PRÁCTICA 0 - INTRODUCCIÓN A MATLAB

SISTEMAS DE CONTROL INTELIGENTE

Mario Fernández Rueda

Francisco González Velasco

PUESTO 10 DE LABORATORIO DEL GRUPO 4A2

Universidad de Alcalá

Primera Parte - MATLAB

Ejercicio 1 – Matrices y Vectores.

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

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

Código:

```
A = [1 2; 3 4; 5 6; 7 8]    % declaración matriz
v = [14; 16; 18; 20]    % declaración vector
```

Salida:

```
A =                 v =

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

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

Código:

```
B = [A v]   % concatenación
```

Salida:

```
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.**

El enunciado de este apartado nos indica que todas las filas de la matriz B deben unirse en una sola fila. Es decir, se mostrará en una única fila primero la fila 1, luego la fila 2 y así en orden hasta el final.

Código:

```
vector_fila = B' % transpuesta de B
vector_fila = vector_fila(:)'  % '(:)' transforma columnas de matriz a vector columna, y transpone (se convierte en fila)
```

Salida:

```
vector_fila =

     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.**

En este apartado ocurre lo mismo que en el anterior pero con las columnas.

Código:

```
vector_columna = B(:) % transforma columnas de matriz a vector columna
```

Salida:

```
vector_columna =

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

Ejercicio 2 – Matrices y Vectores.

1. **El 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".**

Código:

```
dimension = input("Indique el tamaño de la matriz ");
% matriz dimensionXdimension rellenada con valores aleatorios del 1 al 10
A = randi([1, 10], dimension, dimension);
```

2. **A partir de la matriz construida, el script deberá calcular y presentar por pantalla los siguientes datos:**

a. **Matriz generada.**

Código:

```
disp(A);
```

Salida:

```
 9     6     4     5
 9     6     2     7
 3     9    10     6
 7     3     7     7
```

b. **Una segunda matriz formada por las columnas impares de la matriz inicial.**

Código:

```
% se cogen todas las filas
% de las columnas se empieza por la primera y se itera de dos en dos hasta el final
B = A(:,1:2:end)
```

Salida:

```
B =

     9     4
     9     2
     3    10
     7     7
```

c. **El valor de los elementos de la diagonal de la matriz generada.**

Código:

```
C = diag(A)
```

Salida:

```
C =

     9
     6
    10
     7
```

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.**

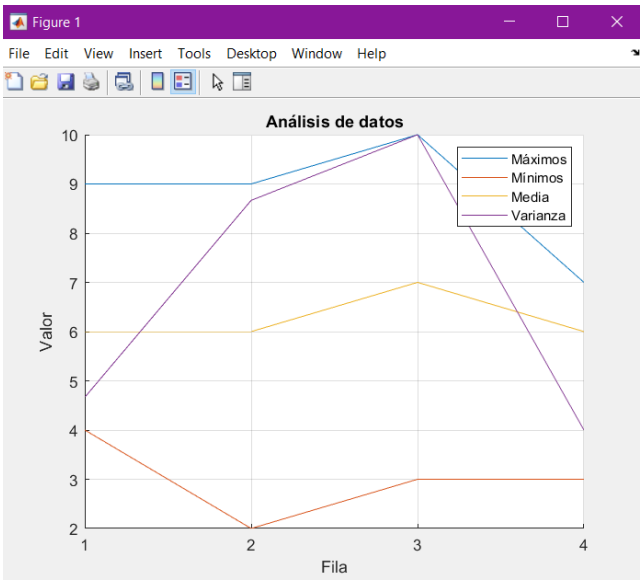
Código:

```
maximos = max(A')
minimos = min(A')
media = mean(A')
varianza = var(A')
```

Se usan las funciones correspondientes de media, mínimos, máximos y varianza, pero sobre la transpuesta de la matriz, porque lo que hacen esas funciones es dar ese resultado de los valores de cada columna. Por tanto, para que estas funciones se hagan sobre las filas se debe aplicar las transpuestas.

Salida:

```
maximos =

     9     9    10     7


minimos =

     4     2     3     3


media =

     6     6     7     6


varianza =

    4.6667    8.6667   10.0000    4.0000
```



Ejercicio 3 – Matrices y Vectores.

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

Código:

```
fprintf("1\n")
dimensiones = input("Indique las dimensiones de la matriz como una lista\n");

if length(dimensiones) == 1
    filas = dimensiones
    cols = dimensiones
else
    filas = dimensiones(1)
    cols = dimensiones(2)
end
```

Salida:

```
Indique las dimensiones de la matriz como una lista
[3 3]

filas =

     3


cols =

     3
```

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.**

   function Matriz = IntroducirMatriz(Dimensiones);

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.**

Código:

rellenarMatriz.m

```
function Matriz = rellenarMatriz(dim)
    Matriz = zeros(dim(1), dim(2));

    for i = 1:dim(1)
        for j = 1:dim(2)
            entrada = input(['Ingrese un valor para la posición (', num2str(i), ',', num2str(j), ') o "r" para rellenar el resto aleatoriamente: '], 's'); % string

            if entrada == 'r'
                for ii = i:dim(1)
                    for jj = (j:dim(2))
                        Matriz(ii, jj) = randi([1, 10]);
                    end
                    j = 1; % recogemos carrete para nueva fila
                end
                return; % Salir de la función ya que la matriz está completa
            else
                % Convertir la entrada a un número y asignarlo a la matriz
                Matriz(i, j) = str2double(entrada);
            end
        end
    end
end
```

```
fprintf("2 y 3\n")
fprintf("Primera matriz\n")
A = rellenarMatriz([filas cols]);
fprintf("Segunda matriz\n")
B = rellenarMatriz([filas cols]);
```

Salida:

```
2 y 3
Primera matriz
Ingrese un valor para la posición (1,1) o "r" para rellenar el resto aleatoriamente: r
Segunda matriz
Ingrese un valor para la posición (1,1) o "r" para rellenar el resto aleatoriamente: r
```

4. Calcule y muestre por pantalla:

- **Las matrices generadas A y B**

Código:

```
fprintf("\n4\n")
A
B
```

Salida:

```
4

A =                         B =

     7     5     6             3     6     7
     5     2     5             5     3    10
     9     9     3             1     2     2
```

