UNIVERSIDAD POLITECNICA DE VALENCIA

ESCUELA POLITECNICA SUPERIOR DE GANDIA

Grado en Ing. Sist. de Telecom., Sonido e Imagen

UNIVERSIDAD
POLITECNICA
DE VALENCIA

ESCUELA POLITECNICA
SUPERIOR DE GANDIA

“Diseño e implementación de un módulo de efectos para señales de audio digital empleando Matlab”

TRABAJO FINAL DE GRADO

Autor/a:
Mikel Etxebeste Ansa

Tutor/a:
Amparo Girona

GANDIA, 2016

RESUMEN

En la industria musical o audiovisual es habitual aplicar distintos efectos a las señales de audio. Esto se puede hacer para adecuar la señal al canal de transmisión o por razones puramente artísticas. En este trabajo se han desarrollado algunos de los efectos más utilizados: compresor, ecualizador paramétrico, wah-wah, Eco, vibrato, flanger, chorus, doubling, reverberador Schroeder y un reverberador por convolución. Cada uno de ellos se basa en un sistema que se caracteriza por su ecuación en diferencias o por su respuesta al impulso. En la actualidad existe numerosa documentación en la que se pueden encontrar soluciones para cada efecto en concreto. El trabajo ha consistido básicamente en elegir las soluciones o esquemas de los efectos desarrollados y a continuación, en aplicar la respuesta del sistema a la señal de audio de entrada. Los efectos se aplican a archivos de formato WAV por lo que no es posible utilizarlo en tiempo real. Para el desarrollo del módulo se ha utiliza el lenguaje de programación MATLAB. Por último se ha creado una interfaz más amigable e intuitiva para facilitar la interacción con el usuario, a través de la herramienta GUIDE de MATLAB.

PALABRAS CLAVE: Efectos, Audio, MATLAB, GUIDE, Convolución.

ABSTRACT

At the present time, It is habitual to use audio effects in audiovisual or music industry. This is made to adecuate the signal to the transmission channel or for artistical reasons. In this work, the most common ones have been developed: compressor, parametric equalizer, wah-wah, eco, vibrato, flanger, chorus, doubling, Schroeder reververator and a convolutional reverberator. Each of them is characterized by its difference equations or by its impulse response. It is possible to find different solutions for each effect among the vast existing documentation. The work has consisted on choosing a solution or scheme for the desired effect, and applying the response of the system to the input signal. The input signal is in a WAV format file so it is not possible to apply the effects in real-time. For the development, MATLAB programming language has been used. Finally, it has been created an intuitive and friendly graphical interface with the MATLAB tool GUIDE.

KEYWORDS: Effects, Audio, Matlab, GUIDE, Convolution.

Índice

1. Introducción y objetivos

Mediante los *efectos digitales de audio*, las señales de audio de entrada son modificadas en función de algunos parámetros de control y se generan señales de audio de salida. Este tipo de efectos son muy empleados actualmente por ingenieros de sonido, músicos o simplemente por los que escuchan música para lograr un efecto concreto en el sonido percibido.

Como se verá en este trabajo, estos efectos se pueden lograr por medio de distintos algoritmos que se pueden implementar por *software*. En este trabajo desarrollaremos algunos de los efectos más comunes utilizando Matlab.

Para la creación del módulo de efectos, además de escribir el código correspondiente a cada efecto, desarrollaremos una interfaz gráfica de usuario para su control mediante la herramienta GUIDE de Matlab.

2. Fundamentos del procesado digital de señal

En este apartado repasaremos distintos conceptos teóricos en los que nos basaremos a la hora de desarrollar los distintos efectos.

2.1. Señales digitales

La representación digital de una señal de audio se consigue por medio de los conversores analógico-digitales *(Analog-Digital Converter ADC)*. El ADC toma muestras equidistantes en el tiempo de la señal analógica *x(t)*. La cuantificación consiste en asignar unos valores fijados a cada una de las amplitudes para los instantes obtenidos en el muestreo. Las señal resultante se representa como $x(n)$.

Una vez que se ha asignado cada muestra a un valor o nivel de cuantificación, se procede a representarlo en formato binario. El número de niveles o valores que se pueden representar depende del numero de bits utilizados. Para un número N de bits obtendremos 2^N niveles de cuantificación. A mayor número N de bits mayor resolución. Los valores típicos de resolución son de 16 y 24 bits.

La distancia en el tiempo entre dos muestras consecutivas se conoce como *período de muestreo* T_s. Recíprocamente se define la *frecuencia de muestreo* como $f_s = 1/T_s$. Los valores más utilizados son 44,1 kHz (CD), 48 kHz (Industria audiovisual) y 96 kHz (estudios profesionales de grabación para las producciones musicales).

En la figura 1 se puede apreciar una muestra de un audio digital con una frecuencia de muestreo de 44,1 kHz y $2^{16} = 65536$ niveles de cuantificación (16 bits). Como la señal puede tomar valores negativos, el rango de niveles va desde -32768 hasta 32768.

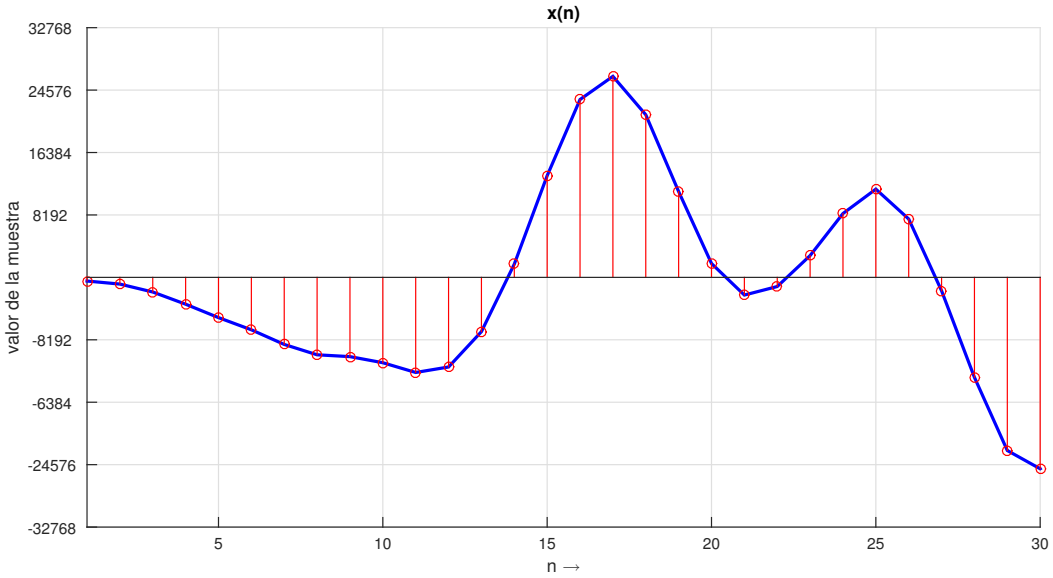

Figura 1: Señal de audio muestreada

Existen distintos formatos para el almacenamiento de las señales digitales de audio (*RIFF, AIFF, WAV...*) . Nuestro módulo de efectos trabajará con archivos de audio de formato *WAV* (mono o estéreo) a diversas resoluciones y frecuencias de muestreo.

2.2. Transformada-Z

La señal $x(n)$ del apartado anterior está descrita en el dominio del tiempo discreto. La transformada-Z es la equivalente digital de la transformada de Laplace para señales en tiempo continuo. Mediante está transformada la señal de $x(n)$ se describe en el dominio de la frecuencia y en función de la variable z.

$$X(z) = \sum_{n=-\infty}^{\infty} x(n) \cdot z^{-n} \tag{1}$$

En ocasiones, puede resultar muy útil trabajar en el dominio de la variable z, para calcular el valor de la señal a la salida de un sistema. Como veremos en el siguiente apartado, ésta se obtiene como el producto de las Transformadas-Z de la señal de entrada $x(n)$ y la respuesta al impulso $h(n)$ del sistema.

2.3. Respuesta al impulso

La respuesta a un impulso o respuesta impulsiva $h(n)$ de un sistema es la que se presenta en la salida, cuando en la entrada se introduce un impulso $\delta(n)$. Un sistema digital queda completamente caracterizado por medio de su respuesta al impulso.

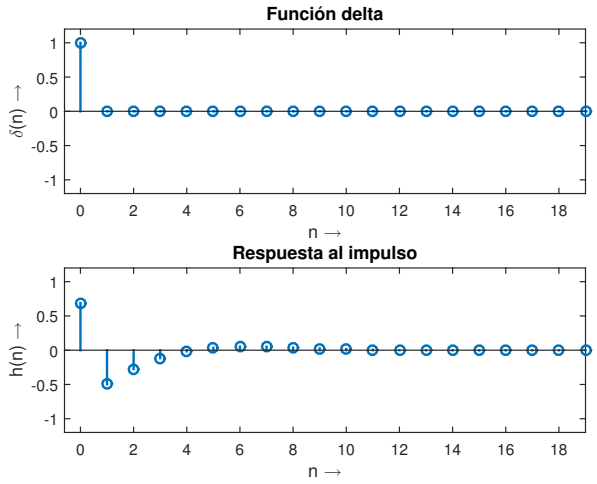

Figura 2: Respuesta al impulso

Si conocemos la respuesta al impulso de un sistema digital podremos calcular la señal de salida $y(n)$ para cualquier señal $x(n)$ a la entrada. Para ello utilizaremos la técnica de la convolución discreta y que se calcula con la ecuacion (3).

$$y(n) = \sum_{n=-\infty}^{\infty} x(k) \cdot h(n-k) = y(n) * h(n) \tag{2}$$

En Matlab se utiliza la función **y=conv(x,h)** para calcular la convolución entre dos señales.

La transformada-Z de una respuesta a impulso es conocida como la **función de transferencia**. Normalmente es más fácil analizar sistemas usando funciones de transferencia en contraposición a las funciones de respuestas a impulso.

$$H(z) = \sum_{n=-\infty}^{\infty} h(n) \cdot z^{-n} \tag{3}$$

La transformada Z de la salida $y(n)$ de un sistema puede determinarse mediante el producto de $H(z)$ y $X(z)$, siendo esta última la transformada-Z de la función entrada $x(n)$.

$$Y(z) = H(z) \cdot X(z) \tag{4}$$

2.4. Sistemas discretos

Un sistema discreto o digital se representa por medio de un algoritmo que tiene como entrada una señal $x(n)$, realiza ciertas operaciones matemáticas y produce como salida otra secuencia de números denominada $y(n)$. Básicamente son tres las operaciones que realiza el sistema: suma, multiplicación y retardo.

Para la representación de los distintos sistemas utilizaremos los *Diagramas de Bloques*. Dichos diagramas describen de forma visual las distintas operaciones que realiza el sistema.

- El retardo de la señal de entrada en un número k de muestras se representa mediante el diagrama de bloques de la figura 3.

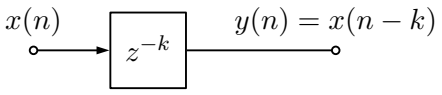

Figura 3: Representación del retardo en un diagrama de bloques

- La multiplicación de la señal de entrada por un coeficiente a se representa en la figura 4.

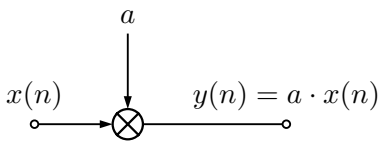

Figura 4: Representación de la multiplicación en un diagrama de bloques

- La suma de dos señales se representa en la figura 5.

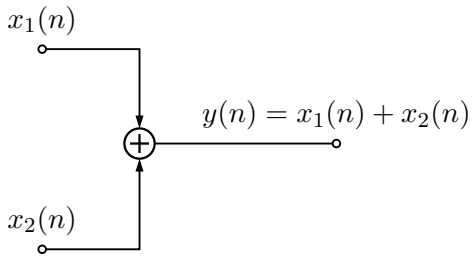

Figura 5: Representación de la suma de dos señales

Un sistema discreto puede ser descrito por medio de su ecuación en diferencias o por su función de transferencia como hemos visto en el apartado 2.3. Son equivalentes, pero a la hora de trabajar a veces puede ser más útil más una u otra. Si se describe el sistema por medio de su ecuación en diferencias en el dominio del tiempo discreto, tendremos:

$$y(n) = \sum_{k=0}^{N} b_k \cdot x(n-k) - \sum_{k=1}^{M} a_k \cdot y(n-k) \tag{5}$$

Por otro lado, si expresáramos el sistema en función de z y en forma de fracción obtendríamos su función de transferencia:

$$H(z) = \frac{\sum_{k=0}^{N} b_k \cdot z^{-k}}{\sum_{k=1}^{M} a_k \cdot z^{-k}} \tag{6}$$

Los coeficientes a_k y b_k definen el filtro y por lo tanto, el diseño consiste en calcularlos. Cada bloque de retardo lleva asociado un coeficiente a_k o b_k. Los coeficientes a_k se asocian a las lineas de retardo que realimentan la señal de entrada y los coeficientes b_k a las lineas que no lo hacen. El orden o grado de una función de transferencia lo define el valor de M (o N, el mayor de los dos).

