrese el valor para la posición (1, 2): r
Ingrese el valor para la posición (1, 3): r
Ingrese el valor para la posición (2, 1): r
Ingrese el valor para la posición (2, 2): r
Ingrese el valor para la posición (2, 3): r
Ingrese el valor para la posición (1, 1): r
Ingrese el valor para la posición (1, 2): r
Ingrese el valor para la posición (2, 1): r
Ingrese el valor para la posición (2, 2): r
A=
  0.3998 0.2599 0.8001
  0.4314 0.9106 0.1818

B=
  0.2638 0.1455
  0.1361 0.8693

No se puede calcular la transpuesta porque las dimensiones de A y B son diferentes.
No se puede calcular la inversa porque una de las matrices no es invertible (determinante igual a cero).
No se puede calcular el determinante porque las dimensiones de A y B son diferentes.
No se puede calcular el producto matricial porque las dimensiones de A y B no son compatibles.
No se puede calcular el producto elemento a elemento porque las dimensiones de A y B son diferentes.
No se pueden concatenar las columnas porque el número de columnas en A y B es diferente.
Rango de A:
  2

Rango de B:
  2

Vector fila concatenado:
  0.3998 0.2599 0.8001 0.2638 0.1455

Vemos como nos sale un aviso de que no se puede calcular la multiplicación, la inversa y el determinante entre otros por las dimensiones de las matrices.

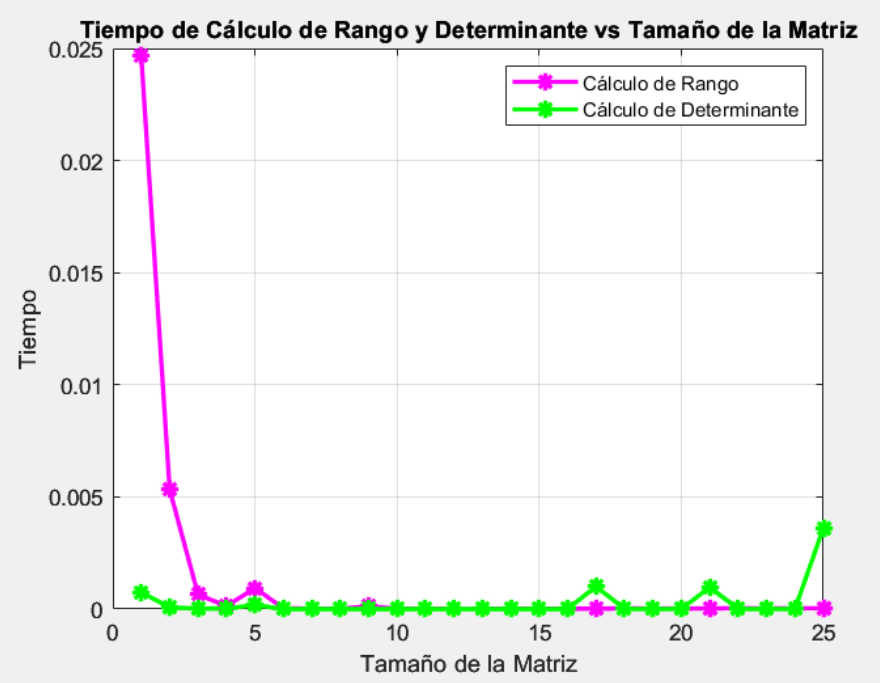
Ejercicio 4

Realice un script en Matlab que permita obtener y representar el tiempo consumido para el cálculo del rango y el determinante de una matriz en función de su tamaño (entre 1x1 y 25x25).

```
% Inicializar variables
tam = 1:25; % Tamaños de la matriz (de 1x1 a 25x25)
tiempoRango = zeros(size(tam)); % Almacenar tiempos de cálculo de rango
```

```
tiempoDeterminante = zeros(size(tam)); % Almacenar tiempos de cálculo de
determinante
for i = 1:length(tam)
tamMatriz = tam(i);
% Generar una matriz aleatoria del tamaño especificado
A = rand(tamMatriz);
% Medir el tiempo para el cálculo del rango
tic; %inicio temporizador
rangoA = rank(A);
tiempoRango(i) = toc; %fin temporizador
% Medir el tiempo para el cálculo del determinante
tic;
determinanteA = det(A);
tiempoDeterminante(i) = toc;
end
% Crear una gráfica
figure;
plot(tam, tiempoRango, '-m*','LineWidth', 2, 'MarkerSize', 8);
hold on;
plot(tam, tiempoDeterminante, '-g*','LineWidth', 2, 'MarkerSize', 8);
hold off;
% Personalizar la gráfica
xlabel('Tamaño de la Matriz');
ylabel('Tiempo');
title('Tiempo de Cálculo de Rango y Determinante vs Tamaño de la Matriz');
legend('Cálculo de Rango', 'Cálculo de Determinante');
grid on;
```

Tras la ejecución del código, obtenemos la siguiente gráfica, representando el rango en color rosa y el determinante en color verde.
Mencionar que para el cálculo del tiempo se utiliza la función "**tic**" para iniciar el temporizador y "**toc**" para finalizarlo una vez se ha calculado lo deseado.



Ejercicio 5

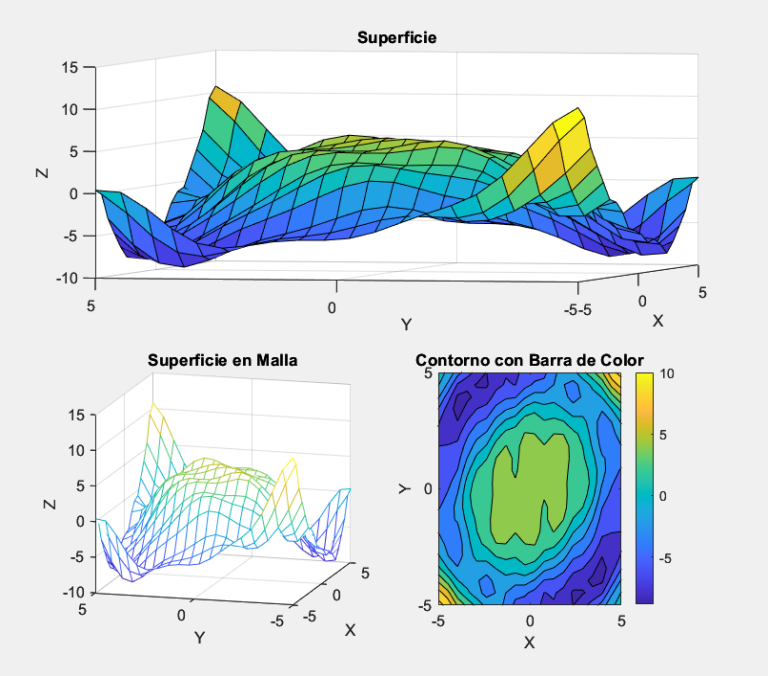
Realice un script en Matlab que dibuje sobre el área −5 ≤ x, y ≤ 5 la superficie, la superficie en forma de malla y el contorno de la función:

$$z = y * sin\left(pi * \frac{x}{10}\right) + 5 * \cos\left((x^2 + y^2)/8\right) + cos(x + y)cos(3x - y).$$

```
vector_x = linspace(-5,5,20);
vector_y = linspace(-5,5,20);
[X,Y]=meshgrid(vector_x, vector_y);
Z= Y.*sin(pi*X/10) + 5*cos((X.^2+Y.^2)/8)+cos(X+Y).*cos(3*X-Y);
figure;
%Grafica en la parte superior centrada
subplot(2, 2, [1, 2])
surf(X,Y,Z)
title('Superficie')
xlabel('X')
ylabel('Y')
zlabel('Z')
%Grafica en la parte inferior izquierda
subplot(2, 2, 3)
```

```
mesh(X,Y,Z)
title('Superficie en Malla')
xlabel('X')
ylabel('Y')
zlabel('Z')
%Grafica en la parte inferior derecha
subplot(2, 2, 4)
contourf(X,Y,Z)
colorbar
title('Contorno con Barra de Color')
xlabel('X')
ylabel('Y')
zlabel('Z')
```

En este ejercicio se practica las representaciones de gráficas 3D que se obtiene todo en un solo cuadro:



Ejercicio 6

En este ejercicio se trabajan los sistemas lineales con hasta cuatro incógnitas. Primero se representan los sistemas 1 y 2 siendo A la parte de la izquierda de la igualdad y b la parte de la derecha, respectivamente.

