



Instituto Politécnico Nacional

Unidad Profesional Interdisciplinaria de Biotechnología



Materia:

PROCESAMIENTO DIGITAL DE BIOSEÑALES E IMÁGENES.

Practica 1

Generación y Muestreo de Señales con Matlab y Simulink

Profesores:

Venegas Anaya Darinel

González Alvarado Alejandro
Carlos

Iturbe Gil Carlos

5MV2

México, DF. a 23 de Agosto del 2017



Objetivo General

Representar y reconstruir señales de tiempo continuo y tiempo discreto a través del muestreo de una señal y procesar las señales como funciones matemáticas.

Objetivos Específicos

Representar señales o funciones de tiempo continuo y discreto mediante la herramienta de cómputo Matlab®.

Realizar la discretización de una señal mediante la Frecuencia de Muestreo

Reconstruir funciones discontinuas como función pulso o triangular a partir de funciones especiales como función escalón y función rampa.

Reconstruir funciones definidas por intervalos mediante el proceso de ventana

Utilizar la herramienta Simulink® de Matlab® para la reconstrucción y representación de señales

Implementar un programa de Matlab para la generación de gráficas de la representación de una señal de tiempo continuo o tiempo discreto.

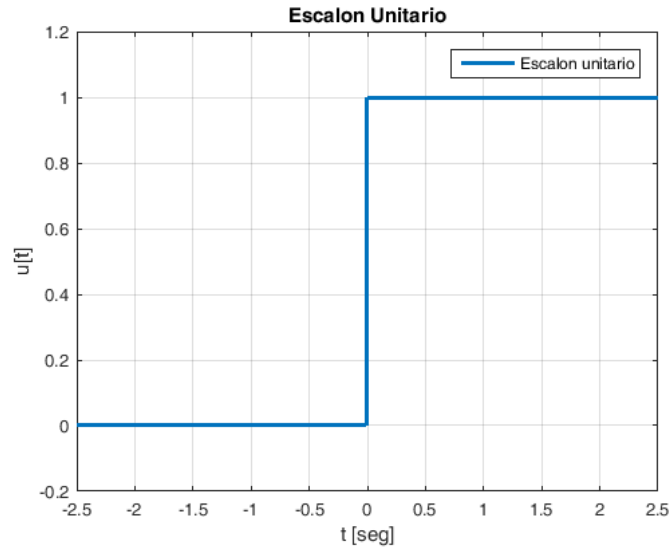
Desarrollo Experimental

EXPERIMENTO 1

CODIGO PARA LA GENERACION Y MUESTREO DE UN ESCALON

```
clc
clear
Fs=100;                                % Numero de Muestras Por Segundo
t=[-2.5:1/Fs:2.5];                    % Tiempo de Muestreo
Lt=length(t);                          % longitud del vector t
for k=1:Lt
    if t(k)<0
        u(k)=0;
    else
        u(k)=1;
    end
end
figure1 = figure;
axes1 = axes('Parent',figure1,'YGrid','on','XGrid','on')
ylim(axes1,[-0.2 1.2]);
box(axes1,'on');
hold(axes1,'on');
plot(t,u,'DisplayName','Escalon unitario','LineWidth',2);
xlabel('t [seg]');
ylabel('u[t]');
title('Escalon Unitario');
legend(axes1,'show');
% este if indica que va a asignar valor de 0 en todos
% los tiempos que sean menores a 0 y pondra
% 1 en los tiempos de muestreo mayores a 0
% generandose asi nuestro escalon unitario
```

GRAFICA DE LA FUNCION ESCALON



EXPERIMENTO 2

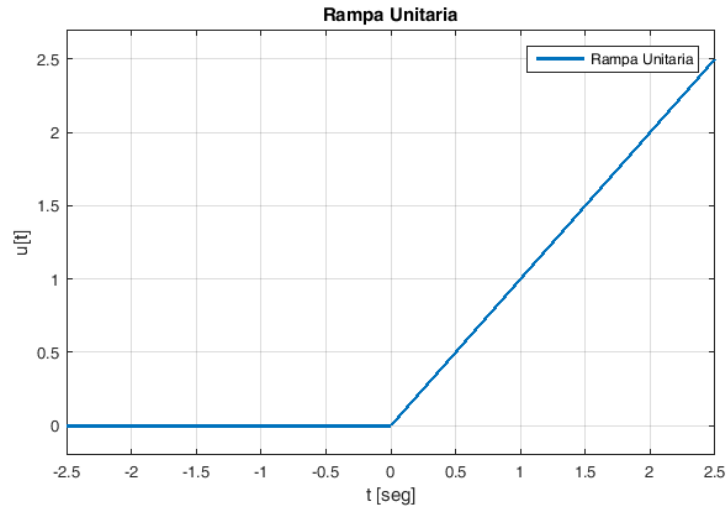
CODIGO PARA LA GENERACION DE UNA RAMPA UNITARIA

```

clc
clear
Fs=100;                                % Numero de Muestras Por Segundo
t=[-2.5:1/Fs:2.5];                      % Tiempo de Muestreo
Lt=length(t);                           % longitud del vector t
for k=1:Lt
for k=1:Lt
if t(k)<0
u(k)=0;
else
u(k)=1;
end
end
r=t.*u;
figure1 = figure;
axes1 = axes('Parent',figure1,'YGrid','on','XGrid','on')
ylim(axes1,[-0.2 2.7]);
box(axes1,'on');
hold(axes1,'on');
plot(t,r,'DisplayName','Rampa Unitaria','LineWidth',2);
xlabel('t [seg]');
ylabel('u[t]');
title('Escalon Unitario');
legend(axes1,'show');