Una vez tenemos los coeficientes calculados podremos filtrar la señal de entrada. Para filtrar la señal utilizaremos la función **y=filter(b,a,x)** de Matlab donde b y a son vectores que contienen los coeficientes.

2.5. Sistemas IIR-FIR

De acuerdo con el tipo de respuesta al impulso existen dos tipos de sistemas: IIR y FIR.

En los sistemas **IIR** ***(Infinite Impulse Response)*** la salida del sistema depende de los valores de la salida en instantes anteriores. En el ejemplo de la figura 6, la salida $y(n)$ se suma a la señal original $x(n)$ después de haber pasado a través de dos bloques de retardo.

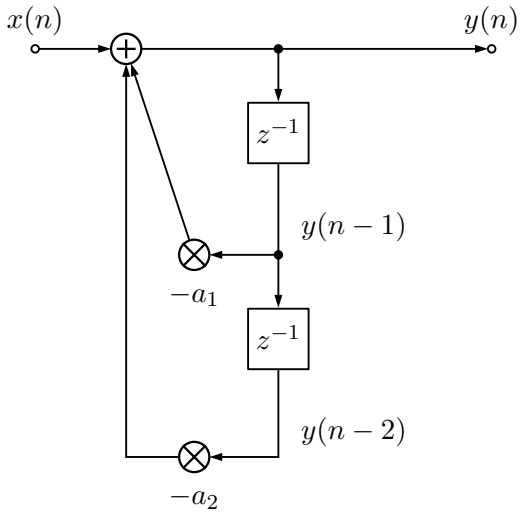

Figura 6: Ejemplo de sistema IIR

La ecuación (8) muestra la ecuación en diferencias del sistema IIR mostrado en la figura 6.

$$y(n) = x(n) - a_1 \cdot y(n-1) - a_2 \cdot y(n-2) \tag{7}$$

La transformada-Z de la ecuación (8) nos conduce a

$$Y(z) = X(z) - a_1 \cdot z^{-1}Y(z) - a_2 \cdot z^{-2}Y(z) \tag{8}$$

$$X(z) = Y(z) \cdot (1 + a_1 z^{-1} + a_2 z^{-2}) \tag{9}$$

Y si resolvemos $Y(z)/X(z)$ obtendríamos la función de transferencia:

$$H(z) = \frac{Y(z)}{X(z)} = \frac{1}{1 + a_1 \cdot z^{-1} + a_2 \cdot z^{-2}} \tag{10}$$

Si extendieramos el número de bloques de retardo hasta N elementos, la expresión general de la ecuación (8) quedaría como:

$$y(n) = x(n) - \sum_{k=1}^{N} a_k \cdot y(n-k) \tag{11}$$

En los sistemas **FIR** ***(Finite Impulse Response)*** la salida del sistema no depende de los valores de la salida en instantes anteriores. Como se puede ver en la figura 7, la señal de entrada $x(n)$ pasa a través de dos bloques de retardo. Las sálidas de estos bloques se suman para crear la señal de salida $y(n)$.

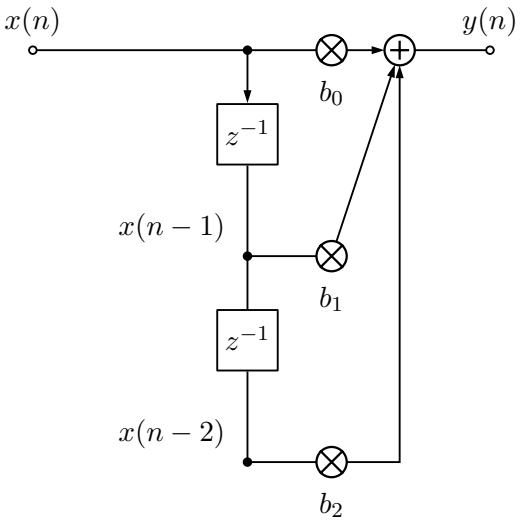

Figura 7: Ejemplo de sistema FIR

La ecuación (13) muestra la ecuación en diferencias del sistema FIR mostrado en la figura 7.

$$y(n) = b_0 \cdot x(n) + b_1 \cdot x(n-1) + b_2 \cdot x(n-2) \tag{12}$$

La transformada Z de la ecuación 13 nos conduce a

$$Y(z) = b_0 \cdot X(z) + b_1 \cdot z^{-1}X(z) + b_2 \cdot z^{-2}X(z) \tag{13}$$

$$Y(z) = X(z) \cdot (b_0 + b_1 \cdot z^{-1} + b_2 \cdot z^{-2}) \tag{14}$$

Y si resolvemos $Y(z)/X(z)$ obtendríamos la función de transferencia:

$$H(z) = \frac{Y(z)}{X(z)} = b_0 + b_1 \cdot z^{-1} + b_2 \cdot z^{-2} \tag{15}$$

Si extendieramos el número de bloques de retardo hasta N elementos, la expresión general de la ecuación (13) quedaría como

$$y(n) = \sum_{k=0}^{N} b_k \cdot x(n-k) \tag{16}$$

En un sistema podemos tener tanto señales que se suman directamente a la salida como señales que se suman a la entrada (retroalimentación). Al sistema IIR descrito podemos sumarle las señales de $x(n)$ retardas a la salida. Un diagrama de bloques más completo del sitema IIR quedaría como se ve en la figura 8.

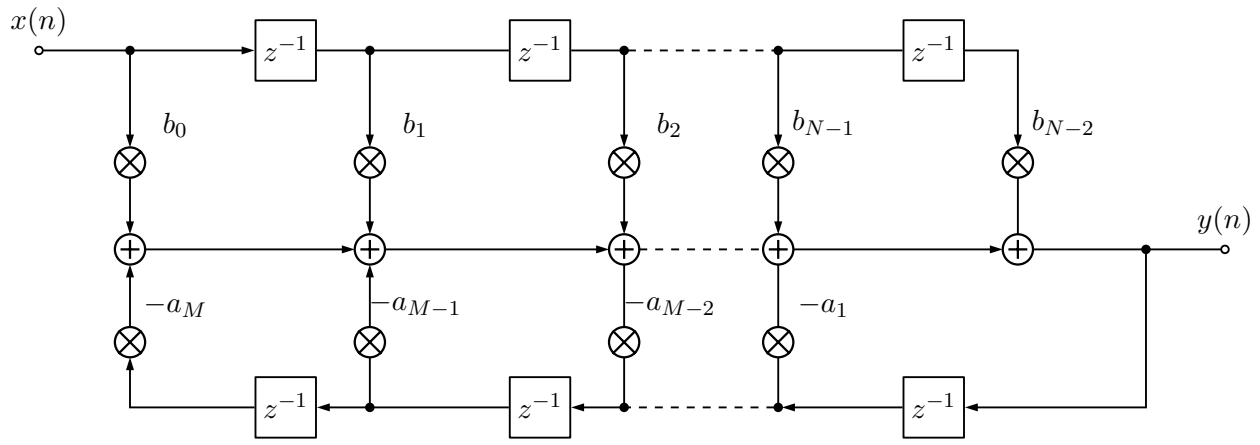

Figura 8: Ejemplo de sistema IIR

La ecuación en diferencias para este tipo de sistema se definió en la ecuación (6).

3. Efectos a implementar

En esta sección se hace una descripción detallada de los efectos implementados en este trabajo.

3.1. Compresión

Un compresor es un procesador de sonido destinado a reducir el margen dinámico de la señal. Esta tarea, se realiza reduciendo la ganancia del sistema, cuando la señal supera un determinado umbral. Los parámetros ajustables de un compresor son los siguientes

- **Umbral (CT):** El compresor trabaja con base en un umbral. Cuando la señal sobrepasa ese umbral se llevará a cabo la compresión, reduciendo el nivel a la cantidad programada con anterioridad. Cuanto más bajo sea el umbral, una mayor parte de señal está siendo procesada. Debemos tener en cuenta, que un umbral por encima del nivel de clip del sistema es inútil. Equivale a poner el compresor en bypass.
- **Proporción (R):** Representa la reducción de la ganancia bajo las condiciones señaladas. Una proporción de 2:1, por ejemplo, significa que una vez que el nivel de la señal excede el umbral, se le permite al nivel de señal aumentar 1 dB por cada 2 dB de aumento de la entrada. En la práctica una relación de 8:1 ó mas, se considera un "limitador".
- **Tiempo de ataque (at):** Es el tiempo que tarda una señal en comprimirse desde que sobrepasa el nivel del umbral. En un buen compresor con tiempo de ataque ajustable, tendremos una gama de ajuste de 500 microsegundos a 100 milisegundos.
- **Tiempo de decaimiento (rt):** Es el tiempo que tarda el compresor en anular el control una vez pasada la sobrecarga. Si el tiempo de liberación es corto, la ganancia volverá a su estado

original rápidamente. Si es largo, el compresor seguirá actuando cuando aparezca la siguiente señal. El rango de ajuste varia entre los 100 ms y los 3 seg.

En este trabajo desarrollaremos un compresor como el descrito en el capítulo 7 de [Zol08]. El diagrama de bloques del compresor se muestra en la figura 9.

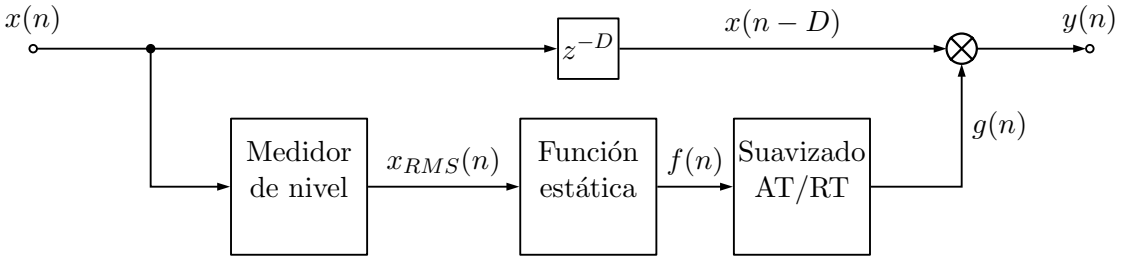

Figura 9: Diagrama de bloques del compresor

El nivel de salida $y(n)$ se obtiene multiplicando la señal retardada de $x(n)$ por un factor $g(n)$ tal y como se muestra en la ecuación (18). El cálculo del factor $g(n)$ se realiza en tres fases. En primer lugar se mide el valor $x_{RMS}(n)$ de cada muestra. En el bloque de la *Función estática* se calcula el factor $f(n)$. Este factor depende del umbral CT, y atenua en una proporción R la porción de la señal de entrada que sobrepasa CT. En el último bloque se suaviza el factor $f(n)$ dependiendo de los valores at o rt y se obtiene el factor $g(n)$.

$$y(n) = g(n) \cdot x(n - D) \tag{17}$$

El retardo D aplicado a la señal de entrada se ha fijado arbitrariamente en 150 muestras. se ha elegido dicho valor por ser el utilizado en la documentación consultada [Zol11].

La función **y = comp (x, Fs, at, rt, CT, R)** realiza los cálculos que describiremos en los siguientes apartados.

3.1.1. Medición de nivel

Como se ha visto en el diagrama de la figura 9, en primer lugar se debe detectar el nivel de la señal de entrada $x(n)$. Para ello desarrollaremos un medidor de nivel como el descrito en [Zol08]. En la figura 10 se puede ver la estructura del medidor en cuestión.

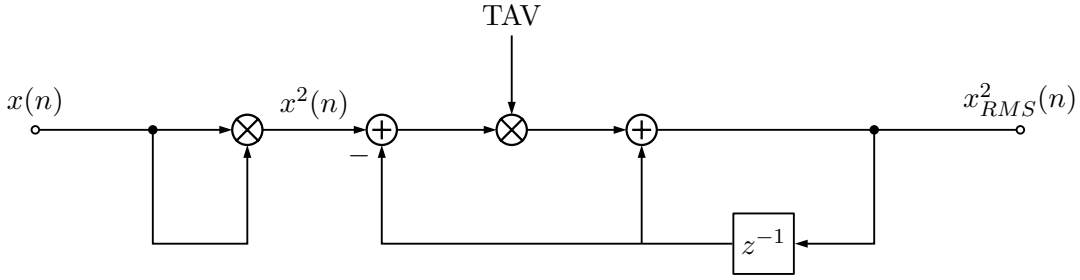

Figura 10: Medidor de nivel

El sistema de medición se caracteriza mediante la ecuacion en diferencias (19)

$$x^2_{RMS}(n) = (1 - TAV) \cdot x^2_{RMS}(n-1) + x^2(n) \cdot TAV \tag{18}$$

El factor TAV es un coeficiente de promedio que depende dela constante de tiempo del medidor tav y se relaciona como se muestra en la ecuación (20)

$$TAV = 1 - exp\left(\frac{-2{,}2 \cdot T_S}{tav/1000}\right) \tag{19}$$

En la ecuación anterior T_S representa el período de muestreo y tav se ha fijado en 125 ms. Este es el valor utilizado por los medidores comerciales para el modo de medición *Fast*.