Después se resuelven los apartados para el primer sistema y a continuación para el segundo:

```
%% Ejercicio 6. Sistemas lineales
```

```
clear all
close all
% Dados los siguientes sistemas lineales de 10 ecuaciones con 4 incógnitas (x1, x2, x3, x4)
% 1. Exprese el sistema de forma matricial en Matlab. Para ello, cree las matrices A y b.
A1 = [0,2,10,7;
2,7,7,1;
1,9,0,5;
4,0,0,6;
2,8,4,1;
10,5,0,3;
2,6,4,0;
1,1,9,3;
6,4,8,2;
0,3,0,9];
b1 = [90;59;15;10;80;17;93;51;41;76];

A2 = [0.110,0,1,0;
0,3.260,0,1;
0.425,0,1,0;
0,3.574,0,1;
0.739,0,1,0;
0,3.888,0,1;
1.054,0,1,0;
0,4.202,0,1;
1.368,0,1,0;
0,4.516,0,1];
b2 = [317;237;319;239;321;241;323;243;325;245];
% 2. Escriba un script en que permita:
% SISTEMA 1:
% a) Obtener el número de condición de la matriz A respecto a la inversión (numero que me dice
si la matriz A es invertible, usar la funcion rcond(A)
condicion_A1 = cond(A1);
fprintf('Número de condición de A1: %f\n', condicion_A1);
% b) Resolver el sistema de ecuaciones para obtener la matriz x = [x1, x2, x3, x4]'.
x_1 = linsolve(A1, b1); %encuentra la solución para el vector x que satisface la ecuación A1 *
x_1 = b1
% c) Añadir ruido a la matriz b, sumándole un vector aleatorio de media 0 y desviación 1, y
resuelva el sistema de ecuaciones resultante.
n1 = length(b1);
ruido = randn(n1, 1); % Vector aleatorio con media 0 y desviación estándar 1
b1_con_ruido = b1 + ruido;
x1_con_ruido = linsolve(A1, b1_con_ruido); % Resuelve el sistema

% d) Mostrar el resultado (con y sin ruido añadido) por pantalla.
fprintf('\nSolución del sistema de ecuaciones A1 (sin ruido):\n');
fprintf('x1 = %f\n', x_1(1));
fprintf('x2 = %f\n', x_1(2));
fprintf('x3 = %f\n', x_1(3));
fprintf('x4 = %f\n', x_1(4));

fprintf('\nSolución del sistema de ecuaciones A1 (con ruido):\n');
fprintf('x1 = %f\n', x1_con_ruido(1));
fprintf('x2 = %f\n', x1_con_ruido(2));
fprintf('x3 = %f\n', x1_con_ruido(3));
fprintf('x4 = %f\n', x1_con_ruido(4));

% SISTEMA 2:
% a) Obtener el número de condición de la matriz A respecto a la inversión (numero que me dice
si la matriz A es invertible, usar la funcion rcond(A)
condicion_A2 = cond(A2);
fprintf('Número de condición de A2: %f\n', condicion_A2);
```

```
% b) Resolver el sistema de ecuaciones para obtener la matriz x = [x1, x2, x3, x4]’.
x_2 = linsolve(A2, b2); %encuentra la solución para el vector x que satisface la ecuación A1 *
x_1 = b1

% c) Añadir ruido a la matriz b, sumándole un vector aleatorio de media 0 y desviación 1, y
resuelva el sistema de ecuaciones resultante.
n2 = length(b2);
ruido = randn(n2, 1); % Vector aleatorio con media 0 y desviación estándar 1
b2_con_ruido = b2 + ruido;
x2_con_ruido = linsolve(A2, b2_con_ruido); % Resuelve el sistema

% d) Mostrar el resultado (con y sin ruido añadido) por pantalla.
fprintf('\nSolución del sistema de ecuaciones A2 (sin ruido):\n');
fprintf('x1 = %f\n', x_2(1));
fprintf('x2 = %f\n', x_2(2));
fprintf('x3 = %f\n', x_2(3));
fprintf('x4 = %f\n', x_2(4));

fprintf('\nSolución del sistema de ecuaciones A2 (con ruido):\n');
fprintf('x1 = %f\n', x2_con_ruido(1));
fprintf('x2 = %f\n', x2_con_ruido(2));
fprintf('x3 = %f\n', x2_con_ruido(3));
fprintf('x4 = %f\n', x2_con_ruido(4));
```

Los resultados son los siguientes.

En la primera ejecución:

```
Número de condición de A1: 2.725749

Solución del sistema de ecuaciones A1 (sin ruido):
x1 = -2.257095
x2 = 4.933635
x3 = 5.298567
x4 = 3.564928

Solución del sistema de ecuaciones A1 (con ruido):
x1 = -2.134183
x2 = 4.918854
x3 = 5.264886
x4 = 3.609169
```

```
Número de condición de A2: 36.710191

Solución del sistema de ecuaciones A2 (sin ruido):
x1 = 6.359299
x2 = 6.369427
x3 = 316.299207
x4 = 216.235669

Solución del sistema de ecuaciones A2 (con ruido):
x1 = 7.108931
x2 = 6.027286
x3 = 315.459225
x4 = 216.968986
```

2ª ejecución:

```
Solución del sistema de ecuaciones A1 (sin ruido):
x1 = -2.257095
x2 = 4.933635
x3 = 5.298567
x4 = 3.564928

Solución del sistema de ecuaciones A1 (con ruido):
x1 = -2.153140
x2 = 4.871520
x3 = 5.341017
x4 = 3.591679
Número de condición de A2: 36.710191

Solución del sistema de ecuaciones A2 (sin ruido):
x1 = 6.359299
x2 = 6.369427
x3 = 316.299207
x4 = 216.235669

Solución del sistema de ecuaciones A2 (con ruido):
x1 = 6.554932
x2 = 3.720503
x3 = 315.831784
x4 = 226.364684
```

Las conclusiones de los resultados de ambas iteraciones son las siguientes:

Primera ejecución:

- Para A1 (matriz con número de condición 2.725749), las soluciones con y sin ruido son bastante similares, lo que sugiere que el ruido no tiene un impacto significativo en las soluciones.
- Para A2 (matriz con número de condición 36.710191), las soluciones con ruido son notablemente diferentes de las soluciones sin ruido. El alto número de condición de A2 hace que sea muy sensible al ruido, lo que resulta en cambios significativos en las soluciones.

Segunda ejecución:

- En esta iteración, las conclusiones para A1 siguen siendo consistentes: las soluciones con y sin ruido son bastante similares, lo que indica que el ruido no afecta mucho.
- Para A2, nuevamente, el ruido tiene un impacto significativo en las soluciones. Las soluciones con ruido son diferentes de las soluciones sin ruido.

