

ESCUELA POLITÉCNICA SUPERIOR-LINARES
UNIVERSIDAD DE JAÉN

FUNDAMENTOS Y EQUIPOS DE AUDIO

Grado en Ingeniería de Tecnologías de
Telecomunicación

PRACTICA 2: PROCESADO DE AUDIO CON PYTHON

Alumno (s)	
Apellidos, Nombre	Grupo
Juan José Martínez Cámara	A

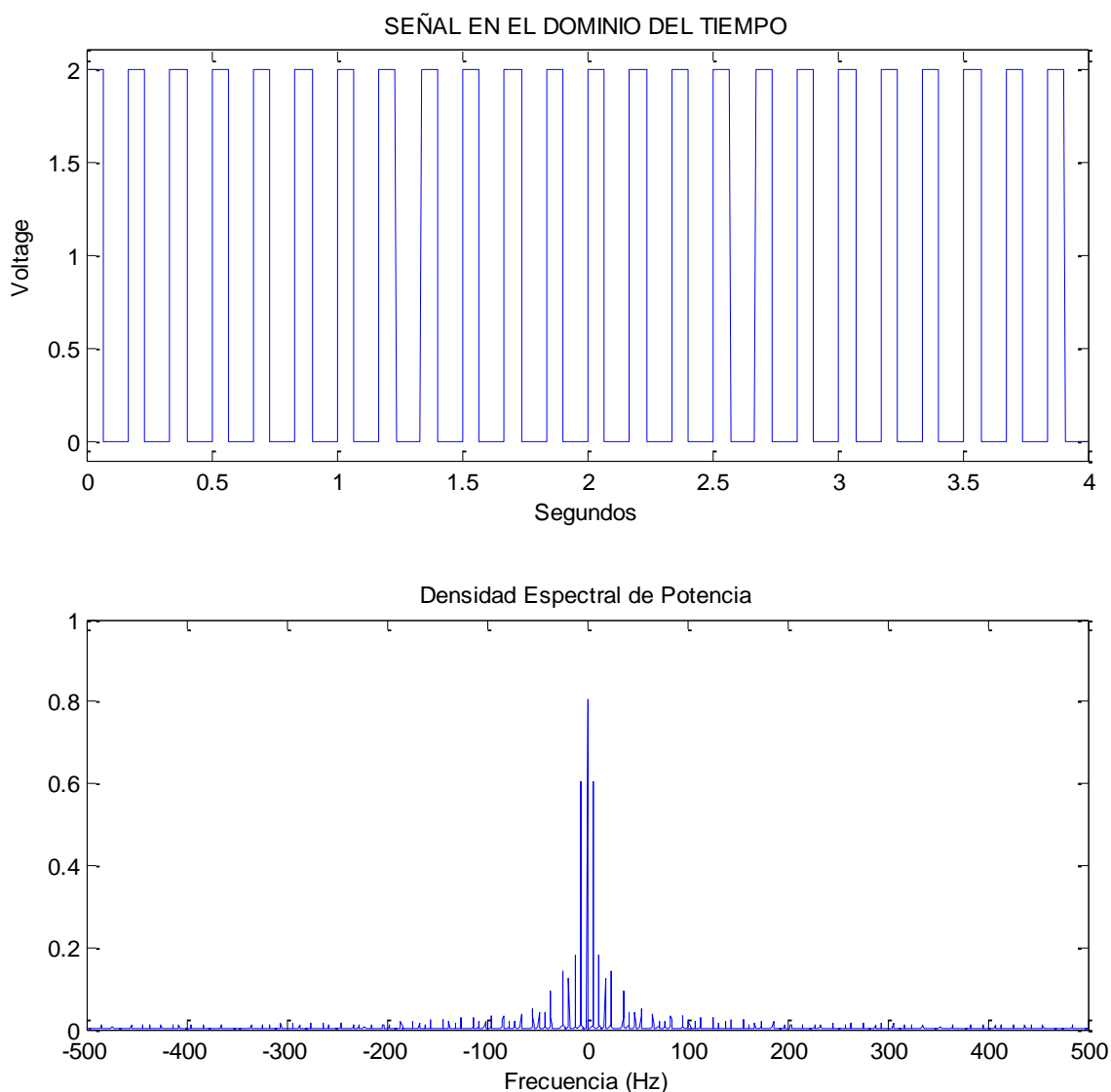
PRÁCTICAS CON PYTHON

Apartado 1. Filtrado de señales periódicas

En esta primera práctica se pretende, de una forma gráfica, comprobar lo señalado por el desarrollo en serie de Fourier. Es decir, cualquier señal periódica se puede construir a partir de exponenciales complejas (senos y cosenos) de frecuencia igual a su frecuencia fundamental (F_0) y sus armónicos.

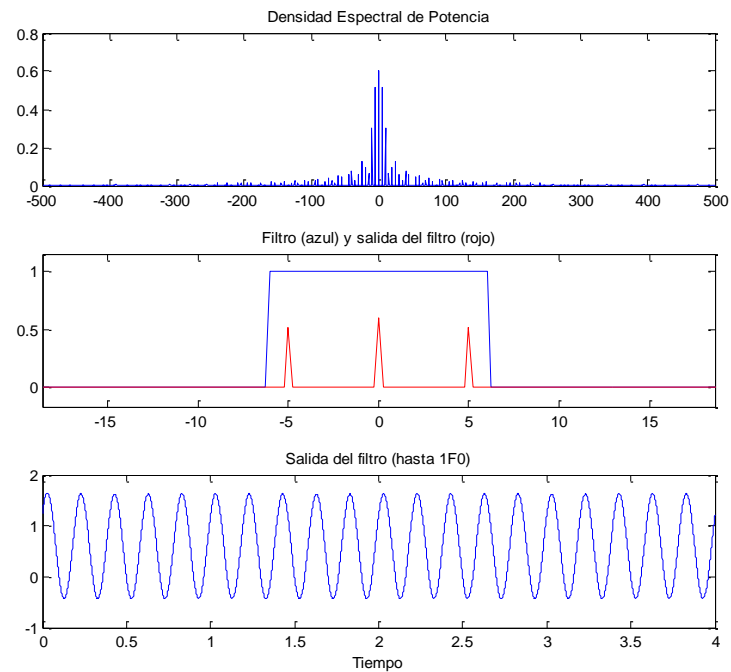
Para ello, utilice la función “fun_cuadrada” dentro del fichero de PYTHON “ex1.py” disponible en docencia virtual. Esta función permite la generación de una señal periódica cuadrada así como su posterior filtrado al objeto de ver el efecto de éste sobre la señal cuadrada original. Cuando se ejecuta, solicita los parámetros necesarios para su funcionamiento.

A continuación se muestra el ejemplo de una señal cuadrada de frecuencia 5 Hz, amplitud 2 y ciclo de trabajo 0.4, filtrada con un filtro paso-bajo de frecuencia de corte 7 Hz (componentes del desarrollo en serie de frecuencias 0 y F_0) y 26 Hz (componentes del desarrollo en serie de frecuencias 0, F_0 , $2F_0$, $3F_0$, $4F_0$ y $5F_0$).



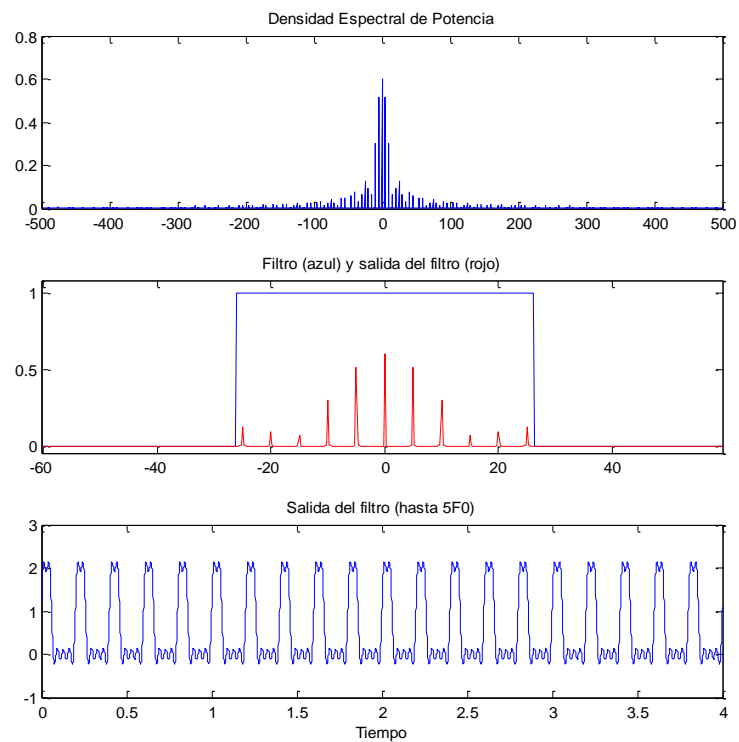
Señal cuadrada de partida: frecuencia 5 Hz, amplitud 2 y ciclo de trabajo 0.4

Como se puede apreciar, cuando la frecuencia de corte del filtro paso-bajo, F_c , vale $F_0 < F_c < 2F_0$, en ese caso la salida es una senoide de frecuencia igual a la de la señal cuadrada ($F_0 = 5$ Hz) con un nivel de continua (componente del desarrollo en serie de Fourier de frecuencia cero)



Señal original filtrada con filtro paso-bajo de 7 Khz de frecuencia de corte

En la gráfica siguiente se muestra el caso de usar una frecuencia de corte de 26 Hz.



Señal original filtrada con filtro paso-bajo de 26 Hz de frecuencia de corte.

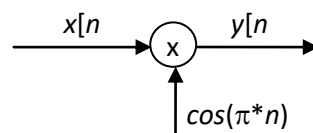
Pruebe a incrementar la frecuencia de corte del filtro LP y comente brevemente en el recuadro que ocurre.

A medida que vamos aumentando nuestra frecuencia de corte, aumentamos nuestra ventana, esto hace que tengamos un número mayor de armónicos haciendo que nuestra señal en el tiempo sea más similar a nuestra señal original en el tiempo, en cambio cuantos menos armónicos cogemos, tendera más a parecer un seno, ya que como se ha visto $F_0 < F_c < 2F_0$, entonces cuanto más nos acerquemos a ese rango de frecuencias, obtendremos una figura más similar a una senoide.

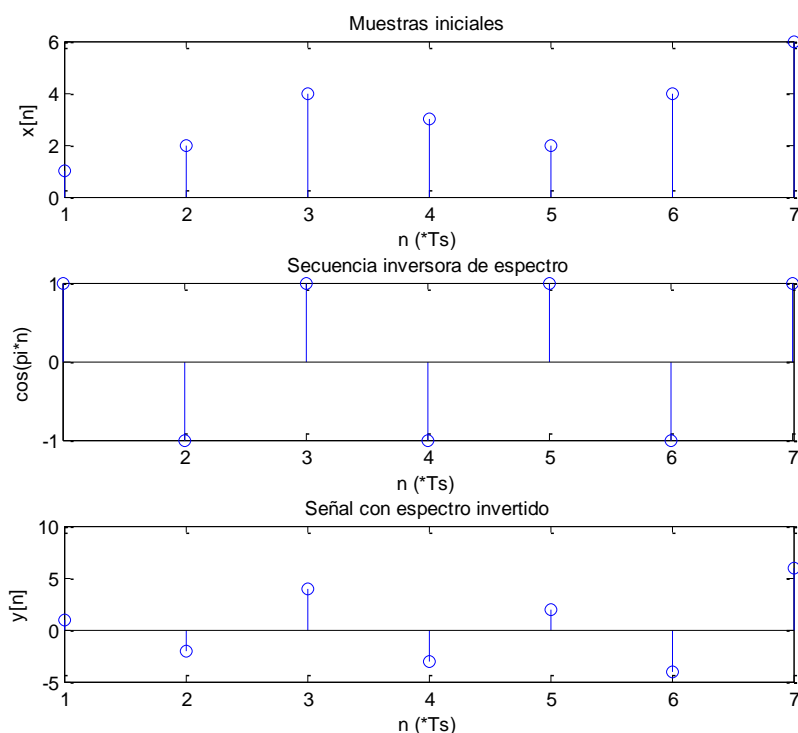
Apartado 2. Inversión del espectro

Este efecto convierte las frecuencias graves en agudas y viceversa. Según el ancho de banda así será la inversión. Este efecto se consigue multiplicando la secuencia de entrada por $\cos(\pi*n)=(-1)^n$. Es decir, por cada 2 muestras, una permanece inalterada y la otra cambia su signo.

El siguiente esquema representa esta operación.



A continuación se representa un ejemplo de aplicación de este efecto a una secuencia de muestras.



Implemente una función de PYTHON, llamada “**inversor**” que responda al siguiente esquema:

```
def inversor(x):
    """ INVERSOR ESPECTRO
    Input:
        - x: Señal de entrada x de tipo numpy array
    Output:
        - y: Señal de salida obtenida como  $y(n) = x(n) * \cos(\pi * n)$ , siendo
            n la muestra actual
    """
```

siendo x el vector de las muestras de entrada e y el vector de las muestras de salida, y que realice la multiplicación de los elementos del vector x por el tren de impulsos de signo alterno.