Después de calcular el valor de $x_{RMS}(n)$, lo trasladaremos al dominio logarítmico y hallaremos el valor $X_{RMS}(n)$ mediante la ecuacion 21.

$$X_{RMS} = 10 \cdot log(x_{RMS}) \tag{20}$$

Cuando trabajemos con señales estereo no aplicaremos la compresión a cada canal por separado. Se necesita un factor $g(n)$ común para ambos canales. Si tuvieramos dos valores $g(n)$ distintos para cada canal se produciria un desplazamiento del balance estereo. Nos basaremos en el medidor propuesto en [Zol08]. El diagrama de bloques de dicho medidor se observa en la figura 11.

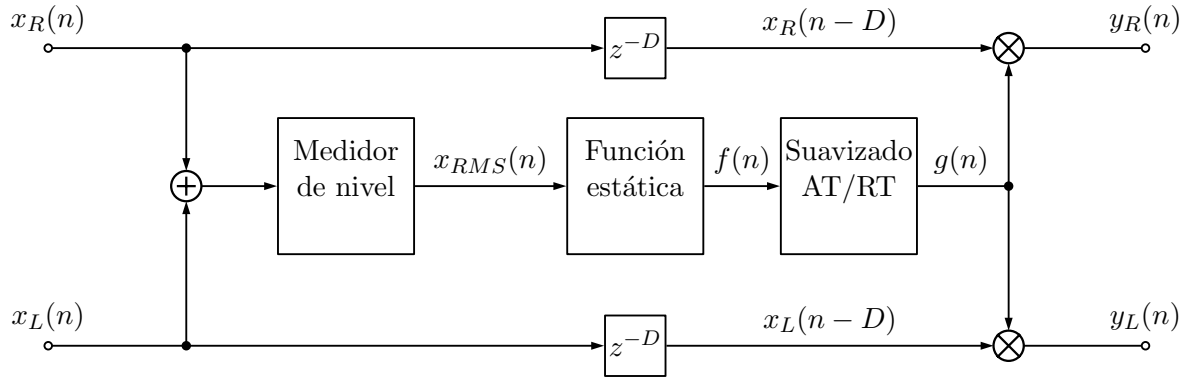

Figura 11: Medidor de nivel para señales estereo

En nuestro caso y al tratarse de señales de audio digitales que varían entre -1 y 1, utilizaremos el fondo de escala **dBFs**. El valor máximo posible se asignará a **0 dB** y el valor 0 a $-\infty$ **dB**.

3.1.2. Función estática

En la figura 12 se representa la relación entre la señal de entrada $x(n)$ y la señal de salida $y(n)$. X y Y son los valores de la señal en el dominio logarítmico de $x(n)$ y $y(n)$ respectivamente.

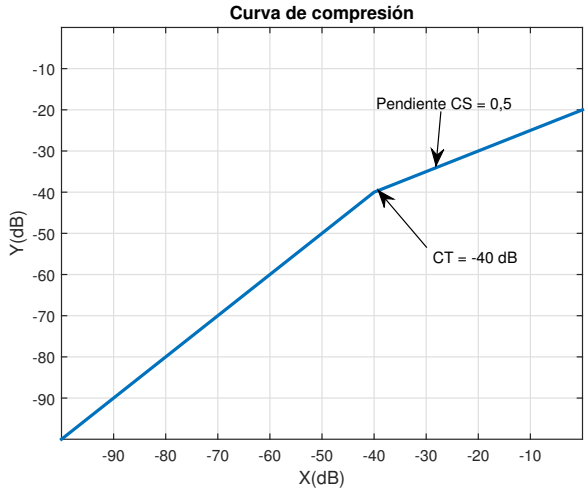

Figura 12: Curva de compresión

Cuando el nivel de la señal de entrada X_{RMS} supera el umbral CT, el nivel de salida es atenuado. La proporción en la que se atenua la señal a la salida depende de la *proporción* R. La relación entre la variación de la señal de entrada y la variación de la señal de salida en el dominio logarítmico se puede ver en la ecuación (22).

$$R = \frac{\Delta P_i}{\Delta P_o} \tag{21}$$

Análogamente se define la pendiente S o CS *(slope)* de la función estática para los valores de X superiores al umbral CT.

$$S = CS = 1 - \frac{1}{R} \tag{22}$$

La función que nos da el valor de la ganancia para un valor de entrada X se conoce como función estática F. Para obtener el valor de F aplicaremos la ecuación (24). F se define como el valor de $f(n)$ en el dominio logarítmico y se mide en decibelios (dB).

$$F = -CS \cdot (X - CT) \tag{23}$$

Como se puede observar en la ecuación el signo de F es negativo. Esto se debe a que en la compresión de audio se atenua la señal de entrada. Hay que recordar que dicha expresión solamente se aplica para niveles de X que superen el umbral CT. Es decir, cuando el valor de $(X - CT)$ es positivo. Si esto no ocurre el valor de F es cero.

En la figura 13 podemos observar la función estática para el ejemplo representado en la figura 12.

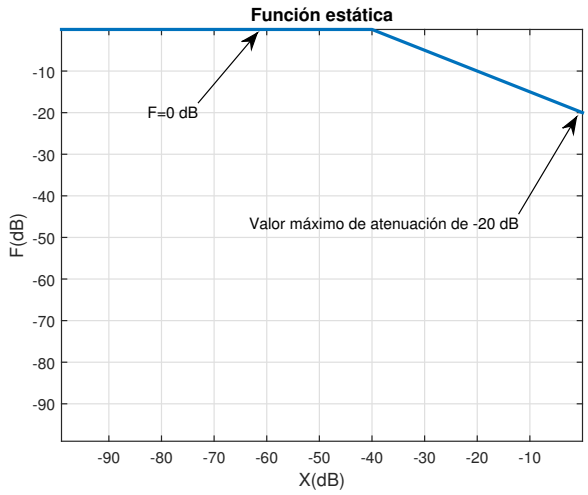

Figura 13: Función estática

3.1.3. Factor de suavizado

Además de variar la señal a la salida en función del nivel de entrada también debemos tener en cuenta los tiempos de ataque (at) y decaimiento (rt). Para ello se define el factor de suavizado $g(n)$. Este valor depende de los valores de at, rt y $f(n)$. En [Zol11] se propone un esquema para realizar dicho cálculo. El diagrama de bloques del sistema puede apreciar en la figura 14.

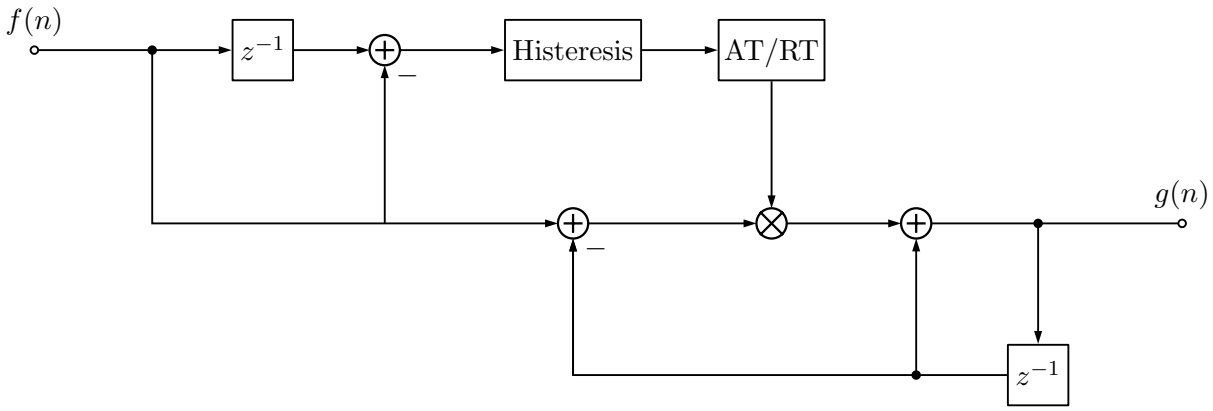

Figura 14: Factor de suavizado g

Tal y como hemos definido el factor TAV, también podemos definir los factores AT y RT.

$$AT = 1 - exp\left(\frac{-2{,}2 \cdot T_S}{at/1000}\right) \tag{24}$$

$$RT = 1 - exp\left(\frac{-2{,}2 \cdot T_S}{rt/1000}\right) \tag{25}$$

En primer lugar, se detecta si el compresor se encuentra en la fase de ataque o decaimiento. Para ello, se compara el factor de control $f(n)$ con el factor de suavizado $g(n-1)$ en el instante previo. Una curva de histeresis determina si el compresor está en la fase de ataque o decaimiento. En el primer

caso k toma el valor AT y en el segundo RT. Todo este proceso se realiza en los bloques *Histeresis* y *AT/RT* del diagrama de la figura 14.

En el sistema propuesto en [Zol11], el proceso anterior se resuelve mediante las siguientes lineas:

```
if f < g
        k = AT; % Si f es menor que g en el instante previo está en la fase AT
else
        k = RT; % Si f es mayor que g en el instante previo está en la fase RT
end;
```

Una vez hallado el valor de k, el valor de $g(n)$ se obtiene mediante la ecuación (27).

$$g(n) = (1 - k) \cdot g(n - 1) + k \cdot f(n) \tag{26}$$

En la figura 15 se pueden ver los niveles de una señal sin comprimir y la misma señal comprimida mediante el comando **y = comp (x, Fs, 50, 100, -20, 2)**.

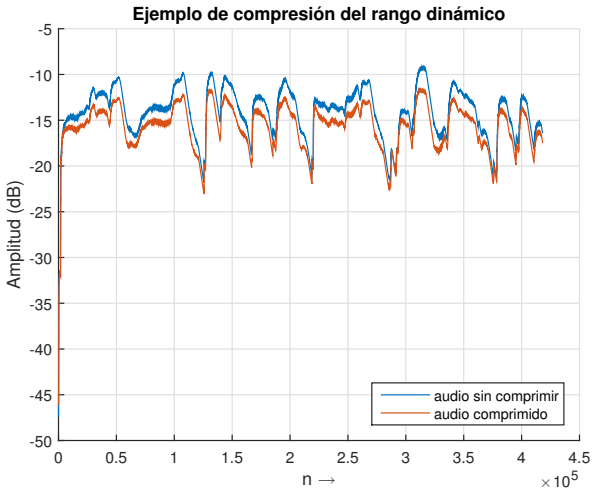

Figura 15: Ejemplo de compresión del rango dinámico de una señal de audio

3.2. Ecualizador paramétrico

Después de pasar la señal a través del compresor viene el ecualizador paramétrico. Estos ecualizadores modifican la respuesta en frecuencia de la señal de entrada $x(n)$. Por ejemplo, se pueden ecualizar instrumentos musicales para corregir o modificar su respuesta en frecuencia. Hay que señalar que estas ecualizaciones dependen mucho del gusto artístico del técnico de sonido y/o instrumentista que para determinados estilos musicales buscan una exageración de niveles en determinadas frecuencias.

3.2.1. Conceptos básicos

Los parámetros principales de los filtros son los siguientes:

- **Frecuencia de corte** f_c. La frecuencia, bien por arriba o bien por debajo, para la cual el nivel de salida se reduce en -3 dB.
- **Frecuencia central** f_c. Es la media geométrica de la frecuencia más baja f_l y más alta f_u del filtro pasa-banda, (el valor de la frecuencia sobre el que actúa el filtro). Corresponde al valor de la frecuencia sobre el cual su acción será máxima.

$$f_c = \frac{f_u - f_1}{2} \tag{27}$$

- **Ancho de banda f_b, factor de calidad Q y factor de amortiguamiento ζ**. El ancho de banda de un filtro f_b es la diferencia entre las frecuencias en las que su atenuación al pasar a través de filtro se mantiene igual o inferior a 3 dB comparada con la frecuencia central de pico f_c.

$$f_b = f_u - f_l \tag{28}$$

El factor Q nos permite ajustar el ancho de banda f_b del filtro en función de la frecuencia central f_c y ámbas se relacionan como se vé en la ecuación 30

$$Q = \frac{f_c}{f_b} \tag{29}$$

El factor de calidad Q se relaciona con el factor de amortiguamiento ζ mediante la siguiente expresión:

$$Q = \frac{1}{2\zeta} \tag{30}$$

- **Ganancia** La ganancia es la cantidad de amplificación o atenuación que provoca el filtro sobre la señal. Se expresa en decibelios para cada filtro y generalmente suele oscilar entre $\pm 12dB$.

