INGENIERÍA
EN AUTOMATIZACIÓN

AUTÓNOMA UNIVERSIDAD DE QUERÉTARO

Facultad de Ingeniería

REPORTE DE TALLER DE ANÁLISIS DE SEÑALES

Alumno: **Diego Joel Zuñiga Fragoso**

Periodo: 2024-1

Catedrático: Dr. Suresh Thenozhi

ÍNDICES

No.	Título	Fecha	Pagina	Calificación
1	Elemental signals	14/02/24	3	
2	Basic operations on signals	18/02/24	8	
3	Fourier Series	22/03/24	13	
4	Fourier Transform	09/05/24	19	
5	Frequency response of Filters	14/05/24	25	
6	LTI Systems	14/05/24	30	
7	Sampling	31/05/24	35	
8	Audio Signal	31/05/24	42	
			Total	**/10**

Diego Joel
Zuñiga Fragoso

317684

Elemental signals	PRACTICA 1 FECHA 14/02/2024

1. OBJETIVO

Utilizando los conocimientos adquiridos en la clase y el software MATLAB, vamos a representar cada una de las señales vistas en el curso, estas las veremos ya sea en su forma discreta o continua. El objetivo es familiarizarse con el software MATLAB al momento de trabajar con este tipo de señales.

2. MARCO TEÓRICO

Las señales elementales, también conocidas como señales estándar, son fundamentales en el estudio de las señales y los sistemas. Estas señales sirven como bloques de construcción básicos para la creación de señales más complejas. De hecho, estas señales elementales pueden ser utilizadas para modelar una gran cantidad de señales físicas que ocurren en la naturaleza.

Entre las señales elementales más comunes se encuentran:

Señales de pulso unitario: Estas señales son breves y de duración finita.	1. Unit *Impulse* function $\delta(t) = \begin{cases} 1 & \text{for} \quad t = 0, \\ 0 & \text{for} \quad t \neq 0. \end{cases}$
Señales sinusoidales: Estas señales oscilan de manera regular y continua. Son fundamentales en la descripción de fenómenos periódicos.	2. Unit *Step* function $u(t) = \begin{cases} 1 & \text{for} \quad t \geq 0, \\ 0 & \text{for} \quad t < 0. \end{cases}$
La señal de rampa unitaria es una función matemática que aumenta linealmente con el tiempo a partir de t = 0. Se define como r(t) = t * u(t), donde u(t) es la función escalón unitario. Es una función continua, monótona creciente y con pendiente 1 para t > 0	3. Unit *Ramp* function $r(t) = \begin{cases} t & \text{for} \quad t \geq 0, \\ 0 & \text{for} \quad t < 0. \end{cases}$
La señal de parábola unitaria aumenta con el cuadrado del tiempo a partir de t = 0. Es una función continua y diferenciable en todo su dominio excepto en el punto t = 0. Es una función monótona creciente y su pendiente aumenta con el tiempo	4. Unit *Parabolic* function $p(t) = \begin{cases} \frac{t^2}{2} & \text{for} \quad t \geq 0, \\ 0 & \text{for} \quad t < 0. \end{cases}$
La señal seno, también llamada sinusoide, es una función matemática que oscila entre dos valores con una frecuencia y fase determinadas	5. *Sinc* function $\text{sinc}(t) = \begin{cases} \frac{\sin \pi t}{\pi t} & \text{for} \quad t \neq 0, \\ 1 & \text{for} \quad t = 0. \end{cases}$

La función exponencial es una función monótona creciente que aumenta o disminuye a un ritmo cada vez mayor para valores positivos de t. Es continua y diferenciable en todo su dominio.	6. *Real Exponential* function $x(t) = \exp^{\alpha t}$ where α is the growing ($\alpha > 0$) or decaying ($\alpha < 0$) factor.

Las señales elementales desempeñan un papel fundamental en diversos campos de la ingeniería, como el diseño de sistemas de control, la transmisión de datos, la electrónica, la instrumentación y la medición de señales. La comprensión de estas señales básicas resulta crucial para entender sistemas de señales complejos y para desarrollar sistemas de ingeniería eficaces y precisos.

3. IMPLEMENTACIÓN EN MATLAB

Código

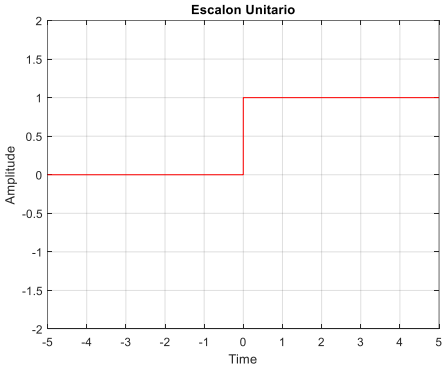
```
%% ESCALON UNITARIO
t =- 5:0.001:5;
y=(t>=0);
figure;
plot(t,y, 'r'), grid on;
xlabel('Time'), ylabel ( 'Amplitude');
ylim([-2 2]), xlim([ -5 5]);
title( 'Escalon Unitario');

%% RAMPA UNITARIA
t =- 5:0.01:5;
r=(t>=0)
h=t .* r;
figure;
plot(t,h, 'b'), grid on;
xlabel('Time'), ylabel('Amplitude');
ylim([-2 5]), xlim([-5 5]);
title( 'Rampa Unitaria');

%% PARABOLA UNITARIA

t =- 5:0.01:5;
r=(t>=0)
h=t .* r .* t;
figure;
plot(t,h, 'g'), grid on;
xlabel('Time'), ylabel('Amplitude');
ylim([-2 5]), xlim([-5 5]);
title( 'Parabola Unitaria');

%% IMPULSO UNITARIO

t =- 5:0.01:5;
r=(t == 0)
figure
plot(t,r,'y'), grid on;
xlabel( 'Time'), ylabel('Amplitude');
```

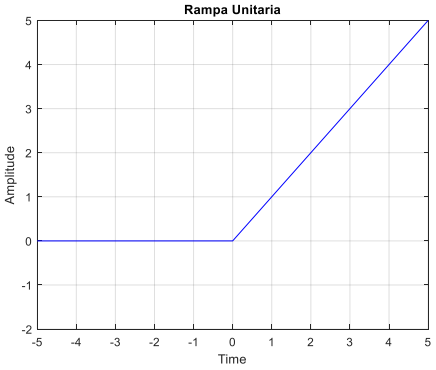
```
ylim([-2 5]), xlim([ -5 5]);
title( 'Impulso Unitario');

%% SINUSOIDAL

t = 0:0.01:6*pi;
frequency = 1;
amplitude = 1;
phase = 0;
sinusoid = amplitude * sin(2*pi*frequency*t + phase);
plot(t, sinusoid), grid on;
xlabel('Time (s)'), ylabel('Amplitude');
title( 'Funcion Sinusoidal');

%% SINC

x = -10:0.1:10;
y = sinc(x);
plot(x, y, 'b'), grid on;
xlabel('x'), ylabel('sinc(x)');
title( 'Funcion Sinc');

%% Funcion Cuadrada

t = 0:0.01:4*pi;
frequency = 1;
duty_cycle = 50;
square_wave = square(2*pi*frequency*t, duty_cycle);
plot(t, square_wave, 'r'), grid on;
xlabel('Time (s)'), ylabel('Amplitude');
title( 'Funcion Cuadrada');
ylim([-1.2, 1.2]);

%% Funcion Triangular

t = 0:0.01:4*pi;
frequency = 1;
triangular_wave = sawtooth(2*pi*frequency*t, 0.5);
plot(t, triangular_wave, 'y'), grid on;
xlabel('Time (s)'), ylabel('Amplitude');
title('Funcion Triangular');
ylim([-1.2, 1.2]);

%% Diente de sierra

t = 0:0.01:4*pi;
frequency = 1;
sawtooth_wave = sawtooth(2*pi*frequency*t);
plot(t, sawtooth_wave, 'b'), grid on;
xlabel('Time (s)'), ylabel ( 'Amplitude');
title( 'Diente de sierra');
ylim([-1.2, 1.2]);

%% Real Exponencial
```

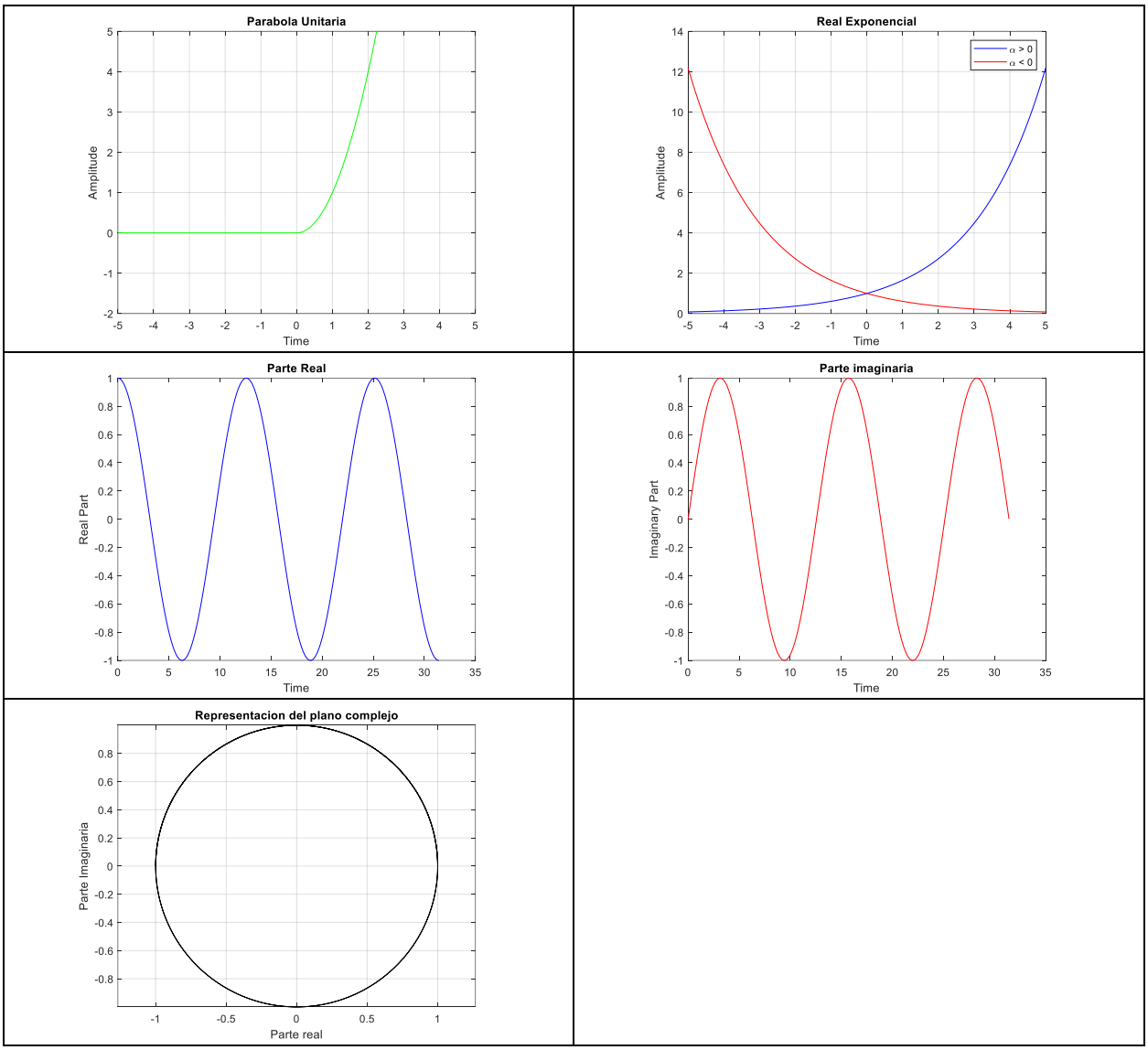
```
t = -5:0.01:5;
alpha_pos = 0.5;
alpha_neg = -0.5;
exponential_pos = exp(alpha_pos * t);
exponential_neg = exp(alpha_neg * t);
plot(t, exponential_pos, 'b'), grid on;;
hold on;
plot(t, exponential_neg, 'r'), grid on;
hold off;
xlabel('Time'), ylabel('Amplitude');
title( 'Real Exponencial');
legend('\alpha > 0', '\alpha < 0');

%% Exponencial Compleja

t = 0:0.01:10*pi;
alpha = 0.5;
commplex_exp = exp(1i * alpha * t);
figure;
plot(t, real(commplex_exp), 'b');
xlabel('Time'), ylabel('Real Part');
title( 'Parte Real');
figure;
plot(t, imag(commplex_exp), 'r');
xlabel('Time'), ylabel('Imaginary Part');
title( 'Parte imaginaria');
figure;
plot(real(commplex_exp), imag(commplex_exp), 'k'), grid on;
xlabel('Parte real'), ylabel('Parte Imaginaria');
title( 'Representacion del plano complejo');
axis equal;
```

Utilice mis conocimientos previos en Matlab y programación en C para representar las señales.

4. RESULTADOS







5. CONCLUSIÓN

En esta práctica, se logró representar con éxito una variedad de señales, utilizando el software MATLAB. Se ha demostrado la capacidad de comprender y aplicar los conocimientos adquiridos en clase sobre las características y propiedades de cada tipo de señal. Además, se ha adquirido familiaridad con el entorno MATLAB y sus herramientas para el manejo y visualización de señales

Diego Joel
Zuñiga Fragoso

317684

Basic operations on signals	PRACTICA 2 FECHA 18/02/2024

1. OBJETIVO

El objetivo de esta práctica es familiarizarme a las operaciones básicas sobre señales, tanto en el dominio del tiempo como en el de la amplitud. A través de la implementación y análisis de diferentes operaciones sobre señales en el software MATLAB, comprender cómo estas operaciones pueden ser utilizadas para modificar y analizar señales de diferentes tipos.

