



UNIVERSIDAD DE BUENOS AIRES
FACULTAD DE INGENIERÍA
Año 2017 - 1^{er} Cuatrimestre

SEÑALES Y SISTEMAS (85.05)

TRABAJO PRÁCTICO ESPECIAL
TEMA: Modificación del *tempo* de un fragmento musical.
FECHA: 17 de junio de 2017

INTEGRANTES:

Manso, Juan - #96133
<juanmanso@gmail.com>

Índice

1. Ejercicio 1	1
1.1. Consideraciones previas	1
1.2. Banda ancha	1
1.3. Banda angosta	2
2. Ejercicio 2	3
2.1. Parámetros del espectrograma	4
2.2. Detección de las notas	4
3. Ejercicio 3	5
3.1. Interpolación	5
3.2. Comparación de espectogramas	5
4. Ejercicio 4	7
4.1. Decimación	8
4.2. Comparación de espectrogramas	8
5. Ejercicio 5	9
5.1. Implementación final	9
5.2. Comparación	9
6. Ejercicio 6	11
6.1. Implementación	11
6.2. Comparación de espectrogramas y modificación de la implementación	12
7. Ejercicio 7	14
7.1. Implementación	14
7.2. Comparación de espectrogramas	14
8. Conclusiones	15

1. Ejercicio 1

Analizar el fragmento musical que contiene el archivo **audio.wav** mediante el espectrograma de banda ancha y banda angosta. Describir qué es lo que se puede observar en ambos casos y explicar el criterio usado para la elección de los parámetros.

1.1. Consideraciones previas

En este Ejercicio se requieren los espectrograma de banda ancha (buena definición en frecuencia) y de banda angosta (buena definición en tiempo). Al aumentar la resolución en tiempo, baja necesariamente la resolución en frecuencia y viceversa. Por lo tanto el espectrograma de banda ancha carecerá de buena definición de las frecuencias en tiempo y en el de banda angosta sucederá lo mismo con las frecuencias entre sí.

En orden de definir un ancho de ventana correcto, se requiere una relación entre las muestras y el tiempo que representan.

$$n = t \cdot F_S \quad (1)$$

Dado que el conversor analógico-digital toma una señal de tiempo total t y se la muestrea con frecuencia de muestreo F_S , la señal discreta tendrá n elementos. A partir de esto, se obtiene la ecuación (1.1). En particular, la señal del archivo **audio.wav** fue muestreada con $F_S = 16\text{ kHz}$.

1.2. Banda ancha

Para una buena resolución en frecuencia, las ventanas del espectrograma deben ser grandes. Considerando que $\Delta f = 0,5\text{ Hz}$ es un intervalo apropiado de frecuencia, se desprende el siguiente cálculo (a partir de (1.1)).

$$\begin{aligned} l_{ventana} &= t_{ventana} \cdot F_S \\ l_{ventana} &= \frac{1}{\Delta f} \cdot F_S \\ l_{ventana} &= \frac{16\text{ kHz}}{0,5\text{ Hz}} \\ \Rightarrow l_{ventana} &= 2 \cdot F_S = 32.000 \end{aligned}$$

Se define además un *overlap* de un tercio del largo de la ventana (para tener suavidad en la curva) y la ventana del tipo *Hamming*. Así se obtiene el espectrograma de la Figura 1.

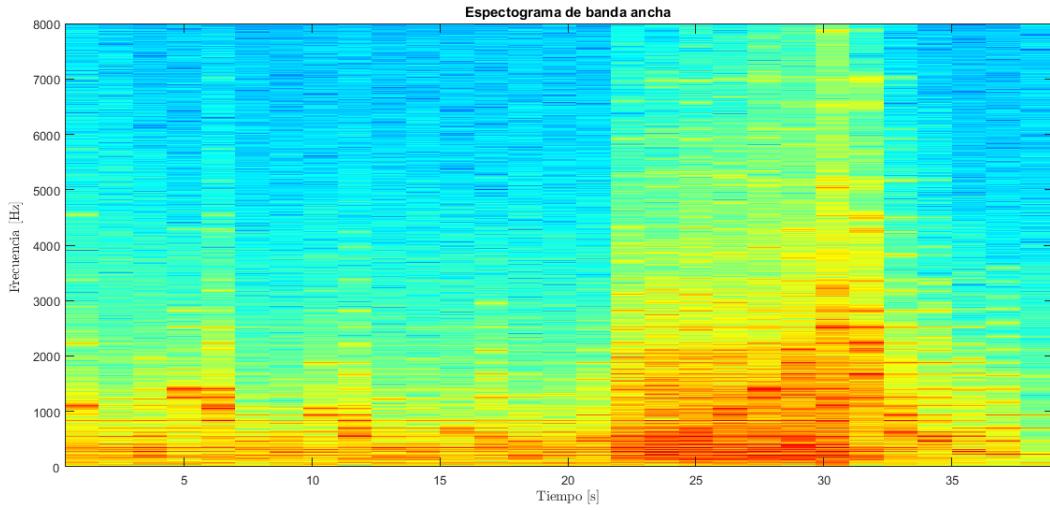
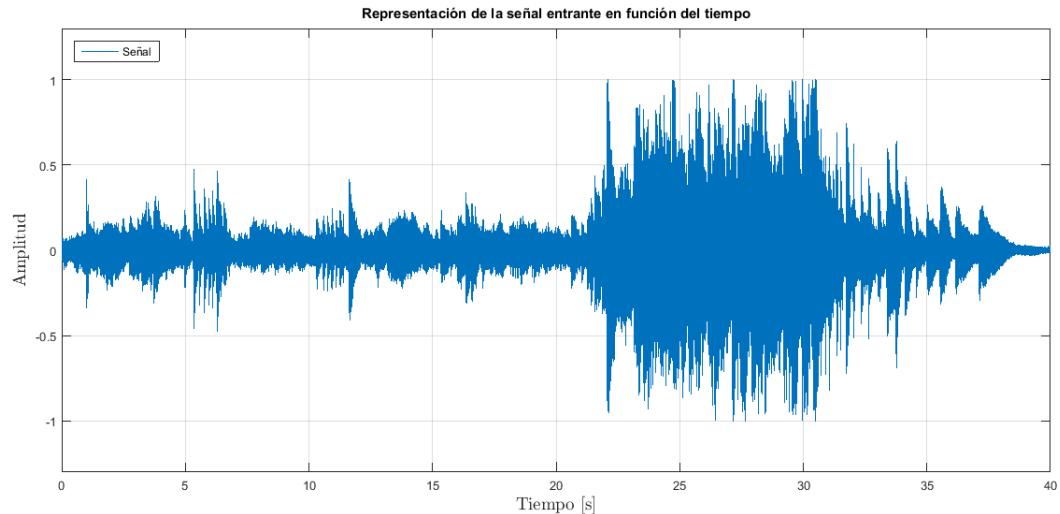


Figura 1: Espectrograma de banda ancha.

1.3. Banda angosta

Figura 2: Señal del archivo **audio.wav**.

Suponiendo que la señal fuese un tren de pulsos, un criterio válido sería que haya 10 ventanas por pulso. A partir del gráfico de la señal de la Figura 2, se escoge como pulso patrón el representado en la Figura 3.

