Universidad Técnica Federico Santa María

DEPARTAMENTO DE INGENIERÍA ELECTRONICA

ELO 314 - Laboratorio de procesamiento Digital de Señales

Transformada Discreta de Fourier en MatLab

Estudiante	ROL
Rodrigo Graves	201621009-1
Ricardo Mardones	201621036-9

Paralelo: 1

Profesor

Gonzalo Carrasco

Ayudante

Jaime Guzmán

Fecha : 27 de junio de 2021

Índice

Índice de figuras

Índice de cuadros

1. Cálculo de la DFT para señales simples

Para esta sección se consideran las señales $x_1(t) = \sin(\omega t)$, $x_2(t) = \cos(\omega t)$, ambas con frecuencia 100 Hz y muestreadas a $f_s = 5\ kHz$ durante 100 ms.

1. Se pide obtener expresiones analíticas para las DTFT de las señales muestreadas $x_1[n],\ x_2[n]$, considerando duración infinita y los 100 ms.

   DTFT de $x_1(n)$ de duración infinita:

   En primer lugar se obtiene la versión muestreada de la señal en el dominio temporal:

   $$x_1[n] = \sin(2\pi f/f_s\ n) = \sin(0{,}04\pi n) = \sin(\omega_0 n)$$

   Luego la DTFT está dada por:

   $$\begin{aligned} X_1(\omega) &= \sum_{n=-\infty}^{\infty} x_1[n]e^{-j\omega n} \\ &= \sum_{n=-\infty}^{\infty} \sin(\omega_0 n)e^{-j\omega n} \\ &= \sum_{n=-\infty}^{\infty} \left(\frac{e^{j\omega_0 n} - e^{-j\omega_0 n}}{2j}\right) e^{-j\omega n} \\ &= \frac{1}{2j}\left(\sum_{n=-\infty}^{\infty} e^{j\omega_0 n}e^{-j\omega n} - \sum_{n=-\infty}^{\infty} e^{-j\omega_0 n}e^{-j\omega n}\right) \\ &= \frac{1}{2j}\left(2\pi\delta(\omega-\omega_0) - 2\pi\delta(\omega+\omega_0)\right) \\ &= \frac{\pi\delta(\omega-\omega_0) - \pi\delta(\omega+\omega_0)}{j} \end{aligned}$$

   Cabe destacar que la expresión obtenida considera $\omega \in [-\pi, \pi]$, si se desea para todo ω debe considerarse periodicidad de periodo 2π del espectro en frecuencia.

   DTFT de $x_2(n)$ de duración infinita:

   Se obtiene la versión muestreada de la señal en el dominio temporal:

   $$x_2[n] = \cos(2\pi f/f_s\ n) = \cos(0{,}04\pi n) = \cos(\omega_0 n)$$

Luego la DTFT está dada por:

$$\begin{aligned}
X_2(\omega) &= \sum_{n=-\infty}^{\infty} x_2[n]e^{-j\omega n} \\
&= \sum_{n=-\infty}^{\infty} \cos(\omega_0 n)e^{-j\omega n} \\
&= \sum_{n=-\infty}^{\infty} \left(\frac{e^{j\omega_0 n} + e^{-j\omega_0 n}}{2}\right) e^{-j\omega n} \\
&= \frac{1}{2}\left(\sum_{n=-\infty}^{\infty} e^{j\omega_0 n}e^{-j\omega n} + \sum_{n=-\infty}^{\infty} e^{-j\omega_0 n}e^{-j\omega n}\right) \\
&= \frac{1}{2}\left(2\pi\delta(\omega-\omega_0) + 2\pi\delta(\omega+\omega_0)\right) \\
&= \pi\delta(\omega-\omega_0) + \pi\delta(\omega+\omega_0)
\end{aligned}$$

Cabe destacar que la expresión obtenida considera $\omega \in [-\pi, \pi]$, si se desea para todo ω debe considerarse periodicidad de periodo 2π del espectro en frecuencia.

DTFT de $x_1[n]$:

Debido a que $x_1[n]$ corresponde a la señal de duración infinita multiplicada por una ventana cuadrada, se obtiene su DTFT como la convolución de el espectro de la señal de duración infinita y el espectro de la ventana cuadrada $W(e^{j\omega})$

El espectro de la ventana está dado por:

$$W(e^{j\omega}) = e^{-j\omega(N-1)/2}\frac{\sin(\omega N/2)}{sin(\omega/2)}$$

donde $N = 500$ ya que la ventana de 100 ms es de 500 muestras.

Luego el espectro de la señal enventanada está dado por:

$$\begin{aligned}
X_1^{(w)}(\omega) &= X_1(\omega) * W(e^{j\omega}) \\
&= \left(\frac{\pi\delta(\omega-\omega_0) - \pi\delta(\omega+\omega_0)}{j}\right) * W(e^{j\omega}) \\
&= \frac{\pi W(e^{\omega-\omega_0}) - \pi W(e^{\omega-\omega_0})}{j}
\end{aligned}$$

Lo cual permite bosquejar el espectro, pues aparecen sincs periódicas en las frecuencias $\pm\omega_0$

DTFT de $x_2[n]$:

Similar al caso anterior, se obtiene su DTFT como la convolución de el espectro de la señal de duración infinita y el espectro de la ventana cuadrada $W(e^{j\omega})$.

Luego el espectro de la señal enventanada está dado por:

$$\begin{aligned}
X_2^{(w)}(\omega) &= X_2(\omega) * W(e^{j\omega}) \\
&= (\pi\delta(\omega-\omega_0) + \pi\delta(\omega+\omega_0)) * W(e^{j\omega}) \\
&= \pi W(e^{\omega-\omega_0}) + \pi W(e^{\omega-\omega_0})
\end{aligned}$$

Lo cual claramente permite bosquejar el espectro, pues aparecen sincs periódicas en las frecuencias $\pm\omega_0$

2. Se obtiene mediante el comando `fft` de MATLAB, con un N = 4096 la magnitud de la DFT para las dos señales analizadas en el inciso anterior, $x_1 = sin(\omega t)$ y $x_2 = cos(\omega t)$, los resultados se graficaron en los rangos $[-\frac{fs}{2}, \frac{fs}{2}]$ y $[0, \frac{fs}{2}]$. Estas gráficas se muestran en las figuras 1 y 2

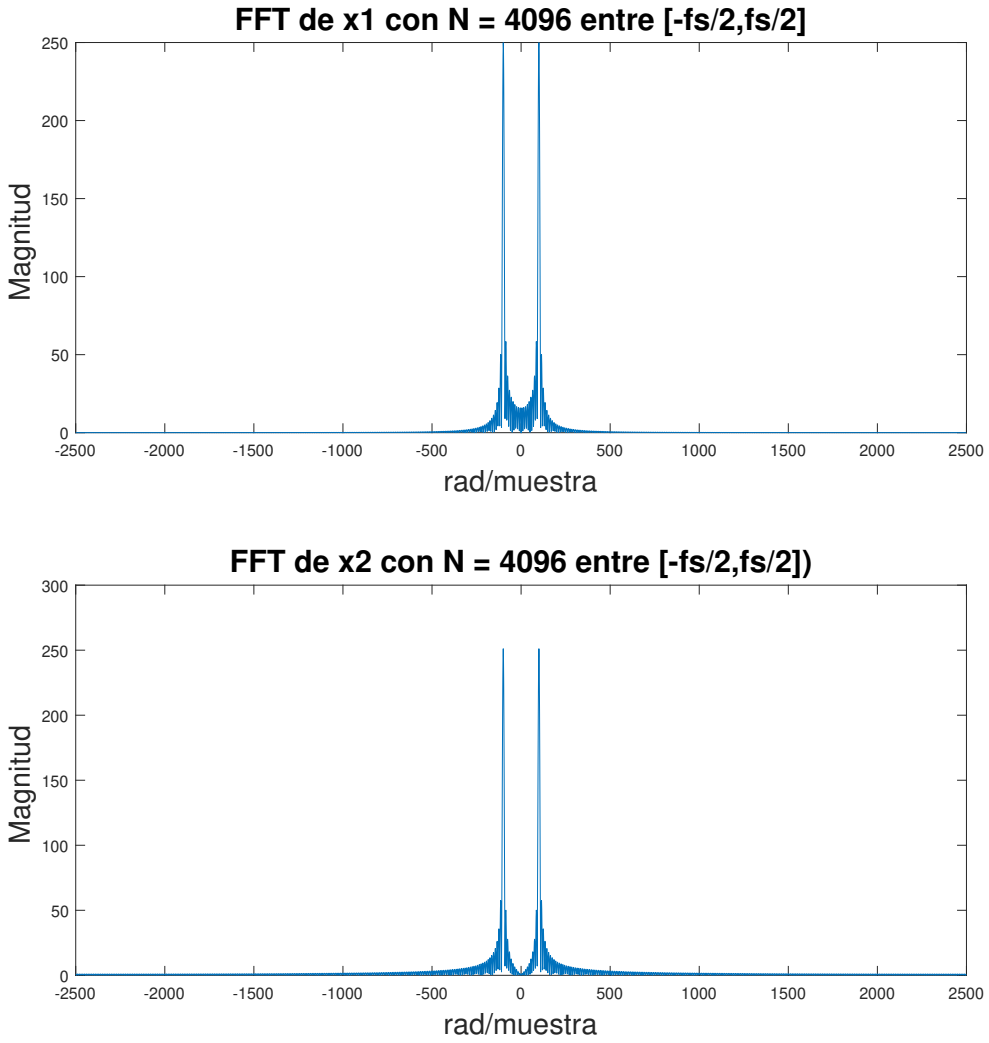

Figura 1: Magnitud de la fft para las señales x_1 y x_2 entre $[-\frac{fs}{2}, \frac{fs}{2}]$ con N = 4096