Por tanto, concluimos que la matriz A1 parece ser bastante estable en términos de soluciones, independientemente de la presencia de ruido.

La matriz A2, debido a su alto número de condiciones, es extremadamente sensible al ruido. Pequeñas variaciones en los datos de entrada pueden resultar en cambios sustanciales en las soluciones.

Es importante considerar el número de condición de una matriz y su sensibilidad al ruido al resolver sistemas lineales. Cuando una matriz tiene un número de condición alto, se deben tomar precauciones adicionales para evitar errores significativos, como técnicas de regularización.

Ejercicio 7

Realice una función de Matlab que permita obtener las raíces de un producto de polinomios y las clasifique en reales y complejas. Para ello ha de realizar los siguientes pasos:

```
% 1. Las entradas y salidas de la función son las que se especifican, según
la siguiente sintaxis:
        % [solución, reales, complejas]=raices(poli_1, poli_2)
   % Ejemplo: [[-1+i, -1-i, -3], 1, 2]=raíces([1 2 2], [1 3])
% a) Recoge los arrays con los que se crean los polinomios.
% b) Solicita si la solución se aplica a uno de los polinomios o al
producto: poli_1, poli_2, prod_poli.
% c) Devuelve las raíces del polinomio indicado y su clasificación (nº
raíces reales y nº raíces complejas.
% d) Representa en el plano complejo la ubicación de las raíces obtenidas.

% x^4 + 3x^3 + 2x^2 + x +3
poli_1 = [1,3,2,1,3];
```

```
% -5x^4+4x^3-2x^2+6x-1
poli_2 = [-5,4,-2,6,-1]

raices(poli_1,poli_2);
```

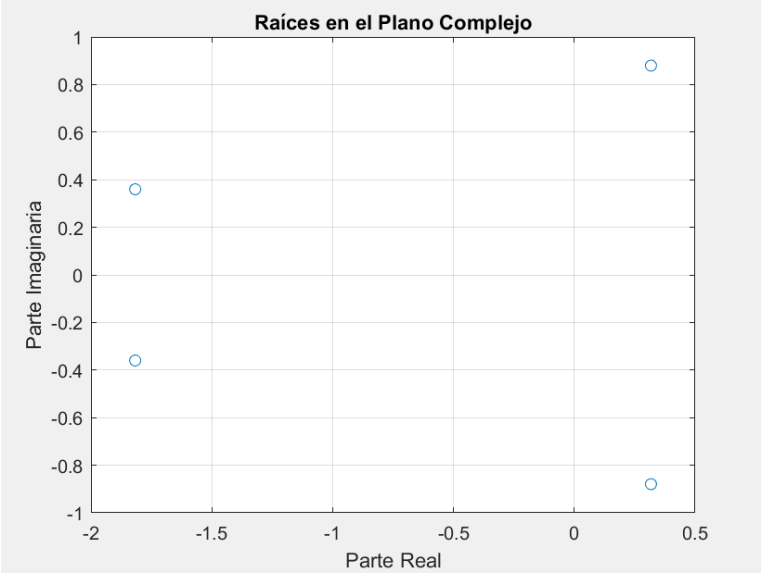
Para ello se crea la siguiente función MATLAB llamada raices.m:

```
function [solucion, reales, complejas] = raices(poli_1, poli_2)
    % Recoge los arrays con los que se crean los polinomios.
    p1 = poly2sym(poli_1);
    p2 = poly2sym(poli_2);

    % Solicita si la solución se aplica a uno de los polinomios o al
producto
    fprintf('Elija el polinomio para encontrar las raíces:\n');
    fprintf('1. Polinomio 1\n');
    fprintf('2. Polinomio 2\n');
    fprintf('3. Producto de Polinomios\n');
    choice = input('Ingrese el número correspondiente al polinomio: ');

    switch choice
        case 1
            % Encuentra las raíces del Polinomio 1
            solucion = roots(poli_1);
        case 2
            % Encuentra las raíces del Polinomio 2
            solucion = roots(poli_2);
        case 3
            % Encuentra las raíces del producto de los polinomios
            prod_poli = p1 * p2;
            solucion = roots(sym2poly(prod_poli));
        otherwise
            error('Opción no válida.');
    end

    % Clasifica las raíces en reales y complejas
    reales = sum(imag(solucion) == 0);
    complejas = sum(imag(solucion) ~= 0);
    fprintf('Soluciones:\n')
    disp(solucion);
    % Representa en el plano complejo la ubicación de las raíces obtenidas
    figure;
```

```
    plot(real(solucion), imag(solucion), 'o');
    title('Raíces en el Plano Complejo');
    xlabel('Parte Real');
    ylabel('Parte Imaginaria');
    grid on;
end
```

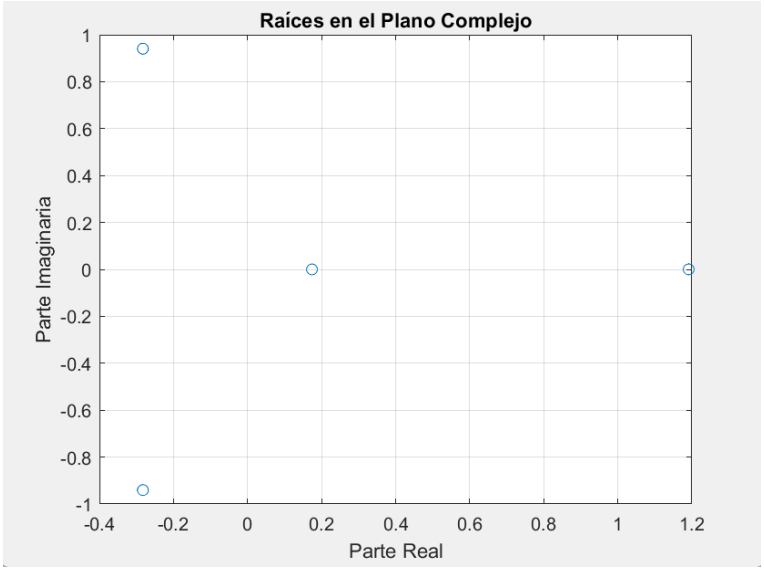
El usuario puede elegir cuál de los polinomios o su producto desea analizar. El resultado mostrará las raíces y su clasificación (número de raíces reales y complejas) junto con una representación gráfica en el plano complejo de las raíces.

```
Soluciones polinomio 1:
```
Elija el polinomio para encontrar las raíces:
1. Polinomio 1
2. Polinomio 2
3. Producto de Polinomios
Ingrese el número correspondiente al polinomio: 1
Soluciones:
 -1.8172 + 0.3597i
 -1.8172 - 0.3597i
  0.3172 + 0.8795i
  0.3172 - 0.8795i