- **La transpuesta e inversa de cada una de las matrices**

Código:

```
A_transpuesta = B'
B_transpuesta = B'

if filas == cols
    A_inversa = inv(A)
    B_inversa = inv(B)
else
    fprintf("No se puede hacer la inversa si la matriz no es cuadrada.\n");
end
```

Salida:

```
A_transpuesta =

     3     5     1
     6     3     2
     7    10     2

A_inversa =

   -1.0000    1.0000    0.3333
    0.7692   -0.8462   -0.1282
    0.6923   -0.4615   -0.2821

B_transpuesta =

     3     5     1
     6     3     2
     7    10     2

B_inversa =

   -2.0000    0.2857    5.5714
    0.0000   -0.1429    0.7143
    1.0000    0.0000   -3.0000
```

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

Código:

```
if filas == cols
    det_A = det(A)
    det_B = det(B)
else
    fprintf("No se puede hacer el determinante si la matriz no es cuadrada.\n");
end

rango_A = rank(A)
rango_B = rank(B)
```

Salida:

```
det_A =

   39.0000


det_B =

     7
```

```
rango_A =

     3


rango_B =

     3
```

- **El producto de A y B (matricial y elemento a elemento)**

Código:

```
if filas == cols
    producto_matricial = A * B
else
    fprintf("No se puede hacer la multiplicación matricial si no coinciden las columnas de la primera matriz y las filas de la segunda.\n");
end

producto_elemento_a_elemento = A .* B
```

En este fragmento de código se usa la condición que se usa para los demás fragmentos de código que es filas == cols. A pesar de que la condición para que se puedan multiplicar 2 matrices es que el número de columnas de la primera sea igual al número de filas de la segunda, se usa la condición mencionada ya que al ser las dos matrices del mismo tamaño, solo se podría hacer la multiplicación en caso de que las matrices sean cuadradas.

Salida:

```
producto_matricial =

    52    69   111
    30    46    65
    75    87   159


producto_elemento_a_elemento =

    21    30    42
    25     6    50
     9    18     6
```

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

Código:

```
vector_fila_A = A';
vector_fila_A = vector_fila_A(:)'

vector_fila_B = B';
vector_fila_B = vector_fila_B(:)'
```

Salida:

```
vector_fila_A =

     7     5     6     5     2     5     9     9     3


vector_fila_B =

     3     6     7     5     3    10     1     2     2
```

- **Un vector columna obtenido concatenando la primera columna de cada una de las matrices**

Código:

```
vector_columna_A = A(:)
vector_columna_B = B(:)
```

Salida:

```
vector_columna_A =     vector_columna_B =

     7                      3
     5                      5
     9                      1
     5                      6
     2                      3
     9                      2
     6                      7
     5                     10
     3                      2
```

Ejercicio 4 - Representación gráfica en 3D.

Realice un script en Matlab que dibuje sobre el área $-5 \leq x, y \leq 5$ la superficie, la superficie en forma de malla y el contorno de la función:

$$z = y * sin\,(pi * x\ 10) + 5\ * cos\,((x\ 2 + y\ 2)/8) + cos(x + y)cos(3x - y)$$

En la misma figura dibuje en la parte superior y centrada la gráfica de la superficie, en la parte inferior izquierda la gráfica de la superficie en forma de malla y en la parte inferior derecha la gráfica del contorno. Además, añada la barra de color al contorno.

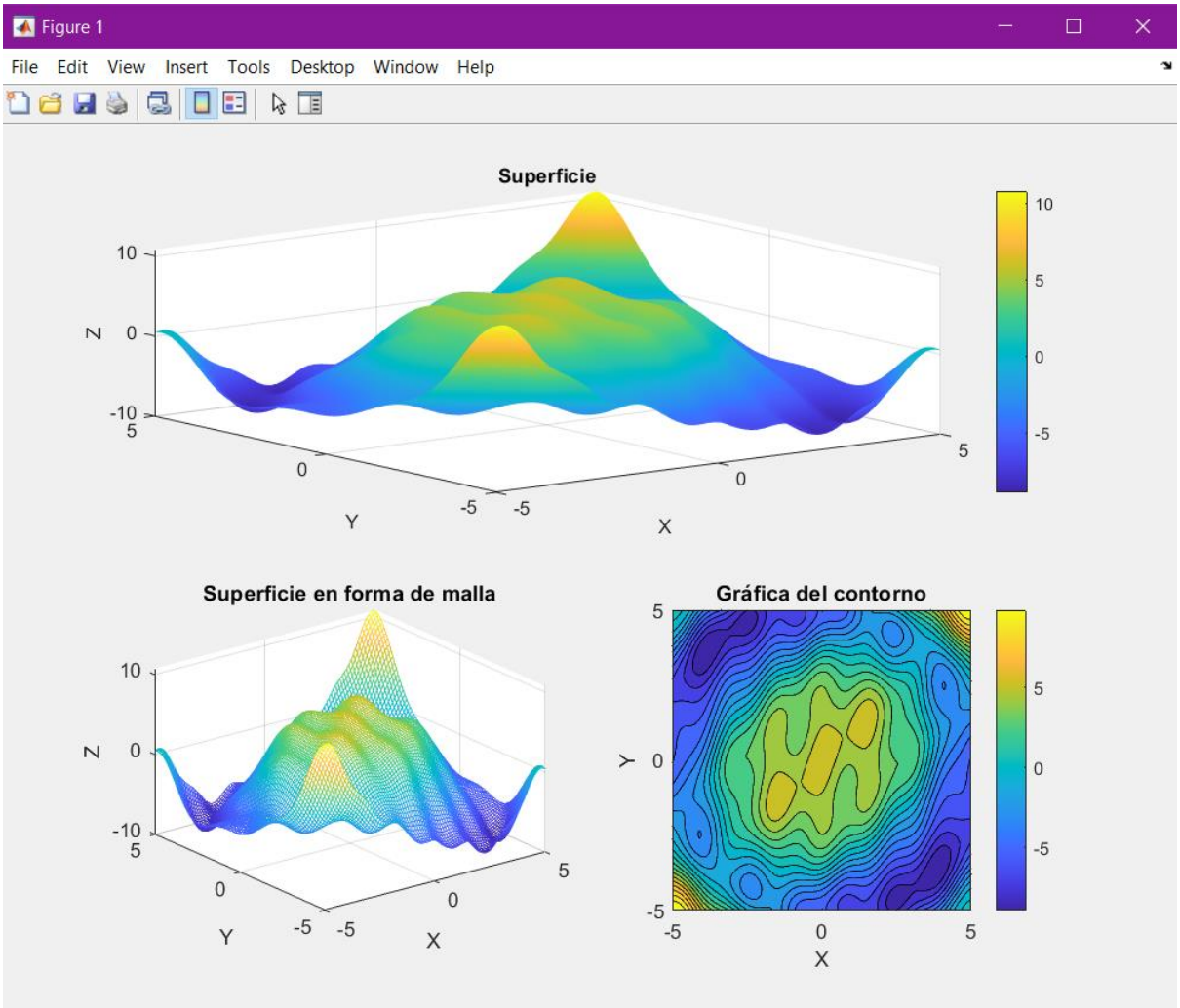
Deben añadirse etiquetas a los ejes, y un título a cada gráfica.

