ELO 314 - Procesamiento Digital de Señales Laboratorio 4: Transformada Discreta de Fourier en MatLab

Preparado por Juan Aguilera e-mail: Juan.Aguileraca@sansano.usm.cl
Cristóbal Huidobro e-mail: cristobal.huidobro@sansano.usm.cl

I. Cálculo de la DFT para señales simples

1) Para el calculo de la expresión analítica tenemos que

$$\omega_0 = \frac{2\pi f}{f_s} = \frac{2\pi \cdot 100}{5000}[rad/muestra]$$

Por lo que las señales quedan expresadas de la siguiente manera:

- $x_1[n] = sin[\omega_0 n] = \frac{e^{j\omega_0 n} - e^{-j\omega_0 n}}{2j}$
- $x_2[n] = cos[\omega_0 n] = \frac{e^{j\omega_0 n} + e^{-j\omega_0 n}}{2}$

Por lo que la DTFT de la señal $x_1[n]$ se calcula de la siguiente manera:

$$\begin{aligned} X_1(\omega) &= \frac{1}{2j}\sum_{n=-\infty}^{\infty}\left(e^{j\omega_0 n} - e^{-j\omega_0 n}\right)e^{-j\omega n} \\ \Rightarrow &\frac{1}{2j}\sum_{n=-\infty}^{\infty}\left(e^{-jn(\omega-\omega_0)} - e^{-jn(\omega+\omega_0)}\right) \\ \Rightarrow &\frac{2\pi}{2j}\left(\delta(\omega-\omega_0) - \delta(\omega+\omega_0)\right) \\ \Rightarrow &\frac{2\pi j}{2}\left(\delta(\omega+\omega_0) - \delta(\omega-\omega_0)\right), \quad \omega \in [0, 2\pi) \end{aligned}$$

De forma similar, para $x_2[n]$ tenemos:

$$\begin{aligned} X_2(\omega) &= \frac{1}{2}\sum_{n=-\infty}^{\infty}\left(e^{j\omega_0 n} + e^{-j\omega_0 n}\right)e^{-j\omega n} \\ \Rightarrow &\frac{1}{2}\sum_{n=-\infty}^{\infty}\left(e^{-jn(\omega-\omega_0)} + e^{-jn(\omega+\omega_0)}\right) \\ \Rightarrow &\frac{2\pi}{2}\left(\delta(\omega-\omega_0) + \delta(\omega+\omega_0)\right), \quad \omega \in [0, 2\pi) \end{aligned}$$

Asumiendo la duración real de la señal tenemos para $x_1[n]$:

$$\begin{aligned} X_1(\omega) &= \frac{1}{2j}\sum_{n=0}^{500}\left(e^{j\omega_0 n} - e^{-j\omega_0 n}\right)e^{-j\omega n} \\ \Rightarrow &\frac{1}{2j}\sum_{n=0}^{500}\left(e^{-jn(\omega-\omega_0)} - e^{-jn(\omega+\omega_0)}\right) \\ \Rightarrow &X_1(\omega) = \begin{cases} -250j & \text{si } \omega = \omega_0 \\ 250j & \text{si } \omega = -\omega_0 \\ 0 & e.o.c \end{cases} \end{aligned}$$

De forma similar, para $x_2[n]$ tenemos que:

$$\begin{aligned} X_2(\omega) &= \frac{1}{2}\sum_{n=0}^{500}\left(e^{j\omega_0 n} + e^{-j\omega_0 n}\right)e^{-j\omega n} \\ \Rightarrow &\frac{1}{2}\sum_{n=0}^{500}\left(e^{-jn(\omega-\omega_0)} + e^{-jn(\omega+\omega_0)}\right) \\ \Rightarrow &X_2(\omega) = \begin{cases} 250 & \text{si } \omega = \pm\omega_0 \\ 0 & e.o.c \end{cases} \end{aligned}$$

Como consecuencia de tomar la duración real de la señal, la DTFT depende de un factor que depende de el numero de muestras.

2) Se aplica el comando de $MATLAB\ fft$, obteniendo como resultadolo que se aprecia en la Figura 1.

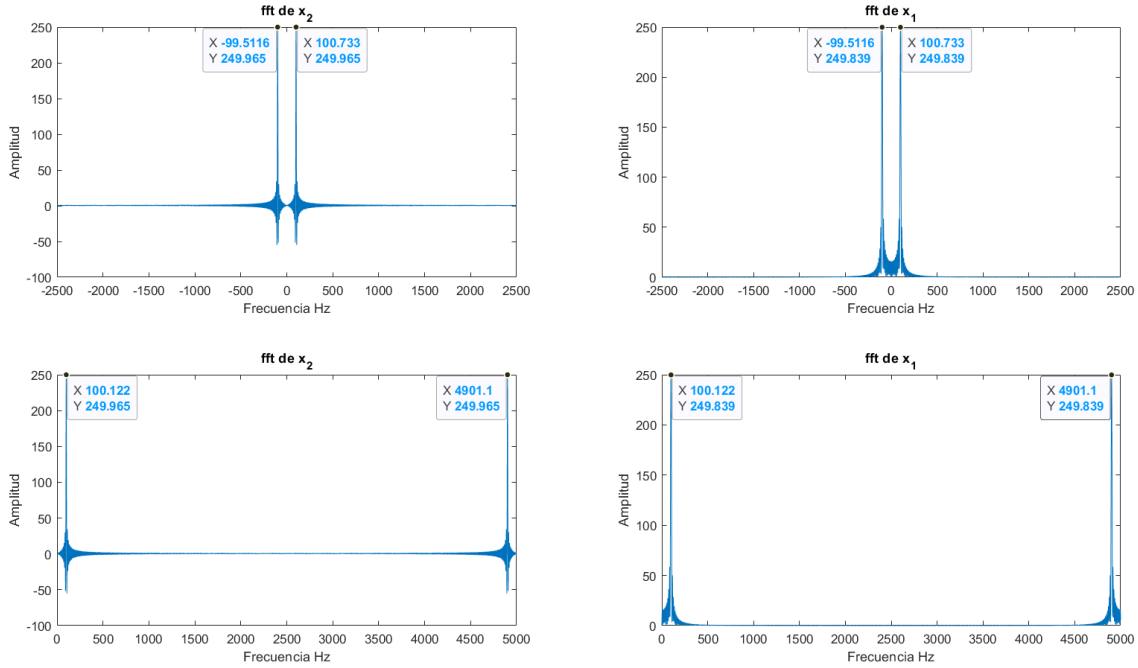

Figura 1: fft de señales x_1 y x_2 para diferentes intervalos de frecuencia.

La magnitud esperada, correspondiente a la suma de los puntos donde $\omega = \pm\omega_0$, es de 250. Se esperaría que para una señal de tiempo infinito, el resultado de la DTFT se comporte como un delta, es decir, que tenga amplitud infinita en los puntos donde $\omega = \pm\omega_0$. A pesar de aumentar el numero de muestras a $N = 4069$, la amplitud en $\omega = \pm\omega_0$ no debería aumentar en más de 250 unidades, esto debido a que la señal utilizada solo tiene un largo de 500 muestras, por lo que, aunque se aumenten las muestras totales, solo 500 de ellas sumaran en la amplitud de los deltas.

Sin embargo se observa que la magnitud es un poco menor a 250, además de presencia de energía en los valores que rodean a ω_0. Esto se debe a que como la señal es de menor largo que el numero de muestras requeridas para la DFT, esta rellena las muestras restantes con 0, lo que sería equivalente a enventanar la señal con una ventana cuadrada.

3) Los valores de magnitud, parte real e imaginara vs frecuencias se muestran a continuación.

a) Magnitud vs Frecuencia

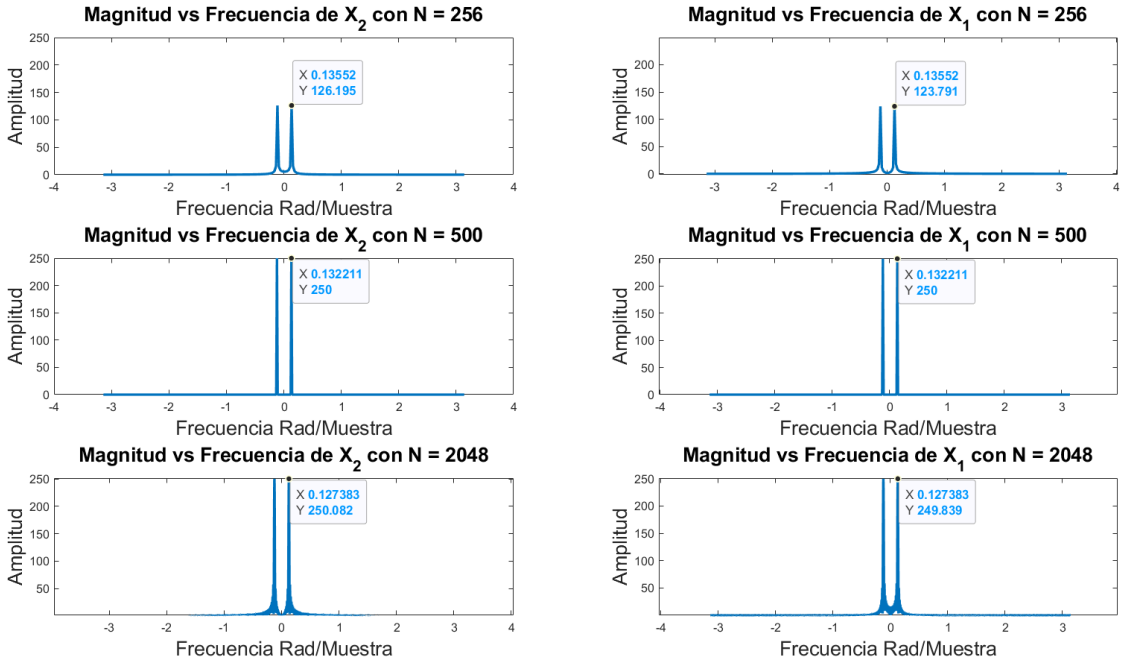

Figura 2: Magnitud vs Frecuencia para diferentes valores de N.

b) Parte Real vs Frecuencia

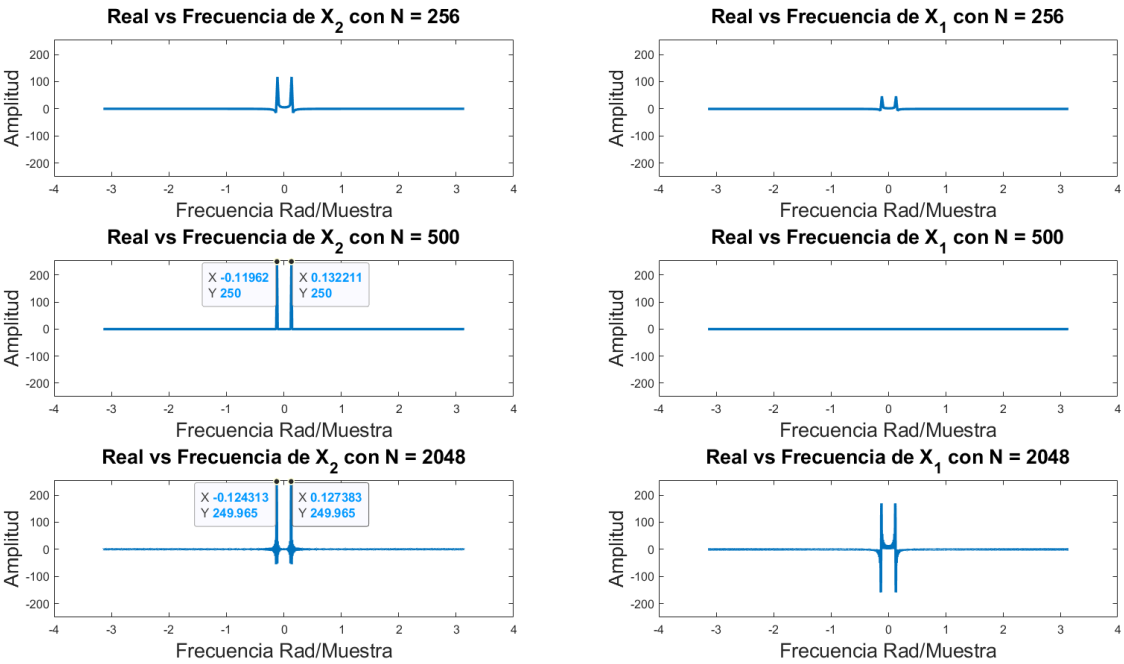

Figura 3: Real vs Frecuencia para diferentes valores de N.