En la figura 16 se puede ver un ejemplo de la repuesta en frecuencia de un filtro. Se trata de un filtro con una frecuencia central f_c en 2000 Hz, una ganancia G de 6 dB y ancho de banda f_b de 1500 Hz.

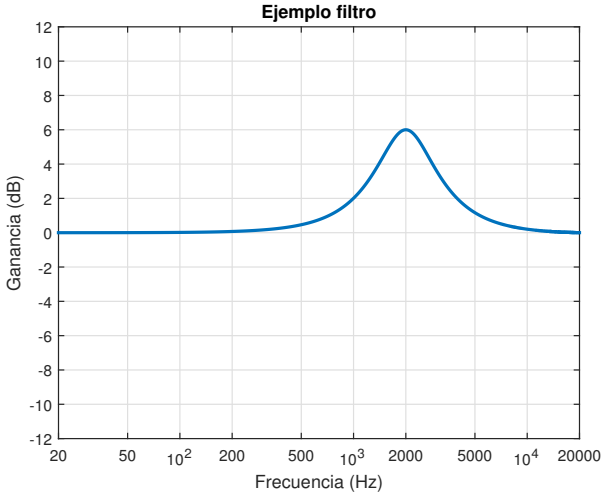

Figura 16: Ejemplo de respuesta en frecuencia de un filtro

El ecualizador parámetrico implementado en este trabajo actuará sobre cuatro bandas de frecuencias: bajas frecuencias (inferiores a 250Hz), media baja (250 a 2.000Hz), media alta (2000 a 4.000Hz) y altas (superiores a 4.000 Hz).

3.2.2. Filtros básicos

Para la realización del ecualizador paramétrico y otros efectos que veremos más adelante, nos basaremos en los siguentes filtros básicos.

- **Pasa-todo (*all-pass*):** En el libro [Zol02] se describe un filtro pasa-todo de segundo orden.

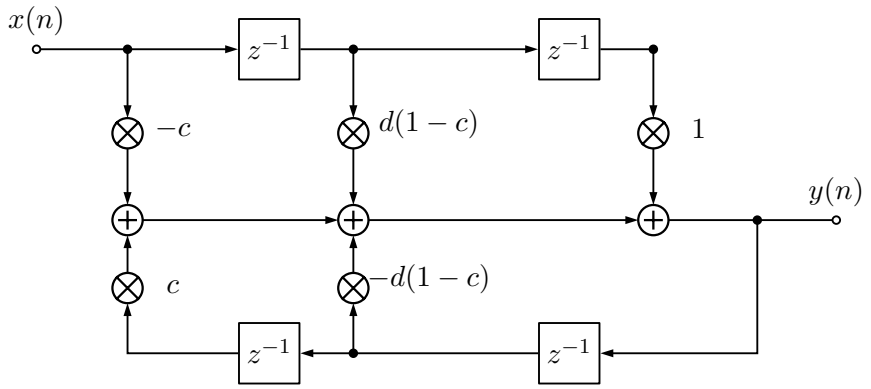

Figura 17: Diagrama de bloques de filtro paso-banda

En la ecuación (32) se muestra la función de transferencia de dicho filtro.

$$A(z) = \frac{-c + d(1-c)z^{-1} + z^{-2}}{1 + d(1-c)z^{-1} + cz^{-2}} \tag{31}$$

$$c = \frac{tan(\pi f_b/f_s) - 1}{tan(\pi f_b/f_s) + 1} \tag{32}$$

$$d = -cos(2\pi f_c/f_s) \tag{33}$$

Por medio del parámetro c se ajusta el ancho de banda del filtro y mediante el parámetro d la frecuencia de corte. Este tipo de filtro no modifica el valor de la magnitud a la salida y únicamente produce cambios en la fase de la señal. Los coeficientes de este tipo de filtros se calculan según lo definido en [Zol08] y se muestran en la tabla 1.

Pasa-todo de segundo orden donde $c = \frac{tan(\pi f_b/f_s)-1}{tan(\pi f_b/f_s)+1}$ y $d = -cos(2\pi f_c/f_s)$				
b_0	b_1	b_2	a_1	a_2
$-c$	$d(1-c)$	1	$d(1-c)$	$-c$

Tabla 1: Coeficientes de un filtro pasa-todo de segundo orden

Como hemos visto en la sección 2.4, estos coeficientes se pueden utilizar con la función **filter** donde a y b son los vectores que contienen los coeficientes, o mediante la ecuación en diferencias del sistema y que en este caso queda como 35.

$$y(n) = b_0 \cdot x(n) + b_1 \cdot x(n-1) + b_2 \cdot x(n-2) - a_1 \cdot y(n-1) - a_2 \cdot y(n-2) \tag{34}$$

- **Paso-banda(*band-pass*):** A partir del filtro pasa-todo anterior podemos crear un filtro pasa-banda de segundo orden. Para ello nos basaremos en el diagrama 18 y que está descrito en la página 43 de [Zol02].

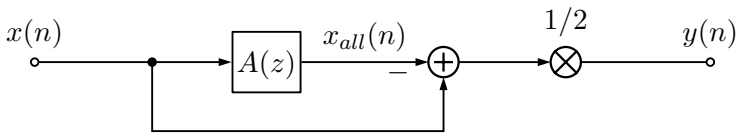

Figura 18: Diagrama de bloques de filtro paso-banda

En la ecuación (36) se muestra la ecuación en diferencias de dicho sistema.

$$y(n) = \frac{1}{2}[x(n) - x_{all}(n)] \tag{35}$$

donde $x_{all}(n)$ es la señal a la salida del filtro pasa-todo.

- **Paso-bajo(*low-pass*):**. En la tabla 2 se muestran los coeficientes para un filtro paso-bajo tal y como vienen descritos en la página 43 de [Zol02]

Pasa-bajo de segundo orden donde $K = tan(\pi f_c/f_s)$				
b_0	b_1	b_2	a_1	a_2
$\frac{K^2}{1+\sqrt{2}K+K^2}$	$\frac{2K^2}{1+\sqrt{2}K+K^2}$	$\frac{K^2}{1+\sqrt{2}K+K^2}$	$\frac{2(K^2-1)}{1+\sqrt{2}K+K^2}$	$\frac{1-\sqrt{2}K+K^2}{1+\sqrt{2}K+K^2}$

Tabla 2: Coeficientes de un filtro pasa-bajo de segundo orden

La función **y=lowpass(x,Fs,Fc)** aplica un filtrado paso-bajo con la Frecuencia de corte F_c, a la señal de entrada x con frecuencia de muestreo F_s.

3.2.3. Filtro Shelving

Este tipo de filtros amplifica o atenúa la señal por encima o por debajo de una determinada frecuencia de corte. En la figura 19 se puede observar la respuesta de este tipo de filtros. Se muestran distintas repuestas de filtros con distintas ganancias. Todas ellas tienen la frecuencia de corte en 100 Hz.

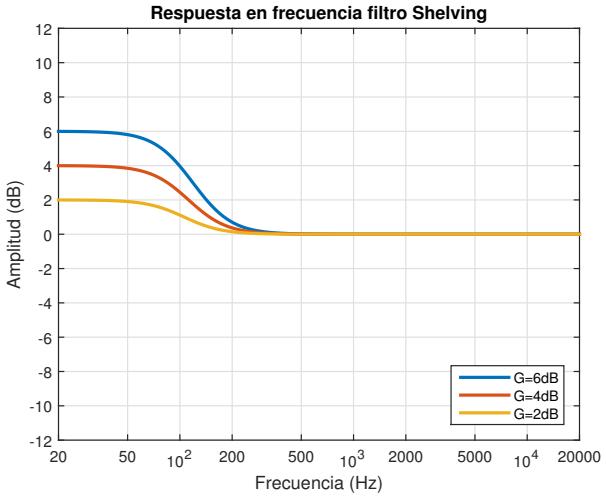

Figura 19: Respuesta filtros Shelving

Los coeficientes para un filtro Shelving de segundo orden se muestran en las tablas 5.4 y 5.5 de [Zol08] y se pueden ver en la tabla 3.

Shelving de baja-frecuencia (Amplificación $V_0 = 10^{G/20}$)				
b_0	b_1	b_2	a_1	a_2
$\frac{1+\sqrt{2V_0}K+V_0K^2}{1+\sqrt{2}K+K^2}$	$\frac{2(V_0K^2-1)}{1+\sqrt{2}K+K^2}$	$\frac{1-\sqrt{2V_0}K+V_0K^2}{1+\sqrt{2}K+K^2}$	$\frac{2(K^2-1)}{1+\sqrt{2}K+K^2}$	$\frac{1-\sqrt{2}K+K^2}{1+\sqrt{2}K+K^2}$
Shelving de baja-frecuencia (Atenuación $V_0 = 10^{-G/20}$)				
b_0	b_1	b_2	a_1	a_2
$\frac{1+\sqrt{2}K+K^2}{1+\sqrt{2V_0}K+V_0^2}$	$\frac{2(K^2-1)}{1+\sqrt{2V_0}K+V_0K^2}$	$\frac{1-\sqrt{2}K+K^2}{1+\sqrt{2V_0}K+V_0K^2}$	$\frac{2(V_0K^2-1)}{1+\sqrt{2V_0}K+V_0K^2}$	$\frac{1-\sqrt{2V_0}K+V_0K^2}{1+\sqrt{2V_0}K+V_0K^2}$
Shelving de alta-frecuencia (Amplificación $V_0 = 10^{G/20}$)				
b_0	b_1	b_2	a_1	a_2
$\frac{V_0+\sqrt{2V_0}K+K^2}{1+\sqrt{2}K+K^2}$	$\frac{2(K^2-V_0)}{1+\sqrt{2}K+K^2}$	$\frac{V_0-\sqrt{2V_0}K+K^2}{1+\sqrt{2}K+K^2}$	$\frac{2(K^2-1)}{1+\sqrt{2}K+K^2}$	$\frac{1-\sqrt{2}K+K^2}{1+\sqrt{2}K+K^2}$
Shelving de alta-frecuencia (Atenuación $V_0 = 10^{-G/20}$)				
b_0	b_1	b_2	a_1	a_2
$\frac{1+\sqrt{2}K+K^2}{V_0+\sqrt{2V_0}K+K^2}$	$\frac{2(K^2-1)}{V_0+\sqrt{2V_0}K+K^2}$	$\frac{1-\sqrt{2}K+K^2}{V_0+\sqrt{2V_0}K+K^2}$	$\frac{2(V_0/K^2-1)}{1+\sqrt{2/V_0}K+K^2/V_0}$	$\frac{1-\sqrt{2/V_0}K+K^2/V_0}{1+\sqrt{2}K+K^2}$

Tabla 3: Shelving de segundo orden donde $K = tan(\pi f_c/f_s)$

Como se puede observar los coeficientes y el valor de V_0 dependen del valor de la ganancia G. Los coeficentes se calculan de distinta manera dependiendo de si atenuamos o amplificamos la señal.

Las funciones **y=lowshelving(x,Fs,G,Fc)** y **y=highshelving(x,Fs,G,Fc)** aplican una ecualización a la señal de entrada x en bajas o altas frecuencias respectivamente. En estas funciones F_c es la frecuencia de corte del filtro, G es la gananacia y F_s la frecuencia de muestreo de la señal de entrada.

3.2.4. Peak

Estos filtros amplifican o atenúan las frecuencias comprendidas en una determinada banda. En la figura 20 se puede observar la respuesta de este tipo de filtros con distintos anchos de banda.

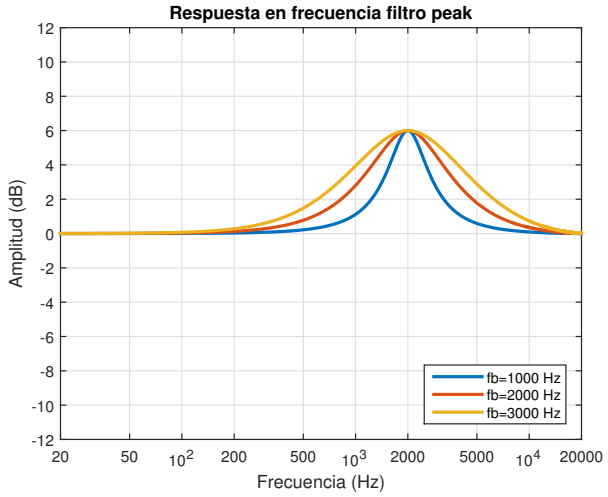

Figura 20: Respuesta filtros peak

Los coeficientes para un filtro Peak de segundo orden se muestran en las tabla 5.3 de [Zol08] y se pueden ver en la tabla 4.