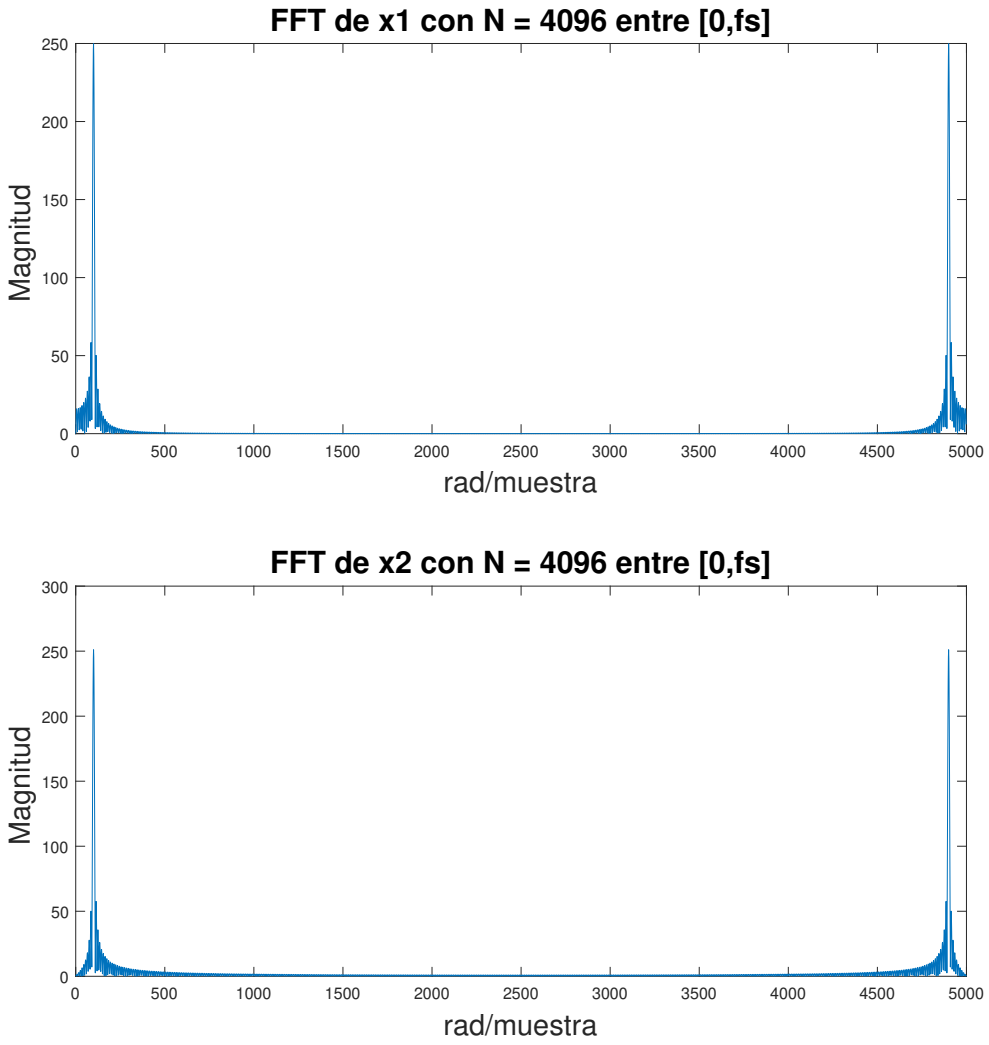

Figura 2: Magnitud de la fft para las señales x_1 y x_2 entre $[0, \frac{fs}{2}]$ con N = 4096

En las gráficas de ambas figuras se puede observar que el espectro tiene una amplitud de 250, lo que tiene sentido ya que este valor corresponde a la suma de todas las muestras y en este caso eran señales de 500 muestras, esto además de el factor de $\frac{1}{2}$ asociado a la transformada de Fourier de señales sinusoidales hace que los resultados obtenidos coincidan con los esperados.

3. Se obtiene la DFT de cada una de las señales anteriores, x_1 y x_2, para valores de $N = 256$, $N = 500$ y $N = 2048$, luego se obtienen gráficos de magnitud v/s Frecuencia además de la parte real y parte imaginaria v/s frecuencia. Las gráficas obtenidas para x_1 se muestran en la figura 3 mientras que las correspondientes a la señal x_2 se encuentran en la figura 4

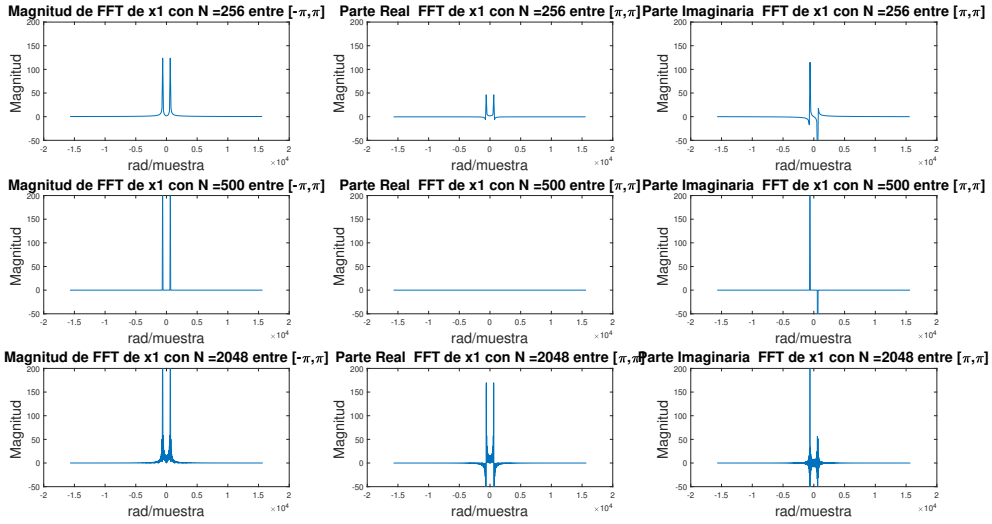

Figura 3: Magnitud y partes real e imaginaria de la señal $x_1 = sin(\omega t)$ para $N = 256, 500, 2048$

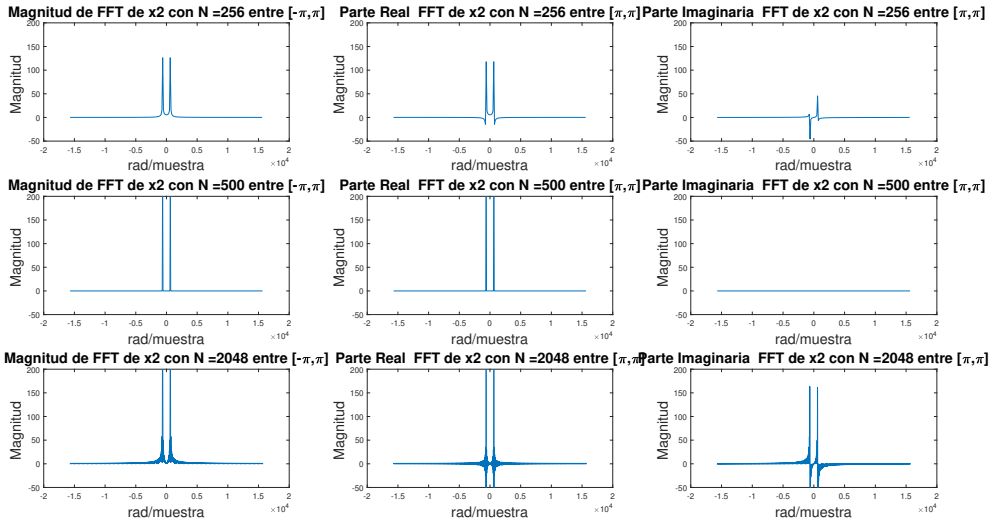

Figura 4: Magnitud y partes real e imaginaria de la señal $x_2 = cos(\omega t)$ para $N = 256, 500, 2048$

De las gráficas anteriores se puede concluir que el valor de N más apropiado es de 500, ya que los resultados obtenidos son los espectros más prolijos y cercanos a lo que se espera teóricamente de la transformada de ambas señales en caso de ser infinitas, esto se debe a que el $N = 500$ permite que los bines de frecuencia coincidan con las señales *sinc* asociadas a las ventanas usadas en la transformación.

En caso que se busque un resultado cercano para señales finitas el valor que proporciona mejores resultados es de $N = 2048$, en las graficas obtenidas con este N se pueden apreciar los lóbulos laterales que se generan por las ventanas asociadas a la transformación.

2. Estimación de frecuencia en presencia de ruido

Se genera una señal de 2 tonos puros de 100 y 500 Hz con amplitudes $0,5$ y $1,5$ respectivamente. Se le agrega a esta señal ruido blanco de distribución normal de media 0 y varianza 2, muestreando la señal resultante durante 100 ms a $f_s = 2000\ Hz$ para cumplir con el criterio de Nyquist.

En la figura 5 se muestra la representación temporal de la señal con ruido. A partir del gráfico no es posible reconocer que la señal limpia corresponde a la superposición de 2 tonos puros ni las frecuencias de estos.

Figura 5: Representación temporal de señal de 2 tonos puros con ruido

Luego se grafica la magnitud del espectro de las señales con y sin ruido usando el comando $fft()$ de MATLAB. Dichos gráficos se muestran en la figura 6

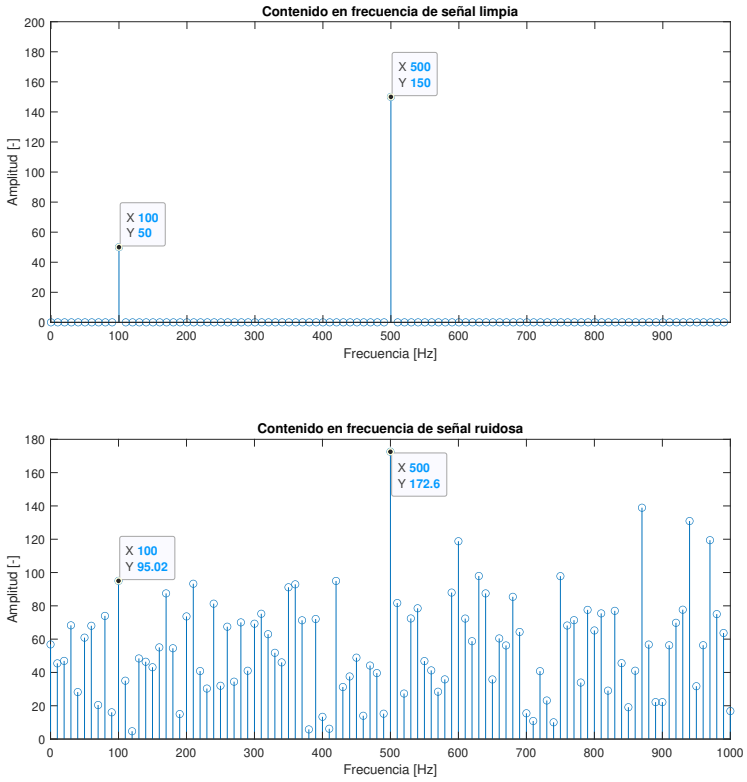

Figura 6: Magnitud del espectro de señal de 2 tonos puros sin y con ruido.