Código:

```
% Área de trabajo
x = linspace(-5, 5, 100); % 100 valores entre -5 y 5 para x
y = linspace(-5, 5, 100);
[X, Y] = meshgrid(x, y); %se crean matrices X e Y a partir de los valores anteriores

% Función Z
% cada vez que se multiplican matrices se hace la operación de punto por
% punto
Z = Y .* sin(pi * X / 10) + 5 * cos((X.^2 + Y.^2) / 8) + cos(X + Y) .* cos(3*X - Y);

% Se crea la figura
figure('Position', [150, 50, 800, 600]);  % Figura de 1200x800 píxeles

% Primera subgráfica: Gráfico de la superficie (parte superior centrada)
subplot(2, 2, [1, 2]); %subplot de 2 filas x 2 columnas. ocupa la posición 1 y 2, o sea las dos de arriba
surf(X, Y, Z); %Representación de la superficie 3D con los puntos de X e Y y los calculados de Z
xlabel('X');
ylabel('Y');
zlabel('Z');
title('Superficie');
shading interp; % Suaviza la superficie
colorbar; % Añade barra de color
view(3);  % Vista 3D estándar
```

```
% Segunda subgráfica: Superficie en forma de malla (parte inferior izquierda)
subplot(2, 2, 3); %subplot de 2 filas x 2 columnas. ocupa la posición 3, o sea la inferior izquierda
mesh(X, Y, Z); %Representación en forma de malla.
xlabel('X');
ylabel('Y');
zlabel('Z');
title('Superficie en forma de malla');
view(3);  % Vista 3D estándar

% Tercera subgráfica: Gráfico del contorno (parte inferior derecha)
subplot(2, 2, 4); %subplot de 2 filas x 2 columnas. ocupa la posición 4, o sea la inferior derecha
contourf(X, Y, Z, 20);  % Dibuja el contorno relleno con 20 niveles
xlabel('X');
ylabel('Y');
title('Gráfica del contorno');
colorbar;  % Añadir barra de color
```

Salida:



Ejercicio 5 – Transformadas de señales.

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

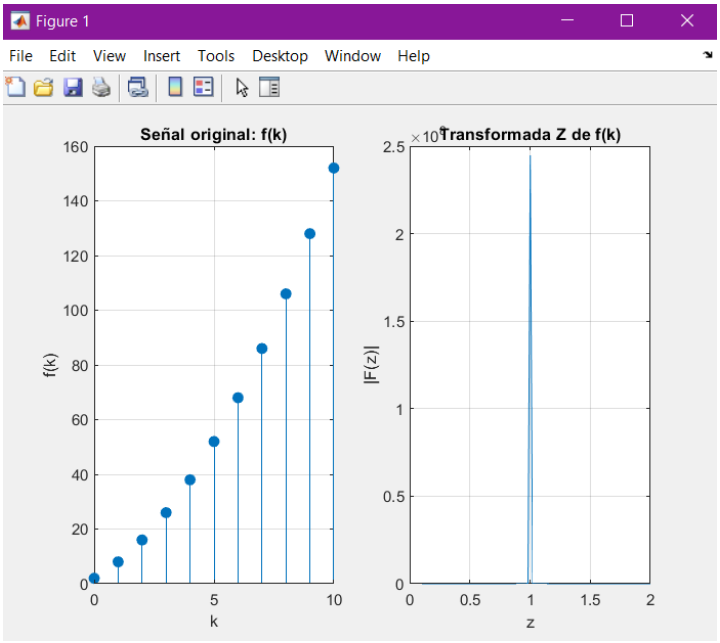
Código:

```
fprintf("Puesto 10 Laboratorio\n")
fprintf("EJERCICIO 5. Transformadas de señales. \n")

fprintf("\nApartado 1\n")

syms k z %se definen lss variables simbólicas k y z

% definición de la función f(k)
f_k = 2 + 5*k + k^2;

% transformada Z de f(k)
F_z = ztrans(f_k, k, z);

% resultado de la transformada
disp('Transformada Z de f(k):');
disp(F_z);
```

```
%representación gráfica
% valores numericos para k y resultado de la función de esos valores k
k_vals = 0:10;
f_vals = 2 + 5*k_vals + k_vals.^2;

%grafica de la señal original
figure;
subplot(1, 2, 1);
stem(k_vals, f_vals, 'filled');
title('Señal original: f(k)');
xlabel('k');
ylabel('f(k)');
grid on;

%grafica de la transformada Z (solo como magnitud para valores discretos)
z_vals = linspace(0.1, 2, 100);
F_vals = double(subs(F_z, z, z_vals));  % Evaluar la transformada Z para valores de z

subplot(1, 2, 2);
plot(z_vals, abs(F_vals));
title('Transformada Z de f(k)');
xlabel('z');
ylabel('|F(z)|');
grid on;
```

Salida:

```
Puesto 10 Laboratorio
EJERCICIO 5. Transformadas de señales.

Apartado 1
Transformada Z de f(k):
(2*z)/(z - 1) + (5*z)/(z - 1)^2 + (z*(z + 1))/(z - 1)^3
```



2. Obtenga la transformada z de la siguiente función: $f(k)$ = sen(k) · e −ak . Represente gráficamente, de nuevo, la señales original y transformada.