Peak (Amplificación $V_0 = 10^{G/20}$)				
b_0	b_1	b_2	a_1	a_2
$\frac{1+\frac{V_0}{Q_\infty}K+K^2}{1+\frac{1}{Q_\infty}K+K^2}$	$\frac{2(K^2-1)}{1+\frac{1}{Q_\infty}K+K^2}$	$\frac{1-\frac{V_0}{Q_\infty}K+K^2}{1+\frac{1}{Q_\infty}K+K^2}$	$\frac{2(K^2-1)}{1+\frac{1}{Q_\infty}K+K^2}$	$\frac{1-\frac{1}{Q_\infty}K+K^2}{1+\frac{1}{Q_\infty}K+K^2}$
Peak (Atenuación $V_0 = 10^{-G/20}$)				
b_0	b_1	b_2	a_1	a_2
$\frac{1+\frac{1}{Q_\infty}K+K^2}{1+\frac{V_0}{Q_\infty}K+K^2}$	$\frac{2(K^2-1)}{1+\frac{V_0}{Q_\infty}K+K^2}$	$\frac{1-\frac{1}{Q_\infty}K+K^2}{1+\frac{V_0}{Q_\infty}K+K^2}$	$\frac{2(K^2-1)}{1+\frac{V_0}{Q_\infty}K+K^2}$	$\frac{1-\frac{V_0}{Q_\infty}K+K^2}{1+\frac{V_0}{Q_\infty}K+K^2}$

Tabla 4: Filtro Peak de segundo orden donde $K = tan(\pi f_c/f_s)$

De la misma forma que en el apartado anterior el valor los coeficientes y V_0 dependen del valor de la ganancia G y del factor de calidad Q_∞. A su vez, Q_∞ depedende del ancho de banda f_b y de la frecuencia central del filtro f_c como se vio en la ecuacion (30).

La función **y = peakfilt(x,Fs,G,Fc,Fb)** aplica una ecualización a la señal de entrada x en una determinada banda de frecuencias. En esta función F_c es la frecuencia de central del filtro, G es la gananacia, F_b es el ancho de banda y F_s la frecuencia de muestreo de la señal de entrada.

Conectando en cascada los filtros shelving y peak podremos implementar un equalizador paramétrico como el que se vé en la figura 21.

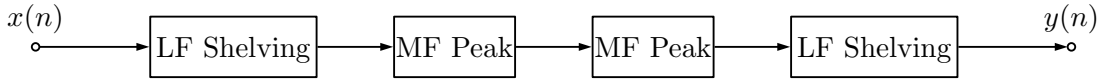

Figura 21: Diagrama de bloques de ecualizador paramétrico

3.3. Wah-wah

El efecto "wah-wah"se consigue variando durante el tiempo la frecuencia central de un filtro pasa banda. En nuestro caso controlaremos la frecuencia central del filtro con un oscilador de baja frecuencia. A continuación se mezcla la señal con el efecto wah-wah con la señal directa. Implementaremos el efecto basándonos en el diagrama de bloques descrito en la página 56 [Zol02] y que se puede ver en la figura 22.

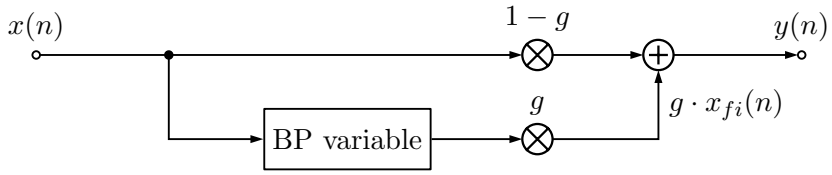

Figura 22: Diagrama de bloques de efecto wah-wah

La ecuación en diferencias de un filtro Wah-wah se define en la ecuación (37)

$$y(n) = (1-g)\cdot x(n) + g \cdot x_{fi}(n) \tag{36}$$

donde $x_{fi}(n)$ es la señal a la salida del fitro pasa-banda.

El hecho de que el valor de F_c sea distinto en cada momento dificulta la utilización de la función **filter** de Matlab. En su lugar, filtraremos la señal de entrada $x(n)$ mediante su ecuación en diferencias. Una vez filtrada, sumaremos ésta a la señal original.

Las función **y=wahwah(g,Fmin,Fmax,Fw,Fs,x)** aplica el efecto wah-wah a la señal original x. Esta función permite fijar distintos parámetros del efecto wah-wah.

Por una parte permite variar la proporción entre la señal original $x(n)$ y la señal $x_{fi}(n)$ que ha sido filtrada con el filtro paso-banda. Esto se consigue ajustando el valor de la ganancia g. Si $g = 1$ unicamente tendremos la señal filtrada a la salida y en consecuencia, si $g = 0$ no se añadirá la señal filtrada a la original.

Como hemos comentado anteriormente la frecuencia central del filtro varía en el tiempo. A la hora de diseñar el efecto wah-wah nos basaremos en el clásico pedal wah-wah "Cry Baby GCB 95".

Figura 23: Pedal wah-wah "Cry Baby GCB 95"

Según los datos del fabricante [CryB], la frecuencia mínima del filtro utilizado varía entre 350 y 450 Hz. Por otro lado la frecuencia máxima varía entre 1.5 y 2.5 kHz. La función **wahwah** desarrollada permite ajustar estos parámetros a través de las variables **Fmin** y **Fmax**. Una vez fijados dichos valores, la frecuencia media del filtro utilizado se obtiene según la ecuación (38)

$$F_m = \frac{F_{min} + F_{max}}{2} \tag{37}$$

En nuestro caso, la variación de la frecuencia central del filtro se producirá por medio de un LFO cuya frecuencia varía entre 0,1 y 4 Hz. Este valor se ajusta por medio de la variable **Fw**. Por último la frecuencia central se calcula como se ve en la ecuación (39).

$$F_c = F_m + F_a \cdot sin(2\pi F_w t) \tag{38}$$

donde F_a es la amplitud de la variación de la señal senoidal.

El filtro paso-banda utilizado se ha descrito en el apartado 3.2.2.

3.4. Efectos basados en retardo o *Delay*

El retardo o *Delay* es un efecto de sonido que consiste en la multiplicación y retardo de la señal. Una vez procesada la señal se mezcla con la original. Las variables básicas de un retardo son:

- **Retardo (τ):** Es el tiempo que tarda en producirse una repetición de la señal original.
- **Mezcla (g):** Es la cantidad de sonido retrasado que se mezcla con el original.

El sistema que realiza el retardo simple de una señal se conoce como *Filtro-peine FIR*.

3.4.1. Estructuras de retardo básicas

La estructura más básica para producir un retardo de M muestras a la señal de entrada se muestra en la figura 24.

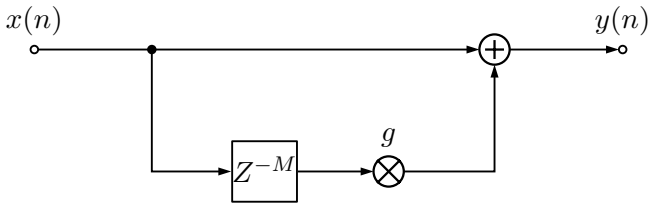

Figura 24: Diagrama de bloques de filtro peine FIR

Este sistema corresponde a la ecuación en diferencias (40).

$$y(n) = x(n) + g \cdot x(n - M) \tag{39}$$

$$M = \tau f_s \tag{40}$$

En la ecuación anterior M es el valor del retardo en número de muestras, τ es el valor del retardo en segundos y f_s es la frecuencia de muestreo de la señal de entrada en Hz.

Este tipo de filtro tiene el calificativo de "peine" ("comb" en inglés) por la forma de su respuesta en frecuencia. En concreto el filtro del diagrama anterior se conoce como filtro tipo peine FIR, ya que la salida del sistema no depende de los valores de la salida en instantes anteriores.

Si modificamos el sistema como se aprecia en la figura 25 y hacemos que la salida del sistema dependa de los valores de la salida en instantes anteriores obtendremos un filtro tipo peine IIR.

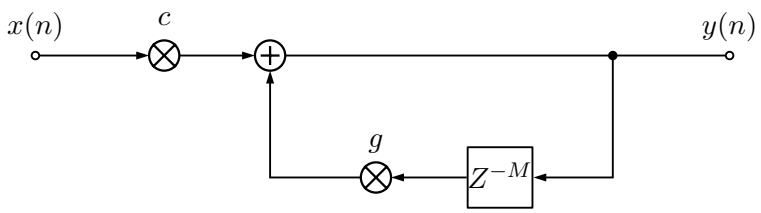

Figura 25: Diagrama de bloques de filtro peine IIR

La ecuación en diferencias del sistema de la figura 25 se muestra en la ecuación (42).

$$y(n) = c \cdot x(n) + g \cdot y(n - M) \tag{41}$$

Por último, la combinación de los filtros peine FIR e IIR nos conduce a un filtro peine universal que veremos en el siguiente apartado.

3.4.2. Unidad de retardo estándar

En [Dat97] se define una unidad de retardo estándar mediante el cual podremos aplicar distintos efectos a la señal original. Esta unidad se basa en la combinación de los filtros peine FIR e IIR descritos en el apartado anterior. En función de los valores que se asignen a distintas variables, se puede configurar el sistema para que produzca un efecto concreto. El digrama de bloques de dicho sistema se puede ver en la figura 26.

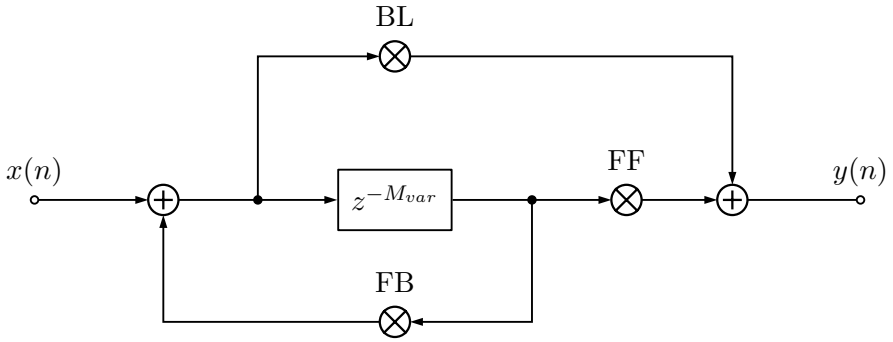

Figura 26: Unidad de retardo estándar

En este caso el retardo en número de muestras M_{var} es variable. A continuación se hace una breve descripción de los parámetros de este sistema:

- **Factor BL ("Blend"):** Factor de ganancia para la señal directa
- **Factor FF ("Feedforward"):** Factor de ganancia para la señal retardada y que no realimenta el sistema.
- **Factor FB ("Feedback"):** Factor de ganancia para la señal retardada y que realimenta el sistema.
- **Retardo ("Delay"):** Tiempo de retardo aplicado. En nuestro caso utilizaremos dicho valor en milisegundos. Para obtener el número de muestras en las que se retarda la señal aplicaremos la relación $M = (\tau/1000) \cdot f_s$.
- **Profundidad ("Depth"):** Para algunos efectos el retardo no es fijo, varía a lo largo del tiempo. La profundidad define el valor de la máxima variación. También lo utilizaremos en milisegundos y calcularemos el retardo en número de muestras de la misma forma que para el Delay.
- **Modulación ("MOD"):** Señal que se utiliza para variar el retardo en el tiempo. Puede tratarse de una señal senoidal, ruido aleatorio... En nuestro caso utilizaremos un vector que con valores comprendidos entre -1 y 1.

El valor del retardo en cada instante n, $M_{var}(n)$ se obtiene a partir de la relación (43).

$$M_{var}(n) = Delay + MOD(n) \cdot Depth \tag{42}$$

Dependiendo de los valores asignados a cada uno de ellos obtendremos un *Eco*, *Vibrato*, *Flanger*, *Chorus* o *Doubling*. En la tabla 5 se muestran los valores propuestos para la unidad de retardo estándar en [Dat97] y [Zol02] en función del efecto que se quiera lograr.

	BL	FF	FB	Retardo	Profundidad	MOD
Eco	1	0-1	0	$>$ 50 ms	0 ms	-
Vibrato	0	1	0	0 ms	0-3 ms	0.1-5 Hz
Flanger	0.7071	0.7071	-0.7071	0 ms	0-2 ms	0.1-1 Hz
Chorus	1	0.7071	0	1-30 ms	1-30 ms	Ruido
Doubling	0.7071	0.7071	0	10-100 ms	1-100 ms	Ruido

Tabla 5: Unidad de retardo estándar

Hay que tener en cuenta que al tratarse de un retardo variable, en ocasiones el valor de M_{var} puede que no sea un valor entero. En [Zol02] se proponen varios métodos de interpolación para resolverlo. En nuestro caso hemos utilizado la interpolación lineal que se describe en la ecuación (44)

$$y(n) = x(n - [M + 1])frac + x(n - M)(1 - frac) \tag{43}$$

En la expresión anterior $frac$ es la parte fraccional del retardo M_{var} en numero de muestras y M la parte entero. Por ejemplo, si en un instante de tiempo tenemos que retardar la señal en $M_{var} = 22569,256$ muestras, la parte entera M tendria un valor de 22569 y $frac$ unvalor de 0,256.

