Práctica 4 de Señales y Sistemas

Señales y sistemas discretos en el dominio de la frecuencia

Francisco Javier Mercader Martínez

Rubén Gil Martínez

1. Transformada de Fourier de secuencias

Cuestiones

Realice un script en Matlab para representar una secuencia **x** y su transformada de Fourier **X**. Las secuencias están muestreadas a una frecuencia **fs**. Para ello siga los siguientes pasos:

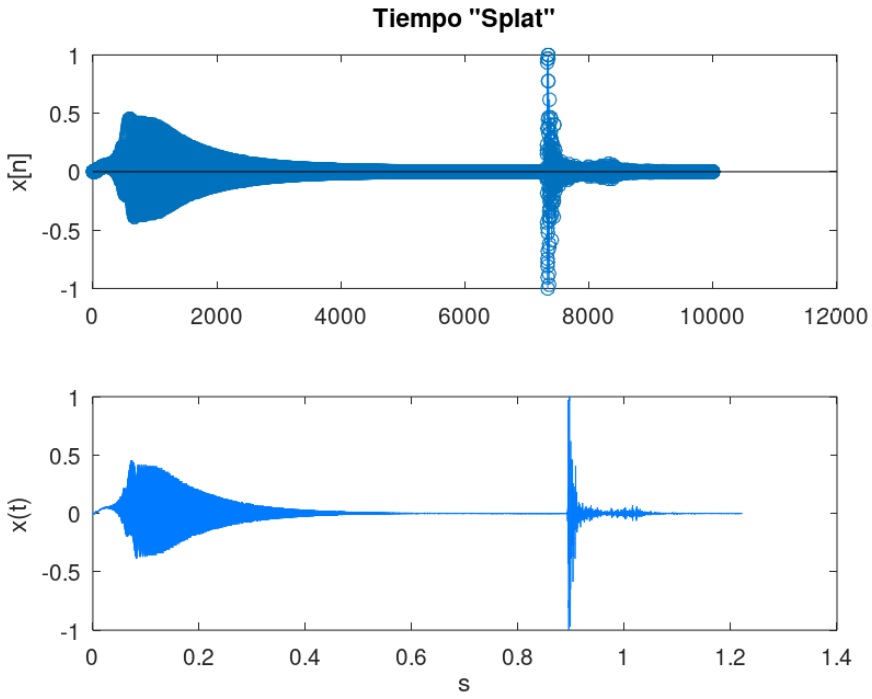
- Inicialice **ft = inline('fftshift(fftn(x))');**
- Las secuencias de audio y su frecuencia de muestreo son accesibles desde Matlab- Para ello tiene que cargar los ficheros indicados más adelante mediante la instrucción **load**.
- Realice la representación gráfica de la secuencia empleando el eje de tiempo discreto. Etiquete adecuadamente cada uno de los ejes indicando claramente su contenido y en caso de que sea necesario las unidades.
- Empleando las instrucciones **subplot** represente la transformada de Fourier de la señal en dominio continuo y la transformada de Fourier de la secuencia. De nuevo incluya el etiquetado de los ejes.
- Reproduzca empleando la instrucción **sound** a la frecuencia de muestreo indicada la secuencia de audio.

Se muestran las secuencias a continuación que se han de estudiar empleando el código programado. Se estudiará la representación en tiempo y frecuencia. De manera cualitativa establezca relaciones entre las frecuencias que escucha y las que se pueden observar en la transformada de Fourier representada:

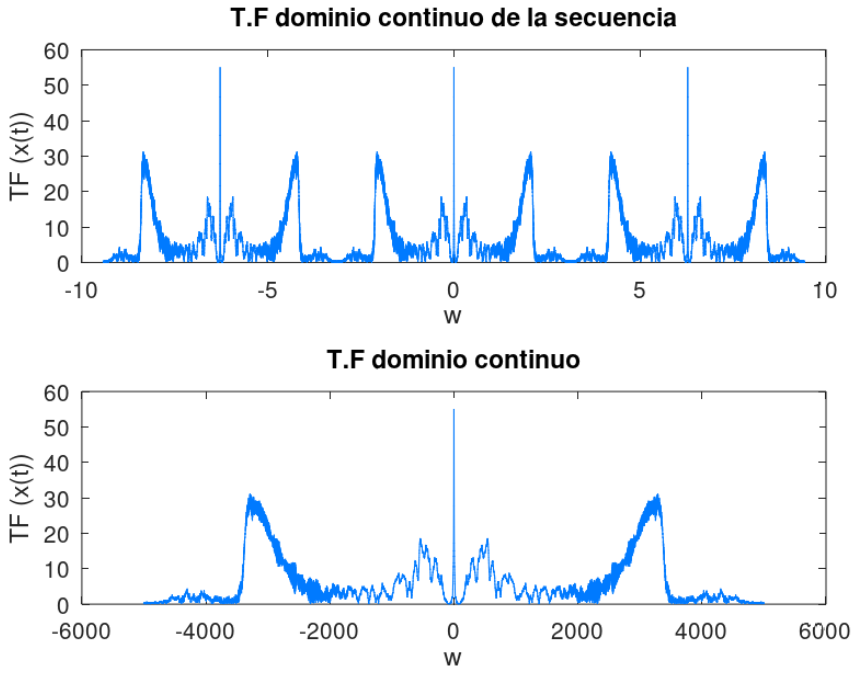
- **Secuencia 'splat' muestreada a 8 KHz:**

```
load splat.mat
```

```
load splat.mat
sound(y, Fs);
fs = Fs;
x = y.';
X = ft(x);
n = (1:length(x))/fs;
figure(1);subplot(2,1,1);
stem(x, "color", "#007AFF"); title('Tiempo "Splat"'); ylabel('x[n]');
subplot(2,1,2);
plot(n,x, "color", "#007AFF"); xlabel('s'); ylabel('x(t)');
```




```
omega = linspace(-3*pi, 3*pi, 3*length(X));
figure(2); subplot(2,1,1);
plot(omega, [abs(X) abs(X) abs(X)], "color", "#007AFF"); xlabel('w'); ylabel('TF (x(t))');
     title('T.F dominio continuo de la secuencia');
w = linspace(-length(X)/2, length(X)/2, length(X));
subplot(2, 1,2); plot(w, abs(X), "color", "#007AFF"); xlabel('w'); ylabel('TF (x(t))');
    title('T.F dominio continuo');
```



La interpretación que hacemos de **Splat** es la siguiente:

1) Vemos como inicia el sonido con la caída del, por lo tanto vemos el pico de la frecuencia y cómo comienza a disminuir hasta llegar cercano a cero.

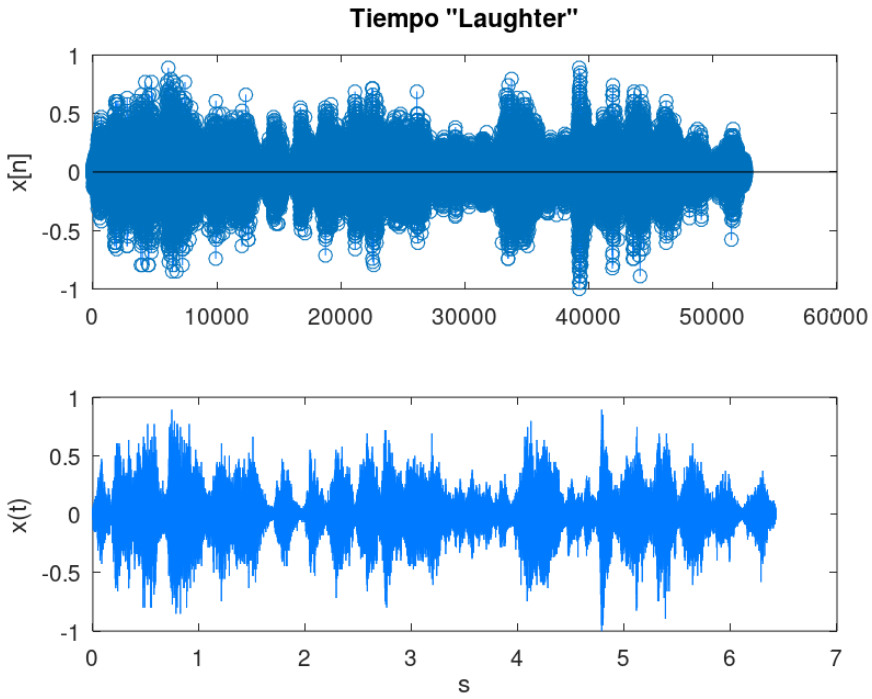
2) Pasa un cierto periodo de tiempo en el que la frecuencia es cercana a cero hasta que se escucha el impacto del objeto por lo que la frecuencia sube rápidamente hasta 40 y como el sonido del impacto es fuerte pero rápido, una

vez sube decrece muy rápidamente.