En el siguiente recuadro escriba el código de la función inversor.

```
def inversor(x):
    y=np.zeros(len(x))
    for n in np.arange(len(x)):
        y[n]=x[n]*((-1)**n)

    return y
```

Igualmente, edite un fichero y llámelo “**denpower**” que implemente una función de PYTHON que calcule y represente la densidad espectral de potencia una señal dada. Esta función debe responder al siguiente esquema:

```
def denpower(x, fs):
    """ DENSIDAD ESPECTRAL DE POTENCIA
    Input:
        - x: Señal de entrada de tipo numpy array
        - fs: Frecuencia de muestreo
    Output:
        - y: Señal de salida obtenida como  $y(n) = x(n) * \cos(\pi * n)$ , siendo
            n la muestra actual
    """
```

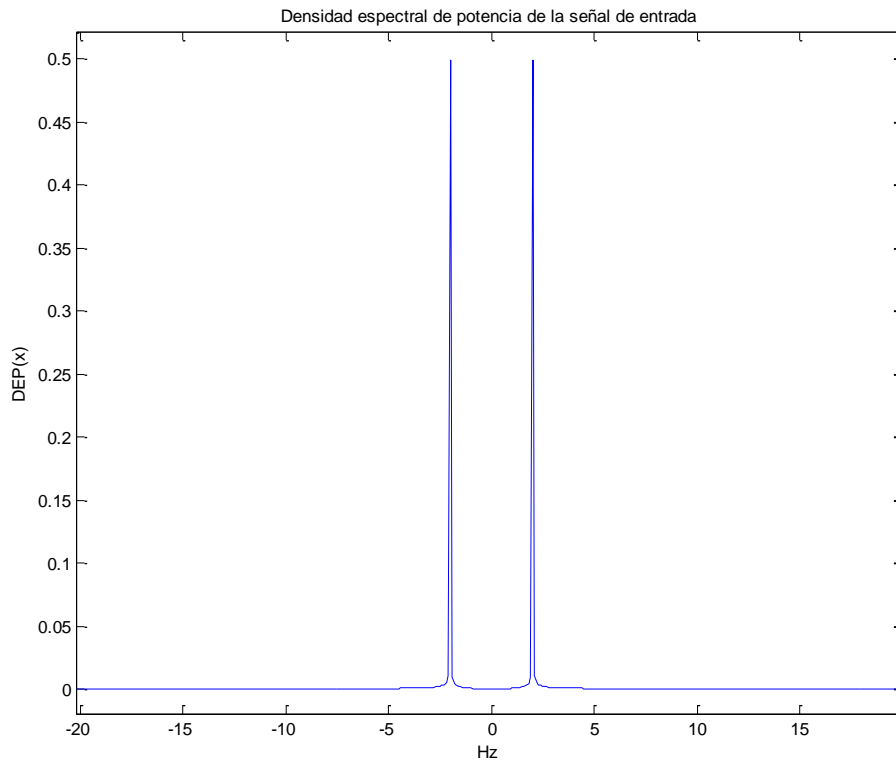
siendo x el vector de muestras de entrada y fs la frecuencia de muestreo con que han sido obtenidas éstas.

Escriba en el siguiente recuadro el código de la función denpower.

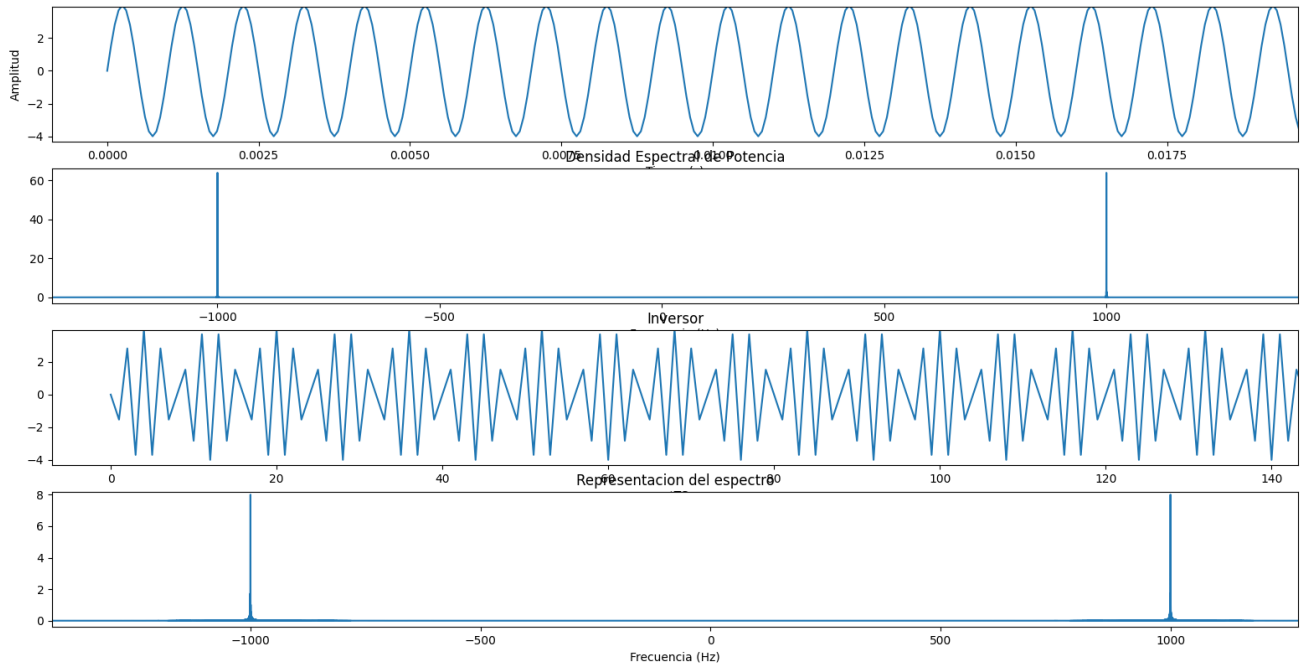
```
def denpower(x, fs):

    DEP=abs(fft(x,10*len(x))/fs)**2;
    f=np.arange(-fs/2,fs/2,fs/len(DEP))
    DEP_FIG =plt.figure()
    dep_fig = DEP_FIG.add_subplot(111)
    dep_fig.plot(f,fftshift(DEP))
    dep_fig.set_xlabel("freq Hz")
    dep_fig.set_ylabel("Voltios V")
    dep_fig.set_title("Densidad espectral de potencia")
    return DEP
```

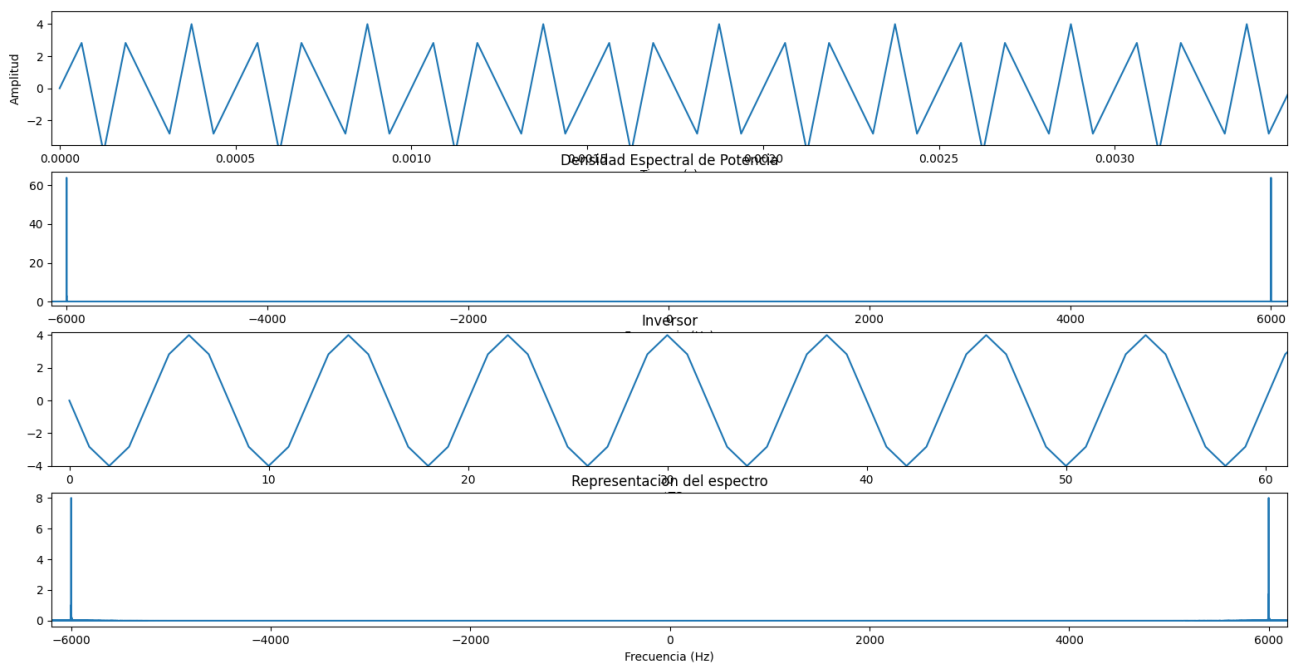
Además, se debe representar gráficamente la distribución en frecuencia de la potencia de la señal. El eje de abscisas debe representar los hertzios de la componente en frecuencia correspondiente. Por ejemplo, en la siguiente figura se representa la densidad espectral de potencia de una senoide de frecuencia 2 hertzios.



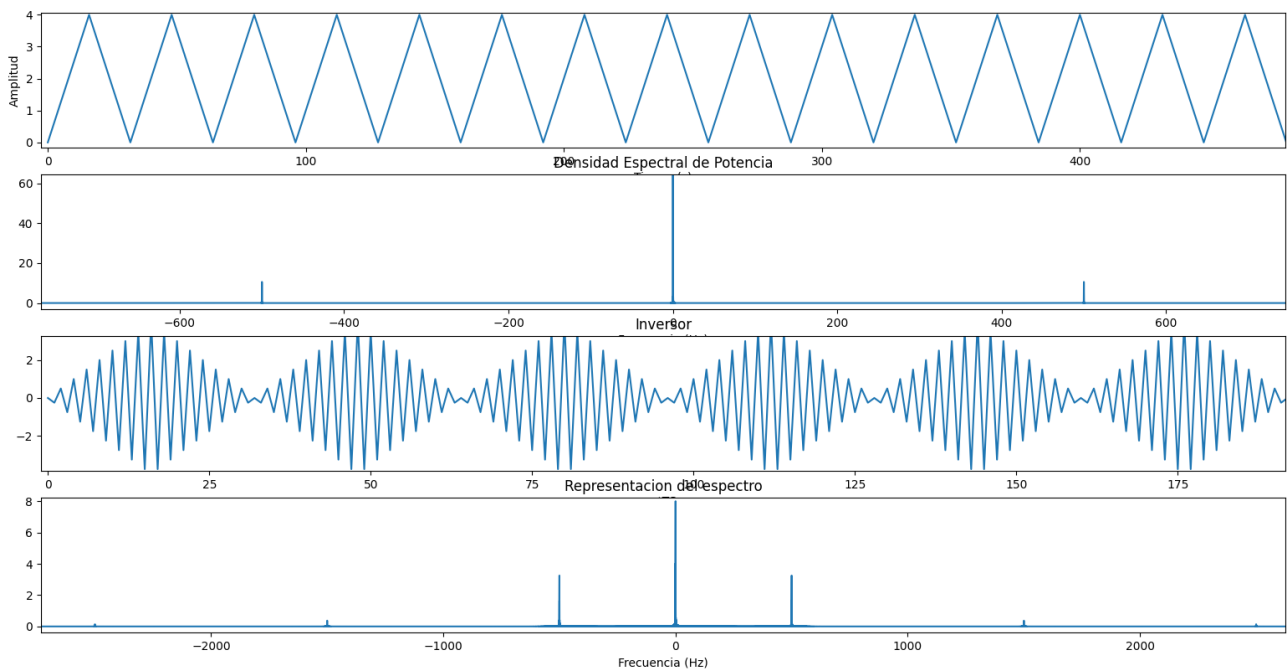
Genere las muestras correspondientes a un tono de 1 KHz muestreado a 16 KHz, fase cero en el origen, amplitud 4, observado en el intervalo [0,4] segundos. A continuación represente, usando la función “SUBPLOT” de PYTHON (ver implementación en el apartado 1 de esta práctica), dichas muestras, su densidad espectral de potencia, las muestras obtenidas a la salida del inversor y el espectro de éstas.



