1 Construcción de filtros digitales mediante ubicación de ceros y polos

El sistema a construir es un ecualizador de tres bandas para señales de audio digital para el cuál se podrá ajustar la ganancia de cada banda independientemente.

Dentro de la carpeta del proyecto **Obligatorio_1** se puede encontrar la carpeta **audios**, con los archivos de audio utilizados en las pruebas, la carpeta **filters** con variables de los filtros guardadas en distintos archivos y los siguientes archivos:

Obligatorio_1.sce Este script es el utilizado para elegir la ganancia de las bandas del ecualizador y genera la gráfica del filtro final.

equalizer.sci Esta función es la que construye los tres filtros y devuelve la suma.

num_den_z.sci Esta función toma los ceros y polos (con sus respectivas ganancias) elegidos para armar el filtro y devuelve el numerador y denominador de la función de transferencia en la variable Z.

z_roots_poles.sci Esta función recibe una frecuencia phi y la transforma al dominio Z, devolviendo el complejo y su conjugado.

audio_processing.sce Este script es el encargado de aplicar los filtros al archivo de audio y genera las gráficas de la transformada de Fourier.

1.1 Construcción de los filtros

En primer lugar se eligieron las frecuencias de corte en la frecuencia adimensionada φ de las bandas pasantes de los filtros:

Pasa bajos 0.17

Pasa banda 0.18 - 0.35

Pasa altos 0.36

Estas bandas fueron elegidas intentando dividir el espectro de frecuencias entre 0 y 0.5 en tres partes iguales. Estas bandas no coinciden con los rangos de frecuencia que habitualmente entran dentro de los bajos, medios y altos, pero esta elección apoya el proceso de construcción de los filtros visualmente.

Teniendo definidas las bandas se utiliza el método de ubicación de ceros y polos para comenzar el armado del filtro. Como primer paso se colocaron polos en las frecuencias de corte deseadas y a partir de ahí se ubicaron otros polos al rededor para acercar el filtro lo más posible a su forma ideal. Los ceros fueron más que nada utilizados para atenuar lo más posible las frecuencias en la banda de rechazo.

Para generar los filtros se utilizó el programa de cálculo Scilab; los resultados se muestran en la figura 1.7.

El proceso de construcción del filtro pasa bajos comienza con la elección de un polo en la frecuencia de corte 0.17 con una ganancia de 0.9 como se muestra en la figura 1.1.

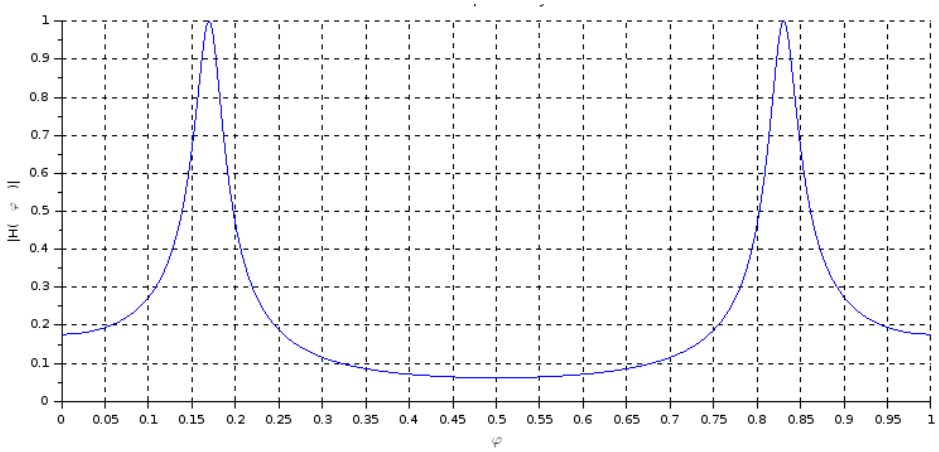

Figura 1.1: Polo único en 0.17 con ganancia 0.9

Luego fue necesario aumentar la ganancia de las frecuencias a la izquierda de 0.17 y para eso se colocaron polos en 0.16 y 0.1, ambos con ganancia 0.9, como se muestra en las figuras 1.2 y 1.3.

Para atenuar las frecuencias de la banda de rechazo se agregó un cero en 0.2, como se muestra en la figura 1.4. Aún pueden verse frecuencias por encima de 0, por lo que se agregaron más ceros obteniendo el resultado de la figura 1.5.

Modificando la ganancia de los polos podemos mejorar la diferencia de alturas que se observa en las figuras anteriores de manera de acercarnos a la forma del filtro ideal. Se llega al resultado de la figura 1.6 ajustando dichas ganancias visualmente.

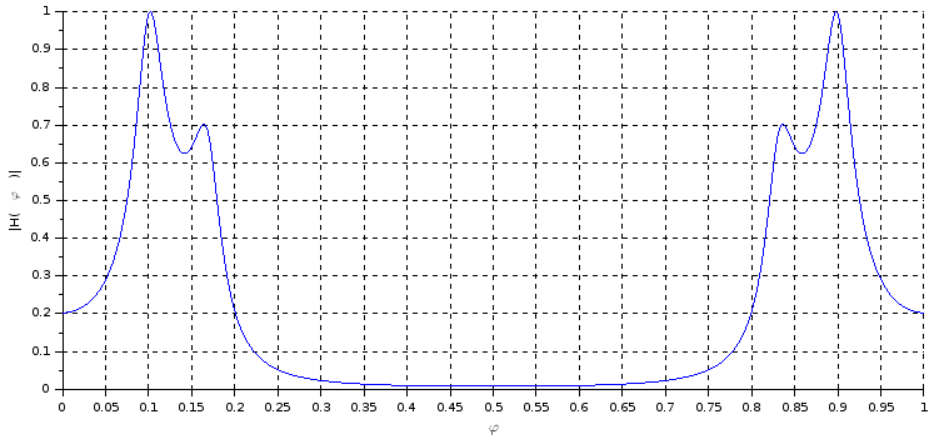

Figura 1.2: Polos en 0.17 y 0.1, ambos con ganancia 0.9

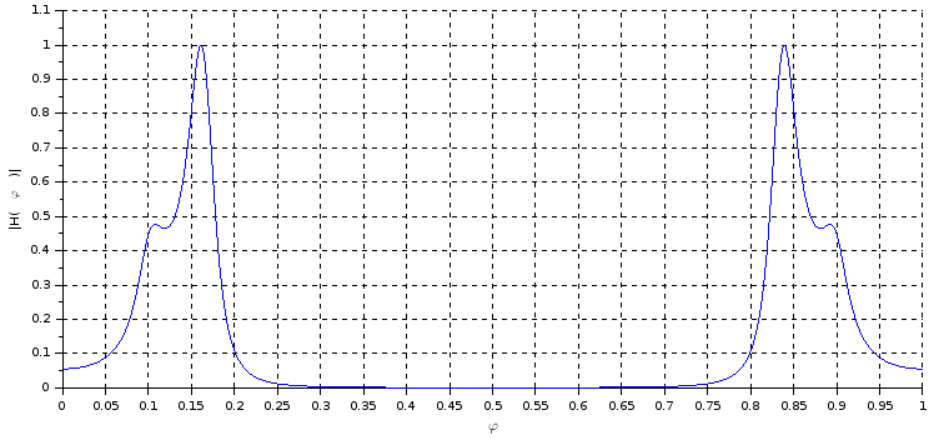

Figura 1.3: Polos en 0.17, 0.16 y 0.1, todos con ganancia 0.9

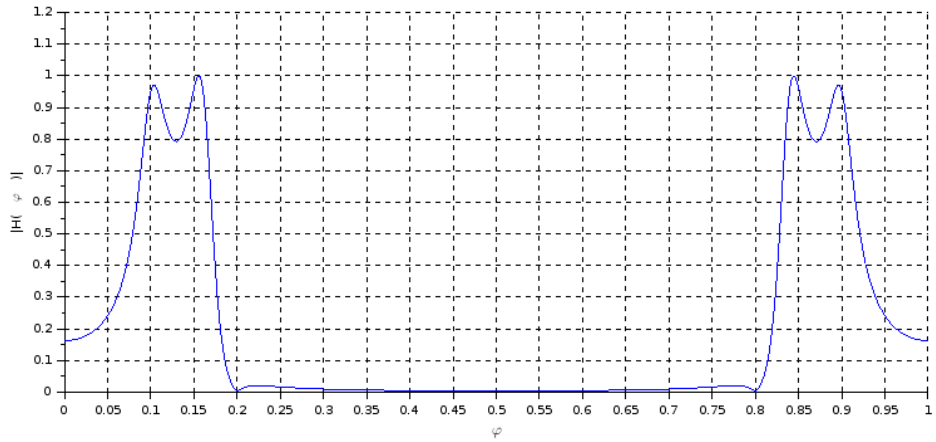

Figura 1.4: Cero en 0.2. Polos en 0.17, 0.16 y 0.1, todos con ganancia 0.9.

