SEÑALES ELEMENTALES

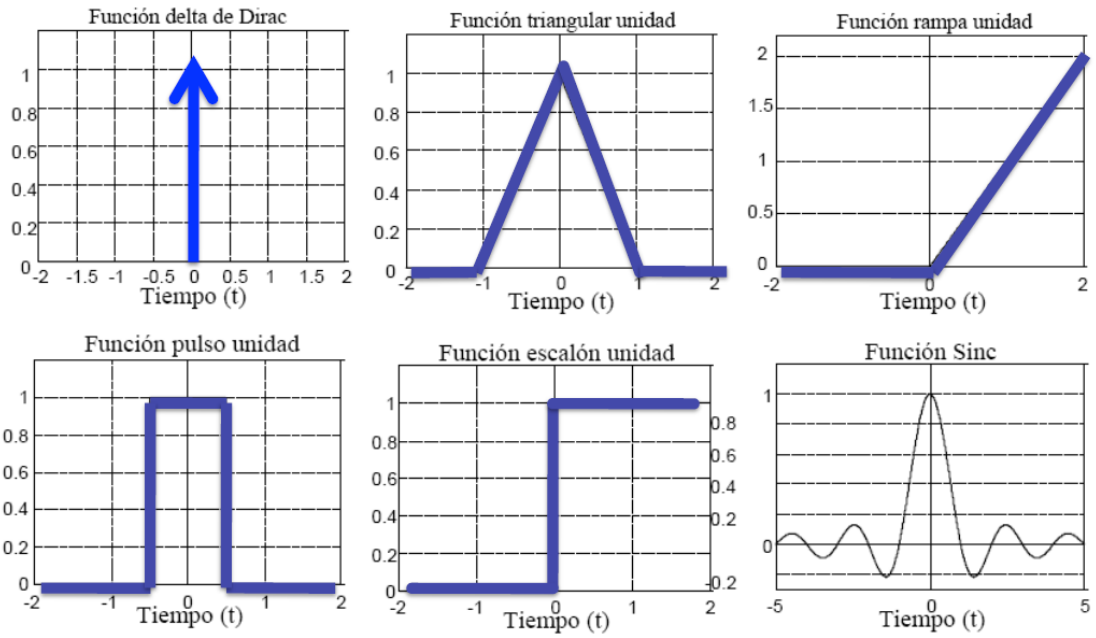
NO. DE PRACTICA: 2(DOS)	Fecha de entrega: 20 DE SEPTIEMBRE
INTEGRANTES DE EQUIPO: 19 NAVIL PINEDA RUGERIO	Practica: ~~Individual~~ En equipo

OBJETIVO: QUE EL ALUMNO REALICE GRÁFICOS Y PROGRAMAS SENCILLOS USANDO MATLAB.

MATERIAL Y EQUIPO: COMPUTADORA CON MATLAB

DESARROLLO:

EJERCICIO 1: ESCRIBA UN LIVESCRIPT EN EL QUE SE IMPLEMENTEN LAS SIGIENTES FUNCIONES:



Para este primer ejercicio se escribe la forma matemática de cada una de las funciones, utilizando sentencias, ya que es una función por partes.

FUNCIÓN DELTA DE DIRAC.

EXPRESIÓN MATEMÁTICA.

$$\delta(t) = 0, t \neq 0$$
$$\int_{-\infty}^{\infty} \boldsymbol{\delta}(t)dt = 1$$

PROGRAMACIÓN.

```
t = -5:0.1:5;
dirac_t = zeros(size(t));
for i = 1:length(t)
    dirac_t(i) = dirac(t(i));
end
plot(t,dirac_t, 'c'), xlabel('t'), ylabel('sinc(t)'), title('funcion delta de dirac');

function res = dirac(t)
    if t ~= 0
        res = 0;
    else
        res = 1;
    end
end
```

FUNCIÓN TRIANGULAR.

Para esta función se utilizan dos funciones más, la función rampa y la función escalón, ya que una depende de la otra.

