



Universidad  
de Alcalá

SISTEMAS DE CONTROL INTELIGENTE

---

**Práctica 0.**  
**Introducción a Matlab**

---

**Jorge Revenga Martín de Vidales**  
**Ángel Salgado Aldao**

Grado en Ingeniería Informática  
Universidad de Alcalá

29 de octubre de 2023

# Índice

<b>I</b>	<b>2</b>
<b>1. Ejercicio 1. Matrices y vectores.</b>	<b>2</b>
1.1. Código . . . . .	2
1.2. Ejecución . . . . .	2
<b>2. Ejercicio 2. Matrices y vectores.</b>	<b>3</b>
2.1. Código . . . . .	3
2.2. Ejecución . . . . .	3
<b>3. Ejercicio 3. Matrices y vectores.</b>	<b>4</b>
3.1. Código . . . . .	4
3.2. Ejecución . . . . .	6
<b>4. Ejercicio 4. Tiempo de cómputo y representación gráfica.</b>	<b>7</b>
4.1. Código . . . . .	7
4.2. Ejecución . . . . .	7
<b>5. Ejercicio 5. Representación gráfica en 3D.</b>	<b>8</b>
5.1. Código . . . . .	8
5.2. Ejecución . . . . .	9
<b>6. Ejercicio 6. Sistemas lineales.</b>	<b>10</b>
6.1. Código . . . . .	10
6.2. Ejecución . . . . .	10
<b>7. Ejercicio 7. Polinomios.</b>	<b>11</b>
7.1. Código . . . . .	11
7.2. Ejecución . . . . .	12
<b>II</b>	<b>13</b>
<b>8. Ejercicio 1. Transformadas de señales.</b>	<b>13</b>
8.1. . . . .	13
8.2. . . . .	14
8.3. . . . .	16
<b>9. Ejercicio 2. Modelado del comportamiento de un robot móvil en Simulink.</b>	<b>17</b>
9.1. Implementación de las ecuaciones de movimiento siguiendo el modelo del enunciado de la práctica. . . . .	17
9.2. Montaje de la simulación del movimiento del robot. . . . .	17
9.3. Simulación del movimiento del robot. . . . .	18
9.4. Simulación del movimiento del robot con velocidades angulares no constantes. .	19
9.4.1. Velocidad angular con función de rampa. . . . .	19
9.4.2. Velocidad angular con función sinusoidal. . . . .	19
9.5. Simulación del movimiento del robot partiendo de la posición (-4, -4). . . . .	20

## Parte I

### 1. Ejercicio 1. Matrices y vectores.

#### 1.1. Código

---

```

1  % Paso 1: Crear la matriz A y el vector v
2  A = [1 2; 3 4; 5 6; 7 8];
3  v = [14; 16; 18; 20];
4
5  % Paso 2: Obtener y visualizar la matriz B concatenando A y v
6  B = [A v];
7
8  % Paso 3: Obtener y visualizar un vector fila concatenando las filas de B
9  vector_fila = [B(1,:) B(2,:) B(3,:) B(4,:)]
10 % Opción 2:
11 vector_fila = reshape(B', 1, []);
12
13 % Paso 4: Obtener y visualizar un vector columna concatenando las columnas de B
14 vector_columna = [B(:,1); B(:,2); B(:,3)]
15 % Opción 2:
16 vector_columna = reshape(B, [], 1);
17
18 % Visualizar los resultados
19 disp('Matriz B:');
20 disp(B);
21 disp('Vector fila resultante de concatenar filas de B:');
22 disp(vector_fila);
23 disp('Vector columna resultante de concatenar columnas de B:');
24 disp(vector_columna);

```

---

#### 1.2. Ejecución

```

Matriz B:
     1     2    14
     3     4    16
     5     6    18
     7     8    20

Vector fila resultante de concatenar filas de B:
     1     2    14     3     4    16     5     6    18     7     8    20

Vector columna resultante de concatenar columnas de B:
     1
     3
     5
     7
     2
     4
     6
     8
    14
    16
    18
    20

```

Figura 1: Ejecución ejercicio 1.

## 2. Ejercicio 2. Matrices y vectores.

### 2.1. Código

---

```

1  % Paso 1: Solicitar al usuario el tamaño de la matriz cuadrada
2  n = input('Indique el tamaño de la matriz: ');
3  % Paso 2: Generar una matriz aleatoria de tamaño n x n
4  matriz = rand(n);
5  % a) Mostrar la matriz generada
6  disp('Matriz generada:');
7  disp(matriz);
8  % b) Obtener una segunda matriz con las columnas impares de la matriz inicial
9  matriz_impares = matriz(:, 1:2:end);
10 disp('Segunda matriz con columnas impares:');
11 disp(matriz_impares);
12 % c) Obtener y mostrar los elementos de la diagonal de la matriz generada
13 diagonal = diag(matriz);
14 disp('Elementos de la diagonal:');
15 disp(diagonal);
16 % d) Calcular y graficar el máximo, mínimo, medio y varianza de cada fila
17 maximos = max(matriz, [], 2);
18 minimos = min(matriz, [], 2);
19 medios = mean(matriz, 2);
20 varianzas = var(matriz, 0, 2);
21 % Crear un gráfico de barras para los valores calculados
22 figure;
23 bar([maximos, minimos, medios, varianzas]);
24 title('Valores por fila');
25 xlabel('Número de fila');
26 ylabel('Valor');
27 legend('Máximo', 'Mínimo', 'Medio', 'Varianza');

```

---

### 2.2. Ejecución

```

Indique el tamaño de la matriz: 5
Matriz generada:
    0.9797    0.5949    0.1174    0.0855    0.7303
    0.4389    0.2622    0.2967    0.2625    0.4886
    0.1111    0.6028    0.3188    0.8010    0.5785
    0.2581    0.7112    0.4242    0.0292    0.2373
    0.4087    0.2217    0.5079    0.9289    0.4588

Segunda matriz con columnas impares:
    0.9797    0.1174    0.7303
    0.4389    0.2967    0.4886
    0.1111    0.3188    0.5785
    0.2581    0.4242    0.2373
    0.4087    0.5079    0.4588

Elementos de la diagonal:
    0.9797
    0.2622
    0.3188
    0.0292
    0.4588

```

Figura 2: Ejecución ejercicio 2.