2. MARCO TEÓRICO

El procesamiento de señales implica la manipulación de las variables independientes y/o dependientes de la señal. Las operaciones básicas sobre señales son las siguientes:

1. Operaciones realizadas sobre la variable independiente (tiempo):

(a) Desplazamiento en el tiempo: El desplazamiento en el tiempo de una señal continua x(t) puede representarse como y(t) = x(t - t0). El desplazamiento en el tiempo de una señal puede resultar en un retardo o adelanto en el tiempo. Si t0 > 0, el desplazamiento es hacia la derecha y entonces el desplazamiento retrasa la señal, y si t0 < 0, el desplazamiento es hacia la izquierda y entonces el desplazamiento adelanta la señal.

(b) Inversión en el tiempo: La inversión en el tiempo, también llamada plegado en el tiempo de una señal x(t) se puede obtener plegando la señal sobre t = 0, denotado por x(-t).

(c) Escalado en el tiempo: El escalado en el tiempo de una señal continua x(t) puede representarse como y(t) = x(αt). El escalado en el tiempo puede ser una expansión o compresión del tiempo. Si α > 1, resulta en una compresión del tiempo por un factor α y si α < 1, resulta en una expansión del tiempo por un factor α.

2. Operaciones realizadas sobre la variable dependiente (amplitud):

(a) Escalado de amplitud: z(t) = βx(t), donde β es el factor de escalado de amplitud. Si β > 1, entonces la señal se amplifica por un factor β y si β < 1, entonces la señal se atenúa por un factor β.

(b) Suma de señales: z(t) = x(t) + y(t).

(c) Multiplicación de señales: z(t) = x(t) × y(t)

Para descomponer una señal en componentes pares e impares: Se dice que una señal de tiempo continuo x(t) es una señal (i) par (simétrica) si satisface la condición x(t) = x(-t) para todo t, y (ii)

impar (antisimétrica) si satisface la condición x(t) = -x(-t) para todo t. No todas las señales tienen que ser puramente pares o puramente impares, pero cada señal puede descomponerse en la suma de partes pares e impares mediante x(t) = x_e(t) + x_o(t), donde el componente par se da por:

$$x_e(t) = 1/2[x(t) + x(-t)]$$

Y la componente impar esta dada por:

$$x_o(t) = -[x(t) - x(-t)]$$

3. **IMPLEMENTACIÓN EN MATLAB**

Se anexa el código con explicaciones

Código

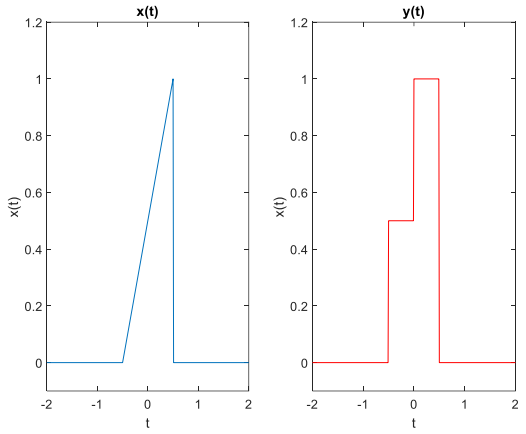
```
%% Señales Originales

figure('Name','Original Signals','NumberTitle','off');
% Variables y funciones
t = -2:10e-3:2;
xt = tripuls(t,1,1);
yt = 0.5*rectpuls(t,1) + 0.5*rectpuls(t-0.25,1/2);

% Graficamos
subplot(1,2,1), plot(t,xt);
title ('x(t)'), xlabel ('t'), ylabel ('x(t)'), ylim ([ -0.1 1.2]);

subplot(1,2,2), plot(t,yt, 'r');
title ('y(t)'), xlabel ('t'), ylabel ('x(t)'), ylim ([ -0.1 1.2]);

%% Time Shifting
% Creamos funciones
xt1 = tripuls(t-0.75,1,1);
xt2 = tripuls(t+0.75,1,1);
yt1 = 0.5*rectpuls(t-0.75,1) + 0.5*rectpuls(t-1,1/2);
yt2 = 0.5*rectpuls(t+0.75,1) + 0.5*rectpuls(t+0.5,1/2);

% Graficamos
figure('Name','Time Shifting','NumberTitle','off');
subplot(1,2,1), plot(t,xt1), hold on, plot(t,xt2, 'r');
title ('Triangular'), xlabel ('t'), ylabel ('Amplitude'), ylim ([ -0.1
1.2]), legend('x( t - 0.75 )', 'x( t + 0.75 )');

subplot(1,2,2), plot(t,yt1), hold on, plot(t, yt2, 'r');
title ('Cuadrada'), xlabel ('t'), ylabel ('Amplitude'), ylim ([ -0.1 1.2]),
legend('y( t - 0.75 )', 'y( t + 0.75 )');

%% Time Reversal
% Creamos funciones
xt = tripuls(-t,1,1);
yt = 0.5*rectpuls(t,1) + 0.5*rectpuls(-t-0.25,1/2);

% Graficamos
figure('Name','Time Reversal','NumberTitle','off');
```

```
subplot(1,2,1), plot(t,xt);
title ('x(t)'), xlabel ('t'), ylabel ('Amplitude'), ylim ([ -0.1 1.2]);

subplot(1,2,2), plot(t,yt,'r'), hold on;
title ('y(t)'), xlabel ('t'), ylabel ('Amplitude'), ylim ([ -0.1 1.2]);

%%  Time Scalling
% Creamos funciones
xt1 = tripuls(0.5*t,1,1);
xt2 = tripuls(1.5*t,1,1);
yt1 = 0.5*rectpuls(0.5*t,1) + 0.5*rectpuls(0.5 * t - 0.25,1/2);
yt2 = 0.5*rectpuls(1.5*t,1) + 0.5*rectpuls(1.5*t - 0.25,1/2);

% Graficamos
figure('Name','Time Scalling','NumberTitle','off');
subplot(1,2,1), plot(t,xt1), hold on, plot(t,xt2, 'r');
title ('Triangular'), xlabel ('t'), ylabel ('Amplitude'), ylim ([ -0.1
1.2]), legend('x( 0.5t )', 'x( 1.5t )');

subplot(1,2,2), plot(t,yt1), hold on, plot(t, yt2, 'r');
title ('Cuadrada'), xlabel ('t'), ylabel ('Amplitude'), ylim ([ -0.1 1.2]),
legend('y( 0.5t )', 'y( 1.5t )');

%% Amplitude Scaling
% Creamos funciones
xt1 = 0.5*tripuls(t,1,1);
xt2 = 1.1*tripuls(t,1,1);
yt1 = 0.5*(0.5*rectpuls(t,1) + 0.5*rectpuls(t - 0.25,1/2));
yt2 = 1.1*(0.5*rectpuls(t,1) + 0.5*rectpuls(t - 0.25,1/2));

% Graficamos
figure('Name','Amplitude Scaling','NumberTitle','off');
subplot(1,2,1), plot(t,xt1), hold on, plot(t,xt2, 'r');
title ('Triangular'), xlabel ('t'), ylabel ('Amplitude'), ylim ([ -0.1
1.2]), legend('0.5x( t )', '1.1x( t )');

subplot(1,2,2), plot(t,yt1), hold on, plot(t, yt2, 'r');
title ('Cuadrada'), xlabel ('t'), ylabel ('Amplitude'), ylim ([ -0.1 1.2]),
legend('0.5y( t )', '1.1y( t )');

%% Signal addition and multiplication.
% Creamos funciones
xt = tripuls(t,1,1);
yt = 0.5*rectpuls(t,1) + 0.5*rectpuls(t-0.25,1/2);
zt1 = xt + yt;
zt2 = xt .* yt;

% Graficamos
figure('Name','Signal addition and multiplication','NumberTitle','off');
subplot(1,2,1), plot(t,zt1);
title ('z(t) = x(t) + y(t)'), xlabel ('t'), ylabel ('Amplitude'), ylim ([ -
0.1 2.1]);

subplot(1,2,2), plot(t,zt2,'r'), hold on;
```

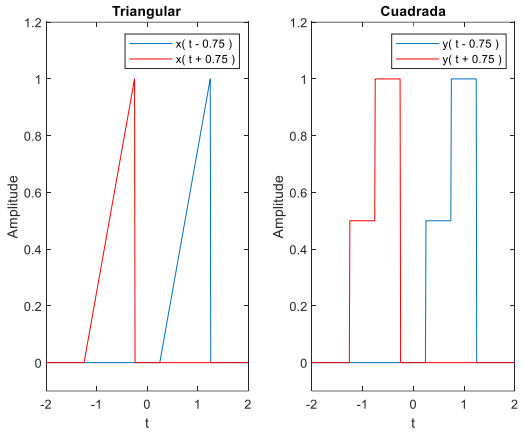
```
title ('z(t) = x(t) * y(t)'), xlabel ('t'), ylabel ('Amplitudes'), ylim ([ -
0.1 1.2]);
```

4. RESULTADOS

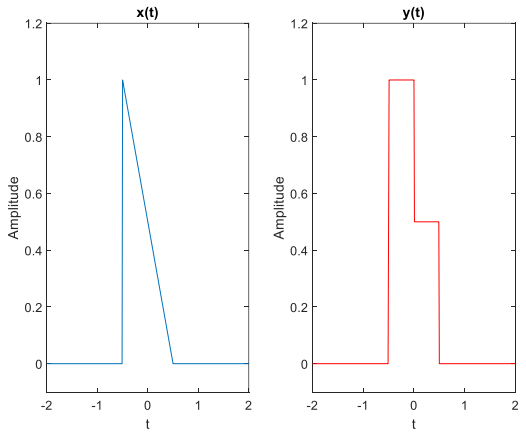
Funciones Originales



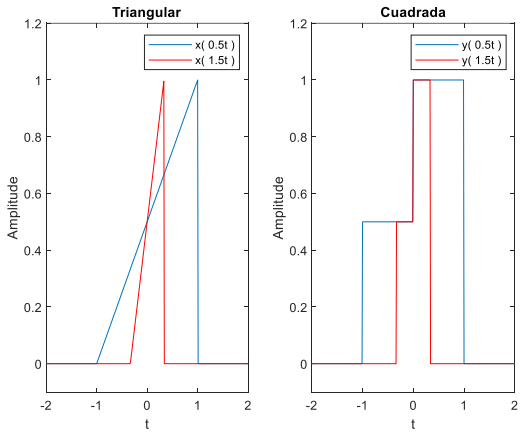
Time Shifting



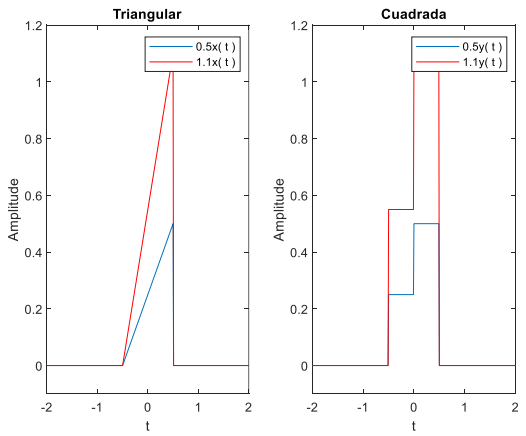
Time Reversal



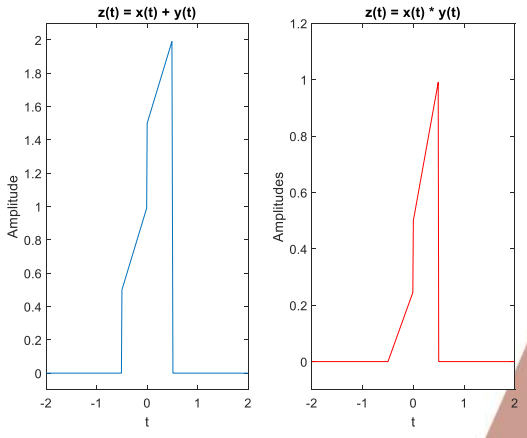
Time Scalling



Amplitude Scalling



Signal Addition and multiplication



5. CONCLUSIÓN

En esta práctica realizada con MATLAB, hemos explorado operaciones básicas sobre señales en los dominios del tiempo y la amplitud. A través de la implementación y análisis de operaciones como la suma, resta y multiplicación, entre otras, hemos consolidado nuestros conocimientos adquiridos en clase, permitiéndonos visualizar de manera más gráfica y aplicada los conceptos teóricos aprendidos.

Diego Joel
Zuñiga Fragoso

317684

Fourier Series	PRACTICA 3 FECHA 22/03/2024

1. OBJETIVO

- Aplicar y demostrar nuestra comprensión profunda de las series de Fourier, basándonos en los conceptos y teorías aprendidos en clase.
- Desarrollar un código en MATLAB que pueda calcular y representar la serie exponencial de Fourier para una función dada.
- Desarrollar un código en MATLAB que pueda calcular y representar la serie trigonométrica de Fourier para una función dada.
- Analizar y visualizar gráficamente la precisión de las aproximaciones de las series de Fourier. Además, investigar cómo la variación en el valor de N afecta la exactitud de estas aproximaciones.