Con respecto a la figura 6 se aprecia que la amplitud de los tonos puros no es la misma en ambos casos. Lo anterior se debe a que el ruido agregado también tiene contenido en frecuencia en las frecuencias de los tonos puros.

Posteriormente se grafica la magnitud en dB del espectro en frecuencia de las señales con y sin ruido, normalizando la amplitud máxima a 0 dB. Dicho gráfico se muestra en la figura 7. En el caso de la señal limpia se observa que la diferencia de amplitud entre la componenente más elevada y el piso de ruido es de -300 dB aproximadamente. En el caso de la señal ruidosa es de aproximadamente -10 dB.

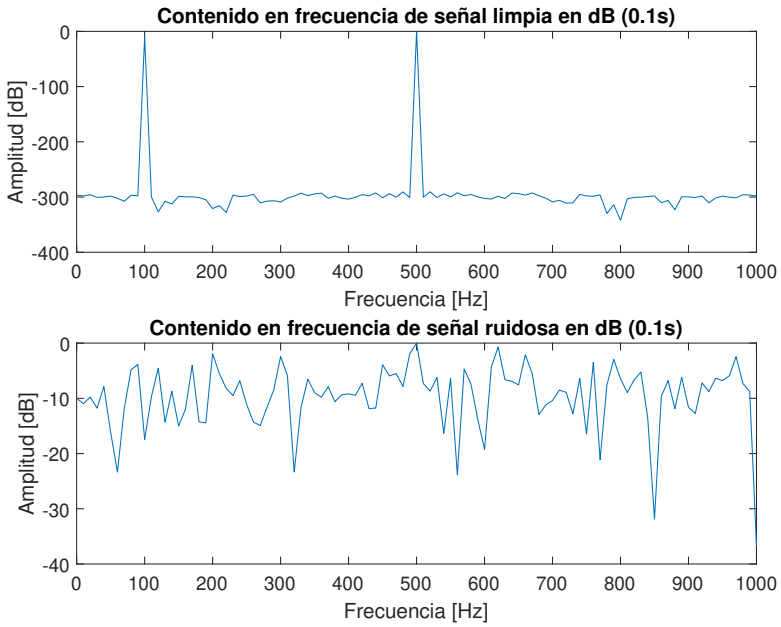

Figura 7: Magnitud en decibeles del espectro de señal de 2 tonos puros sin y con ruido, de duración 1 ms.

Finalmente se cambia la duración de la señal a 1s y se vuelve a graficar la magnitud normalizada en dB del espectro en frecuencia de las señales con y sin ruido. Dicho gráfico se muestra en la figura 8. Las diferencias entre el espectro encontrado con una ventana de 1 s y 0.1 s se debe a que la ventana de mayor largo en frecuencia es más angosta, lo que se traduce en un menor *leakage*.

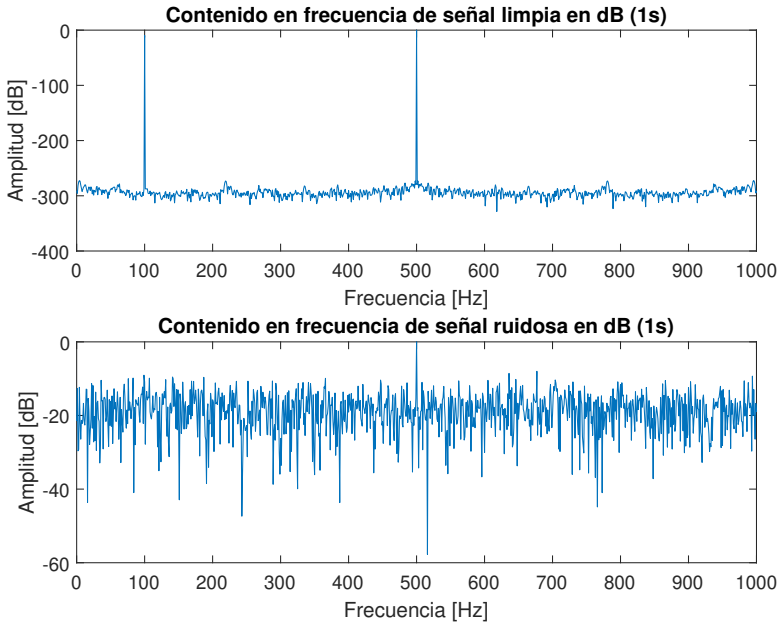

Figura 8: Magnitud en decibeles del espectro de señal de 2 tonos puros sin y con ruido, de duración 1 s.

3. Filtrado de señales en el dominio de la frecuencia

Se pide hacer un filtrado en frecuencia del archivo `nspeech.mat`, el cual presenta un tono molesto.

En primer lugar se obtiene la DFT de dicha señal y se grafica su magnitud en frecuencia, encontrando un peak en los 1685 Hz, localizando así el tono molesto. El gráfico obtenido se muestra en la figura 9.

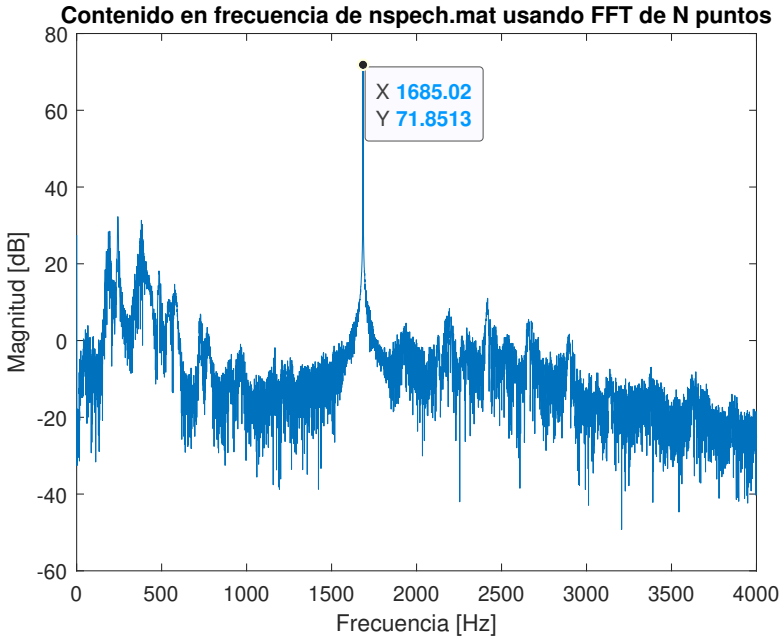

Figura 9: Contenido en frecuencia (magnitud) de señal `nspeech.mat`.

Posteriormente se diseña un filtro de 2 ceros conjugados, de respuesta en frecuencia:

$$H(\omega) = 1 - 2\cos(\theta)e^{-j\omega} + e^{-2j\omega}$$

Donde se ubican los ceros en el ángulo correspondiente a la frecuencia normalizada del tono molesto, es decir

$$\theta = 2\pi\frac{f_0}{f_s} = 2\pi\frac{1685}{8000} = 1{,}322\ \frac{\text{rad}}{\text{muestra}}$$

Con el objetivo de penalizar dicha frecuencia. La magnitud de la respuesta en frecuencia del filtro diseñado se muestra en la figura 10. Se aprecia la penalización correspondiente en la frecuencia del tono molesto.

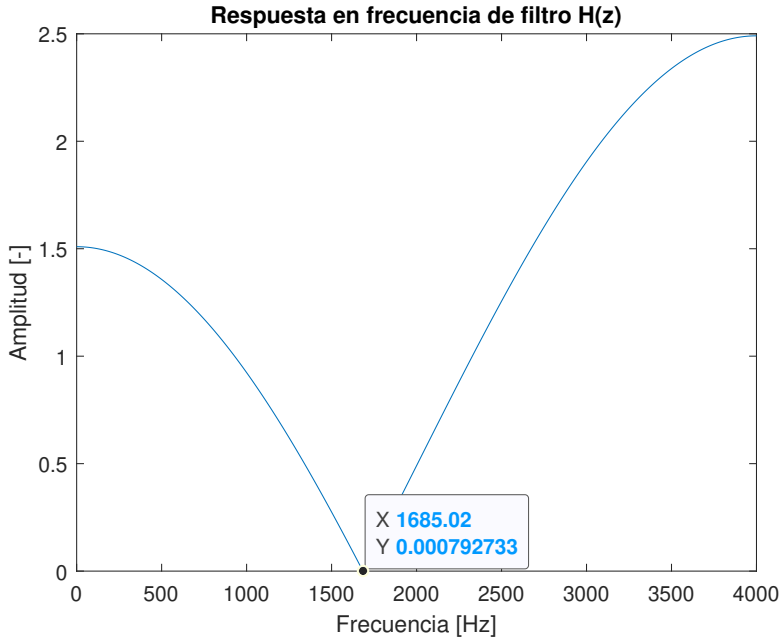

Figura 10: Respuesta en frecuencia (magnitud) de filtro de 2 ceros complejos conjugados diseñado.

Ya con el filtro diseñado, se realiza el filtrado en frecuencia haciendo un producto punto a punto de las respuestas en frecuencia del contenido en frecuencia de la señal `nspeech.mat` y la respuesta en frecuencia de H, teniendo cuidado con que la discretización en frecuencia de la expresión de $H(\omega)$ corresponda a la frecuencia de los bines de la DFT de la señal original.

La magnitud del contenido en frecuencia de la señal filtrada se muestra en la figura 11. En comparación con la figura 9 se aprecia una gran atenuación, más no eliminación del tono molesto. Lo anterior se le atribuye a temas numéricos.

