Proyecto Efectos de Audio

PDSI (Proyecto Final 3)

***ÍNDICE***

[**Introducción 3**](#_dgdl9mb56scu)

[**Funciones y aplicación del filtro 3**](#_2c306hmw33lz)

[Ecualización a Tres Bandas 3](#_d5urb9f3rfz)

[Eco 4](#_d6bgjcxz615h)

[Reverb Simple 4](#_b6fnujyhwwms)

[Reverb Con filtro Pasa-Todo 5](#_g4f0mpd4s1jp)

[Filtro chorus 6](#_ime1hfa7azw2)

[Programa Principal 7](#_8yz6jhl1z7nz)

[**Conclusiones y Sugerencias 9**](#_btdn5k86s7rp)

### 

### 

### 

### 

### 

### 

### 

### 

### 

### ***Introducción***

En el marco de prácticas previas, hemos diseñado sistemas de tiempo lineal e invariante (LTI) para implementar diversos efectos sobre señales de audio.

Aprovechando este conocimiento,el proyecto aborda el desarrollo de un conjunto de funciones en MATLAB para la creación de efectos de audio simples.

Estas funciones estarán enfocadas en la programación de un eco, dos formas de reverberación, un efecto de coro y un proceso de ecualización a tres bandas (bajas, medias y altas frecuencias). (Se profundizará en estos filtros en los siguientes apartados)

El proyecto se presenta como una oportunidad para aplicar los conocimientos teóricos en el diseño de sistemas LTI y el manejo de herramientas como MATLAB para el procesamiento de señales de audio.

Las respuestas frecuenciales de los filtros (todos menos el de coro) se pueden observar al ejecutar el programa principal.

### ***Funciones y aplicación del filtro***

#### Ecualización a Tres Bandas

function y = ecual3B(gb, gm, ga, f1, f2, fs, x)

% Diseñar los filtros

[b\_low, a\_low] = butter(4, f1/(fs/2), 'low'); % Filtro paso bajo

fiir1=filter(b\_low,a\_low,x);

[b\_band, a\_band] = butter(4, [f1/(fs/2), f2/(fs/2)], 'bandpass'); % Filtro paso banda

fiir2=filter(b\_band,a\_band,x);

[b\_high, a\_high] = butter(4, f2/(fs/2), 'high'); % Filtro paso alto

fiir3=filter(b\_high,a\_high,x);

% Aplicar la ganancia a las partes

y\_bajos = gb .\* fiir1;

y\_pasoBanda = gm .\* fiir2;

y\_altos = ga .\* fiir3;

% Unir las partes

y = y\_bajos + y\_pasoBanda + y\_altos;

end

La función ecual3B se trata de un filtro ecualizador de tres bandas que utiliza filtros Butterworth para realizar el procesamiento de señales.

Esta permite ecualizar un audio tanto los rangos bajos, medios y altos en función de las dos frecuencias de límite para los rangos (f1 y f2) a tratar y a su vez la ganancia a aplicar sobre estos rangos, para ello se separa en tres partes las frecuencias que componen la muestra de audio, se aplica (“multiplica”) la ganancia tras separarlas (mediante el uso de filtros Butterworth, en los cuales varía el tipo de filtro en función de las frecuencias a aislar) y se recompone la señal mediante la suma de las tres partes por la ganancia en cuestión asignada a cada rango.

#### Eco

function y = eco(ret, alfa, x, fs)

% Convertir el retardo de milisegundos a número de muestras

R = round(ret \* fs / 1000);

% Aplicar la función de transferencia H(z) = 1 + alfa\*z^{-R}

y = filter([1, zeros(1, R-1), alfa],1, x);

end

La función eco como su nombre indica se ocupa de aplicar un filtro de tipo eco a través de la función de transferencia , para ello se transforma el retardo de milisegundos a muestras para poder calcular el retardo (ret) a aplicar en el filtro.

Para crear los polinomios a aplicar se crea un array que empieza por uno y al que se le añaden R-1 ceros y alfa al final para equivaler a la función de transferencia original.

Una vez creado el polinomio este se aplica mediante la función filter la cual se encarga de crear la respuesta al filtro la cual se recoge en la variable y que es devuelta como resultado por la función.

#### Reverb Simple

function y = reverb1(ret, alfa, x, fs)

% Convertir el retardo de milisegundos a número de muestras

R = round(ret \* fs / 1000);

% Aplicar la función de transferencia H(z) = 1 / (1 - alfa\*z^{-R})

y = filter(1, [1, zeros(1, R-1), -alfa], x);

end

La función reverb1 se encarga de aplicar un efecto de reverb mediante la aplicación de la función de transferencia la cual realimenta la señal original con la señal retardada (ret), lo que produce un eco repetido.

Al igual que en el filtro de eco se adapta el retardo a muestra y se calcula el array del denominador a través del retardo en muestras y se aplica a la señal original con la función filter.