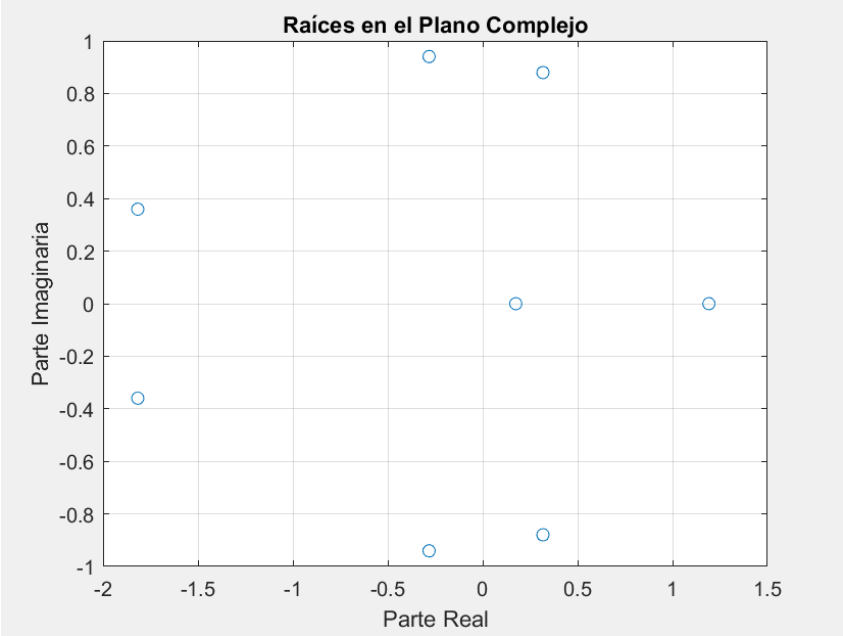


Soluciones polinomio 2:
Ingrese el número correspondiente al polinomio: 2
Soluciones:
  1.1914 + 0.0000i

-0.2827 + 0.9406i
-0.2827 - 0.9406i
0.1740 + 0.0000i



Soluciones producto de polinomios:
Soluciones:
-1.8172 + 0.3597i
-1.8172 - 0.3597i
1.1914 + 0.0000i
-0.2827 + 0.9406i
-0.2827 - 0.9406i
0.3172 + 0.8795i
0.3172 - 0.8795i
0.1740 + 0.0000i

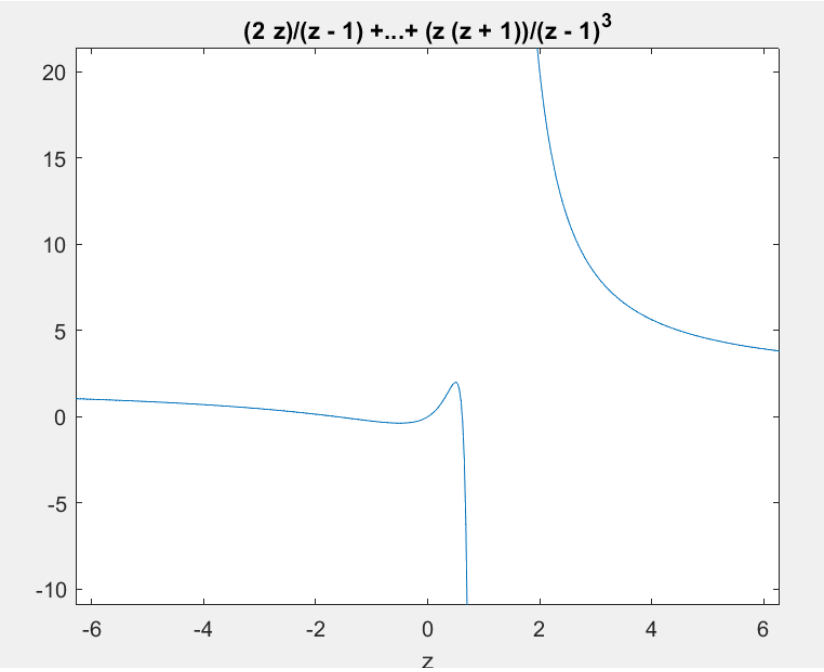


Parte II

Ejercicio 1

```
% 1. Obtenga la transformada z de la siguiente función
syms k a z
% Definimos la funcion f(k)
f_k = 2 + 5*k + k^2;

% Hacemos la transformada de f(k) que será F(z)
F_z = ztrans(f_k);

% Represente gráficamente las señales original y transformada.
% Como solo tiene 1 variable usamos ezplot:
ezplot(f_k);
ezplot(F_z);
```

El resultado de este código será un gráfico que muestra ambas señales en una misma figura. Verás la representación gráfica de la función original f(k) y su transformada Z F(z) en función de sus respectivas variables. Como solo tenemos una variable usamos ezplot para representarlo.

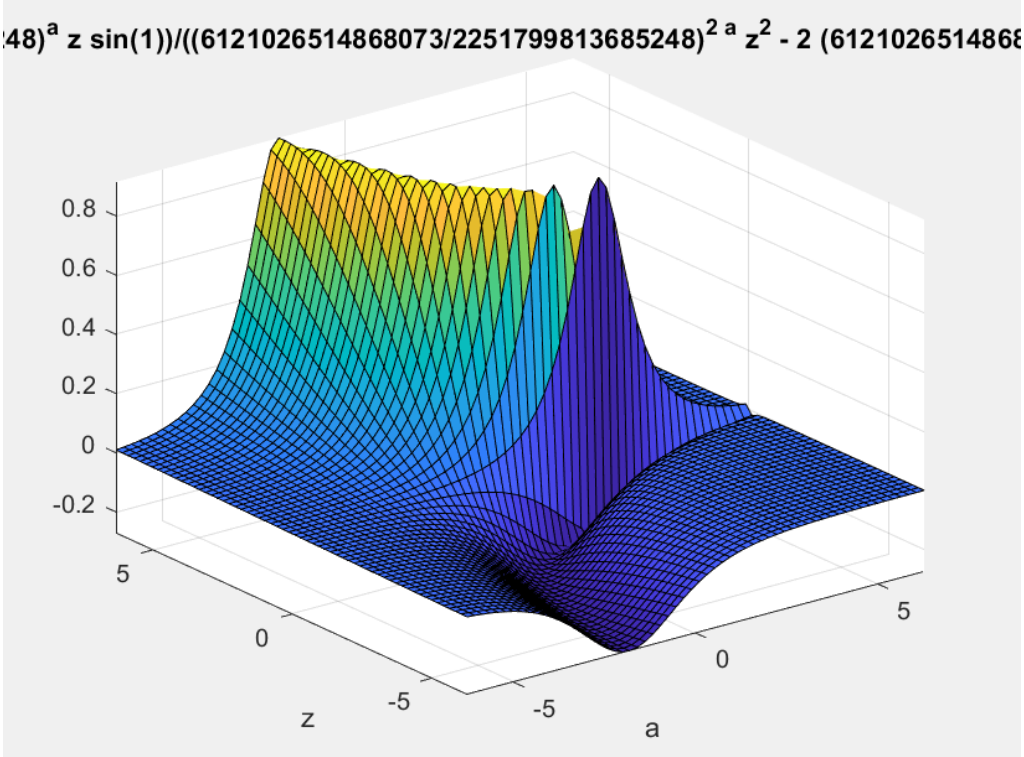



```
% 2. Obtenga la transformada z de la siguiente función

% Definimos la funcion g(k)
g_k = sin(k)*exp(1)^((-a)*k);

% Calculamos la transformada de g(k)
F_z_2 = ztrans(g_k);

% Como tiene dos variables usamos ezsurf para representarlo
ezsurf(g_k)
ezsurf(F_z_2)
```

Para representar gráficamente ambas señales, dado que tienes dos variables k y z), se utiliza la función ezsurf. El primer ezsurf muestra la señal original g(k), que es una función sinusoidal exponencial. El segundo ezsurf muestra la transformada F(z)_2, que es una función en z después de la transformación.