EXPRESIÓN MATEMÁTICA:

- FUNCIÓN RAMPA: $r(t) = \begin{cases} tu(t) & t > 0 \\ 0 & t \leq 0 \end{cases}$
- FUNCIÓN ESCALON: $u(t) = \begin{cases} 0 & t < 0 \\ 1 & t > 0 \end{cases}$
- FUNCIÓN TRIANGULAR: $r(t) = \begin{cases} r(t+1) - 2r(t) + r(t-1) & -1 < t < 1 \\ 0 & resto \end{cases}$

PROGRAMACIÓN.

```
tri_t = zeros(size(t));
for i = 1:length(t)
    tri_t(i) = tri(t(i));
end
plot(t,tri_t, 'r'), xlabel('t'), ylabel('tri(t)'), title('funcion triangular');

% funcion escalon
function res = u(t)
    if t < 0
        res = 0;
    else
        res = 1;
    end
end

% funcion rampa
function res = r(t)
    if t > 0
        res = t * u(t);
    else
        res = 0;
    end
end

% funcion triangular
function res = tri(t)
    if t > (-1) & t < 1
        res = r(t+1) - 2*r(t) + r(t-1);
    else
        res = 0;
    end
end
```

FUNCIÓN RAMPA UNIDAD.

EXPRESIÓN MATEMÁTICA.

$$r(t) = \begin{cases} tu(t) & t > 0 \\ 0 & t \leq 0 \end{cases}$$

PROGRAMACIÓN.

NOTA: Esta función ya está definida en el ejercicio anterior.

```
r_t = zeros(size(t));
for i = 1:length(t)
    r_t(i) = r(t(i));
end
plot(t,r_t, 'c'), xlabel('t'), ylabel('r(t)'), title('funcion rampa');

% funcion rampa
function res = r(t)
    if t > 0
        res = t * u(t);
    else
        res = 0;
    end
end
```

FUNCIÓN PULSO UNIDAD.

EXPRESIÓN MATEMÁTICA.

$$\prod(t) = \begin{cases} \boldsymbol{u}\left(t+\frac{1}{2}\right) - u\left(t-\frac{1}{2}\right) & -\frac{\mathbf{1}}{\mathbf{2}} < t < \frac{\mathbf{1}}{\mathbf{2}} \\ 0 & resto \end{cases}$$

PROGRAMACIÓN.

```
p_t = zeros(size(t));
for i = 1:length(t)
    p_t(i) = p(t(i));
end
plot(t,p_t, 'g'), xlabel('t'), ylabel('p(t)'), title('funcion pulso');

% función pulso
function res = p(t)
    if t > (-1/2) & t < (1/2)
        res = u(t+(1/2)) - u(t-(1/2));
    else
        res = 0;
    end
end
```

FUNCIÓN ESCALÓN UNIDAD.

EXPRESIÓN MATEMÁTICA.

$$u(t) = \begin{cases} 0 & t < 0 \\ 1 & t > 0 \end{cases}$$

PROGRAMACIÓN.

```
u_t = zeros(size(t));
for i = 1:length(t)
    u_t(i) = u(t(i));
end
plot(t,u_t, 'b'), xlabel('t'), ylabel('u(t)'), title('escalon unidad');

% funcion escalon
function res = u(t)
    if t < 0
        res = 0;
    else
        res = 1;
    end
end
```

FUNCIÓN SINC.