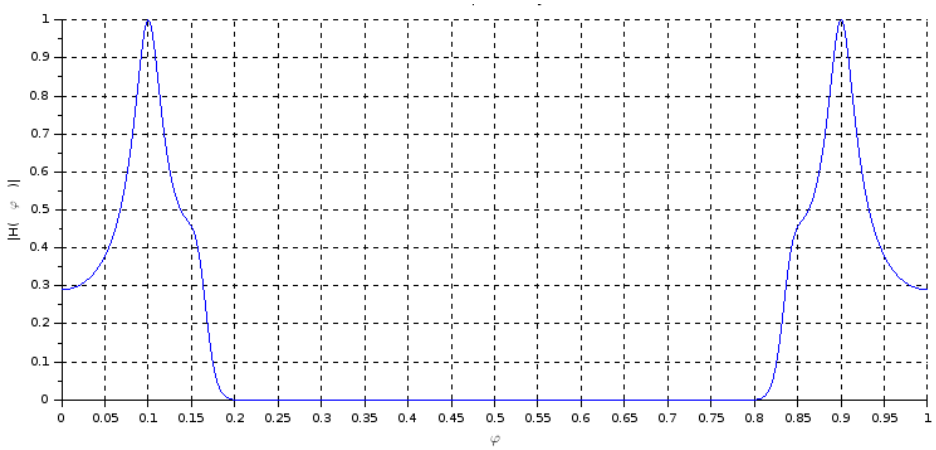

Figura 1.5: Ceros en 0.2, 0.22, 0.3, 0.45. Polos en 0.17, 0.16 y 0.1, todos con ganancia 0.9.

Este mismo procedimiento fue empleado en la construcción del filtro pasa banda y pasa altos, obteniéndose los resultados mostrados en la figura 1.7.

El resultado de sumar estos tres filtros, eligiendo una ganancia igual a 1 para cada banda, es el mostrado en la figura 1.8.

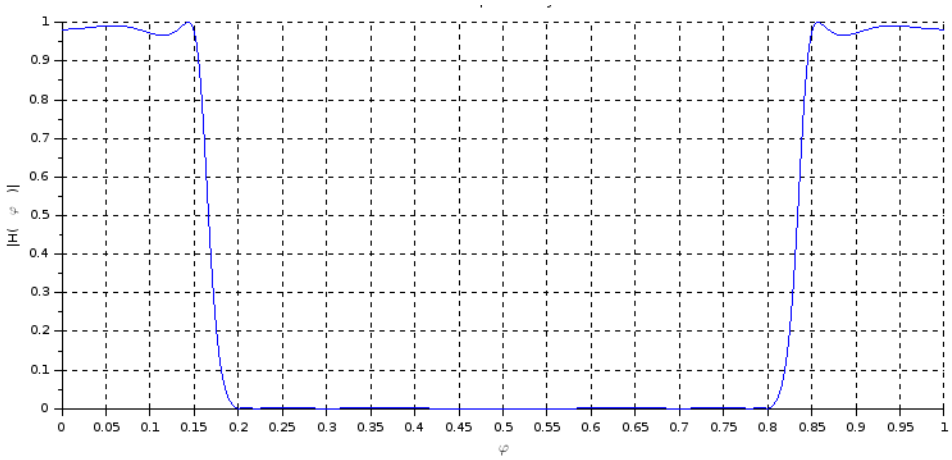

Figura 1.6: Ceros en 0.2, 0.22, 0.3, 0.45. Polo en 0.17 con ganancia 0.83, polo en 0.16 con ganancia 0.9 y polo en 0.1 con ganancia 0.56

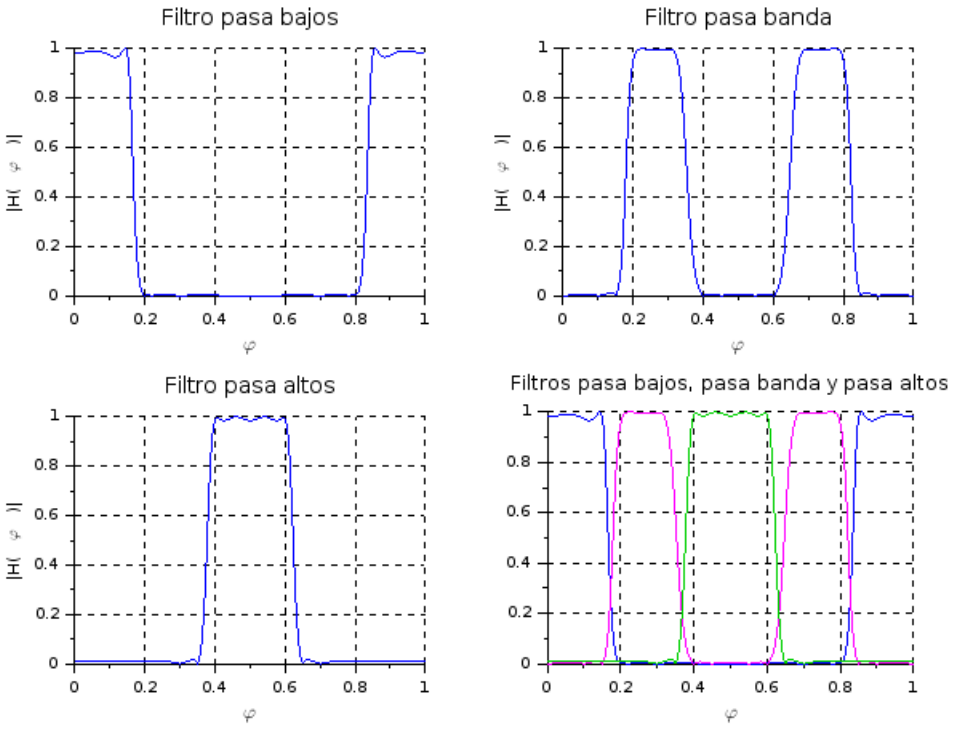

Figura 1.7: Filtros para cada banda.

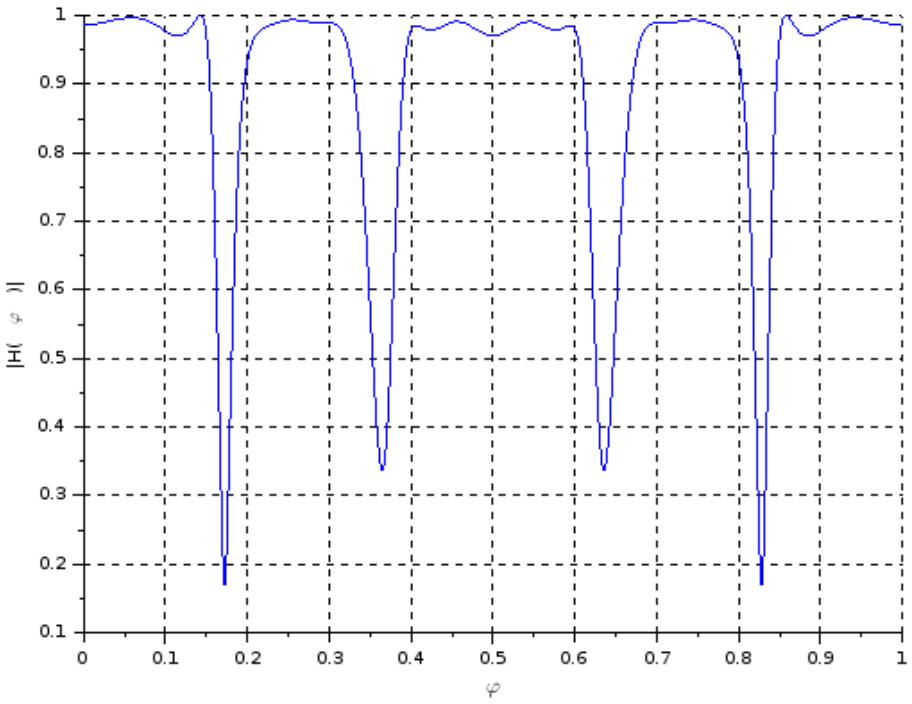

Figura 1.8: Filtro con ganancias: pasa bajos 1, pasa banda 1, pasa altos 1.