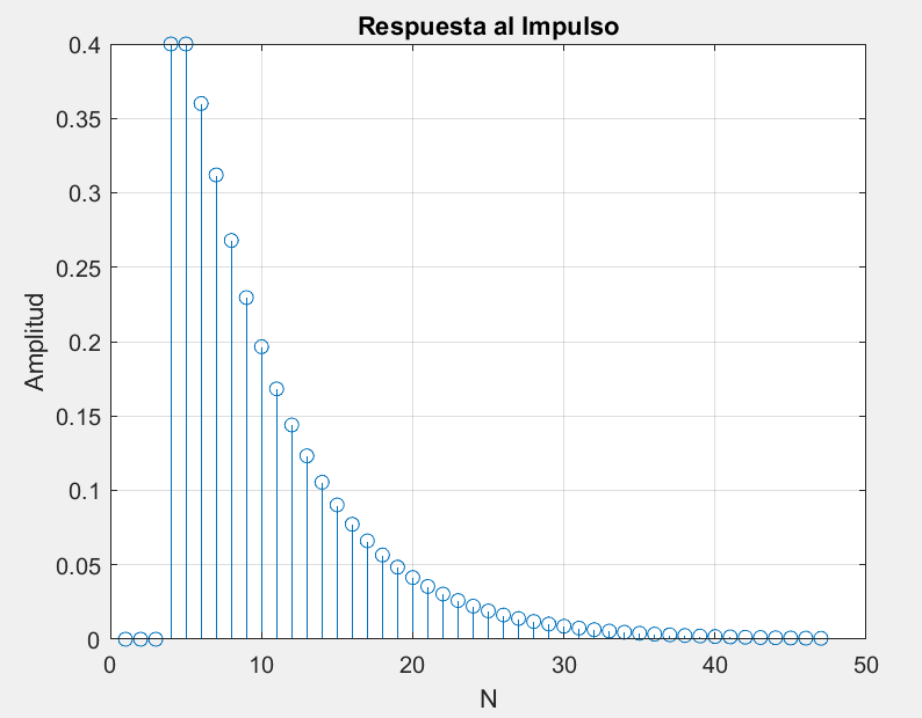
El resultado de este código será una figura 3D que muestra ambas señales en un espacio tridimensional. Verás la representación gráfica de la función original \(g(k) \) y su transformada Z \(F(z)_2 \) en función de las variables k y z.

Esta representación gráfica permite visualizar cómo se relacionan la señal original y su transformada Z en el dominio del tiempo discreto y el dominio Z, respectivamente, en un espacio tridimensional.



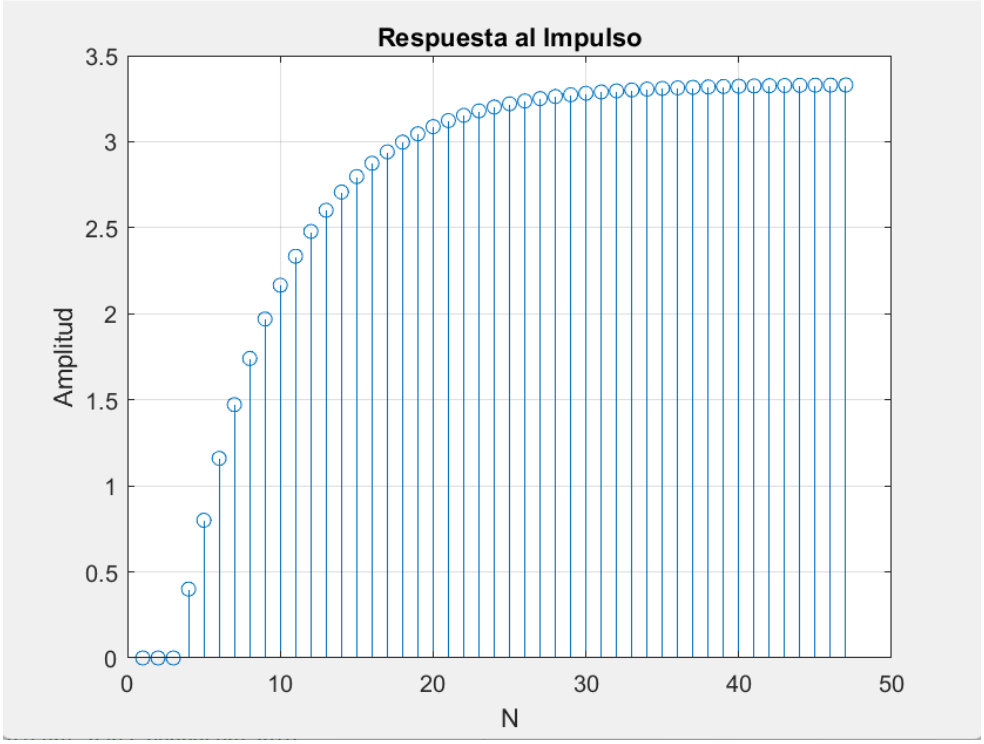

```
% 3. Dada la siguiente función de transferencia discreta:
T_z =(0.4*z^2)/(z^3-z^2+0.1*z+0.02);

% Obtenga y represente la respuesta al impulso del sistema.
num = [0.4]; % Coeficientes del numerador
den = [1 (-1) 0.1 0.02]; % Coeficientes del denominador
ts = 1;
transfer_t = tf(num,den,ts);
impulse_response = impulse(transfer_t);

% Representar la respuesta al impulso
stem(impulse_response);
xlabel('N'); % N representa el número de muestras
ylabel('Amplitud');
title('Respuesta al Impulso');
grid on;
```

El código calcula y representa gráficamente la respuesta del sistema ante un impulso unitario. La representación gráfica muestra cómo el sistema responde a un impulso en función del número de muestras, lo que es útil para comprender el comportamiento del sistema en el dominio del tiempo discreto.



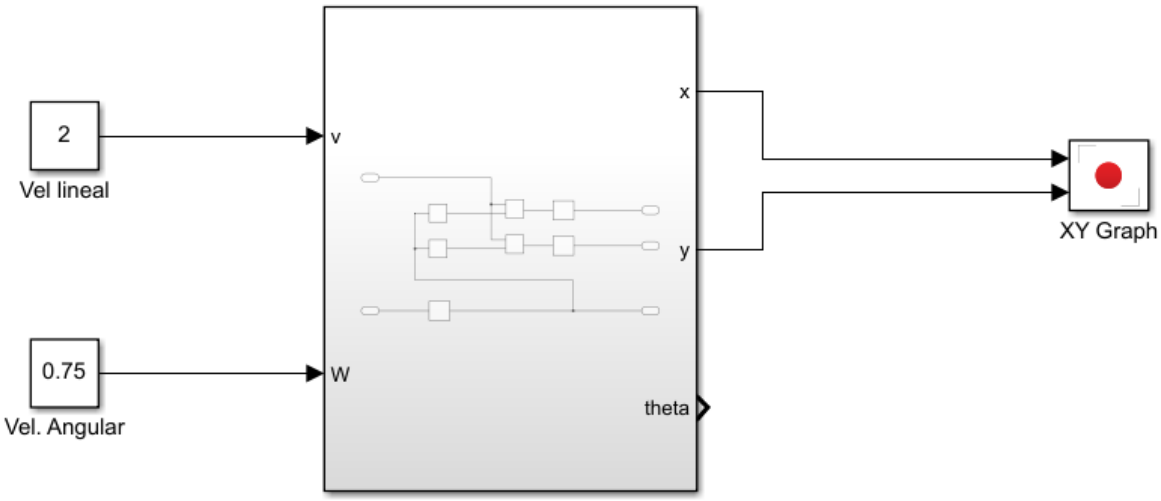

```
% Obtenga y represente la respuesta del sistema ante una entrada escalón.
step = step(transfer_t);
stem(step);
xlabel('N'); % N representa el número de muestras
ylabel('Amplitud');
title('Respuesta al Impulso');
grid on;
```

Representa gráficamente la respuesta del sistema ante una entrada de escalón. El gráfico muestra cómo el sistema responde a un cambio abrupto en la entrada (el escalón) en función del tiempo discreto, lo que es útil para comprender el comportamiento del sistema ante cambios bruscos en la señal de entrada.