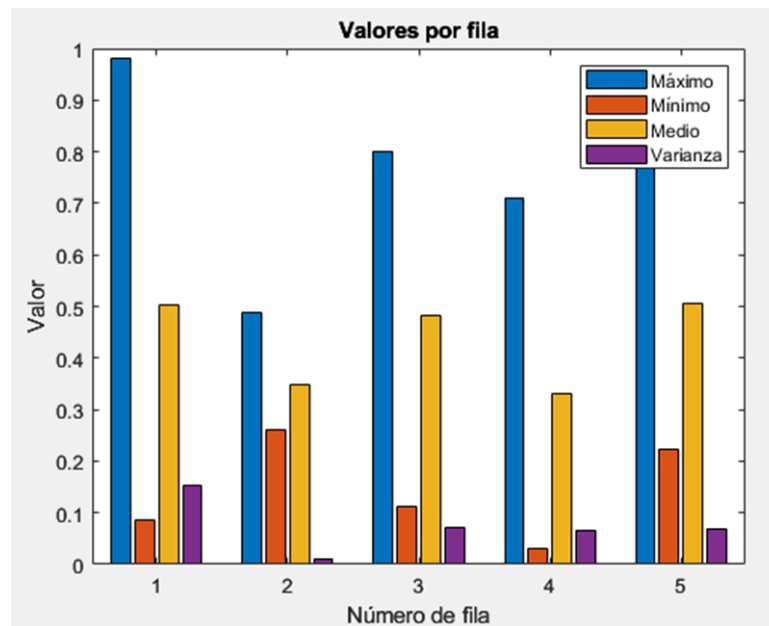


Figura 3: Gráfico ejecución ejercicio 2.

### 3. Ejercicio 3. Matrices y vectores.

#### 3.1. Código

##### IntroducirMatriz.m

---

```

1  % Función para generar y rellenar una matriz
2  function Matriz = IntroducirMatriz(Dimensiones)
3      if numel(Dimensiones) == 1 % Si solo se proporciona un número, asumimos una matriz cuadrada
4          filas = Dimensiones;
5          cols = Dimensiones;
6      elseif numel(Dimensiones) == 2
7          filas = Dimensiones(1);
8          cols = Dimensiones(2);
9      else
10         error('Formato de dimensiones incorrecto. Debe ser [filas cols].');
11     end
12
13     Matriz = zeros(filas, cols);
14
15     for i = 1:filas
16         for j = 1:cols
17             fprintf('Ingrese el valor de la posición (%d, %d) (deje en blanco para aleatorio): ', i, j);
18             valor = input('','s'); % Leer como cadena de caracteres
19
20             if isempty(valor) || isempty(strtrim(valor)) % Si el usuario ingresa un valor vacío
21                 Matriz(i, j) = rand(); % Rellenar con valor aleatorio
22             else
23                 Matriz(i, j) = str2double(valor); % Convertir entrada a número
24             end
25         end
26     end
27 end

```

---

```

1  % Paso 1: Solicitar las dimensiones de las matrices al usuario
2  dimensiones_A = input('Introduce las dimensiones de la matriz A en formato [filas cols]: ');
3  dimensiones_B = input('Introduce las dimensiones de la matriz B en formato [filas cols]: ');
4
5  % Paso 2: Generar las matrices A y B
6  A = IntroducirMatriz(dimensiones_A);

```

---

```
7 B = IntroducirMatriz(dimensiones_B);
8
9 % Paso 4: Realizar los cálculos y mostrar los resultados
10 disp('Matriz A:');
11 disp(A);
12 disp('Matriz B:');
13 disp(B);
14 disp('Transpuesta de A:');
15 disp(A');
16 disp('Transpuesta de B:');
17 disp(B');
18 disp('Inversa de A:');
19 disp(inv(A));
20 disp('Inversa de B:');
21 disp(inv(B));
22
23 if isequal(size(A), size(B)) && size(A,1) == size(A,2)
24     disp('Determinante de A: ');
25     disp(det(A));
26 else
27     disp('A no es una matriz cuadrada, no se puede calcular el determinante.');
```

---

```
28 end
29
30 if isequal(size(B), size(A)) && size(B,1) == size(B,2)
31     disp('Determinante de B: ');
32     disp(det(B));
33 else
34     disp('B no es una matriz cuadrada, no se puede calcular el determinante.');
```

---

```
35 end
36
37 disp(['Rango de A: ' num2str(rank(A))]);
38 disp(['Rango de B: ' num2str(rank(B))]);
39
40 disp('Producto matricial A*B:');
41 if size(A,2) == size(B,1)
42     disp(A * B);
43 else
44     disp('No se puede calcular el producto matricial A*B debido a dimensiones incompatibles.');
```

---

```
45 end
46
47 disp('Producto elemento a elemento A.*B:');
48 if isequal(size(A), size(B))
49     disp(A .* B);
50 else
51     disp('No se puede calcular el producto elemento a elemento A.*B debido a dimensiones incompatibles.');
```

---

```
52 end
53
54 vector_fila = [A(1, :), B(1, :)];
55 disp('Vector fila obtenido concatenando la primera fila de A y B:');
56 disp(vector_fila);
57
58 vector_columna = [A(:, 1); B(:, 1)];
59 disp('Vector columna obtenido concatenando la primera columna de A y B:');
60 disp(vector_columna);
```