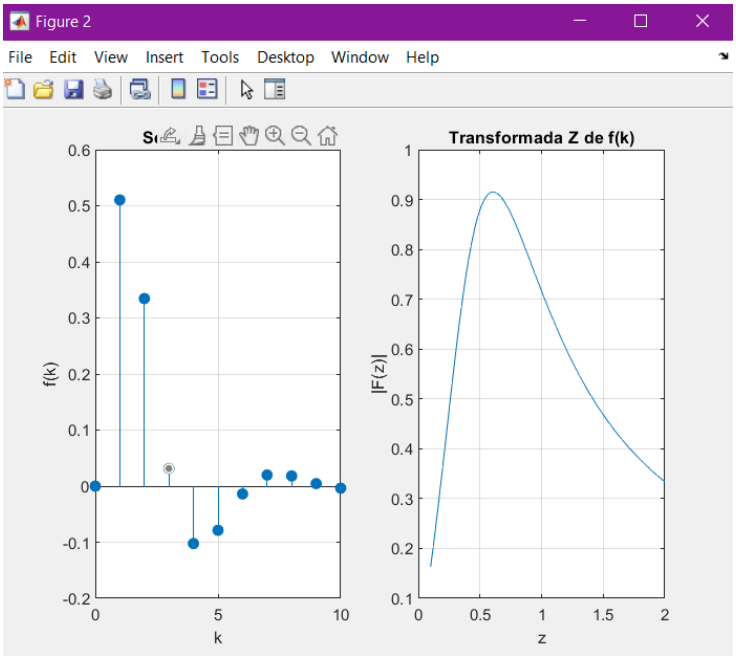
Código:

```
fprintf("\nApartado 2\n")

syms a %se define solo a porque k ya la definimos antes

% función f(k)
f_k = sin(k) * exp(-a*k);

%transformada Z de f(k)
F_z = ztrans(f_k, k, z);

%resultado de la transformada
disp('Transformada Z de f(k):\n');
disp(F_z);

%se define un valor constante de a y los valores para k y se calcula el
%resultado de la funcion
a_val = 0.5;
k_vals = 0:10;
f_vals = sin(k_vals) .* exp(-a_val * k_vals);
```

```
% grafica de la señal original
figure;
subplot(1, 2, 1);
stem(k_vals, f_vals, 'filled');
title('Señal original: f(k)');
xlabel('k');
ylabel('f(k)');
grid on;

% 100 valores equidistantes de z se usarán para graficar la transformada Z
z_vals = linspace(0.1, 2, 100);
% sustituye el valor de a por a_val y z por los de z_vals en la
% transformada Z y double convierte la expresión en valores numéricos para
% graficarla.
F_vals = double(subs(F_z, {a, z}, {a_val, z_vals}));

% grafica de la transformada Z
subplot(1, 2, 2);
plot(z_vals, abs(F_vals));
title('Transformada Z de f(k)');
xlabel('z');
ylabel('|F(z)|');
grid on;
```

Salida:

```
Apartado 2
Transformada Z de f(k):\n
(z*exp(a)*sin(1))/(exp(2*a)*z^2 - 2*cos(1)*exp(a)*z + 1)
```



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:

```
fprintf("\nApartado 3\n")

%función de transferencia discreta
num = [0.4 0 0];  % Numerador: 0.4*z^2
den = [1 -1 0.1 0.02];  % Denominador: z^3 - z^2 + 0.1*z + 0.02
sys = tf(num, den, -1);  % Resultado de función

% representación de respuesta al impulso
figure;
subplot(1,2,1)
impulse(sys);
title('Respuesta al impulso del sistema');
grid on;

%representación de respuesta al escalón
subplot(1,2,2)
step(sys);
title('Respuesta al escalón del sistema');
grid on;
```

Salida:

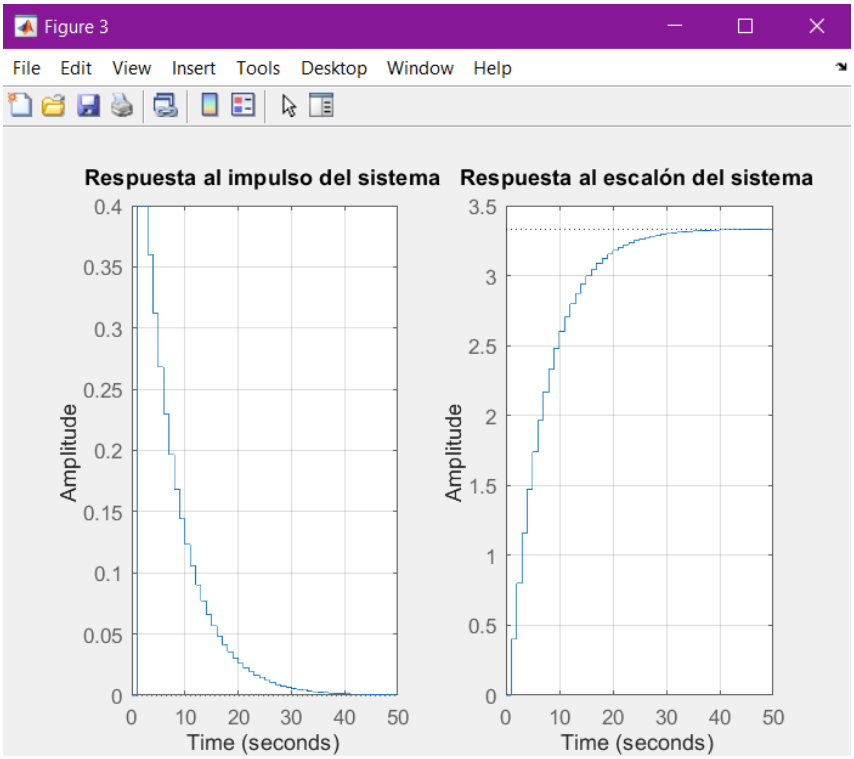
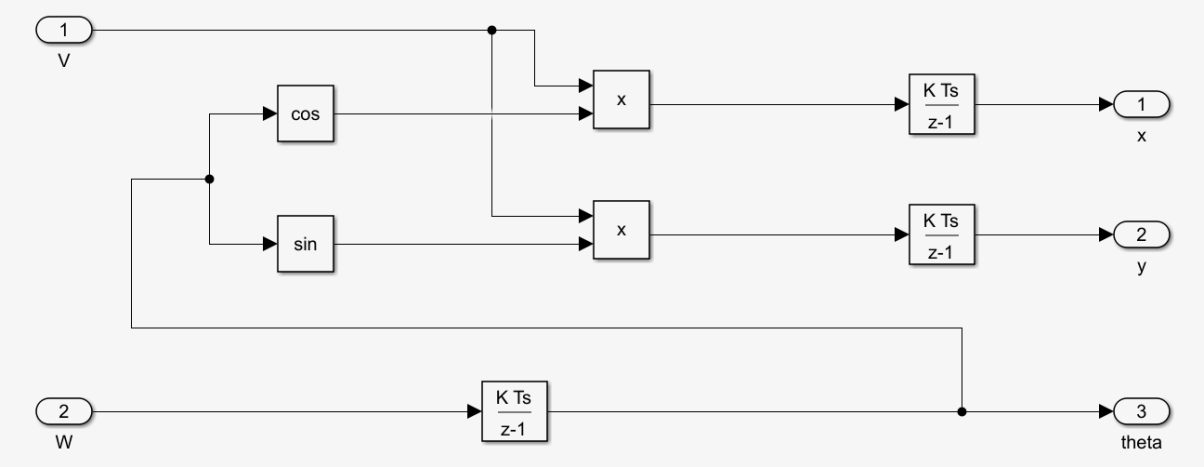