Por lo tanto, como $t_{ventana} = 0,2\text{ s}$:

$$\begin{aligned} 10 \cdot l_{ventana} &= t_{ventana} \cdot F_S \\ l_{ventana} &= \frac{0,2\text{ s} \cdot 16\text{ kHz}}{10} \\ \Rightarrow l_{ventana} &= \frac{F_S}{50} = 320 \end{aligned}$$

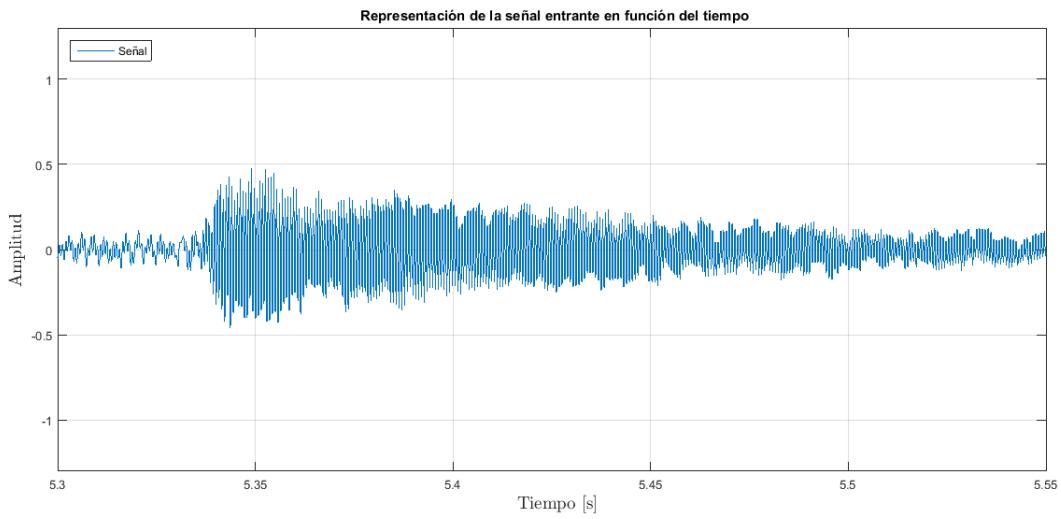


Figura 3: Parte de la señal original que se utiliza como patrón.

Con el mismo *overlap* relativo y utilizando ventanas de *Hamming* se obtiene el espectrograma de la Figura 4.

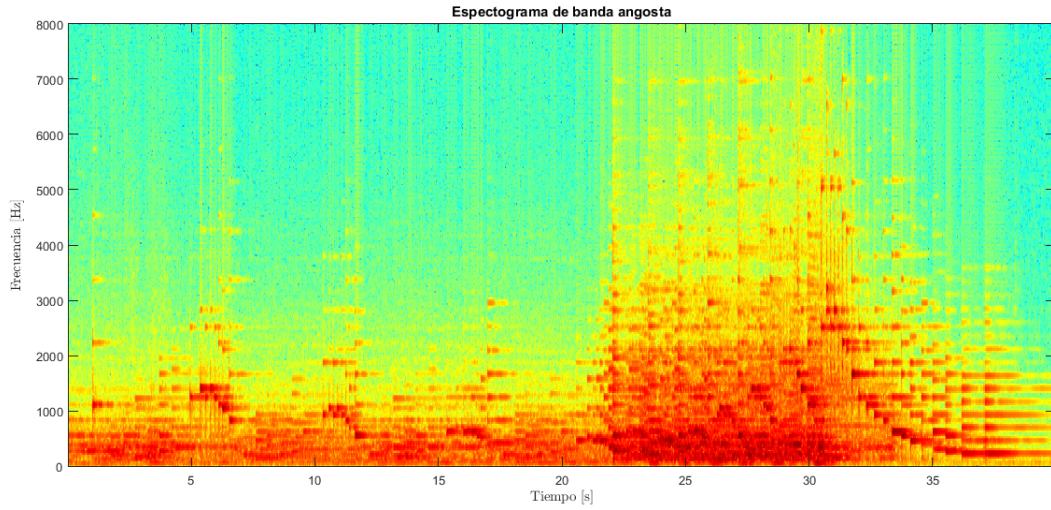


Figura 4: Espectrograma de banda angosta.

2. Ejercicio 2

Usando la información obtenida en el ejercicio anterior, obtener los parámetros del espectrograma que permitan observar de la mejor manera posible las características tiempo-frecuencia de este archivo de audio. Por medio de este espectrograma debe ser posible identificar las siguientes notas del piano cuyas frecuencias aproximadas se representan en la Tabla 1.

Do	Do#	Re	Re#	Mi	Fa	Fa#	Sol	Sol#	La	La#	Si
523	554	587	622	659	698	740	784	831	880	932	988
1046	1108	1174	1244	1318	1386	1480	1568	1662	1760	1864	1976

Tabla 1: Frecuencias que representan las notas musicales en su cuarta y quinta octava.

2.1. Parámetros del espectrograma

De la Tabla 1 se ve que el intervalo más corto de frecuencias es $554 - 523 = 31 = \Delta f_{\min}$. Además por el Ejercicio 1, se sabe que $\Delta t_{\min} = 0,2\text{s}$. Por lo tanto:

$$F_S \cdot 0,2\text{s} = \frac{F_S}{5\text{Hz}} > l_{ventana} > \frac{F_S}{31\text{Hz}}$$

Se propone utilizar $l_{ventana} = \frac{F_S}{10\text{Hz}}$ y resulta el gráfico de la Figura 5. Como criterio se propone un error relativo del 10 %. Si el error es menor, la representación se considerará correcta.

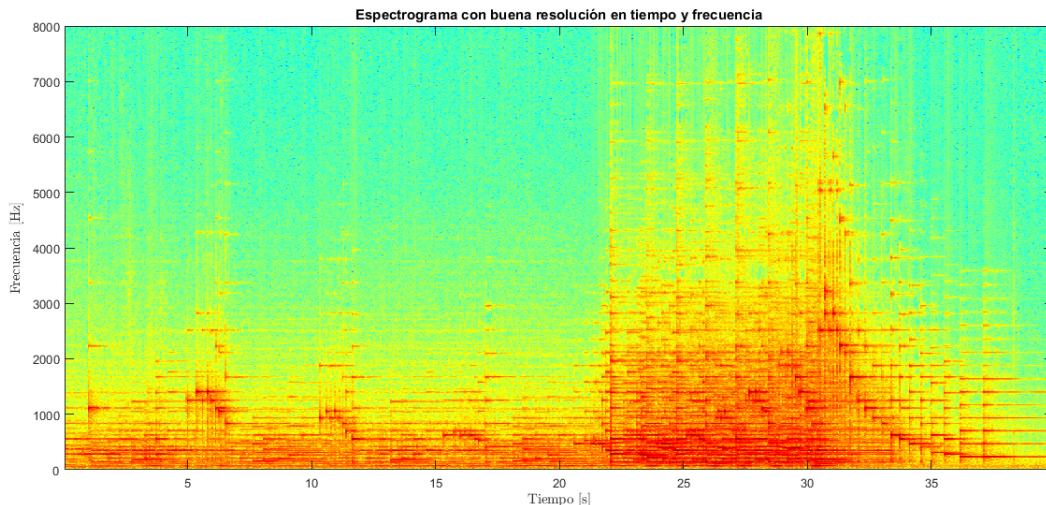


Figura 5: Espectrograma de precisión.

2.2. Detección de las notas

Se buscaron las frecuencias que representan a las notas musicales en el espectrograma y se obtuvieron sus valores más próximos expuestos en la Tabla 2 y sus respectivos errores en la Tabla 3.