---

## 3.2. Ejecución

```

Introduce las dimensiones de la matriz A en formato [filas cols]: [4 3]
Introduce las dimensiones de la matriz B en formato [filas cols]: 3
Ingrese el valor de la posición (1, 1) (deje en blanco para aleatorio):
Ingrese el valor de la posición (1, 2) (deje en blanco para aleatorio):
Ingrese el valor de la posición (1, 3) (deje en blanco para aleatorio):
Ingrese el valor de la posición (2, 1) (deje en blanco para aleatorio):
Ingrese el valor de la posición (2, 2) (deje en blanco para aleatorio):
Ingrese el valor de la posición (2, 3) (deje en blanco para aleatorio):
Ingrese el valor de la posición (3, 1) (deje en blanco para aleatorio):
Ingrese el valor de la posición (3, 2) (deje en blanco para aleatorio):
Ingrese el valor de la posición (3, 3) (deje en blanco para aleatorio):
Ingrese el valor de la posición (4, 1) (deje en blanco para aleatorio):
Ingrese el valor de la posición (4, 2) (deje en blanco para aleatorio):
Ingrese el valor de la posición (4, 3) (deje en blanco para aleatorio):
Ingrese el valor de la posición (1, 1) (deje en blanco para aleatorio):
Ingrese el valor de la posición (1, 2) (deje en blanco para aleatorio):
Ingrese el valor de la posición (1, 3) (deje en blanco para aleatorio):
Ingrese el valor de la posición (2, 1) (deje en blanco para aleatorio):
Ingrese el valor de la posición (2, 2) (deje en blanco para aleatorio):
Ingrese el valor de la posición (2, 3) (deje en blanco para aleatorio):
Ingrese el valor de la posición (3, 1) (deje en blanco para aleatorio):
Ingrese el valor de la posición (3, 2) (deje en blanco para aleatorio):
Ingrese el valor de la posición (3, 3) (deje en blanco para aleatorio):

Matriz A:
    0.5736    0.3617    0.1613
    0.8864    0.9020    0.7586
    0.3618    0.4962    0.6587
    0.2890    0.1431    0.3081

Matriz B:
    0.3474    0.1395    0.2162
    0.5558    0.0512    0.4174
    0.7116    0.8793    0.6015

Transpuesta de A:
    0.5736    0.8864    0.3618    0.2890
    0.3617    0.9020    0.4962    0.1431
    0.1613    0.7586    0.6587    0.3081

Transpuesta de B:
    0.3474    0.5558    0.7116
    0.1395    0.0512    0.8793
    0.2162    0.4174    0.6015

A no es una matriz cuadrada, no se puede calcular el determinante.
B no es una matriz cuadrada, no se puede calcular el determinante.
Rango de A: 3
Rango de B: 3
Producto matricial A*B:
    0.5151    0.2404    0.3720
    1.3491    0.8369    1.0244
    0.8702    0.6550    0.6815
    0.3992    0.3186    0.3075

Producto elemento a elemento A.*B:
No se puede calcular el producto elemento a elemento A.*B debido a dimensiones incompatibles.
Vector fila obtenido concatenando la primera fila de A y B:
    0.5736    0.3617    0.1613    0.3474    0.1395    0.2162

Vector columna obtenido concatenando la primera columna de A y B:
    0.5736
    0.8864
    0.3618
    0.2890
    0.3474
    0.5558
    0.7116

```

Figura 4: Ejecución ejercicio 3.

## 4. Ejercicio 4. Tiempo de cómputo y representación gráfica.

### 4.1. Código

---

```

1  % Inicializar matrices para almacenar los tiempos de procesamiento
2  tiempo_rango = [];
3  tiempo_determinante = [];
4
5  % Bucle para calcular los tiempos
6  for n = 1:25
7      % Generar una matriz aleatoria de tamaño n x n
8      matriz = rand(n, n);
9
10     % Calcular el tiempo para el cálculo del rango
11     tic;
12     rango = rank(matriz);
13     tiempo_rango(n) = toc;
14
15     % Calcular el tiempo para el cálculo del determinante
16     tic;
17     determinante = det(matriz);
18     tiempo_determinante(n) = toc;
19 end
20
21 % Crear una figura para representar los tiempos
22 figure;
23 plot(tiempo_rango);
24 hold on;
25 plot(tiempo_determinante);
26 xlabel('Tamaño de la matriz');
27 ylabel('Tiempo (segundos)');
28 title('Tiempo de cálculo del rango y determinante en función del tamaño de la matriz');
29 legend('Tiempo de cálculo del rango', 'Tiempo de cálculo del determinante');
30
31 hold off;

```

---

### 4.2. Ejecución

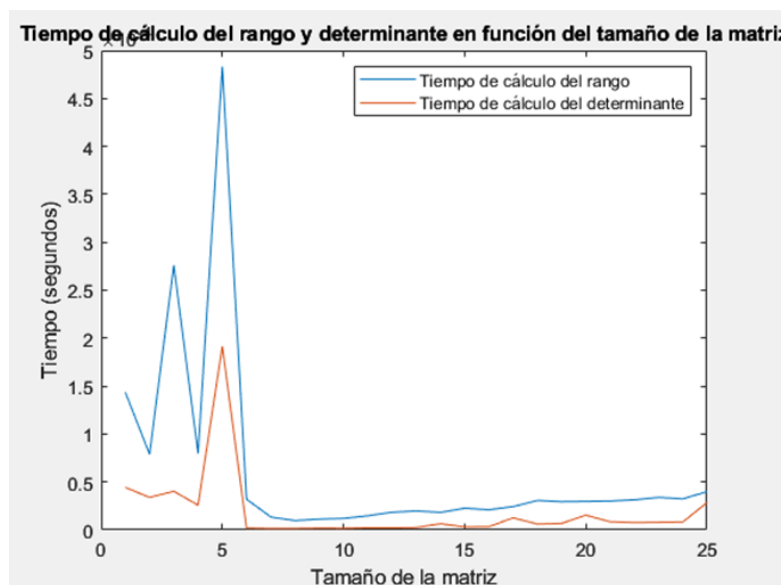


Figura 5: Gráfico ejecución ejercicio 4.