1.2 Procesamiento de audio

A grandes razgos el script *audio_processing.sce* lee el audio en formato wav, se queda con un solo canal, aplica los filtros construidos en la parte anterior, normaliza y calcula la respuesta en tiempo discreto. Esta respuesta se guarda también en formato wav para poder escuchar el audio resultante.

Se utilizaron tres audios distintos, dos de ellos son fragmentos de piezas musicales y el tercero es un barrido de frecuencias, que se utilizó para facilitar la visualización del efecto de los filtros sobre el espectro de frecuencias.

Aplicando el filtro de la figura 1.8, que tiene ganancia 1 en las tres bandas, se obtienen los resultados mostrados en las figuras 1.9, 1.10 y 1.11.

Como los filtros no son ideales, cuando se hace la suma las bandas no quedan delimitadas correctamente, por esta razón, se puede ver que por más que la ganancia es 1, hay zonas en las que cae cerca de cero.

Es fácil ver en la figura 1.9 cómo estas zonas afectan al procesamiento del audio, ya que deberíamos ver a la salida lo mismo que a la entrada pero esto no sucede así. Se muestran también los resultados sobre los otros audios pero este efecto no es tan notorio.

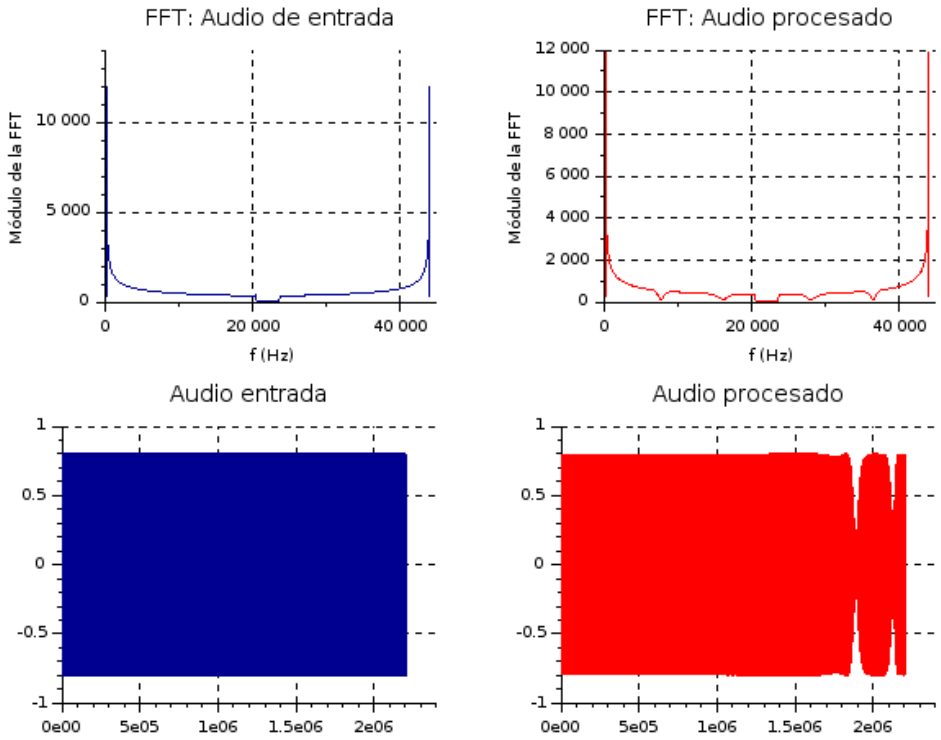

Figura 1.9: Resultado de aplicar el filtro de la figura 1.8 al audio “sweepinv.wav”.

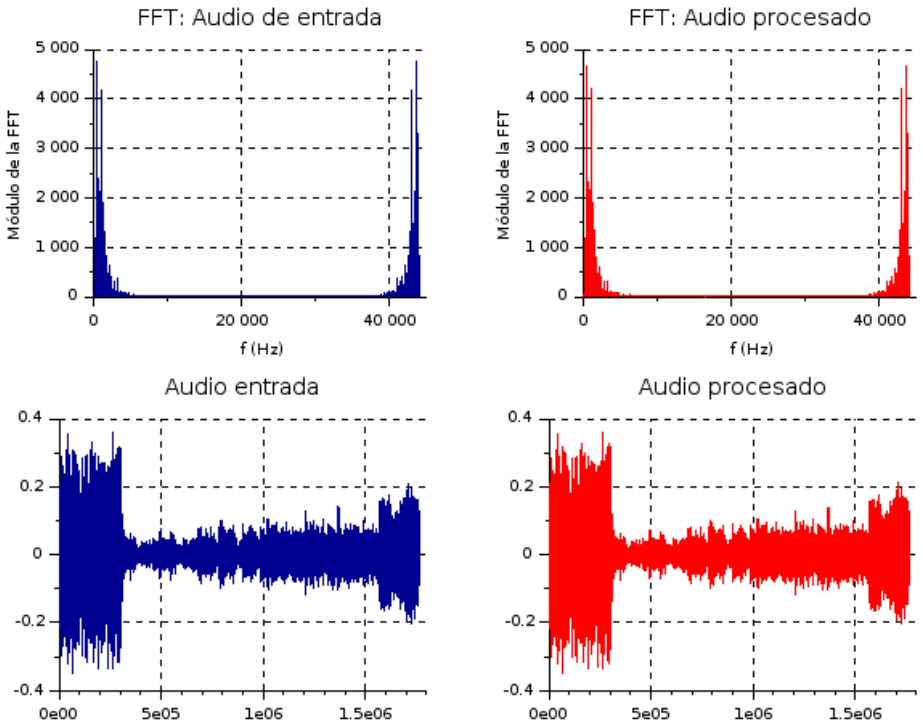

Figura 1.10: Resultado de aplicar el filtro de la figura 1.8 al audio "beth-symph.wav".

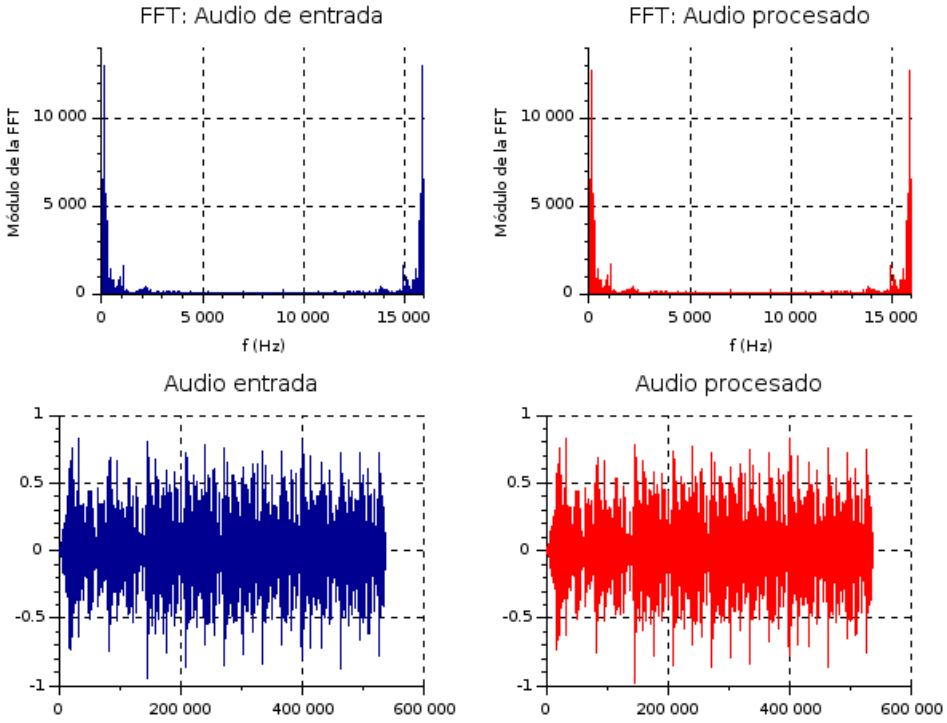

Figura 1.11: Resultado de aplicar el filtro de la figura 1.8 al audio "example.wav".