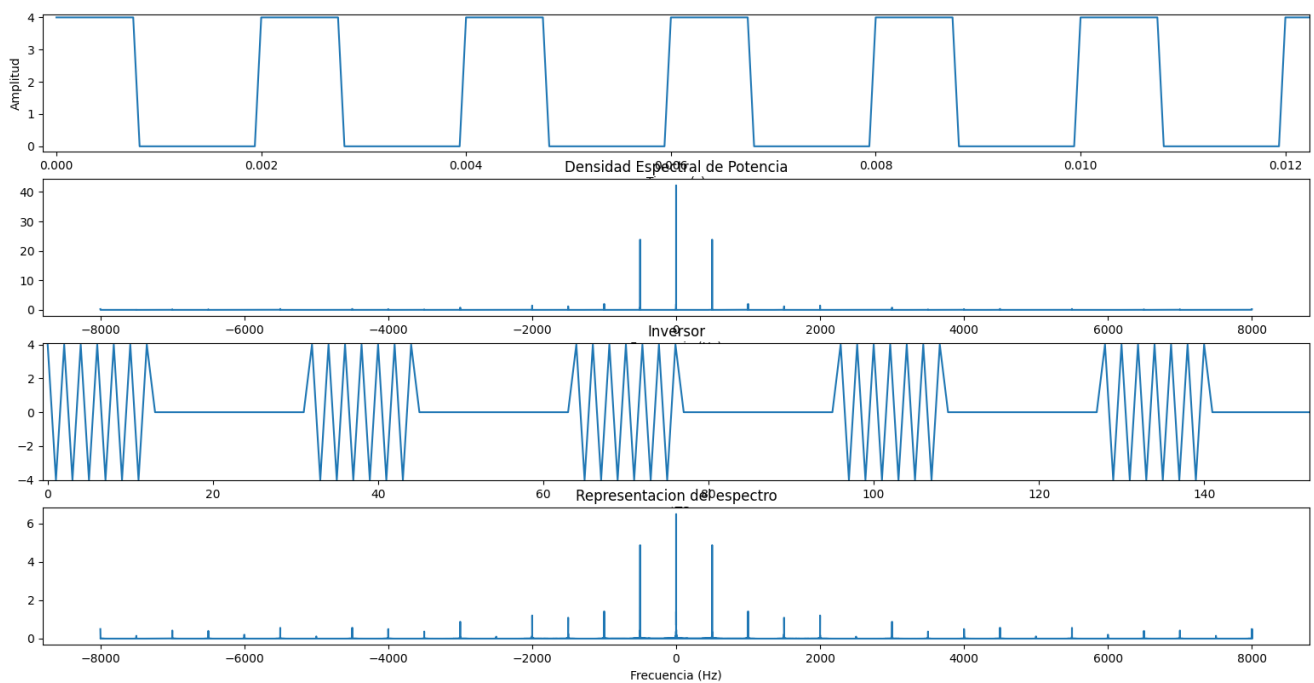
Repita los pasos del caso anterior pero esta vez para una frecuencia de senoide de 6 KHz.



Considere ahora una señal triangular de frecuencia 500 Hz. Represente esta señal en el tiempo, su densidad espectral de potencia, la salida del inversor y el espectro de ésta.



Repita los pasos anteriores pero para una señal cuadrada con ciclo de trabajo 0.4.



Seguidamente, lea un fichero de audio (función “audioread” para leer ficheros en formato WAV), invierta su espectro, y escúchelo (función “audioplay”, por ejemplo).

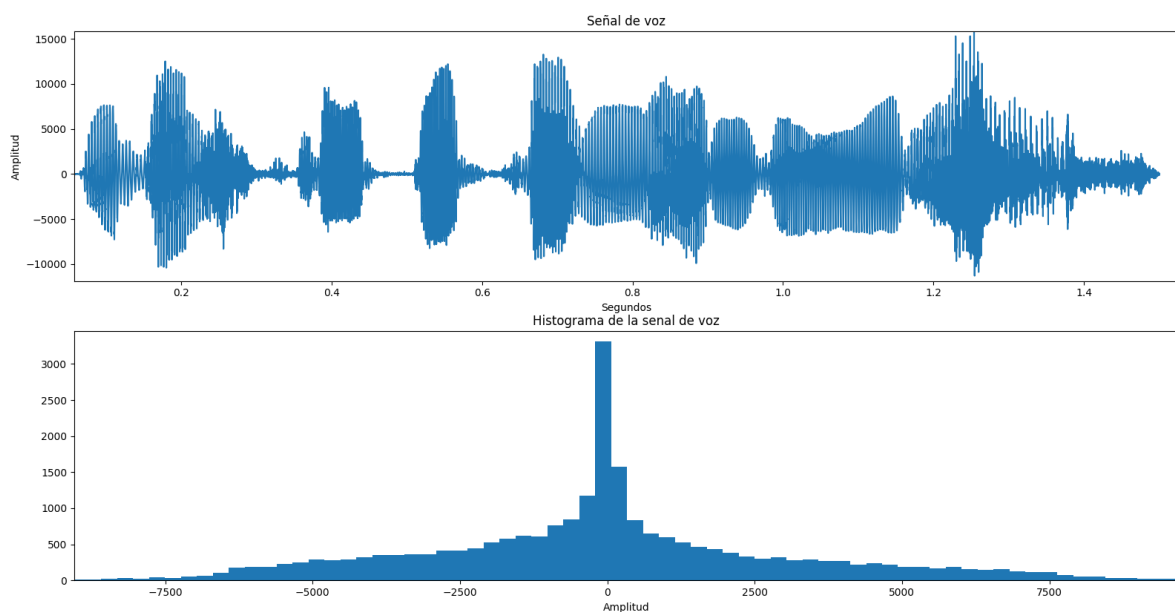
```
fs, x = audioread("ursula.wav")
audioplay(x, fs)
invx = inversor(x)
audioplay(invx, fs)
```

Finalmente, observemos el histograma de la señal de voz, lo que nos ofrecerá una idea de su función densidad de probabilidad.

```
fs, x = audioread("ursula.wav")
audioplay(x, fs)
invx = inversor(x)
audioplay(invx, fs)

t = np.arange(0, float(len(x))/fs, 1.0/fs)
fig = plt.figure()
subplot1 = fig.add_subplot(211)
subplot1.plot(t, x)
subplot1.set_title("Señal de voz"); subplot1.set_xlabel('Segundos')
subplot1.set_ylabel('Amplitud')
subplot2 = fig.add_subplot(212)
subplot2.hist(x, 100)
subplot2.set_title("Histograma de la señal de voz")
subplot2.set_xlabel('Amplitud')
plt.show()
```

Represente las gráficas obtenidas.



Comente el resultado obtenido.

Como resulta ser voz humana el espectro que observamos se distribuye la energía predominantemente en frecuencias bajas-medias

Apartado 3: Grabe su propia voz

Con PYTHON es posible captar la señal de un micrófono y asociarlo a un vector de muestras.

Use la función “audiorecord” y realice varias grabaciones de su voz.

Reproduzca el vector obtenido usando para su reproducción frecuencias de muestreo diferentes a la original con que han sido obtenidas las muestras.

¿Qué ocurre si reproducimos a 32 KHz un fichero de audio grabado a una frecuencia de muestreo a 16Khz?

Estamos realizando un sobremuestreo en el que lo que estamos realizando una compresión en el tiempo y una expansión en frecuencia, esto es llamado interpolación. Si por ejemplo, nuestra señal dura 1 segundo, y la relación es $f_s \cdot t = N^0$ de muestras, pues como se ha comentado, en este caso, lo que hacemos es aumentar el numero de muestras, pero no aumentamos nuestra cantidad de información, Aunque, en el caso de un audio grabado lo que ocurre, es el que el numero de muestras se queda constante, como puede ser el caso en 24000 muestras, entonces, esta disminución se ve afectada en el tiempo/duración de nuestro sonido, como puede es el caso.

EJEMPLO:

Tenemos un audio de 1.5 segundos contiene una frecuencia de muestreo original de 16kHz, con lo que tenemos 24000 muestras, en cambio, si aumentamos la frecuencia de muestreo al doble, 32kHz y conservamos nuestro número de muestras, $t = (N^0 \text{ demuestras} / 32\text{kHz}) = 0.75$ segundos, como vemos, reducimos la duración de nuestro audio a la mitad haciendo que se note a la hora de percibirlo una simulación parecida de “Alvin y las ardillas”, lo que antes se reproducía en 1.5 segundos ahora se reproduce en la mitad de tiempo..

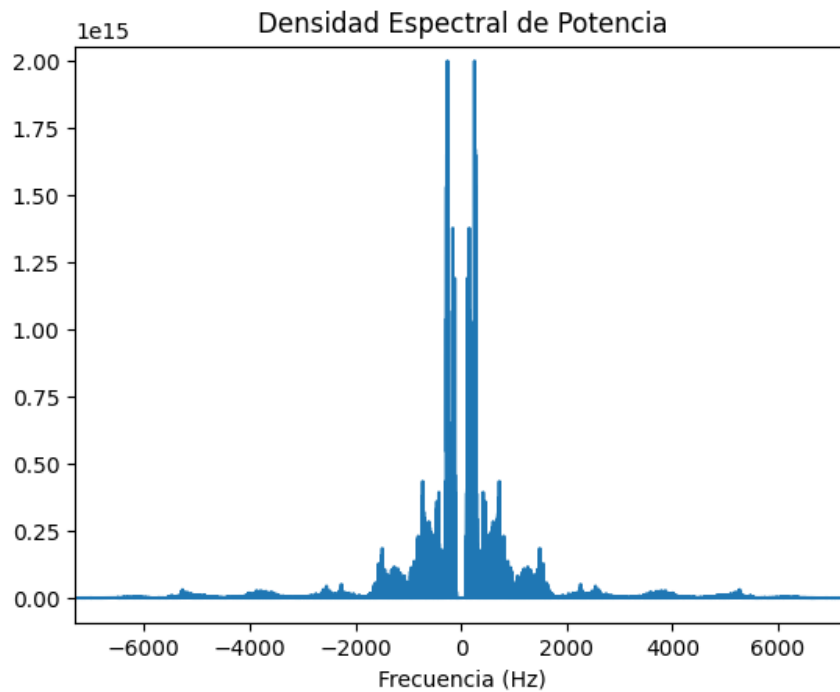
¿Y si ocurre al revés, es decir, grabamos a 32 KHz y reproducimos a 16 KHz?

Se produce el efecto contrario, si antes sufríamos una compresión en el tiempo y ensanchamiento en frecuencia, ahora sufrimos una compresión en frecuencia y un ensanchamiento en el tiempo, con lo cual hará que en el tiempo los cambios entre muestras contiguos sean mas suaves y relajados, se puede ver también con un ejemplo numérico :

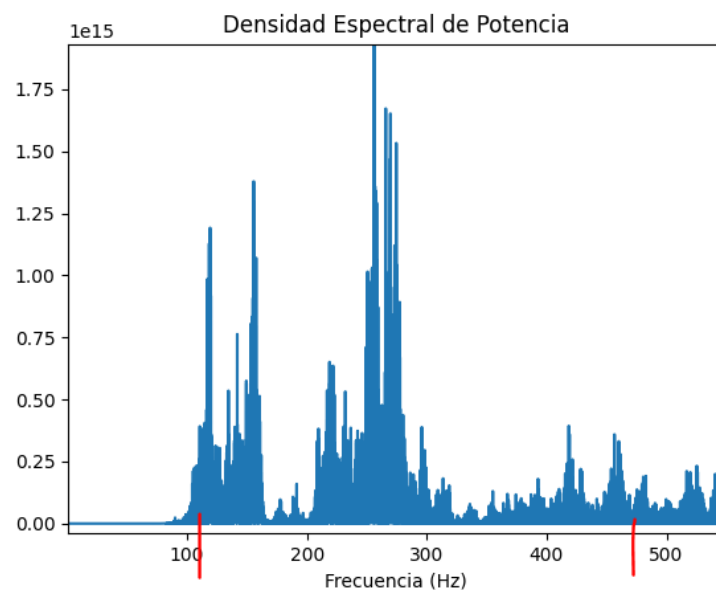
Ejemplo:

Tenemos un audio de 1.5 segundos muestreado a una frecuencia de 32kHz, entonces tenemos 48000 muestras, en cambio, si disminuimos la frecuencia de muestreo a la mitad, como se ha comentado antes ahora tenemos una expansión en el tiempo, haciendo que la duración de la señal sea 3 segundos, ya que $t = (48000 / 16\text{kHz}) = 3$ segundos, el efecto producido en este caso es el conocido como diezmado, que realmente, lo que se hace es reajustar la señal añadiendo muestras promediadas entre la que le corresponda anterior y contigua para que así , la transición sea mas suave con respecto a la señal original.

Usando la función “*denpower*” que ha desarrollado en el apartado 2 de la práctica, anteriormente indicada, represente gráficamente la distribución en frecuencia de la potencia de la señal de voz grabada. Represente la gráfica obtenida.