```

GRAFICA DE LA FUNCION RAMPA UNITARIA



EXPERIMENTO 3

CODIGO PARA LA GENERACIÓN DE UN PULSO

```

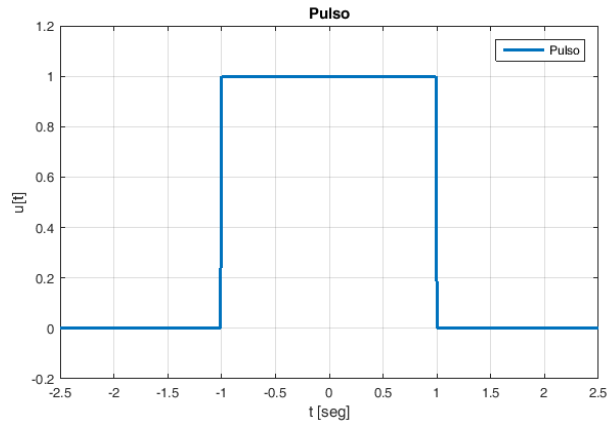
clc
clear
Fs=100;                                % Numero de Muestras Por Segundo
t=[-2.5:1/Fs:2.5];                      % Tiempo de Muestreo
Lt=length(t);                           % longitud del vector t
for k=1:Lt
    if t(k)<-1
        a(k)=0;
    else
        a(k)=1;
    end
end
for k=1:Lt
    if t(k)<1
        b(k)=0;
    else
        b(k)=1;
    end
end
p=a-b
figure1 = figure;
axes1 = axes('Parent',figure1,'YGrid','on','XGrid','on')
ylim(axes1,[-0.2 1.2]);
box(axes1,'on');
hold(axes1,'on');
plot(t,p,'DisplayName','Escalon unitario','LineWidth',2);
xlabel('t [seg]');
ylabel('u[t]');
title('Escalon Unitario');
legend(axes1,'show');

% este if indica que en el intervalo de -2.5 a 2.5
% va a mostrar en la salida 1

```

`% mostrando el pulso`

GRAFICA DE LA FUNCION PULSO



EXPERIMENTO 4

CODIGO PARA LA GENERACION DE UNA ONDA TRIANGULAR

```
clc
clear
Fs=100;                                % Numero de Muestras Por Segundo
t=[-2.5:1/Fs:2.5];                      % Tiempo de Muestreo
Lt=length(t);                           % longitud del vector t
for k=1:Lt
    if t(k)<-1
        u(k)=0;
    else
        u(k)=1;
    end
end
ra=(t+1).*u;
for k=1:Lt
    if t(k)<0
        u(k)=0;
    else
        u(k)=1;
    end
end
rb=t.*u;
for k=1:Lt
    if t(k)<1
        u(k)=0;
    else
        u(k)=1;
    end
end
rc=(t-1).*u;
triang=ra-2*rb+rc;
figure1 = figure;
axes1 = axes('Parent',figure1,'YGrid','on','XGrid','on')
```

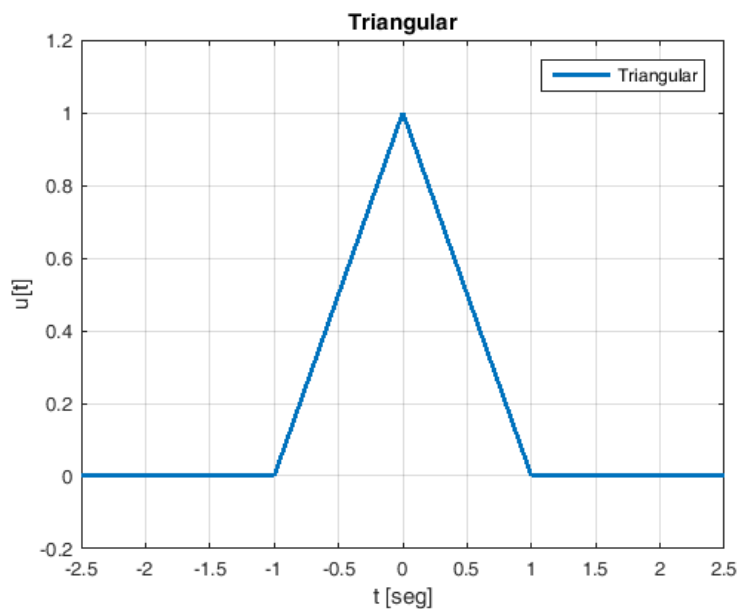
```

ylim(axes1,[-0.2 1.2]);
box(axes1,'on');
hold(axes1,'on');
plot(t,triang,'DisplayName','Triangular','LineWidth',2);
xlabel('t [seg]');
ylabel('u[t]');
title('Triangular');
legend(axes1,'show');

```

%aquí en este caso el if va a generar la grafica de 2 ecuaciones rectas que
%van a ser con pendiente positiva en el lado izquierdo y con pendiente
%negativa en el lado derecho formando así el triangulo

GRAFICA DE LA ONDA TRIANGULAR



Simulink

Se generó un nuevo modelo Simulink y se adecuó de acuerdo a lo establecido en el manual como se puede ver en la figura 2.1 posteriormente se pudieron visualizar las gráficas correspondientes como se muestra en la figura 2.2