## 5. Ejercicio 5. Representación gráfica en 3D.

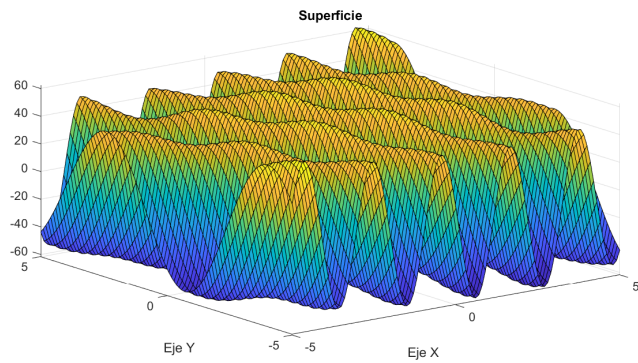
### 5.1. Código

---

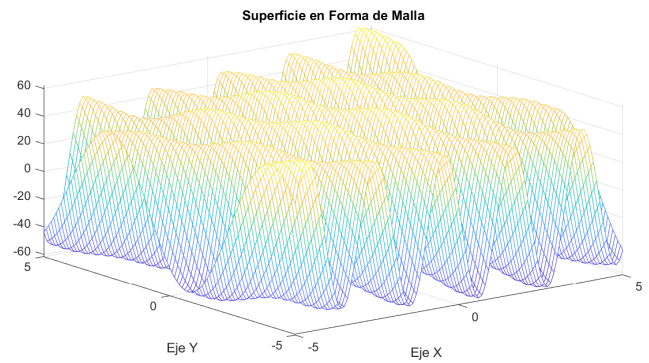
```
1  % Definir el rango de valores para x y y
2  x = -5:0.1:5;
3  y = -5:0.1:5;
4
5  % Crear una malla de valores para x y y
6  [X, Y] = meshgrid(x, y);
7
8  % Calcular la función z en función de x y y
9  Z = Y .* sin(pi * X / 10) + 5 * cos((X.^2 + Y.^2) / 8) + cos(X + Y) * cos(3 * X - Y);
10
11 figure;
12
13 subplot(2, 2, 1);
14 surf(X, Y, Z);
15 title('Superficie');
16 xlabel('Eje X');
17 ylabel('Eje Y');
18
19 subplot(2, 2, 2);
20 mesh(X, Y, Z);
21 title('Superficie en Forma de Malla');
22 xlabel('Eje X');
23 ylabel('Eje Y');
24
25 subplot(2, 2, 3);
26 contourf(X, Y, Z);
27 colorbar;
28 title('Contorno con Barra de Color');
29 xlabel('Eje X');
30 ylabel('Eje Y');
31
32 % Ajustar tamaño a la pantalla
33 set(gcf, 'Position', get(0, 'Screensize'));
34
```

---

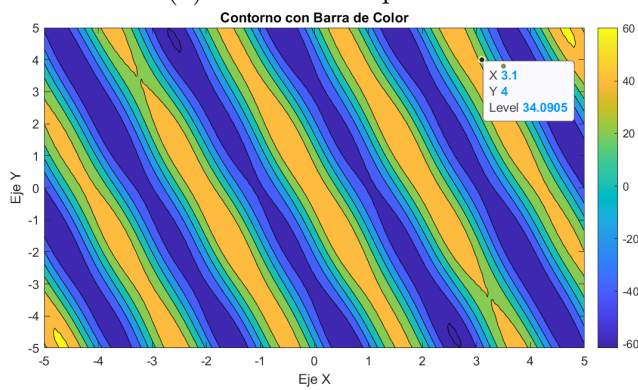
## 5.2. Ejecución



(a) Gráfico de superficie.



(b) Gráfico de superficie en forma de malla.



(c) Gráfico de contorno con barra de color.

Figura 6: Gráficos ejecución ejercicio 5.

## 6. Ejercicio 6. Sistemas lineales.

### 6.1. Código

---

```

1  % Definir las matrices A y b
2  A = [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];
3  As = [90; 59; 15; 10; 80; 17; 93; 51; 41; 76];
4
5  b = [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];
6  bs = [317; 237; 319; 239; 321; 241; 323; 243; 325; 245];
7
8  % 1. Obtener el número de condición de la matriz A
9  c = cond(A);
10 disp(['Número de condición de A: ' num2str(c)]);
11
12 % 2. Resolver los sistemas de ecuaciones A = As y b = bs
13 X = linsolve(A, As);
14 disp('Solución del sistema de ecuaciones A:');
15 disp(X);
16 Y = linsolve(b,bs);
17 disp('Solución del sistema de ecuaciones b (sin ruido):');
18 disp(Y)
19
20 % 3. Añadir ruido a la matriz b y resolver el sistema
21 r = normrnd(0, 1, 10, 1);
22 b2 = bs + r;
23
24 % 4. Resolver el sistema de ecuaciones b = bs
25 B = b;
26 Y = linsolve(B, b2);
27 disp('Solución del sistema de ecuaciones b (con ruido):');
28 disp(Y);

```

---

### 6.2. Ejecución

```

Número de condición de A: 2.7257
Solución del sistema de ecuaciones A:
-2.2571
 4.9336
 5.2986
 3.5649

Solución del sistema de ecuaciones b (sin ruido):
 6.3593
 6.3694
316.2992
216.2357

Solución del sistema de ecuaciones b (con ruido):
 4.3270
 6.0054
317.5485
217.5285

```

Figura 7: Gráfico ejecución ejercicio 6.

## 7. Ejercicio 7. Polinomios.

### 7.1. Código

#### raices.m

---

```

1  function [solucion, reales, complejas] = raices(poli_1, poli_2)
2      % Recoge los arrays con los que se crean los polinomios
3      p1 = poly2sym(poli_1);
4      p2 = poly2sym(poli_2);
5
6      % Solicita al usuario a cuál de los polinomios o al producto se aplicará la solución
7      fprintf('Elija a cuál de los polinomios o al producto desea aplicar la solución:\n');
8      fprintf('1. Polinomio 1\n');
9      fprintf('2. Polinomio 2\n');
10     fprintf('3. Producto de los polinomios\n');
11
12     choice = input('Ingrese el número correspondiente: ');
13
14     switch choice
15         case 1
16             % Calcular raíces del primer polinomio
17             solucion = roots(poli_1);
18         case 2
19             % Calcular raíces del segundo polinomio
20             solucion = roots(poli_2);
21         case 3
22             % Calcular raíces del producto de los polinomios
23             prod_poli = conv(poli_1, poli_2);
24             solucion = roots(prod_poli);
25         otherwise
26             error('Opción no válida. Debe elegir 1, 2 o 3.');
```

---

```

27     end
28
29     % Clasificar las raíces
30     reales = sum(isreal(solucion));
31     complejas = length(solucion) - reales;
32
33     % Representar las raíces en el plano complejo
34     figure;
35     hold on;
36     plot(real(solucion), imag(solucion), 'ro');
37     xlabel('Parte Real');
38     ylabel('Parte Imaginaria');
39     title('Ubicación de las raíces en el plano complejo');
40     hold off;
41
42 end
```

---

```

1  [solucion, reales, complejas] = raices([1 2 2], [1 3]);
2  fprintf('Raíces: ');
3  disp(solucion);
4  fprintf('Raíces reales: %d\n', reales);
5  fprintf('Raíces complejas: %d\n', complejas);
```

---

## 7.2. Ejecución

```
Elija a cuál de los polinomios o al producto desea aplicar la solución:  
1. Polinomio 1  
2. Polinomio 2  
3. Producto de los polinomios  
Ingrese el número correspondiente: 3  
Raíces: -3.0000 + 0.0000i  
        -1.0000 + 1.0000i  
        -1.0000 - 1.0000i  
  
Raíces reales: 0  
Raíces complejas: 3
```

Figura 8: Ejecución ejercicio 7.

