

Maestría en Ingeniería Eléctrica especialización Telecomunicaciones

Comunicaciones Digitales

**Tarea #0**

***Luis Emilio Tonix Gleason***

***Fernando Alberto Madera Torres***

*20/03/2022*

***Dr. Ramon Michel Parra***

Tabla de contenido

[Ejercicio 1 3](#_Toc98919730)

[Ejercicio 2 3](#_Toc98919731)

[Ejercicio 3 4](#_Toc98919732)

[Generacion de ventana cuadrada 5](#_Toc98919733)

[Ejercicio 4 10](#_Toc98919734)

[Radian domain 10](#_Toc98919735)

[Sampling Frequency domain 11](#_Toc98919736)

[Lectura de Audio 13](#_Toc98919737)

[Aplicacion de filtros en audio 14](#_Toc98919738)

[Conversión y comparación en tiempo 15](#_Toc98919739)

[Ejercicio 5 17](#_Toc98919740)

[Observaciones 20](#_Toc98919741)

[Ejercicio 6 21](#_Toc98919742)

[*overlap and add* 21](#_Toc98919743)

[*overlap and save* 22](#_Toc98919744)

# Ejercicio 1

Relación entre las series de Fourier (FS), la transformada de Fourier (FT), La transformada de Fourier en tiempo discreto (FTDT) y la transformada discreta de Fourier (FDT).

**Series de Fourier**

Cualquier forma de onda periódica está formada por un componente promedio y una serie de ondas senoidales y cosenoidales relacionadas armónicamente, donde una armónica es un múltiplo entero de la frecuencia fundamental.

La frecuencia fundamental es la primera armónica y es igual a la frecuencia o rapidez de repetición de la forma de onda. La frecuencia fundamental es la mínima necesaria para representar la forma de onda.

La serie de Fourier se utiliza en el análisis de señales para cambiar una señal en el dominio del tiempo a una señal en el dominio de la frecuencia, se puede obtener una serie de Fourier para cualquier función periódica.

**Transformada de Fourier**

Puede ser aplicada a una señal aperiódica, este limitando un tiempo de observación y dando por hecho que esa ventana observable sea tomada como un periodo.

**Transformada de Fourier en Tiempo Discreto**

Es la transformada para una señal discreta, tiene relación con la transformada z, la transformada de tiempo discreto vive en el circulo unitario de la transformada Z.

**Transformada Discreta de Fourier**

Se muestrea una señal en el dominio del tiempo, en tiempo discretos. Las muestras se guardan en una memoria para realizar un algoritmo que calcula la transformación.

# 

# Ejercicio 2

Dos funciones temporales (No senoidales) y obtener su TF mediante el cálculo directo y compruebe su aproximación mediante la TDF correctamente escalada.

Imagen que contiene Gráfico de líneas

Descripción generada automáticamente

Gráfico, Gráfico de líneas

Descripción generada automáticamente

La norma de este pulso es este sería el escalamiento según su longitud

# Ejercicio 3

Diseñar un filtro pasa altas de 51 coeficientes a partir de 8pi/10

Filtro en la frecuencia de a *π* con el filtro hasta Forma

Descripción generada automáticamente con confianza media

## Generacion de ventana cuadrada

Se define un tiempo de observacion de 625 segundos por ser  el rango es 

bound =[-pi,pi]; % 2pi range lenght

obs = 100 % 100 segundos

obs = 100



%Puntos de corte del filtro

cut\_points = [-pi -8\*pi/10 8\*pi/10 pi];

% tiempo de observacion 100s \* 2pi = 625 muestras

space = bound(1):1/obs:bound(2)+1/100;

%inicia señal de filtro en 0

rad\_space = zeros(1,length(space));

%parametros limite en la señal

%cut\_freq\_id se utilizara despues para el corte en tiempo tambien

cut\_freq\_id = [];

%Counter for cutting points

cnt = 0;

%Iterate over all cutting points and when reach a pair of poins mod(cnt,2)

%assign them the frecuency square window.

for section = cut\_points % Ventana limits

temp = find(space == interp1(space,space,section,'nearest'));%index in linespace

cut\_freq\_id = [cut\_freq\_id temp]; % add index of limits

cnt = cnt +1;

if(mod(cnt,2) == 0) % Cnt is a pair of cutting values

rad\_space(cut\_freq\_id(cnt-1):cut\_freq\_id(cnt)) = 1;

end

end

%Parametros de la grafica

plot(space,rad\_space)

xlim([-pi pi])

xticks([-pi 0 pi 2\*pi 3\*pi])

xticklabels({'-\pi','0','\pi'})

xlabel('rad/s')