c) Parte Imaginaria vs Frecuencia

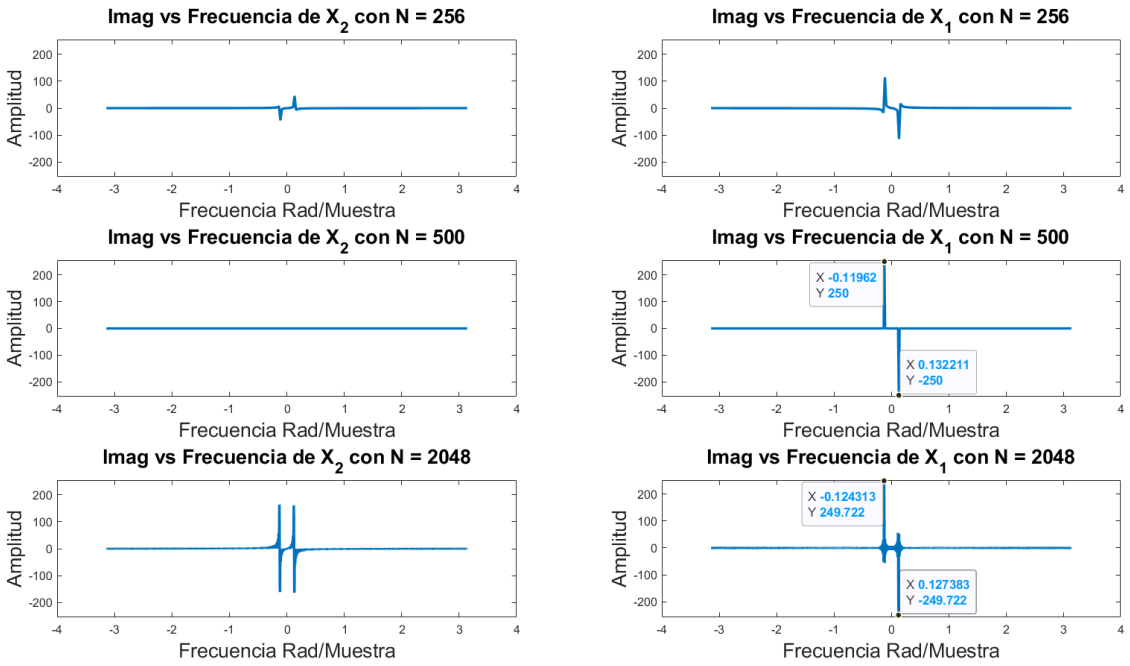

Figura 4: Img vs Frecuencia para diferentes valores de N.

- Para $N = 256$ se observa que la magnitud es inferior a los 250 esperados y se acerca más a 128, esto se debe a que solo se consideran las primeras 256 muestras para el calculo de la DFT. Sin embargo la amplitud es menor a 128 para ambas señales. Lo cual se debe a que 256 no es una cantidad finita de períodos de la señal, por lo que al ser la DFT períodica, se genera una distorsión en el "corte" entre períodos. Esto también provoca que la DFT tenga tanto parte real como imaginaria en los dos casos.
- Para $N = 500$ se observa que, tal como es de esperarse, la magnitud es de 250, $X1$ tiene solo parte real y $X2$ tiene solo parte imaginaria. Generando las DFT esperadas para un coseno y un seno respectivamente. Esto se debe a que 500 muestras es un número completo de períodos de estas señales.
- Para $N = 2048$, al tomar más muestras que las contenidas en la señal, la DFT rellena con ceros las muestras faltantes, generando distorsión en la amplitud y fase de la señal. Se observa que la amplitud es cercana a los 250 esperados pero cada peak está rodeado de una sinc por el enventanamiento cuadrado.

II. Estimación de frecuencias en presencia de ruidos

Se genera la siguiente señal:

$$s_1 = \frac{1}{2}cos(2\pi 100t) + 1{,}5cos(2\pi 500t) \tag{1}$$

A la cual se le suma el ruido de la siguiente manera:

```
fs = 5000;
t=0:1/fs:0.1-1/fs;
s_1 = 0.5*cos(2*pi*100*t)+1.5*cos(2*pi*500*t);
senal = s_1+(sqrt(2)*randn(1,500));
```

- El gráfico de la señal en el tiempo se muestra en la Figura 5, donde no se distingue s_1, ni sus frecuencias debido al ruido.

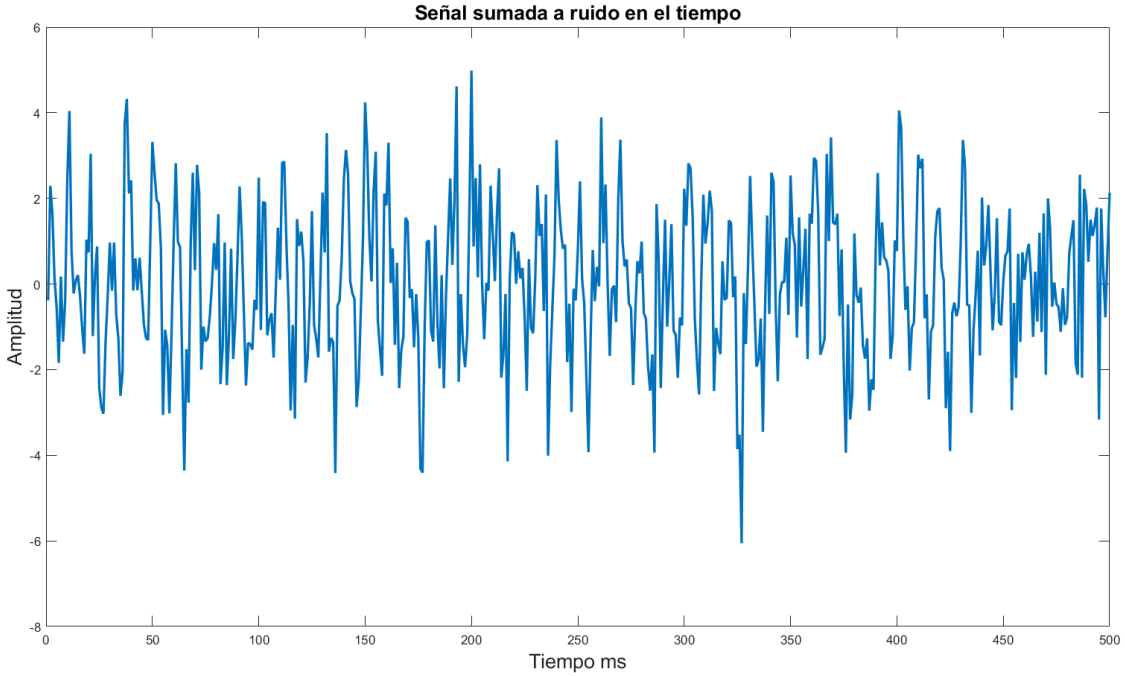

Figura 5: Señal s_1 más ruido comparación en magnitud.

- El gráfico de la magnitud del espectro de la señal con y sin ruido se muestra en la Figura 6, donde se distinguen claramente las frecuencias de interés a pesar del ruido de la señal. Se destaca que las amplitudes que acompañan a cada una de las frecuencias de interés, también corresponden a las esperadas, siendo estas la cantidad de muestras sobre dos, multiplicado por la amplitud de cada uno de los armónicos.

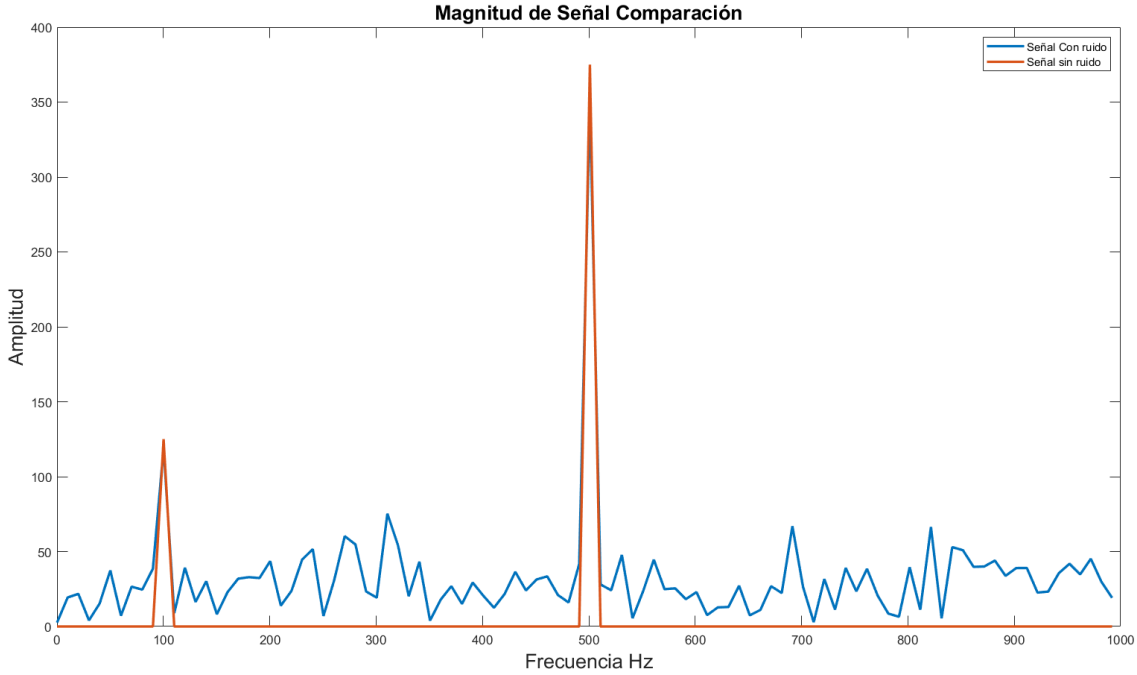

Figura 6: Señal s_1 más ruido en el tiempo.

- El gráfico de la magnitud del espectro en decibeles con y sin ruido se muestra en la Figura 7, donde vemos que aproximadamente hay 28 decibeles desde la señal con mayor amplitud hasta la capa de ruido.