Figura 2.1 Generación de Rampa y Escalón

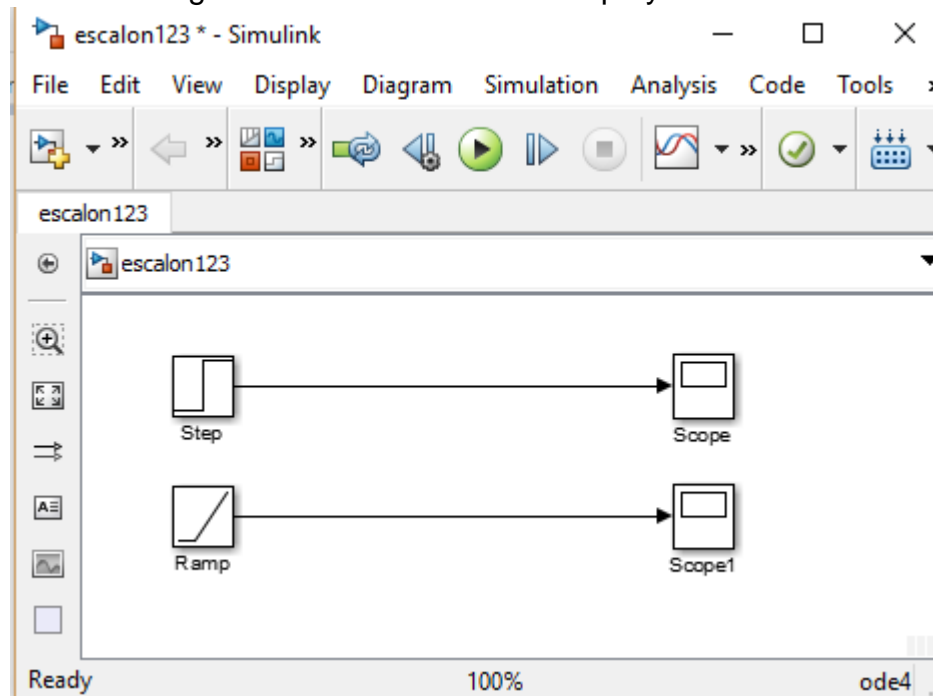
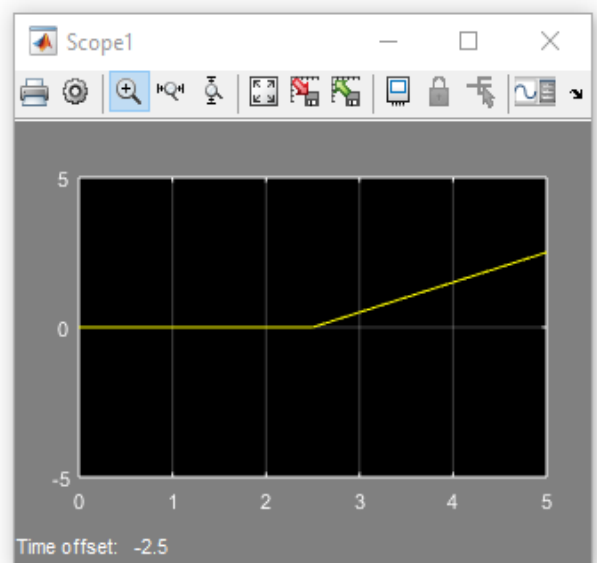
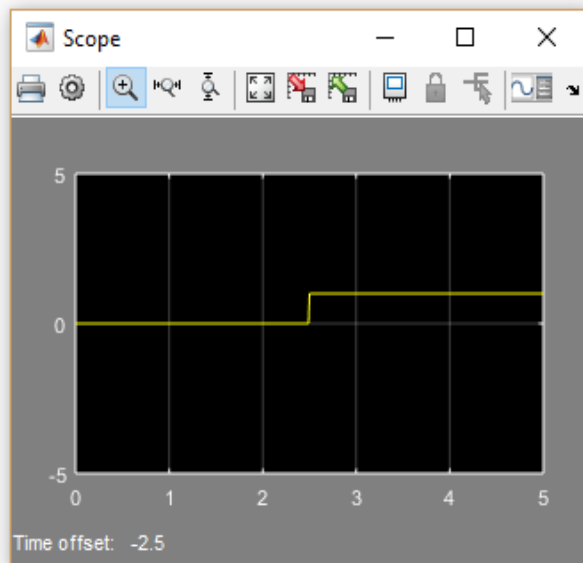
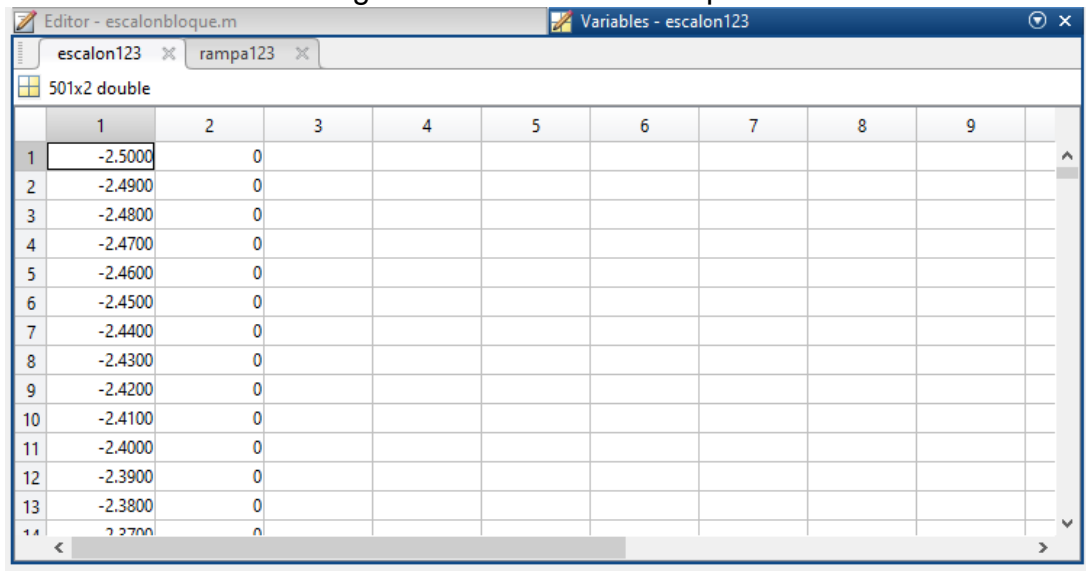