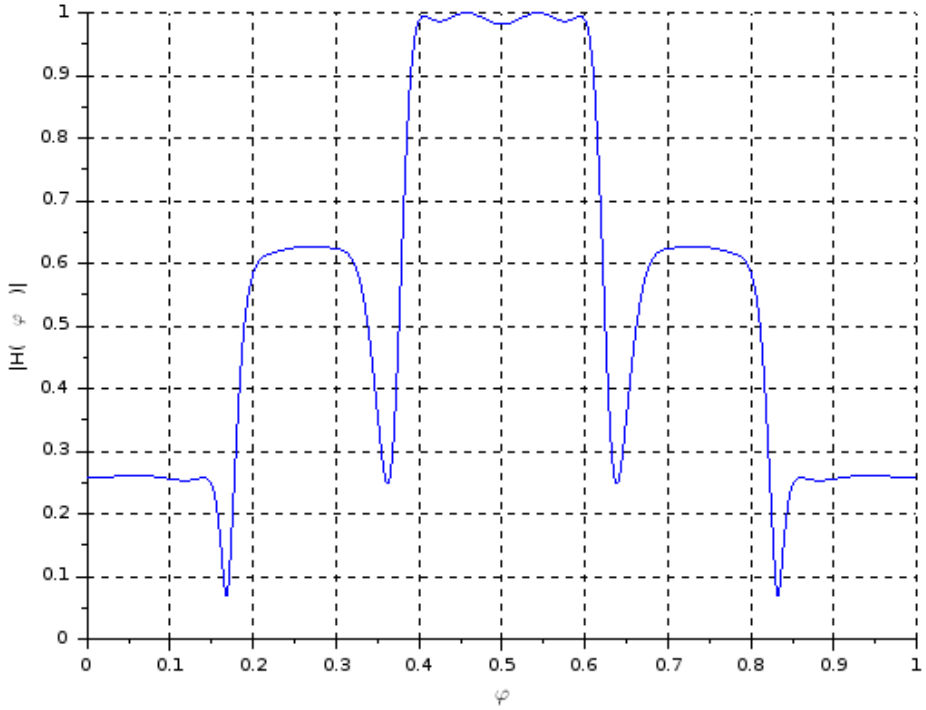

Figura 1.12: Filtro con ganancias: pasa bajos 0.2, pasa banda 0.5, pasa altos 0.8.

Se genera ahora un filtro cuyas ganancias son: 0.2 para los bajos, 0.5 para los medios y 0.8 para los altos.

Las figuras 1.13, 1.14 y 1.15 surgen de aplicar el filtro de la figura 1.12 a los tres audios.

En el procesamiento del audio sweep (1.13) se puede ver claramente el efecto que tiene el filtro sobre la ganancia de las tres bandas.

Es fácil de ver también en la FFT del audio beth-symph, cómo para las frecuencias bajas por ejemplo, podemos ver que el módulo de la FFT disminuye 5 veces en comparación con el del audio original. Esto es de esperar, ya que la ganancia del pasa bajos en este caso es de 0.2 y podemos ver que el audio original va hasta al rededor de 5000.

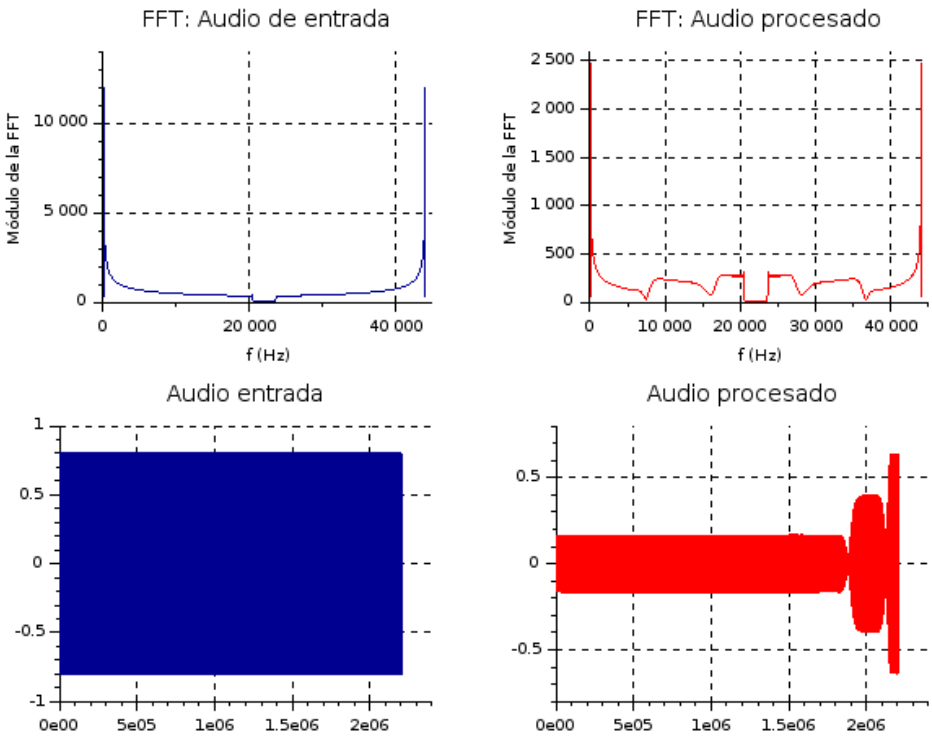

Figura 1.13: Resultado de aplicar el filtro de la figura 1.12 al audio “sweepinv.wav”.

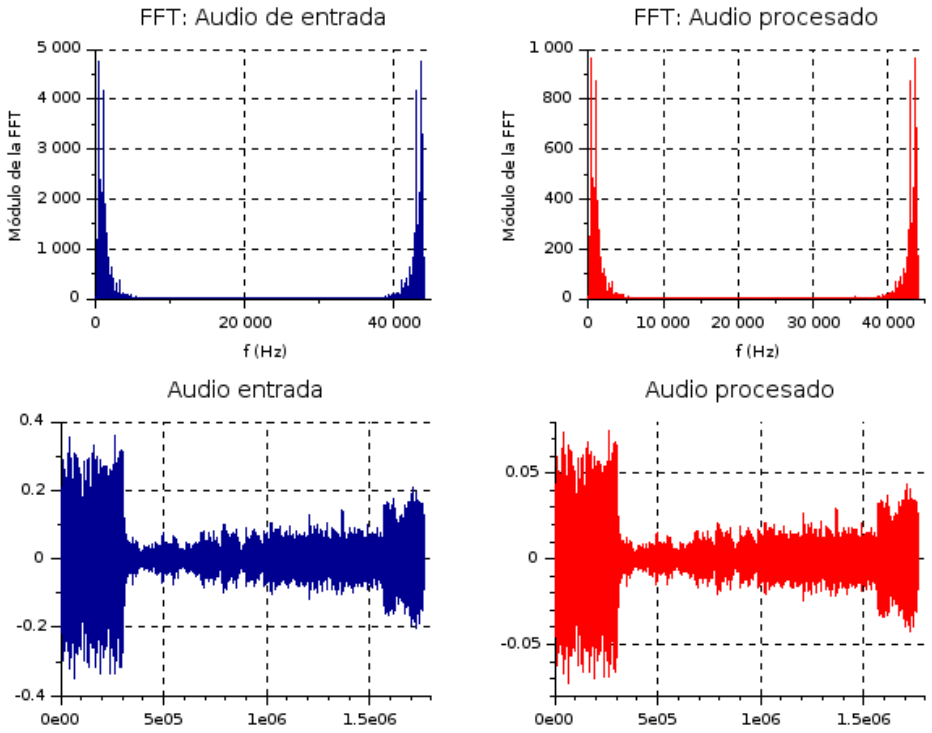

Figura 1.14: Resultado de aplicar el filtro de la figura 1.12 al audio “beth-symph.wav”.

