Proyecto Final: Diseño de filtros digitales

Rodrigo González Zarrabal
Universidad Anáhuac Xalapa - Ing. Mecatrónica

***Abstract*—Para este proyecto intentamos mezclar y luego filtrar diferentes señales de audio mediante filtros digitales.**

I. Introducción

Para poder aplicar filtros digitales a una señal con el fin de procesarla, es necesario comprender ciertas herramientas y artificios matemáticos que conforman el sustento teórico del filtrado digital.

A. Transformada rápida de Fourier (FFT)

La transformada de Fourier esencialmente, toma una señal y la descompone en ondas sinusoidales de diferentes amplitudes y frecuencias.

El teorema de Fourier afirma que cualquier forma de onda en el dominio del tiempo puede ser representada por la suma acumulada de senos y cosenos.

Supongamos que tenemos una señal descrita por un conjunto de n (potencia de dos) pares de datos (t_j, x_j) igualmente espaciados en el tiempo, por un intervalo Δt desde $t = 0$ hasta $t_{final} = (n-1)\Delta t$. La inversa del intervalo Δt, se denomina frecuencia de muestreo f_s. De modo que el vector de tiempos es $t = \frac{(0:n-1)}{f_s}$.

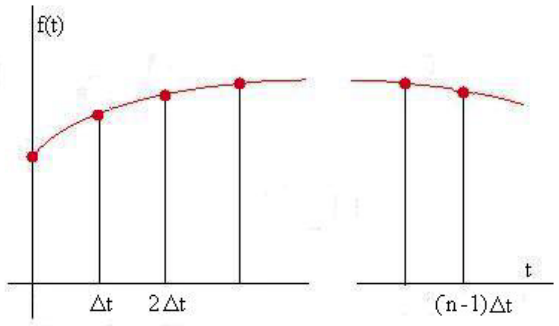

Fig. 1. Señal discreta

La transformada rápida de Fourier, conocida por la abreviatura FFT (del inglés Fast Fourier Transform) es un algoritmo eficiente que permite calcular la transformada de Fourier discreta (DFT) y su inversa. La FFT es de gran importancia en una amplia variedad de aplicaciones, desde el tratamiento digital de señales y filtrado digital en general a la resolución de ecuaciones en derivadas parciales o los algoritmos de multiplicación rápida de grandes enteros. Cuando se habla del tratamiento digital de señales, el algoritmo FFT impone algunas limitaciones en la señal y en el espectro resultante ya que la señal muestreada y que se va a transformar debe consistir de un número de muestras igual a una potencia de dos. La mayoría de los analizadores de FFT permiten la transformación de 512, 1024, 2048 o 4096 muestras. El rango de frecuencias cubierto por el análisis FFT depende de la cantidad de muestras recogidas y de la proporción de muestreo.

Sea $x(n)$ una señal aperiódica discreta en el tiempo. La transformada discreta de Fourier (DFT, por sus siglas en inglés) de esta señal se define como:

$$X(k) = \sum_{n=0}^{N-1} x(n) e^{-j2\pi k \frac{n}{N}}, \qquad k = 0, ..., N-1.$$

en la cual $X(k)$ es un conjunto de números complejos. La evaluación directa de esa fórmula requiere N^2 operaciones aritméticas, pero con un algoritmo FFT se puede obtener el mismo resultado con solo $N \log N$ operaciones. En general, dichos algoritmos dependen de la factorización de n pero, al contrario de lo que frecuentemente se cree, existen FFTs para cualquier n, incluso con n primo.

La idea que permite esta optimización, es la descomposición de la transformada a tratar en otras más simples y éstas a su vez hasta llegar a transformadas de 2 elementos donde k puede tomar los valores 0 y 1. Una vez resueltas las transformadas más simples hay que agruparlas en otras de nivel superior que deben resolverse de nuevo y así sucesivamente hasta llegar al nivel más alto. Al final de este proceso, los resultados obtenidos deben reordenarse.[1]

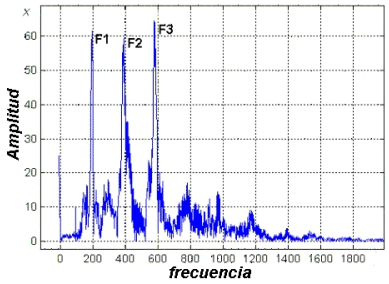

Fig. 2. Espectro de frecuencia

B. *Filtros Digitales*

Los filtros digitales operan sobre las señales que representan al sonido, transformando sus muestras a través de un algoritmo, y convirtiéndolas en nuevas secuencias numéricas cuyas características son distintas a las originales (señal filtrada).