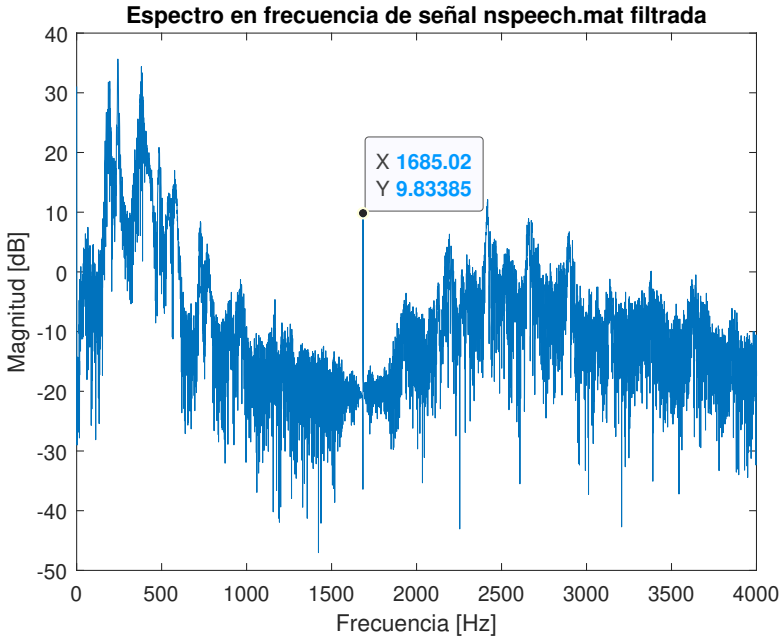

Figura 11: Contenido en frecuencia (magnitud) de señal `nspeech.mat` filtrada.

Finalmente se aplica la DFT inversa a la señal filtrada en el dominio de la

frecuencia para obtener su representación en el dominio del tiempo. Los gráficos de la señal pre y post filtrado se muestran en la figura 12. Visualmente aprecian buenos resultados producto al filtrado. Con respecto al número de puntos para recuperar la señal completa, corresponde al número de puntos de la señal `nspeech.mat`.

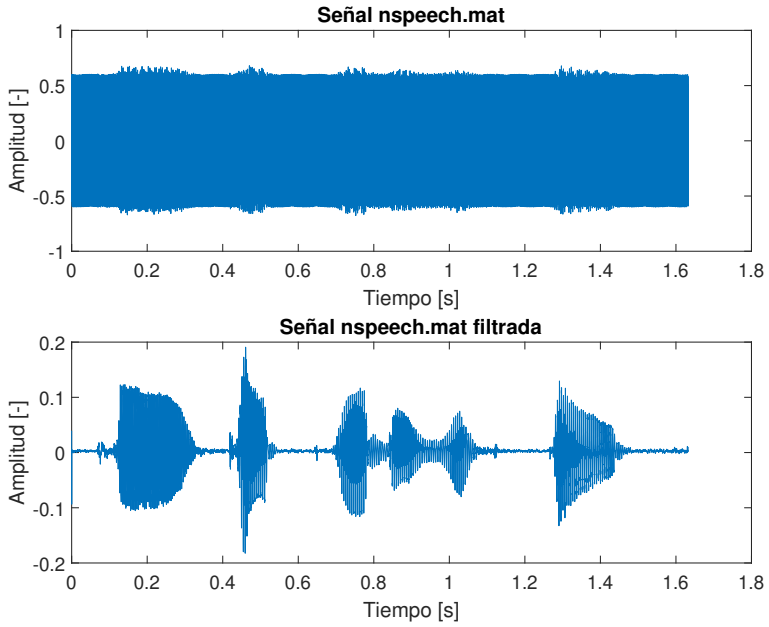

Figura 12: Representación de señal pre y post filtrado en el dominio temporal.

Con respecto a la percepción auditiva, se aprecia una gran disminución del tono molesto, sin embargo, sigue sonando levemente. Se adjuntan archivos de audio `nspeech_filtered.wav` y `nspeech_unfiltered.wav` a la entrega.

4. Cálculo directo de la DFT

Para efectuar el calculo directo de la DFT haciendo uso de la herramienta MATLAB se implementa la función `X= DFTsum(x)` para obtener la DFT de N puntos, siendo `x` el parámetro corresponde a una señal de entrada *x(n)* la cual desarrolla la ecuación 1 en base a *ciclos-for* calculando a la vez los exponenciales directamente. Esta código de implementación de la función se presenta a continuación

```
function X = DFTsum(x)
    N = length(x);
    X = zeros(1,N);
    for k = 0:N-1
        coef = exp(-1j*2*pi*k/N);
        for n = 0:N-1
            X(1,k+1) = X(1,k+1) + x(n+1)*coef^n;

        end

    end

end
```

$$X_N(k) = X^{(N)}(k) = \sum_{n=0}^{N-1} x(n) e^{-j2\pi kn/N} \tag{1}$$

1. Para probar la función implementada se generaron las siguientes señales

$$x_1(n) = \delta(n)$$
$$x_2(n) = 1$$
$$x_3(n) = e^{-j2\pi n/N}$$
$$x_4(n) = cos(2\pi n/N)$$

   Se grafica la salida de la función en el rango $[-\pi, \pi]$ para cada señal considerando 8 muestras para cada una, se grafica además, en el mismo rango de valores el resultado que entrega MATLAB al aplicar el comando `fft` a las señales generadas. Las gráficas mencionadas se muestran en la figura 13

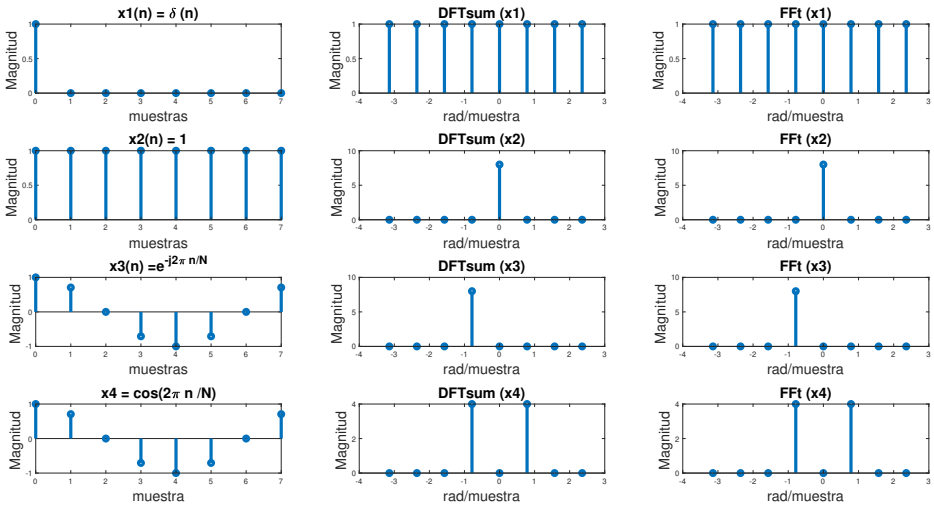

Figura 13: Señales x_1, x_2, x_3 y x_4 generadas junto a sus respectivas DFT obtenida con la función implementada y FFT obtenida con comandos en MATLAB

Se puede apreciar cómo el resultado que se obtiene mediante el comando `fft` de MATLAB es idéntico al que se obtiene al utilizar la función `DFTsum` siendo consistente con lo que se espera de su funcionamiento según la teoría. En el caso particular de la señal x_3 que corresponde a una señal compleja se presenta gráfica solo de la parte real asociada a su espectro en frecuencia por lo que el resultado es similar al de la señal x_4 pero con un solo impulso ($recordar\ 2cos(\phi) = e^{j\phi} + e^{j\phi}$).

Usando el comando de MATLAB `immse` se obtuvo la siguiente tabla resumen entre los valores numéricos obternidos usando la función `DFTsum` y el resultado entregado por el comando `fft` de MATLAB

Señal	MSE $DFT\ \ v/s\ \ fft$
$x_1(n) = \delta(n)$	0
$x_2(n) = 1$	$1{,}2195 \cdot 10^{-30}$
$x_3(n) = e^{-j2\pi n/N}$	$7{,}1631 \cdot 10^{-31}$
$x_4(n) = cos(2\pi n/N)$	$3{,}1880 \cdot 10{-}31$

Cuadro 1: Cuadro resumen para el error cuadrático medio entre el resultado de la función `DFTsum` y el resultado entregado por el comando `fft` de MATLAB para cada señal de prueba.

2.

3. Se considera la señal $x_1(t) = cos(wt)$ con frecuencia de 100 Hz, muestreada a 5kHz durante 1000 ms.

   Se grafica el error en la magnitud del espectro obtenido con la función `DFTsum` y `fft`. Dicho gráfico se muestra en la figura 14, donde se observa un máximo error del orden de 10^{-9}, por lo que se asume despreciable. El error cuadrático medio obtenido es de $5{,}4744 \cdot 10^{-22}$.

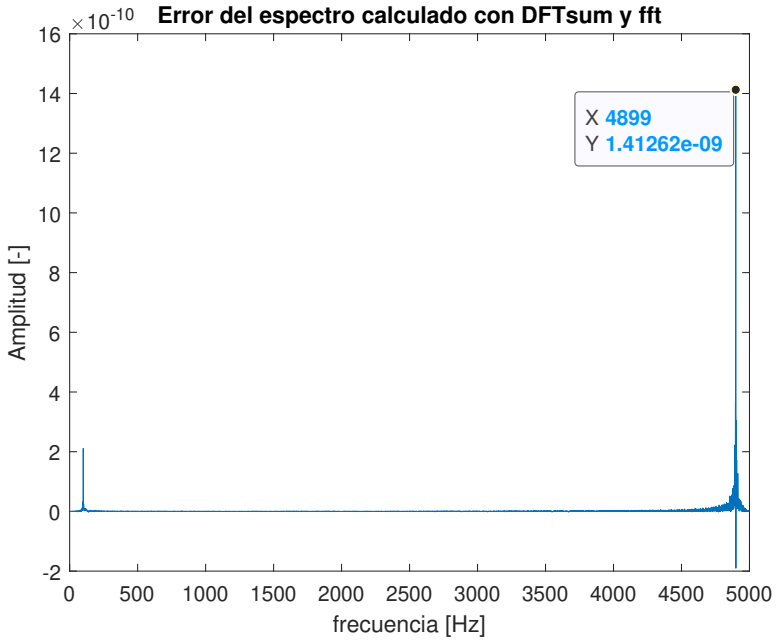

Figura 14: Error de magnitud del espectro utilizando `DFTsum` con respecto a `fft` en MATLAB.