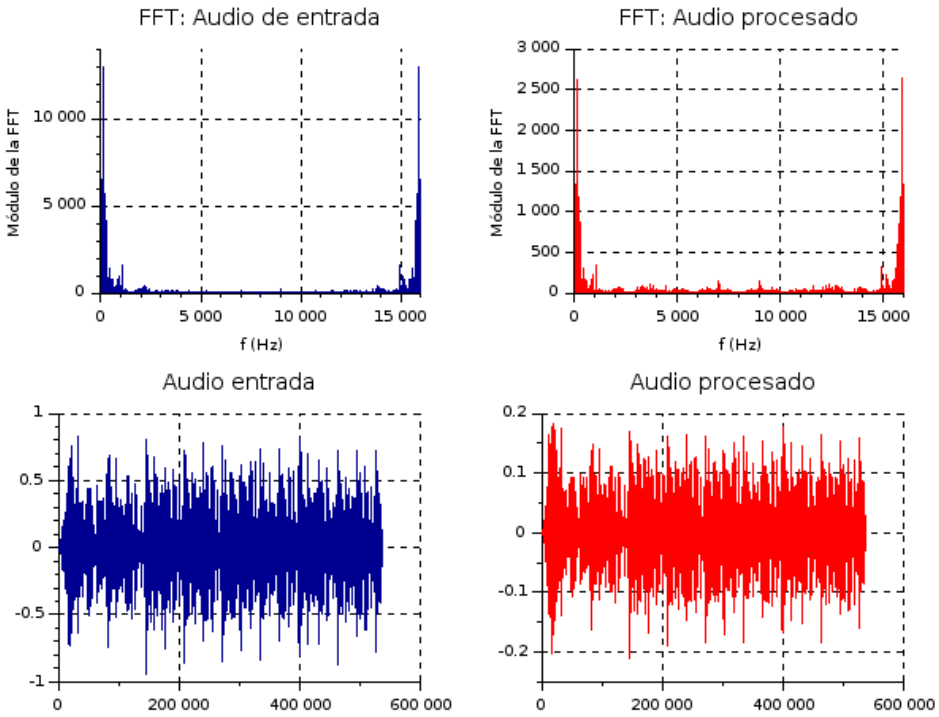

Figura 1.15: Resultado de aplicar el filtro de la figura 1.12 al audio “example.wav”.

En la siguiente prueba se generó un filtro con ganancia 0.1 en las frecuencias bajas, 1 en las medias y 0 en las altas. El resultado se muestra en la figura 1.16.

Cuando se aplica el filtro de la figura 1.16 a los audios se obtienen los resultados mostrados en las figuras 1.17, 1.18 y 1.19.
En este caso podemos ver que, por más que a las frecuencias altas se les da una ganancia de 0, el filtro no es perfecto y presenta una cierta ganancia por encima de 0.

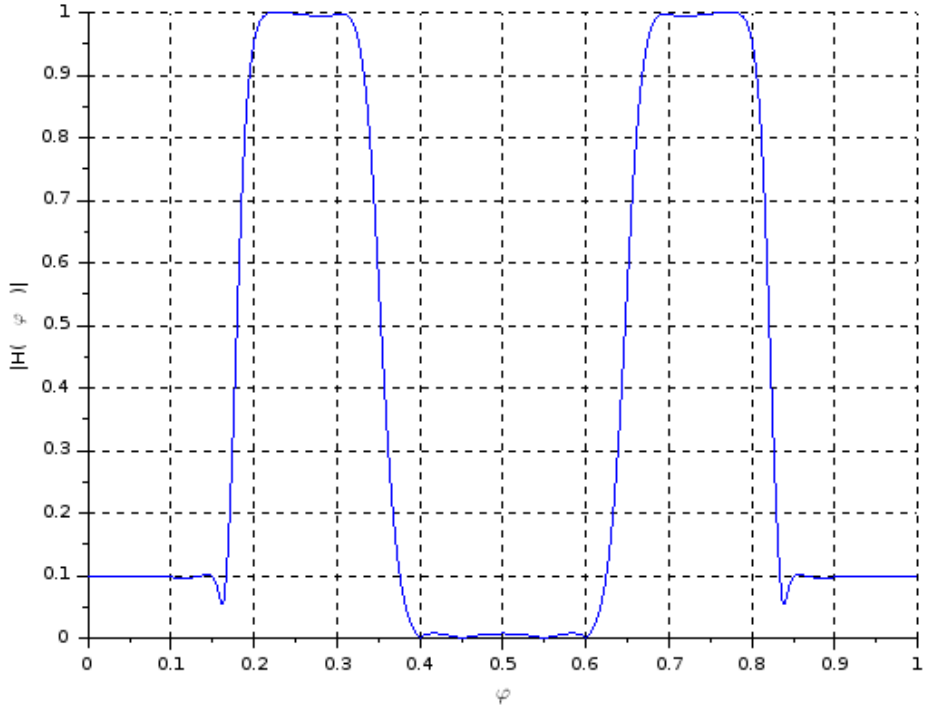

Figura 1.16: Filtro con ganancias: pasa bajos 0.1, pasa banda 1, pasa altos 0.

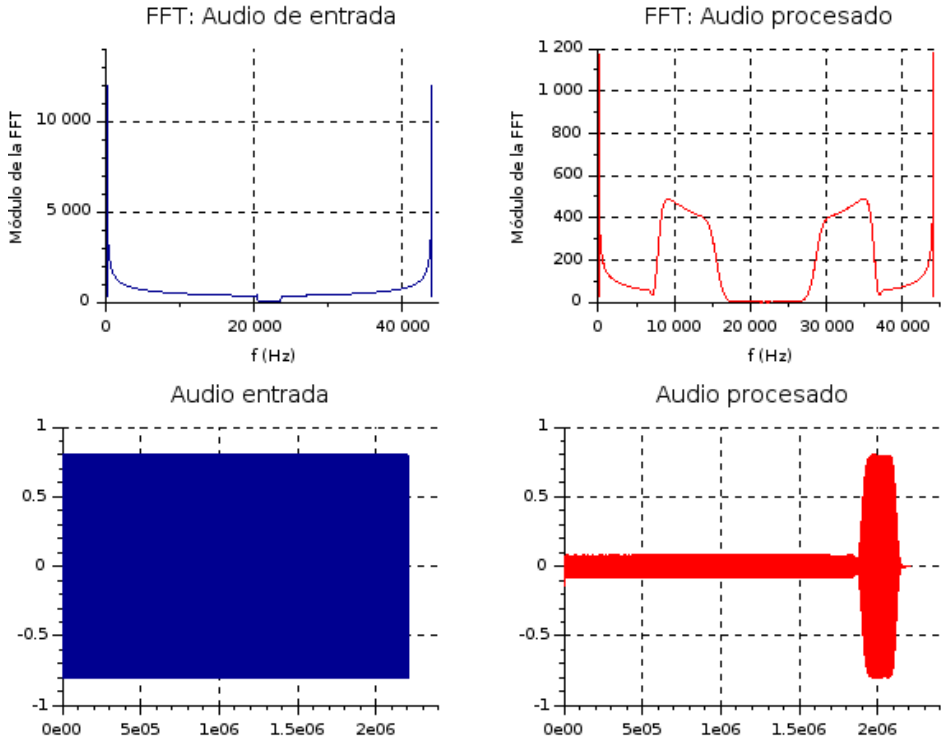

Figura 1.17: Resultado de aplicar el filtro de la figura 1.16 al audio “sweepinv.wav”.