EXPRESIÓN MATEMÁTICA.

$$sinc(t) = \frac{\sin(\pi t)}{\pi t}$$

PROGRAMACIÓN.

```
sinc_t = zeros(size(t));
for i = 1:length(t)
    sinc_t(i) = sinc(t(i));
end
plot(t,sinc_t, 'b'), xlabel('t'), ylabel('sinc(t)'), title('funcion sinc');

% funcion sinc
function res = sinc(t)
    res = sin(pi*t) / (pi*t);
end
```

EJERCICIO 2: DESARROLLE LAS TRANSFORMACIONES DECRITAS EN LA IMAGEN PARA CADA UNA DE LAS FUNCIONES DESCRITAS ANTERIORMENTE

- Desplazamiento en el tiempo:
  - Señal adelantada y retrasada en el tiempo
    $x(t-t_0)$, desplazamiento a la derecha. (Retrasada)
    $x(t+t_0)$, desplazamiento a la izquierda. (Adelantada)
- Reflexión:
  Inversión en el tiempo de $x(t) = x(-t)$
- Cambios lineales de escala en la variable independiente:
  - Compresión en el tiempo de $x(t) = x(2t)$
  - Dilatación en el tiempo de $x(t) = x(t/2)$

A partir de las funciones creadas en el ejercicio anterior, se les aplica una transformación, las cuales son programadas como funciones que reciben como parámetros la función que se quiere transformar (Dirac, Sinc, etc.), los valores de tiempo (t) para ser evaluados, y si es necesario un parámetro adicional.

DESPLAZAMIENTO EN EL TIEMPO.

RETRASO.

Esta función recibe como parámetros: la función que se quiere aplicar, la constante de retraso, es decir el valor que nos indica qué tan desplazada será la función, y finalmente los valores de tiempo para evaluar la función (pasados como arreglo). En el cuerpo de la función a cada valor del tiempo se le resta la constante de retraso, y se evalúa en la función a transformar, finalmente se devuelve un arreglo con los valores de la señal ya retrasada en el tiempo.

```
function res_t = retraso(funcion, f_des, t)
    res_t = zeros(size(t));
    for i = 1:length(t)
        val = t(i)-f_des;
        res_t(i) = funcion(val);
    end
end
```

ADELANTO.

Esta función recibe los mismos parámetros que la función anterior, y realiza casi el mismo procedimiento, con la única diferencia de que la contante al ser ahora de adelanto no se resta, sino que se suma a cada valor del tiempo.

```
function ad_t = adelanto(funcion, f_des, t)
    ad_t = zeros(size(t));
    for i = 1:length(t)
        val = t(i)+f_des;
        ad_t(i) = funcion(val);
    end
end
```

REFLEXIÓN.

Para esta función no se necesita ninguna constante adicional, por lo que solo se recibe la función a transformar y los valores de tiempo, en el cuerpo de la función multiplica el valor del tiempo por -1, de tal forma que los valores positivos se vuelvan negativos, y los valores negativos se vuelvan positivos, dando un efecto de una función espejo.

```
function inv_t = reflexion(funcion, t)
    inv_t = zeros(size(t));
    for i = 1:length(t)
        val = t(i) * (-1);
        inv_t(i) = funcion(val);
        %inv_t = inv_t * (-1);
    end
end
```

CAMBIOS LINEALES DE ESCALA EN LA VARIABLE INDEPENDIENTE.

COMPRESIÓN EN EL TIEMPO.

Para la función de compresión se pasa como parámetro la función a transformar, los valores de tiempo, y a cada uno de esos valores se le multiplica por 2, y se evalúa.

```
function comp_t = compresion(funcion, t)
    comp_t = zeros(size(t));
    for i = 1:length(t)
        val = t(i) * 2;
        comp_t(i) = funcion(val);
    end
end
```

DILATACIÓN EN EL TIEMPO.

Esta función realiza el mismo procedimiento que la función anterior, pero el valor del tiempo en lugar de ser multiplicado es dividido entre dos.

```
function dil_t = dilatacion(funcion, t)
    dil_t = zeros(size(t));
    for i = 1:length(t)
        val = t(i) / 2;
        dil_t(i) = funcion(val);
    end
end
```

Para graficar las transformaciones se hace en una sola gráfica, para cada función.