2. MARCO TEÓRICO

La representación de señales durante un cierto intervalo de tiempo en términos de la combinación lineal de funciones ortogonales se llama serie de Fourier. El análisis de Fourier también se llama a veces análisis armónico. Las series de Fourier son aplicables solo para señales periódicas. No se pueden aplicar a señales no periódicas. Hay disponibles tres clases importantes de métodos de series de Fourier. Son:

1. Forma trigonométrica: Si las funciones ortogonales son funciones trigonométricas, entonces se llama serie de Fourier trigonométrica. La serie infinita de términos de seno y coseno se puede usar para formar la serie de Fourier trigonométrica y se puede escribir como (denominada ecuación de síntesis):

$$x(t) = a_0 + \sum_{n=1}^{\infty} a_n \cos(n\omega_0 t) + \sum_{n=1}^{\infty} b_n \sin(n\omega_0 t)$$

Donde n es un entero y los coeficientes de Fourier a_0, a_n, b_n se calculan utilizando las siguientes expresiones (denominadas ecuación de análisis):

$$a_0 = \frac{1}{T}\int_{t_0}^{t_0+T} x(t)dt$$

$$a_n = \frac{2}{T}\int_{t_0}^{t_0+T} x(t)\cos(n\omega_0 t)dt$$

$$b_n = \frac{2}{T}\int_{t_0}^{t_0+T} x(t)\sin(n\omega_0 t)dt$$

2. Forma exponencial: Si las funciones ortogonales son funciones exponenciales complejas, entonces se llama serie de Fourier exponencial. En este caso, la función x(t) se expresa como una suma ponderada de las funciones exponenciales complejas. Aunque la forma trigonométrica es una forma común de las series de Fourier, la forma exponencial compleja es más general y generalmente más conveniente y más compacta. La serie de Fourier exponencial puede expresarse utilizando la siguiente ecuación de síntesis:

$$x(t) = \sum_{n=-\infty}^{\infty} c_n \exp^{jn\omega_0 t}$$

Donde el coeficiente de Fourier c_n se calcula utilizando la siguiente ecuación de análisis:

$$c_n = \frac{1}{T}\int_{t_0}^{t_0+T} x(t)\exp^{-jn\omega_0 t}\,dt$$

3. Forma de coseno: Aquí, la función de coseno se utiliza como la función ortogonal. Puede considerarse como un caso especial de la forma trigonométrica.

3. IMPLEMENTACIÓN EN MATLAB

Se anexa el código con explicaciones

Código

```
t = -3*pi:10e-3:3*pi;
N = 100;

T = 2*pi; A = 1; w = 2*pi/T;
%% Funcion cuadrada antisimetrica
xt = A*square(w*t);

% Serie exponencial de Fourier
exp_yt = 0;
cn = zeros(2*N+1,1);
index = 1;

% Sumatoria
for n = -N : N
    % Calculo valor de Cn
    if(mod(n,2))
        % Par
```

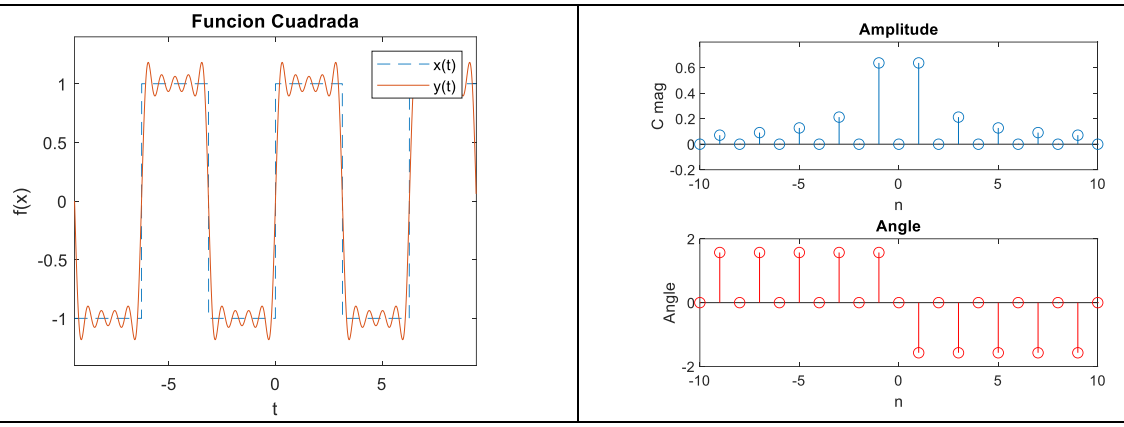
```
        cn(index) = (-j)*(2*A)/(pi*n);
    else
        % Impar
        cn(index) = 0;
    end

    % Calculo valor de serie de fourier
    exp_yt = exp_yt + cn(index)*exp(j*n*w*t);

    % Aumento el indice
    index = index+1;
end

% Calculo valores de magnitud y angulo de Cn
n = -N : N;
cmag = abs(cn);
cph = angle(cn);

% Grafico funciones
figure;
plot(t,xt,'--'), xlim ([ -3*pi 3*pi ]), ylim([-1.4 1.4]), xlabel ('t'),
ylabel ('f(x)'), title('Exponencial de Fourier');
hold on, plot(t,exp_yt), legend('x(t)','y(t)');

% Grafico magnitud y angulo de Cn
figure;
subplot(2,1,1), stem(n,cmag),ylim ([ -0.2 0.8]), xlabel ('n'), ylabel ('C
mag'), title('Amplitude');
subplot(2,1,2), stem(n,cph, 'r'), ylim ([ -2 2]), xlabel ('n'), ylabel
('Angle'), title('Angle');

% Serie trigonometrica de Fourier
tri_yt = 0;
an = zeros(N,1);
bn = zeros(N,1);

index = 2;

an(1) = 0;
bn(1) = 0;

% Sumatoria
for n = 1 : N
    % Calculo valor de an y bn
    if(mod(n,2))
        % Par
        an(index) = 0;
        bn(index) = 4/(n*pi);
    else
        % Impar
        an(index) = 0;
        bn(index) = 0;
    end

    % Calculo valor de serie de fourier
```

```
    tri_yt = tri_yt + an(index)*cos(n*w*t) + bn(index)*sin(n*w*t);

    % Aumento el indice
    index = index+1;
end

n = 0 : N;

% Grafico funciones
figure;
plot(t,xt,'--'), xlim ([ -3*pi 3*pi ]), ylim([-1.4 1.4]), xlabel ('t'),
ylabel ('f(x)'), title('Trigonometrica de fourier');
hold on, plot(t,tri_yt), legend('x(t)','y(t)');

% Grafico de an y bn
figure;
subplot(2,1,1), stem(n,an),ylim ([ -0.2 0.8]), xlabel ('n'), ylabel ('a_n'),
title('a_n');
subplot(2,1,2), stem(n,bn, 'r'), ylim ([ -0.2 1.5]), xlabel ('n'), ylabel
('b_n'), title('b_n');
```

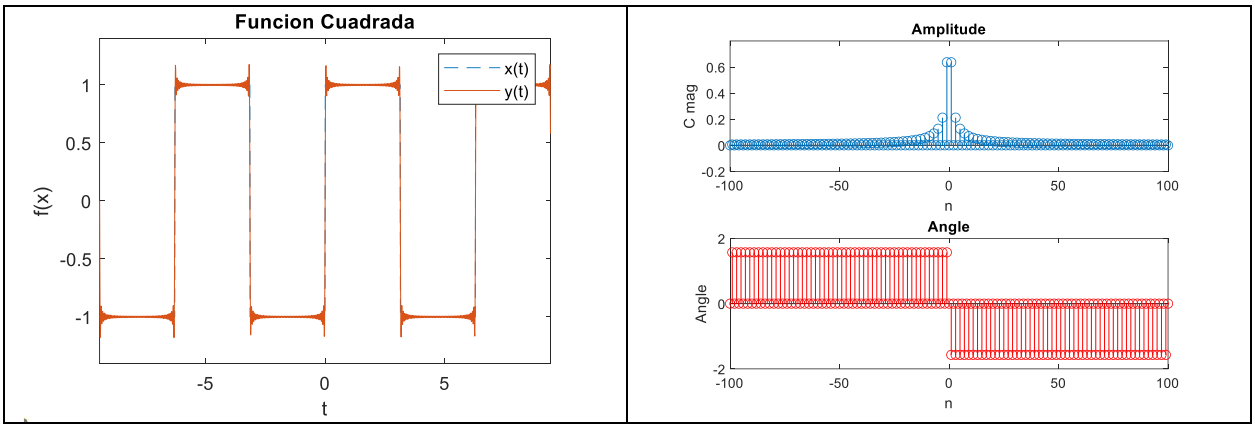
4. RESULTADOS

Exponencial de Fourier de función cuadrada antisimétrica

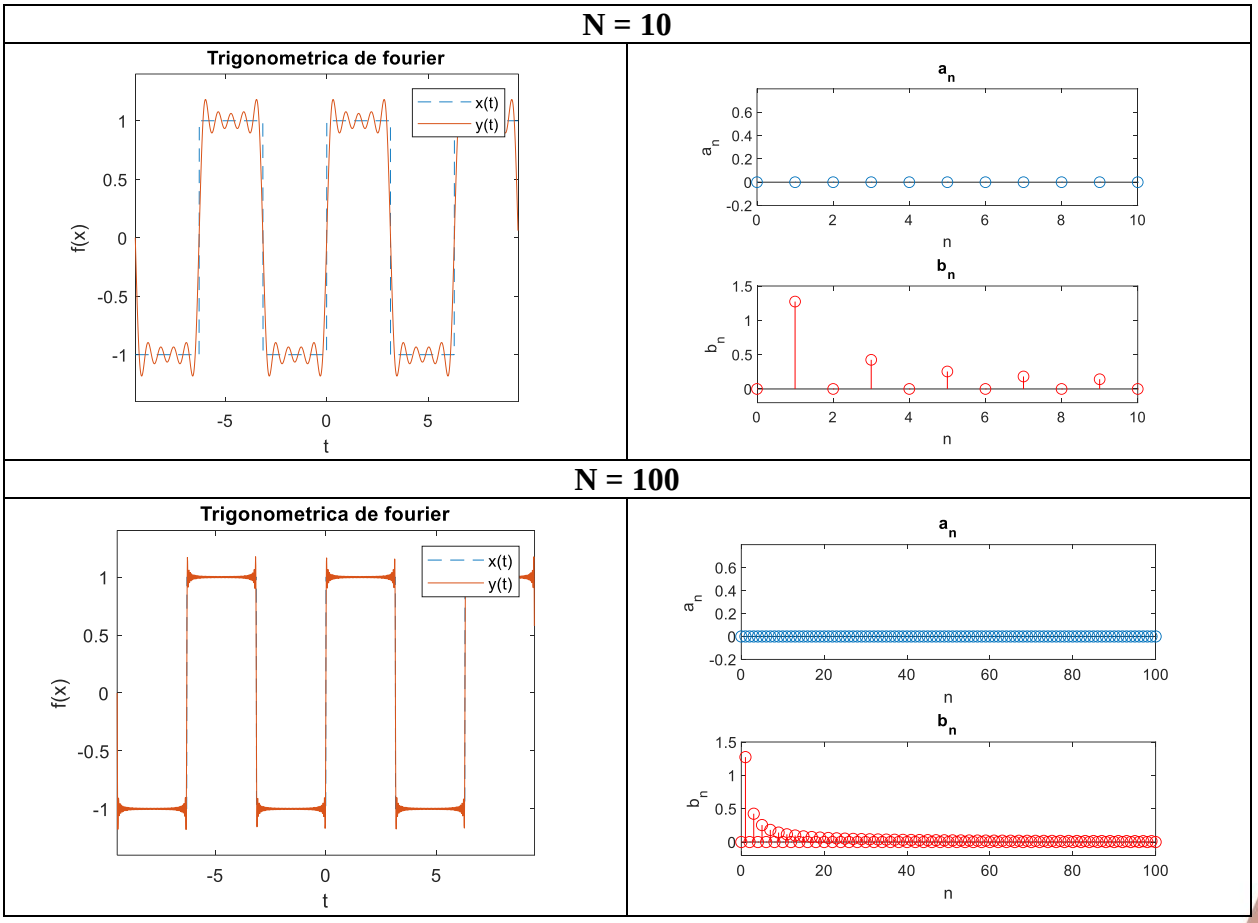
N = 10



N = 100



Trigonometrica de Fourier de función cuadrada antisimétrica



5. CONCLUSIÓN

La práctica de las series de Fourier, tanto en su forma exponencial como trigonométrica, es una herramienta esencial en el análisis de señales y sistemas. A través de esta práctica, hemos podido aplicar nuestros conocimientos teóricos a problemas prácticos utilizando MATLAB. Hemos desarrollado códigos para calcular y visualizar estas series, y hemos observado cómo la precisión de la aproximación varía con el número de términos en la serie. Esta experiencia nos ha

proporcionado una comprensión más profunda de las series de Fourier y su aplicación en el análisis de señales. En conclusión, esta práctica ha sido una oportunidad valiosa para consolidar nuestros conocimientos y habilidades en este importante tema

Diego Joel
Zuñiga Fragoso

317684

Fourier Transform	PRACTICA 5 FECHA 09/05/2024

1. OBJETIVO

Desarrollar una comprensión sólida y práctica de la Transformada de Fourier continua, y aprender a calcular la magnitud y el ángulo de una señal en el dominio de la frecuencia utilizando MATLAB. Esto nos permitirá analizar y entender mejor las señales en términos de sus componentes frecuenciales.