#### Reverb Con filtro Pasa-Todo

Código:

function y = reverb2(ret, alfa, x, fs)

% ret retardo en milisegundos (debe convertirse en número de muestras R)

% alfa factor de amortiguación

% x vector que contiene la señal de entrada

% fs frecuencia de muestreo

% y vector que contiene la señal de salida

% Convertir el retardo de milisegundos a número de muestras

R = round(ret \* fs / 1000);

% Inicializar los coeficientes del filtro

b = [-alfa, zeros(1, R-1), 1];

a = [1, zeros(1, R-1), -alfa];

% Aplicar el filtro

y = filter(b, a, x);

end

El filtro implementa una reverberación con un filtro "pasa todo" que busca proporcionar un efecto de sonido más natural.

Este filtro de reverberación utiliza un retardo (R) convertido de milisegundos a número de muestras, un factor de amortiguación (alfa) y opera sobre una señal de entrada (x) con una frecuencia de muestreo dada (fs). La salida de la función se almacena la variable "y".

En este filtro se implementa la función de transferencia , que a diferencia de la otra versión del filtro de reverberación no solo se introduce un eco repetido sino que se introduce la parte del denominador de la función de transferencia la cual se encarga de dar “naturalidad” al resultado obtenido.

#### Filtro chorus

Código:

function y = coro(ret, fret, aret, nvoces, x, fs)

% ret: retardo medio de la 2ª voz en milisegundos

% fret: frecuencia a la que varía el retardo de la 2ª voz en Hz

% aret: amplitud de la variación del retardo de la 2ª voz en milisegundos

% nvoces: número de voces

% x: señal de entrada

% fs: frecuencia de muestreo

% Inicializar la señal de salida

y = zeros(size(x));

% Adaptar de ms a ciclos

ret2 = ret/1000 \* fs;

aret2 = aret/1000 \* fs;

% Crear la segunda voz con retardo variable

retardo\_segunda\_voz = ret2 + aret2 \* sin(2 \* pi \* fret/fs \* (1:length(x)));

retardo\_segunda\_voz=sort(retardo\_segunda\_voz);

for i = 1:length(x)

indice\_retardado = round(i - retardo\_segunda\_voz(i));

indice\_retardado = max(1, min(indice\_retardado, length(x)));

y(i) = y(i) + x(indice\_retardado);

end

% Crear voces adicionales

for n = 2:nvoces

% Calcular retardo para voces adicionales (variación en cada iteración)

random = rand;

retardo\_adicional = retardo\_segunda\_voz + round(((15 + 10 \* random) / 1000) \* fs);

retardo\_adicional=sort(retardo\_adicional);

for i = 1:length(x)

% Calcular retardo adicional para cada voz

indice\_retardado = round(i - retardo\_adicional(i));

indice\_retardado = max(1, min(indice\_retardado, length(x)));

y(i) = y(i) + x(indice\_retardado);

end

end

y=y+x; %Añadir la señal original

end

El propósito de la función es implementar un efecto de ¨coro” mediante un retardo variable determinado por una amplitud y frecuencia determinadas.

Para ello se pasa las variables de entrada a las unidades correctas (para milisegundos muestras y para hercios ciclos), y se calcula el retardo añadiendo al retardo medio la parte variable. Esto genera un vector de retardos cuyo valor va variando según sea el valor del retardo añadido, sin embargo, este vector está desordenado para evitar que se produzcan errores a la hora de introducir la segunda muestra retardada se ordena este vector de forma creciente asegurando que si el retardo es pequeño (por ejemplo 1 ms) no se introducirán valores que estén en desorden.

Tras esto se calcular el valor de indice a introducir y se comprueba que sea un valor dentro de los índices de la muestra original, en caso de que sea menor a 1 se introduce 1 como el índice a tomar (normalmente este suele ser silencio con lo cual no afecta a la función), en caso de que sea superior se introduce el valor máximo de la función.

A continuación este proceso se repite para las otras voces añadiendo un retardo aleatorio a los elementos del vector original para la primera versión retardada de entre 15 y 25 ms (dato que al consultar en videos de Youtube acerca de este filtro y como funciona suele ser un consenso del retardo plausible entre la voz original y las retardadas).

Por último una vez montado el vector con las voces retardadas (y), se le añade la voz original (x).