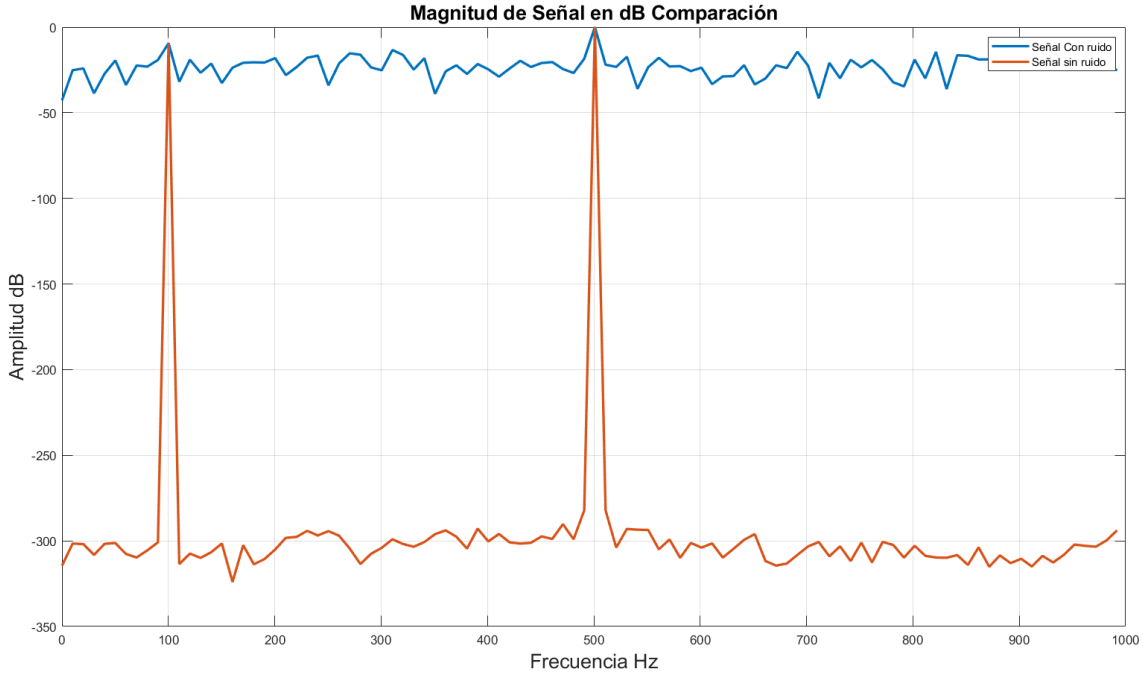

Figura 7: Señal s_1 más ruido, comparación en decibeles.

Al aumentar la duración de la señal a 1 segundo, se consigue el resultado en mostrado en la Figura 8, donde la señal de más duración tiene una resolución mucho mayor en su espectro en frecuencia, esto se debe a que, al contar con más muestras en el tiempo, se tienen del mismo modo más muestras en el espectro de frecuencia.
Se observa que la diferencia en amplitud entre la señal y el ruido aumenta al aumentar la duración de la señal. Esto se debe a que la amplitud de los peaks de la señal dependen directamente del número de muestras, sin embargo el ruido es uniforme en el espectro, por lo que su magnitud es invariante al aumentar el largo.

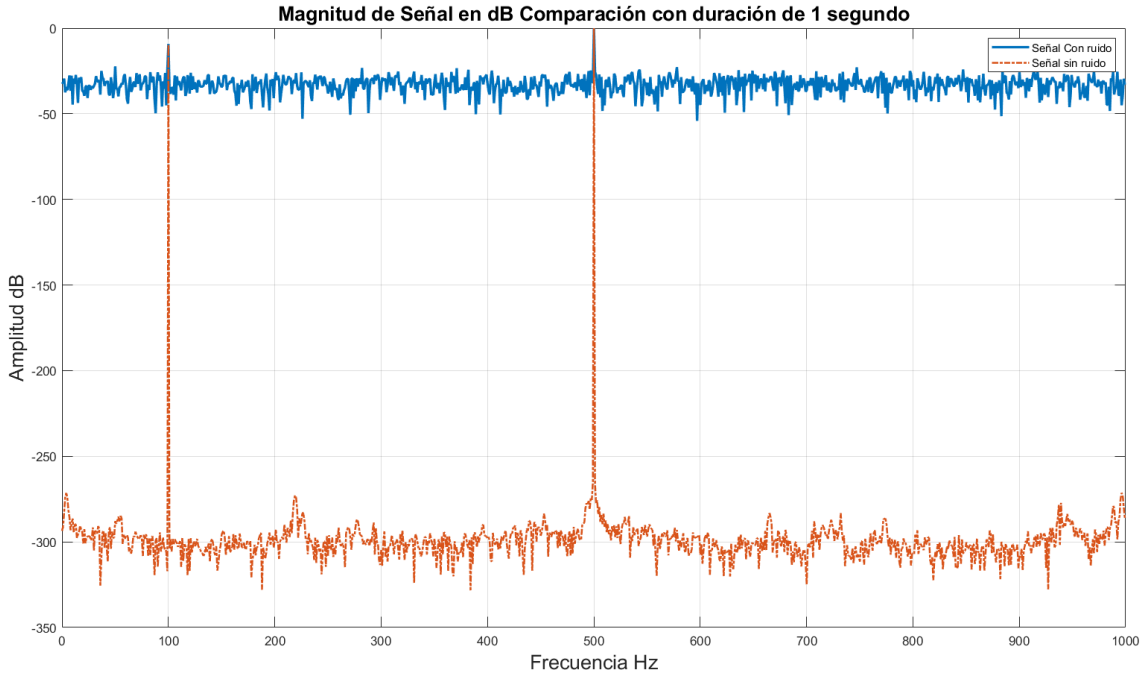

Figura 8: Señal s_1 más ruido, comparación en decibeles, duración 1 s.

III. Filtrado de señales en el dominio de la frecuencia

Se aplica fft a la señal $nspeech$ original, obteniendo su espectro en frecuencia, el cual se muestra con una amplitud en dB en la Figura 9, donde se extrae la frecuencia del tono no deseado, igual a $f = 1685{,}4\ Hz$.

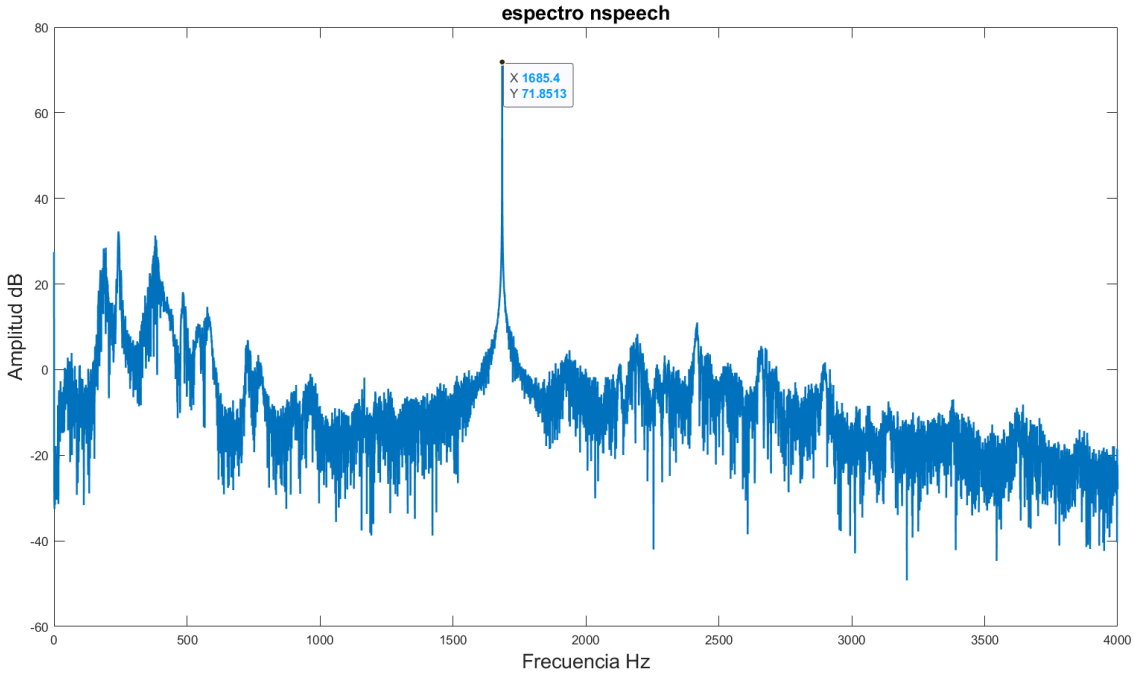

Figura 9: Magnitud del espectro de $nspeech$ en dB.

Se Obtiene el parámetro $\theta = 2\pi\frac{f}{f_s}$, por lo que $\theta = 1{,}3237\ Rad/Muestra$.

Se muestra la magnitud en dB del filtro en la Figura 10 donde se aplico el valor de θ. En la gráfica se observa una caída de cerca de 70 dB en la frecuencia de interés.

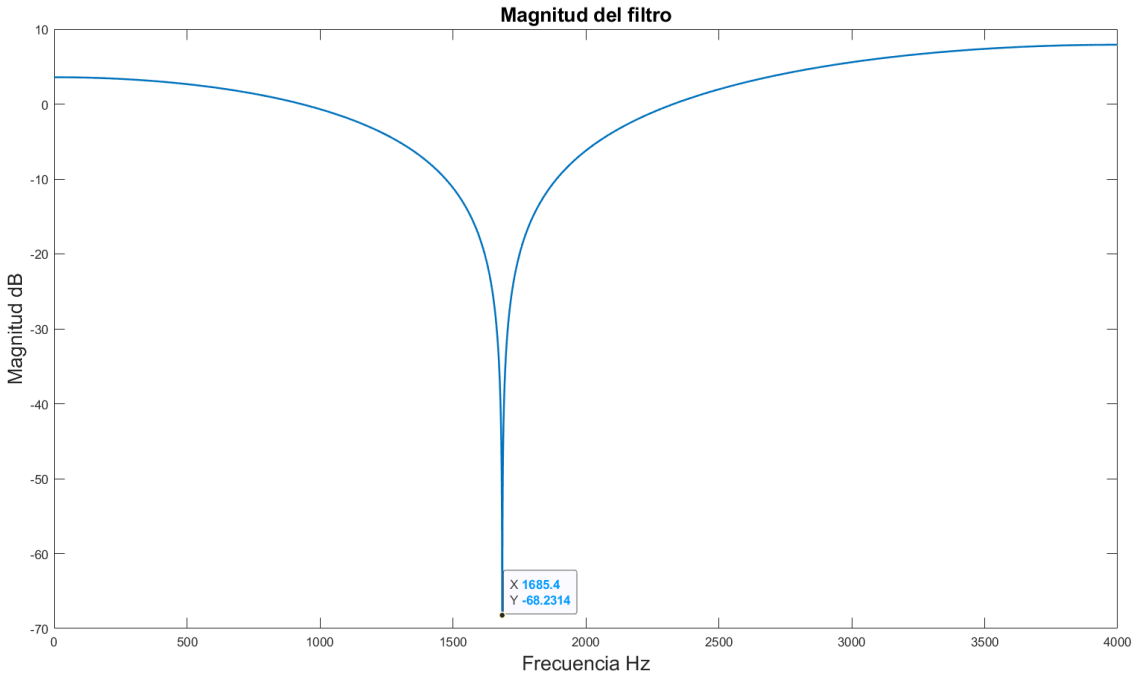

Figura 10: Magnitud del filtro en dB.