2. MARCO TEÓRICO

En el mundo del análisis de señales y sistemas, la transformada de Fourier (TF) se alza como una herramienta fundamental, ofreciendo una descomposición profunda de señales en sus componentes de frecuencia. Esta técnica matemática revela el comportamiento de las señales en el dominio de la frecuencia, lo que la convierte en un recurso invaluable en diversos campos.

La TF funciona bajo el principio de que cualquier señal, ya sea periódica o no, puede representarse como una suma de ondas sinusoidales de distintas frecuencias y amplitudes. Al aplicar la TF, se obtiene una representación espectral de la señal, donde cada componente revela la contribución de una frecuencia específica a la señal original.

$$X(f) = \int_{-\infty}^{\infty} x(t)e^{-j2\pi ft}dt$$

Las aplicaciones de la TF son tan amplias como diversas, abarcando desde el procesamiento de señales y el análisis de imágenes hasta las comunicaciones y la mecánica. Entre sus aplicaciones más destacadas encontramos:

- Procesamiento de señales de audio: La TF permite analizar la composición frecuencial de una señal de audio, identificar sus componentes armónicos, visualizar espectrogramas y aplicar técnicas de filtrado para eliminar ruido o enfatizar frecuencias de interés.
- Procesamiento de imágenes: En el ámbito del procesamiento de imágenes, la TF juega un papel crucial en técnicas como el filtrado espacial, la detección de bordes y el análisis de texturas. Al aplicarse sobre una imagen, la TF revela la distribución de la información espacial en diferentes frecuencias, permitiendo manipular la imagen de manera precisa.

- Comunicaciones: La TF es esencial en el diseño de sistemas de comunicación, ya que permite analizar la modulación de señales, evaluar la eficiencia espectral y comprender los efectos de la interferencia.
- Mecánica: En el campo de la mecánica, la TF se utiliza para analizar vibraciones, ondas acústicas y el comportamiento estructural de materiales.

3. IMPLEMENTACIÓN EN MATLAB

Se anexa el código con explicaciones

Código

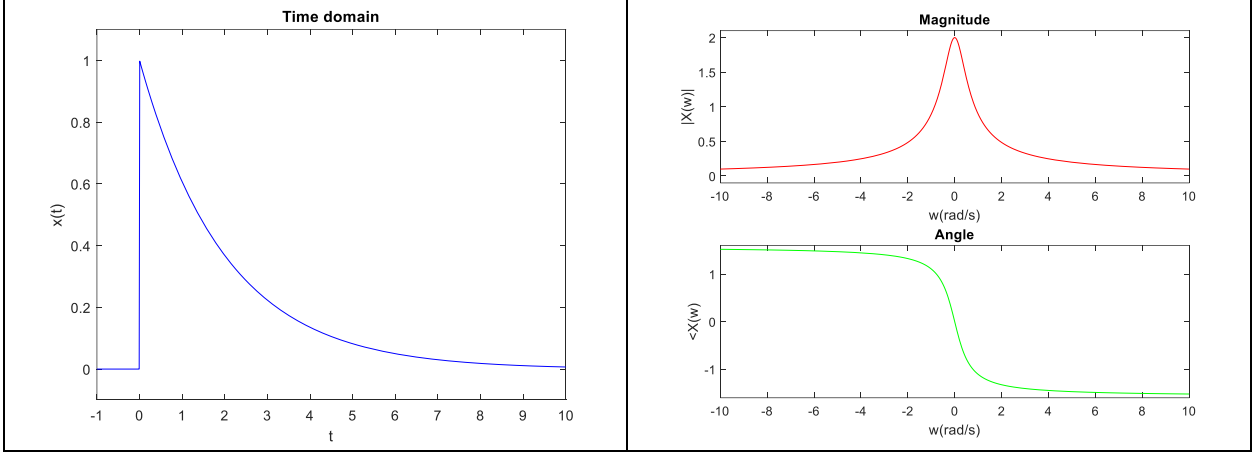
```
% Declaramos variables
syms t;
syms w;

%% Funcion 1
% Declaramos funciones
xt = exp(-0.5*t) * heaviside(t);
xw = fourier(xt);
xwmag = abs(xw);
xwph = angle(xw);

% Graficamos funciones
figure;
fplot(t,xt,'b', [-1 10]), xlim ([ -1 10 ]), ylim([-0.1 1.1]), xlabel ('t'),
ylabel ('x(t)'), title('Time domain');

figure;
subplot(2,1,1), fplot(w,xwmag,'r', [-10 10]), xlim ([ -10 10 ]), ylim([-0.1
2.1]), xlabel ('w(rad/s)'), ylabel ('|X(w)|'), title('Magnitude');
subplot(2,1,2), fplot(w,xwph,'g', [-10 10]), xlim ([ -10 10 ]), ylim([-1.6
1.6]), xlabel ('w(rad/s)'), ylabel ('<X(w)'), title('Angle');
%% Funcion 2
% Declaramos funciones
xt = rectangularPulse( -0.5, 0.5, t);
xw = fourier(xt);
xwmag = abs(xw);
xwph = angle(xw);

% Graficamos funciones
figure;
fplot(t,xt,'b', [-1 1]), xlim ([ -1 1 ]), ylim([-0.1 1.1]), xlabel ('t'),
ylabel ('x(t)'), title('Time domain');

figure;
subplot(2,1,1), fplot(w,xwmag,'r', [-25 25]), xlim ([-25 25]), ylim([-0.1
1.1]), xlabel ('w(rad/s)'), ylabel ('|X(w)|'), title('Magnitude');
subplot(2,1,2), fplot(w,xwph,'g', [-25 25]), xlim ([-25 25]), ylim([-0.1
3.4]), xlabel ('w(rad/s)'), ylabel ('<X(w)'), title('Angle');
%% Funcion 3
% Declaramos funciones
```

```
xt = sinc(t);
xw = fourier(xt);
xwmag = abs(xw);
xwph = angle(xw);

% Graficamos funciones
figure;
fplot(t,xt,'b', [-10 10]), xlim ([ -10 10 ]), ylim([-0.3 1.1]), xlabel
('t'), ylabel ('x(t)'), title('Time domain');

figure;
subplot(2,1,1), fplot(w,xwmag,'r', [-10 10]), xlim ([ -10 10 ]), ylim([-0.1
1.1]), xlabel ('w(rad/s)'), ylabel ('|X(w)|'), title('Magnitude');
subplot(2,1,2), fplot(w,xwph,'g', [-10 10]), xlim ([ -10 10 ]), ylim([-0.5
0.5]), xlabel ('w(rad/s)'), ylabel ('<X(w)'), title('Angle');

%% Funcion 4
% Declaramos funciones
xt = exp(-0.5*abs(t));
xw = fourier(xt);
xwmag = abs(xw);
xwph = angle(xw);

% Graficamos funciones
figure;
fplot(t,xt,'b', [-10 10]), xlim ([ -10 10 ]), ylim([-0.1 1.1]), xlabel
('t'), ylabel ('x(t)'), title('Time domain');

figure;
subplot(2,1,1), fplot(w,xwmag,'r', [ -5 5 ]), xlim ([ -5 5 ]), ylim([-0.1
4.2]), xlabel ('w(rad/s)'), ylabel ('|X(w)|'), title('Magnitude');
subplot(2,1,2), fplot(w,xwph,'g', [ -5 5 ]), xlim ([ -5 5 ]), ylim([-1.6
1.6]), xlabel ('w(rad/s)'), ylabel ('<X(w)'), title('Angle');

%% Funcion 5
% Declaramos funciones
xt = exp(-2*t)*cos(5*t)*heaviside(t);
xw = fourier(xt);
xwmag = abs(xw);
xwph = angle(xw);

% Graficamos funciones
figure;
fplot(t,xt,'b', [-0.1 3]), xlim ([ -0.1 3 ]), ylim([-0.4 1.1]), xlabel
('t'), ylabel ('x(t)'), title('Time domain');

figure;
subplot(2,1,1), fplot(w,xwmag,'r', [ -100 100 ]), xlim ([ -100 100 ]),
ylim([-0.1 0.4]), xlabel ('w(rad/s)'), ylabel ('|X(w)|'),
title('Magnitude');
subplot(2,1,2), fplot(w,xwph,'g', [ -100 100 ]), xlim ([ -100 100 ]),
ylim([-2 2]), xlabel ('w(rad/s)'), ylabel ('<X(w)'), title('Angle');

%% Funcion 6
% Declaramos funciones
xt = t*exp(-0.5*t)*heaviside(t);
```

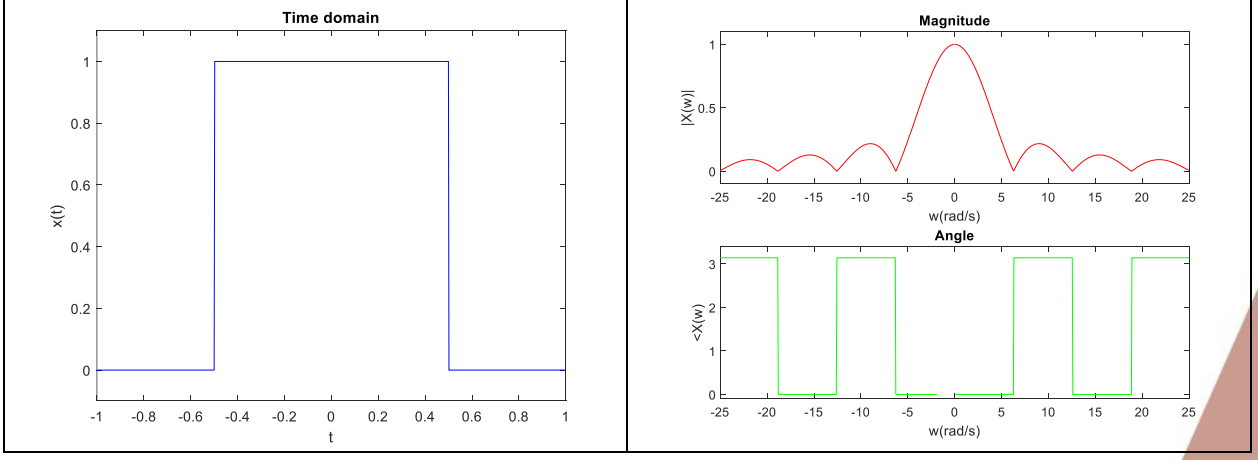
```
xw = fourier(xt);
xwmag = abs(xw);
xwph = angle(xw);

% Graficamos funciones
figure;
fplot(t,xt,'b', [-0.5 20]), xlim ([ -0.5 20 ]), ylim([-0.1 0.8]), xlabel
('t'), ylabel ('x(t)'), title('Time domain');

figure;
subplot(2,1,1), fplot(w,xwmag,'r', [ -8 8 ]), xlim ([ -8 8 ]), ylim([-0.1
4.5]), xlabel ('w(rad/s)'), ylabel ('|X(w)|'), title('Magnitude');
subplot(2,1,2), fplot(w,xwph,'g', [ -8 8 ]), xlim ([ -8 8 ]), ylim([-4.5
4.5]), xlabel ('w(rad/s)'), ylabel ('<X(w)'), title('Angle');
```