Figura 2.2 Scope 1 (escalón unitario) & Scope 2 (rampa unitaria)



Ahora teniendo este tipo de señales se utilizó el Workspace del software Matlab para obtener los valores correspondientes a cada gráficas como se muestra en la figura 2.3

Figura 2.3 Datos de Workspace

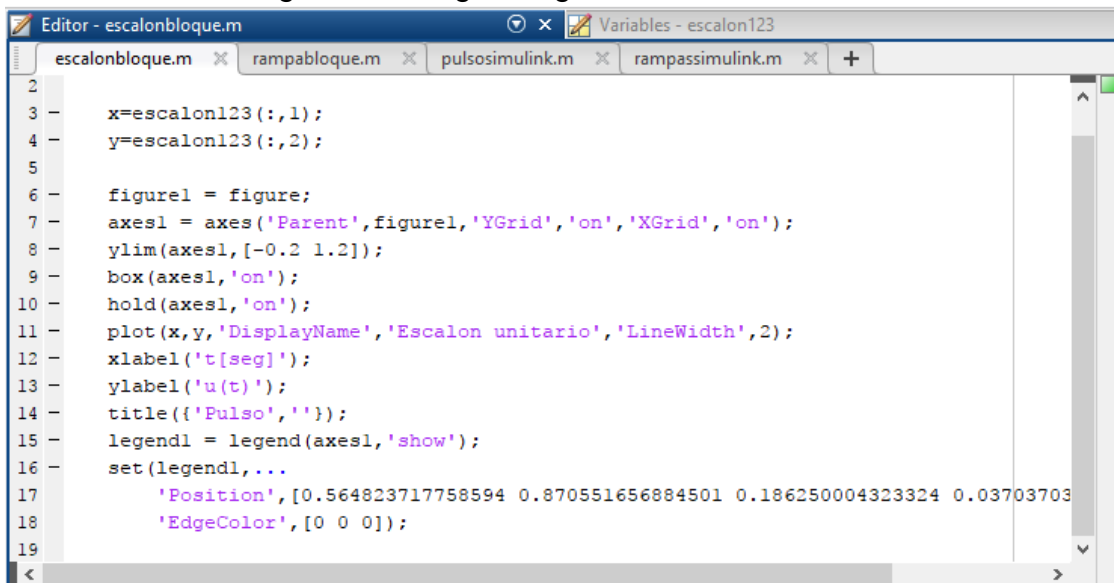


The screenshot shows the MATLAB Workspace window with the variable 'escalon123' selected. The variable is a 501x2 double array. The first column contains values ranging from -2.5000 to -2.3700, and the second column contains the value 0. The table below represents the data shown in the workspace.

	1	2	3	4	5	6	7	8	9
1	-2.5000	0							
2	-2.4900	0							
3	-2.4800	0							
4	-2.4700	0							
5	-2.4600	0							
6	-2.4500	0							
7	-2.4400	0							
8	-2.4300	0							
9	-2.4200	0							
10	-2.4100	0							
11	-2.4000	0							
12	-2.3900	0							
13	-2.3800	0							
14	-2.3700	0							

Se reutilizo el código generado anteriormente al editar la figura, a continuación se muestra el código correspondiente utilizado para generar graficas con las mismas condiciones a las ya generadas por ambos Scopes, como se puede observar el nombre de los e

Figura 2.4 Programa grafica escalón unitario



```

2
3 - x=escalon123(:,1);
4 - y=escalon123(:,2);
5
6 - figure1 = figure;
7 - axes1 = axes('Parent',figure1,'YGrid','on','XGrid','on');
8 - ylim(axes1,[-0.2 1.2]);
9 - box(axes1,'on');
10 - hold(axes1,'on');
11 - plot(x,y,'DisplayName','Escalon unitario','LineWidth',2);
12 - xlabel('t[seg]');
13 - ylabel('u(t)');
14 - title({'Pulso',''});
15 - legend1 = legend(axes1,'show');
16 - set(legend1,...
17     'Position',[0.564823717758594 0.870551656884501 0.186250004323324 0.03703703
18     'EdgeColor',[0 0 0]);
19