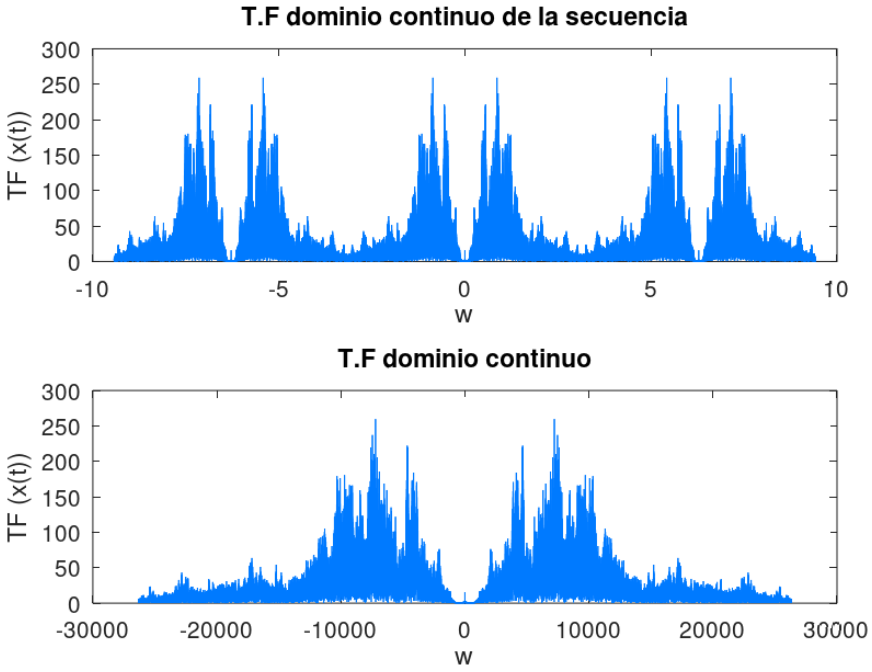
- **Secuencia 'laughter' muestreada a 8 KHz:**

```
load laughter.mat
```

```
load laughter.mat
sound(y, Fs);
fs = Fs;
x = y.';
X = ft(x);
n = (1:length(x))/fs;
figure(3); subplot(2,1,1);
stem(x, "color", "#007AFF");title('Tiempo "Laughter"'); ylabel('x[n]');
subplot(2,1,2); plot(n,x,"color", "#007AFF"); xlabel('s'); ylabel('x(t)');
```




```
omega = linspace(-3*pi, 3*pi, 3*length(X));
figure(4); title('T.F dominio continuo'); subplot(2, 1,1);
plot(omega, [abs(X) abs(X) abs(X)],"color", "#007AFF"); xlabel('w'); ylabel('TF (x(t))');
    title('T.F dominio continuo de la secuencia');
w = linspace(-length(X)/2, length(X)/2, length(X));
 subplot(2, 1,2); plot(w, abs(X), "color", "#007AFF"); xlabel('w'); ylabel('TF (x(t))');
     title('T.F dominio continuo');
```



La interpretación que hacemos de la frecuencia de **Laughter** sería la siguiente:

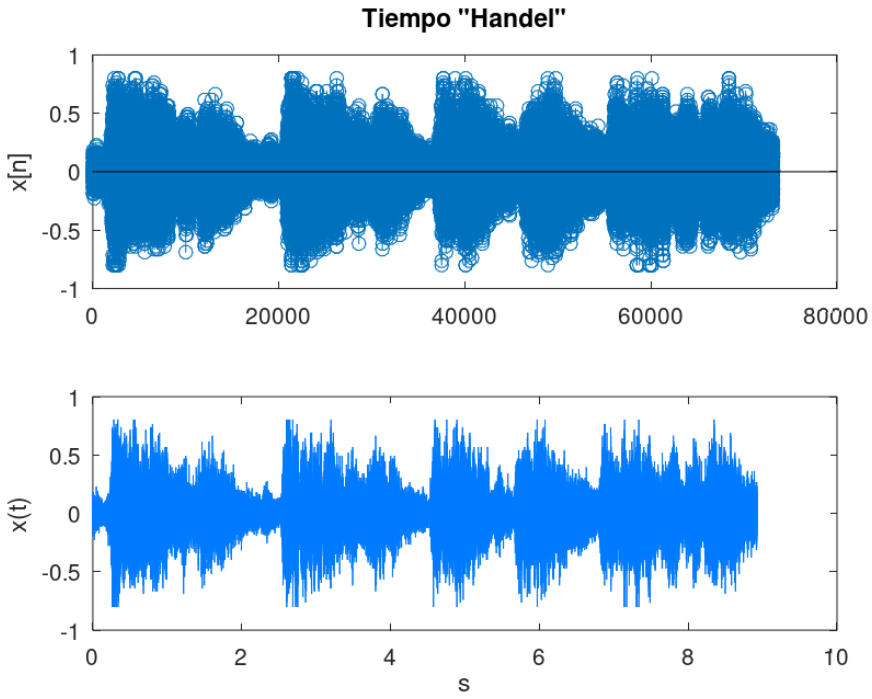
1) Podemos observar cómo crece rápidamente la frecuencia con el inicio de las carcajadas de la multitud y se producen tres variaciones de bajada y subida que coinciden con el espacio de sonido vacío entre unas carcajadas y otras.

2) Una vez comienza a disminuirse la cantidad de carcajadas podemos observar cómo la frecuencia disminuye progresivamente, pero no disminuye de forma drástica desde un pico de frecuencia hasta cero tal como ocurría con la frecuencia anterior.

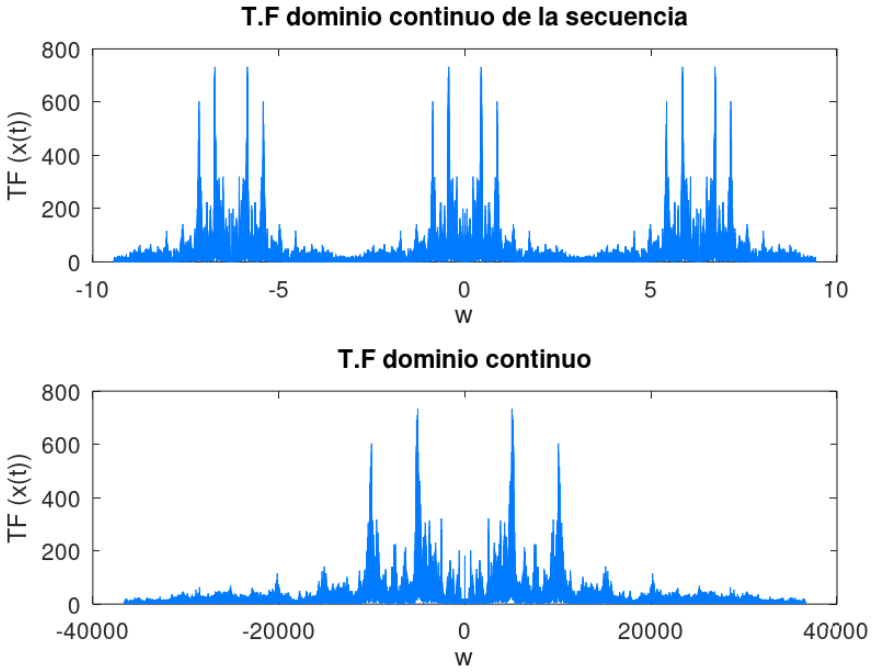
- **Secuencia 'handel' muestreada a 8 KHz:**

```
load handel.mat
```

```
load handel.mat
sound(y, Fs);
fs = Fs;
x = y.';
X = ft(x);
n = (1:length(x))/fs;
figure(5);subplot(2,1,1);
stem(x);title('Tiempo "Handel"'); ylabel('x[n]');
subplot(2,1,2);
plot(n,x,"color", "#007AFF"); xlabel('s'); ylabel('x(t)');
```




```
omega = linspace(-3*pi, 3*pi, 3*length(X));
figure(6); title('T.F dominio continuo'); subplot(2, 1,1);
plot(omega, [abs(X) abs(X) abs(X)], "color", "#007AFF"); xlabel('w'); ylabel('TF (x(t))');
    title('T.F dominio continuo de la secuencia');
w = linspace(-length(X)/2, length(X)/2, length(X));
 subplot(2, 1,2); plot(w, abs(X), "color", "#007AFF"); xlabel('w'); ylabel('TF (x(t))');
     title('T.F dominio continuo');
```