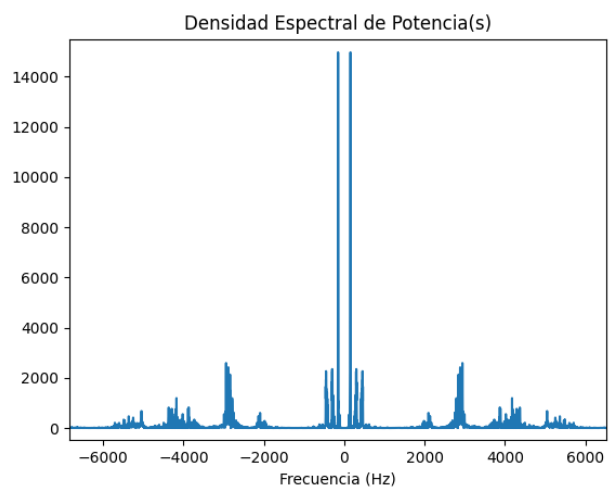
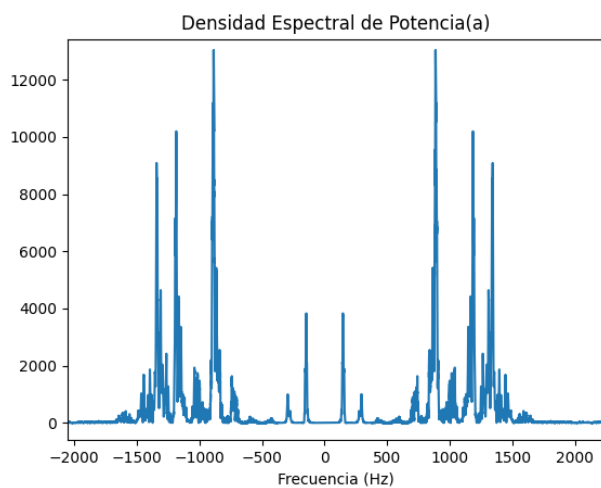
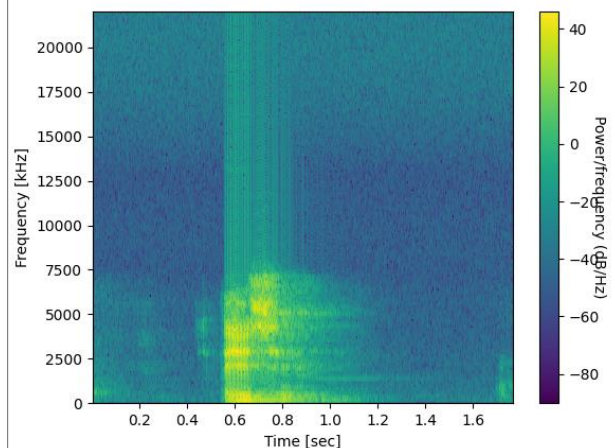
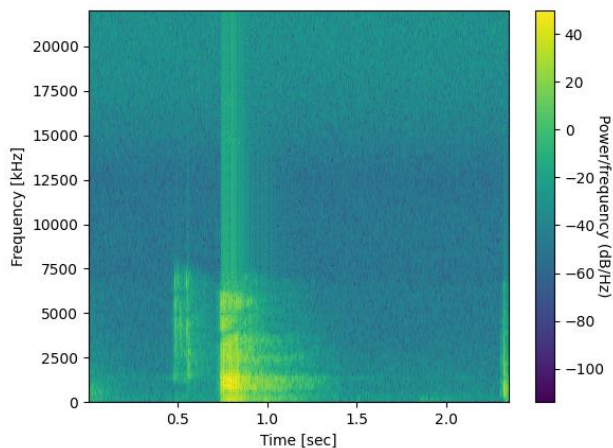


Podemos ver claramente como la voz humana se concentra mayormente entre los 250Hz-3000Hz, es cierto que nos encontramos fonemas por encima del rango de los 3000Hz, como se puede observar, es el caso, pero principalmente tenemos la energía concentrada donde predominan todos los fonemas, también es cierto que la voz masculina, concentra su energía en las frecuencias de 100Hz-500Hz, como se puede ver en el caso, es así.

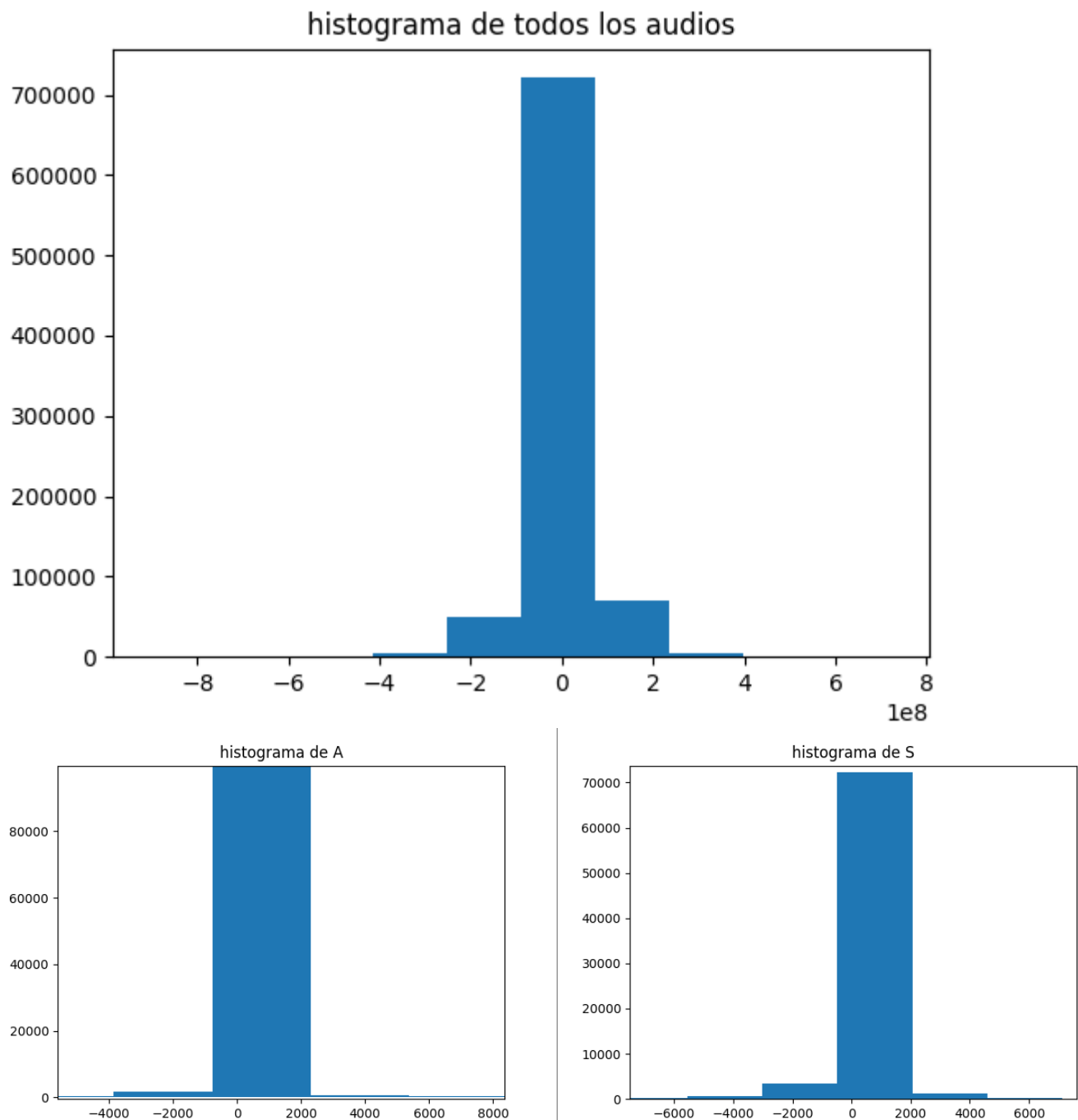


Por último, se va a representar la distribución en frecuencia de algunos fonemas. En concreto, realice pruebas grabando una vocal (por ejemplo la “a”) y un fonema sordo (por ejemplo, la “s”). **Use la función de PYTHON “spectrogram” para conocer cómo se distribuye la energía de la señal en frecuencia a lo largo del tiempo. Aparte, también use la función “denpower”. Compare los resultados obtenidos.**

Como era de esperar y se puede observar, se concentra a altas frecuencias nuestra densidad de potencia para el fonema a y para el fonema s en bajas



Finalmente, y tal como se hizo en la práctica anterior, observemos el **histograma para cada una de las grabaciones de voz realizadas**. Compare los histogramas obtenidos para la vocal y el fonema sordo.

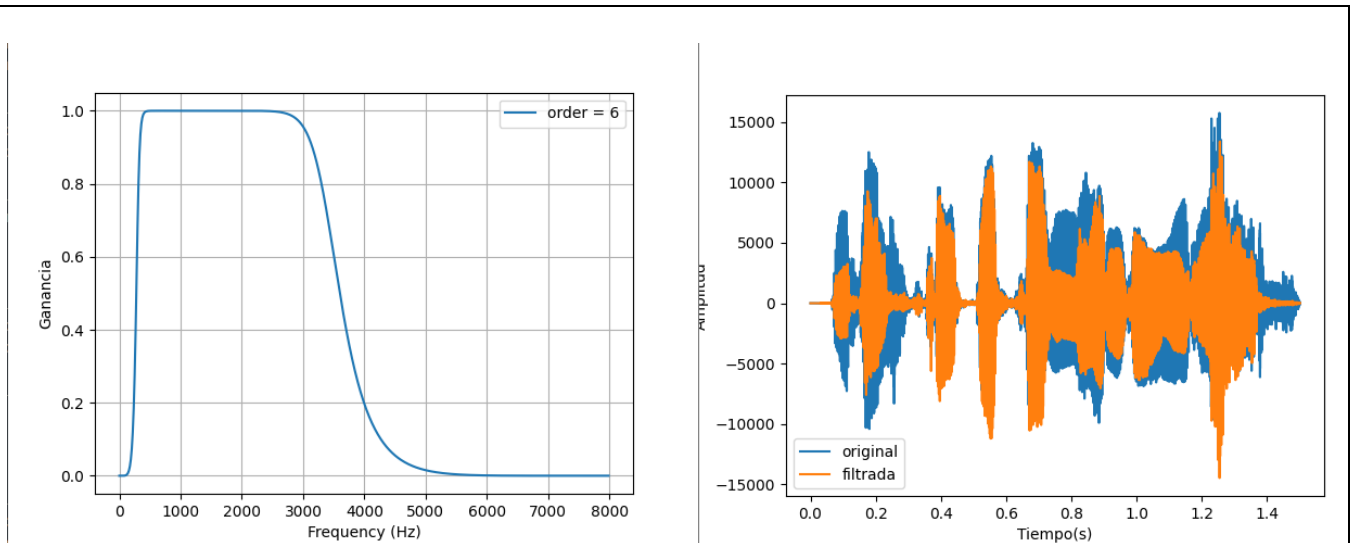


Por último, se va a filtrar la señal de voz utilizando varios filtros paso-banda. Para ello, realice una **nueva grabación de voz** usando una frecuencia de muestreo de 48 KHz. Use la función “`butter_bandpass_filter`” de PYTHON y filtre el vector de muestras de audio obtenido con los tres filtros siguientes:

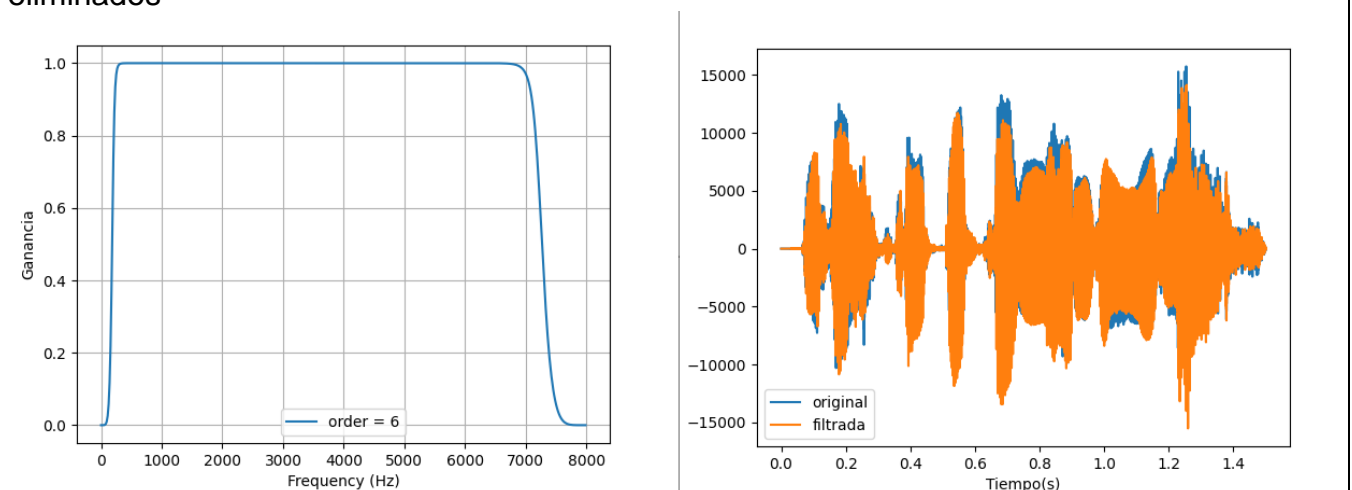
- Filtro paso-banda 300Hz-3400Hz (calidad telefónica).
- Filtro paso-banda 200Hz-7200Hz (calidad AM).
- Filtro paso-banda 100Hz-11000Hz (calidad Reporting, voz de alta calidad).
- Filtro paso-banda 50Hz-15000Hz (calidad FM).

Repita los pasos anteriores pero usando como entrada el fichero “ursula.wav”.