```

Figura 2.5 Programa grafica rampa unitaria

```

Editor - rampabloque.m
escalonbloque.m  rampabloque.m  pulsosimulink.m  rampassimulink.m  +
2
3  x1=rampal23(:,1);
4  y1=rampal23(:,2);
5
6  figure1 = figure;
7  axes1 = axes('Parent',figure1,'YGrid','on','XGrid','on');
8  ylim(axes1,[-0.2 2.8]);
9  box(axes1,'on');
10 hold(axes1,'on');
11 plot(x1,y1,'DisplayName','Escalon unitario','LineWidth',2);
12 xlabel('t[seg]');
13 ylabel('u(t)');
14 title({'Pulso',''});
15 legend1 = legend(axes1,'show');
16 set(legend1,...
17     'Position',[0.564823717758594 0.870551656884501 0.186250004323324 0.03703703
18     'EdgeColor',[0 0 0]);
19

```

Al correr el programa correspondiente a cada tipo de grafica nos encontramos con lo siguiente como se puede ver en la figura 2.6 y figura 2.7

Figura 2.6 Grafica de escalón generada mediante programación

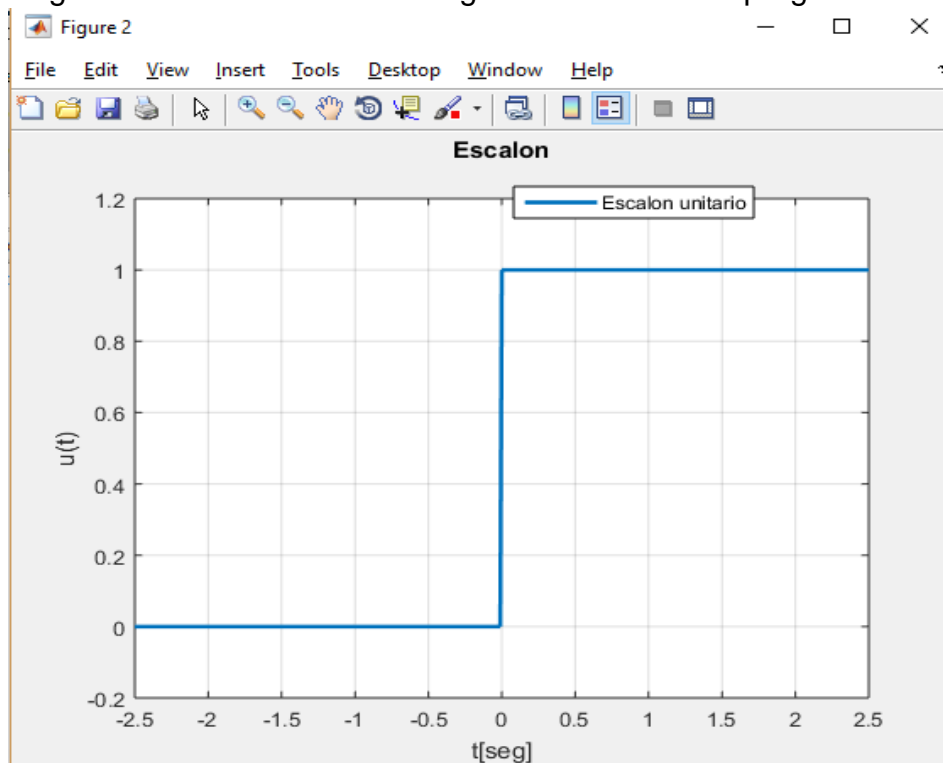
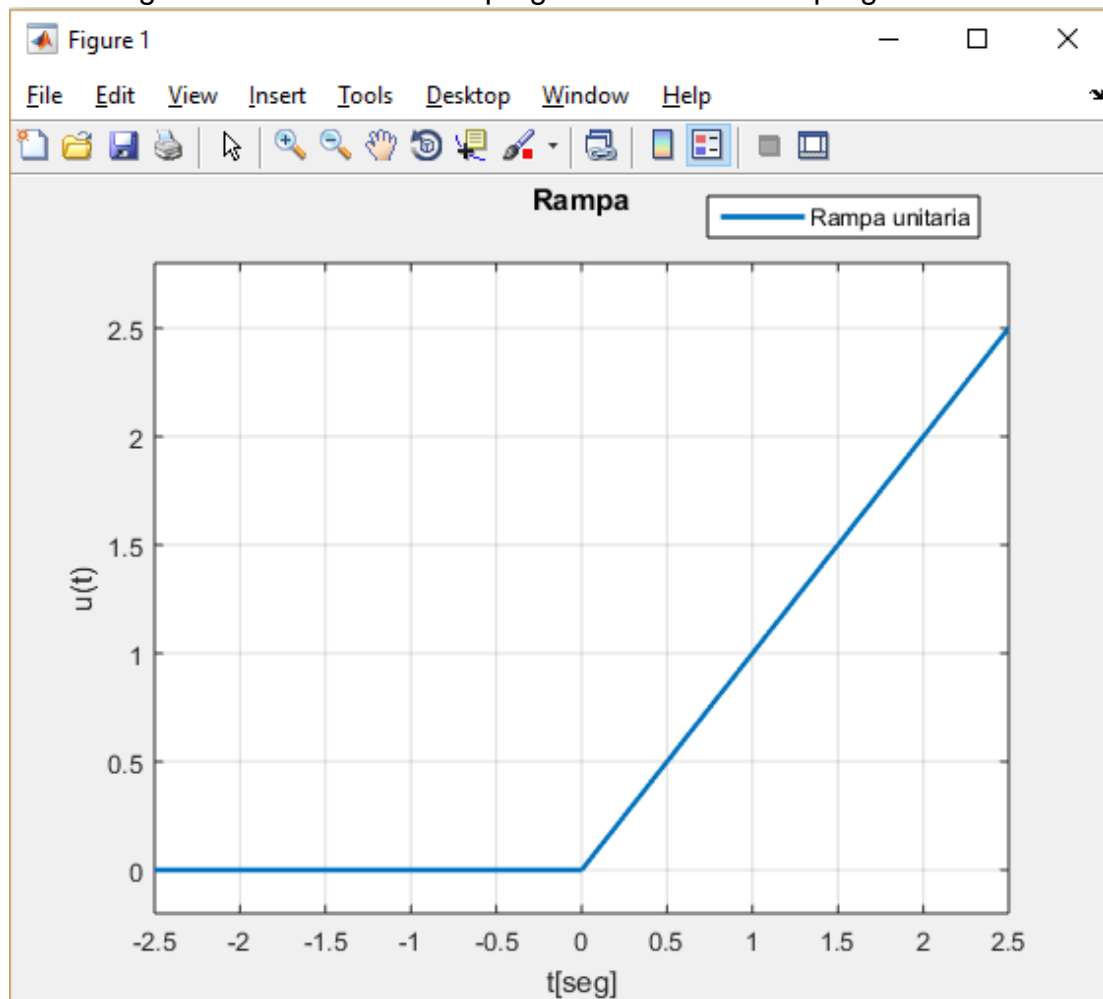