Se aplica el filtrado a la DFT de la señal $nspeech$ mediante una operación punto a punto segun el siguente codigo de $Matlab$:

```
f=1685.4;
w=2*pi*f/fs;  %-> omega
filtro=1-2*cos(w)*exp(-j*2*pi*(w_vector)/fs)+exp(-2*j*2*pi*(w_vector)/fs);
fft_filtrada = fft_senal.*filtro;
```

El resultado de esta operación, con magnitud en dB, se muestra en la Figura 11, donde se observa que el tono no deseado disminuye su amplitud cerca de 68 dB, que corresponde al rechazo del filtro. Si bien el tono no fue completamente eliminado, es prácticamente inaudible.

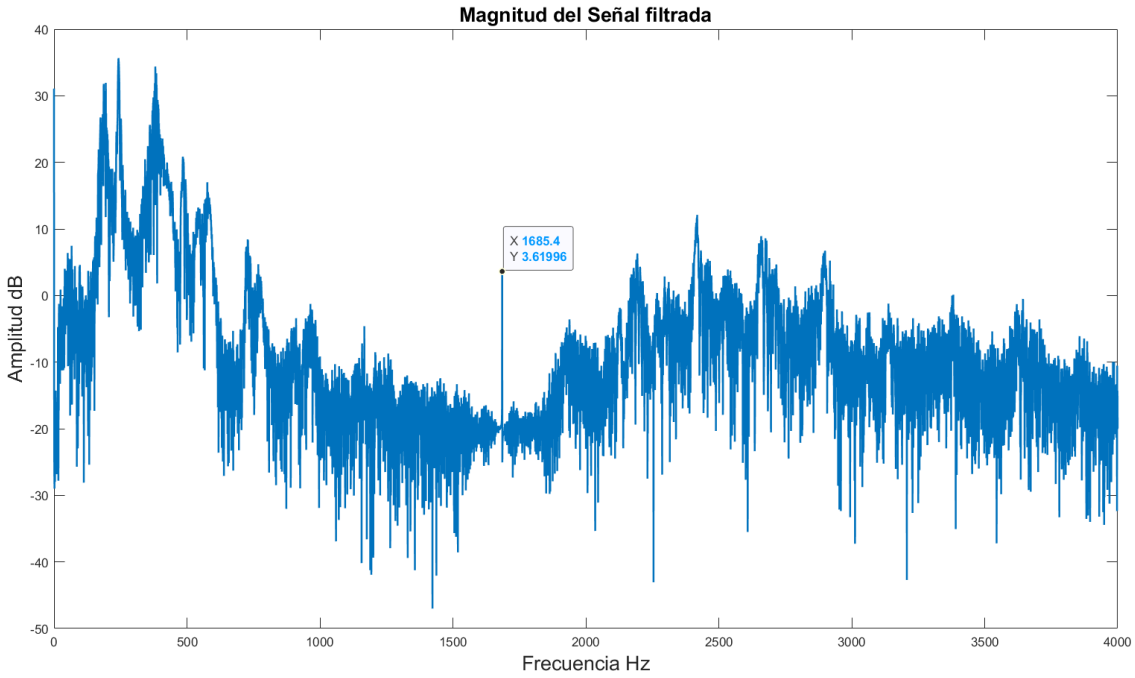

Figura 11: Magnitud de la señal filtrada en dB.

El resultado de aplicar la transformada inversa $ifft$ se muestra en la Figura 12, Donde se observa la estructura temporal de la señal filtrada casi por completo recuperada comparada con la señal original que es prácticamente inteligible. al escuchar ambas señales, podemos entender completamente el mensaje de la señal filtrada a diferencia de la original, donde predomina el sonido del tono no deseado.

Tanto la señal original como la señal filtrada tienen el mismo numero de muestras.

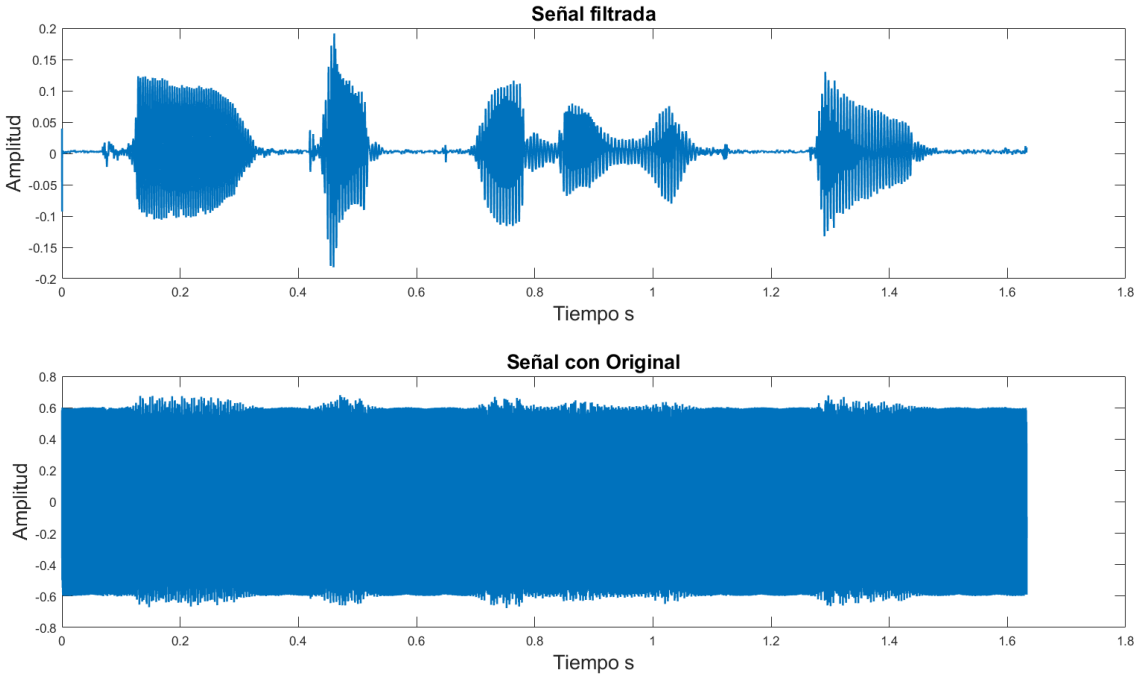

Figura 12: Señal filtrada vs Señal original en el tiempo.

IV. CÁLCULO DIRECTO DE LA DFT

La implementación en $MATLAB$ de la función $DFTsum$ se presenta a continuación:

```
function X=DFTsum(x)
N = length(x);
X = zeros([1,N]);
for k=1:length(x)
    for n=1:length(x)
        X(k)=X(k)+x(n)*exp(-j*2*pi*(k-1)*(n-1)/N);
    end
end
end
```

Donde se implementa la ecuación siguiente en forma directa utilizando dos ciclos for.

$$X^{(N)}[k] = \sum_{n=0}^{N-1} x[n] e^{\frac{-j2\pi kn}{N}}$$

1) Se prueba la función para $N = 8$ con las señal a continuación:
   a) $x_1[n] = \delta[n]$
   b) $x_2[n] = 1$
   c) $x_3[n] = e^{\frac{-j2\pi n}{N}}$
   d) $x_4[n] = cos(\frac{2\pi n}{N})$

   El resultado de probar estas señales con el filtro diseñado y la función fft de $MATLAB$ se muestra en la Figura 13, que son equivalentes en magnitud.

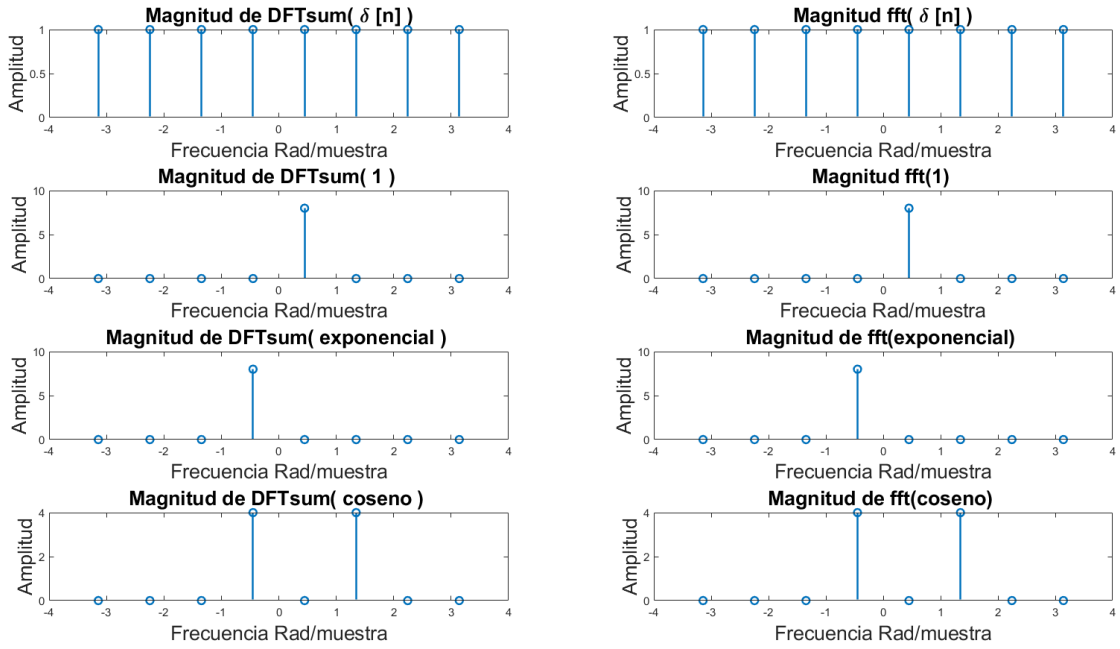

Figura 13: Comparación con las distintas funciones.

El error cuadrático medio el resultado proporcionado por las funciones $DFTsum$ y fft se muestra para cada señal en la tabla I, Donde el error mostrado es pequeño y se aprecia que los resultados obtenidos son aceptables.

Señal	MSE
$x_1[n]$	0
$x_2[n]$	3.8323e-30
$x_3[n]$	9.8353e-30
$x_4[n]$	3.1977e-30

Tabla I: MSE entre Función $DFTsum$ y fft de $MATLAB$

2) El calculo de la DTFT de las señales se muestra a continuación:

a)

$$x_1[n] \xleftrightarrow{\text{DTFT}} \sum_{n=0}^{7} \delta[n] e^{-j\omega n}$$
$$x_1[n] \xleftrightarrow{\text{DTFT}} 1$$

b)

$$x_2[n] \xleftrightarrow{\text{DTFT}} \sum_{n=0}^{7} 1 e^{-j\omega n}$$
$$x_2[n] \xleftrightarrow{\text{DTFT}} e^{-j\omega 7/2} \frac{sin(4\omega)}{sin(\omega/2)}$$

c) Sea $\omega_0 = \frac{2\pi}{N}$

$$x_3[n] \xleftrightarrow{\text{DTFT}} \sum_{n=-\infty}^{\infty} e^{-j\omega_0 n} e^{-j\omega n}$$
$$x_3[n] \xleftrightarrow{\text{DTFT}} 2\pi\delta(\omega - \omega_0) * e^{-j\omega 7/2} \frac{sin(4\omega)}{sin(\omega/2)}, \qquad \omega \in [0, 2\pi)$$
$$x_3[n] \xleftrightarrow{\text{DTFT}} 2\pi e^{-j(\omega-\omega_0)7/2} \frac{sin(4(\omega - \omega_0))}{sin((\omega - \omega_0)/2)}$$

d) Sea $\omega_0 = \frac{2\pi}{N}$

$$x_4[n] \xleftrightarrow{\text{DTFT}} \sum_{n=-\infty}^{\infty} cos(\omega_0 n) e^{-j\omega n}$$
$$x_4[n] \xleftrightarrow{\text{DTFT}} \pi(\delta(\omega + \omega_0) + \delta(\omega - \omega_0)) * e^{-j\omega 7/2} \frac{sin(4\omega)}{sin(\omega/2)}, \qquad \omega \in [0, 2\pi)$$
$$x_3[n] \xleftrightarrow{\text{DTFT}} \pi e^{-j(\omega-\omega_0)7/2} \left(\frac{sin(4(\omega - \omega_0))}{sin((\omega - \omega_0)/2)} + \frac{sin(4(\omega + \omega_0))}{sin((\omega + \omega_0)/2)} \right)$$

3) Se comparan las señales generadas por la función $DTFsum$ y fft con el siguiente script de $MATLAB$

```
fs=5000;
t=0:1/fs:1-1/fs;
x1=cos(2*pi*100*t);
dft_cos=DFTsum(x1(1:500));
fft_cos=fft(x1(1:500));
error_plot = (abs(dft_cos)-abs(fft_cos)).^2
MSE=immse(dft_cos,fft_cos)
```

Con lo que obtenemos un MSE de 1.9344e-24, y se obtiene el gráfico mostrado en la Figura 14, donde se aprecia como el error aumenta al aumentar la frecuencia, pero sigue siendo un error despreciable.