3. RESULTADOS

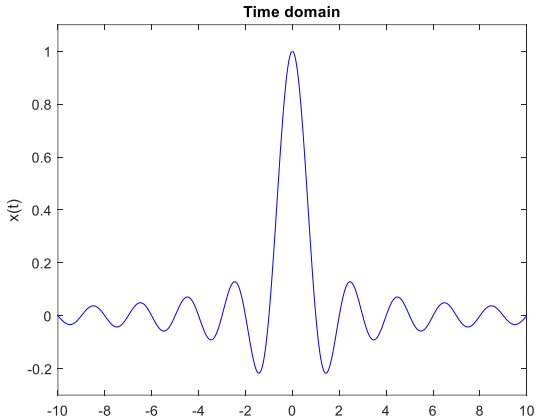
$$x(t) = e^{-0.5t}u(t)$$

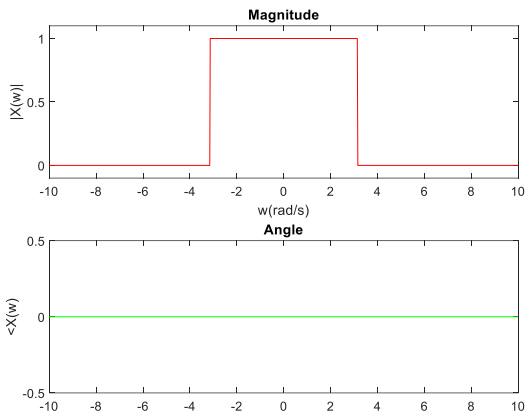


$$x(t) = \{\, 1 \; for -0.5 \leq t \leq 0.5 \mid 0 \; else$$

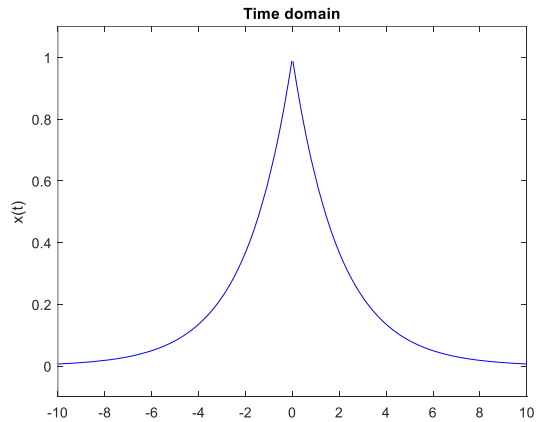


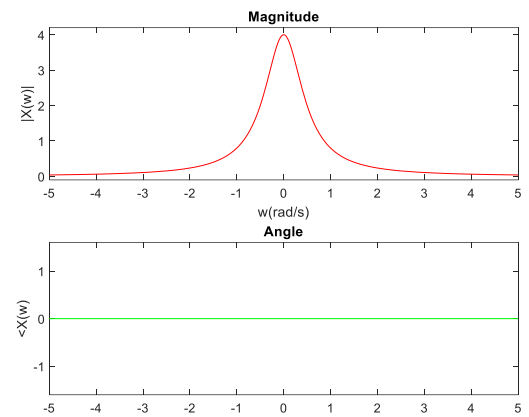
$$x(t) = sinc(t)$$



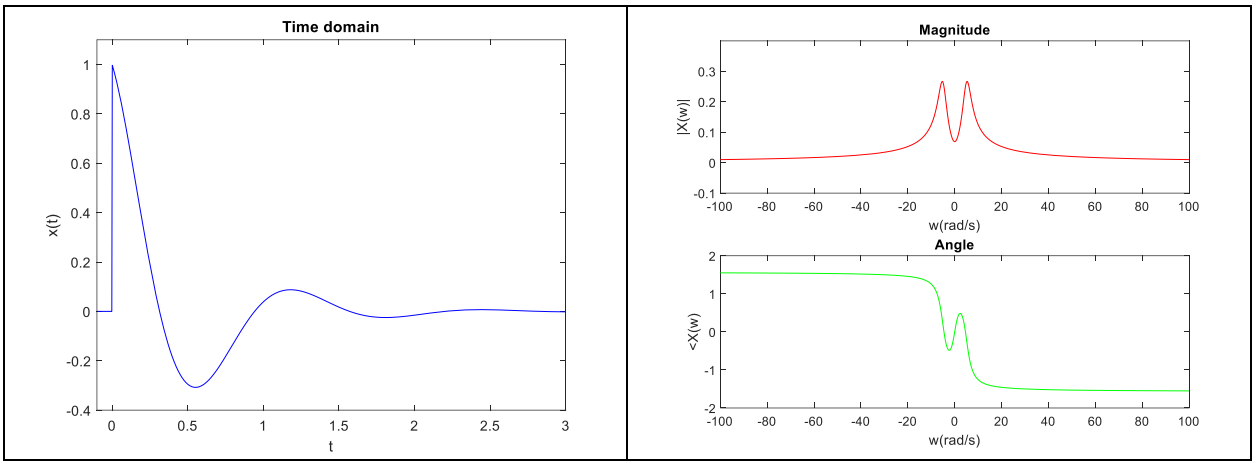


$$x(t) = e^{-0.5|t|}$$

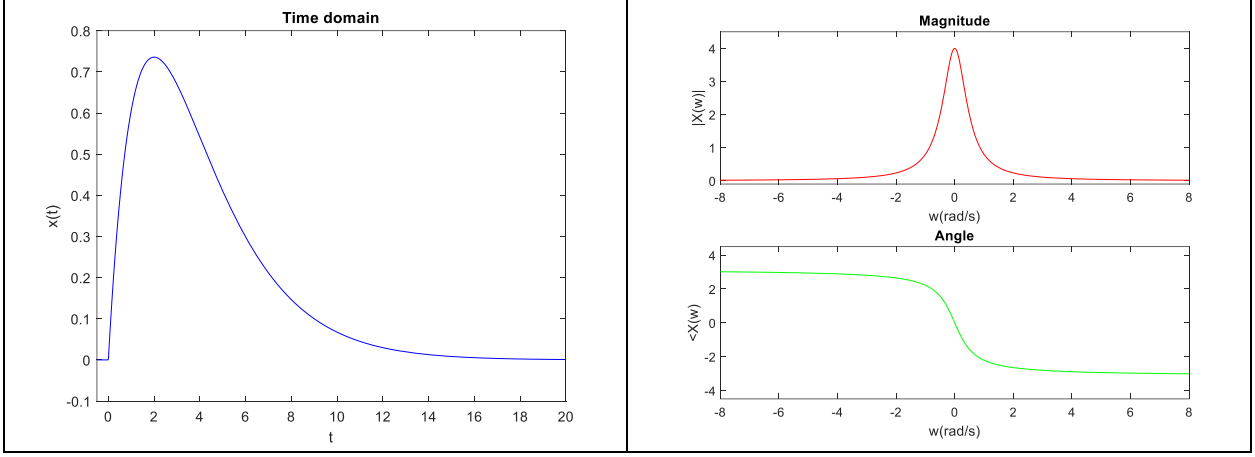




$$x(t) = e^{-2t}cos(5t)u(t)$$



$$x(t) = te^{-0.5t}u(t)$$



4. CONCLUSIÓN

En esta práctica, calculamos la Transformada de Fourier continua en MATLAB, obteniendo la magnitud y el ángulo de la señal en el dominio de la frecuencia. Esta experiencia nos permitió apreciar la importancia de la Transformada de Fourier en el análisis de señales y la potencia de MATLAB para realizar estos cálculos. En resumen, la práctica reforzó nuestro entendimiento de la Transformada de Fourier y su aplicación en el procesamiento de señales.

Diego Joel
Zuñiga Fragoso

317684

Frequency response of Filters	PRACTICA 6 FECHA 14/05/2024

1. OBJETIVO

En esta práctica, vamos a sumergirnos en el mundo de los filtros usando MATLAB. Los filtros son como porteros, deciden qué frecuencias pueden pasar y cuáles no. Los hay de varios tipos: paso bajo, paso alto, paso banda y rechazo banda. Cada uno tiene su propia función de transferencia, que es como su regla de decisión. Realizaremos ejemplos de estos filtros utilizando las funciones de MATLAB.

2. MARCO TEÓRICO

Los filtros son dispositivos electrónicos o algoritmos que permiten el paso o bloqueo selectivo de ciertas frecuencias en una señal, siendo esenciales en numerosas aplicaciones de procesamiento de señales y comunicaciones. Los filtros se clasifican según su respuesta en frecuencia, siendo los principales tipos los filtros paso bajo, paso alto, paso banda y rechazo banda. La función de transferencia H(s) de un filtro pasa bajo se define como:

$$H_{\mathrm{LPF}}(s) = K\left(\frac{\omega_c}{s+\omega_c}\right)^N$$

La función de transferencia H(s) de un filtro pasa alto se define como:

$$H_{\mathrm{HPF}}(s) = K\left(\frac{s}{s+\omega_c}\right)^N$$

Para filtros de pasa banda y rechaza banda se definen de la siguiente manera:

$$H_{\mathrm{BPF}}(\omega) = H_{\mathrm{LPF}}(\omega)H_{\mathrm{HPF}}(\omega) = K\frac{s\omega_{c1}}{(s+\omega_{c1})+(s+\omega_{c2})}$$

$$H_{\mathrm{BSF}}(s) = H_{\mathrm{LPF}}(s) + H_{\mathrm{HPF}}(s) = K\left(\frac{\omega_{c1}}{(s+\omega_{c1})} + \frac{s}{(s+\omega_{c2})}\right)$$

Los filtros pasa banda y rechazo banda de segundo orden tienen funciones de transferencia más complejas, pero pueden diseñarse utilizando técnicas como la transformación de frecuencia o el diseño directo en el dominio de Laplace.

La importancia de estudiar la respuesta en frecuencia de estos filtros radica en su aplicación en una variedad de campos, como telecomunicaciones, procesamiento de audio y video, sistemas de control, medicina, entre otros. Por ejemplo, en telecomunicaciones, los filtros se utilizan para la selección de canales, eliminación de ruido y recuperación de señales. En medicina, los filtros son esenciales en la adquisición y procesamiento de señales biomédicas, como en electrocardiogramas (ECG) y electroencefalogramas (EEG), donde ayudan a detectar y analizar patrones específicos de señales. Estas aplicaciones ilustran la importancia y versatilidad de los filtros en diversas áreas de la ingeniería y la ciencia.

3. IMPLEMENTACIÓN EN MATLAB

Se anexa el código con explicaciones

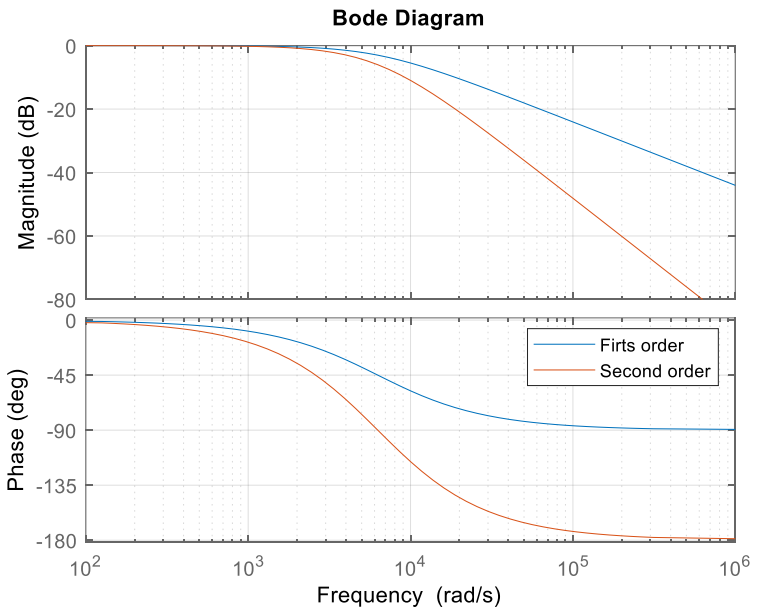
Código

```
%% Low Pass Filter

% Filtro de 1° orden
N1 = 2000*pi;
D1 = [1 2000*pi];
H1 = tf(N1, D1);
hold on;

% Filtro de 2° orden
N2 = 4000000*pi*pi;
D2 = [1 4000*pi 4000000*pi*pi];
H2 = tf(N2, D2);

% Graficacion de resultados
bode (H1,H2);
legend('Firts order','Second order');
grid on;
hold off;

%% High Pass Filter

% Filtro de 1° orden
N3 = [1 0];
D3 = [1 20*pi];
H3 = tf(N3, D3);
hold on;

% Filtro de 2° orden
N4 = [1 0 0];
D4 = [1 40*pi 100*pi*pi];
H4 = tf(N4, D4);

% Graficacion de resultados
figure, bode (H3,H4);
legend('Firts order','Second order');
grid on;
hold off;
```

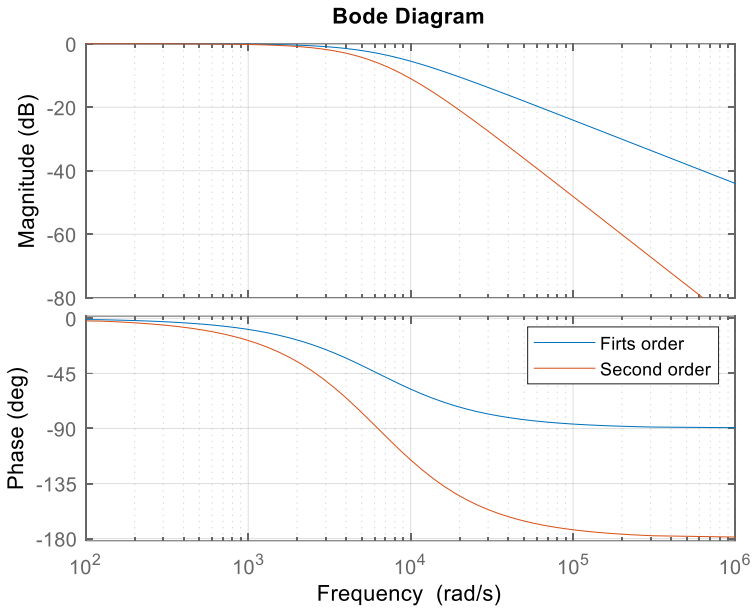
```
%% Band Pass Filter
N5 = [2000*pi 0];
D5 = [1 2020*pi 40000*pi*pi];
H5 = tf(N5, D5);

% Graficacion de resultados
figure, bode (H5);
grid on;

%% Band Stop Filter
N6 = [1 40*pi 40000*pi*pi];
D6 = [1 2020*pi 40000*pi*pi];
H6 = tf(N6, D6);

% Graficacion de resultados
figure, bode (H6);
grid on;
```