Transformaciones a Dirac:

```
des_d = retraso(@dirac, 3, t);
des_i = adelanto(@dirac, 3, t);
ref_t = reflexion(@dirac,t);
comp_t = compresion(@dirac, t);
dil_t = dilatacion(@dirac, t);
plot(t,des_d, 'c'), xlabel('t'), title('retraso a dirac');
hold on;
plot(t,des_i, 'r'), xlabel('t'), title('adelanto a dirac');
plot(t,ref_t, 'g'), xlabel('t'), title('reflexion a dirac');
plot(t,comp_t, 'b'), xlabel('t'), title('compresion a dirac');
plot(t,dil_t, 'k'), xlabel('t'), title('dilatacion a dirac');
hold off;
```

Transformaciones a triangular

```
des_d = retraso(@tri, 3, t);
des_i = adelanto(@tri, 3, t);
ref_t = reflexion(@tri,t);
comp_t = compresion(@tri, t);
dil_t = dilatacion(@tri, t);
plot(t,des_d, 'c'), xlabel('t'), title('retraso a triangular');
hold on;
plot(t,des_i, 'r'), xlabel('t'), title('adelanto a triangular');
plot(t,ref_t, 'g'), xlabel('t'), title('reflexion a triangular');
plot(t,comp_t, 'b'), xlabel('t'), title('compresion a triangular');
plot(t,dil_t, 'k'), xlabel('t'), title('dilatacion a triangular');
hold off;
```

Transformaciones a rampa

```
des_d = retraso(@r, 3, t);
des_i = adelanto(@r, 3, t);
ref_t = reflexion(@r,t);
comp_t = compresion(@r, t);
dil_t = dilatacion(@r, t);
plot(t,des_d, 'c'), xlabel('t'), title('retraso a rampa');
hold on;
plot(t,des_i, 'r'), xlabel('t'), title('adelanto a rampa');
plot(t,ref_t, 'g'), xlabel('t'), title('reflexion a rampa');
plot(t,comp_t, 'b'), xlabel('t'), title('compresion a rampa');
plot(t,dil_t, 'k'), xlabel('t'), title('dilatacion a rampa');
hold off;
```

Transformaciones a pulso

```
des_d = retraso(@p, 3, t);
des_i = adelanto(@p, 3, t);
ref_t = reflexion(@p,t);
comp_t = compresion(@p, t);
dil_t = dilatacion(@p, t);
plot(t,des_d, 'c'), xlabel('t'), title('retraso a pulso');
hold on;
plot(t,des_i, 'r'), xlabel('t'), title('adelanto a pulso');
plot(t,ref_t, 'g'), xlabel('t'), title('reflexion a rampa');
plot(t,comp_t, 'b'), xlabel('t'), title('compresion a pulso');
plot(t,dil_t, 'k'), xlabel('t'), title('dilatacion a pulso');
hold off;
```

Transformaciones a escalón

```
des_d = retraso(@u, 3, t);
des_i = adelanto(@u, 3, t);
ref_t = reflexion(@u,t);
comp_t = compresion(@u, t);
dil_t = dilatacion(@u, t);
plot(t,des_d, 'c'), xlabel('t'), title('retraso a escalon');
hold on;
plot(t,des_i, 'r'), xlabel('t'), title('adelanto a escalon');
plot(t,ref_t, 'g'), xlabel('t'), title('reflexion a escalon');
plot(t,comp_t, 'b'), xlabel('t'), title('compresion a escalon');
plot(t,dil_t, 'k'), xlabel('t'), title('dilatacion a escalon');
hold off;
```

Transformaciones a sinc

```
des_d = retraso(@sinc, 3, t);
des_i = adelanto(@sinc, 3, t);
ref_t = reflexion(@sinc,t);
comp_t = compresion(@sinc, t);
dil_t = dilatacion(@sinc, t);
plot(t,des_d, 'c'), xlabel('t'), title('retraso a sinc');
hold on;
plot(t,des_i, 'r'), xlabel('t'), title('adelanto a sinc');
plot(t,ref_t, 'g'), xlabel('t'), title('reflexion a sinc');
plot(t,comp_t, 'b'), xlabel('t'), title('compresion a sinc');
plot(t,dil_t, 'k'), xlabel('t'), title('dilatacion a sinc');
hold off;
```