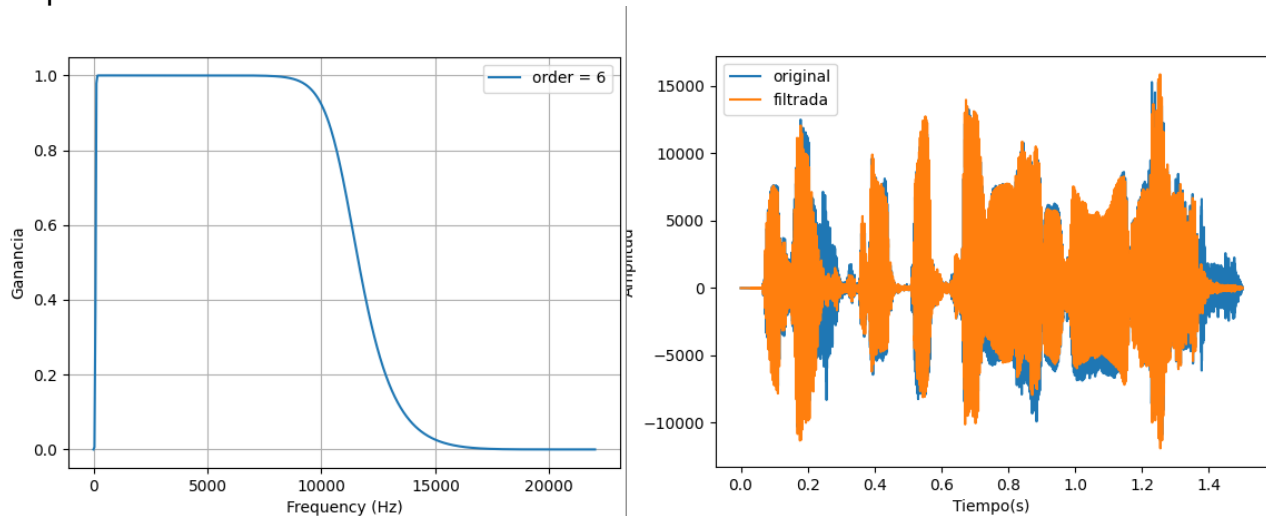
En el siguiente cuadro, comente cómo varía la calidad de la señal conforme modificamos el ancho de banda



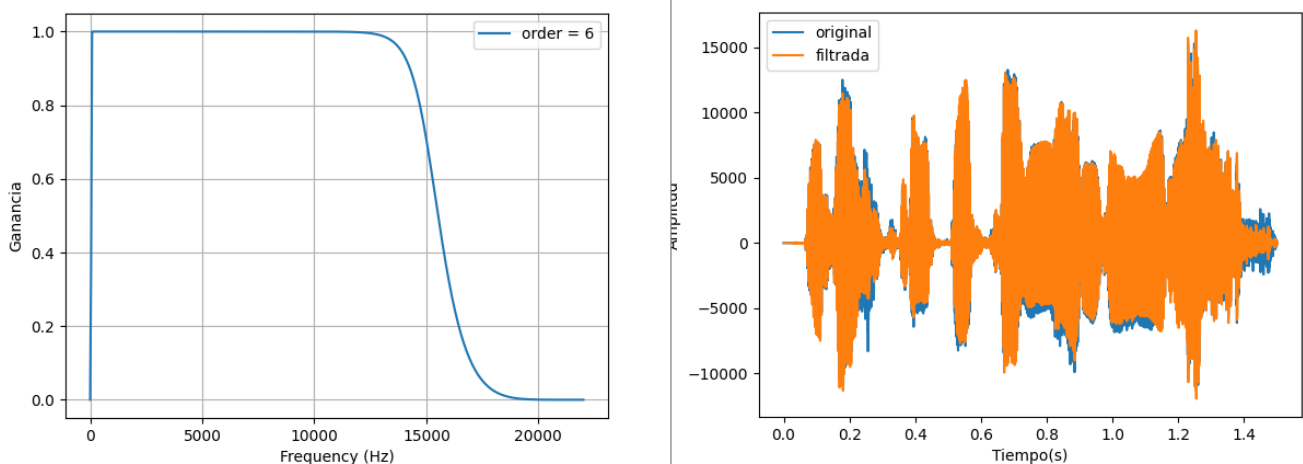
En este caso nuestro ancho de banda es similar al usado en una línea telefónica con lo cual todos los fonemas que se encuentren por debajo de 300 o por encima de 3400Hz, son eliminados



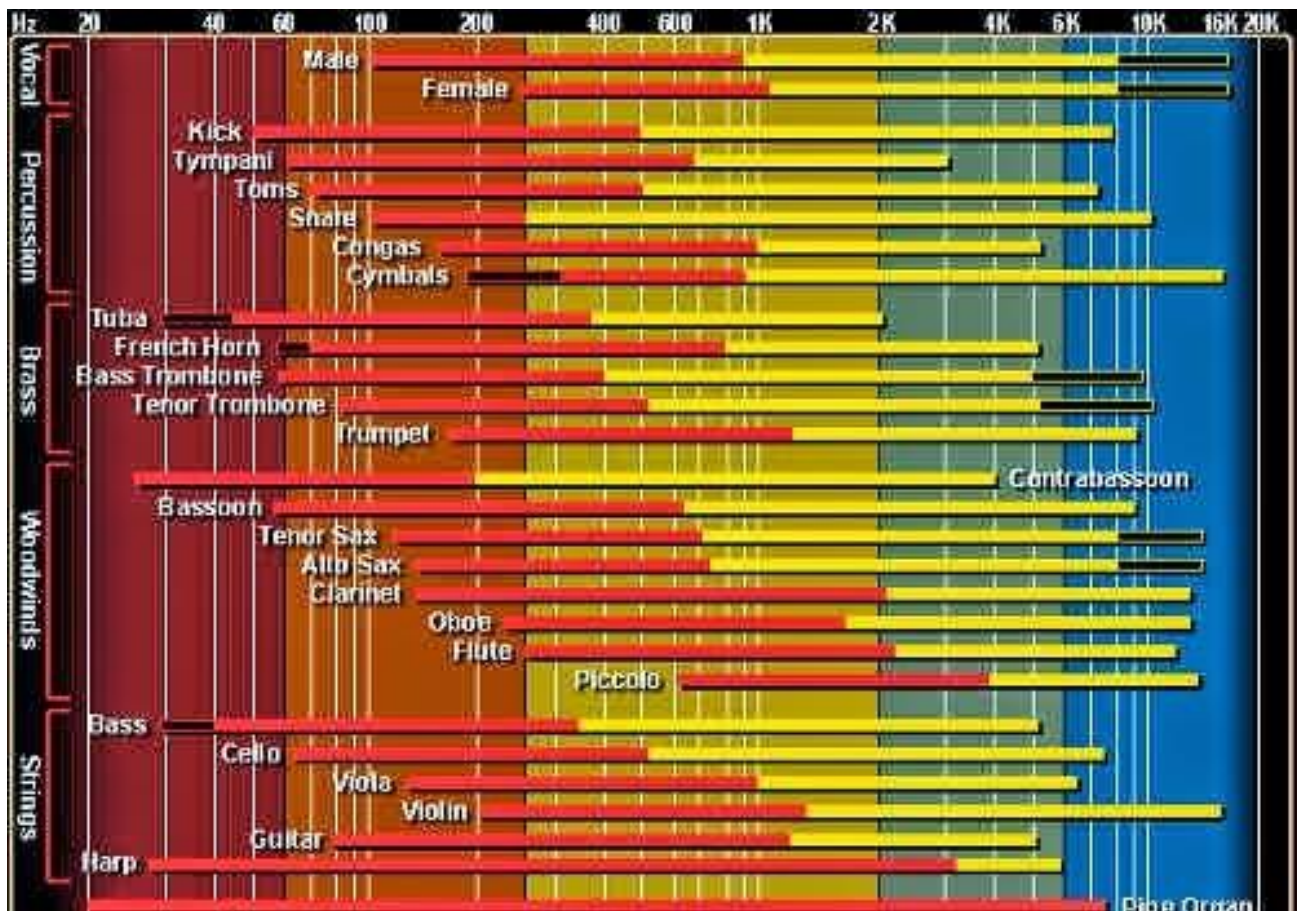
Usamos ahora, un filtro de alta calidad en la que recoge todo el espectro de voz humana al completo 100-11000Hz



Ahora se aumenta aun mas la calidad hasta FM , que se comprende desde los 50-15000Hz



Una vez graficados todas las señales filtradas de manera distinta, se puede apreciar como a medida que se aumenta el ancho de banda del filtro, nuestra similitud a la señal original es mas notoria, asemejándose mas a medida que avanzamos y esto es debido a que estamos recogiendo mas rangos de frecuencias donde ya no se concentra solamente la voz humana, si no la gran mayoría de los instrumentos que contiene el audio original, se entiende mejor con la imagen proporcionada en la siguiente página :

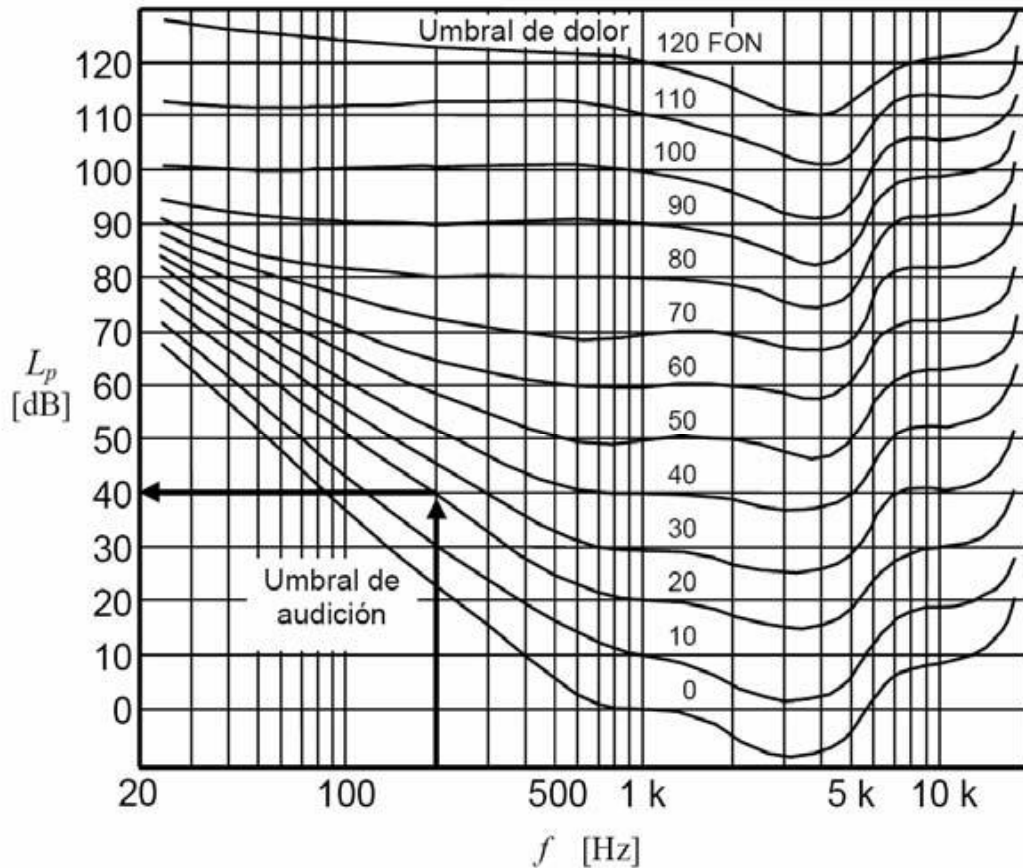


Como se estaba comentando anteriormente, a medida que se aumenta el espectro de nuestro filtro, somos capaces de captar mas instrumentos contenidos en el audio original.

Apartado 4. Evaluación del nivel sonoro intensidad subjetiva

En esta práctica se pretende distinguir entre los conceptos de intensidad y sonoridad (o intensidad subjetiva). La primera se corresponde con el nivel de amplitud (potencia) de la señal que llega al oído, y la segunda se corresponde con la sensación de amplitud-nivel con que se percibe en éste. Como se apreciará a través del desarrollo de esta práctica, el nivel de sonoridad (intensidad subjetiva) con que se percibe una señal en el oído depende de su frecuencia. Como sabemos por teoría, aparte de la frecuencia, también depende del nivel de la señal (comportamiento no lineal del oído),

A continuación se representan los contornos de igual sonoridad (curvas "isofónicas" de igual nivel de sonoridad (su unidad es el *fonio*)) propuestas por Fletcher y Munson.