La frecuencia corresponde al final de la ópera **El Mesías** de Haendel, *Aleluya*. En la interpretación se escuchan cinco *Aleluya*s donde en cuatro de ellos se produce un pico debido a la entonación del principio de estrofa (***"A"***), podemos observar una frecuencia similar que se va repitiendo a excepción del quinto donde la entonación cambio totalmente para marcar el final de frase, por lo que en la última estrofa se observa la reducción de frecuencia con respecto a las otras.

2. Comparativa del coste computacional DFT y FFT

Cuestiones

Implemente dos funciones en Matlab:

- La primera debe emplear la implementación directa con Matlab y su entrada será la secuencia a transformar. La función proporcionará la transformada. Llame a la función **mdft.m**
- La segunda debe implementar la versión eficiente basada en el algoritmo de diezmado en tiempo. El nombre de la función será **ffdt.m**

Redacte un script en Matlab que obtenga una secuencia real x de longitud N con muestras aleatorias empleando el comando **rand**.

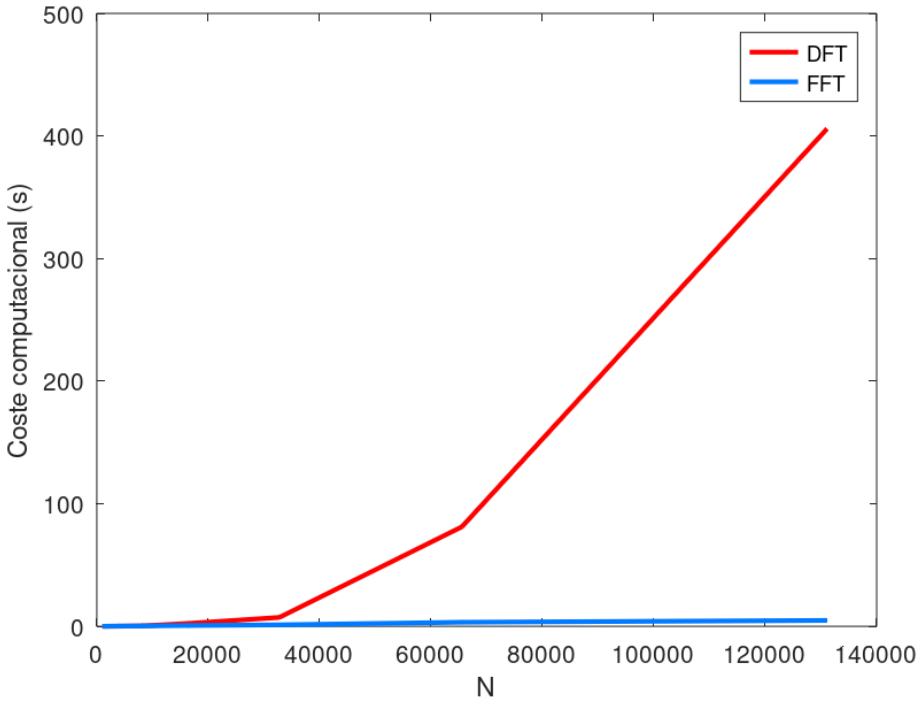
Empleando los comandos **tic** y **toc** establezca el tiempo empleado en calcular su DFT utilizando la función de Matlab **mdft.m** y mediante la versión eficaz del mismo denominada **ffdt.m**

```
N = 64;

x = rand(1, N);

% Calcular DFT usando mdft.m
tic
X_mdft = mdft(x)
toc

% Calcular DFT usando fftdt.m
tic
X_mdft = fftdt(x)
toc
```