5. Cálculo matricial de la DFT

1. En esta sección se busca poder obtener la DFT de gracias a un como un producto entre una matriz y un vector, de modo que $X = Ax$, donde X y x son vectores columna de $N \times 1$ y A es una matriz cuadrada de $N \times N$. Este producto es equivalente a

$$X_k = \sum_{n=1}^{N} A_{kn} X_n$$

donde A_{kn} es el elemento (k, n) de la matriz dado por

$$A_{kn} = e^{-j2\pi(k-1)(n-1)/N}$$

Para poder implementar esta forma en primera instancia se debe implementa la función `genAmatrix(N)`, para lo que se utilizó el siguiente código en base a *ciclos-for*

```
function A = genAmatrix(N)
    A = zeros(N,N);

    for k =1:N
        for  n = 1:N
            A(k,n) = exp(-1j*2*pi*(k-1)*(n-1)/N);
        end

    end

end
```

Para evaluar la función generadora de la matriz A se compara sus resultados con los que se obtienen al generar la misma matriz con elcomando *dftmtx* de MATLAB para los valores N = 2,4,8,16 y 32 . Se el error cuadrático medio entre ambos métodos, obteniendo datos que se presentan en la siguiente tabla

N	MSE $genAmatrix(N)$ v/s $dftmtx$
2	$3{,}749 \cdot 10^{-33}$
4	$4{,}5930 \cdot 10^{-32}$
8	$7{,}9370 \cdot 10^{-31}$
16	$7{,}7116 \cdot 10^{-30}$
32	$3{,}1181 \cdot 10^{-29}$

Cuadro 2: Cuadro resumen para el error cuadrático medio entre el resultado de la función `genAmatrix(N)` y el resultado entregado por el comando `dftmtx` de MATLAB para N = 2,4,8,16 y 32.

Todos los errores obtenidos son despreciables para fines prácticos debido al orden de magnitud que poseen en comparación a los valores que se utilizan en los cálculos de interés.

Para poder efectuar finalmente el cálculo de la DFT mediante matrices se implementa la función `DFTmatrix(x)`, la que invoca a la función `genAmatrix(N)`. Para esto se utiliza el código que se presenta a continuación

```
function X = DFTmatrix(x)
    N = length(x);
    A = genAmatrix(N);
    X = sum(A'.*x');
end
```

2. Se pide graficar la parte imaginaria y la parte real de las matrices de DFT para $N = 8$ y 64.

   Las matrices de la DFT de 8 puntos se muestran en la figura 15.

   Con respecto a la parte real se aprecia simetría con respecto a la diagonal principal (matriz simétrica). Sin consderar la primera fila y columna, se observa simetria con respecto a la columna y fila central.

   Con respecto a la parte imaginaria, se aprecia simetría con respecto a la diagonal principal (matriz simétrica). Sin consderar la primera fila y columna, se observa simetria con respecto también a la diagonal secundaria.

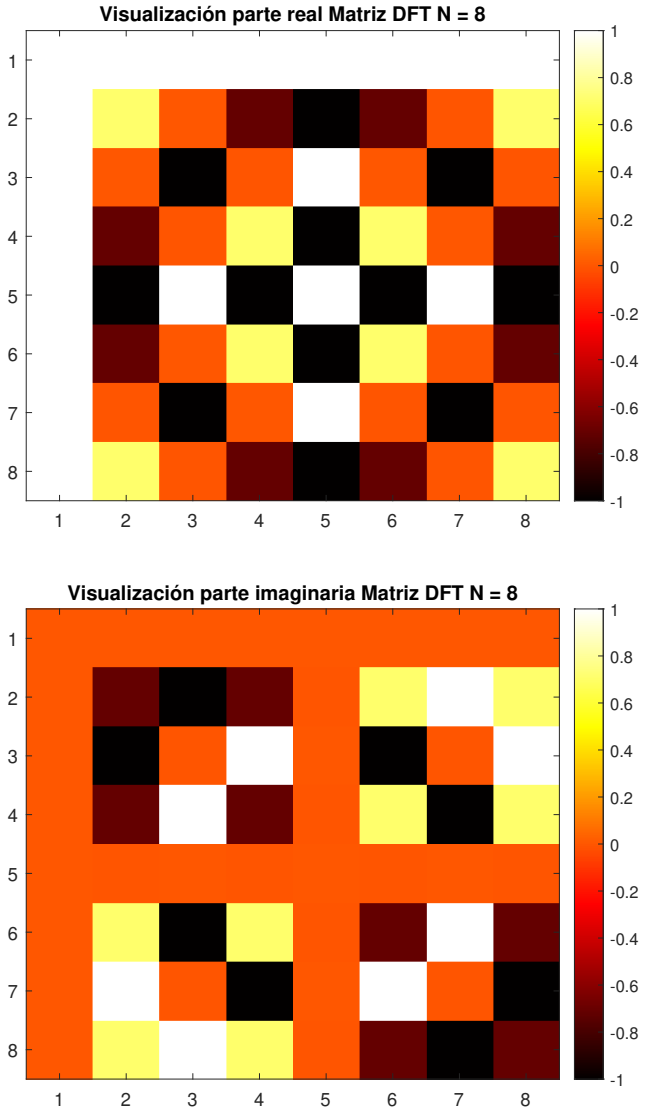

Figura 15: Visualización parte real e imaginaria de matriz DFT de 8 puntos.

Las matrices de la DFT de 64 puntos se muestran en la figura 16.

No se aprecian diferencias en las simetrías apreciadas en la parte real e imaginaria con respecto a la matriz de 8 puntos.

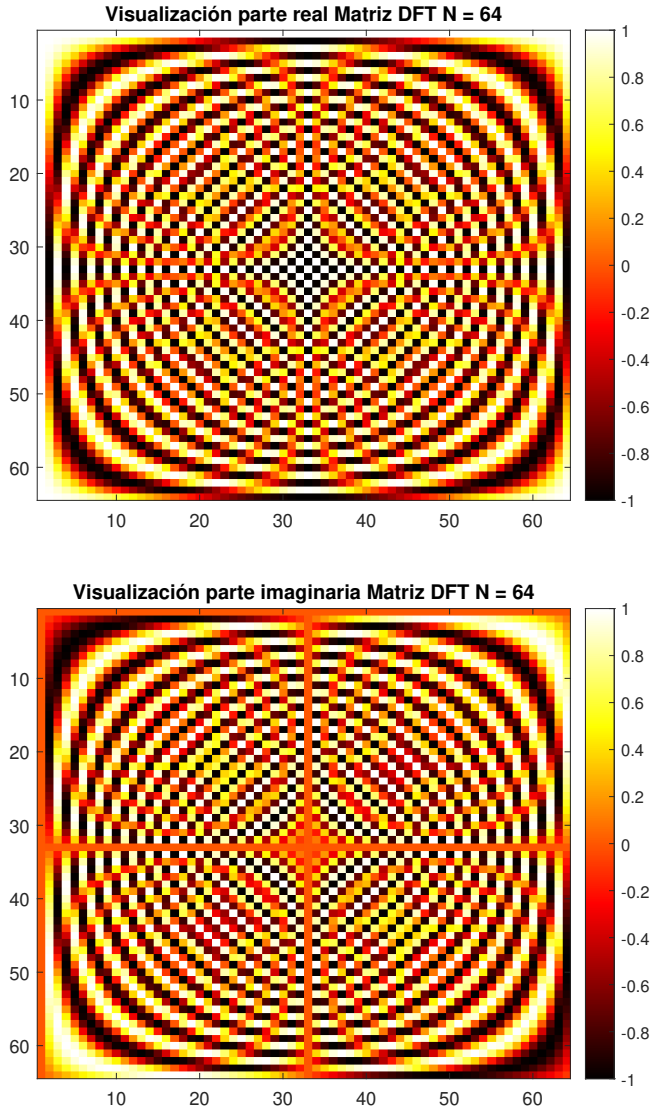

Figura 16: Visualización parte real e imaginaria de matriz DFT de 64 puntos.

3. Para probar la implementación de la función `DFTmatrix(x)`, se procede de la misma forma que al probar la implementación de la función `DFTsum(x)` del inciso anterior. Se usan las mismas señales $x1, x_2, x_3$ y x_4 también con 8 muestras para cada una de ellas. Se grafica la para cada señal el resultado entregado por la función implementada así como el resultado que se obtiene haciendo uso del comando *fft* de MATLAB. Las gráficas mencionadas se muestran en la figura 17

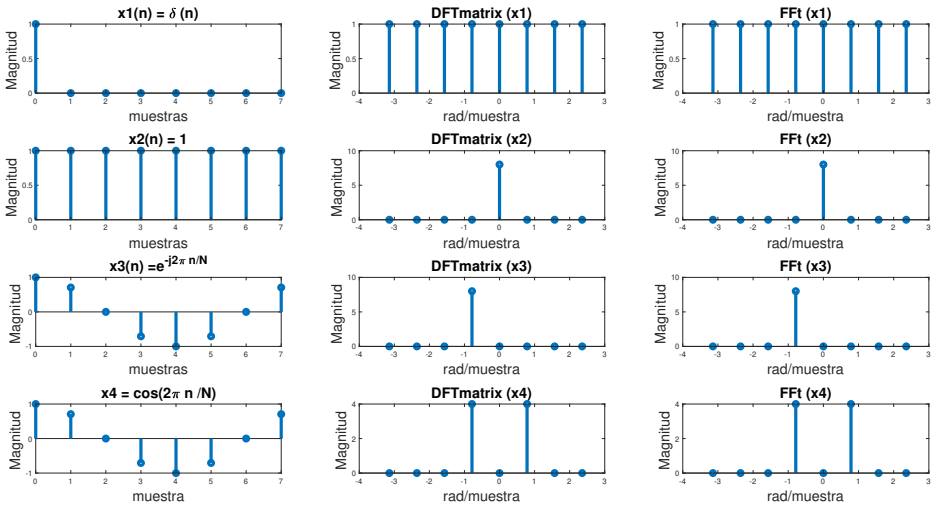

Figura 17: Señales x_1, x_2, x_3 y x_4 generadas junto a sus respectivas DFT obtenida con la función implementada y FFT obtenida con comandos en MATLAB

Como se puede apreciar en las gráficas, al igual que en el caso anterior con la función `DFTsum(x)`, esta vez para la función `DFTmatrix(x)` se obtienen resultados cualitativamente idénticos a los que se obtienen haciendo uso del comando *fft* de MATLAB. Cabe mencionar que nuevamente para el caso de la señal x_3 se ha graficado unicamente la parte real correspondiente.