Do	Do#	Re	Re#	Mi	Fa	Fa#	Sol	Sol#	La	La#	Si
523,44	554,69	585,94	625	656,25	695,31	742,19	781,25	828,13	882,81	929,69	992,19
1046,9	1109,4	1171,9	1242,2	1320,3	1390,6	1484,4	1570,3	1664,1	1765,6	1867,2	1976,6

Tabla 2: Frecuencias del espectrograma más próximas a las de la tabla 1.

Do	Do#	Re	Re#	Mi	Fa	Fa#	Sol	Sol#	La	La#	Si
0,4375	0,6875	1,0625	3	2,75	2,6875	2,1875	2,75	2,875	2,8125	2,3125	4,1875
0,875	1,375	2,125	1,8125	2,3125	4,625	4,375	2,3125	2,0625	5,625	3,1875	0,5625

Tabla 3: Diferencia entre las frecuencias del espectrograma que representan a las notas y las reales.

A partir de estas tablas se ve que la máxima diferencia es de 5,625 (5^{ta} octava del La) y el error relativo máximo es de 0,5% (4^{ta} octava de Re). Al estar dentro del criterio propuesto, el espectrograma permite identificar las notas requeridas.

3. Ejercicio 3

Se pretende duplicar la duración del fragmento musical para lo cual se propone interpolar el archivo de audio. No está permitido usar las funciones de matlab/octave que implementan la interpolación directamente. Grafique el espectrograma de la nueva secuencia y compárela con la obtenida en el Ejercicio 2.

3.1. Interpolación

El algoritmo de interpolación lineal se simplifica al contar con el álgebra matricial propio del Matlab. La rutina se resume a continuación:

```

1 doble_audio = linspace(0,1,length(audio)*2);
2 doble_audio(1:2:end) = audio;
3 doble_audio(2:2:end-1)=(audio(2:end)-audio(1:end-1))/2+audio(1:end-1);
```

3.2. Comparación de espectogramas

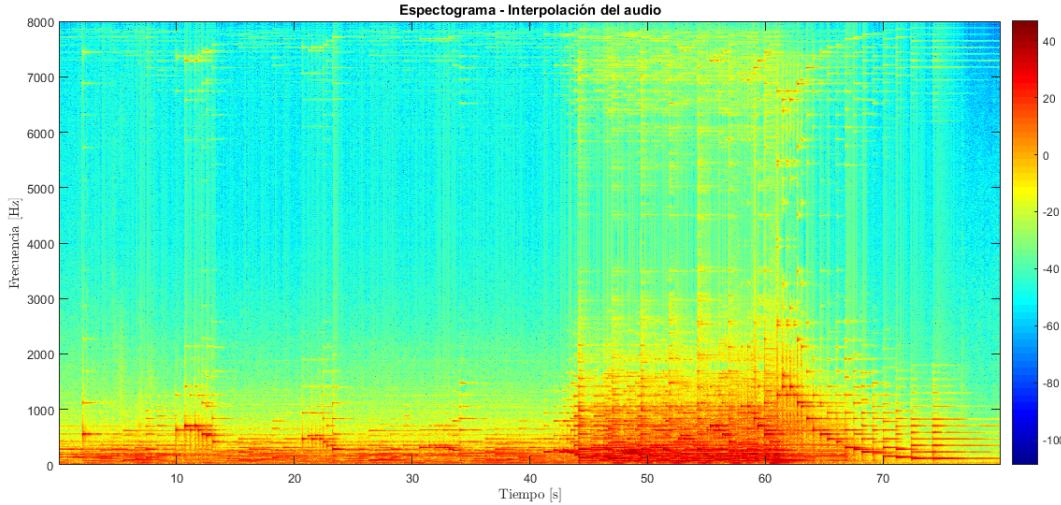


Figura 6: Espectrograma de la señal interpolada.

Al interpolar la señal original, se obtiene el doble de muestras. Ante una misma F_S (Figura 6), la duración de la señal interpolada será el doble. A su vez, el espectro de frecuencias se comprime en 2.

Realizando el espectrograma de la señal interpolada con $F_S^* = 2F_S$ (Figura 7), se ve que las frecuencias son las mismas. Por lo tanto, si se reconstruyen ambas señales con sus respectivas frecuencias de muestreo se obtiene la misma señal.

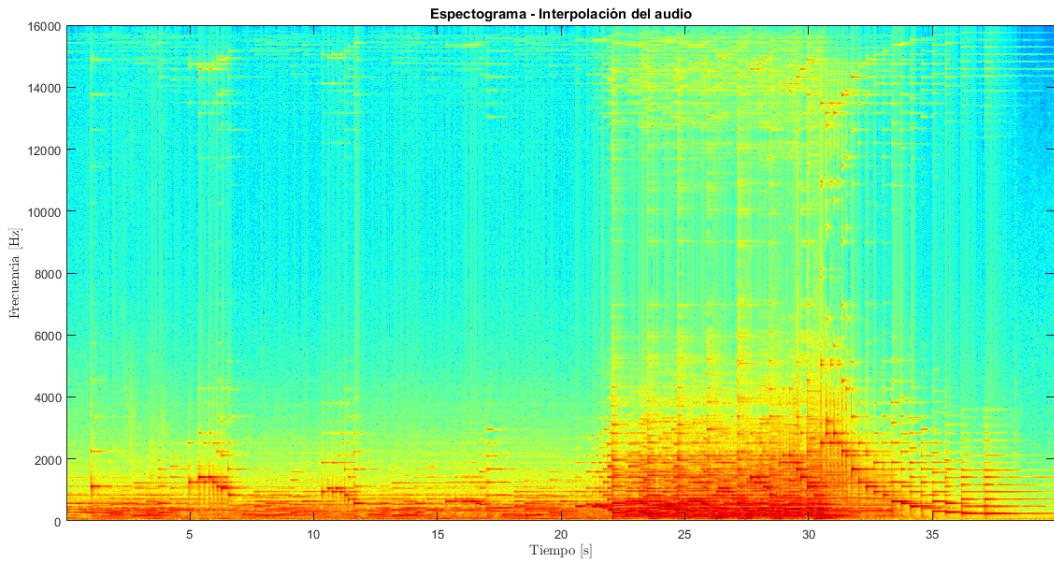


Figura 7: Espectrograma de la señal interpolada con $F_S^* = 2F_S$.

Por otro lado se ve que, debido a la interpolación, se presentan componentes de alta frecuencia que la original carecía ($f > 8 \text{ kHz}$ con $F_S^* = 2F_S$). Esto se debe a que, al ser periódica la DFT, la compresión del espectro copia parte del mismo espectro espejado. Por lo tanto, se propone realizar un filtrado pasabajos. El filtro a utilizar es una ventana rectangular en frecuencia. En la Figura 8 se expone el espectrograma de la señal interpolada y filtrada posteriormente.