Gráfico, Histograma, Gráfico de burbujas

Descripción generada automáticamente

Convertimos las ventanas de frecuencia a tiempo.

ventana\_tiempo = real(ftot(rad\_space));

plot(ventana\_tiempo)

Gráfico, Gráfico de cajas y bigotes

Descripción generada automáticamente

Generación de ventanas en tiempo, con 51 coeficientes. Y una venta de hamming

ventanacuad = zeros(1,length(space));

coeff = 51;

middle = (length(space))/2;

left = int32((middle-coeff/2));

right = int32((middle+coeff/2));

ventanacuad(left:right) = hamming(coeff+1)';

plot(ventanacuad)

Gráfico, Histograma

Descripción generada automáticamente

filtro\_cuad=ventana\_tiempo.\*ventanacuad;

plot(space,rad\_space);

hold on

plot(space,real(ttof(filtro\_cuad))/max(real(ttof(filtro\_cuad))))

xlim([-pi pi])

xticks([-pi 0 pi 2\*pi 3\*pi])

xticklabels({'-\pi','0','\pi'})

xlabel('rad/s')

legend({'filtro ideal','filtro hamming'},'Location','southwest')

hold off

Forma

Descripción generada automáticamente con confianza baja

# Ejercicio 4

Diseñar 4 filtros de 45 coeficientes, cada uno con las frecuencias positivas de corte ideales en

filtro1=Forma

Descripción generada automáticamente con confianza baja

filtro2=Forma

Descripción generada automáticamente con confianza media

filtro3=Forma

Descripción generada automáticamente con confianza media

filtro4=Forma

Descripción generada automáticamente con confianza media

Definimos un tiempo de observación de 1 seg. Esto quiere decir que si muestreamos a 8000 muestras por segundo tendremos 1/8000 como tiempo de observación.

## Radian domain

bound =[-pi,pi];

space = bound(1)-1/100:1/100:bound(2)+1/100;

%generate 4 spaces

rad\_space = zeros(4,length(space));

range = 0:pi/4:pi;

% 4 graphs to be ploted

figure(1)

t = tiledlayout(2,2);

side =[-1,1];

for n=2:1:5

%index in space

for m=side % negative, positve side

start\_filter = find(space == interp1(space,space,m\*range(n-1),'nearest'));

end\_filter = find(space == interp1(space,space,m\*range(n),'nearest'));

%4d tensor of filters, n-1 acces to layer of tensor.

rad\_space(n-1,min(start\_filter,end\_filter):max(start\_filter,end\_filter)) = 1;

end

%plotting stuff

nexttile

plot(space,rad\_space(n-1,:));

xlim([-pi pi])

xticks([-pi -pi\*3/4 -pi/2 -pi/4 0 pi/4 pi/2 pi\*3/4 pi])

xticklabels({'-\pi','-3\pi/4','-\pi/2','-\pi/4','0','\pi/4','\pi/2','3\pi/4','\pi'})

xlabel('rad/s')

end

Gráfico

Descripción generada automáticamente

## Sampling Frequency domain

Dividimos el eje por . Esto para hacer la conversion de  a Forma

Descripción generada automáticamente con confianza media

 , Forma

Descripción generada automáticamente con confianza baja

Posteriormente multiplicamos por la frequencia de muestreo  para obtener el maximo espacio libre de alias .

Det tal forma que la venta queda de

Forma

Descripción generada automáticamente con confianza media

Fs = 8000; % sampling frequency

Coeff = 45; % num of coeffiientes

Forma

Descripción generada automáticamente con confianza media

bound\_fs = (bound./(2\*pi)).\*(Fs);

1 second of observation time

space\_fs = bound\_fs(1):1:bound\_fs(2); % 1 second

%tensors space

filter\_fs = zeros(4,length(space\_fs));

filter\_time = zeros(4,length(space\_fs));

windows = zeros(4,length(space\_fs));

range\_fs = (range./(2\*pi)).\*Fs;

% 4 graphs to be ploted

figure(2)

dt\_fs = tiledlayout(2,2);

side =[-1,1]; % positive and negative side of the spectra

for n=2:1:5

%index in space

for m=side % negative, positve side

start\_filter = find(space\_fs == interp1(space\_fs,space\_fs,m\*range\_fs(n-1),'nearest'));

end\_filter = find(space\_fs == interp1(space\_fs,space\_fs,m\*range\_fs(n),'nearest'));