3. RESULTADOS

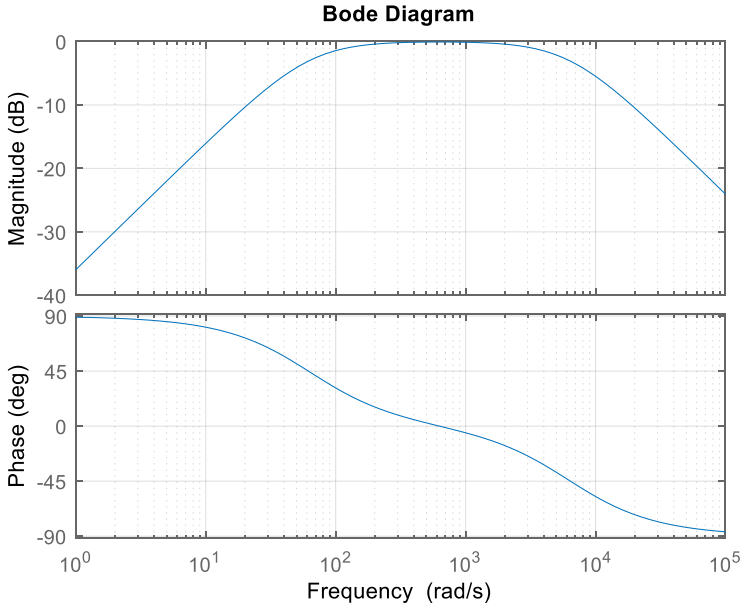
Low Pass Filter



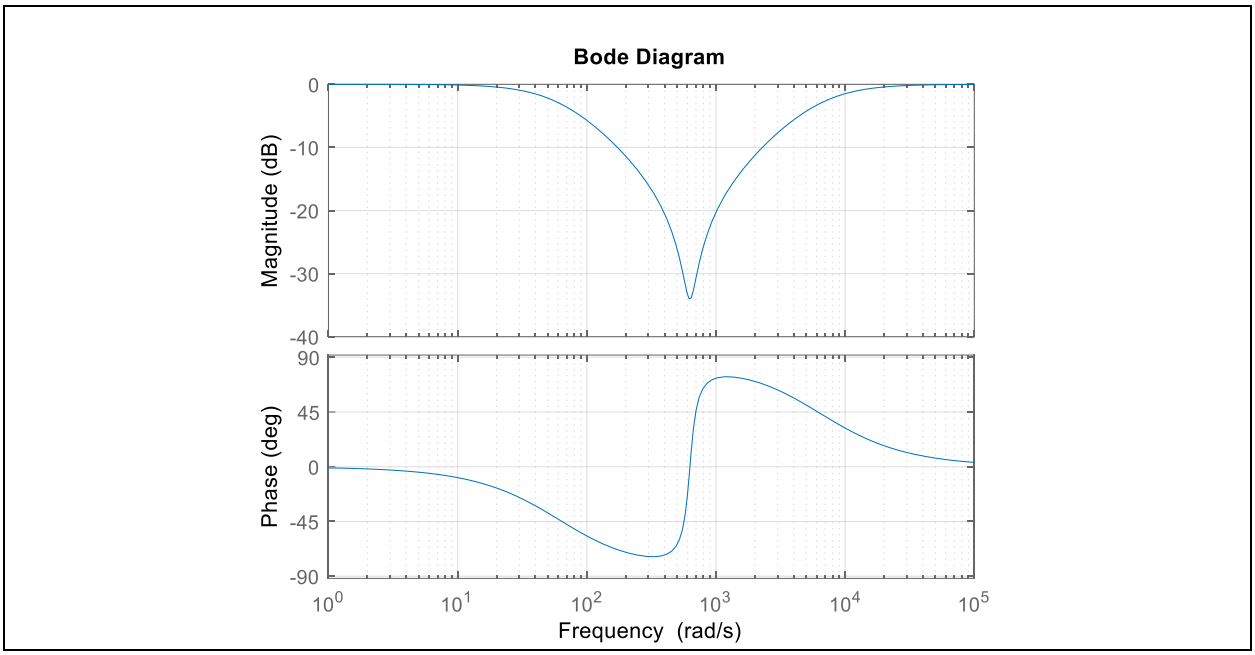
High Pass Filter



Band Pass Filter



Band Stop Filter



4. CONCLUSIÓN

En conclusión, esta práctica nos ha permitido explorar y entender profundamente los distintos tipos de filtros utilizando MATLAB. Hemos logrado representar correctamente los filtros paso bajo, paso alto, paso banda y rechazo banda, lo que nos ha proporcionado una visión clara de su funcionamiento y aplicaciones

Diego Joel
Zuñiga Fragoso

317684

LTI Systems	PRACTICA 6 FECHA 14/05/2024

1. OBJETIVO

El objetivo principal de esta práctica es profundizar en la comprensión y análisis detallado de un sistema lineal e invariante en el tiempo (LTI) causal. Utilizaremos MATLAB para modelar y analizar su comportamiento dinámico. Se aplicarán funciones específicas para identificar la función de transferencia, la respuesta al impulso y al escalón, y se analizarán los resultados obtenidos para interpretar el comportamiento del sistema LTI en diversos escenarios de señales de entrada.

2. MARCO TEÓRICO

a. Función de transferencia: La función de transferencia de un sistema en el dominio s (dominio de Laplace) se define como la razón entre la transformada de Laplace de la salida del sistema (Y(s)) y la transformada de Laplace de la entrada del sistema (X(s)), con condiciones iniciales nulas.

$$H(s) = \frac{Y(s)}{X(s)}$$

- Ceros: Se definen como los valores para los cuales (H(s) = 0).
- Polos: Se definen como los valores de (s) para los cuales (H(s) = \infty). Son las raíces del numerador y denominador de la función de transferencia, respectivamente. El cálculo de polos y ceros es muy importante en el análisis de sistemas lineales e invariantes en el tiempo (LTI). Por ejemplo, la estabilidad del sistema se puede determinar a partir de la ubicación de los polos del sistema.

b. Respuesta al impulso: La respuesta al impulso es la salida de un sistema LTI cuando la entrada es una función impulso. Puede calcularse encontrando la transformada inversa de Laplace de la función de transferencia (H(s)), es decir, [$h(t) = \mathcal{L}^{-1}[(H(s)]$].

c. Respuesta al escalón: La respuesta al escalón es la salida de un sistema LTI cuando la entrada es una función escalón. Puede representarse como (sH(s)).

d. Integral de convolución: La operación de convolución se utiliza para obtener la respuesta de salida de un sistema LTI. Si la entrada y la respuesta al impulso del sistema son (x(t)) y (h(t)), respectivamente, entonces la salida (y(t)) del sistema se obtiene mediante la operación de convolución:

$$y(t) = \int_{-\infty}^{\infty} x(\tau)h(t-\tau)d\tau$$

3. IMPLEMENTACIÓN EN MATLAB

Se anexa el código con explicaciones

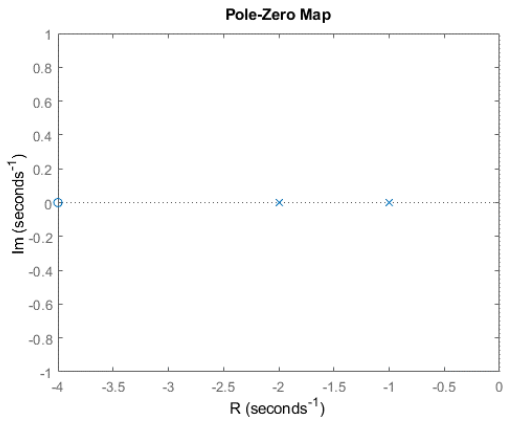
Código

```
%% Definicion de ecuacion diferencial
% Define coefficients for the differential equation
a2 = 1;  % Coeficiente de d^2y/dt^2
a1 = 3;  % Coeficiente de dy/dt
a0 = 2;  % Coeficiente de y(t)

b2 = 0;  % Coeficiente de d^2x/dt^2
b1 = 1;  % Coeficiente de dx/dt
b0 = 4;  % Coeficiente de x(t)

%% Funcion de transferencia
% Definir numerador y denominador de la funcion de transferencia
numerator = [b2 b1 b0];   % Coeficientes de x(t)
denominator = [a2 a1 a0]; % Coeficientes de y(t)

% Crear funcion de transferencia
sys = tf(numerator, denominator);

%% Respuesta por impulso y escalon

% Graficar el pole-zero map de la funcion de transferencia
figure;
pzmap(sys), xlabel('R'), ylabel('Im'), title('Pole-Zero Map');

% Graficar la respuesta por impulso
figure;
impulse(sys), title('Respuesta por impulso');

% Graficar la respuesta por escalon
figure;
step(sys),  title('Respuesta por escalon');

%% Salida del sistema
syms s t tau;
hr = ilaplace(1/(s +1));
h = hr *heaviside(t);

% Funciones de entrada
x1 = dirac(t);
x2 = heaviside(t);
x3 = heaviside(t)*heaviside(5-t);

% Funciones de salida usando la integral de convolucion
y1 = int ( subs (x1 , tau )* subs (h ,t - tau ) ,tau , - inf , inf ) ;
y2 = int ( subs (x2 , tau )* subs (h ,t - tau ) ,tau , - inf , inf ) ;
y3 = int ( subs (x3 , tau )* subs (h ,t - tau ) ,tau , - inf , inf ) ;
```

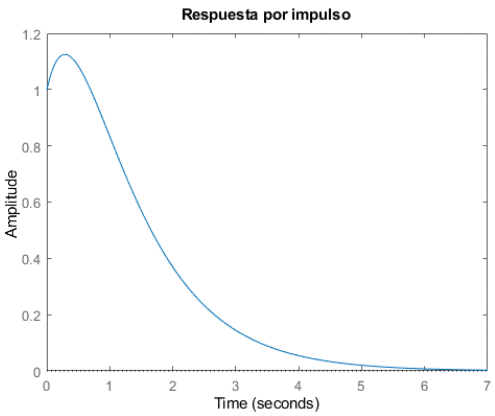
```
% Graficacion de resultados
figure;
fplot(x1);
hold on;
fplot(y1), xlabel('t'), ylabel('Amplitud'), legend('x1(t)', 'y1(t)'), xlim([
-1 10]), ylim([ -0.07 1.1]);

figure;
fplot(x2);
hold on
fplot(y2), xlabel('t'), ylabel('Amplitud'), legend('x_2(t)', 'y_2(t)'),
xlim([-1,10]), ylim([ -0.07 1.1]);

figure;
fplot(x3);
hold on
fplot(y3), xlabel('t'), ylabel('Amplitud'), legend('x_3(t)', 'y_3(t)'),
xlim([-1 10]), ylim([ -0.07 1.1]);
```

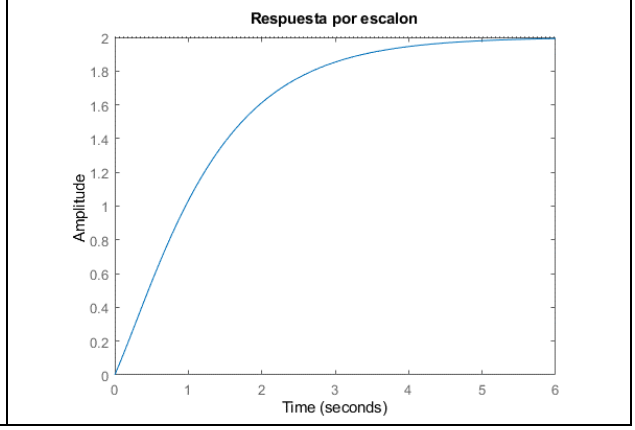
3. RESULTADOS

1. Find the transfer function of the system and plot its pole and zero map.



2. Plot its impulse and step responses.





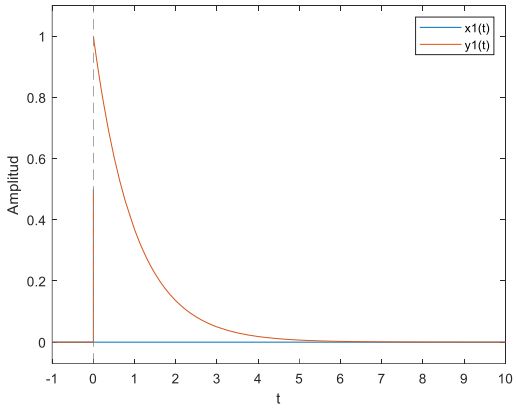
4. Find the output of the system for the following input signals, using convolution integral.

(a) $x_1(t) = \delta(t)$

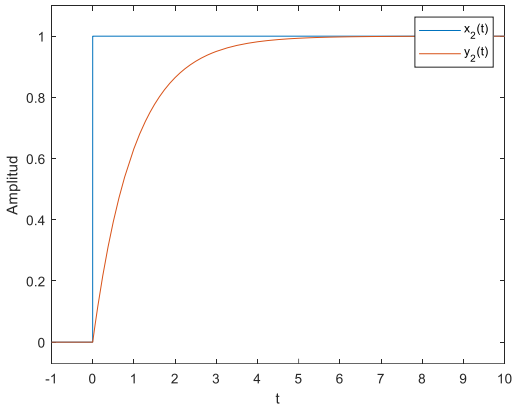
(b) $x_2(t) = u(t)$

(c) $x_3(t) = \begin{cases} 1 & \text{for} \quad 0 \le t \le 5, \\ 0 & \text{else.} \end{cases}$

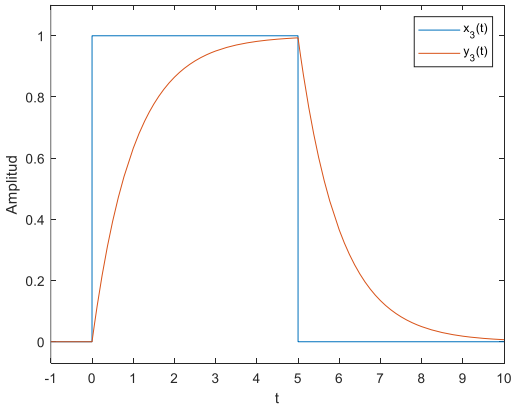
a)



b)



c)



4. CONCLUSIÓN

A través de esta práctica, pude profundizar significativamente en la comprensión y el análisis detallado de los sistemas LTI causales. Al utilizar MATLAB para modelar y graficar las respuestas y salidas de estos sistemas, logré conectar mejor la teoría vista en clase con su aplicación práctica. Al observar cómo se comportan las respuestas según los criterios específicos de cada sistema, pude apreciar y relacionar de manera más clara y comprensiva la teoría con los resultados prácticos.