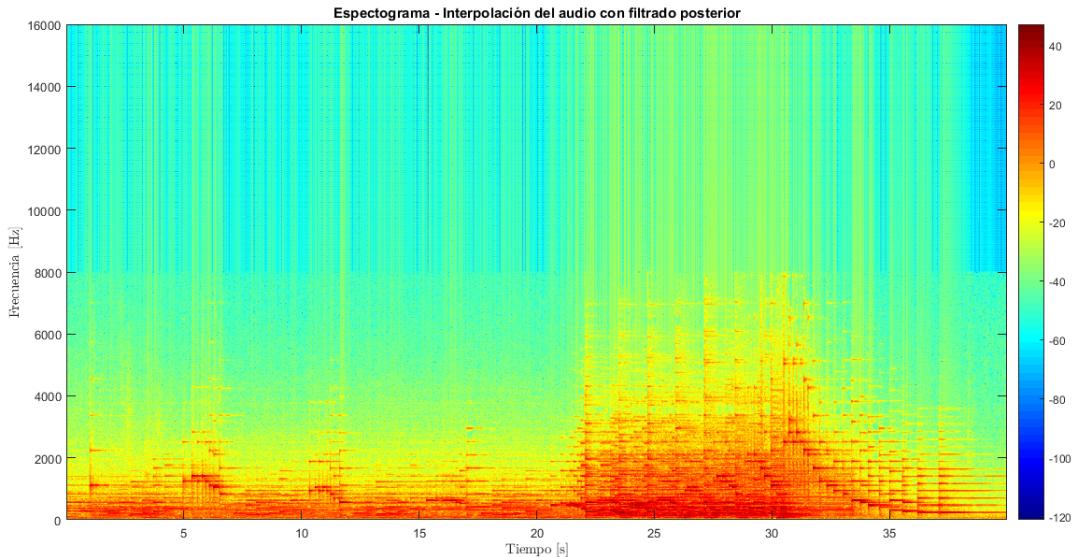


Figura 8: Espectrograma de la señal interpolada filtrada con $F_S^* = 2F_S$.

Se podría interpretar que al aplicar el filtro la señal presentaría más ruido (zonas previamente en azul se grafican amarillas ahora). Esto se debe a que el graficador toma el valor más pequeño y lo define como azul oscuro. Se puede ver en las Figuras 9 y 10 que un mismo punto mantiene el valor de z (denominada *Index* en los gráficos). Por lo tanto, la señal filtrada es más fiel a la original dado que carece de los componentes de alta frecuencia parásitos.

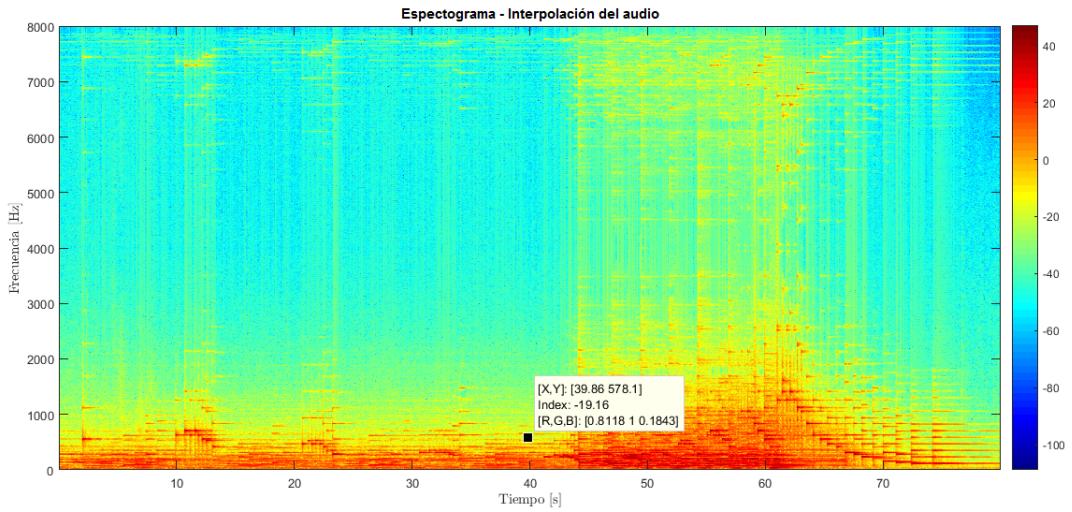


Figura 9: Espectrograma para comparar - Interpolación sin filtro.

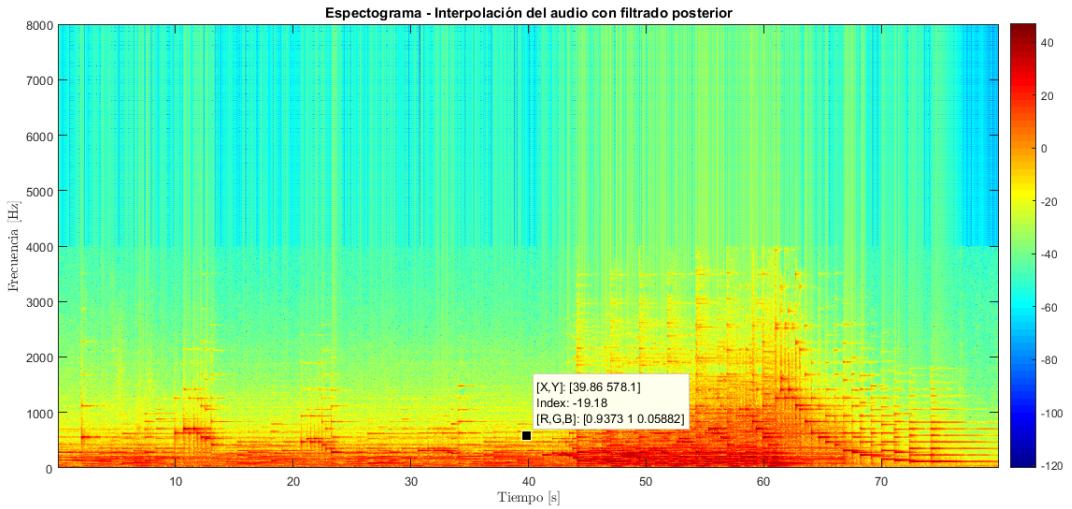


Figura 10: Espectrograma para comparar - Interpolación con filtro.

4. Ejercicio 4

Reducir a la mitad la duración del fragmento musical utilizando el proceso de decimación. No está permitido usar las funciones de matlab/octave que implementan la decimación directamente. Recuerde que para evitar el aliasing es necesario realizar el adecuado filtrado pasabajos. Utilice alguno de los métodos de diseño explicados en la teórica especificando el criterio utilizado. Grafique el espectrograma de la nueva secuencia y compárela con la obtenida en el Ejercicio 2.

4.1. Decimación

Se recuerda que, cuando se decima por N una señal discreta que fue muestreada, al comprimir el espectro de frecuencias, la señal útil queda entre $-\pi/N$ y π/N (en vez de $-\pi, \pi$). Todo lo demás tendrá aliasing al pasar por el conversor digital analógico (DAC). Por ende, como se pretende decimar por $N = 2$, se aplica un pasababajos ideal entre $-\pi/2$ y $\pi/2$ (o en frecuencia $-F_S/4$, $F_S/4$). Dado que la función `fft()` da los valores entre 0 y 2π , sólo se requiere eliminar desde $\pi/2$ hasta

$\frac{3}{2}\pi$. Se utiliza el parámetro 'symmetric' de la función `ifft()` para que sea posible reconstruir al poner ceros desde $\pi/2$ hasta 2π en el espectro de frecuencias.

A continuación se expone la implementación utilizada:

```

1 aux=audio;
2
3 % Pasaje a frecuencia
4 NFFT=2^nextpow2(length(aux));
5 fft_mitad=fft(aux,NFFT);
6
7 % Frecuencias menores y mayores a |Fs/4| ==> =0
8 fft_mitad(fix(length(fft_mitad)/4)+1:end)=0;
9
10 % Reconstrucción
11 % Como se eliminó desde 3/2*pi hasta 2*pi, queda asimétrica, por lo tanto
12 % se utiliza el parámetro 'symmetric' para solucionarlo.
13 aux=ifft((fft_mitad),'symmetric');
14
15 % Decimación.
16 mitad_audio=aux(1:2:end-1);
17 mitad_audio=mitad_audio(1:round(length(audio_input)/2));

```

4.2. Comparación de espectrogramas

Como se sabe, al comprimir el dominio del tiempo, se expande el dominio de las frecuencias. Así, se ve en la Figura 11 compensando la frecuencia de muestreo, el eje de frecuencias se expande en 2 con respecto a la original (Figura 5).