Figure 3
File Edit View Insert Tools Desktop Window Help
Respuesta al impulso del sistema
Amplitude
Time (seconds)
Respuesta al escalón del sistema
Amplitude
Time (seconds)

Segunda Parte - SIMULINK

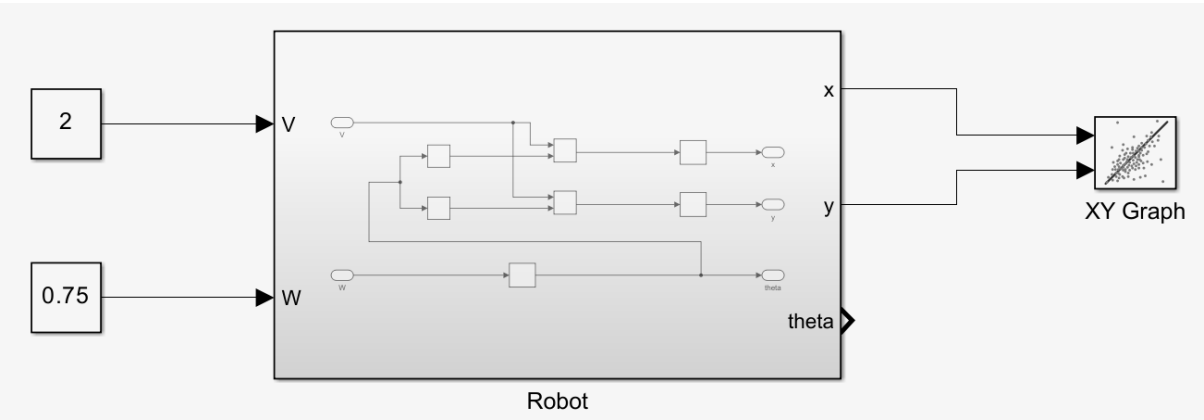
1. Implemente el modelo de la Figura 2 (todos los bloques utilizados pueden encontrarse en la librería estándar de Simulink).

Modelo:



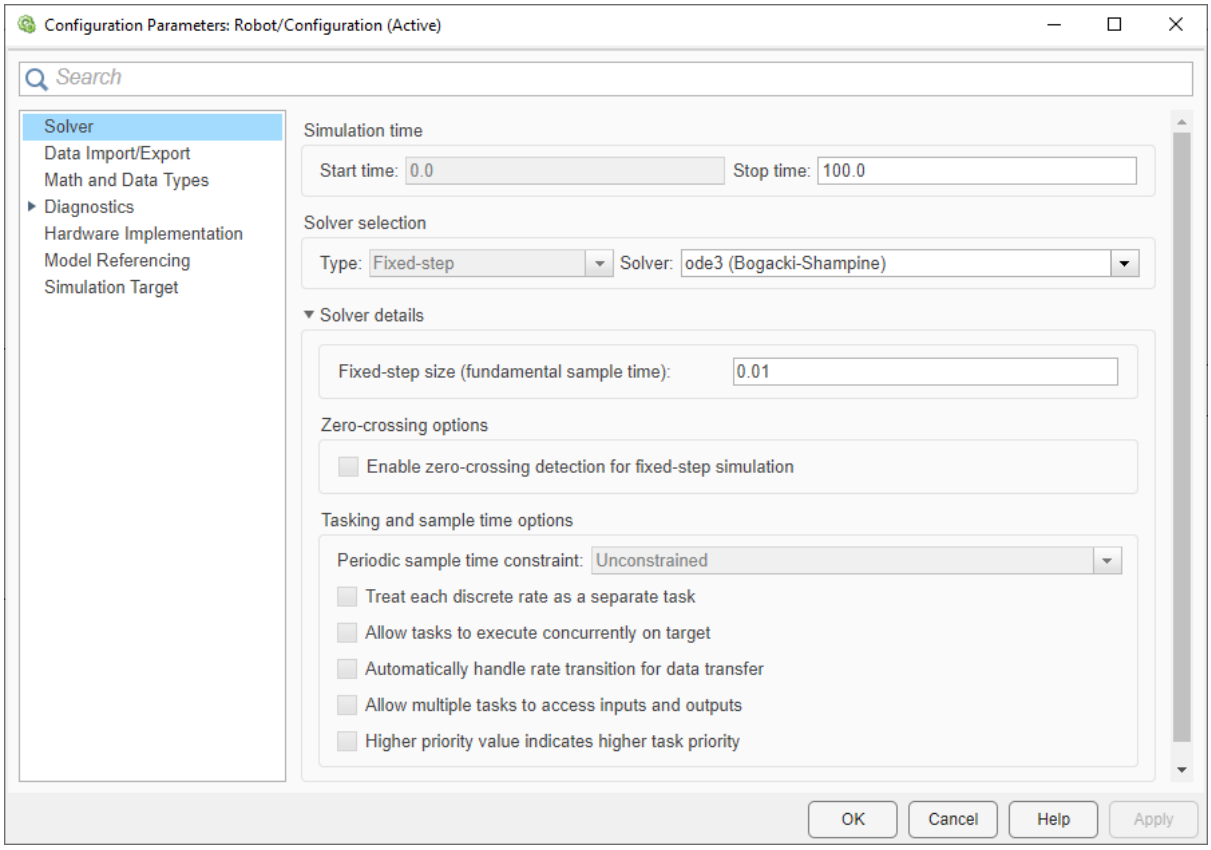
2. Una vez completado el apartado anterior, cree el subsistema mostrado en la Figura 1 y simule su funcionamiento con velocidad lineal y angular constante creando el sistema mostrado en la Figura 1.