A continuación se presenta una tabla con los errores cuadráticos medios asociados a ambas funciones implementadas que homologan el comportamiento del comando *fft* de MATLAB.

Señal	MSE DFTsum v/s fft	MSE DFTmatrix v/s fft
$x_1(n) = \delta(n)$	0	0
$x_2(n) = 1$	$1{,}2195 \cdot 10^{-30}$	$3{,}8323 \cdot 10^{-30}$
$x_3(n) = e^{-j2\pi n/N}$	$7{,}1631 \cdot 10^{-31}$	$3{,}8254 \cdot 10^{-30}$
$x_4(n) = cos(2\pi n/N)$	$3{,}1880 \cdot 10{-}31$	$5{,}9170 \cdot 10^{-31}$

Cuadro 3: Cuadro resumen para el error cuadrático medio entre el resultado de las funciones `DFTsum` y `DFTmatrix` con respecto al resultado entregado por el comando `fft` de MATLAB para cada señal de prueba.

Todos los errores encontrados son despreciables ya que su orden de magnitud no es significativo respecto a las magnitudes con las que se trabaja en el contexto actual.

4. Se utiliza el comando `timeit` de MATLAB para medir el tiempo de procesamiento de los métodos $fft, DFTsum$ y $dftmtx$, considerando $N = (100 : 100 : 5000)$ puntos.

   Se grafica el tiempo de procesamiento para los valores de N, donde se escoge escala logarítmica para fines comparativo. El gráfico se muestra en la figura 18.

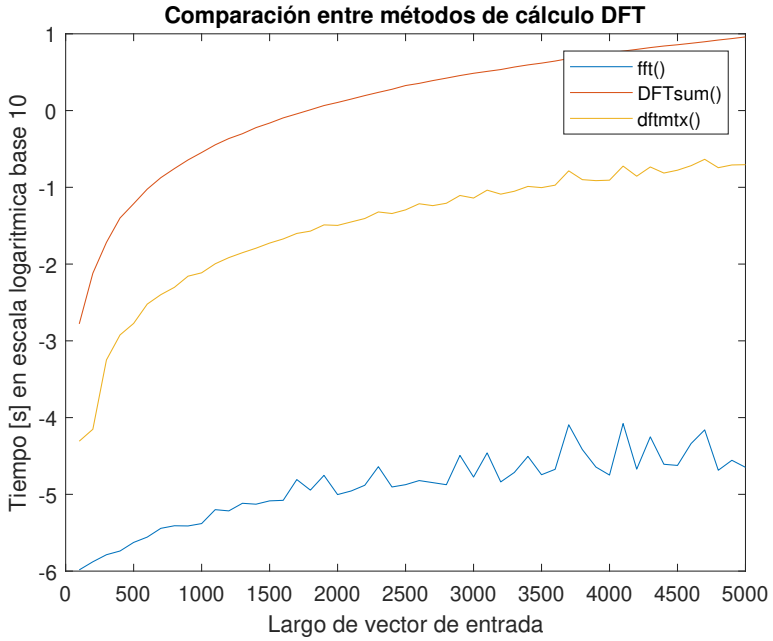

Figura 18: Tiempo de procesamiento para funciones $fft, DFTsum$ y $dftmtx$, considerando $N = (100 : 100 : 5000)$ puntos.

Con respecto al tiempo de procesamiento se concluye que:

- Para los largos probados de entrada, la fft resulta más rápida que el método $dftmtx$ y este a su vez que el $DFTmtx$.
- La razón del tiempo de procesamiento entre $DFTsum$ y fft es de aproximadamente 10^{-6}, a partir de largos de 1000 muestras. Antes de las 1000 muestras la razón va en ascenso
- La razón del tiempo de procesamiento entre $DFTsum$ y fft es de aproximadamente 10^{-4}, a partir de largos de 1000 muestras. Antes de las 1000 muestras la razón va en ascenso.
- La La razón del tiempo de procesamiento entre $DFTsum$ y $dftmtx$ es aproximadamente 10^{-2}, para todos los largos de datos probados.

Con respecto a la memoria utilizada, claramente el método $dftmtx$ es el que ocupa más debido a que gerena la matriz de transformación que termina siendo de $N \times N$

6. Algoritmo Radix-2 de la FFT

1. Se experimenta con lograr la FFT con solo una reducción de orden. Para lo anterior se escribe la función $X = DFTdc(x)$, la cual implementa una etapa del algoritmo recursivo de la FFT y luego utiliza la función $DFTsum$ para obtener la DFT de las señales de muestras pares e impares.

   El código escrito para la función $DFTdc$ se muestra a continuación:

```
function X = DFTdc(x)
    N  = length(x);
    W  = exp(-1j*2*pi/N);
    Wk = W.^(0:(N/2-1));

    x0 = zeros(1,N/2);
    x1 = zeros(1,N/2);

    for i = 1:N/2
        x0(i) = x(2*i-1); %OBS "par"   1,3,5,... en MATLAB
        x1(i) = x(2*i);   %OBS "impar" 2,4,6,... en MATLAB
    end

    X0 = DFTsum(x0); X1 = DFTsum(x1);

    X_izq = X0 + Wk.*X1;
    X_der = X0 - Wk.*X1;

    X = [X_izq X_der];
end
```

   Posteriormente se utiliza la función generada para obtener la DFT de 3 señales de prueba:

   - Señal a: $x(n) = \delta(n)$
   - Señal b: $x(n) = 1$
   - Señal c: $x(n) = e^{-j\frac{2\pi n}{8}}$

   Las magnitudes de la DFT obtenidas con la función para las señales de prueba se muestran en la figura 19, donde además se muestran las magnitudes de la DFT utilizando el comando fft de MATLAB con fines comparativos. Como era de esperarse, no se aprecian diferencias visuales en los resultados obtenidos.

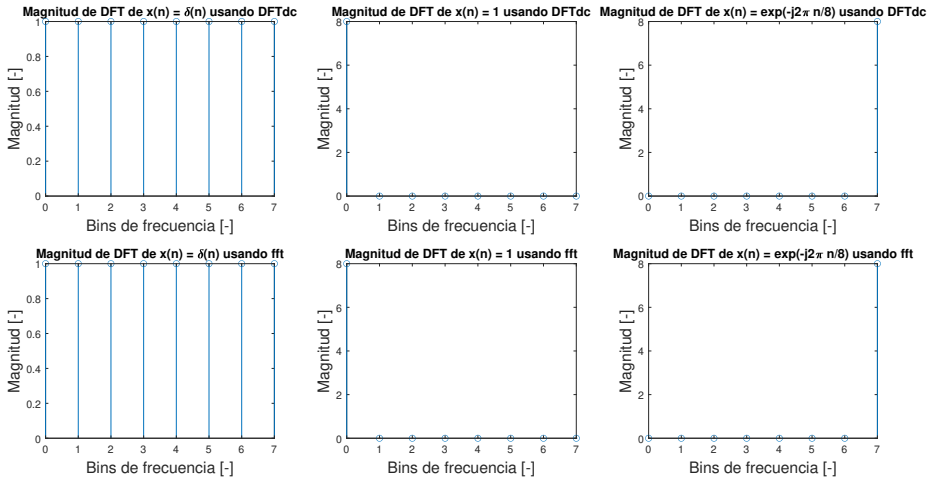

Figura 19: Comparación entre magnitudes de DFT obtenidas con función $DFTdc$ y fft en MATLAB.

Finalmente se obtiene el tiempo de procesamiento para la función $DFTdc$ y fft, obteniendo:

- $DFTdc$: $3{,}64123 \cdot 10^{-5}$ s
- fft: $9{,}84100 \cdot 10^{-7}$ s

Se aprecia que la implementación de la fft de MATLAB es muy superior a la función generada al calcular una DFT de 8 puntos.

2. Se experimenta con FFTs de 2, 4 y 8 puntos, programando las funciones $FFT2, FFT4$ y $FFT8$, las cuales corresponden a implementaciones explícitas de las ecuaciones del algoritmo fft.

   En primer lugar se implementa la función $FFT2$, a partir de las ecuaciones:

$$X^{(2)}(0) = x(0) + x(1)$$
$$X^{(2)}(1) = x(0) - x(1)$$

   Donde $X^{(2)}$ corresponde a la DFT de la señal x, la cual tiene 2 puntos.

   El código escrito para la función $FFT2$ se muestra a continuación:

```
function X = FFT2(x)
    X = zeros(1,2);

    X(1) = x(1) + x(2);
    X(2) = x(1) - x(2);
end
```

   Las funciones $FFT4$ y $FFT8$, corresponden a la implementación con $N = 4$ y 8 respectivamente de las siguientes ecuaciones:

$$X^{(N)}(k) = X_0^{(N/2)}(k) + W_N^k X_1^{(N/2)}(k)$$
$$X^{(N)}(N/2+k) = X_0^{(N/2)}(N/2+k) - W_N^k X_1^{(N/2)}(N/2+k)$$

   Donde

- $X^{(N)}$ corresponde a la DFT de la señal x, la cual tiene N puntos
- $X_0^{(N/2)}(k)$ y $X_0^{(N/2)}(k)$ corresponden a la DFT de la señal de muestras pares e impares de x.
- $W_N = e^{(-j2\pi/N)}$ corresponde al factor de twiddle.