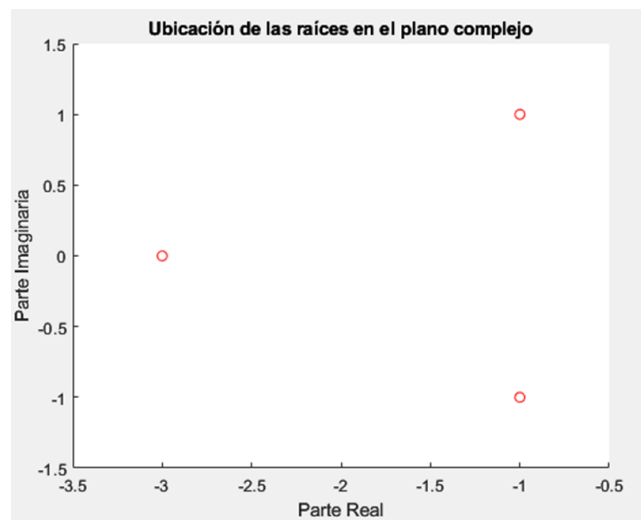


Figura 9: Gráfico ejecución ejercicio 7.

## Parte II

### 8. Ejercicio 1. Transformadas de señales.

#### 8.1.

Obtenga la transformada  $z$  de la siguiente función:  $f(k) = 2 + 5k + k^2$ . Represente gráficamente las señales original y transformada.

#### Código

---

```

1  % 1.1 Representar gráficamente f(k) = 2 + 5k + k^2 y su transformada
2
3  syms k z; % Define las variables simbólicas k y z
4  f_k = 2 + 5*k + k.^2;
5
6  % Crear una función anónima que evalúe la expresión simbólica f(k)
7  f = matlabFunction(f_k);
8
9  % Evaluar la función en el rango de índices (vamos a usar 1:10)
10 valores = f(1:10);
11
12 figure; % Crea una gráfica nueva
13
14 % Graficar la función discreta utilizando stem
15 stem(1:10, valores);
16 title("f(k)");
17 xlabel('Índices');
18 ylabel('Valores');
19
20 F_z = ztrans(f_k, k, z); % Calcula la transformada Z de f(k)
21
22 % Crear una función anónima que evalúe la expresión simbólica F(z)
23 f = matlabFunction(F_z);
24
25 % Evaluar la función en el rango de índices (vamos a usar 1:10)
26 valores = f(1:10);
27
28 figure; % Crea una gráfica nueva
29
30 % Graficar la función discreta utilizando stem
31 stem(1:10, valores);
32 title("Transformada Z de f(k)");
33 xlabel('Índices');
34 ylabel('Valores');

```

---

## Ejecución

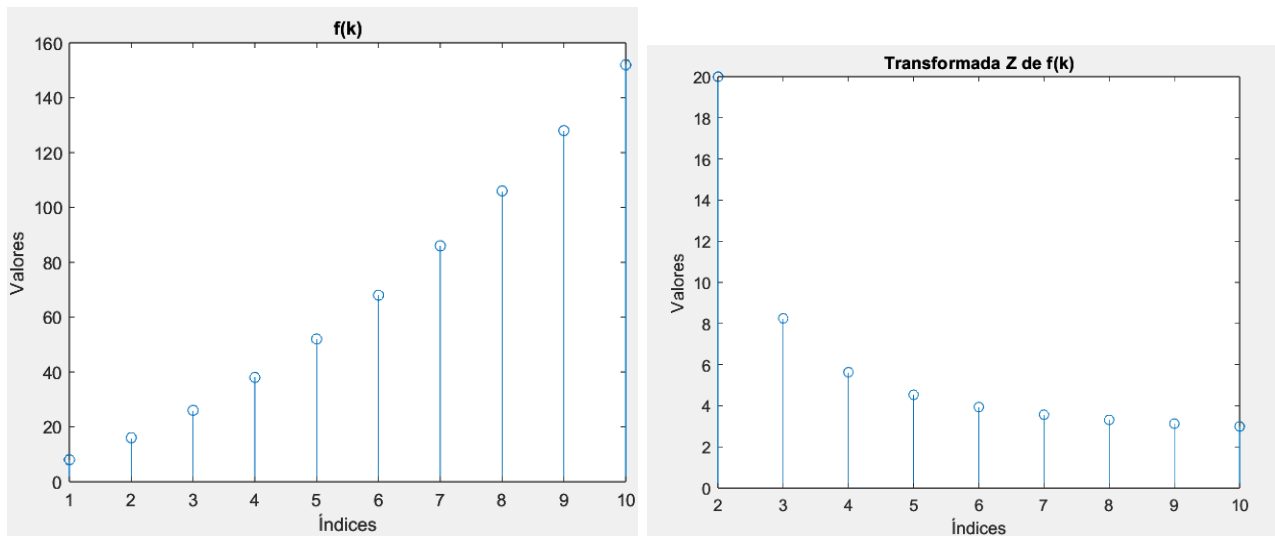


Figura 10: Ejecución ejercicio 1.1.

## 8.2.

Obtenga la transformada  $z$  de la siguiente función:  $f(k) = \sin(k) \cdot e^{-ak}$ . Represente gráficamente, de nuevo, las señales original y transformada.