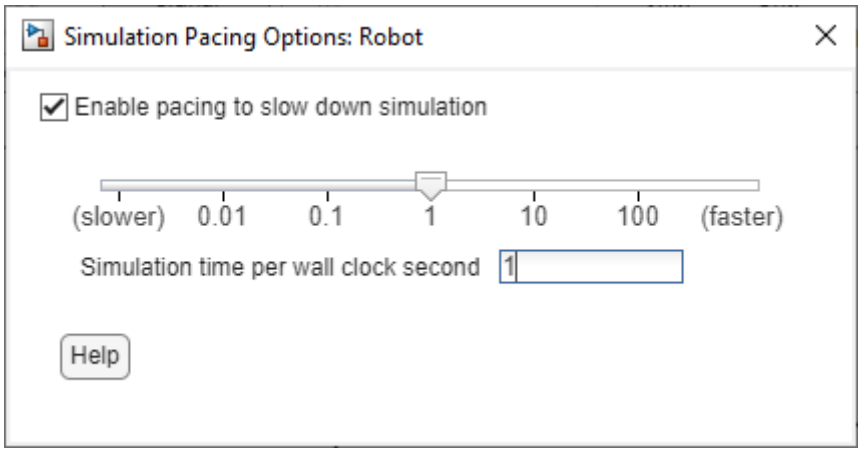
Subsistema:



3. Configure los parámetros de la simulación (menú "Simulation/Model Simulation Parameters" y menú Simulation/Pacing options) de acuerdo con la Figura 4.

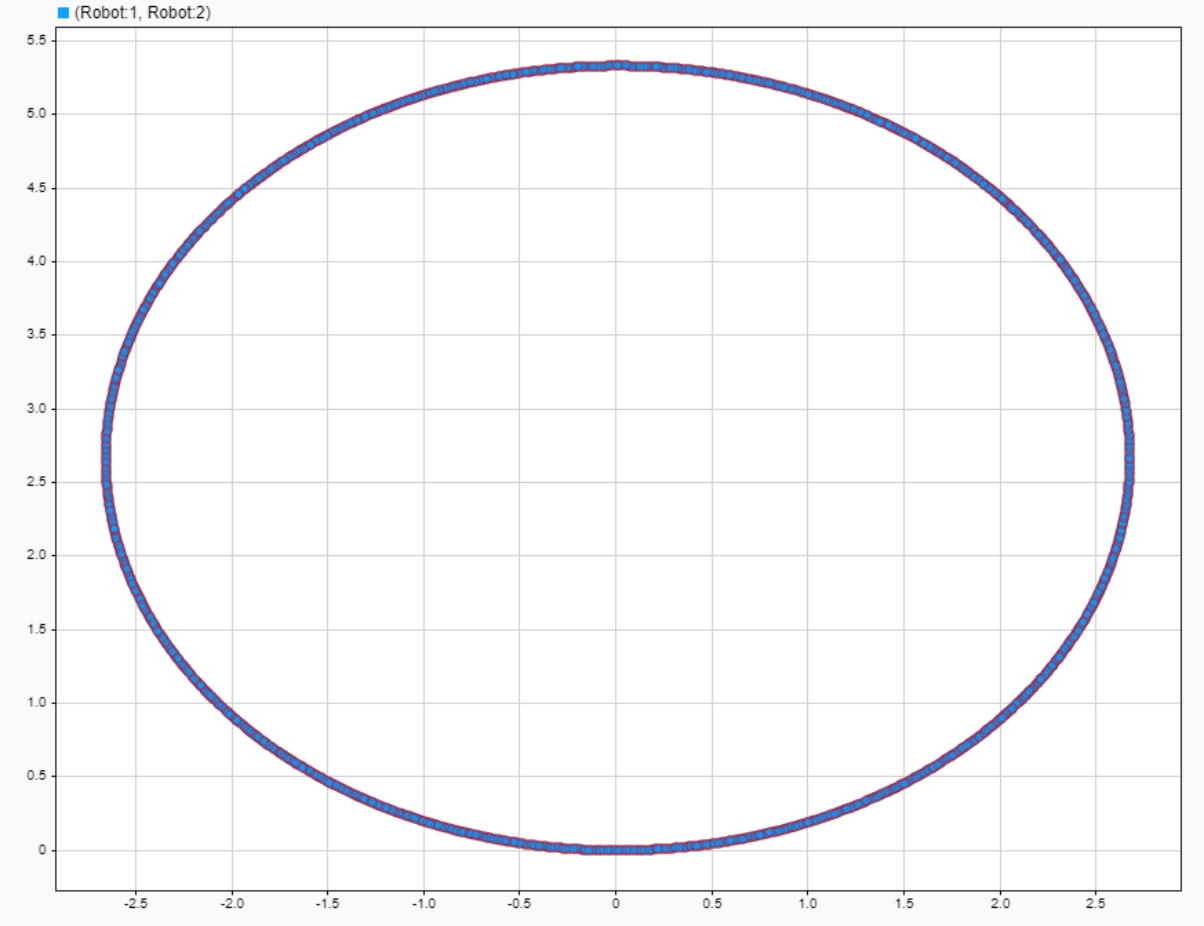
Configuración de parámetros:





3. Visualizando la gráfica generada por el módulo XY Graph, compruebe que el funcionamiento del robot se ajusta a lo esperado.