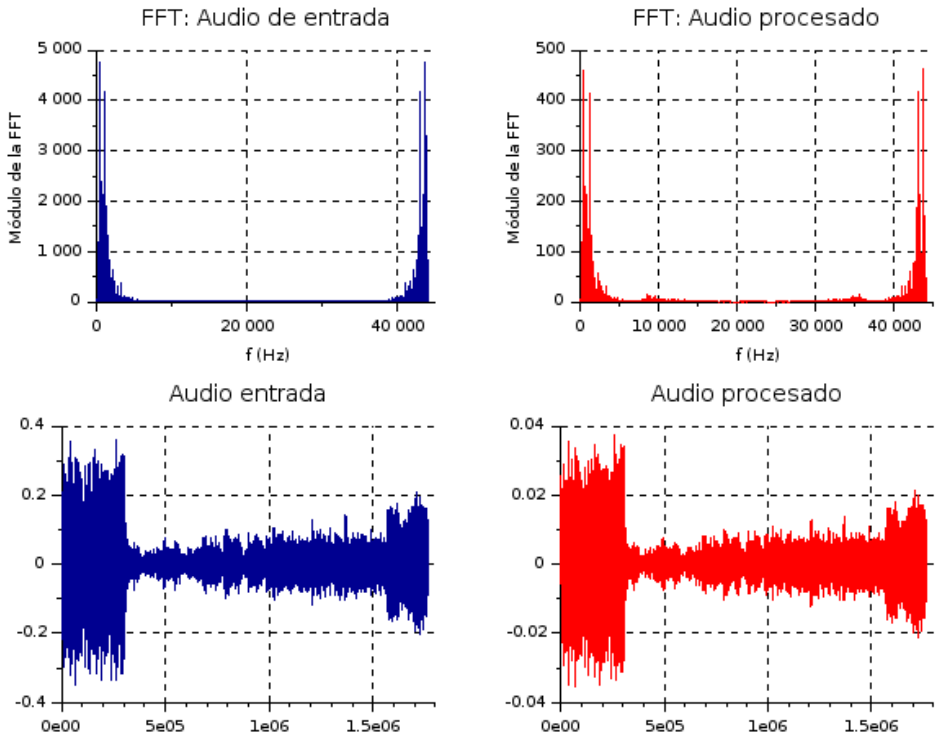

Figura 1.18: Resultado de aplicar el filtro de la figura 1.16 al audio “beth-symph.wav”.

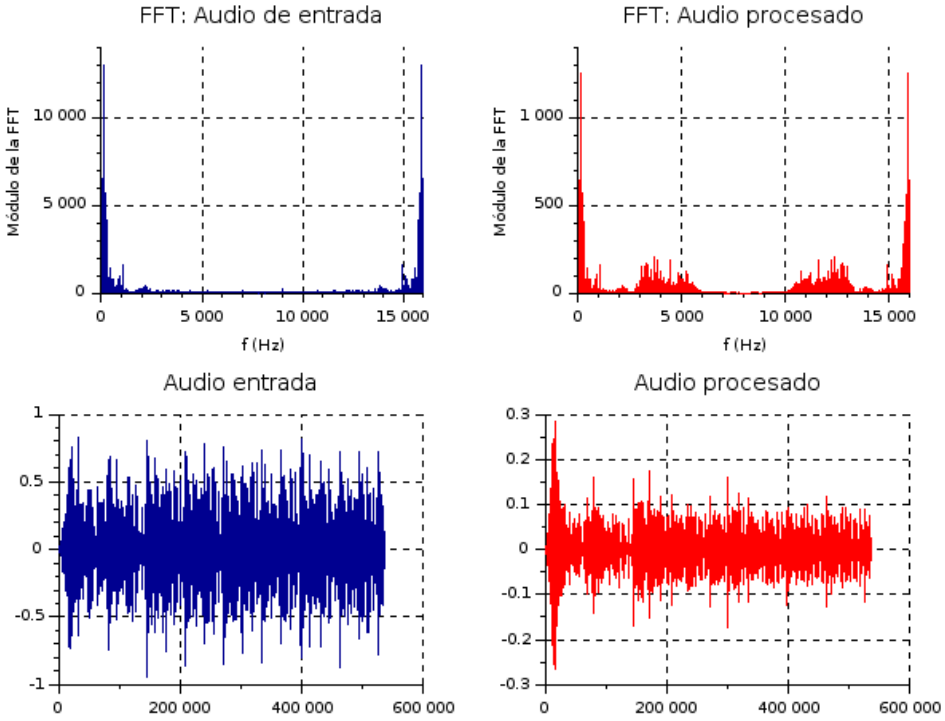

Figura 1.19: Resultado de aplicar el filtro de la figura 1.16 al audio “example.wav”.

2 Filtros digitales FIR de fase lineal

El proyecto elegido para esta parte del obligatorio consiste en construir un codificador y decodificador para señales vocales con el fin de enmascarar la señal y dificultar escuchas no autorizadas.

Dentro de la carpeta del proyecto **Obligatorio_2** se pueden encontrar carpetas con los audios separados en original, codificado y decodificado, una carpeta con los filtros por banda guardados cada uno en un archivo y los siguientes archivos:

Obligatorio_2.sce Este archivo contiene la ruta hacia el archivo de audio que se quiere codificar y decodificar, y el llamado a las funciones que lo procesan.

filtros.sci Este archivo contiene todas las funciones utilizadas para la creación y graficado de los filtros.

audioWav.sci Esta función se encarga de graficar la FFT de un archivo de audio.

cod_decod.sce Contiene las funciones para codificar y decodificar una señal de audio.

2.1 Procedimiento

El proceso de codificación se da de la siguiente manera:

1. Se crean tres filtros de igual ancho de banda para separar la señal de audio en las tres bandas a permutar.
2. Una vez aplicado el filtro se tiene el audio dividido en tres bandas.
3. La permutación se da al modular cada banda.
4. La modulación tiene como resultado dos copias de la banda a desplazar, por lo que es necesario filtrar para guardar solo la copia ubicada en la banda deseada.
5. Teniendo las tres bandas desplazadas se las suma para obtener el resultado final.

El proceso de decodificación sigue los mismos pasos cambiando simplemente la frecuencia a la cual se modula para volver a ubicar las bandas en su posición original.

2.2 Elección de las bandas

Las señales vocales se caracterizan por contener su información dentro del rango de los 300 a 4000 Hertz. Tomando de 0 a 4000 Hertz, por simplicidad, como el ancho total de la señal se eligió dividirlo en tres partes iguales generando tres zonas de igual ancho de banda, en este caso, 1333 Hertz. De esta manera se facilita el desplazamiento de las bandas y esto permite, además, reutilizar los filtros de la separación en bandas para el filtrado luego de la modulación. De haber elegido bandas de tamaños distintos hubiera sido necesario crear nuevos filtros para procesar la señal modulada utilizándose innecesariamente más recursos de cálculo.

Se consideran como frecuencias altas aquellas frecuencias que estén en el rango de los 1600 a 20000 Hertz y las bandas definidas en este problema solo alcanzan los 4000 Hertz, por lo tanto, los filtros que definirán estas bandas serán construidos utilizando un pasabajos con frecuencia de corte 1333 Hz y dos pasa banda ubicados entre 1333 Hz - 2666 Hz y 2667 Hz - 4000 Hz.

Estos filtros serán construidos utilizando una ventana de Kaiser ya que con ésta se obtiene un mayor aprovechamiento de los rangos de error permitidos.

2.3 Construcción de los filtros

Habiendo elegido los intervalos de frecuencia que ocupará cada banda, quedan determinadas las frecuencias φ_s, φ_{s1} y φ_{s2} de cada filtro.

La elección del $\Delta\varphi$ en un principio fue de 10Hz para que la zona de transición fuera corta pero haciendo esto se observó que el archivo de audio luego de ser codificado aumentaba su tamaño en más del doble de kilo bytes y luego al decodificarlo volvía a aumentar. Además el archivo, si bien se decodificaba correctamente y era posible entender el audio, contenía un intervalo de silencio al principio y al final del mismo.

Esto se debía a que para un $\Delta\varphi$ tan pequeño el N es muy grande y la cantidad de ceros que se agregan en la convolución del audio con el filtro es mucho mayor.

Visto esto se eligió entonces un $\Delta\varphi$ de 200 Hz obteniendo un mejor resultado aumentando apenas unos kilo bytes el tamaño del archivo final y sin silencios perceptibles.

2.4 Codificador

La función codificar del archivo *cod_decod.sce* se encarga de crear los filtros para cada banda utilizando los parámetros comentados en la sección anterior. Estos filtros se guardan en distintos archivos junto con sus frecuencias para ser utilizados mas adelante en la función decodificar.

Luego de creados los filtros se separa el audio en bandas, se modulan, se vuelven a filtrar y se obtiene el archivo de audio codificado al sumar estos tres resultados. Las imágenes 2.1 a 2.6 ilustran el proceso de codificación.

En la figura 2.1 se muestra la FFT del audio original. La figura 2.3 muestra el resultado de dividir este audio utilizando las bandas definidas por los filtros de la figura 2.2.