Empleando lo siguientes valores para $N = 2^{10}, 2^{11}, 2^{12}, 2^{13}, 2^{14}, 2^{15}, 2^{16}$ y 2^{17} represente en una misma gráfica los tiempos de cálculo empleados por las dos implementaciones de la DFT respecto al valor de N. Etiquete correctamente ambos ejes e incluya la correspondiente leyenda para diferenciar ambas curvas.

```
N_values = 2.^(10:17);

array_mdft = zeros(1, length(N_values));
array_fftdt = zeros(1, length(N_values));

% Calcular DFT y coste computacional
for i = 1:length(N_values)
    x = rand(1, N_values(i));

    tic
    X_mdft = mdft(x);
    array_mdft(i) = toc;

    tic
    X_fftdt = fftdt(x);
    array_fftdt(i) = toc;
end
```

```
figure(7); plot(N_values, array_mdft, "color", '#FF0000', "LineWidth", 2, N_values, array_fftdt
    , "color", '#007AFF', "LineWidth", 2);
xlabel('N'); ylabel('Coste computacional (s)'); legend('DFT', 'FFT')
```



Podemos observar como la Transformada Rápida de Fourier (FFT) es mucho más eficeiente a nivel de coste computacional que la Transformada Discreta de Fourier (DFT).

3. Comparativa de convolución lineal vs convolución circular

Cuestiones

Implemente el código necesario para generar y representar la convolución lineal de:

- Una secuencia de números aleatorios de 64 muestras definida en $n = 0..63$.
- Un filtro FIR casual que realice el promedio deslizante de 8 muestras.

Utilice la función **`convcirc(x1,x2,N)`** para realizar la convolución circular de N puntos de secuencia y la respuesta al impulso del filtro. Considere el mínimo N posible y represente gráficamente el resultado, etiquetando correctamente el eje temporal, $n = 0..N-1$.

Defina la convolución circular de la siguiente manera

```
convcirc = inline('real(ifft(fft(x1, N) .* fft(x2, N), N))', 'x1', 'x2', 'N');
```

- Especifique si proporcionan el mismo resultado la convolución lineal y la convolución circular. Indique las muestras que difieren.
- Incremente el tamaño N de la convolución circular. Indique el valor de N para el que se obtiene un mismo resultado para ambas convoluciones.

```
n = 0:63;
x1 = rand(1, 64);

% Filtro FIR
x2 = ones(1, 8) / 8

% Definir la función de convolución circular
convcirc = inline('real(ifft(fft(x1, N) .* fft(x2, N), N))', 'x1', 'x2', 'N');

N = 64;
y_circ = convcirc(x1, x2, N);
y_lineal = conv(x1, x2);

isequal(round(y_circ), round(y_lineal(1:N)))

diff = y_lineal(1:N) - y_circ;
diff

% Aquí podemos observar que los resultados no son iguales, por lo que vamos a
% aumentar el valor de N usando la condición
```

```
diff =

 Columns 1 through 21:

  -0.4127  -0.3199  -0.2345  -0.1467  -0.1291  -0.0339  -0.0300  -0.0000  -0.0000  -0.0000  -0.0000
   0.0000   0.0000         0  -0.0000  -0.0000   0.0000   0.0000   0.0000         0   0.0000

 Columns 22 through 42:

   0.0000         0   0.0000         0   0.0000  -0.0000   0.0000   0.0000  -0.0000  -0.0000  -0.0000
  -0.0000         0         0  -0.0000  -0.0000         0         0  -0.0000         0         0

 Columns 43 through 64:

  -0.0000   0.0000         0         0  -0.0000  -0.0000   0.0000         0   0.0000  -0.0000  -0.0000
   0.0000   0.0000  -0.0000  -0.0000         0  -0.0000         0  -0.0000  -0.0000  -0.0000         0
```

Para que la convolcuión circular y la convolución lineal coincidan se tiene que cumplir la siguiente condición:

- Que la duración de la señal de salida sea mayor o igual a la suma de las duraciones de las señales menos 1 (**length(x1) + length(x2) - 1**).

```
N = length(x1) + length(x2) - 1; % N = 71
y_circ = convcirc(x1, x2, N);
y_lineal = conv(x1, x2);

isequal(round(y_circ), round(y_lineal(1:N)))

% Los resultados que nos devuelven son muy parecidos salvo por ciertos errores de redondeo
```