%4d tensor of filters, n-1 acces to layer of tensor.

filter\_fs(n-1,min(start\_filter,end\_filter):max(start\_filter,end\_filter)) = 1;

end

%time signal of square filter

filter\_time(n-1,:) = real(ftot(filter\_fs(n-1,:)));

%in this case time signal is centered at 0 so 45 coeff means -22,0,22

middle = (length(space\_fs))/2;

left = int32((middle-Coeff/2));

right = int32((middle+Coeff/2));

%inserting hamming filtering

windows(n-1,left:right) = hamming(Coeff+1)';

%convolution

windows(n-1,:) = windows(n-1,:).\*filter\_time(n-1,:);

%normalization

windows(n-1,:) = real(ttof(windows(n-1,:)))/max(real(ttof(windows(n-1,:))));

%plotting stuff

nexttile

plot(space\_fs,filter\_fs(n-1,:));

hold on;

plot(space\_fs,windows(n-1,:));

legend({'filtro ideal','filtro hamming'},'Location','southwest')

hold off;

end

Interfaz de usuario gráfica, Diagrama

Descripción generada automáticamente

## Lectura de Audio

figure(7)

[y, audioFS] = audioread("Audios/grape-juice.wav");

Ajustamos el sample rate para garantizar los 8000 samples/sec

s\_down\_resample = resample(y, Fs, audioFS);

m\_t = s\_down\_resample(1:(Fs+1)); %cut one second final message signal

m\_t\_f = real(ftot(m\_t));

plot(space\_fs,m\_t\_f);

Gráfico, Histograma

Descripción generada automáticamente

## Aplicacion de filtros en audio

Simplmente queda como la multiplicacion en frecuencia de las dos señales.

figure(4)

filtered\_signals = [m\_t\_f m\_t\_f m\_t\_f m\_t\_f]';

dt\_filtered = tiledlayout(2,2);

for n=1:1:4

filtered\_signals(n,:) = filtered\_signals(n,:).\*windows(n,:);

%plotting stuff

nexttile

plot(space\_fs,m\_t\_f);

hold on;

plot(space\_fs,filtered\_signals(n,:));

legend({'Señal original','Señal filtrada'},'Location','southwest')

hold off;

end

Escala de tiempo

Descripción generada automáticamente

## Conversión y comparación en tiempo

figure(5)

filtered\_time = real (ftot(filtered\_signals));

dt\_filtered\_time = tiledlayout(2,2);

time = 0:1/Fs:1;

for n=1:1:4

filtered\_time(n,:) = rescale(filtered\_time(n,:),min(m\_t),max(m\_t));

%plotting stuff

nexttile

plot(time,m\_t);

hold on;

plot(time,filtered\_time(n,:));

legend({'Señal original','Señal filtrada'},'Location','southwest')

hold off;

end

Diagrama

Descripción generada automáticamente

# Ejercicio 5

Un diferenciador realiza la operación y(t) = dx(t)/dt, que en la frecuencia es la multiplicación por jw, por lo tanto, la operación de diferenciación se puede ver como el filtrado con un filtro cuya función de transferencia es jw.

bound = [-pi,pi];

w = bound(1)-1/100:1/100:bound(2)+1/100;

derivator = real(ftot(1i\*w));

plot(w,derivator);

Gráfico, Gráfico de cajas y bigotes

Descripción generada automáticamente

1. Diseñe un filtro FIR de 41, 51 y 61 coeficientes que haga una operación de derivada. (puede utilizar cualquier ventana)

Se crear un filtro de hamming muestreando 41,51,61 coeffiecientes.