La implementación de $FFT4$, se muestra a continuación:

```
function X = FFT4(x)
    X = zeros(1,4);

    x0 = [x(1) x(3)]; %Muestras "pares"
    x1 = [x(2) x(4)]; %Muestras "impares"

    X0 = FFT2(x0);
    X1 = FFT2(x1);

    X(1) = X0(1) +   (1)*X1(1);
    X(2) = X0(2) + (-1j)*X1(2);

    X(3) = X0(1) -   (1)*X1(1);
    X(4) = X0(2) - (-1j)*X1(2);
end
```

La implementación de $FFT8$ se muestra a continuación:

```
function X = FFT8(x)
    X = zeros(1,8);

    x0 = [x(1) x(3) x(5) x(7)]; %Muestras "pares"
    x1 = [x(2) x(4) x(6) x(8)]; %Muestras "impares"

    X0 = FFT4(x0);
    X1 = FFT4(x1);

    X(1) = X0(1) +                (1)*X1(1);
    X(2) = X0(2) +   (exp(-1j*pi/4))*X1(2);
    X(3) = X0(3) +             (-1j)*X1(3);
    X(4) = X0(4) + (exp(-1j*3*pi/4))*X1(4);

    X(5) = X0(1) -                (1)*X1(1);
    X(6) = X0(2) -   (exp(-1j*pi/4))*X1(2);
    X(7) = X0(3) -             (-1j)*X1(3);
    X(8) = X0(4) - (exp(-1j*3*pi/4))*X1(4);
end
```

Posteriormente se utiliza la función $FFT8$ generada para obtener la DFT de 3 señales de prueba del punto anterior.

Las magnitudes de la DFT obtenidas con la función para las señales de prueba se muestran en la figura 20, donde además se muestran las magnitudes de la DFT utilizando el comando fft de MATLAB con fines comparativos. Como era de esperarse, no se aprecian diferencias visuales en los resultados obtenidos.

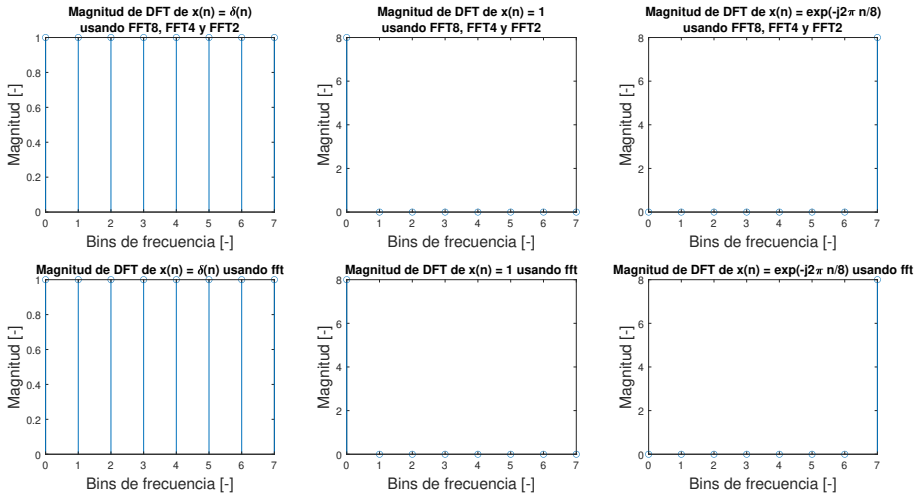

Figura 20: Comparación entre magnitudes de DFT obtenidas con función $FFT8$ y fft en MATLAB.

Finalmente se obtiene el tiempo de procesamiento para la función $FFT8$ y fft, obteniendo:

- $FFT8$: $1{,}63822 \cdot 10^{-5}$ s
- fft: $7{,}75409 \cdot 10^{-7}$ s

Se aprecia que la implementación de la fft de MATLAB sigue siendo muy superior a la función generada al calcular una DFT de 8 puntos. Sin embargo, cabe destacar que hubo una mejora con respecto al algoritmo $DFTdc$.

3. Se crea la función recursiva $X = fft\_stage(x)$, la cual aplica el algoritmo FFT para señales de largo 2^n. El código se muestra a continuación:

```
function X = fft_stage(x)
    N = length(x);

    if (N == 2)
        X = FFT2(x);
        return
    else
        x0    = zeros(1,N/2); x1    = zeros(1,N/2);

        for i = 1:N/2
            x0(i) = x(2*i-1); %OBS "par"   1,3,... en MATLAB
            x1(i) = x(2*i);   %OBS "impar" 2,4,... en MATLAB
        end

        W = exp(-1j*2*pi/N);  Wk = W.^(0:(N/2-1));
        X0 = fft_stage(x0) ;  X1 = fft_stage(x1) ;

        X_izq = X0 + Wk.*X1;
        X_der = X0 - Wk.*X1;

        X = [X_izq X_der];
    end
end
```

Se utiliza el comando `timeit` de MATLAB para comparar el tiempo de procesamiento al utilizar de entrada la señal del punto **4.3**, considerando largos de $N = [2^{10}, 2^{11}, \ldots, 2^{19}]$. Los gráficos con los tiempos de cómputo se muestran en la figura 21

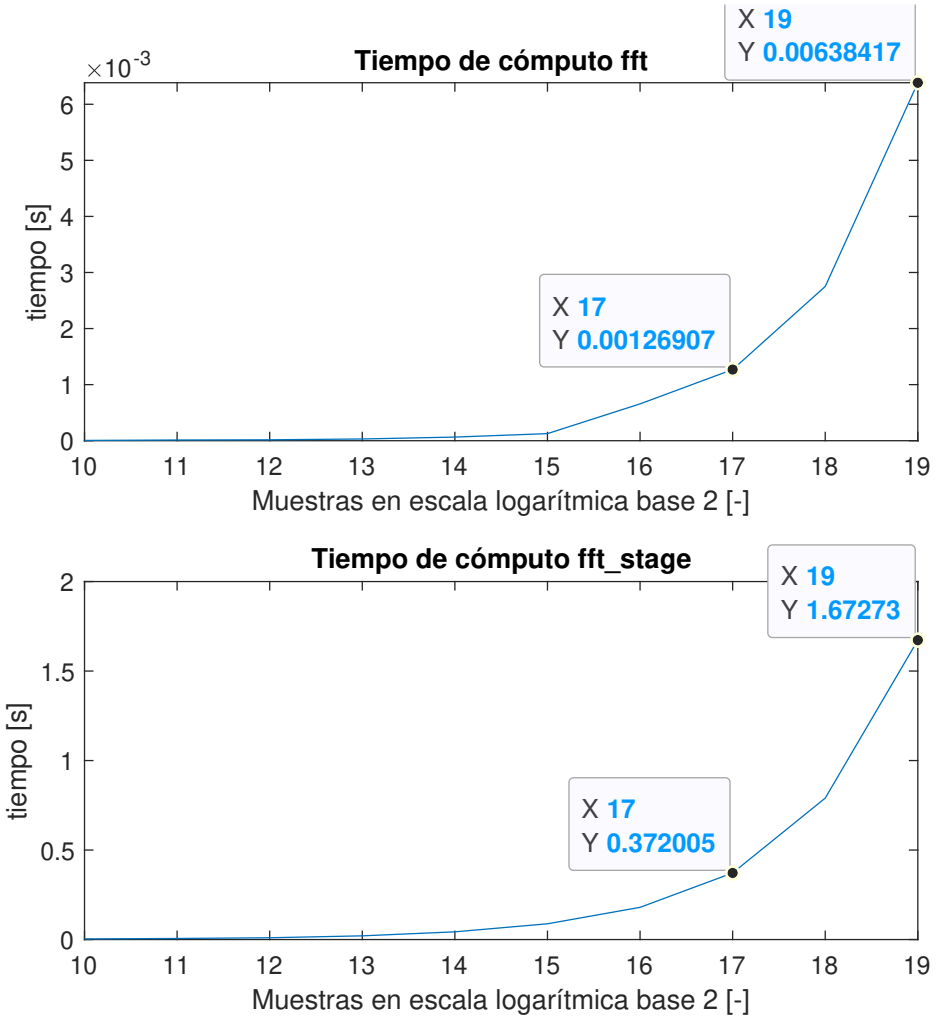

Figura 21: Tiempos de cómputo de función fft_stage y fft para distintos largos de señal.

De los resultados de tiempo de cómputo obtenidos se concluye que:

- La implementación de MATLAB es muy superior a fft_stage. La razón entre el tiempo de cómputo de la fft con respecto a fft_stage es del orden de $10^{-}3$.
- La curva de tiempo de cómputo en ambos algoritmos son similares.

7. Algoritmo Goertzel

Se busca poder realizar una implementación del *algoritmo de Goertzel* en lenguaje C mediante el uso de *S-Function* en MATLAB trabajando sobre el proyecto entregado como material de apoyo para la experiencia del laboratorio.

1. Se crea la función *computeGoertzel()* que recibe como argumentos el puntero al estado del filtro Goertzel y una muestra de entrada con el código que se muestra a continuación

```
1  static double computeGoertzel(goertzelState_t *state,
2                                double filterInput)
3
4  //Incrementa contador
5      state->samplesCounter += 1;
6  //IIR
7  state->outputs[2] = state->outputs[1];
8  state->outputs[1] = state->outputs[0];
9  state->outputs[0] = filterInput +
10                     (state->A1)*state->outputs[1]
11                     - state->outputs[2];
12
13 //FIR
14   if (GOERTZEL_N <= state->samplesCounter){
15     state->binReal = state->cosW*state->outputs[1]
16                     - state->outputs[2];
17     state->binImag = state->sinW*state->outputs[1];
18     state->binMag = sqrt((state->binReal*state->binReal)
19                          +(state->binImag*state->binImag)  );
20
21     resetGoertzel(state);
22   }
23
24     return state->binMag;
25 }
```

Otras funciones relevantes en el funcionamiento del algoritmo de Groetzel son aquellas que inicializan y reinician los estados del filtro. La función `initGoertzel()` encargada de la inicialización se muestra a continuación, mientras que la función `resetGoertzel` solo reestablece en cero los valores del vector de salidas y el contador de muestras presentes en la estructura entregada para el laboratorio.

```
1  static void initGoertzel(goertzelState_t *state,
2                           uint64_t kFrequency)
3  {
4    state->samplesCounter     = 0;
5    state->cosW = cos(2*M_PI*kFrequency/GOERTZEL_N);
6    state->sinW = sin(2*M_PI*kFrequency/GOERTZEL_N);
7    state->A1  = 2*cos(2*M_PI*kFrequency/GOERTZEL_N);
8
9    state->outputs[0] = 0.0;
10   state->outputs[1] = 0.0;
11   state->outputs[2] = 0.0;
12
13   state->binReal =0.0;
```

```
14    state->binImag=0.0;
15    state->binMag = 0.0;
16
17    return;     }
```

El algoritmo de Goertzel entrega la magnitud de bines específicos en el espectro de una señal de N muestras, para evaluar el correcto funcionamiento de la implementación hecha se utiliza el archivo de audio *dtmfSequence_16_16.wav*, para el cual se obtiene la DFT para los primeros 256 valores con el comando de MATLAB *fft* y luego se grafica indicando el valor que resulta en los bines 8,9 y 10. Por otro lado se simula el algoritmo con simulink y se analiza la primera muestra (definida por los 256 primeros datos del archivo de audio) y se compara la magnitud presente en el gráfico de la DFT con la magnitud obtenida mediante simulación.