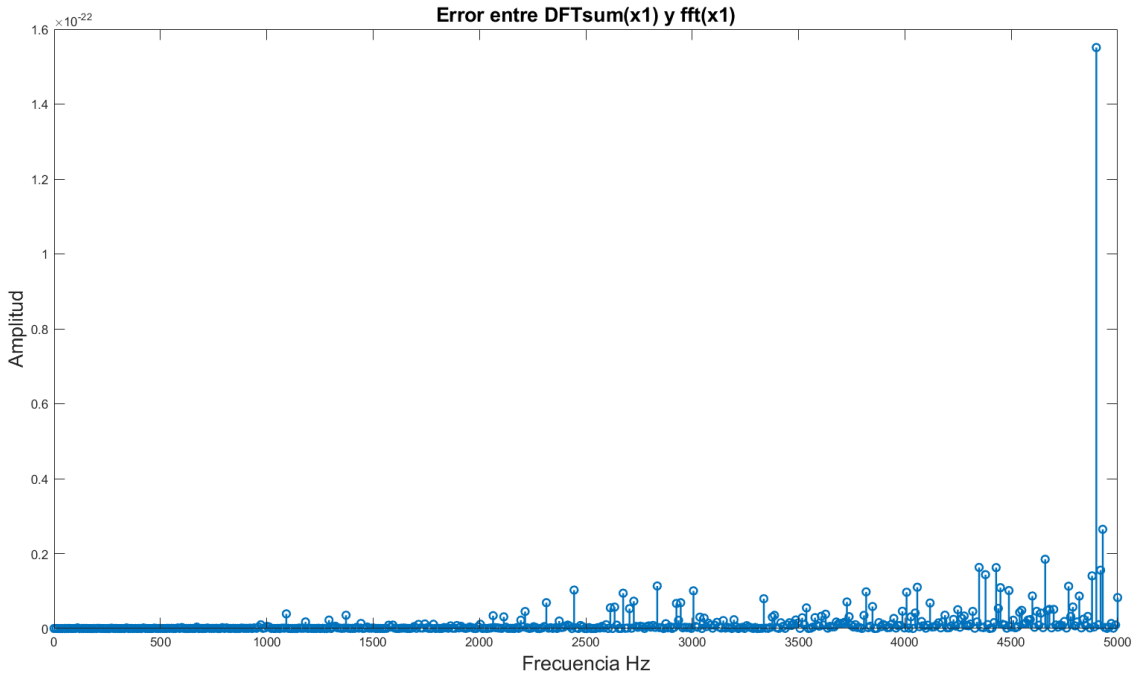

Figura 14: Magnitud del error entre usar DFTsum y fft.

V. Cálculo matricial de la DFT

1) La implementación en $MATLABde$ la función $genAmatrix$ se presenta a continuación:

```
function A=genAmatrix(N)
A = zeros([N,N]);
for k=1:N
    for n=1:N
    A(k,n)=exp(-2*j*pi*(k-1)*(n-1)/N);
    end
end
end
```

A continuación se muestra la implementación de $DFTmatrix$:

```
function X=DFTmatrix(x)
N = length(x);
X = zeros([1,N]);
A = genAmatrix(N);
X = x*A;
end
```

En la tabla II se muestra el error entre la función creada y la proporcionada por $MATLAB$, donde, al duplicar N el error aumenta en un orden de magnitud. Estos errores siguen siendo insignificantes.

N	MSE
2	3.7494e-33
4	4.5930e-32
8	7.9370e-31

Tabla II: MSE entre Función $genAmatrix$ y $dftmtx$ de $MATLAB$

2) En la Figura 15 se muestran los gráficos de la parte real e imaginaria para distintos N en escala de colores, donde se comprueba la simetría de las matrices. En cada caso, el eje x corresponde al numero de columna y el eje y el número de fila.

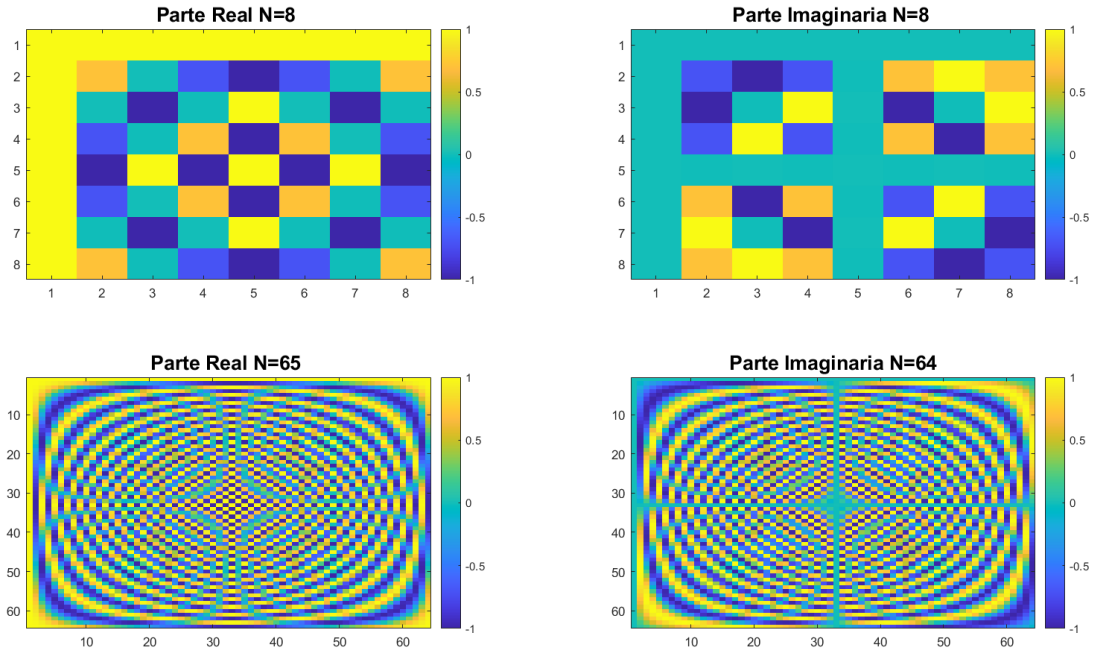

Figura 15: Escala de colores para $genAmatrix$ de distintos tamaños.

3) El resultado de probar estas señales con la función diseñado $DFTmatrix$ y la función fft de $MATLAB$ se muestra en la Figura 16, que son equivalentes en magnitud. En la Figura 17 se muestra el resultado de aplicarle a las funciones anterioremente descritas la señal $x_5 = cos(\omega t)$ con 5000 muestras.

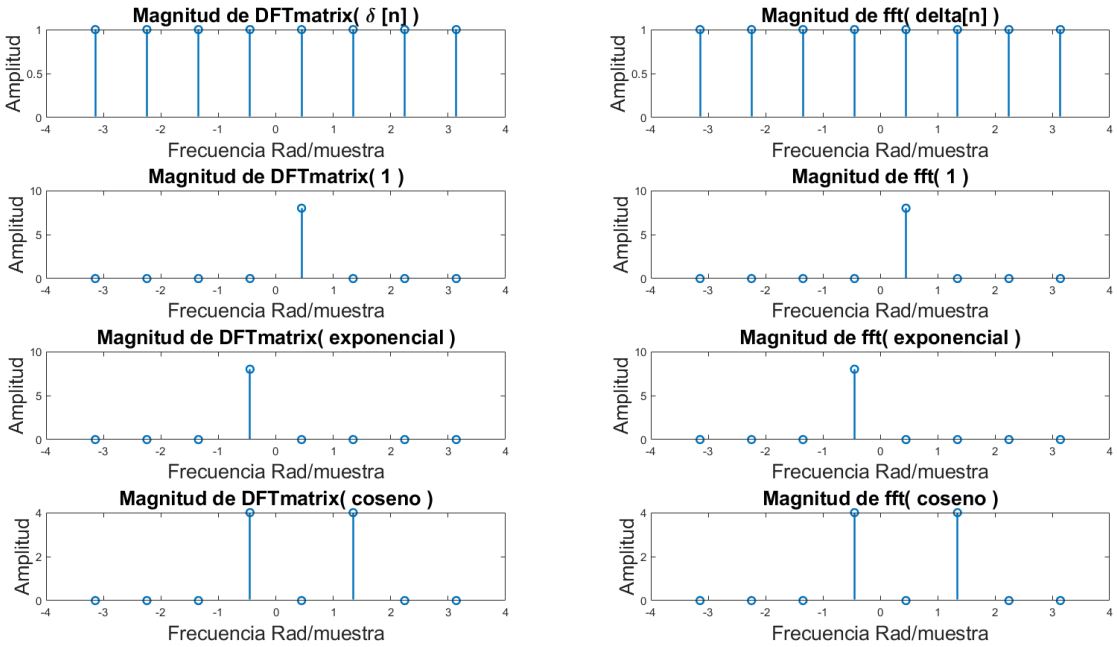

Figura 16: Comparación con las distintas funciones.

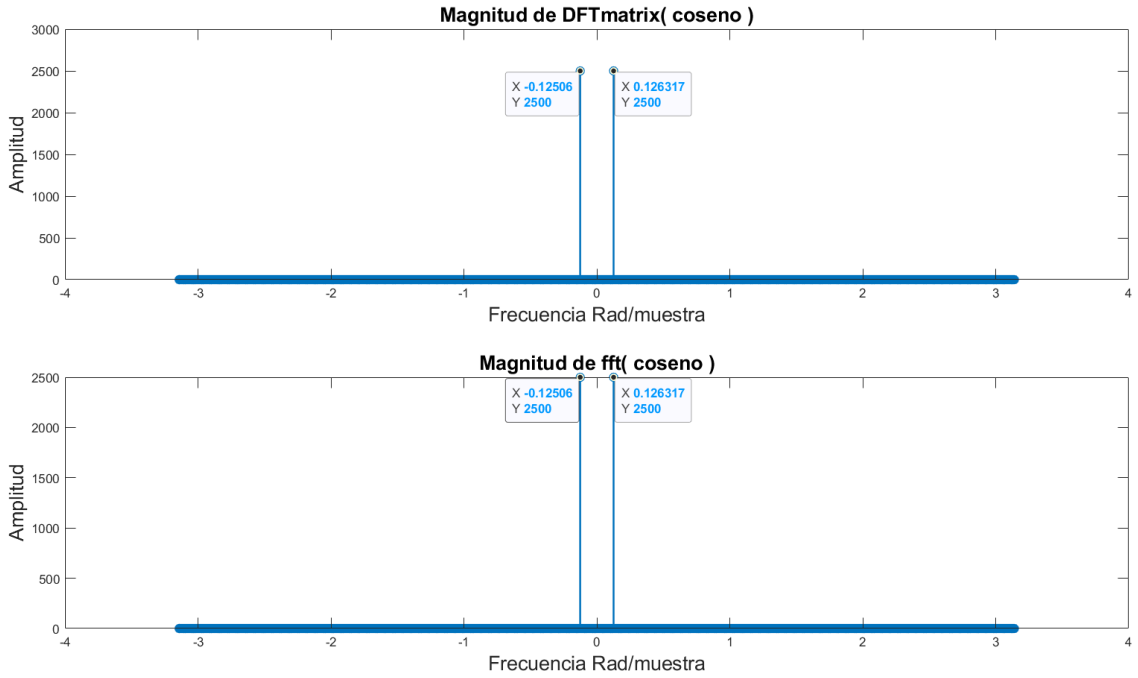

Figura 17: Comparación con las distintas funciones.