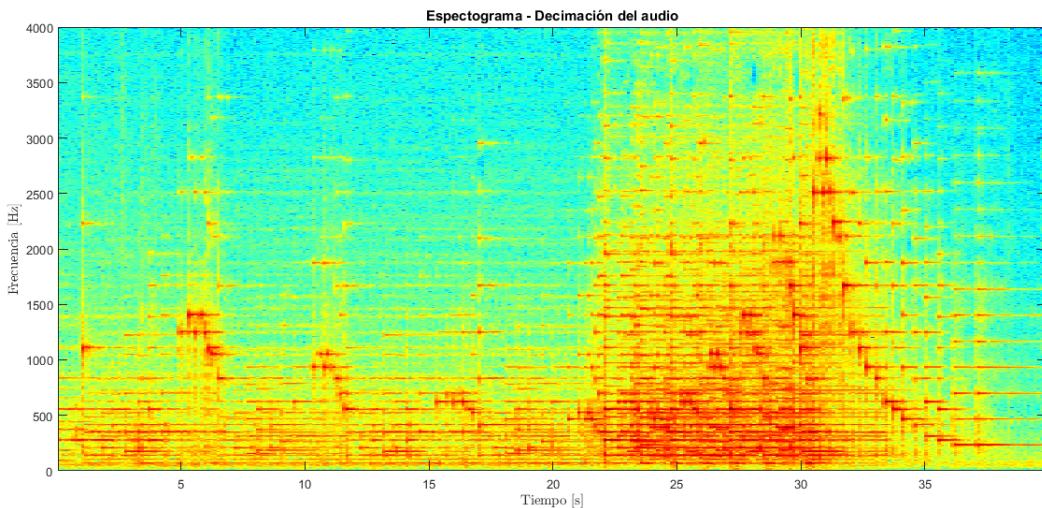


Figura 11: Espectrograma de la señal decimada con frecuencia de muestreo compensada.

5. Ejercicio 5

Desarrolle un programa que a partir del espectrograma de la señal de audio permita reconstruir la señal original. Grafique el error entre la señal original y la reconstruida.

5.1. Implementación final

```

1 % La funcion rec_spec se encarga de reconstruir la senal a partir del
2 % vector S y el overlap utilizado.
3
4 function [output] = rec_spec(S, ventana, noverlap)
5
6 window=ventana';
7
8 % Como el specgram va de 0 a Fs/2 y el ifft va de 0 a Fs, entonces hay que
9 % "rellenar" el S del specgram con el espejo conjugado.
10 s0 = [S; flipud(conj(S(2:end-1,:)))];
11
12 % Se pasa al dominio del tiempo
13 aux=real(ifft(s0));
14
15 % Lo que sale de la ifft es una matriz de vectores, donde estos ultimos son
16 % las ventanas. Por lo tanto hay que concatenar las ventanas para tener la
17 % original.
18 output=aux(1,:)./window;
19
20 l0=size(aux);
21 rows=l0(1);
22 % Se observa que, a partir del overlap presente, se deben excluir los
23 % primeros noverlap elementos de las siguientes ventanas.
24 if(rows>1)
25   for x=1:(rows-1)
26     output=[output aux(x+1,noverlap+1:end)./window(noverlap+1:end)];
27   end
28 end
29
30 end

```

5.2. Comparación

De la Figura 12, se ve que las frecuencias de la reconstruida carecen de la definición de la original. En particular, en la zona de frecuencias medias, la reconstruida presenta más naranja que su contraparte que contiene sectores más azules y rojas (mayor exactitud).

La razón de esta diferencia es la utilización de ventanas. El proceso de seccionar una porción de la señal es equivalente a multiplicarla por un pulso de longitud finita. Por Fourier se sabe que la multiplicación de dos señales en tiempo, resulta en una convolución de sus transformadas en frecuencia. Así, de utilizar un pulso rectangular por ejemplo, se haría la convolución con una curva que no es una delta, alterando la señal (en el ejemplo, una función *sinc*). Por lo tanto, al reconstruirla a partir de su transformada, la señal presenta impresión en el dominio de las frecuencias debido al ventaneo previo.

En particular, tomando como punto de comparación al tiempo 30s, para las frecuencias de 380Hz y 1100Hz la diferencia es de aproximadamente 30% como se ve en la Figura 13.

Al utilizar la ventana de *Hamming*, se minimiza el lóbulo más próximo al fundamental, siendo éste muy angosto. Del mismo modo que con la ventana rectangular, la señal debe ser compensada del efecto de la ventana. Por lo tanto, en las líneas 18 y 26 del código de la función, se divide

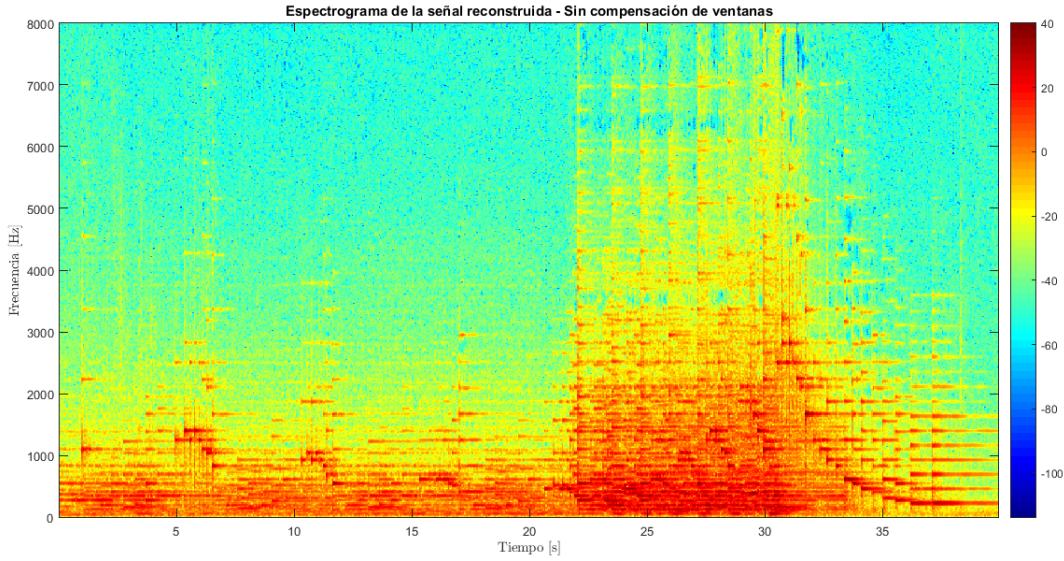


Figura 12: Espectrograma de la señal reconstruida sin compensación de ventanas.