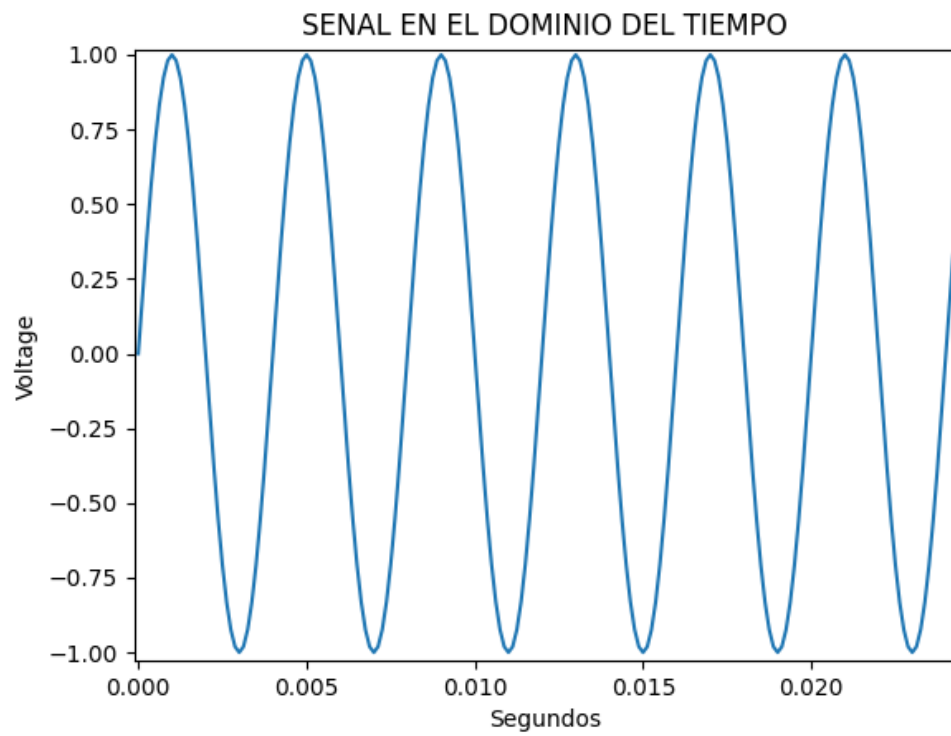


Cree una función en PYTHON que genere un tono. Sus parámetros de entrada serán la amplitud, frecuencia del tono, frecuencia de muestreo y número de muestras. Escriba en el siguiente recuadro el código de dicha función.

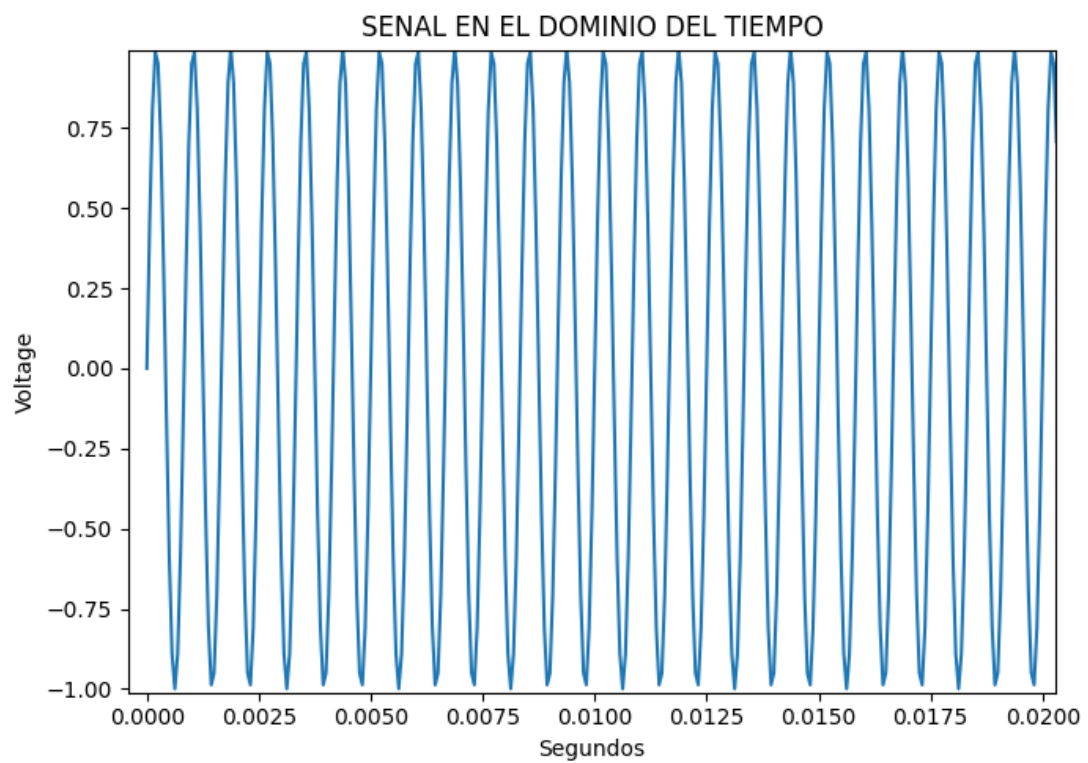
```
def sintono(A, f0, fs, N):
    t=np.arange(0,N/fs,1/fs)
    x=2*math.pi*f0*t
    signal = np.sin(x)*A
    return signal,t
```

Seguidamente, para una frecuencia de muestreo de 44.1 KHz, genere 3 tonos diferentes de amplitud unidad y frecuencias 250 Hz, 1.2 KHz y 11 KHz. Usando la función "plot" de PYTHON, represéntelos en el dominio del tiempo (unidad del eje de abscisas en segundos).

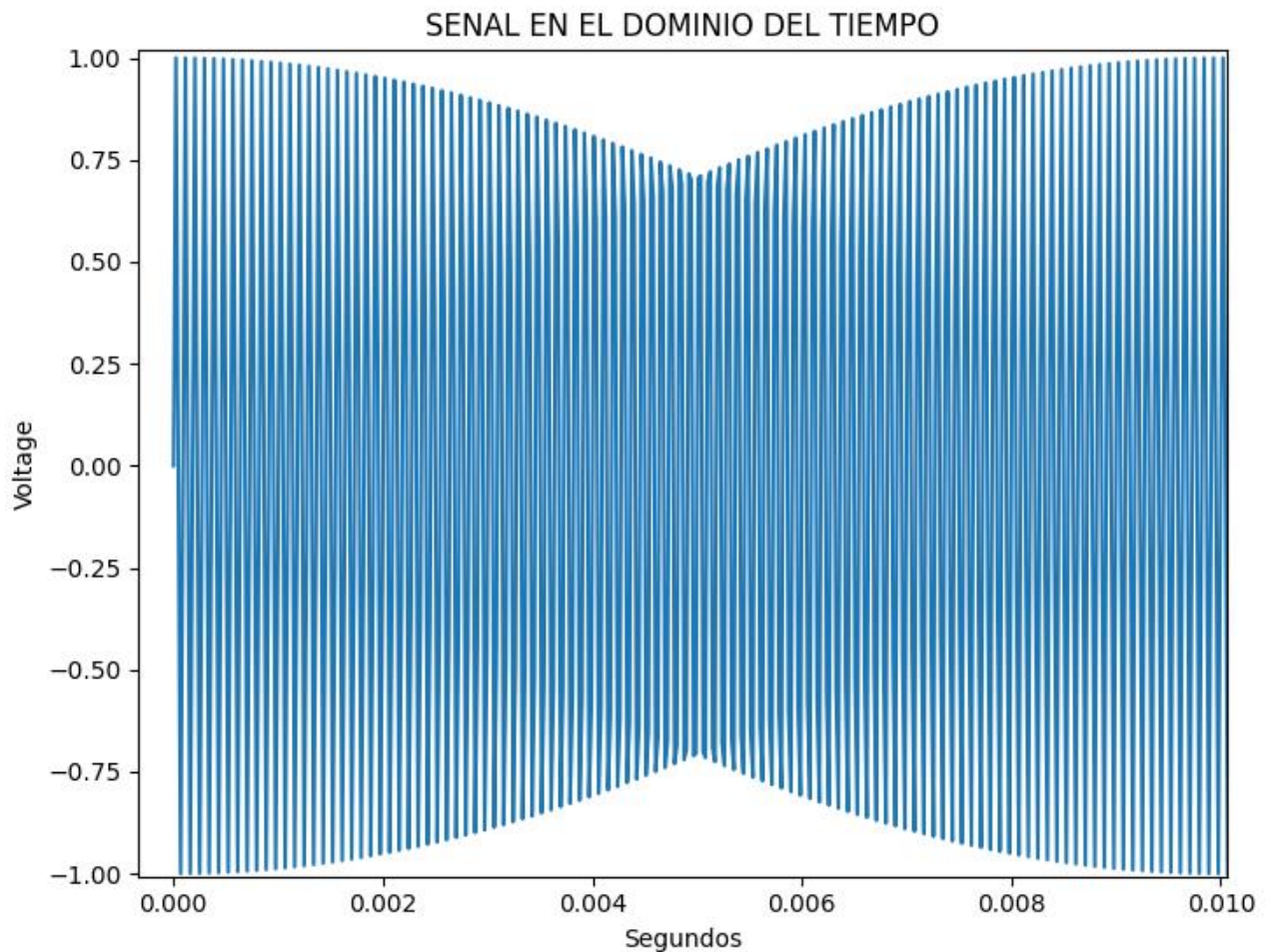
Frecuencia : 250Hz



Frecuencia: 1.2kHz



Frecuencia :11kHz



A continuación, escuche los tonos generados (función “audioplay”) prestando especial atención al nivel con que son percibidos en su oído. **¿Suenan con igual amplitud?**

No, en cuanto percepción, el tono emitido con una frecuencia de 1.2kHz, se nota mas sonoro que el emitido en 11kHz y 250 Hz, esto es debido a las curvas de Fletcher y Munson. En las que como se puede observar, para un mismo nivel de intensidad, la sonoridad es mayor en la frecuencia de 1.2 Khz.

Repita la escucha anterior de los 3 tonos pero con amplitudes de éstos de 0.5, 0.1, 0.05, 0.01 y 0.005. Para este punto, usaremos la función “audiowrite” de PYTHON para generar los distintos ficheros de audio. Después los abriremos con audacity para su escucha.

No modifique los niveles de los altavoces o de la salida de la tarjeta de audio del PC. Indique si, para un mismo nivel de amplitud, aprecia variaciones en el nivel con que percibe cada tono en su oído.

Amplitud	Resultados
0.5	En 250Hz, presenta un nivel mas bajo que con 11 kHz o 1.2 kHz, y entre 11 kHz y 1.2 kHz, el nivel que resulta percibido con mas intensidad es el correspondiente a 1.2 kHz.
0.1	Sigue habiendo una ligera variación en el nivel con respecto 11 kHz y 1.2 kHz, aunque con 250Hz presenta un nivel mas bajo.
0.05	Si se percibe una variación en el nivel de manera significativa, siendo el de mas nivel 1.2k Hz, aunque el nivel de 11k Hz no contiene con respecto 1.2k Hz una variación muy notoria a diferencia como ocurre con 250Hz.
0.01	Si se percibe una variación en el nivel de manera significativa, siendo el de mas nivel 1.2k Hz y seguido de 11k Hz.
0.005	Si se percibe una variación en el nivel de manera significativa, siendo el de mas nivel 1.2k <u>Hz</u> y seguido de 11k Hz.

Genere tonos de amplitud 0.05 y escúchelos. **Indique para qué frecuencias percibe el tono con mayor nivel.**

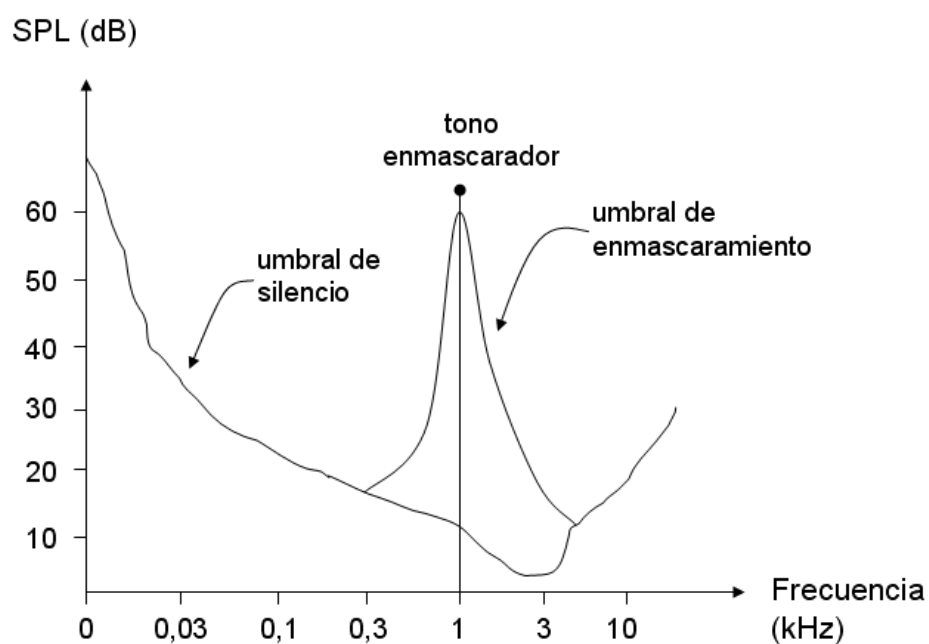
Frecuencia	Resultados
100 Hz	
500 Hz	
2 KHz	Se percibe con más nivel este tono
4 KHz	Se percibe con más nivel este tono
8 Khz	
12 KHz	

Por último, repita las pruebas anteriores (modificación de la frecuencia) pero incorporando la variación de la amplitud del tono. Compruebe la validez de las curvas de Fletcher y Munson.

Frecuencia	AMPLITUD					
	1	0.5	0.1	0.05	0.01	0.005
100 Hz						
500 Hz						
2 KHz						
4 KHz						
8 KHz						
12 KHz						

Apartado 5. Enmascaramiento en frecuencia

Un efecto conocido del oído es el denominado enmascaramiento. Éste corresponde con la modificación del umbral de silencio debido a la presencia de un sonido. A este sonido se le denomina “enmascarante”. Cualquier otro sonido cuyo nivel esté por debajo de ese nuevo umbral, llamado umbral de enmascaramiento, no será percibido por el oído.