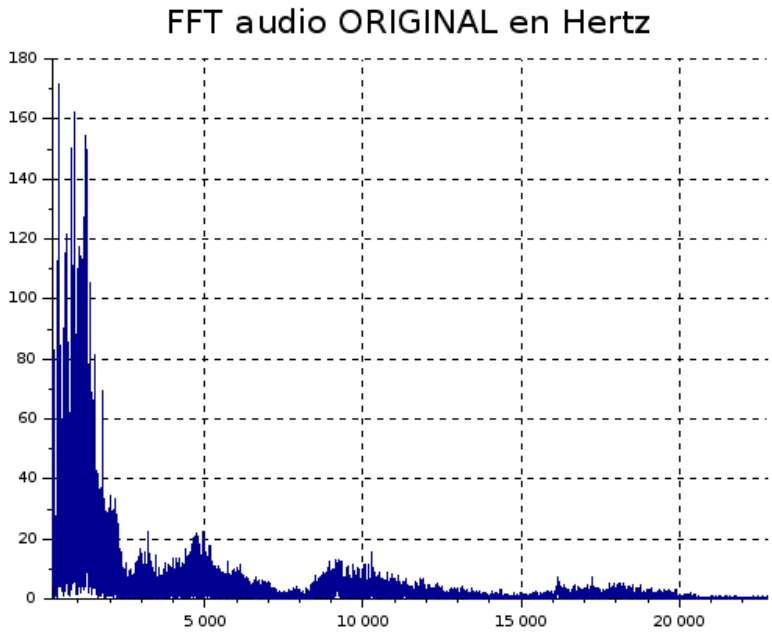

Figura 2.1: Audio “preamble10.wav”.

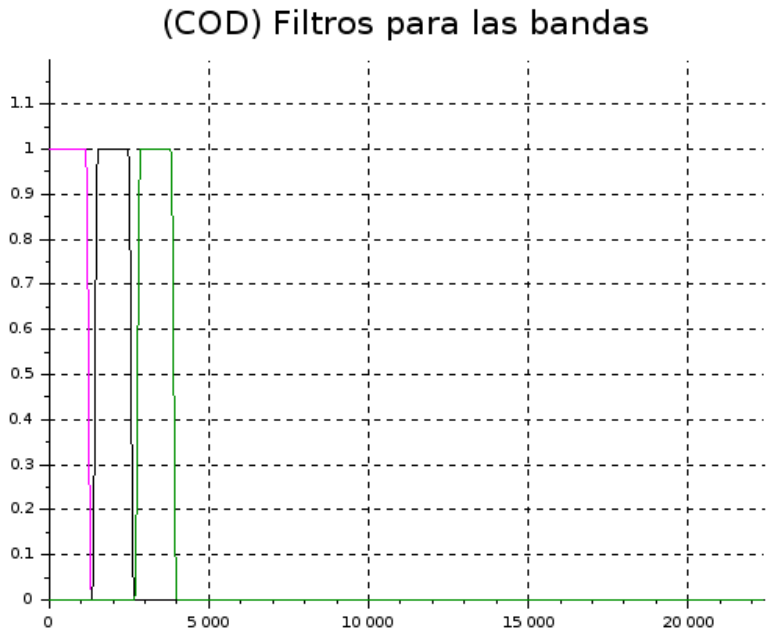

Figura 2.2: Filtros para separación en bandas.

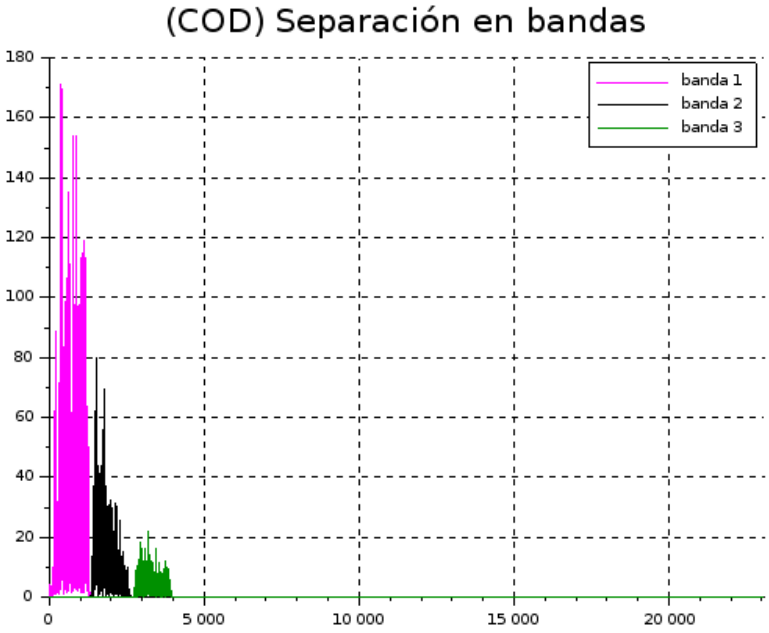

Figura 2.3: Audio “preamble10.wav” separado en bandas.

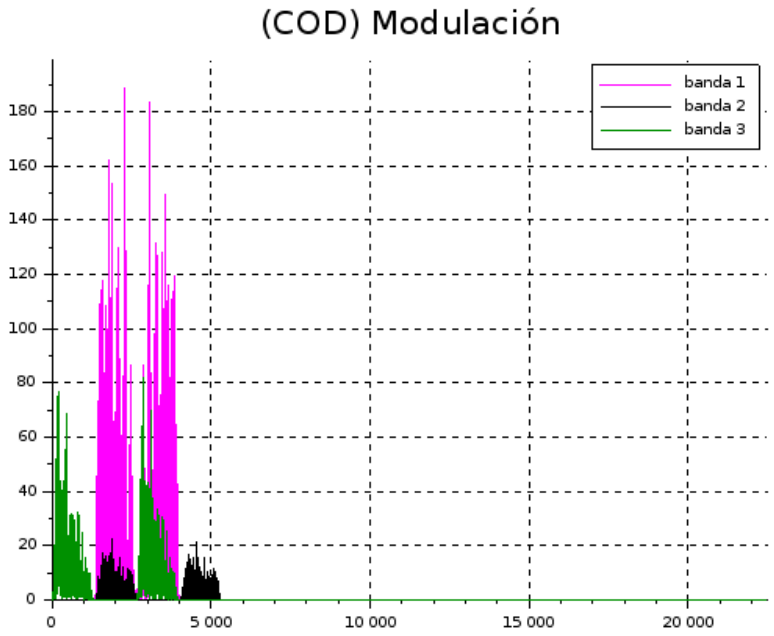

Figura 2.4: Audio “preamble10.wav” modulado.

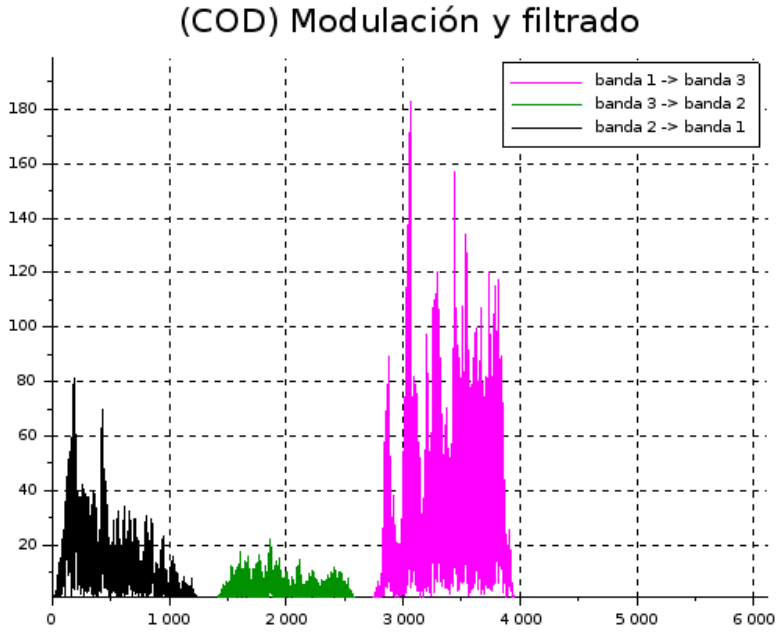

Figura 2.5: Audio “preamble10.wav” modulado y filtrado.