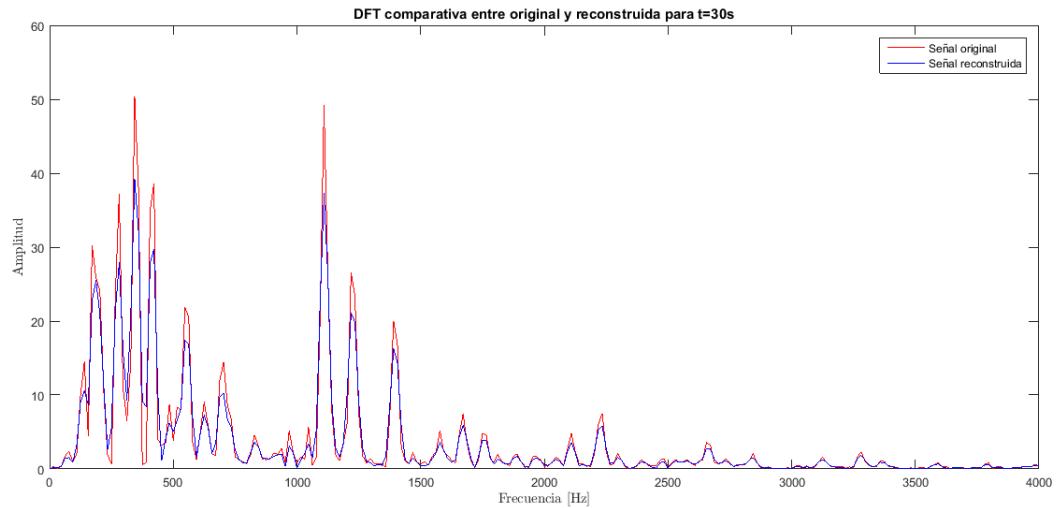


Figura 13: DFT de las señales orginal y reconstruida a tiempo $t = 30$ s

la ventana a concatenar por los coeficientes de la ventana de *Hamming*. Este cambio logra una reconstrucción completa de la señal original. A continuación se presenta el espectrograma de la implementación final:

Como se puede ver en la Figura 14, la reconstrucción mejora. En particular, al hacer la misma comparación en $t = 30$ s (Figura 15) se comprueba que la reconstrucción es precisa.

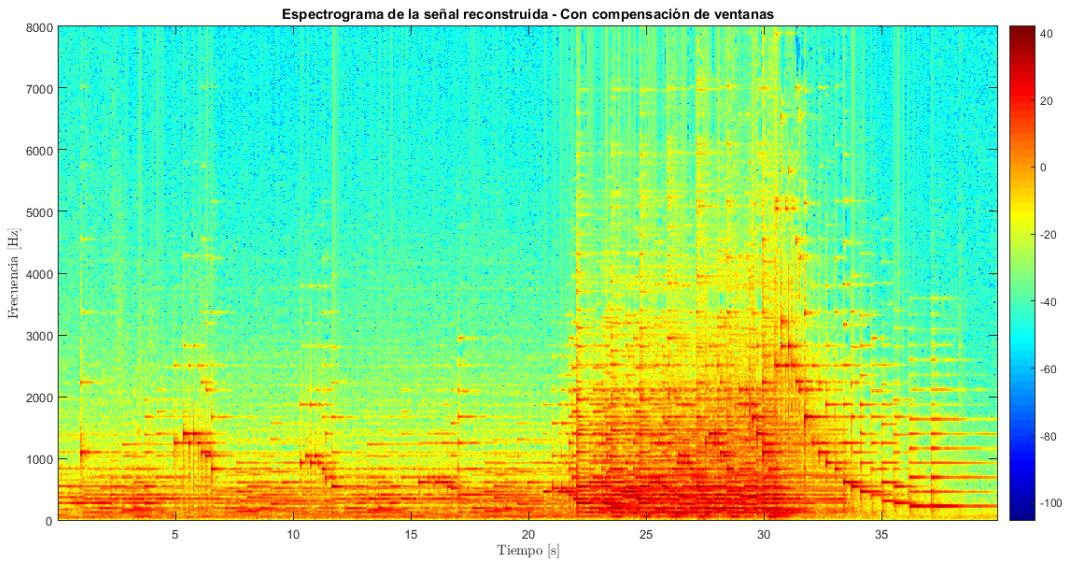


Figura 14: Espectrograma de la señal reconstruida con compensación de ventanas.

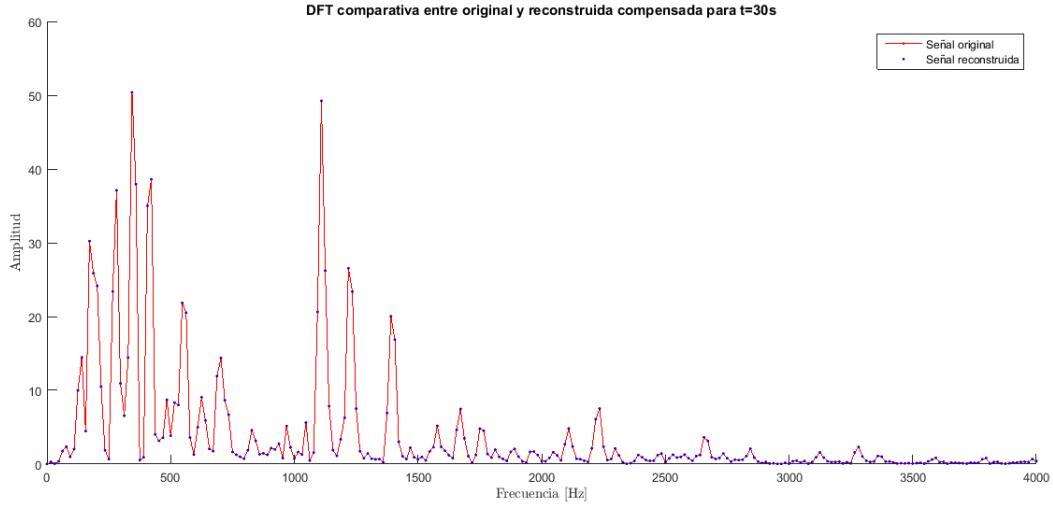


Figura 15: DFT de las señales orginal y reconstruida compensada a tiempo $t = 30$ s

6. Ejercicio 6

Suponiendo que utilicen la función `specgram`, cada columna contiene información del espectro para ese segmento de la señal. Con el objeto de duplicar la duración de la secuencia original interpole una columna cada dos utilizando la información de las filas. Luego utilizando el programa desarrollado en el punto anterior encuentre la nueva señal de audio que debería durar el doble de tiempo. Grafique su espectrograma y compárela con el original.

6.1. Implementación

Para la interpolación, se utilizaron los mismos pasos del Ejercicio 3 con la excepción de que se interpola la matriz resultante del `specgram()`. Luego, se reconstruye con la función `rec_spec()` del Ejercicio 5.