La función **y = stdelay(x,Fs,BL,FF,FB,Delay,Depth,MOD)** produce a la salida una señal filtrada por la unidad de retardo estándar descrita en esta sección. Todas las funciones desarrolladas para crear los efectos descritos en la tabla 5 harán uso de esta función.

3.4.3. Eco

El efecto *eco* produce una mezcla de la señal original y la señal retardada en la que las dos señales se distinguen perfectamente. Si el retardo entre dos señales es inferior a 50 ms, el oído humano los "integra" y los percibe como un único sonido. Por lo tanto el retardo no puede ser inferior a 50 ms.

El retardo es fijo y se modifica por medio de la variable *Delay*.

El hecho de que el retardo no se pueda variar implica que la profundidad sea nula ($Depth = 0$) y que no exista ninguna señal que module dicha variación ($MOD = 0$).

Para crear el efecto de eco utilizaremos la función **y = eco(x,Fs,Delay,g)**. En esta función se toman como valores $BL = 0$, $FF = g$ y $FB = 0$. El valor de g debe estar entre 0 y 1.

3.4.4. Vibrato

El *vibrato* es un efecto en el que se produce una variación períodica del tono de un sonido. La variación se produce por la diferencia de la distancia entre la fuente y el receptor. En nuestro caso variar la distancia es el equivalente a variar el retardo de una señal.

En este caso no existe un retardo fijo ($Delay = 0$). El retardo solamente se modifica por la profundidad a la que ajustemos el efecto ($Depth$) y que variará entre 0 y 3 ms. Además dicho efecto varía de forma senoidal con una frecuencia entre 0.1 y 5 Hz. Con la variable $Fmod$ introducimos el valor en Hz de la frecuencia de la señal senoidal que variará el retardo. Con estos valores crearemos el vector MOD que pasaremos como variable a la función **stdelay** por medio de la ecuación (45).

$$MOD = sin(2\pi tFmod)) \tag{44}$$

Para crear el *vibrato* utilizaremos la función **y = vibrato(x,Fs,Depth,Fmod)**. En esta función se toman como valores $BL = 0$, $FF = 1$ y $FB = 0$. Con la variable $Fmod$ introducimos el valor en Hz de la frecuencia de la señal senoidal que variará el retardo.

3.4.5. Flanger

El *flanger* es un efecto de sonido que produce un característico sonido metalizado oscilante. A diferencia del vibrato, en este caso se mezcla la señal directa con las señal retardada. La salida del retardo también se utiliza para realimentar el sistema.

En este tampoco existe un retardo fijo ($Delay = 0$). La variación del retardo se procuce ajustando la profundidad entre 0 y 2 ms. Además dicho efecto varía de forma senoidal con una frecuencia $Fmod$ entre 0.1 y 1 Hz. El vector MOD se calcula de la misma forma que en el caso del *vibrato*

Para crear este efecto utilizaremos la función **y = flanger(x,Fs,Depth,Fmod)**. En esta función se toman como valores $BL = 0{,}7071$, $FF = 0{,}7071$ y $FB = -0{,}7071$.

3.4.6. Chorus

El *chorus* es un efecto que a partir del sonido de un solo instrumentista trata de simular la sensación de que son dos o más pretendiendo tocar lo mismo (tocar en unísono). La señal directa se mezcla con una señal de retardo aleatorio. La variación de tono consecuente, se traduce en la percepción de que existe más de una fuente sonora.

Según [Zol02] el retardo fijo puede variar entre 1 y 30 ms. La profundidad también puede variar entre 1 y 30 ms. En este caso la señal que modula el retardo es un ruido aleatorio que se ha filtrado con un filtro paso-bajo. Para crear el vector MOD hemos utilizado la siguiente linea en Matlab:

```
for c=1:columnas
        for n=1:length(x)
                MOD(n,c)=2*rand-1;
        end;
end;
```

Por medio de estas lineas creamos un vector (o matriz de dos columnas para el caso de señales estéreo) con valores aleatorios entre -1 y 1. Como se observa se ha hecho uso de la función **rand** de Matlab. La variable *columnas* nos indica si se trata de una señal mono ($columnas = 1$) o estéreo ($columnas = 2$). Además, a continuación se filtrara la señal MOD con un filtro paso-bajo con frecuencia de corte en 5 Hz tal y como se describe en la página 71 de [Zol02]. Para ello se utilizará la función **lowpass** descrita en el apartado 3.2.2.

Para crear el *chorus* utilizaremos la función **y = chorus(x,Fs,Delay,Depth)**. En esta función se toman como valores $BL = 1$, $FF = 0{,}7071$ y $FB = 0$.

3.4.7. Doubling

También conocido como *slapback*. Cuando una voz intenta grabar la misma pieza en una segunda pista mientras escucha la primera se dice que se produce el efecto de *doubling*.

Según [Zol02] el retardo fijo puede variar entre 10 y 100 ms y la profundidad entre 1 y 100 ms. El retardo se varía de la misma forma que para *chorus*.

Utilizaremos la función **y = doubling(x,Fs,Delay,Depth)** para crear este efecto. En esta función se toman como valores $BL = 0{,}7071$, $FF = 0{,}7071$ y $FB = 0$.

3.5. Reverberación

En condiciones de escucha normales, el sonido es modificado por el entorno en su trayecto desde la fuente hasta el receptor. Se producen algunos efectos de tipo "espacial" debidos a las características geométricas y físicas del entorno. De modo general, nos referimos a dicho efecto "espacial" como **reverberación**. En un espacio cerrado se crean un número elevado de reflexiones. Los sonidos reflejados varían en función de la absorción de los cerramientos y del aire. La reverberación nos da información sobre las naturalezan y textura de los materiales del entorno.

El parámetro más importante que caracteriza el entorno es el **Tiempo de reverberación o decaimiento):** Es el tiempo que el sonido tarda en extinguirse una vez que la fuente ha dejado de emitirlo y generalmente se mide en segundos. *W.C.Sabine* lo definio como el tiempo que tarda en caer 60 dB. Se puede decir que en general, tiempos cortos emulan generalmente espacios reducidos, y tiempos largos, grandes espacios. En el ejemplo de la figura 27 el sonido tarda 2 segundos en caer desde los -10 dB hasta los -70 dB.

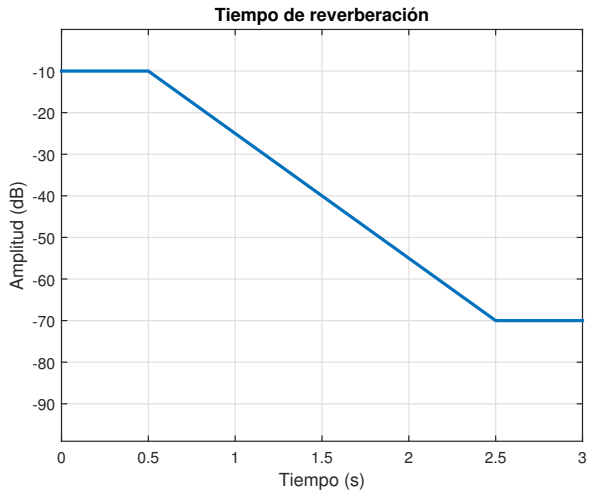

Figura 27: Tiempo de reverberación

Si quisieramos expresar el tiempo de reverberación en número de muestras para una frecuencia de muestreo F_s, utilizaríamos la expresión (46).

$$T_{dM} = F_s \cdot T_d \tag{45}$$

En este trabajo nos centraremos en el desarrollo de la unidad de reverberación basandonos en dos modelos descritos en dos modelos: *El reverberador de Schroeder* y *el reverberador por convolución*

3.5.1. Reverberador de Schroeder

A principios de los años 60, Manfred Schroeder de los laboratorios Bell, propuso un esquema para un reverberador artificial. En el diagrama 28 se puede ver dicho modelo.

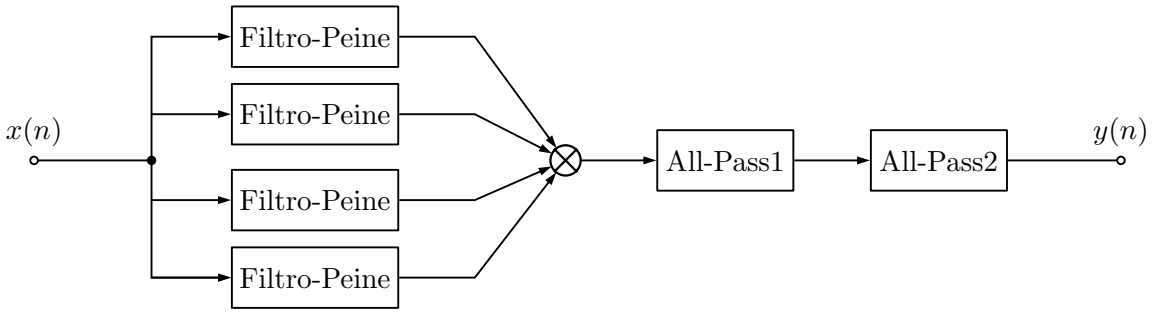

Figura 28: Diagrama de bloques de reverberador Schroeder

En [Raf09] se describe detalladamente dicho esquema. El reverberador de Schroeder consta de cuatro filtros-peine IIR en paralelo seguidos por dos filtros pasa-todo recursivos en serie.

El diagrama de bloques del filtro peine utilizado se puede ver en la figura 29.

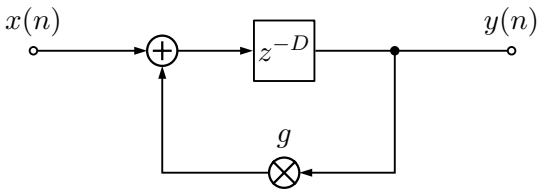

Figura 29: Diagrama de bloques de filtro peine

La ecuación en diferencias del filtro peine se muestra en la ecuación (47)

$$y(n) = x(n - D) + g \cdot y(n - D) \tag{46}$$

La respuesta al impulso de este tipo de filtro se puede apreciar en la figura 30. Como se puede observar, la señal de salida va dreciendo en cada impulso retardado en D muestras. Si el valor decrece en mayor proporción, el valor del tiempo de reverberación T_d final será más corto. En consecuencia, si decrece en menor proporción el tiempo será mayor.

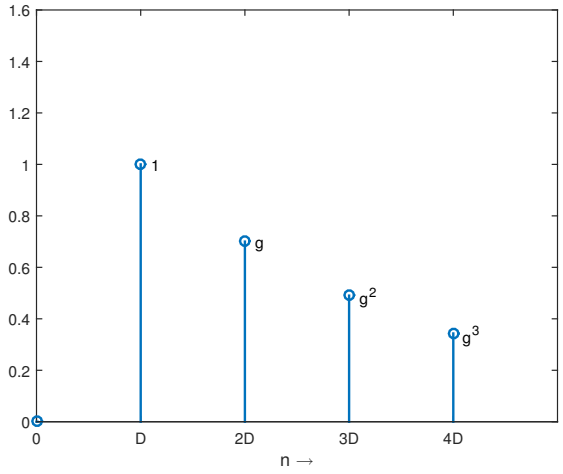

Figura 30: Respuesta al impulso del filtro peine

Según [Raf09] los valores de los retardos de los filtros peine deben estar entre 30 ms y 50 ms. Nuestro reverberador utilizará los valores de 29,7 ms, 37,1 ms, 41,1 ms y 43,7 ms para cada uno de los filtros peine. Si calculamos el retardo en número de muestras para una frecuencia de muestreo de 44,1 kHz, obtendríamos los valores de 1310, 1636, 1813 y 1927.

En nuestro caso, el usuario podrá ajustar tiempo de reverberación deseado a través de la variable T_d. Como se ve en la ecuación (48), la ganancia g del filtro está relacionado con el retardo D y de T_d. Por lo tanto, la solución del problema consistirá en calcular la ganancia g para cada filtro.

$$T_d = \frac{T_{dM}}{F_s} = D \cdot \left(\frac{\log 10^{-3}}{\log g} \right) \tag{47}$$

Por otro lado, el diagrama de bloques del filtro pasa-todo que vamos a utilizar se puede apreciar en la figura 31.

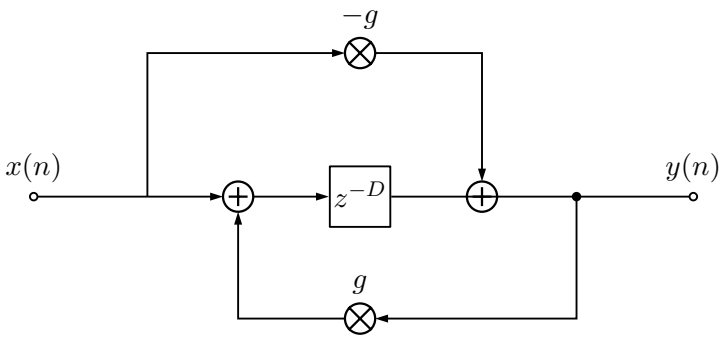

Figura 31: Diagrama de bloques de filtro pasa-todo

La ecuación en diferencias del filtro pasa-todo de primer orden se define como (49)

$$y(n) = x(n - D) - g \cdot x(n) + g \cdot y(n - D) \tag{48}$$

La repuesta al impulso de este tipo de filtros se muestra en la figura 32.