Siga los siguientes pasos:

- Genere un tono de 1 KHz, amplitud 0.8, frecuencia de muestreo de 16 KHz, y 2 segundos de duración. Escúchelo (función "audioplay" de PYTHON).
- Genere un segundo tono de 1.5 KHz, frecuencia de muestreo de 16 KHz, 2 segundos de duración y amplitud de valor un quinto al del primer tono generado. Escuche este segundo tono.
- Sume ambos tonos y escuche la señal resultante. **Compruebe que se escucha un único tono. Incremente la frecuencia del tono más débil hasta perciba los dos tonos. Anote la frecuencia para la que comienza a distinguir el 2º tono.**

Se puede distinguir en la siguiente octava de 1 kHz, pero, para asegurarme de que se va a percibir correctamente 2 tonos simultáneos, he preferido poner 4000Hz, esto se debe a que como podemos ver en la curva de Fletcher y Munson, a la altura de 4000Hz, si por ejemplo emitimos un sonido con 85 dB, para un 1K Hz si son 85 dB, pero cuando normalizamos 85dB con 4000 Hz a 1 kHz, estos 85dB , crecen, siendo mayores a 1 kHz y si estos 85 dB emitidos con 4kHz son comparados con 1 kHz, haciendo que su sonoridad aumente, como es el caso.

Práctica 6. Simulación del efecto Haas

El efecto Haas, también es conocido como "Efecto de precedencia o prioridad". El nombre se debe a Hemult Hass, médico alemán que lo describió en su tesis doctoral.

El oído humano no sólo depende de cómo de alto sea un sonido para establecer de dónde viene, sino que además utiliza el tiempo de retardo que emplea en llegar hasta nosotros. En otras palabras, si el sonido proviene de diversas fuentes, el cerebro únicamente toma en cuenta el sonido que proviene de la fuente más cercana.

Este efecto tiene mucha importancia tanto en los sistemas estéreo, donde se cuida la colocación de los altavoces y el oyente, como en los sistemas de cine en casa, donde es importante establecer un tiempo de retardo entre los altavoces traseros y los delanteros, para que, aunque los canales traseros estén más cerca de nosotros que los delanteros, el sonido parezca que proviene de delante.

El efecto Haas describe cómo, a nivel de percepción, si varios sonidos independientes llegan a nuestro cerebro en una ventana temporal inferior a 50 ms, éste los fusiona y los interpreta como uno sólo. Esto se debe a que el cerebro deja de percibir la dirección y entiende los sonidos posteriores como un eco o reverberación del primero.

Esta interpretación el cerebro la hace de cuatro modos distintos:

- Si el retardo es inferior a 5 ms, el cerebro localiza al sonido en función de la dirección que tuviera el primer estímulo, aunque los otros provengan de direcciones diametralmente opuestas. En un sistema estéreo, el sonido se escucha en un punto central muy focalizado situado en la bisectriz del ángulo de los dos altavoces, con la impresión subjetiva de ser de doble intensidad por la suma de la emisión de los dos canales.
- Si el retardo llega entre 5 y 35 ms, el oyente escucha un único sonido, sigue siendo de intensidad doble pero empieza a distinguir la procedencia del mismo. En un sistema estéreo, el sonido se

escucha en un punto central alto más amplio, en la bisectriz del ángulo de los dos altavoces.

- Si el retardo llega entre 35 y 50 ms, se oye el sonido de manera separada de las diferentes fuentes, pero procedente de la fuente inicial, con algo menos de intensidad. En un sistema estéreo el sonido se sigue escuchando proveniente de entre los dos altavoces, pero se distingue la colocación de ambos. Es decir, empieza a perder "consistencia" la escena sonora.
- Si el retardo llega con un tiempo superior a 50 ms, el sonido se oye de manera separada y procedente de cada una de las fuentes, con su respectiva intensidad. En un sistema estéreo, no existiría escena sonora, notaríamos perfectamente dónde está cada altavoz y qué sonido proviene de cada uno. **En este caso, se produce el efecto conocido como eco.**

En esta práctica se pretende reproducir el efecto Haas. Para ello, realice los siguientes pasos:

- Grabe una o dos palabras intentando que la calidad obtenida sea alta.
- A continuación, implemente una función en PYTHON que introduzca un retardo a la señal original y que éstas, tanto la original como la retardada, se escuchen por el canal izquierdo-derecho del auricular. Tome como referencia el siguiente ejemplo:

```
eco = float(input('Indique el retardo con que llega el eco al oyente (ms) '))
fs, x = audioread('haas_left.wav')
ceros = np.zeros(int(fs*eco/1000.0))

haas = np.vstack((np.hstack((x, ceros)), np.hstack((ceros, x))))
audioplay(haas, fs)
audiowrite('haas' + str(int(eco)) + '.wav', fs, haas)
```

Pruebe con los siguientes valores de retardo: 0, 10, 20, 30, 40, 50, 60, 70, 80, 90, 100, 200 y 500 ms.

Disposicion espacio sonoro

Retardo (ms)	Resultados
0	Se escucha como el original, ya que no ha sufrido ninguna modificación
10	En este momento, se puede empezar a distinguir el efecto flanger en frecuencia, crea un efecto peine.
20	Aquí es mas notable como se empiezan a distinguir la procedencia del audio.
30	Es distinguible la procedencia del sonido, parece que hay mas de una fuente de origen, en el propio origen
40	Se nota la procedencia del origen del sonido, pero se nota como que no se encuentra las fuentes bien distribuidas
50	Se empieza a notar como una especie de eco, pero, aun no es lo suficientemente notable
60	Se produce como pequeños ecos
70	Parecido a si nos encontramos en una habitación cerrada
80	Mismo efecto, pero mas notorio

90	Parece una especie de teatro(sitio cerrado amplio) en el que se produce mucho eco.
100	Se escucha como si tuviéramos dos procedencias distintas del sonido en la que una con respecto a la otra se produce con menos intensidad
200	En este caso resulta confuso
500	Confuso la procedencia original del audio, parece que estamos girando en torno al audio original.

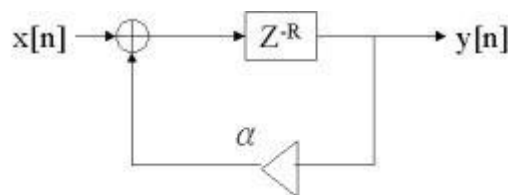
Indique en el recuadro de abajo los resultados obtenidos (si se aprecia eco o no, efecto estéreo?, origen del sonido,...)

Si, se aprecia el efecto eco, al principio empezamos a notar mas de una fuente proveniente desde el origen, pero a medida que aumentamos nuestro retardo, esa fuente se vuelve en ecos y después si seguimos aumentando, empieza a ser confuso la procedencia del sonido, hasta el caso limite en el que por ejemplo, en 500ms, parece que tienes dos fuentes distintas en cada lado de la cabeza, realmente lo que se encuentra al principio antes de llegar a la percepción de un eco, es una amplitud de la disposición del espacio sonoro, percibimos como si estuviéramos acercándonos al escenario de un concierto que tenemos justo delante de nosotros, esto ocurre desde 0ms hasta 50ms que es el límite para la aparición del primer eco.

Práctica 7. Filtro Multi-eco

Este filtro realiza un infinito número de ecos espaciados R periodos simples con decaimiento exponencial de amplitud.

El diagrama de bloques del filtro multi-eco se muestra a continuación.

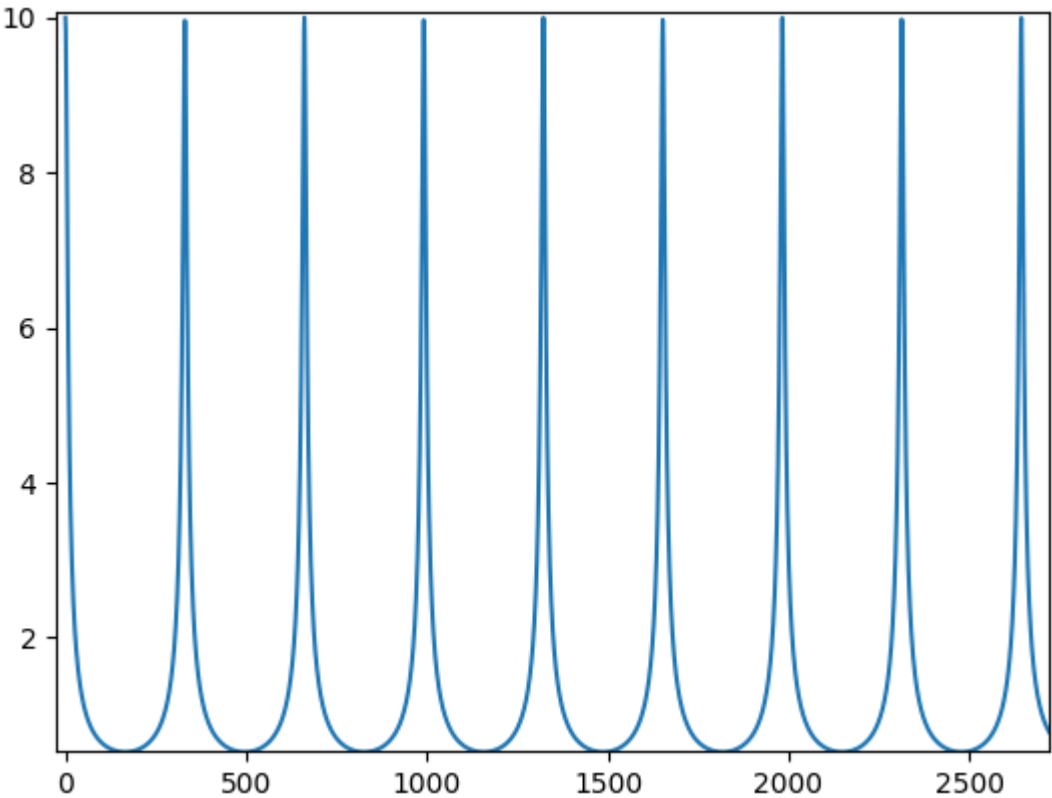


Su función de transferencia viene dada por:

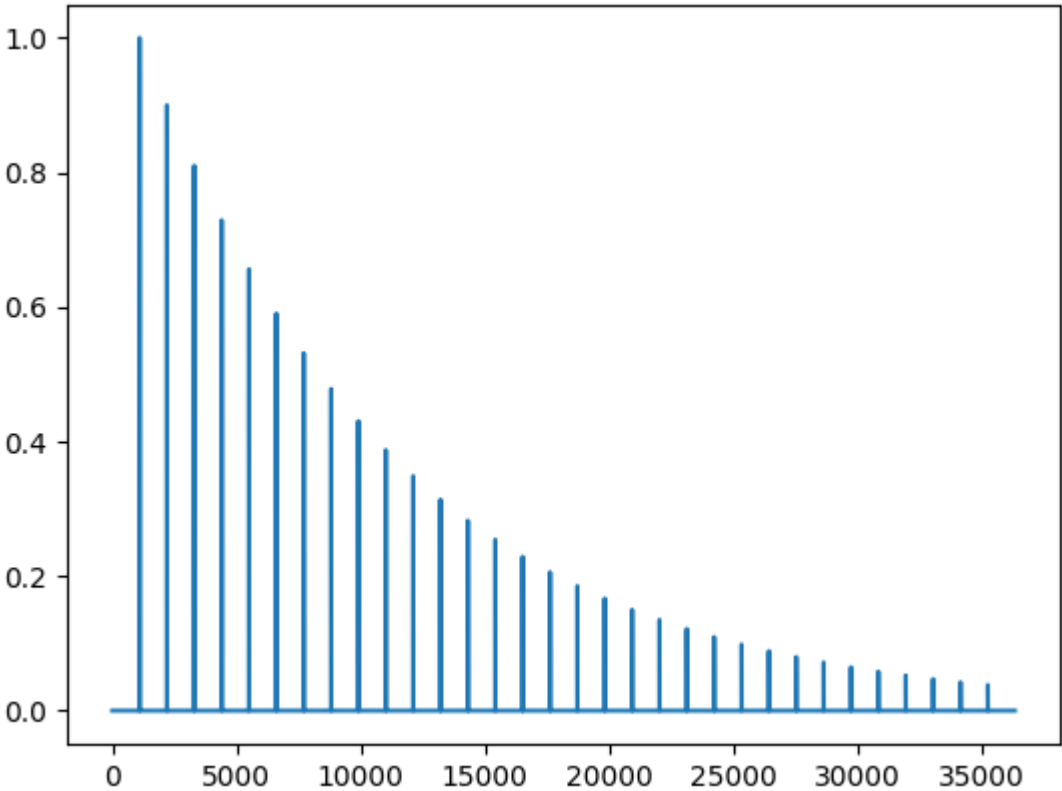
$$H(z) = \frac{z^{-R}}{1 - \alpha z^{-R}}$$

Representa a la respuesta al impulso, $h[n]$, del sistema anterior y su respuesta en frecuencia, $H(z)$.