La figura 2.4 muestra el resultado de la modulación de estas bandas. Se puede observar que las copias no quedan simétricas como uno esperaría pero esto se debe a que la frecuencia de modulación utilizada mueve las señales lo suficiente como para que las copias "den la vuelta". A los efectos del proceso de codificación esto no es importante ya que las copias que vemos al revéz se irán luego de filtrar, como podemos verlo en la figura 2.5.

El resultado de la codificación puede verse entonces en la figura 2.6. Podemos observar claramente el movimiento de las bandas manteniendo sus formas originales.

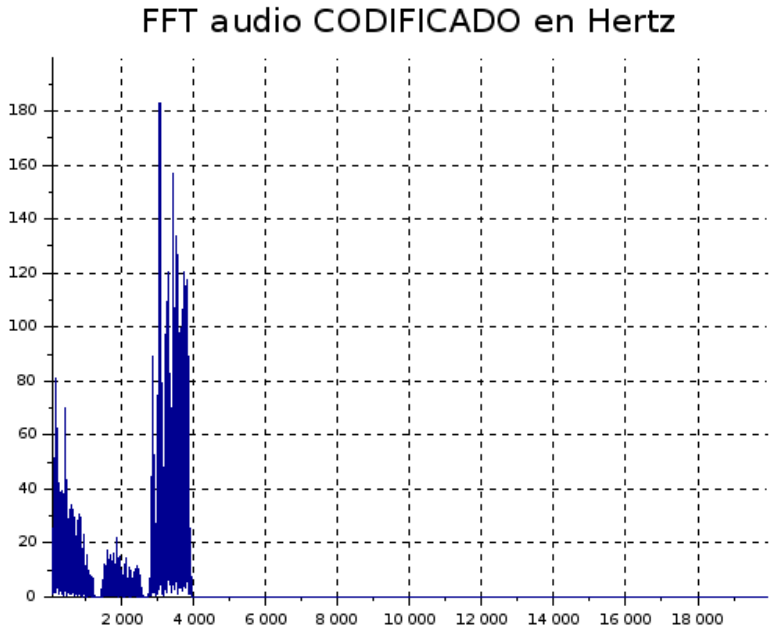

Figura 2.6: Audio "preamble10.wav" codificado.

2.5 Decodificador

El orden en el que se decidió permutar las bandas permite decodificar la señal simplemente aplicando la función de codificación dos veces consecutivas. El resultado de hacer esto o de utilizar la función de decodificar es exactamente el mismo y ambos pueden probarse en el código de Scilab.

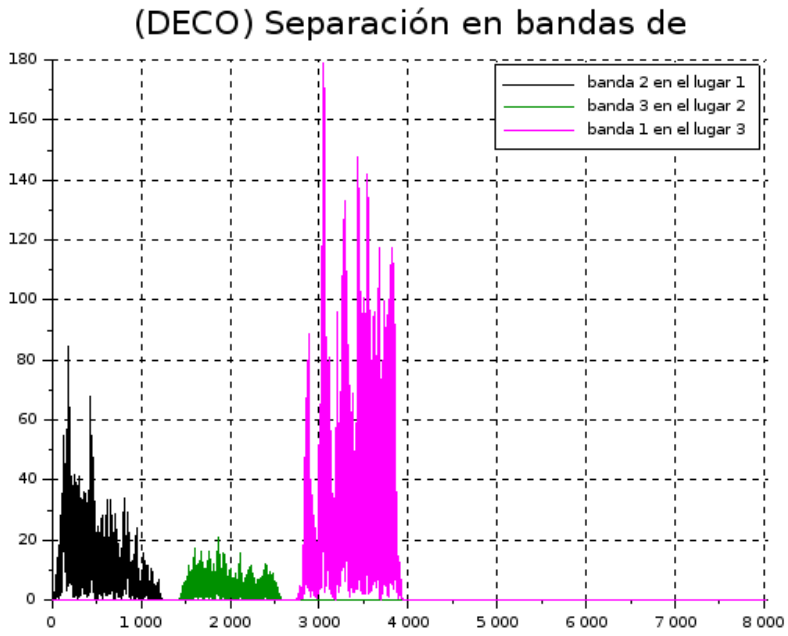

Figura 2.7: Audio “preamble10.wav” codificado separado en bandas.

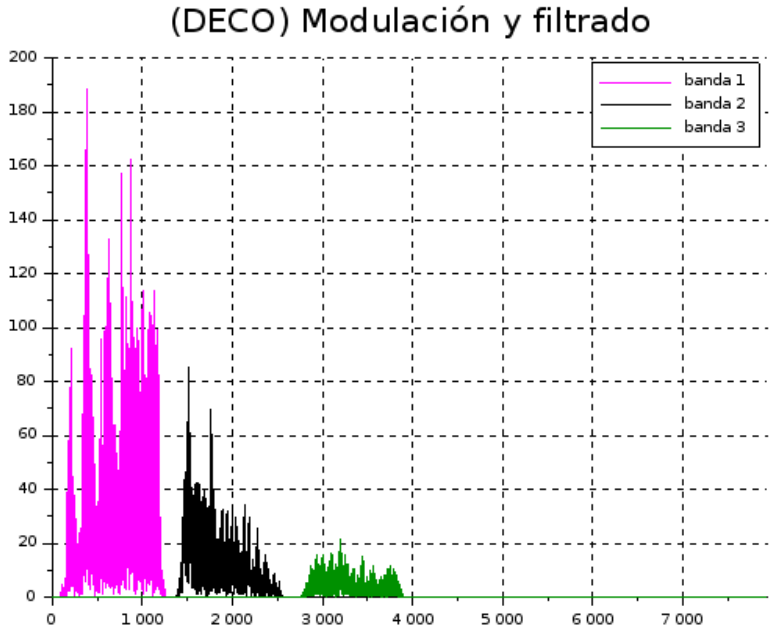

Figura 2.8: Audio “preamble10.wav” modulado y filtrado.

La figura 2.7 muestra como llega el audio codificado manteniendo los colores de las bandas originales para mayor claridad. Podemos ver entonces en esta figura que la banda 2 está ocupando el lugar de la 1, la banda 3 está en el segundo lugar y la banda 1 se encuentra en el último intervalo. Resta entonces, para recuperar el audio original, volver a permutar estas bandas.

Como se observa en la figura 2.9 la señal original se recupera en el rango de 0 a 4000 Hertz en el que se trabajó. Escuchando los audios se puede comprobar que las frecuencias mayores a 4000 Hertz perdidas no afectan a la claridad del audio.

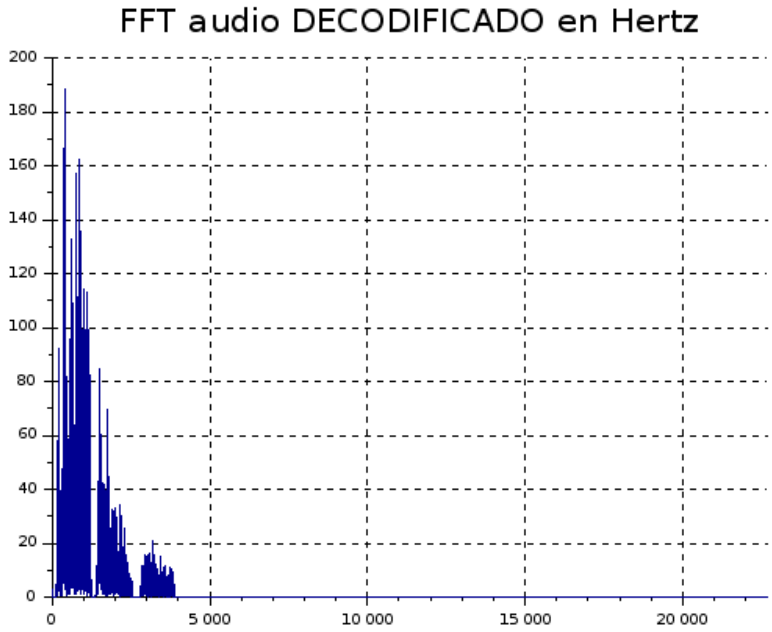

Figura 2.9: Audio “preamble10.wav” decodificado.