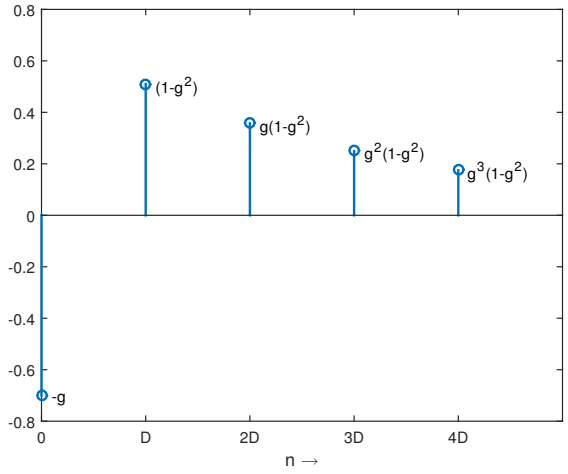

Figura 32: Respuesta al impulso del filtro pasa-todo

En cuanto a los dos filtros pasa-todo, en [Raf09] se proponen los valores de 96,83 ms y 32,92 ms para el retardo. Los valores del retardo en nº de muestras D son de 4270 y 1452 respectivamente.

En cuanto al tiempo de reverberación T_d de estos filtros, su valor es fijo. Utilizaremos los valores de 5 ms y 1,7 ms tal y como se proponen en [Raf09].

La relación entre g, T_d y D cambia un poco respecto a como se hacía en los filtros de tipo peine anteriores. La relación se aprecia en la ecuación (50).

$$T_{dM} = F_s \cdot T_d = D \cdot \left(\frac{\log 10^{-3}}{\log g} - 1 \right) \tag{49}$$

Como se comentado, el valor T_d de los filtros pasa-todo no es ajustable. En las ecuaciones (51) y (52) se calculan las ganancias de los dos filtros.

$$g_1 = 10^{\frac{-3D_1}{F_s T_{d1} + D_1}} = 1,403818592 \cdot 10^{-3} \tag{50}$$

$$g_2 = 10^{\frac{-3D_2}{F_s T_{d2} + D_2}} = 1,403756225 \cdot 10^{-3} \tag{51}$$

La función **y = reverbSchroeder(x,Fs,Td)** realiza todos los cálculos descritos anteriormente y aplica el efecto de reverberación con un tiempo T_d a la señal entrada x.

3.5.2. Reverberador por convolución

En la sección 2.3 se ha explicado el concepto de *convolución*. Para aplicar el efecto de reverberación mediante dicha técnica, necesitamos tener una base de datos de respuestas al impulso de distintos recintos.

Dichas respuestas se obtienen simplemente, grabando el sonido de un ruido impulsivo (*explosión de globo, disparo de una pistola...*). El sonido grabado debe ser necesariamente **mono** y de formato *WAV* para poder utilizarlo en nuestro módulo de efectos. La frecuencia de muestreo no tiene por qué tener el mismo valor que la de la señal de entrada. El algoritmo utilizado compara las dos y convierte la frecuencia de muestreo F_{sh} de la respuesta al impulso , a la frecuencia de muestreo F_s de la señal de entrada.

Dicha conversión se realiza por medio de la función **y = srconv(x,fsin,fsout)** disponible en la página web de Mathworks [Math] y creada por Lawrence Rabiner. Esta función modifica la señal de entrada x y con una frecuencia de muestreo $fsin$, de forma que se reproduce correctamente a una frecuencia de muestreo $fsout$.

La función **y = reverbConv(x,Fs,h,Fsh)** aplica la reverberación por convolución a la señal x teniendo como respuesta al impulso la señal h.

4. Módulo de efectos *TFG_Efectos.m*

Para la creación de los distintos efectos descritos en los apartados anteriores se ha utilizado la herramienta software **Matlab** en su versión R2014b de 32 bits. Se han creado distintos archivos que aplican los efectos a las señales de entrada. Además también se han creado otras funciones que son necesarias para la correcta ejecución de las funciones principales. En la tabla 6 se muestra un resumen de todas las funciones implementadas. Los archivos correspondientes tienen el mismo nombre que la función y la extensión **.m*.

Los efectos se aplican siguiendo el orden mostrado en la figura 33.

Efecto	Función
Compresor	y=comp(x,Fs,at,rt,CT,R)
Filtro Shelving bajas frecuencias	y=lowshelving(x,Fs,G,Fc)
Filtro Shelving altas frecuencias	y=highshelving(x,Fs,G,Fc)
Filtro Peak	y=peakfilt(x,Fs,G,Fc,Fb)
Eco	y=eco(x,Fs,Delay,g)
Doubling	y=doubling(x,Fs,Delay,Depth)
Chorus	y=chorus(x,Fs,Delay,Depth)
Flanger	y=flanger(x,Fs,Depth,Fmod)
Vibrato	y=vibrato(x,Fs,Depth,Fmod)
Wah-Wah	y=wahwah(g,Fmin,Fmax,Fw,Fs,x)
Reverberador método Schroeder	y=reverbSchroeder(x,Fs,Td)
Reverberador método convolución	y=reverbConv(x,Fs,h,Fsh)
Filtro paso-bajo	y=lowpass(x,Fc,Fs)
Modificador de frecuencia de muestreo	y=srconv(x,fsin,fsout)
Unidad retardo estándar	y=stdelay(x,Fs,BL,FF,FB,Delay,Depth,MOD)

Tabla 6: Resumen de funciones

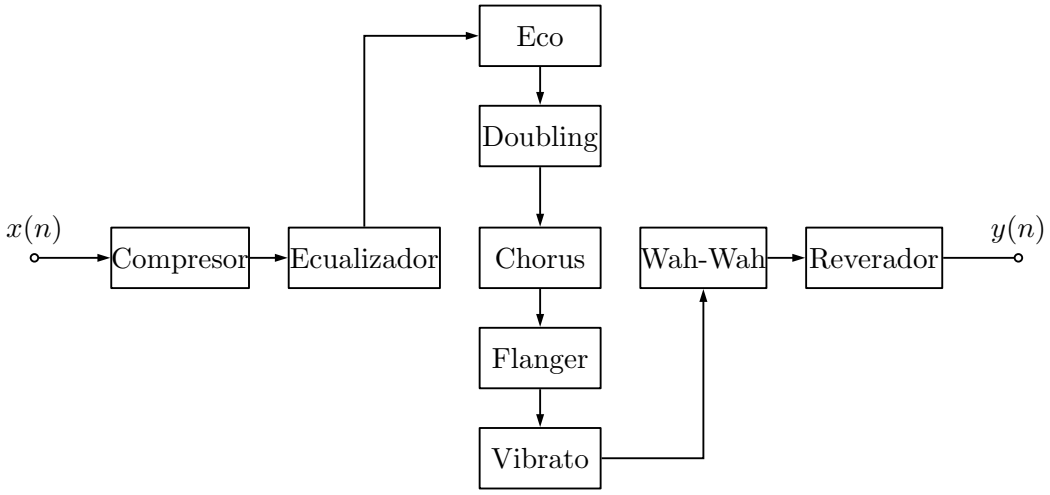

Figura 33: Orden de aplicación de lo efectos

La aplicación funciona tanto para archivos de audio mono como estéreo. Para las señales estereo la función se aplica a los dos canales de la señal. Esto es así para todos los efectos excepto en el compresor. La forma de resolverlo se ha explicado en el apartado correspondiente.

En lo Anexos se muestran los códigos de todas las funciones mencionadas.

4.1. Interfaz gráfica de usuario

Para poder interactuar con las funciones definidas para cada efecto se ha desarrollado una interfaz gráfica de usuario mediante la herramienta GUIDE de Matlab. Como resultado se han creado los archivos *TFG_Efectos.fig* y *TFG_Efectos.m*. El módulo de efectos se inicia ejecutando el archivo *TFG_Efectos.m* en un entorno Matlab. Para la correcta ejecución, es necesario tener todos los archivos mencionados en la tabla 6 además del archivo *TFG_Efectos.fig* en la misma carpeta que el script *TFG_Efectos.m*.

Figura 34: Módulo de efectos

El funcionamiento del módulo de efectos es muy intuitivo. Cada efecto lleva incorporado un "checkbox" que permite activar o desactivar el efecto. También dispone de dos botones con los que podemos escuchar la señal original o modificada. Por último, con el botón *Guardar* se crea otro archivo *WAV* con los efectos aplicados.

En los siguientes apartados se muestra un resumen de cómo se han utilizado los distintos elementos en el diseño mediante GUIDE destacando los puntos más importantes. Dentro del archivo *TFG_Efectos.m* cada uno de ellos dispone de unas lineas dedicadas a las que se puede acceder directamente como se vé en la figura 35.

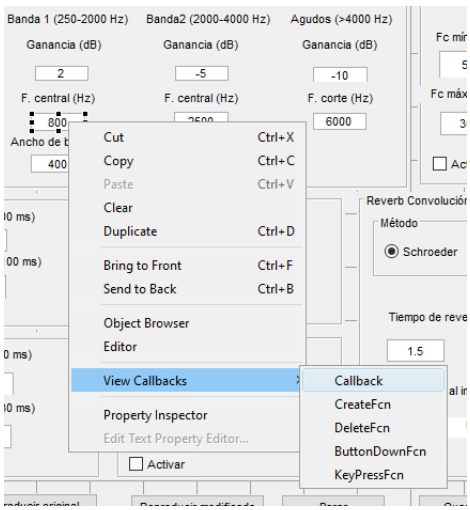

Figura 35: Acceso al código mediante View Callbacks

4.1.1. *Edit Text*

Las variables que utilizan las funciones se introducen mediante los cuadros *Edit Text*. Simplemente se introduce el valor de la variable mediante el teclado. Sin embargo, se han configurado para que

solamente permitan introducir los valores para un rango fijado. En el siguiente ejemplo se muestra el código para la *Edit Text* que introduce el valor de la frecuencia central del ecualizador paramétrico. En primer lugar, se declara $Fc2$ como variable global. Es decir, podrá ser utilizada por otras funciones en cualquier parte del programa. A continuación se convierte el dato introducido de formato "string" al formato numérico "double". En las siguientes tres lineas se comprueba que el valor introducido sea un número ("isnan") y que esté entre 250 Hz y 2000 Hz. La variable del ejemplo será utilizada por la función **y = peakfilt(x,Fs,G,Fc,Fb)** donde corresponderá a la frecuencia central Fc del filtro.

```
global Fc2
Fc2 = str2double(get(hObject,'String'));
if isnan (Fc2) || Fc2 < 250 || 2000 < Fc2
    msgbox('La frecuencia central debe estar entre 250 y 2000 Hz')
end
```

Este es el procedimiento que se ha utilizado en todas las *Edit Text* del programa aunque logicamente, las condiciones cambiarán para cada variable.

4.1.2. *Static Text*

Otros elementos utilizados en el diseño de la interfaz gráfica son las*Static Text*. Simplente se trata de cuadros de texto que permiten introducir notas para una mejor compresión del programa. En nuestro caso los hemos utilizada para especificar los rangos de valores. Dichos cuadros se editan desde el ("Property Inspector") de GUIDE tal y como se vé en la figura 36.

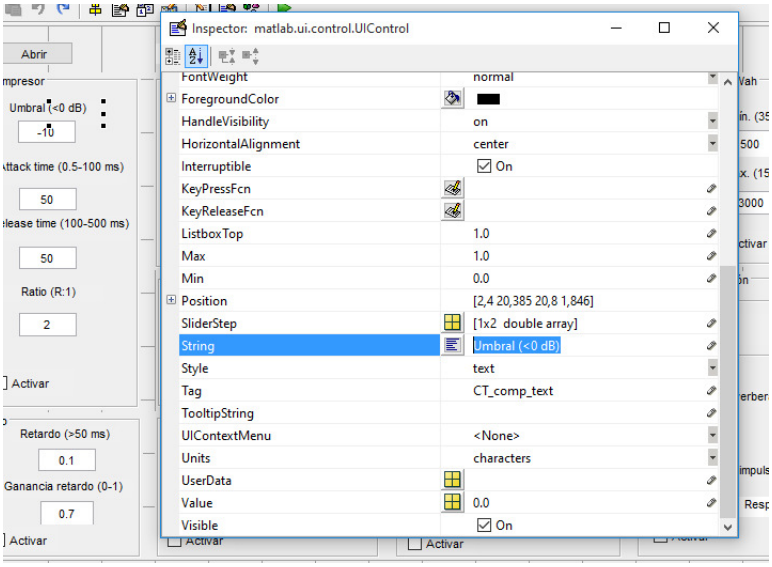

Figura 36: Edición de los cuadros de texto

4.1.3. *Check Box*

También se han introducido los "checkbox" o casillas para activar o desactivar los distintos efectos. En el ejemplo se muestran las lineas del "checkbox" del compresor. En primer lugar se declara la variable *comp_estado* como global y en la segunda linea se le asigna el valor de 0 o 1. Dicho valor dependerá del estado en el que se encuentra la casilla de activación del compresor. Si está activado valdrá 1 y si no es así 0.

```
global comp_estado
comp_estado = get(handles.comp_check,'Value');
```

Las variables para cada efecto siempre tomarán la forma de *NombreEfecto_estado* y se utilizarán para aplicar los efectos mediante el bucle condicional **if** como se verá más adelante.