## Código

```

1  %1: Representación en 3D, en función de k y de a
2  syms k a z; % Define las variables simbólicas k y a
3  f_k = sin(k) * exp(-a * k); % Define la función
4
5  % Crea una malla de valores de k y a
6  [k_values, a_values] = meshgrid(1:10, 1:10);
7
8  % Evalúa la función en la malla de valores
9  f_values = double(subs(f_k, [k, a], {k_values, a_values}));
10
11 % Grafica en 3D
12 figure;
13 surf(k_values, a_values, f_values);
14 title('Gráfica 3D de f(k, a)');
15 xlabel('k');
16 ylabel('a');
17 zlabel('f(k, a)');
18
19
20 F_z = ztrans(f_k, k, z); % Calcula la transformada Z de f(k)
21
22 % Evalúa la función en la malla de valores
23 F_values = double(subs(F_z, [z, a], {k_values, a_values}));
24
25 % Grafica en 3D
26 figure;
27 surf(k_values, a_values, F_values);
28 title('Gráfica 3D de f(z, a)');
29 xlabel('k');
30 ylabel('a');
31 zlabel('f(k, a)');
32
33 %2: Representación en 2D, con a = -1

```

```

34
35 f_k = sind(k) * exp(1*k);
36
37 % Crear una función anónima que evalúe la expresión simbólica f(k)
38 f = matlabFunction(f_k);
39
40 % Evaluar la función en el rango de índices (vamos a usar 1:10)
41 valores = f(1:10);
42
43 figure;
44 % Graficar la función discreta utilizando stem
45 stem(1:10, valores);
46 title("f(k)");
47 xlabel('Índices');
48 ylabel('Valores');
49
50 F_z = ztrans(f_k, k, z); % Calcula la transformada Z de f(k)
51
52 % Crear una función anónima que evalúe la expresión simbólica F(z)
53 f = matlabFunction(F_z);
54
55 % Evaluar la función en el rango de índices 1:10
56 valores = f(1:10);
57
58 figure;
59 % Graficar la función discreta utilizando stem
60 stem(1:10, valores);
61 title("Transformada Z de f(k)");
62 xlabel('Índices');
63 ylabel('Valores');

```

## Ejecución

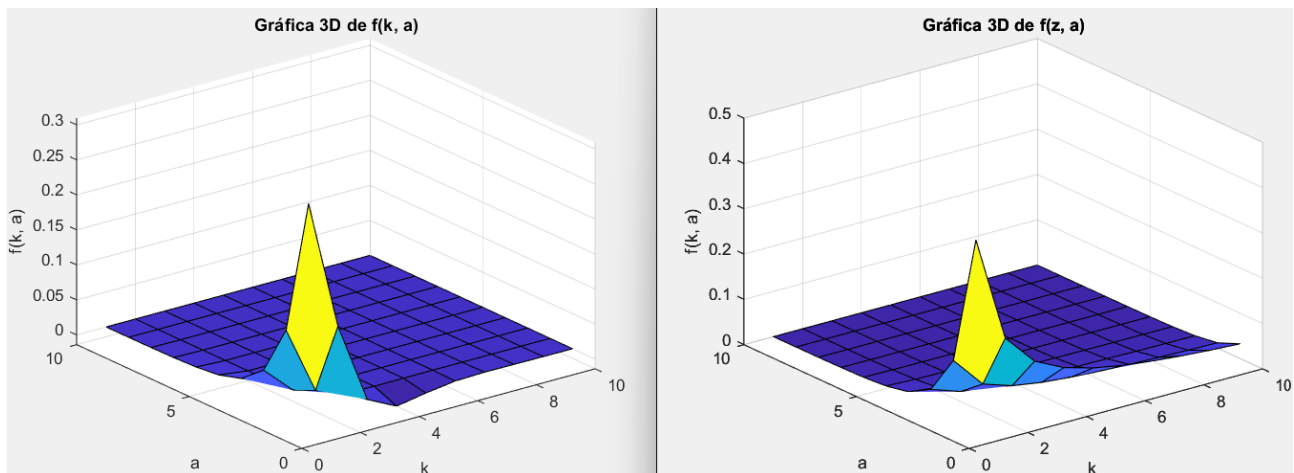


Figura 11: Ejecución 1 ejercicio 1.2.



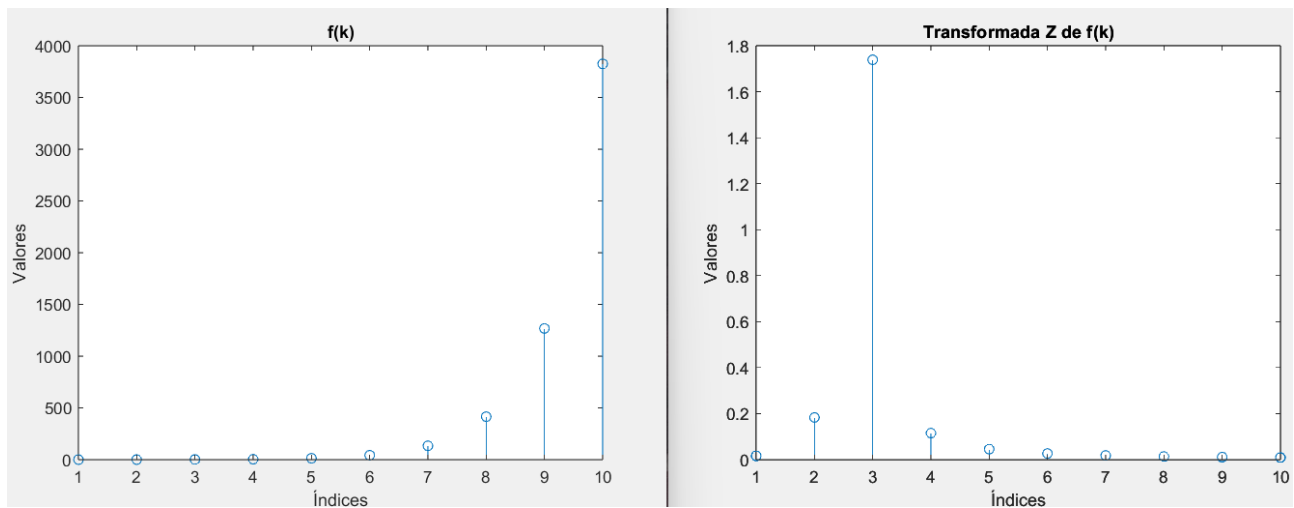


Figura 12: Ejecución 2 ejercicio 1.2.