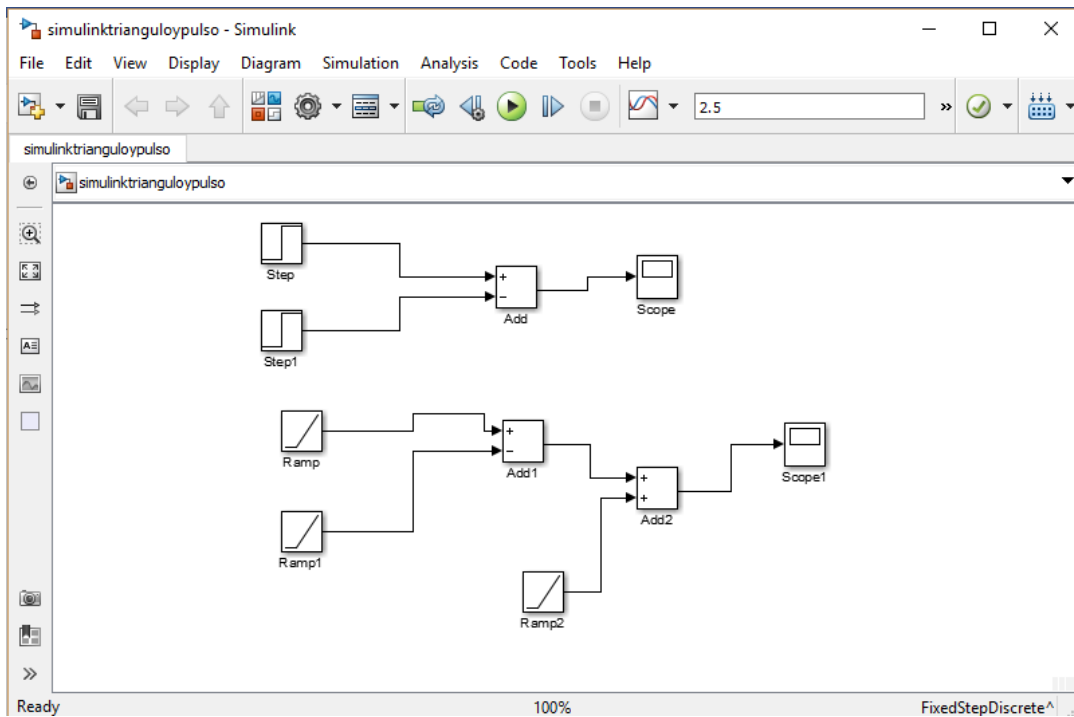
Figura 2.7 Grafica de rampa generada mediante programación



A continuación se realizó un procedimiento similar pero en este caso lo que se buscaba era generar una señal pulso y triangular mediante suma y resta de bloques tanto escalón como rampa para generar las señales en el Scope

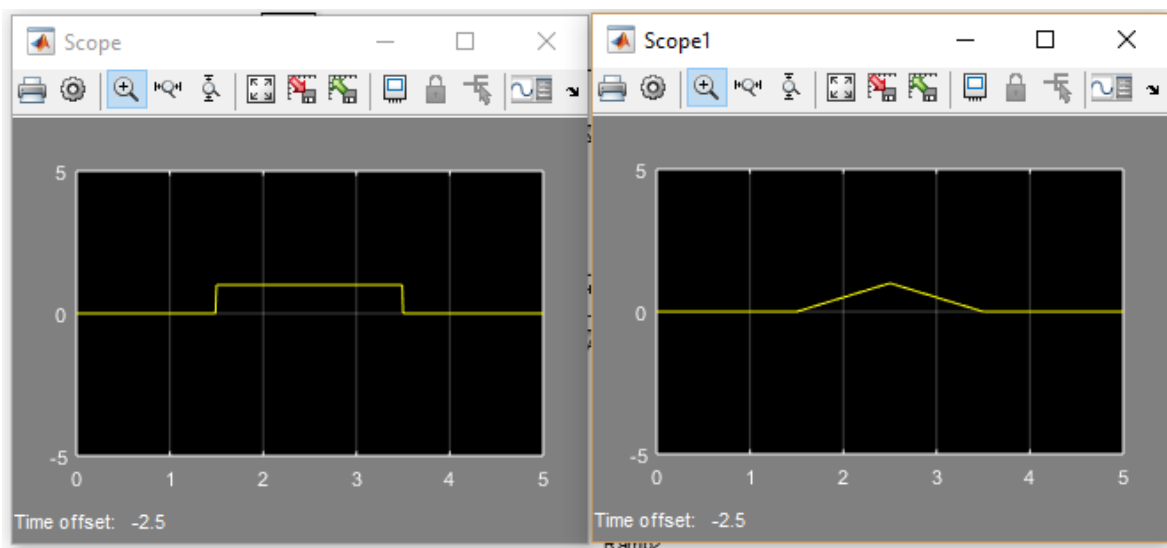
Como se muestra figura 2.8 se realizaron las operaciones correspondientes para generar las señales