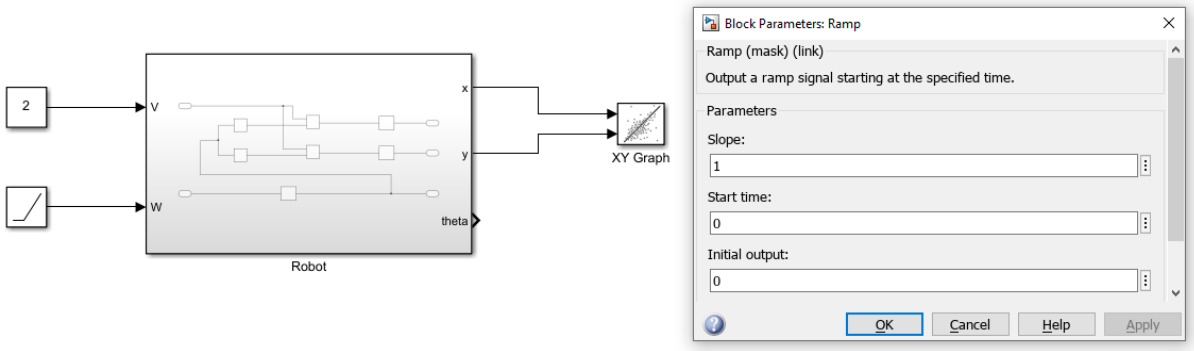
Gráfica XY con velocidad lineal y angular constantes:



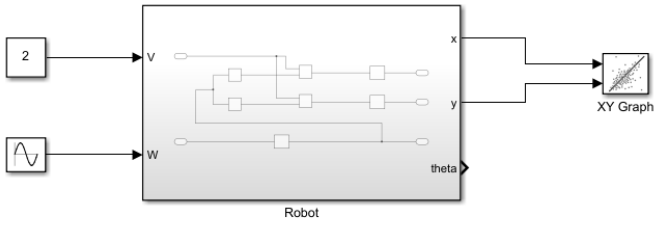
El dibujo de la gráfica del movimiento generado del robot se ajusta con las velocidades constantes insertadas como parámetro. Se muestra un círculo debido a que el robot gira mientras se le atribuye una velocidad lineal constante.

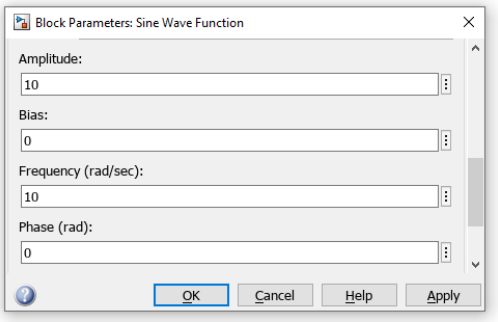
4. Realice de nuevo la simulación con velocidades angulares no constantes (estas fuentes están disponibles en la librería de Simulink/sources):

a-Rampa



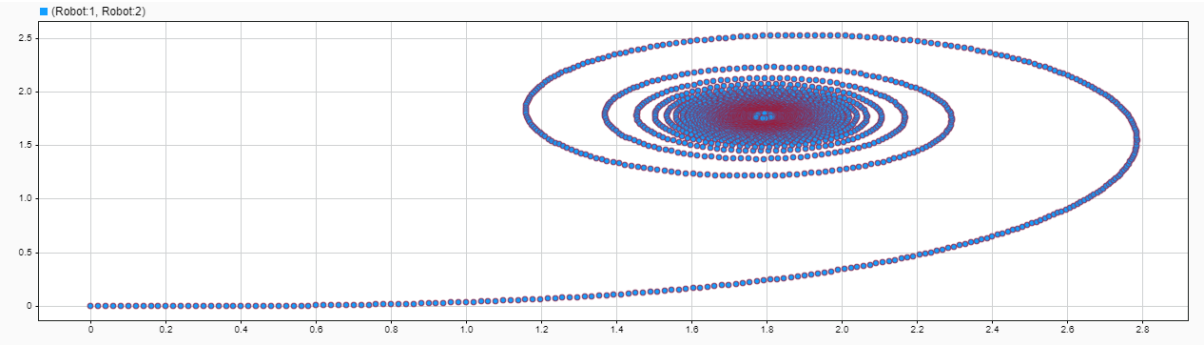
b-Variación Sinusoidal





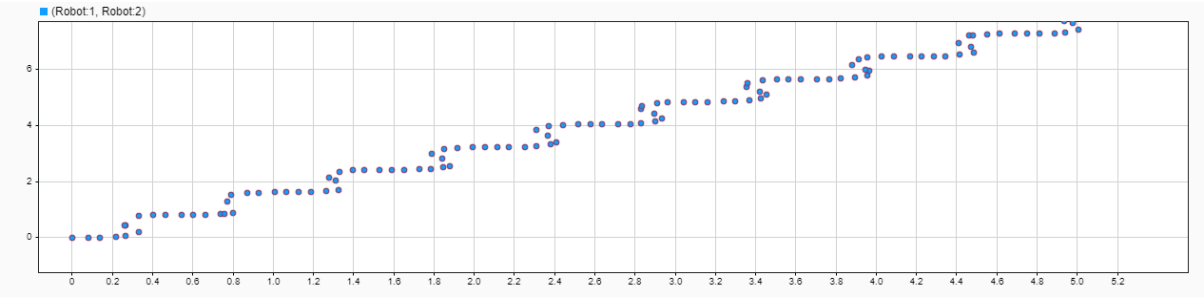
5. Estudie de nuevo las trayectorias realizadas por el robot y compruebe que se ajustan al comportamiento esperado.

a-Rampa



A medida que la velocidad angular aumenta, los círculos se van haciendo más pequeños lo que produce una forma de espiral, se ajusta con la gráfica obtenida.