6.2. Comparación de espectrogramas y modificación de la implementación

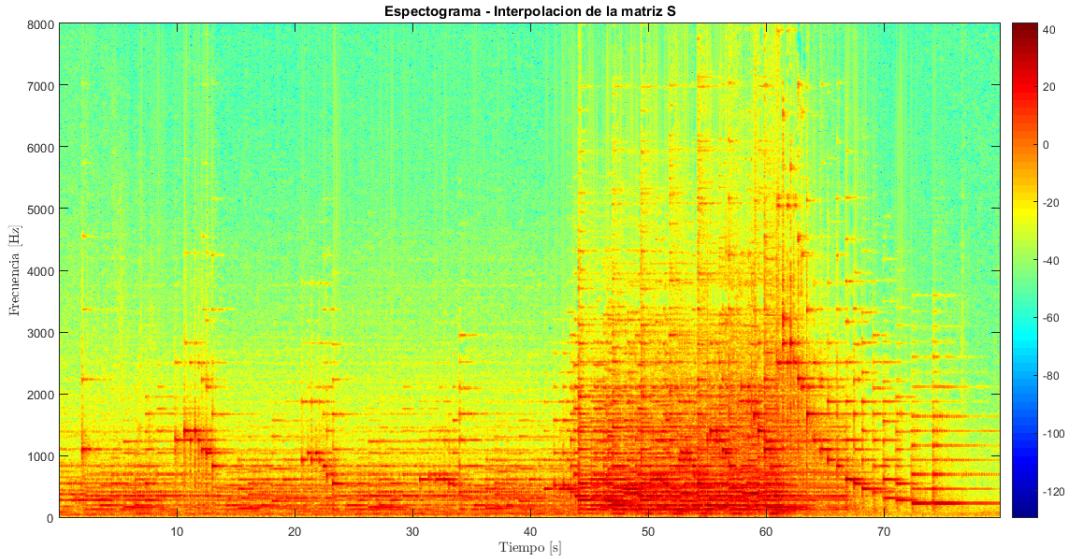


Figura 16: Espectrograma de la señal resultante de la interpolación de las columnas de S.

Como se ve en la Figura 16, la interpolación de las columnas del espectrograma no altera las frecuencias de la señal. Este resultado tiene sentido, dado que al interpolar la señal con sus componentes en frecuencia determinadas, dichas frecuencias se mantienen constantes logrando que la señal original duplique su longitud. Al igual que en el Ejercicio 3, la diferencia de colores se debe a la escala que utiliza el `specgram()` para graficar.

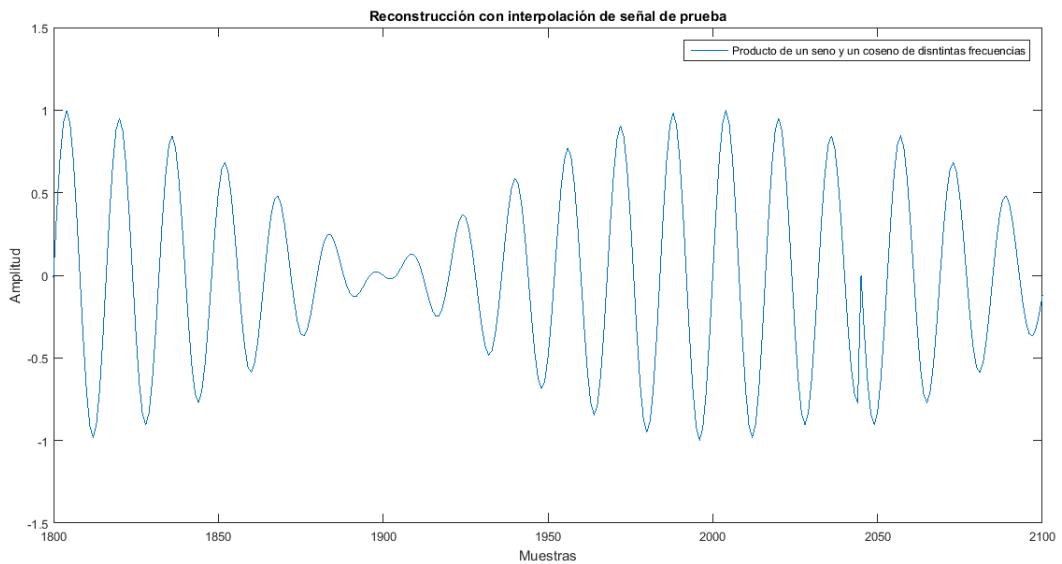


Figura 17: Discontinuidad de la señal al expandir las columnas del espectrograma y volver a tiempo.

Sin embargo, al escuchar dicho audio se encuentran ruidos impropios de la señal original. Haciendo énfasis en la señal temporal, se ve que la unión de las ventanas no es suave (Figura 17). La

razón de este comportamiento indeseado es la interpolación de las columnas. Las nuevas columnas se transforman en tiempo como ventanas nuevas cuyos puntos iniciales y finales pueden diferir de los que se concatenarán después.

Es por eso que se propone expandir las columnas con la columna previa. Así se duplicaría el largo del audio concatenando ventanas con si mismas (y luego la última concatenaría suavemente con la siguiente ventana). Ésto mejora auditivamente la señal, dando un efecto de duración más evidente en vez del eco generado por la primer implementación. A pesar de esto, la curva sigue siendo discontinua persistiendo el ruido.

Se propone por lo tanto pedir que la unión sea continua y que la derivada de la misma también lo sea. Dicha implementación se presenta a continuación con su gráfico en la Figura 18:

```

1  for x=1:(rows-1)
2      concat=aux(x+1,noverlap+1:end)./window(noverlap+1:end);
3
4      for y=1:(length(concat)-1)
5          checkeo1=(concat(y)-output(end))/output(end);
6
7          if (-0.1<=checkeo1 && checkeo1<=0.1)
8              checkeo2=(concat(y+1)-concat(y))/(output(end)-output(end-1));
9
10         if (0.9<=checkeo2 && checkeo2<=1.1)
11             concat=[concat(y:end)];
12             break;
13         end
14     end
15 end
16 output=[output concat];
17 end

```

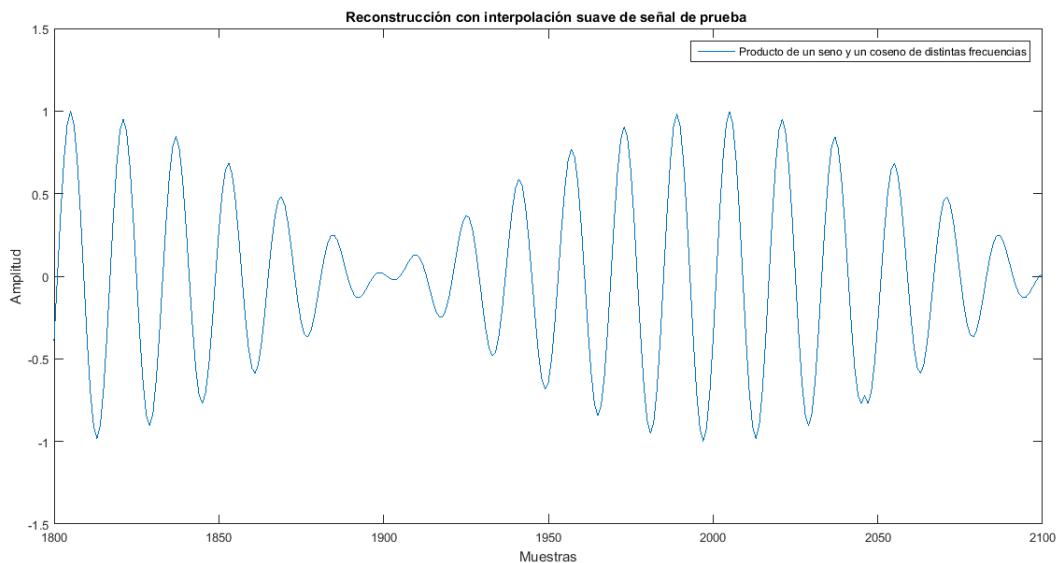


Figura 18: Mejora en la discontinuidad de la reconstrucción de la señal.