Coeff = [41,51,61];

FIR = zeros (length(Coeff),length(w));

%inserting hamming filtering

middle = (length(w))/2;

figure(1);

dt\_fs = tiledlayout(2,2);

for n=1:1:3

left = int32((middle-Coeff(n)/2));

right = int32((middle+Coeff(n)/2));

FIR(n,left:right) = hamming(Coeff(n)+1)';

FIR(n,:) = derivator.\*FIR(n,:);

nexttile

plot(FIR(n,:));

axis([-30+middle middle+30 -1 1])

%normalization

FIR(n,:) = ttof(FIR(n,:))/max(ttof(FIR(n,:)));

end

Gráfico, Histograma

Descripción generada automáticamente

2. Dada las aproximaciones con los filtros diferenciadores h1[n] = {-1,1}, h2[n] = {0.5, 0,-0,5}. Con el cero en el primer coeficiente en ambos filtros. Obtenga de ambos su función de transferencia analítica.

figure(3)

time = -1:1:3;

h\_1 = [0,-1,1,0,0];

h\_2 = [0,.5,0,-5,0];

plot(time,h\_1);

hold on;

plot(time,h\_2);

hold off;

Gráfico, Gráfico de líneas

Descripción generada automáticamente

3. Compare las funciones de transferencia obtenida por a) y b) y concluya que tanto se aproximan al derivador real.

4. Introduzca una señal coseno en múltiplos de  desde 0 a *π* (es decir, su frecuencia angular discreta) a los filtros anteriores, y verifique a partir de las señales generadas por simulación si los filtros efectivamente realizan la operación de derivación: justifique sus conclusiones. (grafique la señal de salida y la entrada en un mismo cuadro para apoyar sus observaciones)

Forma

Descripción generada automáticamente con confianza media

Forma

Descripción generada automáticamente con confianza media

Forma

Descripción generada automáticamente con confianza media

signal = cos(10\*w);

Aplicacion de filtro derivador

figure(6)

derivative= real(ftot(ttof(signal).\*FIR(1,:)));

derivative= rescale(derivative,-1,1);

plot(w,derivative);

hold on;

plot(w,signal);

hold off;

axis([0 pi -1 1])

legend({'derivada','original'},'Location','southwest')

Gráfico, Gráfico de líneas, Histograma

Descripción generada automáticamente

### Observaciones

Pese al filtro estar normalizado en frecuencia y tener un factor de coeficientes considerables fue necesario un escalamiento de la señal, dado que la señal derivada salía muy pequeña.

# Ejercicio 6

Explicar en qué consiste la técnica *overlap and add* y *overlap and save* para el filtrado de secuencias arbitrariamente largas mediante el filtrado por multiplicación en la frecuencia.

## *overlap and add*

La entrada se divide en bloques que no se solapan entre si xm(n) cada longitud L.

Cada bloque de entrada xm(n) es individualmente filtrado y recibido para producir la salida del bloque ym(n)

Imagen que contiene objeto, antena

Descripción generada automáticamente

Imagen que contiene Gráfico

Descripción generada automáticamente

Diagrama

Descripción generada automáticamente

**Algoritmo**

1. Separar la señal x(n) en bloques que no se solapen entre si xm(n) de longitud L.
2. Hacer Zero Pad h(n) para tener la longitud N=L+M-1
3. Tomar la N-DFT de h(n) para dar H(k), k=0,1,2,3, … N-1
4. Para cada boque m:
   1. Zero pad xm(n) para que sea de longitud N=L+M-1
   2. Obetenr la N-DFT de xm(n) para tener Xm(k), k=0,1,2,3, … N-1
   3. Multiplicar: Ym(k) = Xm(k)\* Hm(k), k=0,1,…N-1
   4. Obtener la N-IDFT de Ym(k) para dar ym(n) n=0,1,…,N-1
5. De y(n) con superposición del ultimo M-1 muestras de ym(n) con el primer M-1 muestras de ym+1(n) y sumando el resultado.

## *overlap and save*

Gráfico, Gráfico de cajas y bigotes

Descripción generada automáticamente

Diagrama

Descripción generada automáticamente

**Algoritmo**

1. Insertar M-1 ceros en el principio de la secuencia de entrada x(n).
2. Separar en partes la señal de entrada en bloques superposicionados xm(n) de longitud N= L+M-1 donde la longitud del solapamiento es M-1
3. Zero pad h(n) para que sea de longitud N=L+M-1
4. Obtener N-DFT de h(n) para poder obtener H(k), k=0,1, … N-1
5. Para cada bloque m:
   1. Obtener la N-DFT de xm(n) para dar Xm(k) n=0, 1,…,N-1
   2. Multiplicar: Ym(k) = Xm(k)\* Hm(k), k=0, 1,…N-1
   3. Obtener la N-IDFT de Ym(k) para tener ym(n), k=0,1,2,3, … N-1
   4. Descartar la primera M-1 puntos de cada bloque de salida ym(n).
6. Formar y(n) sumando el restante (por ejemplo, del último) L muestras por cada bloque de ym(n).