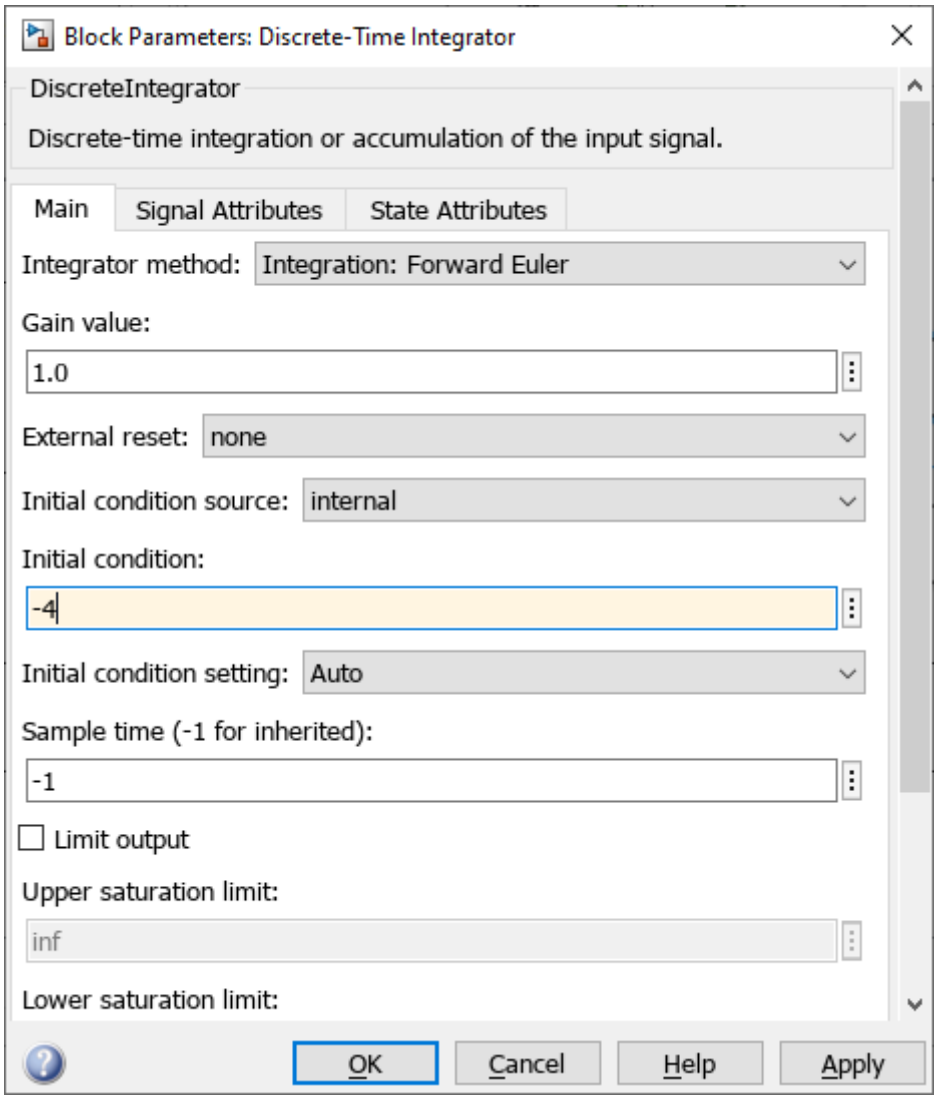
b-Variación Sinusoidal



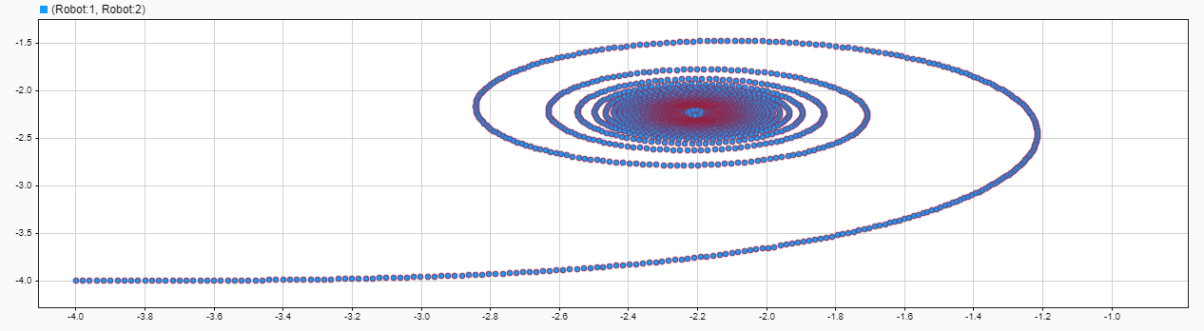
En este caso, la velocidad angular puede ser negativa. Debido a ello, la forma descrita no es una curva constante, ahora la gráfica son curvas de izquierda a derecha.

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

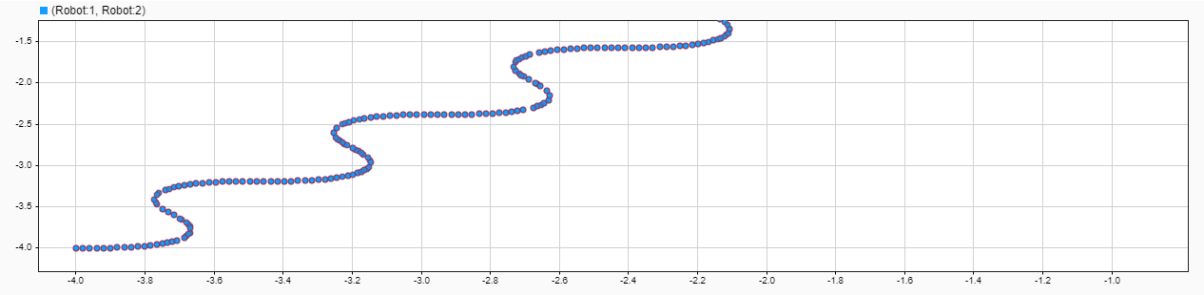
El componente Discrete-Time Integrator es el que genera las cordenadas para la gráfica final. Es el encargado de tomar el resultado de las operaciones trigonométricas en tiempo discreto y realizar la acumulación de los resultados en el tiempo. El valor inicial asignado para empezar a realizar la acumulación de señales es el 0 para x e y por defecto. Cambiando ese valor por -4 para los integradores de x e y se consigue que la simulación comience en (-4, -4).



a-Rampa



b-Variación Sinusoidal



Como se puede observar, las gráficas son iguales con la excepción del punto de partida.