Figura 2.8 Bloques Simulink triangulo y pulso



A partir de la figura 2.8 se generaron los siguientes gráficos presentes en la figura 2.9

Figura 2.9 Pulso y Rampa



Dadas las siguientes graficas se realizó el mismo procedimiento reutilizando el código de la figura y utilizando el workspace dando como resultado los siguientes códigos en la figura 2.10 y figura 2.11 mientras que en la figura 2.12 y figura 2.13 encontramos el resultado de los códigos obteniendo las correspondientes gráficas.

Figura 2.10 Programación pulso

```
Editor - pulsosimulink.m
escalonbloque.m  rampabloque.m  pulsosimulink.m  rampassimulink.m  +
1
2 - x3=sumasteps(:,1);
3 - y3=sumasteps(:,2);
4
5 - figure1 = figure;
6 - axes1 = axes('Parent',figure1,'YGrid','on','XGrid','on');
7 - ylim(axes1,[-0.2 1.2]);
8 - box(axes1,'on');
9 - hold(axes1,'on');
10 - plot(x3,y3,'DisplayName','Pulso','LineWidth',2);
11 - xlabel('t[seg]');
12 - ylabel('u(t)');
13 - title({'Pulso',''});
14 - legend1 = legend(axes1,'show');
15 - set(legend1,...
16     'Position',[0.564823717758594 0.870551656884501 0.186250004323324 0.03703703
17     'EdgeColor',[0 0 0]);
18
```