Como se puede ver en la Figura 18, la continuidad de la señal mejora considerablemente. Analizando diferentes uniones de ventanas de la señal (para comprobar la generalidad del método) se encontraron problemas como el de la Figura 19. En dicho gráfico se ve que la señal tiene cierta suavidad en la unión, pero la senoide envolvente cambia de signo su pendiente. Dado que el método es básico, no se pretende obtener una señal perfecta, por lo tanto se pasa por alto este error.

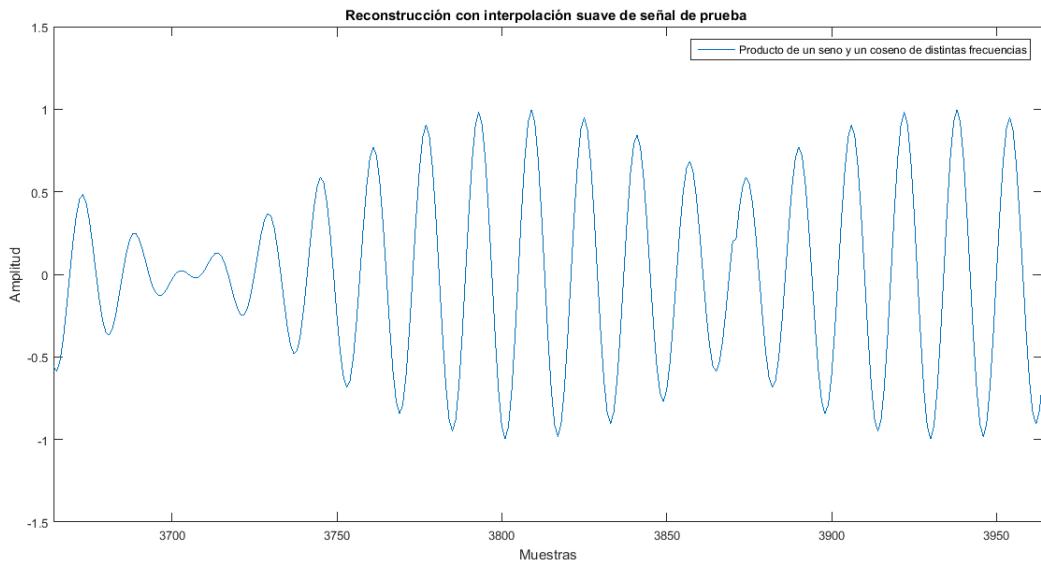


Figura 19: Problema al concatenar ventanas.

Al utilizar el algoritmo descripto por el espectrograma de `audio.wav`, mejora considerablemente la calidad del nuevo audio. Sin embargo, por los problemas de suavidad y cambio de pendientes, se encuentran momentos más ruidosos que otros. Puede deberse también a que el error en ciertas zonas es imperceptible para un oido no entrenado y en otros lo es. De todos modos, la nueva implementación mejora considerablemente al audio.

7. Ejercicio 7

Ahora se busca reducir la duración del audio a la mitad para lo cual se debe decimar el espectrograma por columnas. Recuerde que para evitar el aliasing es necesario realizar el adecuado filtrado pasabajos.

7.1. Implementación

Para resolver este ejercicio, se realizó la misma rutina que la del Ejercicio 4 pero, al contar con los coeficientes de la transformada en el vector del `specgram()` no se requiere el pasaje a frecuencia ni el filtrado de dicho ejercicio (por la misma razón del ejercicio 6). Después de ello, se reconstruye con la función del Ejercicio 5.

7.2. Comparación de espectrogramas

Al igual que en el Ejercicio 6, el espectrograma es similar al del Ejercicio 5 exceptuando la longitud del mismo. Utilizando la nueva implementación que suaviza las uniones, el audio contiene los mismos problemas que el Ejercicio anterior.

Merece ser mencionado que no se realizó un filtrado pasabajos de la señal, porque la interpolación es de muestras con `dft` predeterminada previamente. Otra propuesta interesante sería interpretar a las filas del espectrograma como señales individuales. Así se podría realizar la `dft` de dichas señales y aplicar el filtro *antialiasing*. Dado que las señales no fueron muestreadas por *Nyquist* y su posible dependencia con las ventanas y overlap utilizados, aplicar dicho método puede resultar erróneo. Sin embargo es interesante el enfoque de analizar las variaciones de amplitud de una ventana de frecuencias como una señal temporal ordinaria.

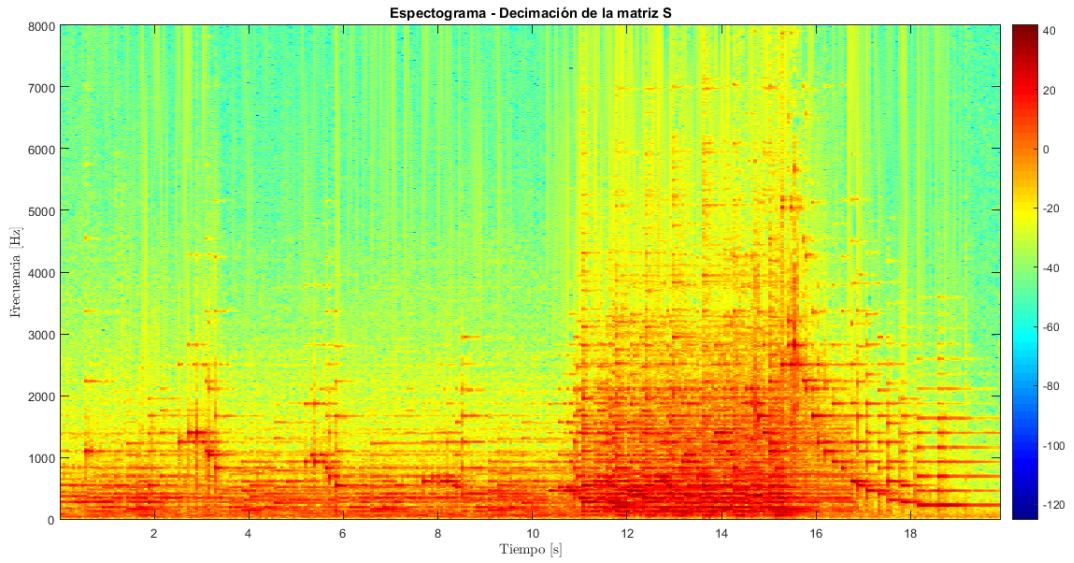


Figura 20: Espectrograma de la señal resultante de la decimación de las columnas de S .

8. Conclusiones

Del análisis y resolución de los Ejercicios 3 y 4 se puede concluir que los métodos de expansión o decimación de muestras no son efectivos para variar la duración de señales. La razón principal es que la variación en la duración genera una variación en el espectro de frecuencias. A pesar de ello, estos métodos son útiles para compresión o expansión del tamaño de los archivos de audio. Si se modifica la frecuencia de muestreo del DAC de modo que la duración y frecuencias sean similares a la original, lo que se obtiene es una variación en el tamaño del archivo. Así, si la señal inicial fue sobremuestreada (es decir el espectro de la misma es de banda mucho más angosta que F_S), una decimación y reconstrucción apropiada mantendría la calidad previa del audio haciendo que éste pese menos.

Por el otro lado, el método propuesto por los Ejercicios 6 y 7 es más apropiado para variar la duración de la señal, manteniendo el espectro original. En estos casos, las diferencias numéricas son pequeñas pero auditivamente apreciables. Para mayor precisión haría falta recurrir a otros métodos como por ejemplo concatenar ventanas en el mismo valor y que éste sea 0 (método disponible en software de edición de audio como *Cubase* y *Pro Tools*).