### 8.3.

Dada la siguiente función de transferencia discreta:

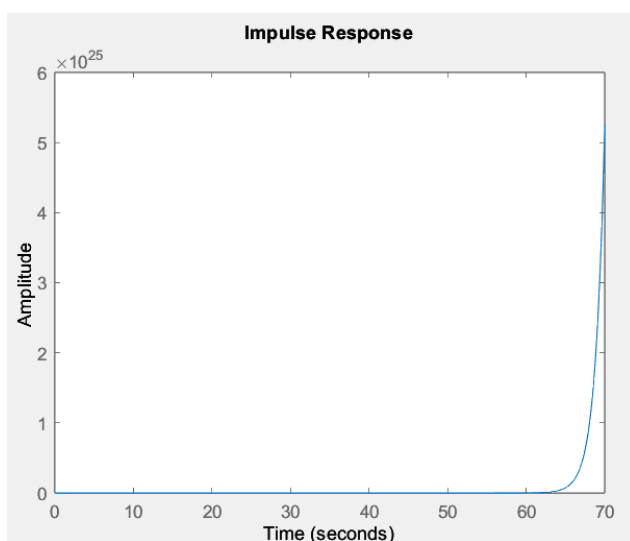
$$T(z) = \frac{0,4 \cdot z^2}{z^3 - z^2 + 0,1z + 0,02}$$

Obtenga y represente la respuesta al impulso del sistema.

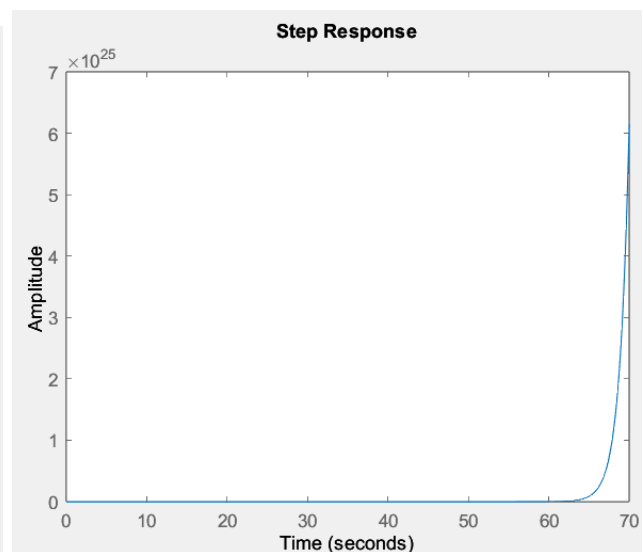
Obtenga y represente la respuesta del sistema ante una entrada escalón.

### Código

```
1 sys = tf([0.4 0 0], [1 -1 0.1 0.02]); %Definición del sistema a partir de su función de transferencia
2 impulse(sys) %Respuesta al impulso del sistema
3 figure;
4 step(sys) %Respuesta a entrada escalón
```



(a) Ejecución 1 ejercicio 1.2.



(b) Ejecución 2 ejercicio 1.3.

## 9. Ejercicio 2. Modelado del comportamiento de un robot móvil en Simulink.

### 9.1. Implementación de las ecuaciones de movimiento siguiendo el modelo del enunciado de la práctica.

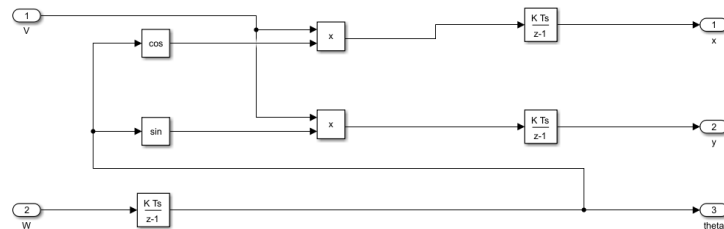


Figura 14: Implementación de las ecuaciones de movimiento del robot.

### 9.2. Montaje de la simulación del movimiento del robot.

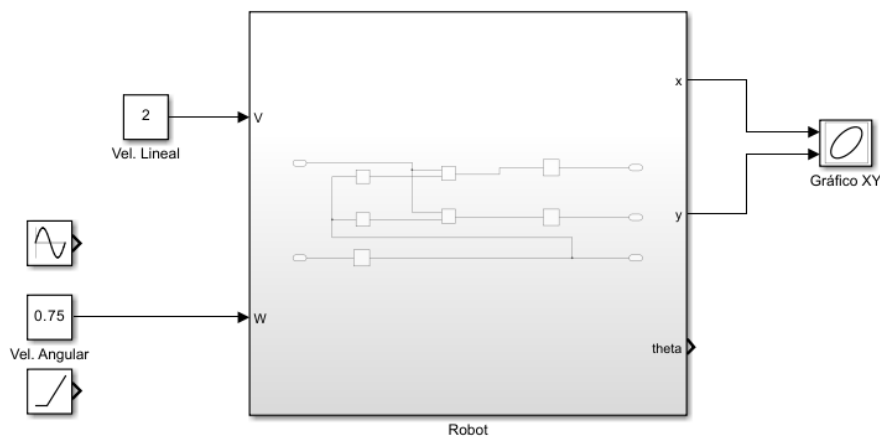


Figura 15: Sistema robot.

### 9.3. Simulación del movimiento del robot.

El movimiento del robot al tener una velocidad angular constante debería describir una circunferencia. Al hacer funcionar el sistema observamos (figura 18) la generación de una elipse, pero al fijarnos detenidamente podemos ver que el ancho y alto de la elipse son iguales y el espacio está siendo deformado automáticamente por Simulink (Cada cuadrado de la cuadrícula tiene un ancho de 0.2 y un alto de 0.5). En conclusión, el comportamiento del sistema es el esperado.

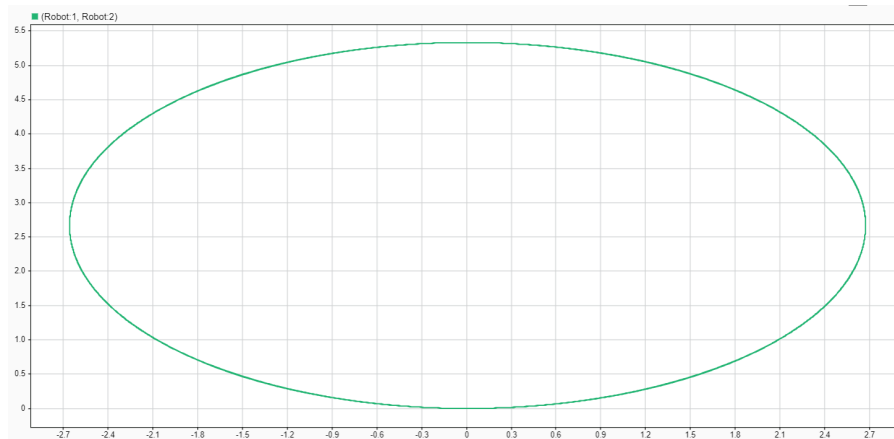


Figura 16: Movimiento del robot con Vel. Angular constante ( $\times 0.75$ ).