4.1.4. *Push Button*

Para interactuar con el programa también se han utilizado los botones o "Push Button". Se trata de elementos que al pulsarlos ejecutan el código específico creado para ellos.

- El primer boton que tendremos que utilizar será el de "Abrir". Dicho botón abre un explorador de archivos y permite seleccionar un audio WAV. El archivo seleccionado se guardará en la variable x mediante el comando *[x, Fs] = wavread(fullpathname)*. Además crearemos la variable *columnas* con el número de columnas de la señal de entrada. Esto nos servirá para detectar si la señal es mono o estereo. Si tiene una columna será mono y si tiene 2 estéreo. También se crearán los objetos *playerx* y *playery* que se utilizarán para reproducir las señales originales o modificadas respectivamente.

```
...
function abrir_Callback(hObject, eventdata, handles)
...
[filename, pathname]=uigetfile({'*.wav'},'File Selector');
fullpathname = strcat(pathname, filename);
[ x, Fs ] = wavread(fullpathname);
columnas = size(x,2);
set(handles.archivo,'string',fullpathname);
playerx = audioplayer(x,Fs);
playery = audioplayer(x,Fs);
...
```

  Si seleccionamos el reverberador convolucional tendremos que seleccionar una respuesta al impulso para poder aplicar la función **y = reverbConv(x,Fs,h,Fsh)**. Para ello tenemos otro botón "Abrir" para la reverberación. En este caso la respuesta al impulso se guarda en la variable h.

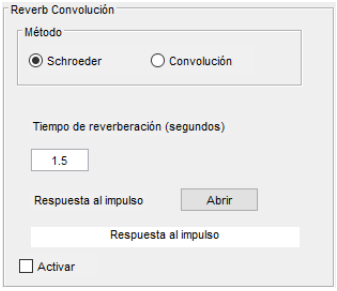

Figura 37: Botón "Abrir" de la reverberación

- En nuestro caso el botón más importante será el de "Aplicar". En la siguiente muestra se incluyen solamente las primeras lineas del código de "Aplicar" ya que el proceso es el mismo para todos los ejemplos. En primer lugar se verifica si la variable de la casilla de activación del efecto está activada o no (si vale 0 o 1). En el caso de se que esté activada se aplica la función correspondiente a la señal de entrada y tomando las variables introducidas en las "Edit Text" correspondientes. En el caso de que no este activada (*comp_estado=0*), se copia la señal de entrada a la variable de la salida del efecto ($y\_comp = x$).

La variable de salida de cada efecto pasa a ser la señal de entrada del efecto siguiendo siguiendo el ordén mostrado en el diagrama 33. En el ejemplo se observa que la salida del compresor (*y_comp* = *x*) es la señal de entrada del filtro shelving de bajas frecuencias del ecualizador paramétrico (*y_ec1 = lowshelving(y_comp,Fs,G1,Fc1*).

```
...
function aplicar_Callback(hObject, eventdata, handles)
...
global comp_estado x Fs columnas y_comp at_comp rt_comp CT_comp R_comp
if comp_estado == 1
    y_comp = comp(x,Fs,at_comp,rt_comp,CT_comp,R_comp);
else
    y_comp = x;
end
%%%%%
global eq_estado y_ec y_ec1 y_ec2 y_ec3 G1 Fc1 G2 Fc2 Fb2 G3 Fc3 Fb3 G4 Fc4
if eq_estado == 1
    y_ec1  = lowshelving(y_comp,Fs,G1,Fc1);
    y_ec2  = peakfilt(y_ec1,Fs,G2,Fc2,Fb2);
    y_ec3  = peakfilt(y_ec2,Fs,G3,Fc3,Fb3);
    y_ec   = highshelving(y_ec3,Fs,G4,Fc4);
else
    y_ec = y_comp;
end

...
Se repite el proceso para todos los efectos
...

%%%%%
global y playery
y = y_reverb;
playery = audioplayer(y,Fs);
...
```

Por último, se guarda la salida del último efecto aplicado (en nuestro caso el reverberador) en la variable *y* y en el objeto *playery*.

- Con los botones "Reproducir original", "Reproducir modificado" y "Parar" se podrá controlar la reprodución de los audios mediante las funciones *play* o *stop*.

```
...
function original_Callback(hObject, eventdata, handles)
...
global playerx
play(playerx);

...
function modificado_Callback(hObject, eventdata, handles)
...
global playery
play(playery);

...
function parar_Callback(hObject, eventdata, handles)
...
global playerx playery
stop(playerx);
stop(playery);
```

- Con el botón "Reset" asignaremos el valor 0 a todas las variables. Además se reflejarán los canbios en los correspondientes "Edit Text". En el ejemplo se muestran únicamente las lineas de las variables del compresor.

```
...
function reset_Callback(hObject, eventdata, handles)
...
global CT_comp at_comp rt_comp R_comp
CT_comp = 0;
at_comp = 0;
rt_comp = 0;
R_comp = 0;
set(handles.CT_comp,'string',num2str(CT_comp));
set(handles.at_comp,'string',num2str(at_comp));
set(handles.rt_comp,'string',num2str(rt_comp));
set(handles.R_comp,'string',num2str(R_comp));
...
```

- Con "Guardar" crearemos un archivo WAV con los efectos aplicados a través de la función *wavwrite*.

```
...
function cambios_Callback(hObject, eventdata, handles)
...
global y Fs
[file,path] = uiputfile('*.wav','Guardar archivo');
fullpathname = strcat(path, file);
wavwrite(y,Fs,fullpathname);
```

- Por último, podremos salir del programa en cualquier momento por medio de "Salir".

```
...
function salir_Callback(hObject, eventdata, handles)
...
clearvars -global
close TFG_efectos
...
```

4.1.5. *Radio Button* y *Button Group*

En el caso del reverberador digital habrá que seleccionar entre el basado en el esquema de Schroeder y el convolucional. No es posible seleccionar los dos simultáneamente. Para ello se crea un elemento "Radio Button" para cada uno de ellos y se agrupan mediante "Button Group". Con las siguientes lineas de código se consigue conmutar el valor de las variable *radio_schro* y *radio_conv*. Si se desea aplicar la reverberación por el método de Schroeder y se selecciona su *Radio Button* correspondiente, las variables tomarán los valores $radio\_schro = 1$ y $radio\_conv = 0$. Si lo que se desea es aplicar el método de la convolución, las variables tomarán los valores $radio\_schro = 0$ y $radio\_conv = 1$.

```
...
function radio_schro_Callback(hObject, eventdata, handles)
...

...
function radio_conv_Callback(hObject, eventdata, handles)
...

...
function uibuttongroup3_SelectionChangedFcn(hObject, eventdata, handles)
...
global radio_schro radio_conv
switch get(eventdata.NewValue,'Tag')
```

```
    case 'radio_schro'
        radio_schro = 1;
        radio_conv = 0;
    case 'radio_conv'
        radio_conv = 1;
        radio_schro = 0;
end
...
```

4.2. Instalador *TFG_Efectos*

Además de poder ejecutar el archivo *TFG_Efectos.m* en sistemas que tengan instalado alguna versión de Matlab, tambíen se ha creado un instalador para poder ejecutarlo en sistemas que no lo tengan. Para poder ejecutar el archivo *Instalador_TFG_Efectos.exe* es necesario tener conexión a internet. Esto se debe a que es necesario conectarse con la web de Mathworks para poder instalar el entorno de ejecución *Matlab Compiler Runtime*.

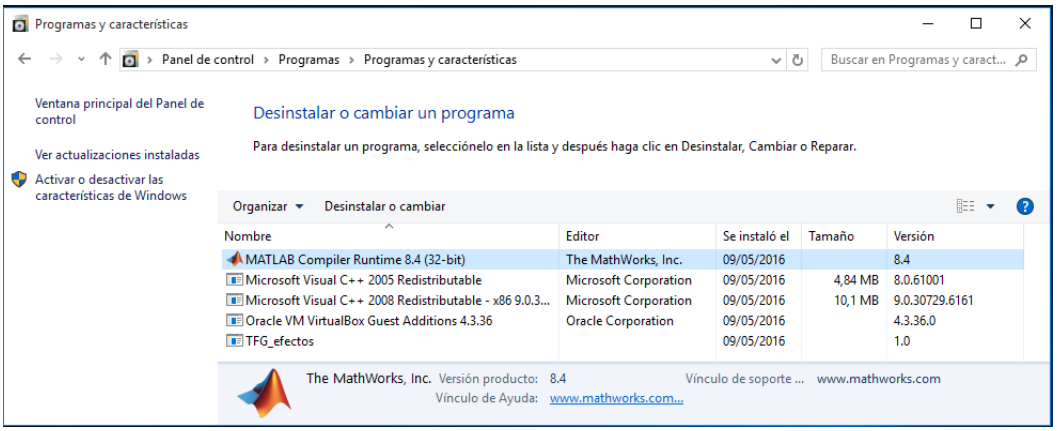

Figura 38: *Matlab Compiler Runtime*

Si no se dispone de conexión a internet, se puede instalar el módulo de efectos utilizando el archivo *Instalador_mcr_TFG_Efectos.exe*. Este archivo instala directamente el *Matlab Compiler Runtime* en el sistema.

5. Conclusiones

Con el desarrollo de los últimos años de la tecnología informática, la implementación de efectos digitales de audio por software se ha convertido en una alternativa práctica y barata a los equipos tradicionales analógicos. En la actualidad existen infinidad de plugins y DSPs programables que permiten la aplicación de los efectos. El objetivo de este trabajo no ha sido obtener un producto novedoso ni mejor que los existentes. Sin embargo, puede ser útil y pedagógico para aquel que se acerca por primera vez al mundo del desarrollo de efectos digitales de audio por software.

En cuanto a las mejoras, se podrían introduccir muchas. Para empezar, se pueden añadir los efectos que se deseen. En la bibliografía citada se pueden encontrar las bases teóricas de muchisímos efectos.

Un ámbito en el que se pueden encontrar multitud de modelos es el de la reverberación. En este trabajo solamente hemos implementado el reverberador de Schroeder y el convolucional. Pero

existen muchos más. Al fin y al cabo, el diseño de un sistema reverberador tiene en gran medida una componente artística que implica una gran variedad en las soluciones.

Por otra parte, el módulo implementado trabaja con grabaciones de audio y en consecuencia no permite aplicar los efectos en tiempo real. Se podría ampliar su ámbito de uso (ej. audio en directo) desarrollando un módulo que permita aplicar los efectos en tiempo real.

Estas son solamente algunas de las mejoras que se podrían introducir en el módulo *TFG_Efectos* o en trabajos futuros similares.

6. Bibliografía

[Zol02] DAFX - Digital Audio Effects de *Udo Zölzer*, John Wiley & Sons, Ltd

[Zol11] DAFX - Digital Audio Effects, 2nd edition de *Udo Zölzer*, John Wiley & Sons, Ltd

[Zol08] Digital Audio Signal Processing, 2nd edition de *Udo Zölzer*, John Wiley & Sons, Ltd

[Dat97] Effect Design Part 2: Delay-Line Modulation and Chorus de *Jon Dattorro*, J. Audio Eng Sot., Vol 45, No. 10, 1997 October

[Raf09] Technical Report NWU-EECS-09-08. A Digital Reverberator Controlled through Measures of the Reverberation de *Zafar Rafii y Bryan Pardo*, Electrical Engineering and Computer Science Department, Northwestern university

[Math] Página web de Mathworks

[Mars] Página web personal de David Marshall: http://www.cs.cf.ac.uk/Dave/CM0268/

[CryB] Manual Cry baby GCB 95, Dunlop Manufacturing, Inc.

A. Ficheros

En este anexo se muestra un resumen de los ficheros anexados a esta memoria:

- *comp.m.*
- *chorus.m.*
- *doubling.m.*
- *eco.m*
- *flanger.m*
- *highshelving.m*
- *Instalador_TFG_efectos.exe*
- *Instalador_mcr_TFG_efectos.exe*
- *lowpass.m*
- *lowshelving.m*
- *peakfilt.m*
- *reverbConv.m*
- *reverbSchroeder.m*
- *srconv.m*
- *stdelay.m*
- *TFG_efectos.fig*
- *TFG_efectos.m*
- *vibrato.m*
- *wahwah.m*