Figura 2.11 Grafica Pulso

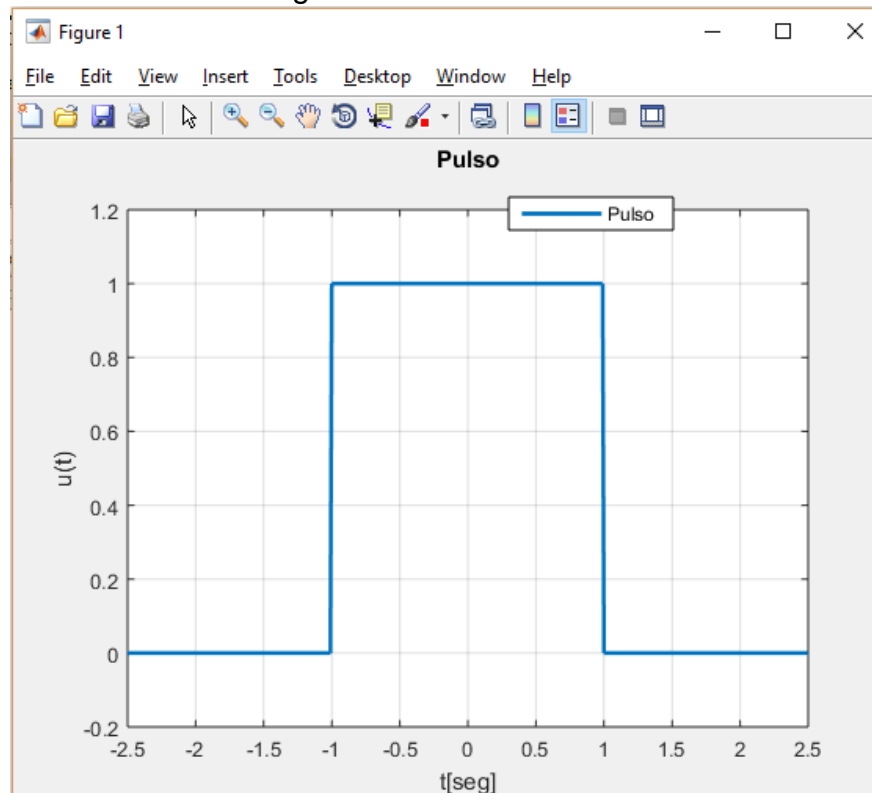


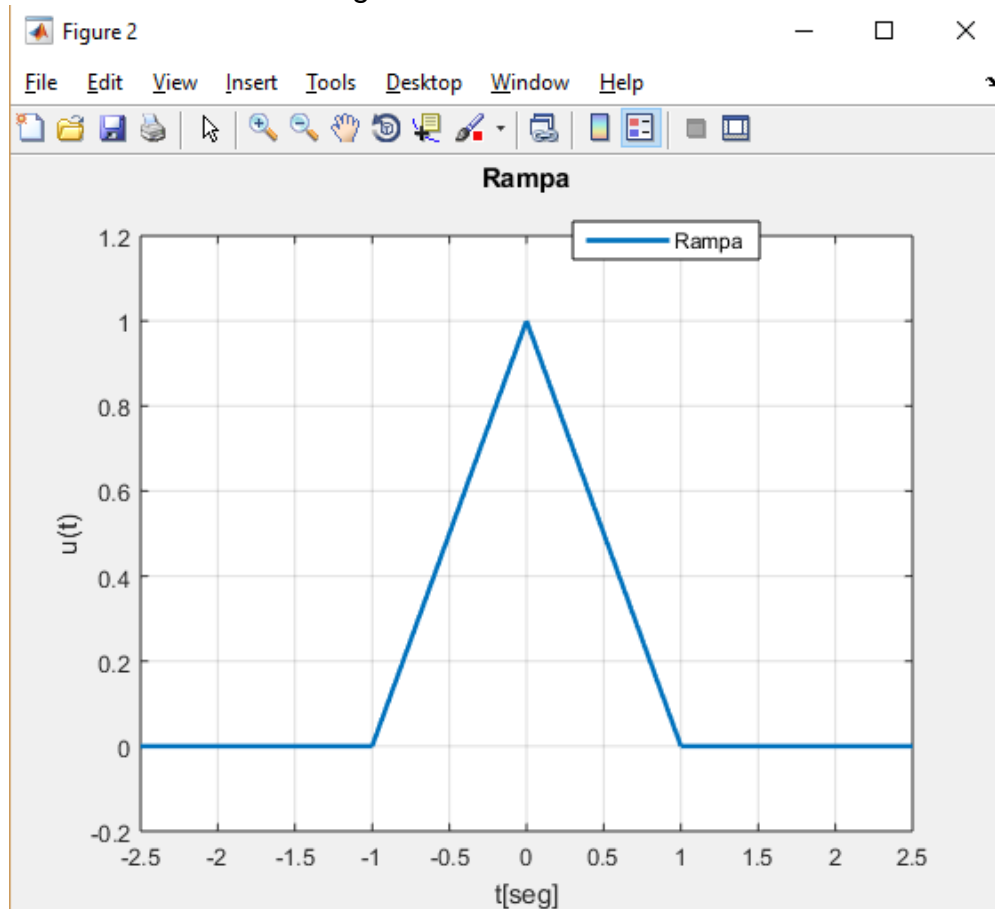
Figura 2.12 Programación rampa

```

Editor - rampassimulink.m
Variables - escalon123
escalonbloque.m rampabloque.m pulsosimulink.m rampassimulink.m +
1 - x4=rampasuma(:,1);
2 - y4=rampasuma(:,2);
3
4 - figure1 = figure;
5 - axes1 = axes('Parent',figure1,'YGrid','on','XGrid','on');
6 - ylim(axes1,[-0.2 1.2]);
7 - box(axes1,'on');
8 - hold(axes1,'on');
9 - plot(x4,y4,'DisplayName','Rampa','LineWidth',2);
10 - xlabel('t[seg]');
11 - ylabel('u(t)');
12 - title({'Rampa',''});
13 - legend1 = legend(axes1,'show');
14 - set(legend1,...
15     'Position',[0.564823717758594 0.870551656884501 0.186250004323324 0.0370370362
16     'EdgeColor',[0 0 0]);
17

```

Figura 2.13 Grafica Pulso



Cuestionario

1. Investiga cómo se pueden mostrar varias graficas en una figura con el comando subplot

En general, el comando subplot brinda una figura de configuración simétrica de paneles, que son donde se colocan las gráficas. Y cada panel es independiente de los otros. Basta con indicar como se compondrá la matriz de la imagen (a,b) y que espacio p ocupará cada una de las gráficas; es decir, subplot (m, n, p) hace la división del espacio de la figura. Matlab asignara el espacio con en paneles regulares para cada gráfica

3. Explica la diferencia del comando plot y stem

El comando básico para la representación de gráficos 2D es el comando plot. Por defecto, MATLAB dibuja uniendo los puntos con línea continua de color azul y un grosor determinado, una señal generada en Matlab es inherentemente de naturaleza discreta. Para visualizar una señal en tiempo discreto se puede hacer uso del comando stem. Específicamente stem(n, x), bosqueja los datos contenidos en el vector x como una señal de tiempo discreto con los valores de tiempo definidos por el vector n. Los vectores n y x deben tener dimensiones compatibles,

4. Muestra el atajo para conectar dos bloques con la tecla ctrl

Para añadir un conector pulsar el botón derecho del ratón y la tecla ctrl

5. Muestra como configurar un solo bloque sumador para que construya la señal $y = x_1 - x_2 - x_3 + x_4$

El bloque contendrá 4 entradas, donde corresponde +--+ con 1 salida

6. Investiga que bloques de Simulink® te pueden servir para la multiplicación y división de señales.

El bloque Product realiza multiplicación mientras que el otro sera el bloque Divide

7. Investiga que operaciones matemáticas puedes aplicar a una señal con los bloques 'Math Function' y 'Triginometric Function'.

Math Function realiza una función matemática mientras que Trigonometric Function especifica una función trigonométrica en la entrada

Conclusión

De acuerdo a lo realizado en el laboratorio de Procesamiento digital de imágenes y bioseñales se logró representar y reconstruir señales de tiempo continuo y tiempo discreto a través del muestreo de una señal y procesar las señales como funciones matemáticas. Además se logró construir funciones discontinuas como función pulso o triangular a partir de funciones especiales como función escalón y función rampa. Se reconstruyeron funciones definidas por intervalos mediante el proceso de ventana. Una herramienta vital fue el uso de Simulink® de Matlab® para la reconstrucción y representación de señales además de esto se implementó un programa de Matlab para la generación de gráficas de la representación de una señal de tiempo continuo o tiempo discreto.