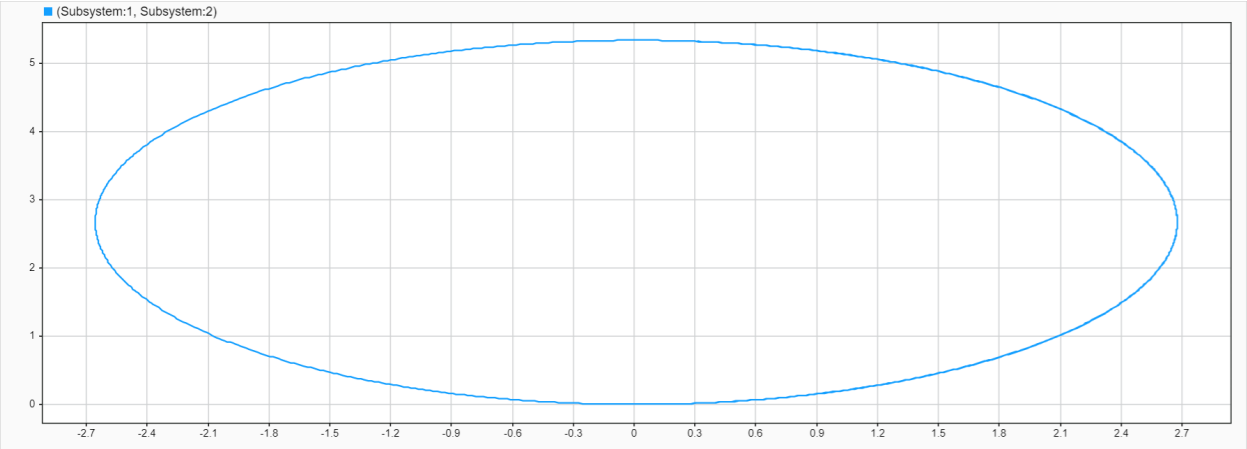


Ejercicio 2

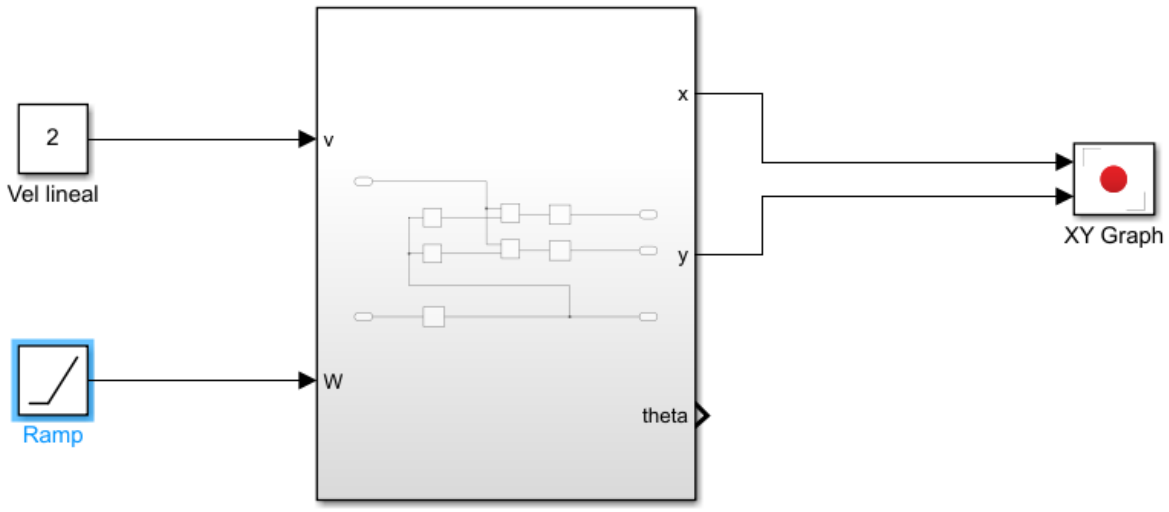
Ejecutamos el sistema con velocidad lineal y angular constante:

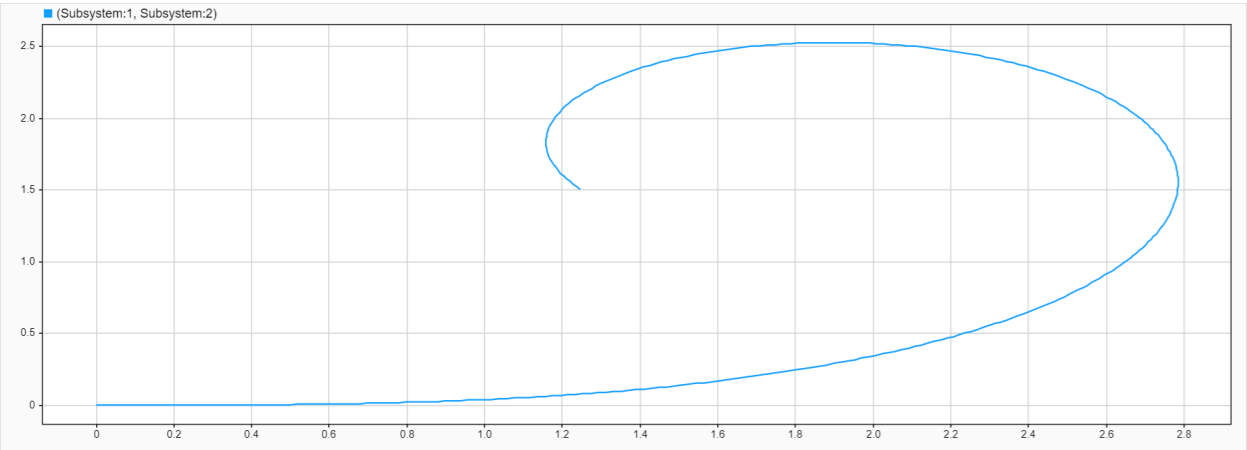


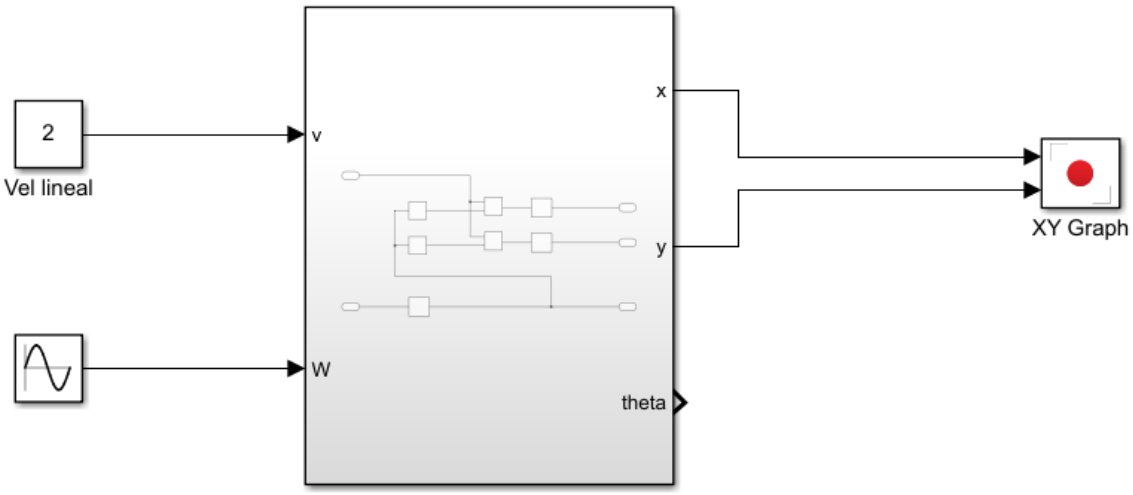
El resultado del gráfico xy es el siguiente:

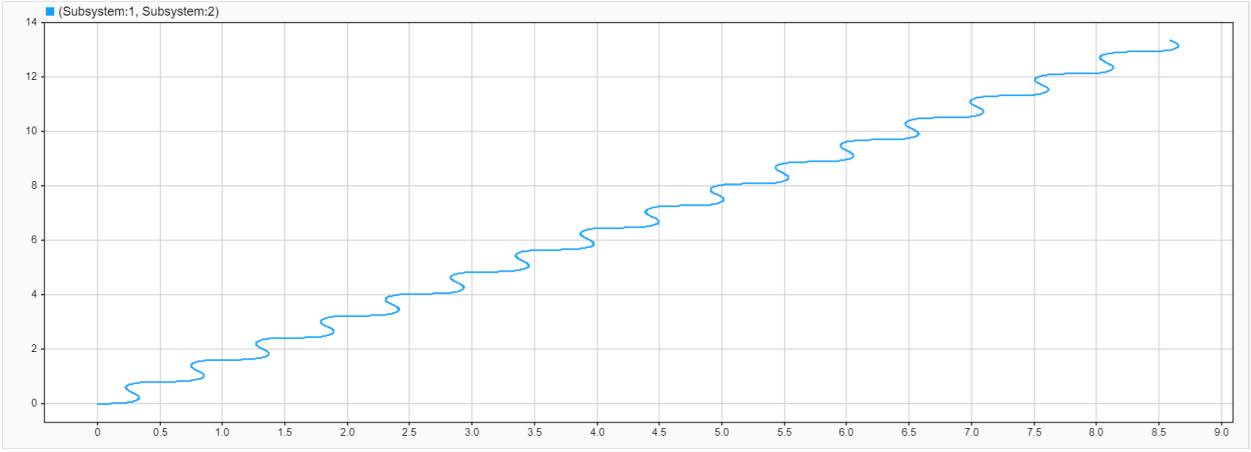


Ejecutamos con velocidad angular no constante (rampa):