EJERCICIO 3: REALICE REPRESENTACION GRAFICA Y TABULAR DE LAS SIGUIENTES SEÑALES.

$$x[n] = [\mathbf{1}, \mathbf{2}, \mathbf{8}, \mathbf{15}, \mathbf{3}, \mathbf{2}, \mathbf{8}, \mathbf{16}, \mathbf{24}, \mathbf{8}, \mathbf{3}]$$

$$\boldsymbol{y}[n] = [\mathbf{15}, \mathbf{2}, \mathbf{5}, \mathbf{3}, \mathbf{3}, \mathbf{8}, \mathbf{2}, \mathbf{1}, \mathbf{4}, \mathbf{2}, \mathbf{7}]$$

Para la representación gráfica de las señales, primero se define un arreglo con los valores que se indican anteriormente, después se define otro arreglo con valores de 0 a N, donde N es el tamaño de la señal menos 1, en este caso 10, para que la señal sea graficada como coordenadas. Se utiliza la función plot para verla de forma “continua”, y steam para la representación tabular.

```
% primera señal
x = [1,2,8,15,3,2,8,16,24,8,3];
% segunda señal
y =[15,2,5,3,3,8,2,1,4,2,7];

% arreglo de valores de 0 a n para evaluar graficar la señal
n = 0:length(x)-1;

% representación gráfica.
plot(n, x);
hold on;
% representación tabular
stem(n, x, 'marker', 'o', 'linewidth', 1.5);
title('representación tabular y gráfica de x[n]');
grid on;
```

```
hold off;

% representación gráfica.
plot(n, y);
hold on;
% representación tabular
stem(n, y, 'marker', 'o', 'linewidth', 1.5);
title('representación tabular y gráfica de y[n]');
grid on;
hold off;
```

SUMA

$$\boldsymbol{m}[n] = \boldsymbol{x}[\boldsymbol{n}] + \boldsymbol{y}[\boldsymbol{n}]$$

Para esta operación Matlab permite hacer la suma directamente con los arreglos, utilizando el operador **+.** Y de la misma forma, utilizando plot y steam se grafica la señal.

```
suma = x + y;
% representación gráfica.
plot(n, suma);
hold on;
% representación tabular
stem(n, suma, 'marker', 'o', 'linewidth', 1.5);
title('representación tabular y gráfica de la suma');
grid on;
hold off;
```

MULTIPLICACIÓN

$$\boldsymbol{m}[n] = \boldsymbol{x}[\boldsymbol{n}]\boldsymbol{y}[\boldsymbol{n}]$$

Para esta operación se debe utilizar el operador .* que realiza una multiplicación elemento por elemento de un arreglo, en este caso de la señal, y se grafican con plot y steam.

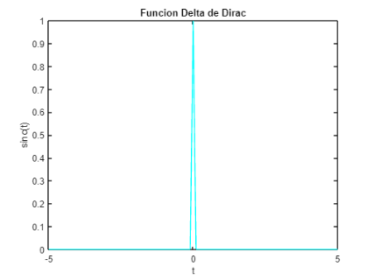
```
mult = x .* y;
% representación gráfica.
plot(n, mult);
hold on;
% representación tabular
stem(n, mult, 'marker', 'o', 'linewidth', 1.5);
title('representación tabular y gráfica de la multiplicación');
grid on;
hold off;
```

RESULTADOS:

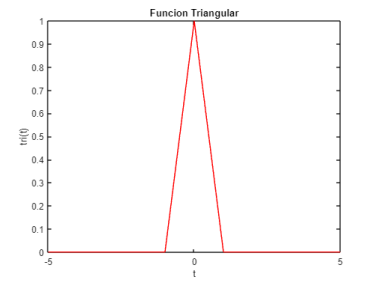
Ejercicio 1.