## 9.4. Simulación del movimiento del robot con velocidades angulares no constantes.

### 9.4.1. Velocidad angular con función de rampa.

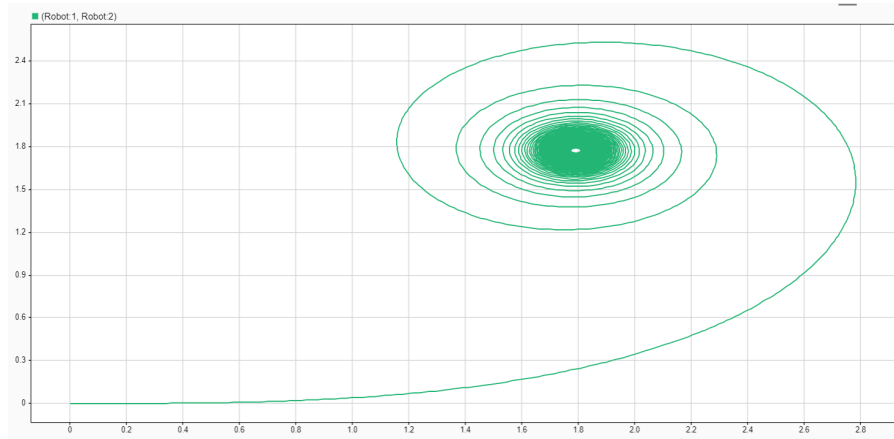


Figura 17: Movimiento del robot con Vel. Angular con función rampa.

Dado que la función de rampa en Simulink es creciente, esta hace que la velocidad angular del robot aumente constantemente, resultando en que la posición del robot converja gradualmente hacia un punto específico. Podemos observar que el sistema exhibe el comportamiento esperado.

### 9.4.2. Velocidad angular con función sinusoidal.



Figura 18: Movimiento del robot con Vel. Angular con función sinusoidal.

Podemos observar que el movimiento del robot es similar a una función trigonométrica (sen/cos), aplanada automáticamente por el programa y claramente siguiendo una trayectoria diagonal. Esta trayectoria diagonal probablemente se deba a la velocidad y posiciones iniciales del robot.

## 9.5. Simulación del movimiento del robot partiendo de la posición (-4, -4).

Para alterar la posición inicial del robot, se ha realizado una modificación en el sistema, añadiendo 4 bloques "Decrement Real World.<sup>a</sup> las variables de salida x e y.

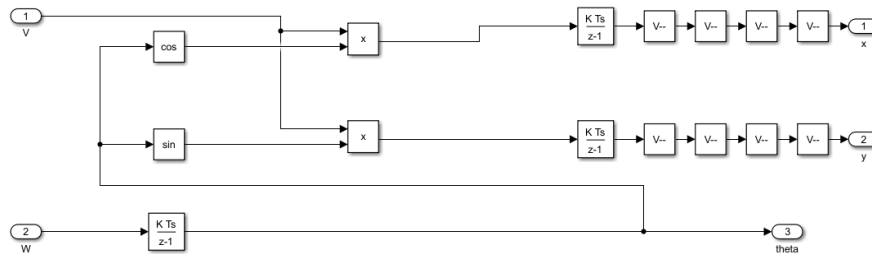
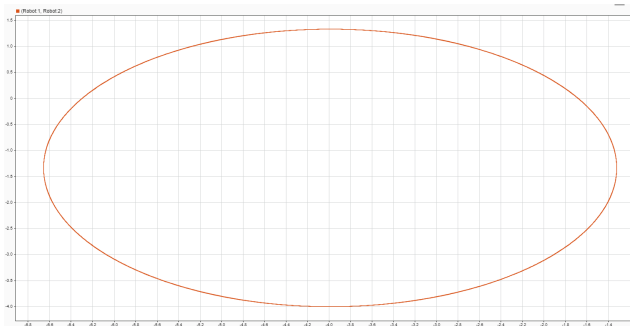
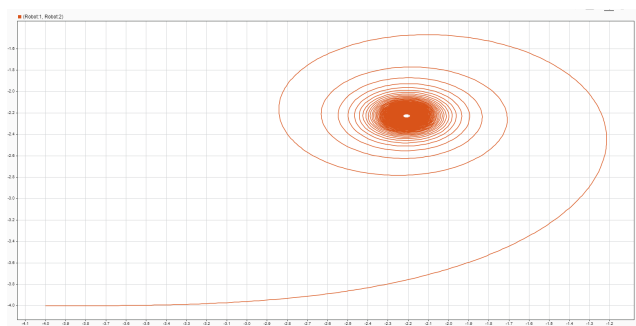


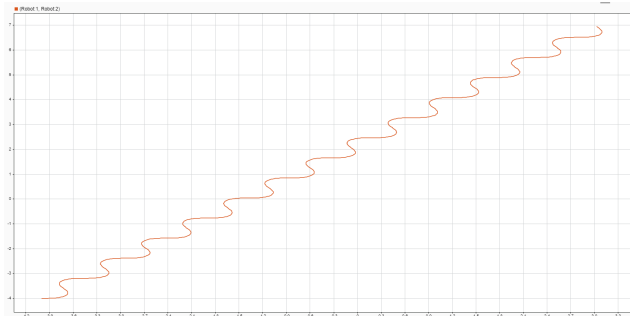
Figura 19: Movimiento del robot con Vel. Angular con función sinusoidal.



(a) Movimiento del robot con Vel. Angular constante ( $\times 0.75$ ) empezando en (-4, -4)



(b) Movimiento del robot con Vel. Angular con función rampa empezando en (-4, -4)



(c) Movimiento del robot con Vel. Angular con función sinusoidal empezando en (-4, -4)

Las gráficas tienen la misma forma, con cada punto a desplazado 4 unidades hacia abajo y hacia la izquierda