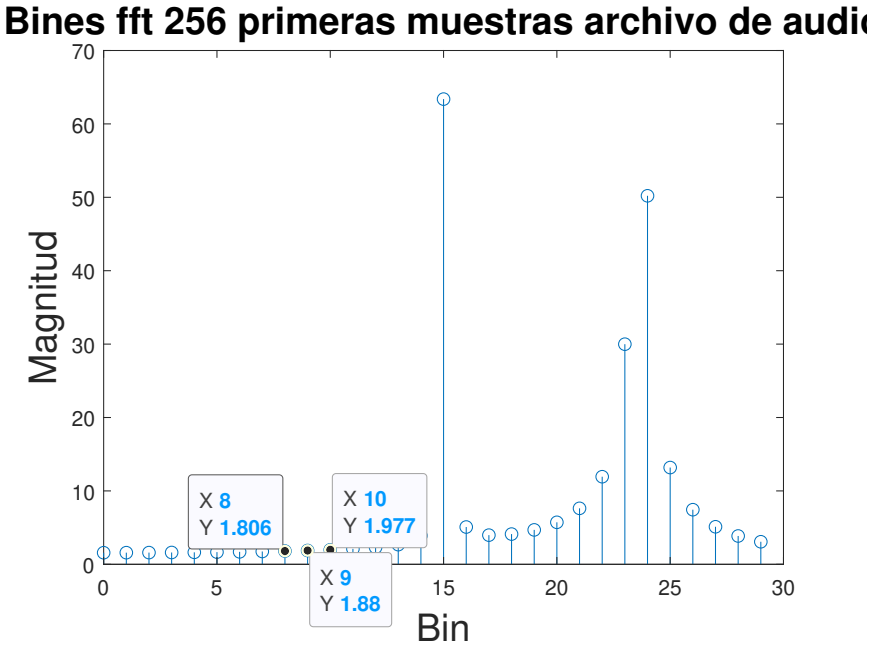

Figura 22: Gráfica de los bines obtenidos por la fft de las primeras 256 muestras del archivo de audio *dtmfSequence_16_16.wav*.

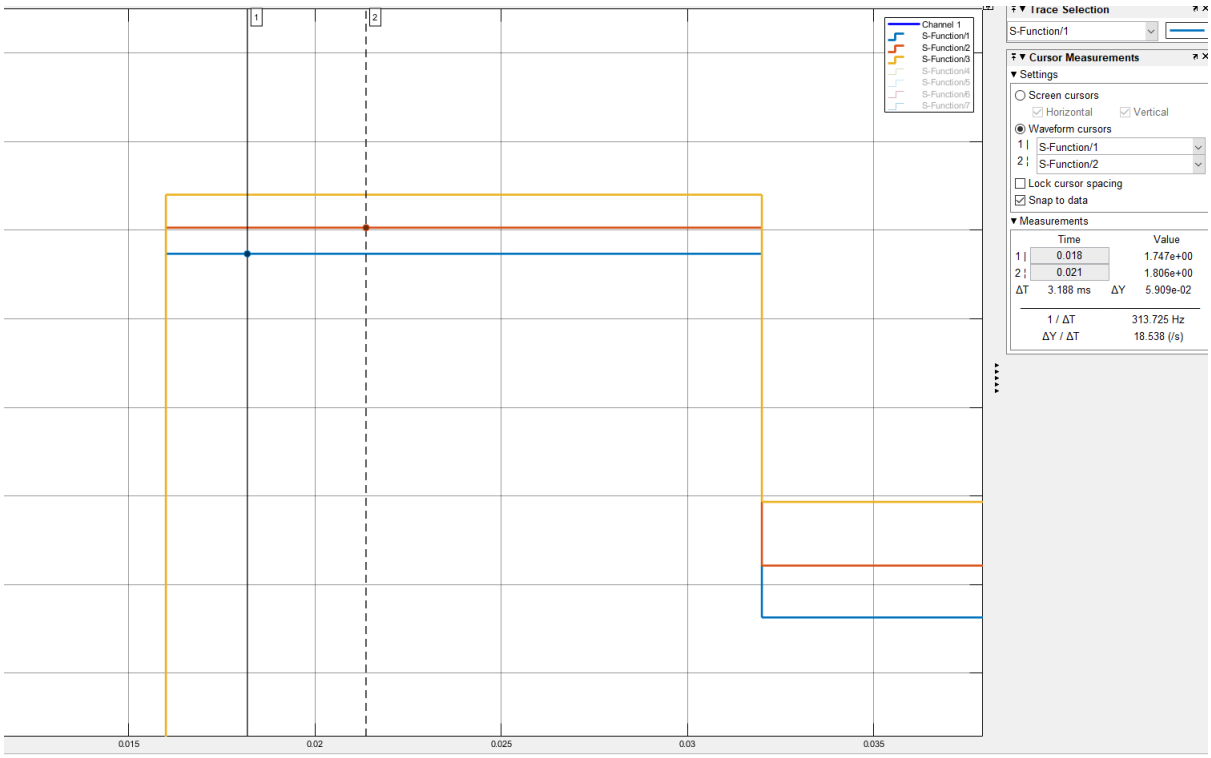

Figura 23: Magnitud de los bines 8, 9 y 10 asociados al archivo de audio *dtmfSequence_16_16.wav*, obtenidas mediante simulación.

En la última figura las leyendas *S-Function/1*, *S-Function/2* y *S-Function/3* corresponden a las magnitudes de los bines 8, 9 y 10 respectivamente. Comparando los valores que se muestran en el cuadro de datos a la derecha con los valores que indican los cuadros *data tip* para los bines 8, 9 y 10 se puede ver que coinciden por lo que se concluye que la implementación del algoritmo de Goertzel funciona de manera correcta.

2. Se instancia un total de 7 filtro Goertzel para lograr obtener los bins de frecuencia más cercanos a las frecuencias de los DTMF de la experiencia anterior, considerando el mismo archivo de usado anteriormente, el cual posee una frecuencia de muestreo de 16 *ksps* y nuevamente se analizan una cantidad N = 256 de muestras.

   Para escoger correctamente los k bins correspondiente a cada frecuencia DTMF se usa la expresión

$$k - Bin = \frac{f_{DTMF} \cdot N}{Fs}$$

   Redondeando de ser necesario el resultado, pues los $k - bines$ han de ser números enteros. De esta forma se obtiene la siguiente tabla

frecuencia DTMF	k-Bin
697 Hz	11
770 Hz	12
852 Hz	14
941 Hz	15
1209 Hz	19
941 Hz	15
1336 Hz	21
1336 Hz	21

Cuadro 4: Cuadro resumen para el error cuadrático medio entre el resultado de la función `DFTsum` y el resultado entregado por el comando `fft` de MATLAB para cada señal de prueba.

Se modifica la sección de código correspondiente quedando de la siguiente forma

```
1  #define GOERTZEL1_K_BIN          (11)
2  #define GOERTZEL2_K_BIN          (12)
3  #define GOERTZEL3_K_BIN          (14)
4  #define GOERTZEL4_K_BIN          (15)
5  #define GOERTZEL5_K_BIN          (19)
6  #define GOERTZEL6_K_BIN          (21)
7  #define GOERTZEL7_K_BIN          (24)
```

Usando los comandos *subplot* y *bar* se grafican los resultados para las 7 salidas de los filtros Goertzel, y tal como se esperaba, se cumple que en todo momento existen solo dos de las 7 componentes (bines) con magnitudes altas , correspondientes a las frfecuencias DTMF. Esta gráfica se muestra en la figrua 24

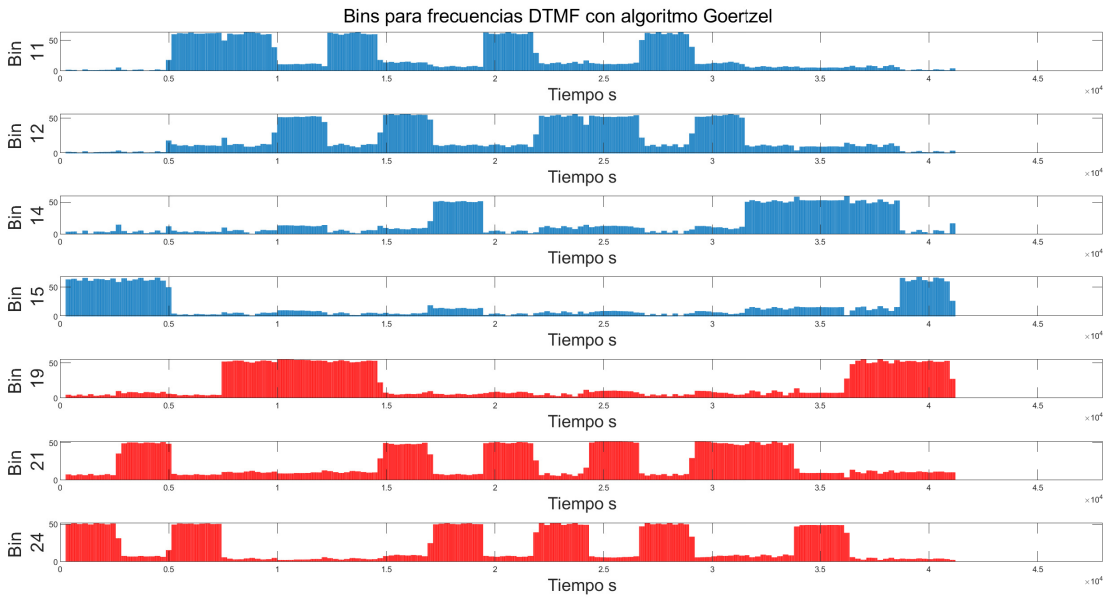

Figura 24: Bins asociados a las frecuencias DTMF graficados en el tiempo usando el algoritmo de Goertzel