A continuación, se presentan los resultados de graficar cada una de las funciones explicadas en el desarrollo del presente documento.

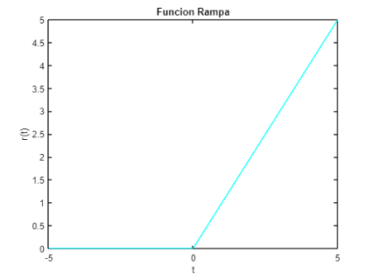
FUNCIÓN DELTA DE DIRAC.



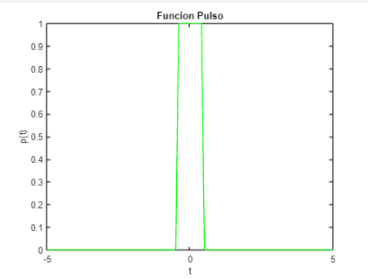
FUNCIÓN TRIANGULAR.



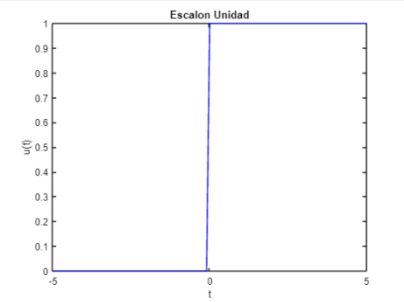
FUNCIÓN RAMPA UNIDAD.



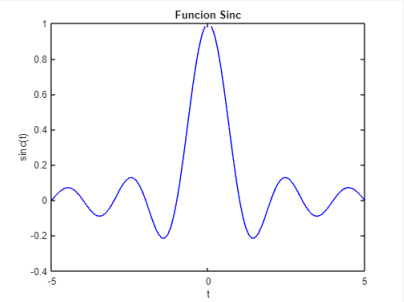
FUNCIÓN PULSO.



FUNCIÓN ESCALÓN UNIDAD.



FUNCIÓN SINC.

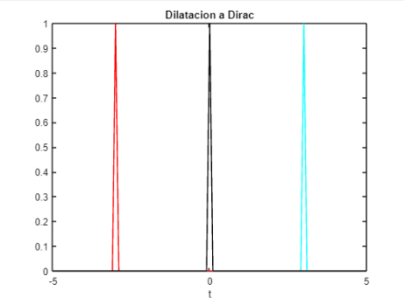


Ejercicio 2.

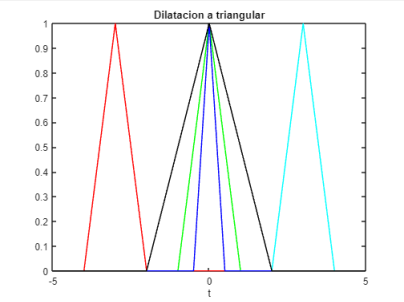
Se presentan los resultados de cada transformación en una sola gráfica para cada función.

LEYENDA: CYAN=RETRASO, ROJO=ADELANTO, VERDE=REFLEXIÓN, AZUL=COMPRESIÓN, NEGRO=DILATACIÓN.

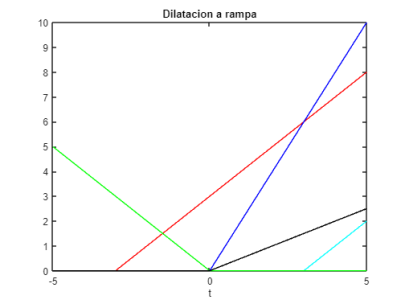
FUNCIÓN DELTA DE DIRAC.



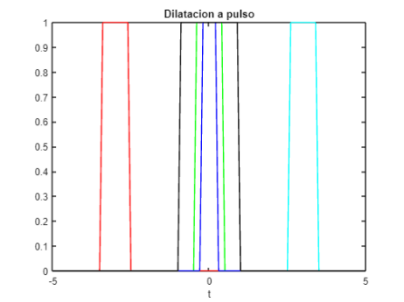
FUNCIÓN TRIANGULAR.