En la tabla III se muestra el error entre las función creadas y entre la función $DFTmatrix$ y fft, donde se mantiene el orden de magnitud para los errores al cambiar N, excepto para la señal x_5, debido al alto numero de muestras.

Señal	MSE(DFTmatrix,fft)	MSE(DFTmatrix,DFTsum)
$x_1[n]$	0	0
$x_2[n]$	3.6901e-30	3.3671e-32
$x_3[n]$	9.8474e-30	2.9198e-32
$x_4[n]$	3.0622e-30	1.2749e-32
x_5	2.2216e-21	2.1528e-21

Tabla III: MSE entre Función $DFTsum$ y fft de $MATLAB$

4) Se miden los tiempos de comparación para la señal $x_5 = cos(\omega t)$ con 5000 muestras utilizando el siguiente script:

```
fs=5000;
t=0:1/fs:1-1/fs;
x1=cos(2*pi*100*t);
f1 = @()DFTsum(x1);
f2 = @()DFTmatrix(x1);
t1_0 = timeit(f1);
t2_0 = timeit(f2);
```

Con lo que se obtienen los tiempos mostrados en la tabla IV, donde se persive una mejora usando matrices en el calculo.

Señal	Tiempo DFTsum s	Tiempo DFTmatrix s
x_5	5.67202599995000	5.28868089995000

Tabla IV: Tiempo de ejecución para Función $DFTsum$ y $DFTmatrix$

Se comparan los tiempos usando diferentes números de muestras N para la funcion $x_5 = cos(\omega t)$ utilizando el siguiente script de $MATLAB$:

```
fs=5000;
t=0:1/fs:1-1/fs;
t1=zeros([1,50]);
t2=zeros([1,50]);
t3=zeros([1,50]);
x5=cos(2*pi*100*t);
```

```
i=1;
for N=100:100:5000
    f1 = @()DFTsum(x5(1:N));
    f2 = @()DFTmatrix(x5(1:N));
    f3 = @()fft(x5(1:N));
    f4 = @()dftmtx(N);
    t1(i) = timeit(f1);
    t2(i) = timeit(f2);
    t3(i) = timeit(f3);
    t4(i) = timeit(f4);
    i=i+1;
end
```

El grafico de los tiempos de ejecución para las diferentes funciones se muestra en la Figura 18, donde se aprecia como el tiempo de ejecución tiene una curva exponencial para los métodos implementados DFTsum y DFTmatrix. Se aprecia como el tiempo de ejecución de la función fft tiene tiempos más cortos, y con una curva logaritmica. Al apreciar los tiempo de la función $dftmtx$ de matlab se concluye que la creación de la matriz no influye de sobre manera en los tiempos de ejecución del algoritmo $DFTmatrix$.
Finalmente la función fft es la más eficiente, se destaca que, en $MATLAB$ la función $DFTmatrix$ es más eficiente que la función $DFTsum$ por el uso de matrices, ya que el programa esta optimizado para las operaciones con estas. Por la forma de trabajar porciones de datos, la función fft ocupa más recuersos de memoria.

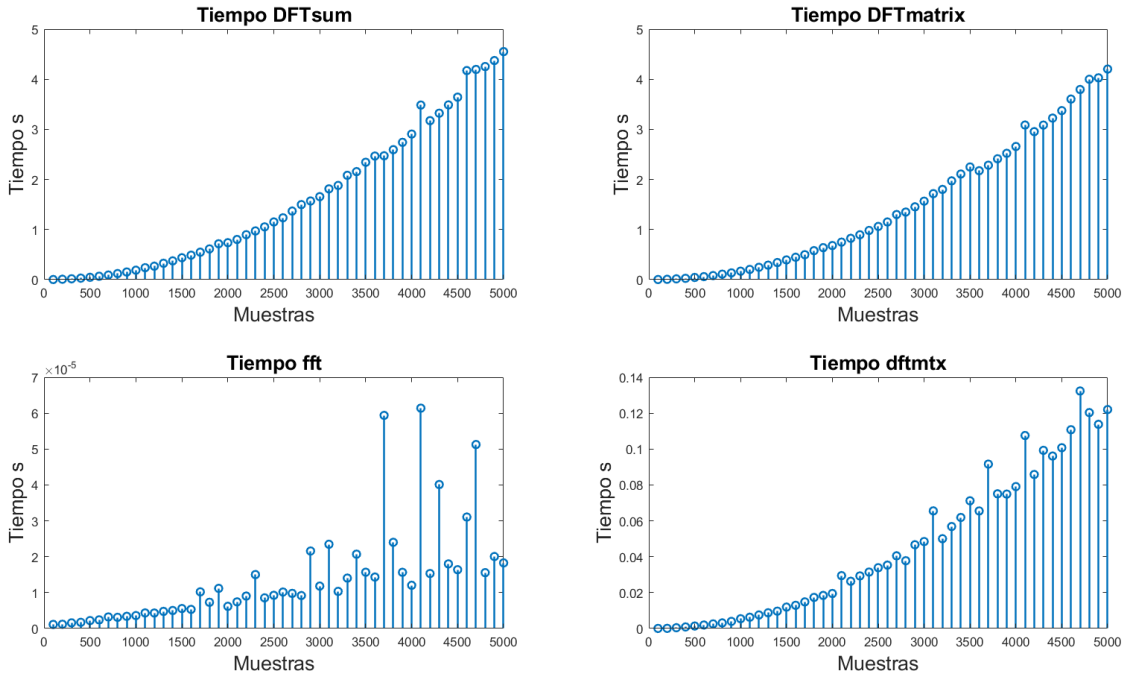

Figura 18: Comparación con las distintas funciones.

VI. Algoritmo Radix-2 de la FFT

1) Se implementó la siguiente función para implementar la DFT con reducción de orden.

```
    function X=dftdc(x)
    x_par=zeros([1,ceil(length(x))*0.5]);
        for n=1:length(x)
            if mod(n,2)~=0
                x_par((n+1)/2)=x(n);
            end
        end
    x_impar=zeros([1,ceil(length(x))*0.5]);
        for n=1:length(x)
            if mod(n,2)==0
                x_impar(n/2)=x(n);
            end
```

```
        end
    X_par=DFTsum(x_par);
    X_impar=DFTsum(x_impar);
    k=0:length(X_impar)-1;
    Wnk=exp(-j*2*pi*k/length(x));
    X=[X_par+Wnk.*X_impar, X_par-Wnk.*X_impar];
    end
```

Se aplicó el algoritmo para las mismas señales que en VI-1, demostrando, en los gráficos de la figura 19 que se obtiene adecuadamente la dft de la señal.

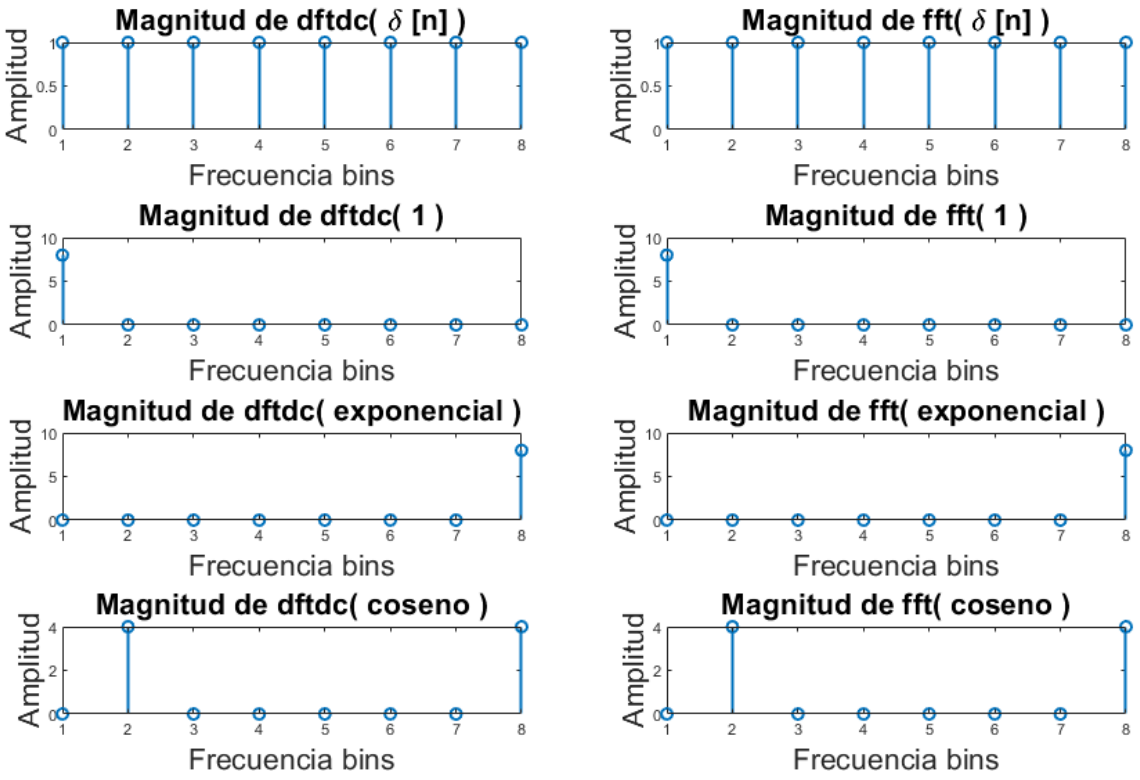

Figura 19: Comparación entre fft y dftdc

Los tiempos de ejecución para ambas funciones con la señal $\delta(n)$ de ocho muestras se observan en la tabla V. Donde se observa que la fft de MATLAB sigue siendo más rápida.