#### Programa Principal

% Parámetros para las funciones de filtro

[x,fs]=audioread("E101.WAV");

% Parámetros para las funciones de filtro ecual3B

gb = 2; % Ganancia para bajos

gm = 0.5; % Ganancia para banda media

ga = 0.7; % Ganancia para altos

f1 = 200; % Frecuencia de corte baja en Hz

f2 = 600; % Frecuencia de corte alta en Hz

% Aplicar el filtro ecual3B

output\_ecual3B = ecual3B(gb, gm, ga, f1, f2, fs, x);

%representar efectos de los filtros

[b\_low, a\_low] = butter(4, f1/(fs/2), 'low'); % Filtro paso bajo

[b\_band, a\_band] = butter(4, [f1/(fs/2), f2/(fs/2)], 'bandpass'); % Filtro paso banda

[b\_high, a\_high] = butter(4, f2/(fs/2), 'high'); % Filtro paso alto

figure, freqz(b\_low,a\_low),title("efecto del filtro sobre frecuencias bajas");

figure, freqz(b\_band,a\_band),title("efecto del filtro sobre frecuencias medias");

figure, freqz(b\_high,a\_high),title("efecto del filtro sobre frecuencias altas");

% Parámetros para las funciones de filtro eco

ret\_eco = 500; % Retardo en milisegundos

alfa\_eco = 0.5; % Factor de amortiguación

% Aplicar el filtro eco

output\_eco = eco(ret\_eco, alfa\_eco, x, fs);

%representar efecto del filtro

R = round(ret\_eco \* fs / 1000);

figure, freqz([1, zeros(1, R-1), alfa\_eco],1),title("efecto del filtro de eco");

% Parámetros para las funciones de filtro reverb1 y reverb2

ret\_reverb = 100; % Retardo en milisegundos

alfa\_reverb = 0.7; % Factor de amortiguación

% Aplicar los filtros reverb1 y reverb2

output\_reverb1 = reverb1(ret\_reverb, alfa\_reverb, x, fs);

%representar efecto del filtro

R = round(ret\_reverb \* fs / 1000);

figure,freqz(1, [1, zeros(1, R-1), -alfa\_reverb]),title("efecto del filtro de reverb simple");

output\_reverb2 = reverb2(ret\_reverb, alfa\_reverb, x, fs);

%representar efecto del filtro

R = round(ret\_reverb \* fs / 1000);

b = [-alfa\_reverb, zeros(1, R-1), 1];

a = [1, zeros(1, R-1), -alfa\_reverb];

figure,freqz(b,a),title("efecto del filtro de reverb con filtro pasa-todo");

% Parámetros para la función de filtro coro

ret\_coro = 20; % Retardo medio de la 2ª voz en milisegundos

fret\_coro = 1; % Frecuencia a la que varía el retardo de la 2ª voz en Hz

aret\_coro = 5; % Amplitud de la variación del retardo de la 2ª voz en milisegundos

nvoces\_coro = 3; % Número de voces

% Aplicar el filtro coro

output\_coro = coro(ret\_coro, fret\_coro, aret\_coro, nvoces\_coro, x, fs);

% Graficar las señales de entrada y salida de cada filtro

t=(0:length(x)-1) / fs;

figure;

subplot(3, 2, 1);

plot(t, x);

title('Señal de entrada');xlabel('Tiempo (s)');

subplot(3, 2, 2);

plot(t, output\_ecual3B);

title('Salida de ecual3B');xlabel('Tiempo (s)');

subplot(3, 2, 3);

plot(t, output\_eco);

title('Salida de eco');xlabel('Tiempo (s)');

subplot(3, 2, 4);

plot(t, output\_reverb1);

title('Salida de reverb1');xlabel('Tiempo (s)');

subplot(3, 2, 5);

plot(t, output\_reverb2);

title('Salida de reverb2');xlabel('Tiempo (s)');

subplot(3, 2, 6);

plot(t, output\_coro);

title('Salida de coro');xlabel('Tiempo (s)');

% Ajustar el diseño de la figura

sgtitle('Pruebas de Filtros de Audio');

%escuhar resultado y original

duracion\_audio = (length(x) / fs)+5;

soundsc(x,fs);

pause(duracion\_audio);

soundsc(output\_ecual3B,fs);

pause(duracion\_audio);

soundsc(output\_eco,fs);

pause(duracion\_audio);

soundsc(output\_reverb1,fs);

pause(duracion\_audio);

soundsc(output\_reverb2,fs);

pause(duracion\_audio);

soundsc(output\_coro,fs);

pause(duracion\_audio);

La función del programa principal es mostrar tanto la respuesta frecuencial de los filtros LTI (todos menos el de coro) y el resultado de todos los filtros sobre la función original.

A su vez reproduce los audios desde el original, hasta sus versiones modificadas, empezando por el original, seguido de la ecualización a tres bandas, el eco, la reverberación simple, la reverberación con filtro “pasa-todo” y acabando con el resultado del filtro de coro.

### ***Conclusiones y Sugerencias***

El proyecto permite una aproximación a un punto de vista más cercano a lo que sería realmente el desarrollo de una aplicación de modificación de audio y filtrado, permitiendo ver la utilidad de los conocimientos adquiridos en la asignatura a lo largo del cuatrimestre en un entorno más cercano a la realidad y a un posible futuro laboral.

Sugerencias

-Algunos ejemplos de como empezar con el filtro de coro se agradecería, algo similar a las funciones transferencia de los anteriores filtros sería de mucha ayuda (no hablamos estrictamente de función de transferencia para el filtro de coro sino una pista de como tratarlo adaptado al caso pedido).