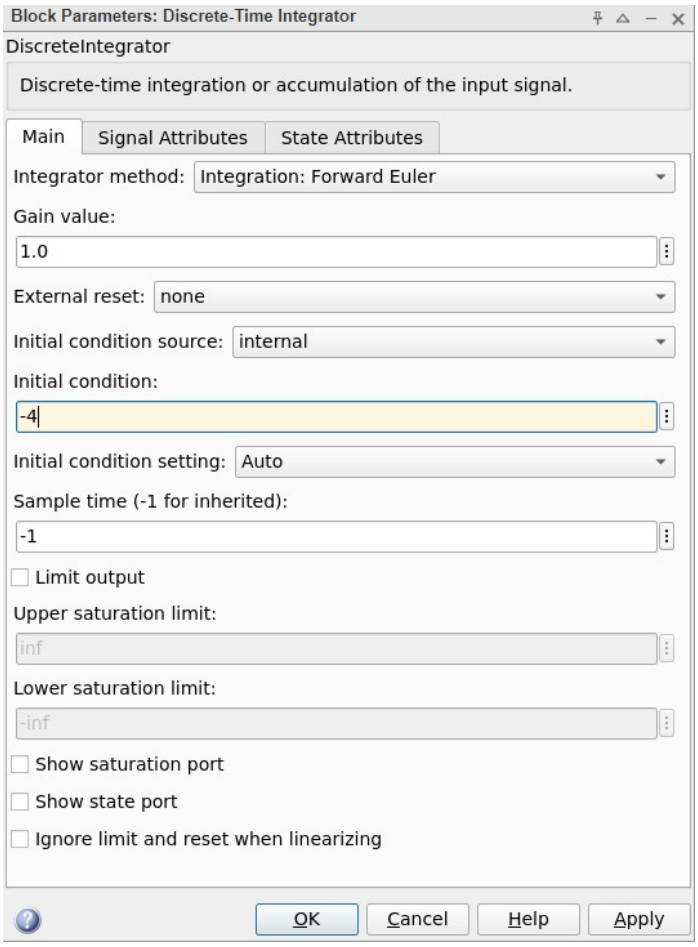


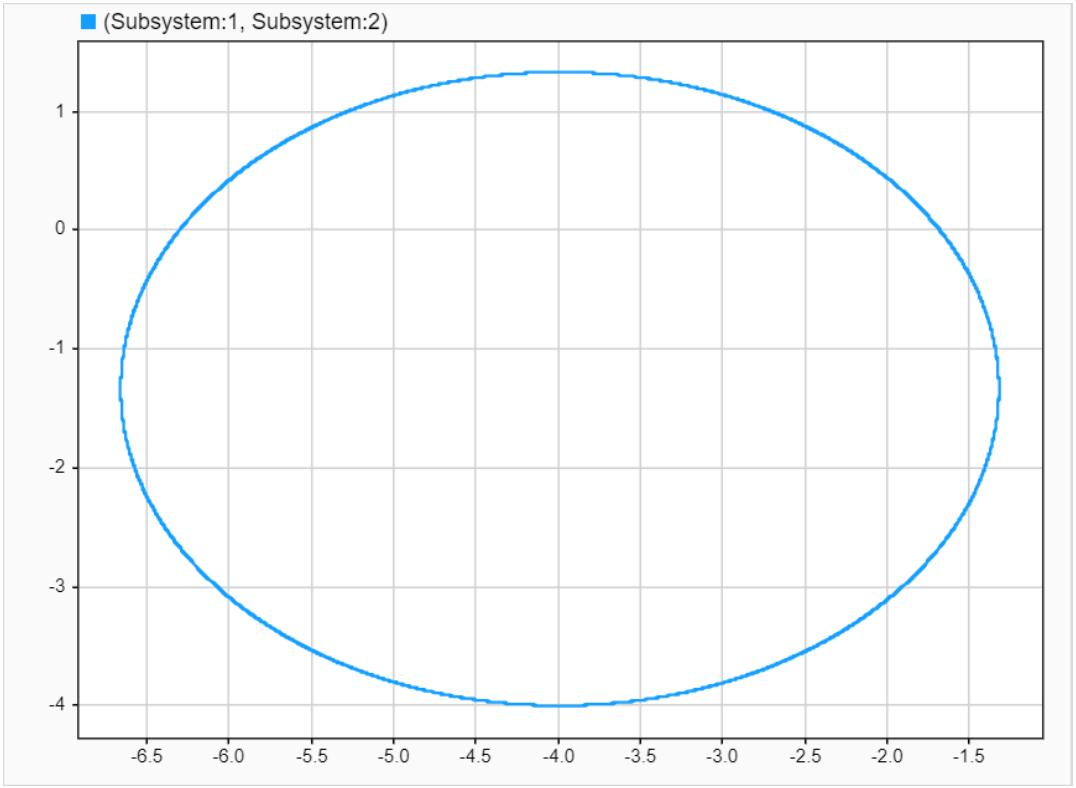


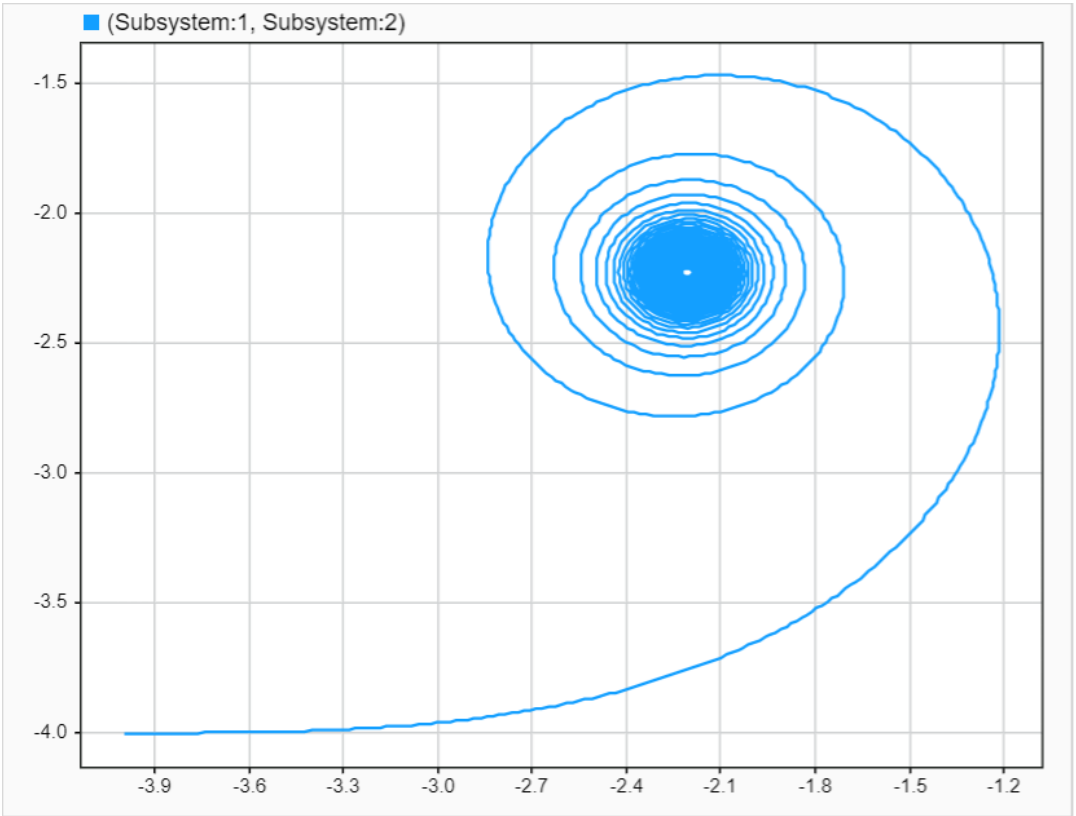


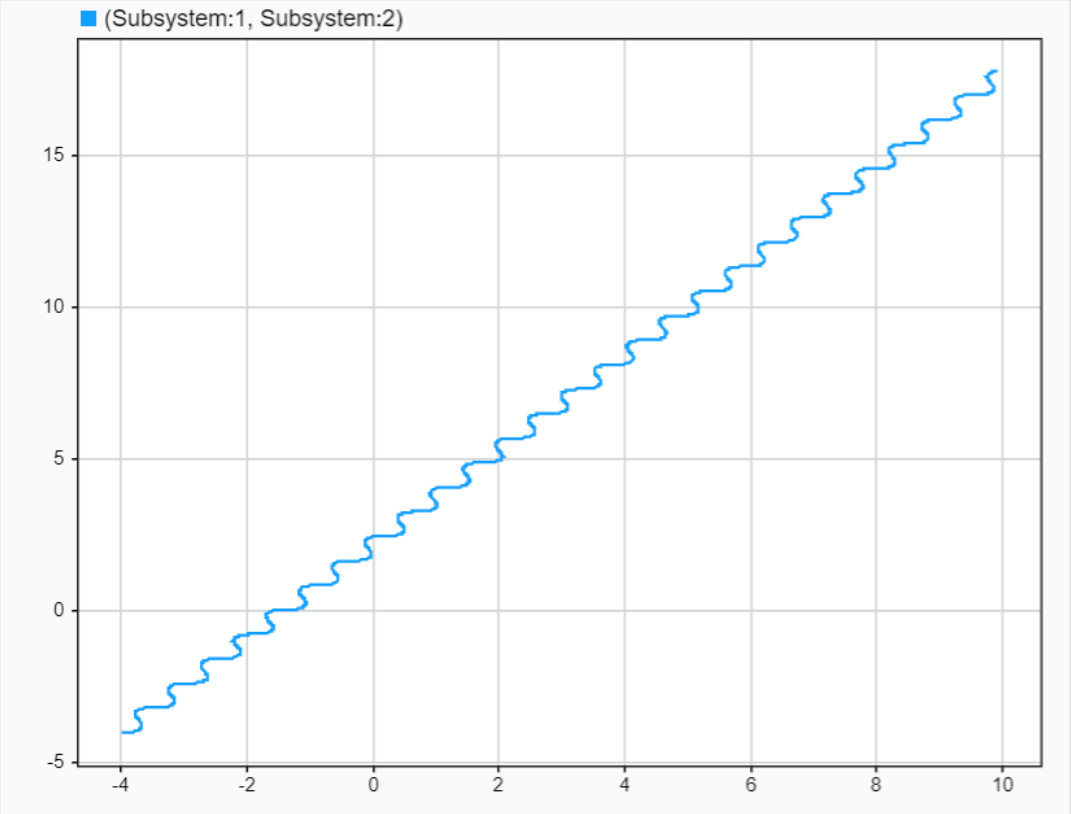


Modifique la posición inicial del robot para que comience a moverse desde el punto (-4,-4) y realice estas simulaciones de nuevo.


Block Parameters: Discrete-Time Integrator
DiscreteIntegrator
Discrete-time integration or accumulation of the input signal.
Main
Signal Attributes
State Attributes
Integrator method:
Integration: Forward Euler
Gain value:
1.0
External reset:
none
Initial condition source:
internal
Initial condition:
-4
Initial condition setting:
Auto
Sample time (-1 for inherited):
-1
Limit output
Upper saturation limit:
inf
Lower saturation limit:
-inf
Show saturation port
Show state port
Ignore limit and reset when linearizing
OK
Cancel
Help
Apply


(Subsystem:1, Subsystem:2)
1
0
-1
-2
-3
-4
-6.5
-6.0
-5.5
-5.0
-4.5
-4.0
-3.5
-3.0
-2.5
-2.0
-1.5


(Subsystem:1, Subsystem:2)
-1.5
-2.0
-2.5
-3.0
-3.5
-4.0
-3.9
-3.6
-3.3
-3.0
-2.7
-2.4
-2.1
-1.8
-1.5
-1.2



Las gráficas funcionan de la misma manera tal y como se esperaba, sólo cambia la posición inicial, cuyas coordenadas ahora son (-4,-4) en vez de (0,0).

Problemas encontrados y soluciones

Al iniciar esta actividad, nos enfrentamos a la inexperiencia con el lenguaje por primera vez, lo que generó desafíos específicos en aspectos como la notación de ecuaciones y la utilización de términos y conceptos distintivos de dicho lenguaje. Para abordar la cuestión, optamos por consultar la documentación y llevar a cabo investigaciones en línea.

Otra de las complicaciones surgidas estuvo relacionada con la configuración de coordenadas en el ejercicio 2.6 de la segunda sección. Al ingresar las coordenadas, no se observaba ninguna representación en la gráfica XY. Esta problemática se atribuyó a la necesidad de regenerar el system block.