Señal	Tiempo DFTdc s	Tiempo fft s
x_5	2.368314411764706e-05	7.550727772227772e-07

Tabla V: Tiempo de ejecución para Función $DFTdc$ y fft de MATLAB

2) Se implementan las siguientes funciones, las cuales calculan la FFT de una señal para 8, 4 y 2 puntos respectivamente, en este caso, la función FFT8 llama a FFT4, y la función FFT4 llama a FFT2:

```
function X = FFT8(x)
Wn = exp(-1j*2*pi/8);
X = zeros([1,8]);
x0 = [x(1) x(3) x(5) x(7)];
X0 = FFT4(x0);
x1 = [x(2) x(4) x(6) x(8)];
X1 = FFT4(x1);
    for k=1:4
        X(k) = X0(k)+Wn^(k-1)*X1(k);
        X(k+4) = X0(k)-Wn^(k-1)*X1(k);
    end
```

```
12          end
13
14          function X = FFT4(x)
15          Wn = exp(-1j*2*pi/4);
16          X = zeros([1,4]);
17          x0 = [x(1) x(3)];
18          X0 = FFT2(x0);
19          x1 = [x(2) x(4)];
20          X1 = FFT2(x1);
21              for k=1:2
22                  X(k) = X0(k)+Wn^(k-1)*X1(k);
23                  X(k+2) = X0(k)-Wn^(k-1)*X1(k);
24              end
25          end
26
27          function X = FFT2(x)
28          X(1) = x(1)+x(2);
29          X(2) = x(1)-x(2);
30          end
```

Se graficaron las señales anteriores con FFT8, corroborando que las funciones implementadas calculan correctamente la dft. Los gráficos se muestran en la figura 20

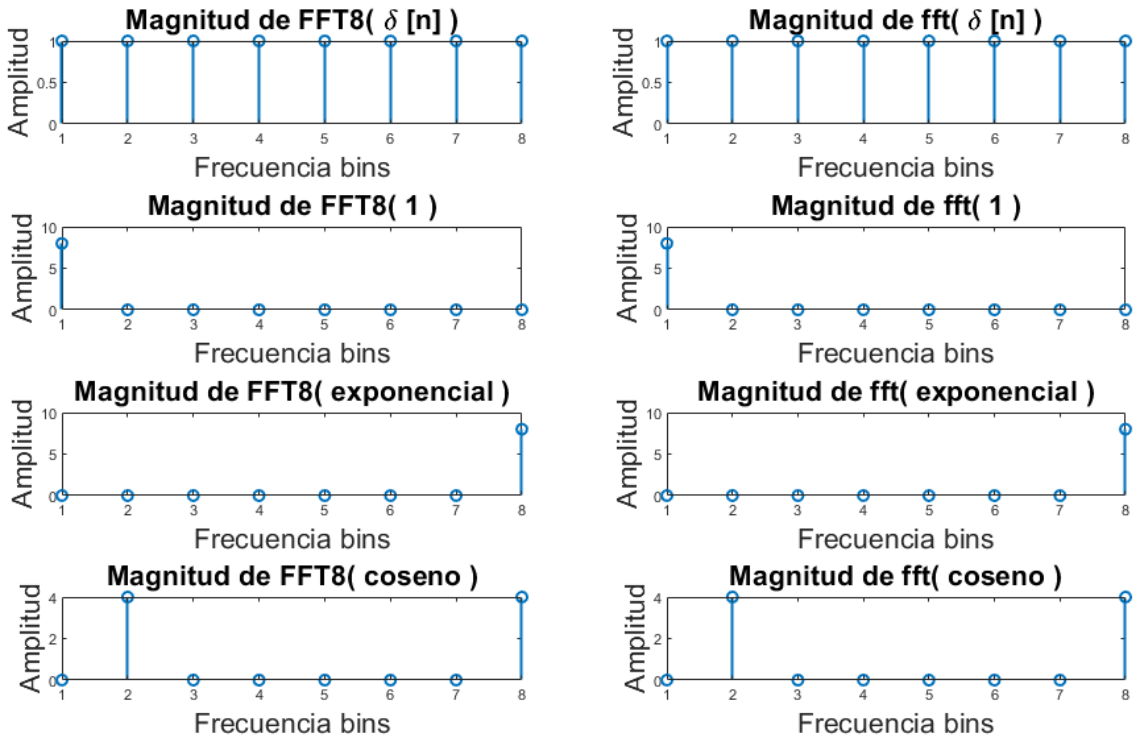

Figura 20: Comparación entre fft y FFT8

El tiempo de procesamiento obtenido para el delta $\delta(n)$ de 8 muestras en este caso es de $1{,}303173888888889e-05$ s, siendo un poco mas leve que para la función del punto anterior, pero aún superior a la $fft()$ de MATLAB.

3) A partir de la lógica de las funciones anteriores, se implementa finalmente la función recursiva $fft\_stage(x)$, que obtiene la fft de la señal x.

```
function X = fft_stage(x)
if length(x) == 2
    X =FFT2(x);
else
    x_par=zeros([1,ceil(length(x))*0.5]);
    for n=1:length(x)
        if mod(n,2)~=0
            x_par((n+1)/2)=x(n);
        end
    end
    x_impar=zeros([1,ceil(length(x))*0.5]);
    for n=1:length(x)
        if mod(n,2)==0
            x_impar(n/2)=x(n);
        end
    end
    X_par = fft_stage(x_par);
    X_impar = fft_stage(x_impar);

    k=0:length(X_impar)-1;
    Wnk=exp(-j*2*pi*k/length(x));
    X=[X_par+Wnk.*X_impar, X_par-Wnk.*X_impar];
end
end
```

Se comprobó tanto la dft calculada por la función ff_stage como la fft de MATLAB para 256 muestras de un coseno con frecuencia 100. Se observa que la función fft_stage calcula la fft correctamente

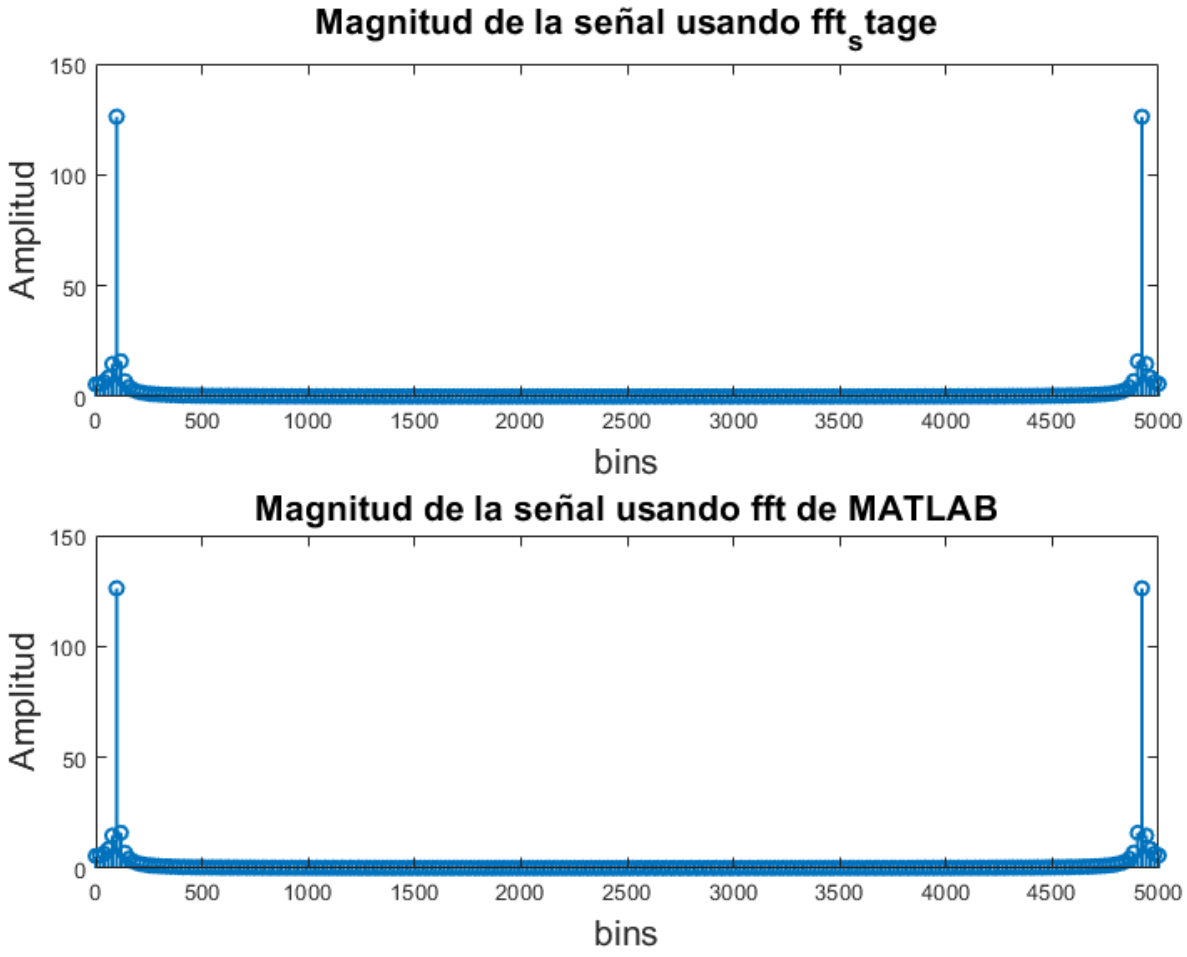

Figura 21: Comparación entre algoritmos resultados de funciones fft y fft_stage

Comparamos los tiempos de la función fft_stage generada y la función fft de $MATLAB$ con el siguiente script:

```
fs=5000;
t=linspace(0,3-1/fs,2^19);
x1=cos(2*pi*100*t);
i=1;
for k=10:19
    N = 2^k;
    f1 = @()fft_stage(x1(1:N));
    f2 = @()fft(x1(1:N));
    t1(i) = timeit(f1);
    t2(i) = timeit(f2);
    i=i+1;
end
```

Los resultados de esto se muestran en la Figura 22, donde las muestras N = [0, 1, ..., 10] corresponden a n = [2^{10}, 2^{11}, ... 2^{19}]. Se aprecia que los tiempos de ejecución de ambas funciones escalan de igual forma. Los tiempos la función fft_stage son buenos respecto a las otras funciones que se implementaron, pero la función fft sigue siendo más eficiente.

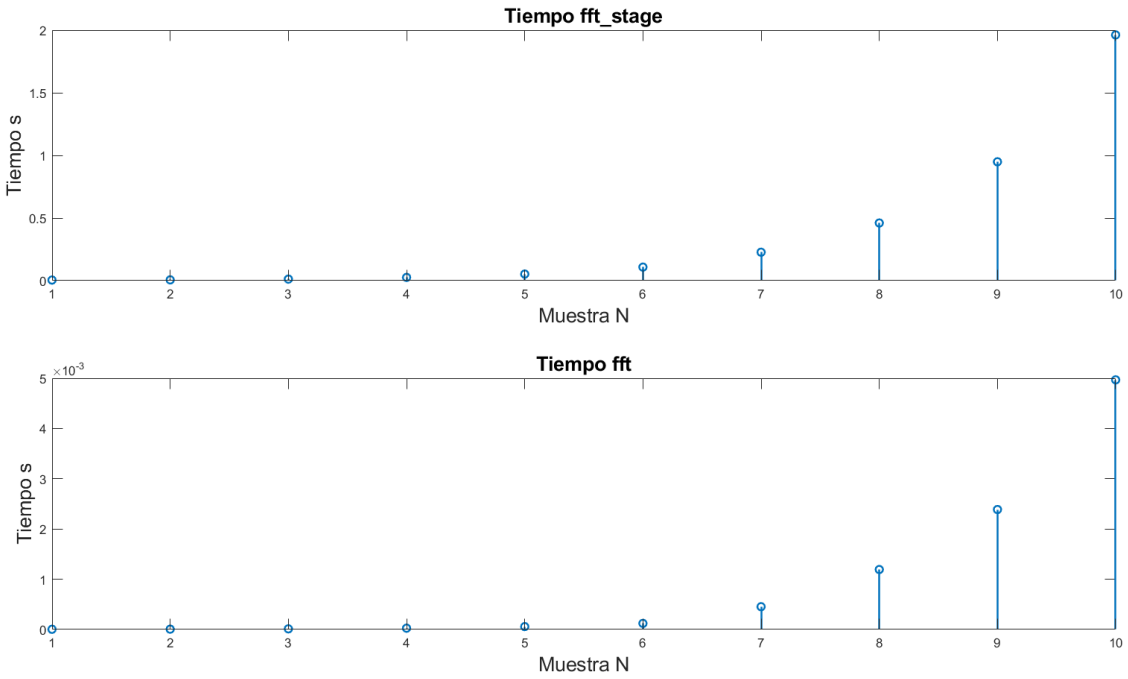

Figura 22: Comparación entre tiempos resultados de funciones fft y fft_stage

VII. Implementación en S-Functions: Algoritmo de Goertzel

1) Se creó la siguiente función (ejecutada una sola vez) para inicializar los parámetros del filtro Goertzel:

```
static void initGoertzel(goertzelState_t *state, uint64_t kFrequency)
{
  double omega = (2*M_PI*kFrequency/GOERTZEL_N);
  state->samplesCounter = 0;
  state->cosW = cos(omega);
  state->sinW = sin(omega);
  state->A1= -2*cos(omega);
  state->outputs[0] = 0.0;
  state->outputs[1] = 0.0;
  state->outputs[2] = 0.0;
  state->binReal=0;
  state->binImag=0;
  state->binMag=0;
    return;
}
```