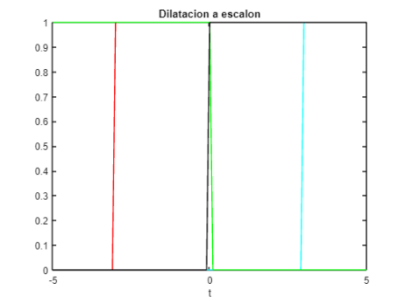
FUNCIÓN RAMPA UNIDAD.



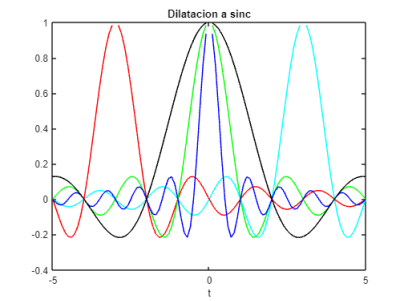
FUNCIÓN PULSO.



FUNCIÓN ESCALÓN UNIDAD.



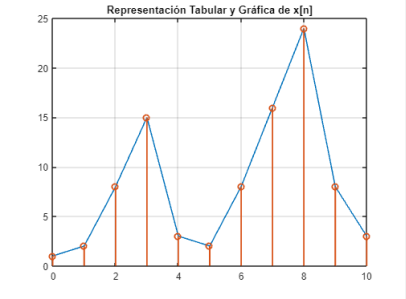
FUNCIÓN SINC.



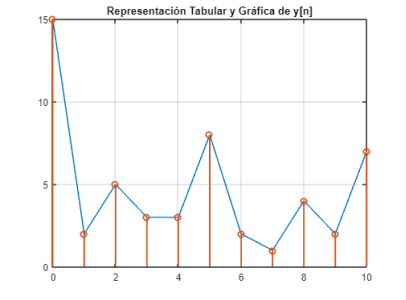
Ejercicio 3.

Se presentan las gráficas de las señales X y Y, y las gráficas de las operaciones suma y multiplicación.

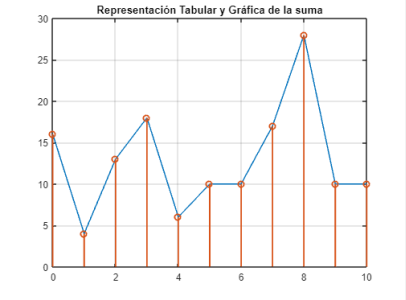
REPRESENTACIÓN GRÁFICA Y TABULAR DE SEÑAL X[N].



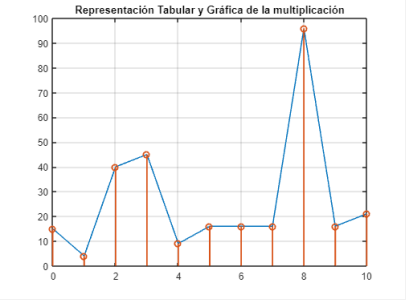
REPRESENTACIÓN GRÁFICA Y TABULAR DE SEÑAL Y[N].



REPRESENTACIÓN GRÁFICA Y TABULAR DE SUMA.



REPRESENTACIÓN GRÁFICA Y TABULAR DE MULTIPLICACIÓN.



CONCLUSIÓN:

La representación de señales en Matlab resulta práctica, ya que utilizando solo una función matemática, o un vector se puede representar el comportamiento de una señal, además, permite aplicar operaciones y transformaciones tan solo utilizando los operadores que ya tiene incluidos el lenguaje de programación, o bien creando funciones desde cero o desde un toolbox. Así como mostrar de forma gráfica las señales y sus transformaciones, con el fin de observar el efecto que cada una tiene sobre la señal, y poder analizarla visualmente.

FECHA FINAL DE ENTREGA: **lunes, 18 de septiembre de 2023**