Respuesta en frecuencia:

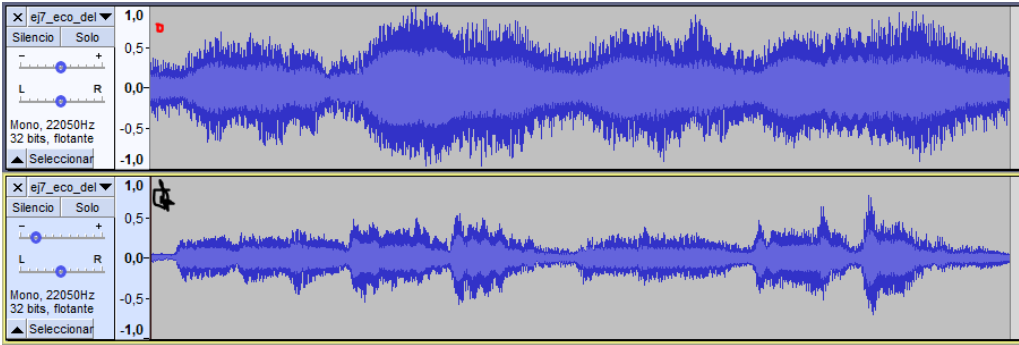


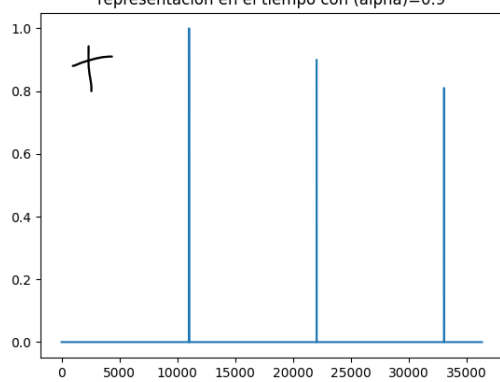
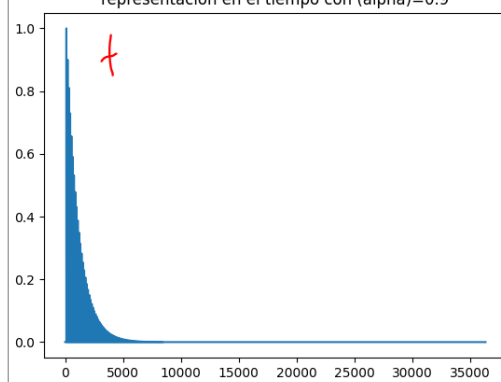
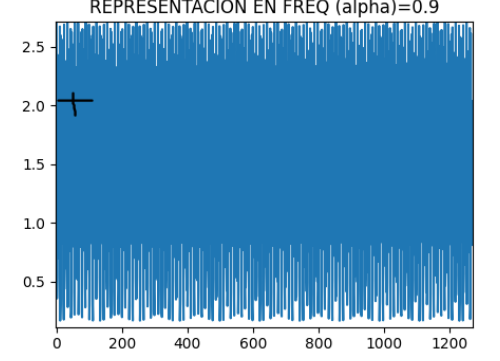
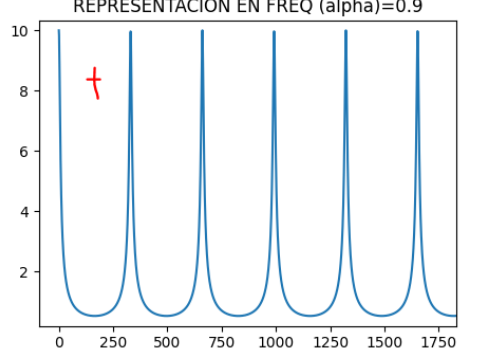
Respuesta en el tiempo:



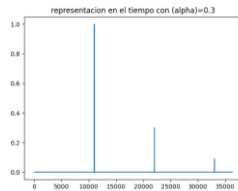
Comente los resultados obtenidos al realizar las siguientes pruebas con el fichero el fichero “clackson.wav”:

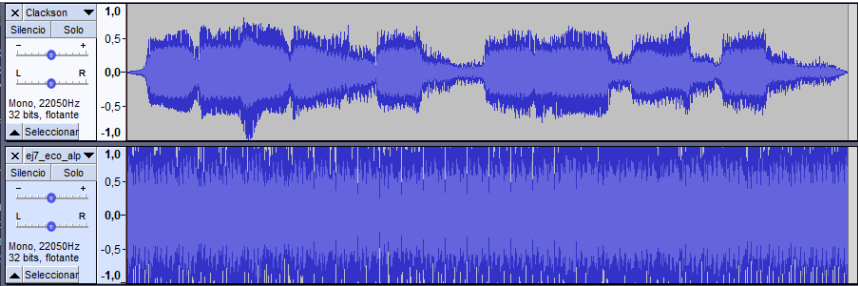
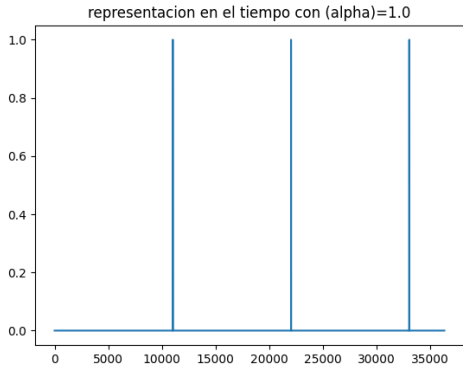
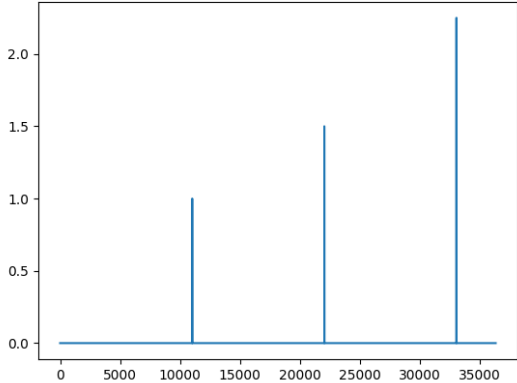
- a) Manteniendo constante en valor del parámetro α en 0.9, varíe el valor del retardo introducido asignando los siguientes valores: 1 ms, 5 ms, 20 ms, 50 ms, 100 ms, 200 ms, 500 ms, 2 segundos. Obviamente, para determinar el número de etapas de retardo, R , que debe incluir ha de considerar la frecuencia de muestreo. Escuche los resultados obtenidos. **Indique para qué valores de retardo se percibe el eco.**

Retardo (ms)	¿Eco?
1	Con un 1ms de retardo, resulta inapreciable ninguna alteración con respecto al audio original.
5	En este caso sigue siendo un delay pequeño con lo que, los distintos ecos de la señal no pueden ser audibles como eventos independientes por lo que realmente escucho es una combinación de ambas acontecido de un cambio tímbrico .
20	<p>Este sonido, se escucha con mas “fuerza” a comparación de la aplicación de un retardo de 5ms, esto es porque en este caso, los armónicos con una duración de 20ms o alguna fracción entera o algún múltiplo en frecuencia de este valor hará que los ecos que se van generando continuamente, se combinen en fase haciendo esto, que se sumen, esto hace que se vea reforzada la señal, en cambio, en 5 ms, la mayoría, entran en contrafase, haciendo que su nivel de señal decaiga, es fácil de ver en Audacity:</p> <div></div> <p>El negro se aplica a 5ms y el rojo a 20 ms</p>
50	En 50ms tenemos la situación limite de la aparición de eco y en efecto, es así, se empieza a notar los primeros ecos, se empiezan a percibir eventos de forma separada, aunque la percepción de los mismos es muy ligera.
100	En este caso, se puede ver de la siguiente manera, tenemos una duración de la señal de 1.6 segundos y tenemos un eco cada 100ms con lo cual, tenemos 16 ecos, con lo cual si nos damos cuenta a medida que nuestro retardo aumente, realmente, lo que estamos haciendo, es realizar menos retardos sobre nuestra señal original.
200	En este caso se corrobora aun mas lo que he escrito en el apartado superior, Al escucharlo, se notara la reducción de ecos.
500	<p>Como estamos viendo a medida que aumentamos los ms, nuestros ecos se van reduciendo, pero esto se puede ver a parte de auditivamente y representando la respuesta de nuestro filtro en el tiempo:</p> <p>Negro=500ms</p> <p>Rojo=5ms</p>

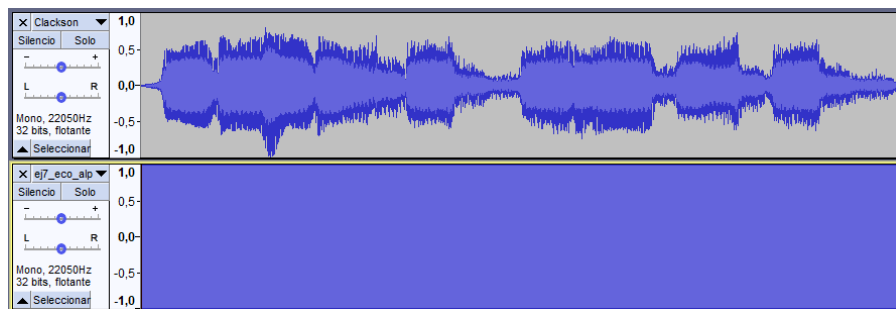
	<div><div>representacion en el tiempo con (alpha)=0.9</div><div>representacion en el tiempo con (alpha)=0.9</div></div> <p>También es de esperar, que si decidimos verlo en frecuencia, como bien, sabemos, cuanto mas separación en el tiempo, mas unión en frecuencia como se puede observar,</p> <div><div>REPRESENTACION EN FREQ (alpha)=0.9</div><div>REPRESENTACION EN FREQ (alpha)=0.9</div></div> <tr><td>2000</td><td>En este caso, no se escucha nada ya que el tiempo de retardo introducido es mayor que el tiempo de duración original de la señal.</td></tr>	2000	En este caso, no se escucha nada ya que el tiempo de retardo introducido es mayor que el tiempo de duración original de la señal.
2000	En este caso, no se escucha nada ya que el tiempo de retardo introducido es mayor que el tiempo de duración original de la señal.		

b) Ahora mantenga constante el valor del retardo en 50 milisegundos y varíe los valores del parámetro α asignándole los valores de 0.03, 0.1, 0.3, 0.5, 0.7, 0.85, 1, 1.5. Escuche los resultados obtenidos. **Indique para qué valores de retardo se percibe el eco**

α	¿Eco?
0.03	No se percibe ningún eco asociado a este valor de α .
0.1	Al igual que en el caso anterior, no se percibe aun ningún eco asociado a este valor.
0.3	Análogamente a los apartados anteriores, esto es debido a : <div><div>representacion en el tiempo con (alpha)=0.3</div></div> Como se puede ver, el siguiente eco, tiene una amplitud, mucho menor que la principal, con lo cual, acaba enmascarando el primer eco a los demás .
0.5	Se empieza a percibir como unas pequeñas alteraciones al sonido original.
0.7	En esta situación es notorio el eco.

0.85	<p>En este caso al igual que en el anterior , resulta notorio, aunque hay que decir, que ya nos encontramos cerca de la situación limite que es 0.99, ya que no podemos olvidar que al final estamos realizando un procesamiento digital de la señal y cuando procesamos una señal con un polo en el que nos encontramos un polo con un modulo unidad o superior, nuestro sistema se comporta se manera inestable, como vamos a poder comprobar en los siguientes casos.</p>
1	<p>Este es el caso limite en el que tenemos inestabilidad, aunque en este caso es inestable pero se puede implementar, esto es debido a que se encuentra truncada nuestra señal original, veamos esta inestabilidad:</p> <p>Si comparamos la señal original con Alpha 1 en Audacity vemos que :</p>  <p>Si representamos dicha respuesta impulsiva en el tiempo, tenemos que :</p>  <p>Es de esperar esta respuesta ya que nuestro en el filtro en el tiempo es</p> $a^n u[n]$ <p>Con lo cual, como cualquier exponente de 1 el resultado es 1, nuestro filtro tiene este comportamiento.</p>
1.5	<p>En este caso, la inestabilidad es total, ya que nuestro sistema ahora, tiende a infinito.</p> 

En Audacity, veremos una saturación del audio:



(aunque en nuestro caso, tenemos truncada la señal)

¿Como se puede ver numéricamente?

Bien, el polo que tenemos en nuestro filtro es $(1-1.5z^{-R})$, con lo cual el modulo del mismo, es mayor de la circunferencia unidad.

$$a^n u[n] \quad \frac{1}{1-(a/z)} = \frac{z}{z-a} \quad |z| > |a|$$

Como estamos viendo, en este caso, Alpha vale 1.5, con lo cual, en el tiempo, a medida que avancemos, resulta una exponencial creciente, volviendo inestable nuestro sistema, en cambio, si Alpha es menor que 1 como por ejemplo 0.8, a medida que avancemos en el tiempo, nuestra exponencial ira decreciendo hasta hacerse 0, ya que, algo multiplicando por si mismo menor que cero, siempre será menor de cero y menor al valor original, en cambio, ocurre todo lo contrario para 1.5, a medida que aumentemos el exponente en un numero mayor de 1, tendremos un comportamiento exponencial decreciente.

Gráficamente, se ve mejor en la siguiente imagen