Diego Joel
Zuñiga Fragoso

317684

Sampling	PRACTICA 7 FECHA 31/05/2024

1. OBJETIVO

El propósito de esta práctica es examinar el proceso de muestreo de señales utilizando el software MATLAB y analizarlo tanto en el dominio del tiempo como en el de la frecuencia. Los objetivos de aprendizaje incluyen comprender el proceso de muestreo y su relación con la reconstrucción de señales, así como reconocer la importancia del teorema de muestreo de Nyquist y la tasa de Nyquist para prevenir la distorsión y el efecto de aliasing en las señales muestreadas, y así evitar una pérdida de información.

2. MARCO TEÓRICO

Muestreo de Señales

El muestreo es el proceso de convertir una señal continua en una secuencia de valores discretos. Este procedimiento es fundamental en el procesamiento digital de señales (DSP), ya que permite que las señales analógicas sean procesadas y analizadas utilizando sistemas digitales. El muestreo se lleva a cabo mediante la toma de muestras de la señal continua a intervalos regulares de tiempo.

Teorema de Muestreo de Nyquist

El teorema de muestreo de Nyquist, también conocido como el teorema de muestreo de Shannon-Nyquist, establece que una señal continua puede ser completamente reconstruida a partir de sus muestras si la tasa de muestreo es al menos el doble de la frecuencia máxima presente en la señal. Matemáticamente, si

$$f_s \geq 2f_{max}$$

Esta tasa mínima de muestreo se conoce como la tasa de Nyquist. Cumplir con esta condición evita la distorsión y el aliasing en la señal muestreada.

Aliasing

El aliasing es un fenómeno que ocurre cuando una señal es muestreada a una tasa inferior a la tasa de Nyquist. Como resultado, las frecuencias más altas se pliegan y aparecen como frecuencias más bajas en la señal muestreada. Este efecto puede causar distorsiones significativas y pérdida de información en la señal reconstruida.

Reconstrucción de Señales

La reconstrucción de una señal muestreada implica convertir las muestras discretas de vuelta a una señal continua. Esto se logra utilizando un filtro de reconstrucción, que suele ser un filtro pasa-

bajos ideal que elimina las frecuencias superiores a la mitad de la tasa de muestreo (la frecuencia de Nyquist). Si la señal ha sido muestreada adecuadamente según el teorema de Nyquist, la señal original puede ser recuperada sin pérdida de información.

Importancia de la Tasa de Muestreo

Seleccionar una tasa de muestreo adecuada es crucial para evitar el aliasing y asegurar la precisión en la representación de la señal original. En aplicaciones prácticas, a menudo se utiliza una tasa de muestreo ligeramente superior a la tasa de Nyquist para proporcionar un margen de seguridad adicional y compensar posibles imperfecciones en el sistema de muestreo.

Aplicaciones del Muestreo

El muestreo de señales tiene numerosas aplicaciones en la ingeniería y la tecnología. Algunos ejemplos incluyen:

- Telecomunicaciones: Transmisión de voz y datos digitales.
- Audio Digital: Grabación y reproducción de música en formatos digitales como MP3 y WAV.
- Procesamiento de Imágenes: Digitalización de imágenes y videos para su almacenamiento y procesamiento.
- Sistemas de Control: Implementación de controladores digitales en sistemas de automatización y robótica.

3. IMPLEMENTACIÓN EN MATLAB

Se anexa el código con explicaciones

Código

```
%% Señal Original
% Definimos variables
Fs = 5000;
Ts = 1/ Fs;
Tmax = 0.1;
t = 0: Ts : Tmax - Ts;

% Creamos funciones
xt = sin (200* pi * t )+ 1.7* sin (140* pi*t );
X = fft (xt);
L = length (xt);
P2 = abs (X / L);
P1 = P2 (1: L /2+1);
P1 (2: end -1) = 2* P1 (2: end -1);
f = Fs *(0:( L /2) ) /L;

% Graficamos en dominio del tiempo
figure;
plot (t ,xt), xlim ([ 0 0.05]), xticks (0:0.01:0.05), ylim ([ -3 3]), xlabel
('t'), ylabel ('Amplitude ');
grid on;

% Graficamos en dominio de la frecuencia
```

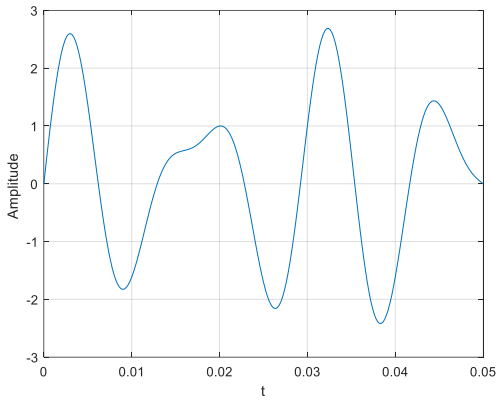
```
figure;
plot (f ,P1, 'r'), xlabel ('f(Hz)'), ylabel ('Magnitude '), xlim ([0 200]),
xticks (0:50:200), ylim ([0 2]), yticks (0:0.5:2) ;
grid on;

%% f_s = f_m
% Definimos variables
Fs = 100;
Ts = 1/ Fs;
Tmax = 0.1;
t = 0: Ts : Tmax - Ts ;

% Creamos funcione
x = sin (200* pi * t )+ 1.7* sin (140* pi*t ) ;
X = fft (x);
L = length ( x ) ;
P2 = abs ( X / L ) ;
P1 = P2 (1: L /2+1) ;
P1 (2: end -1) = 2* P1 (2: end -1) ;
f = Fs *(0:( L /2) ) /L ;

% Graficamos en dominio del tiempo
figure
stem (t ,x);
hold on;
plot (t ,x,':b'), xlim ([ 0 0.05]), xticks (0:0.01:0.05), ylim ([ -3 3]),
yticks ( -3:1:3), xlabel ('t'), ylabel ('Amplitude '), legend
('x_s1(nT)','x_s1(t)');
hold off;
grid on;

figure
plot (f , P1 ,'r'), xlabel ('f(Hz)'), ylabel ('Magnitude'), xlim ([0 200]),
xticks (0:50:200), ylim ([0 2]), yticks (0:0.5:2);
grid on;

%% f_s = 2f_m
% Definimos variables
Fs = 200;
Ts = 1/ Fs;
Tmax = 0.1;
t = 0: Ts : Tmax - Ts ;

% Creamos funcione
x = sin (200* pi * t )+ 1.7* sin (140* pi*t ) ;
X = fft (x);
L = length ( x ) ;
P2 = abs ( X / L ) ;
P1 = P2 (1: L /2+1) ;
P1 (2: end -1) = 2* P1 (2: end -1) ;
f = Fs *(0:( L /2) ) /L ;

% Graficamos en dominio del tiempo
figure
stem (t ,x);
hold on;
```

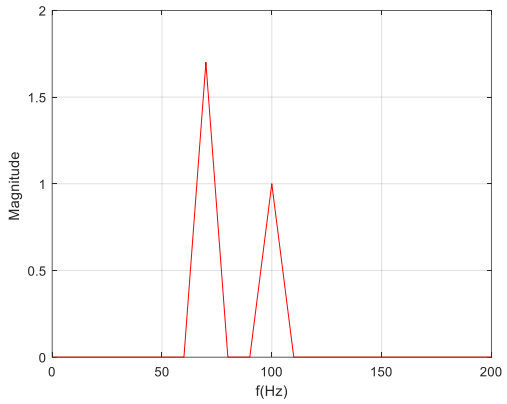
```
plot (t ,x,':b'), xlim ([ 0 0.05]), xticks (0:0.01:0.05), ylim ([ -3 3]),
yticks ( -3:1:3), xlabel ('t'), ylabel ('Amplitude '), legend
('x_s1(nT)','x_s1(t)');
hold off;
grid on;

figure
plot (f , P1 ,'r'), xlabel ('f(Hz)'), ylabel ('Magnitude'), xlim ([0 200]),
xticks (0:50:200), ylim ([0 2]), yticks (0:0.5:2);
grid on;

%% f_s = 3f_m
% Definimos variables
Fs = 300;
Ts = 1/ Fs;
Tmax = 0.1;
t = 0: Ts : Tmax - Ts ;

% Creamos funcione
x = sin (200* pi * t )+ 1.7* sin (140* pi*t ) ;
X = fft (x);
L = length ( x ) ;
P2 = abs ( X / L ) ;
P1 = P2 (1: L /2+1) ;
P1 (2: end -1) = 2* P1 (2: end -1) ;
f = Fs *(0:( L /2) ) /L ;

% Graficamos en dominio del tiempo
figure
stem (t ,x);
hold on;
plot (t ,x,':b'), xlim ([ 0 0.05]), xticks (0:0.01:0.05), ylim ([ -3 3]),
yticks ( -3:1:3), xlabel ('t'), ylabel ('Amplitude '), legend
('x_s1(nT)','x_s1(t)');
hold off;
grid on;

figure
plot (f , P1 ,'r'), xlabel ('f(Hz)'), ylabel ('Magnitude'), xlim ([0 200]),
xticks (0:50:200), ylim ([0 2]), yticks (0:0.5:2);
grid on;

%% f_s = 20f_m
% Definimos variables
Fs = 2000;
Ts = 1/ Fs;
Tmax = 0.1;
t = 0: Ts : Tmax - Ts ;

% Creamos funcione
x = sin (200* pi * t )+ 1.7* sin (140* pi*t ) ;
X = fft (x);
L = length ( x ) ;
P2 = abs ( X / L ) ;
P1 = P2 (1: L /2+1) ;
P1 (2: end -1) = 2* P1 (2: end -1) ;
```

```
f = Fs *(0:( L /2) ) /L ;

% Graficamos en dominio del tiempo
figure
stem (t ,x);
hold on;
plot (t ,x,':b'), xlim ([ 0 0.05]), xticks (0:0.01:0.05), ylim ([ -3 3]),
yticks ( -3:1:3), xlabel ('t'), ylabel ('Amplitude '), legend
('x_s1(nT)','x_s1(t)');
hold off;
grid on;

figure
plot (f , P1 ,'r'), xlabel ('f(Hz)'), ylabel ('Magnitude'), xlim ([0 200]),
xticks (0:50:200), ylim ([0 2]), yticks (0:0.5:2);
grid on;
```