Luego, la siguiente función permite filtrar la señal según la formula de Goertzel, y devuelve el valor del BIN requerido luego de N muestras:

```
static double computeGoertzel(goertzelState_t *state, double filterInput)
{
state->outputs[2]=state->outputs[1];
state->outputs[1]=state->outputs[0];
state->outputs[0]=filterInput+(2*state->cosW*state->outputs[1])-(state->outputs[2]);
(state->samplesCounter)+=1;

if(state->samplesCounter==GOERTZEL_N)
{
  state->binReal=state->cosW*state->outputs[1]-state->outputs[2];
  state->binImag=state->sinW*state->outputs[1];
  state->binMag=sqrt(state->binReal*state->binReal+state->binImag*state->binImag);
  resetGoertzel(state);
}
return state->binMag;
}
```

Para comprobar el funcionamiento del programa, se miden los bins $8, 9, 10$ de la transformada de Fourier de las primeras 256 muestras, lo que se observa en la figura

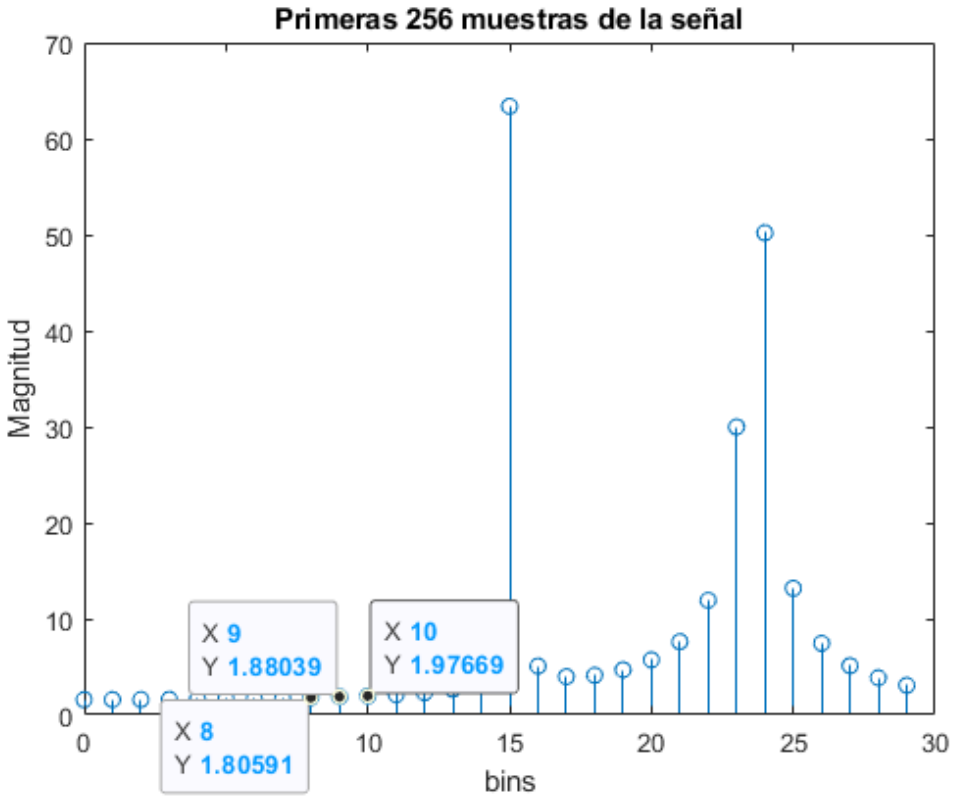

Figura 23: Bins del primer frame de la señal

Luego, se comparan con la salida luego de N muestras de la S-Function, seleccionando los bins $8, 9, 10$. Donde se observa que los valores obtenidos corresponden efectivamente a las magnitudes del bin.

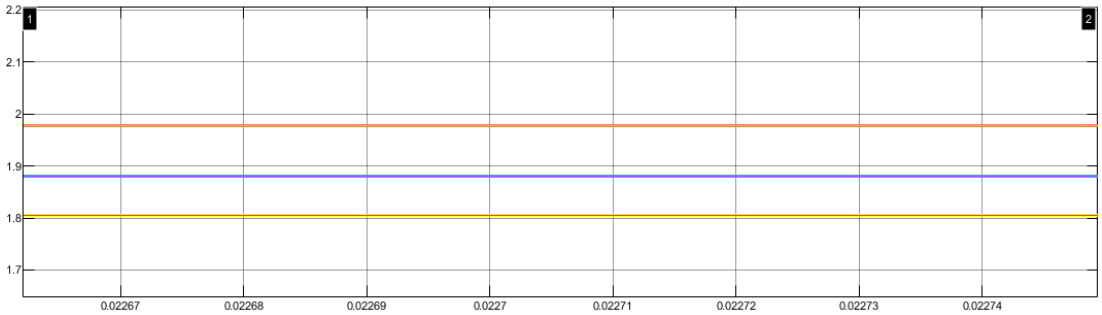

Figura 24: Bins obtenidos en la salida, 8 en amarillo, 9 en azul y 10 en marrón

2) Para elegir los bins, se calculan transformadas de Fourier de 256 puntos en distintos momentos de la señal, correspondiente a distintos tonos. Luego se eligen los dos bins con mayor magnitud (Para identificar cada tono más fácilmente, se utilizó la señal *dtmfSequenceSpaced_16_16.wav*). Obteniéndose la siguiente tabla:

Primer sample	bin 1	bin 2
0	15	24
4000	15	21
8000	11	24
11000	11	19
13000	12	19
17000	14	24

Tabla VI: Bins obtenidos

De esta forma, los 7 bins obtenidos son $11, 12, 14, 15, 19, 21, 24$
Luego, se ejecuta el código a cada uno de estos bins, creando 7 estructuras de tipo *goertzelState_t*. Y aplicando las funciones a cada una de ellas.

```
        /* Inicializacion */
          initGoertzel( &gGoertzelState1 , GOERTZEL1_K_BIN );
          initGoertzel( &gGoertzelState2 , GOERTZEL2_K_BIN );
          initGoertzel( &gGoertzelState3 , GOERTZEL3_K_BIN );
          initGoertzel( &gGoertzelState4 , GOERTZEL4_K_BIN );
          initGoertzel( &gGoertzelState5 , GOERTZEL5_K_BIN );
          initGoertzel( &gGoertzelState6 , GOERTZEL6_K_BIN );
          initGoertzel( &gGoertzelState7 , GOERTZEL7_K_BIN );
```

```
    /* Funcion principal */
        void goertzelFunction(double input1,
                double *output1,
                double *output2,
                double *output3,
                double *output4,
                double *output5,
                double *output6,
                double *output7
                )
        {
          *output1 = computeGoertzel(&gGoertzelState1 , input1);
          *output2 = computeGoertzel(&gGoertzelState2 , input1);
          *output3 = computeGoertzel(&gGoertzelState3 , input1);
          *output4 = computeGoertzel(&gGoertzelState4 , input1);
          *output5 = computeGoertzel(&gGoertzelState5 , input1);
          *output6 = computeGoertzel(&gGoertzelState6 , input1);
          *output7 = computeGoertzel(&gGoertzelState7 , input1);
        }
```

El resultado en la salida de la s-Function es mandado a MATLAB mediante la función *toWorkspace*.
Luego, cada para cada bin i de $i = 1 : 7$, se gráfica según el comando:

```
        bar(out.tout,out.bins(:,i))
```

Los gráficos obtenidos se muestran en la figura 25, donde se comprueba que en cada momento solo hay dos bins en alto, (correspondientes a los valores de DTMF del tono pulsado).

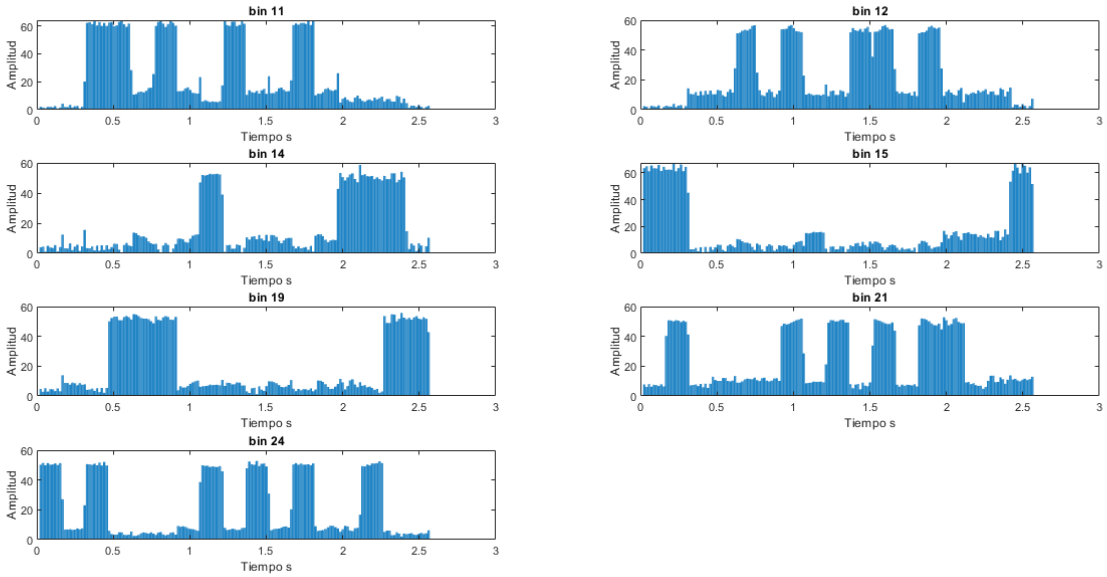

Figura 25: coeficientes de salida para los 7 bins elegidos

De la secuencia decodificada del informe anterior se puede extraer que los bins corresponden a lo siguiente:

Bin	Coordenada
11	Primera fila
12	Segunda fila
14	Tercera fila
15	Cuarta fila
19	Primera columna
21	Segunda columna
24	Tercera columna

Tabla VII: Coordenadas del bin

Luego, de DTMF, a partir de la tabla VIII y los graficos de la figura 25, se corrobora que la secuencia es efectivamente $\#031415926535897*$.

1	2	3
4	5	6
7	8	9
*	0	#

Tabla VIII: Coordenadas numéricas