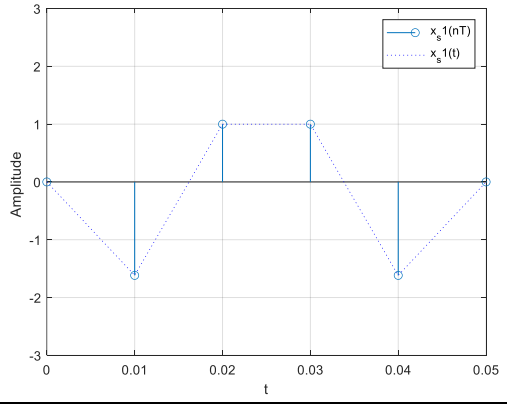
3. RESULTADOS

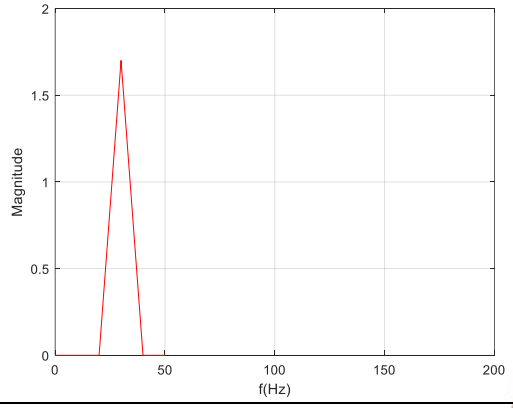
Señal original



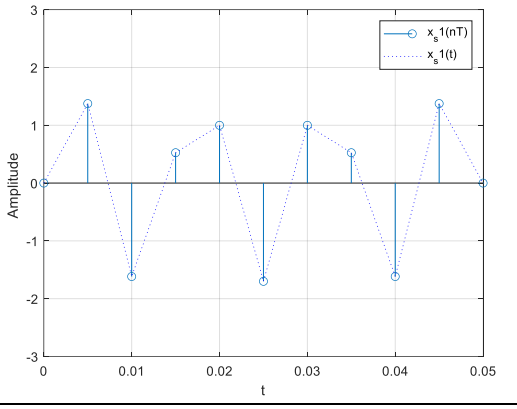


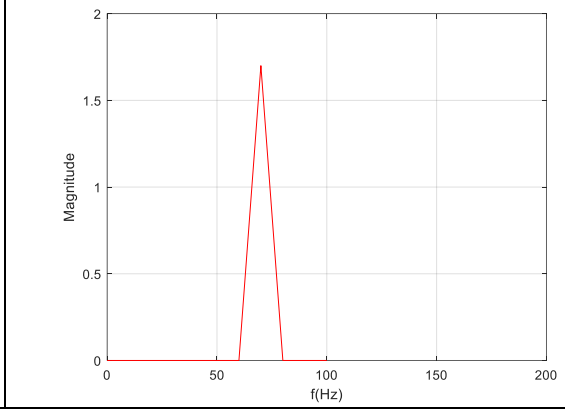
Fs = Fm



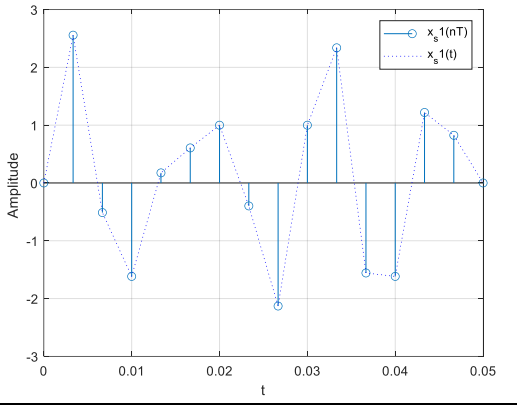


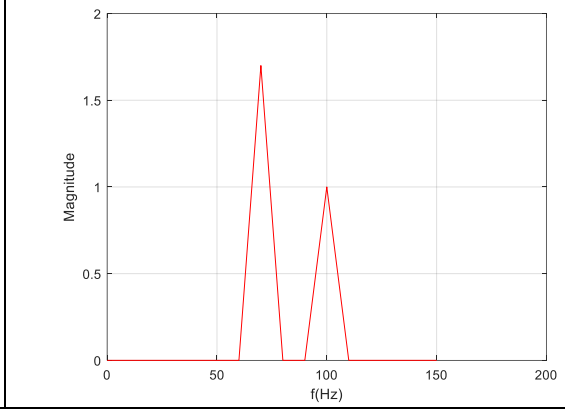
Fs = 2Fm



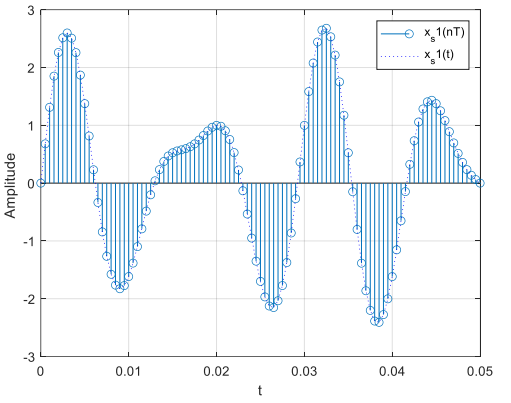


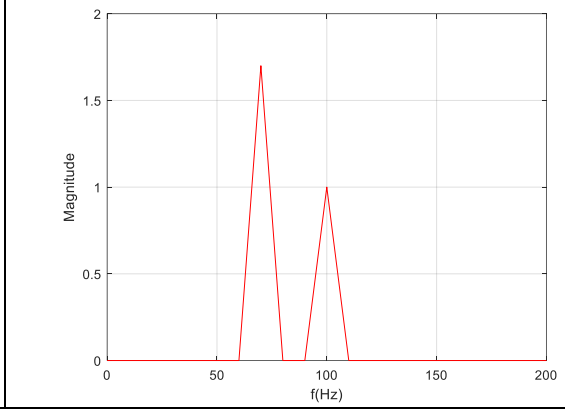
Fs = 3Fm





Fs = 20Fm





Como podemos observar, a partir de una frecuencia de muestreo 3 veces mayor que la frecuencia de la señal, aunque nuestro dominio del tiempo deja mucho que desear, nuestra transformada de Fourier representa muy bien las frecuencias que componen la señal.

4. CONCLUSIÓN

El estudio del muestreo de señales y su análisis en los dominios del tiempo y la frecuencia es esencial para comprender y aplicar técnicas de procesamiento digital. El teorema de muestreo de Nyquist y la tasa de Nyquist son conceptos fundamentales que garantizan la correcta muestreo y reconstrucción de señales, previniendo la distorsión y el aliasing. Estos principios son aplicables en una amplia variedad de campos, siendo esenciales a la hora de obtener y transferir información entre sistemas digitales y analógicos.

Diego Joel
Zuñiga Fragoso
317684

Audio Signal	PRACTICA 8 FECHA 31/05/2024

1. OBJETIVO

El objetivo de esta práctica es grabar diferentes tipos de señales de audio utilizando una aplicación específica y analizar estas señales en los dominios del tiempo y de la frecuencia mediante la Transformada de Fourier. Por lo que se generara y grabara señales de una sola frecuencia, un barrido de frecuencias y señales simples como tonos o alarmas. Posteriormente, se realizará la Transformada de Fourier para obtener la representación en el dominio de la frecuencia y visualizar los resultados mediante gráficos.

2. MARCO TEÓRICO

1. Introducción a las Señales de Audio

Las señales de audio son representaciones eléctricas de ondas sonoras que pueden ser capturadas, procesadas y reproducidas. Estas señales se caracterizan por parámetros como la frecuencia, la amplitud y la fase. En el contexto digital, el audio se convierte en una señal discreta mediante un proceso llamado muestreo, en el que se toma una serie de muestras de la señal continua a intervalos regulares.

2. Muestreo y Frecuencia de Muestreo

El muestreo es el proceso de convertir una señal de audio continua en una serie de valores discretos a intervalos de tiempo definidos. La frecuencia de muestreo (Fs) determina cuántas muestras se toman por segundo. Según el teorema de muestreo de Nyquist, para evitar la pérdida de información y evitar el aliasing, la frecuencia de muestreo debe ser al menos el doble de la frecuencia máxima presente en la señal de audio.

3. Transformada de Fourier

La Transformada de Fourier es una herramienta matemática que permite convertir una señal del dominio del tiempo al dominio de la frecuencia. La transformada descompone una señal en sus componentes sinusoidales de diferentes frecuencias, proporcionando una representación espectral de la señal. La versión discreta de esta transformada, conocida como Transformada Rápida de Fourier (FFT), es ampliamente utilizada en el procesamiento digital de señales debido a su eficiencia computacional.

4. Espectro de Magnitud

El resultado de la FFT es una serie de números complejos que representan las amplitudes y fases de las componentes sinusoidales de la señal. El espectro de magnitud se obtiene tomando el valor absoluto de estos números complejos, lo que da una idea clara de cómo se distribuyen las diferentes

frecuencias dentro de la señal. Este espectro es útil para identificar características importantes de la señal, como frecuencias dominantes y contenido armónico.

5. Análisis en el Dominio del Tiempo y de la Frecuencia

El análisis en el dominio del tiempo implica observar cómo la amplitud de la señal varía con el tiempo, lo cual es útil para comprender la estructura temporal de la señal. El análisis en el dominio de la frecuencia, por otro lado, permite identificar las diferentes frecuencias que componen la señal y cómo están distribuidas. Ambos análisis son complementarios y esenciales para una comprensión completa de las características de una señal de audio.

6. Aplicaciones Prácticas

El análisis de señales de audio es fundamental en diversas aplicaciones como la música, la comunicación, la acústica y la ingeniería de audio. Permite el desarrollo de tecnologías como el reconocimiento de voz, la mejora de la calidad del sonido, la eliminación de ruido y la síntesis de audio.

7. Implementación en MATLAB

MATLAB es una herramienta poderosa para el procesamiento y análisis de señales debido a sus capacidades avanzadas de cálculo y visualización. En esta práctica, se utilizarán funciones de MATLAB para grabar señales de audio, realizar la Transformada de Fourier y visualizar los resultados en forma de gráficos. Esto permitirá a los estudiantes aplicar conceptos teóricos en un entorno práctico y familiarizarse con técnicas de análisis de señales.

3. IMPLEMENTACIÓN EN MATLAB

Se anexa el código con explicaciones

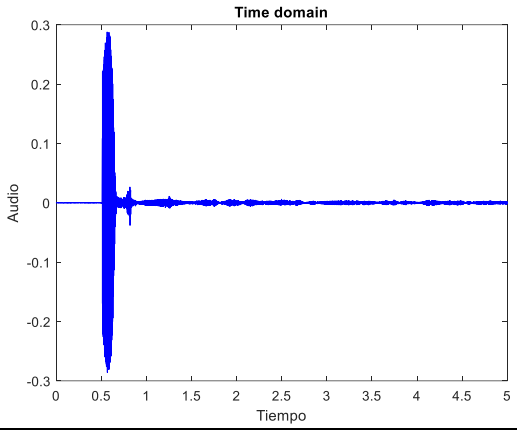
Código

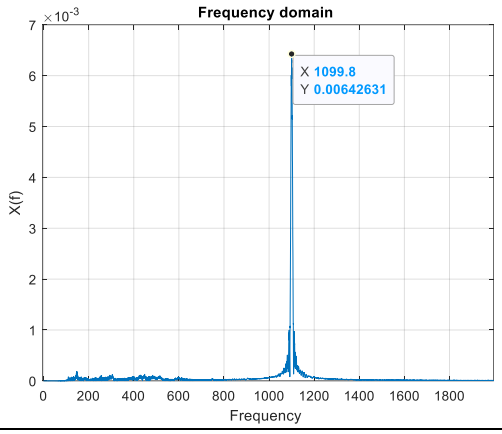
```
%% Realizamos muestreo
Fs = 44000; % Sampling frequency
Ts = 1/ Fs ; % Sampling period
Tmax = 5; % Record duration ( seconds )

t =0: Ts : Tmax - Ts ; % Time vector

rec = audiorecorder ( Fs ,16 ,1) ;
recordblocking( rec , Tmax ) ;
xt = getaudiodata ( rec ) ; % Convert audio into a vector

%% Sacamos transformada de fourier
X = fft (xt) ; % Calculate FFT
L = length ( xt ) ;
P1 = abs ( X / L ) ;
xw = P1 (1: L /2+1) ;
xw (2: end -1) = 2* xw (2: end -1) ;
w = Fs *(0:( L /2) ) / L ;

%% Graficamos resultados
```

```
figure, plot(t,xt,'b'), xlabel ('Tiempo'), ylabel ('Audio'), title('Time
domain');
figure, plot (w , xw ), title ('Frequency domain '), xlim([0 7000]) , xlabel
('Frequency '), ylabel ('X(f)'), grid on;
```

3. RESULTADOS

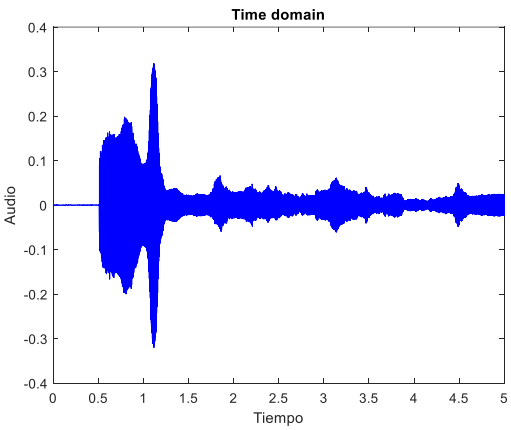
One frequency (1100 hz)

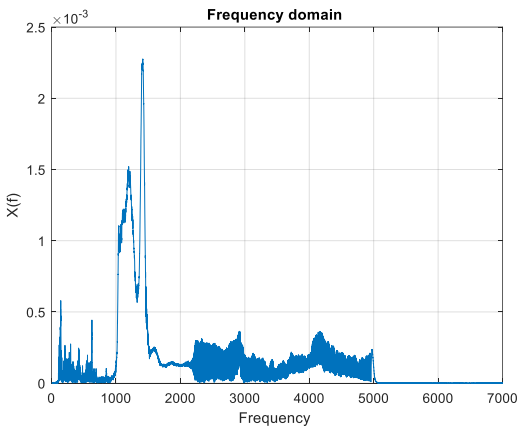




Como podemos observar se identificó perfectamente la frecuencia reproducida de 1100 Hz, aun con todo el ruido que hay en el audio.

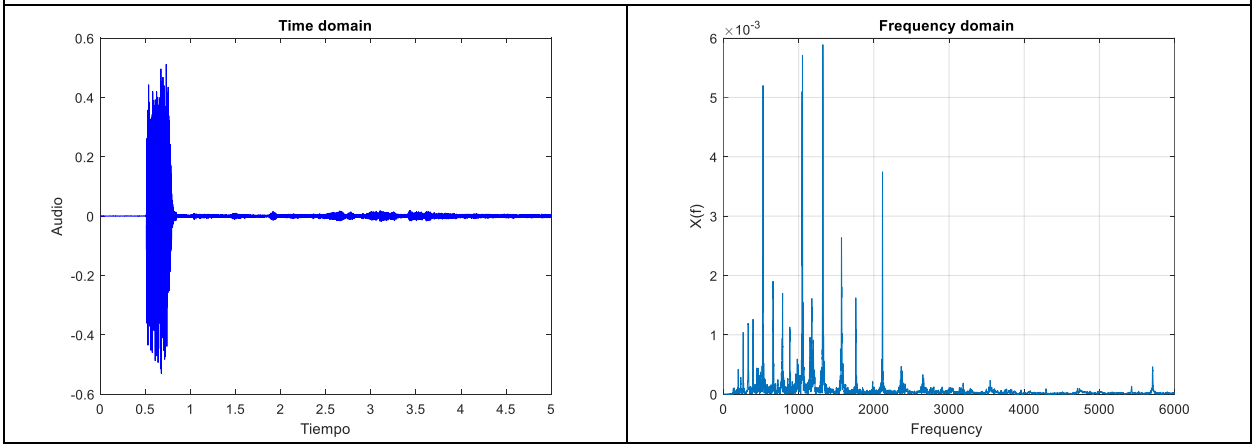
Frequency sweep (1000 hz –> 5000 hz)





Podemos notar que en el dominio de la frecuencia si se nota un aumento de magnitud en el rango de frecuencias reproducido.

Alarma de cellular



Podemos ver de que frecuencias está compuesta el sonido de alarma reproducido, que va de frecuencias desde 0 a 2500 hz.

4. CONCLUSIÓN

En esta se ha logrado el objetivo de grabar y analizar diferentes tipos de señales de audio utilizando herramientas digitales. A través de la generación y grabación de señales de una sola frecuencia, un barrido de frecuencias y señales simples como tonos o alarmas se ha aplicado la Transformada de Fourier para transformar estas señales del dominio del tiempo al dominio de la frecuencia. Los resultados obtenidos han sido los esperados, y nos demuestra una de las muchísimas aplicaciones de esta transformada en nuestro mundo moderno.