Existen diversas clasificaciones de los filtros. Básicamente, podemos distinguir cuatro tipos: pasa bajos, pasa altos, pasa banda y rechazo de banda.

El filtro pasa bajos, como su nombre lo indica, deja pasar las componentes más graves del sonido, y retiene a las componentes agudas cuya frecuencia es superior a la frecuencia de corte del filtro. Su versión más elemental se obtiene sumando a una señal digital una copia de sí misma, retrasada en una muestra

Un filtro paso alto (HPF) es un tipo de filtro electrónico en cuya respuesta en frecuencia se atenúan los componentes de baja frecuencia pero no los de alta frecuencia, éstas incluso pueden amplificarse en los filtros activos.

Un filtro paso bajo corresponde a un filtro electrónico caracterizado por permitir el paso de las frecuencias más bajas y atenuar las frecuencias más altas.

El filtro requiere de dos terminales de entrada y dos de salida, de una caja negra, también denominada cuadripolo o bipuerto, así todas las frecuencias se pueden presentar a la entrada, pero a la salida solo estarán presentes las que permita pasar el filtro. De la teoría se obtiene que los filtros están caracterizados por sus funciones de transferencia, así cualquier configuración de elementos activos o pasivos que consigan cierta función de transferencia serán considerados un filtro de cierto tipo.

Un filtro paso banda es un tipo de filtro electrónico que deja pasar un determinado rango de frecuencias de una señal y atenúa el paso del resto. Se puede construir un filtro paso banda puede ser usar un filtro paso bajo en serie con un filtro paso alto entre los que hay un rango de frecuencias que ambos dejan pasar. Para ello, es importante tener en cuenta que la frecuencia de corte del paso bajo sea mayor que la del paso alto, a fin de que la respuesta global sea paso banda (esto es, que haya solapamiento entre ambas respuestas en frecuencia).[2]

II. Desarrollo

El programa fue creado mediante MATLAB App Designer, al ejecutar la aplicación aparece la ventana principal donde se puede realizar la mezcla de las señales.

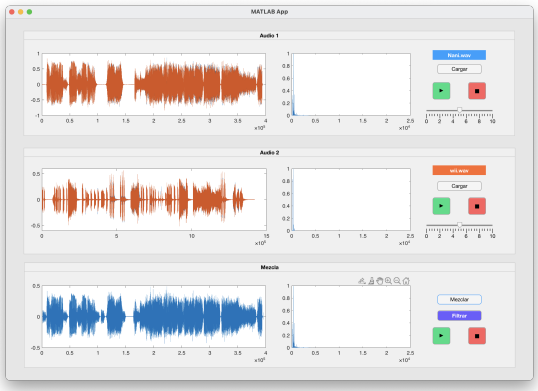

Fig. 3. Ventana principal de la aplicación

Para el proceso de diseño de la aplicación se comienza por la recuperación de datos. Para esto se carga dos archivos de audio en formato wav y se captura la señal y su frecuencia de muestreo.

```
function cargarS1ButtonPushed(app, event)
    [file,path] = uigetfile('*.wav');
    if isequal(file,0)
        app.Track1.Text = 'Audio no seleccionado';
    else
        [app.s1,app.Fs] = audioread(file);
         app.Track1.Text = file;

        cla(app.S1,"reset");
        plot(app.S1,app.s1);

        app.ns1 = length(app.s1);
        nfft = app.Fs;
        espectro = fft(app.s1,nfft);
        ff = espectro(1:nfft/2);
        ff = ff/max(ff);

        f = (0:nfft/2-1)*app.Fs/nfft;
        my = abs(ff).^2;

        cla(app.S1_fft,"reset");

        plot(app.S1_fft,f,my);
    end
```

Fig. 4. Carga de los archivos de audio

Una vez que se cargo la señal a trabajar se procede a graficar la señal en dominio del tiempo y su espectro de frecuencia.

Aquí también se crea el objeto reproductor de audio para permitir escuchar la señal.

La transformada de Fourier se normaliza a 1 de manera que el pico máximo tena magnitud unitaria, esto no afecta la proporción de la gráfica pero permite visualizar de mejor manera los ejes de la misma.

```
function mezclar(app,s,fq)
    app.mezcla = audioplayer(s,fq);

    cla(app.mezclaS1S2,"reset");
    plot(app.mezclaS1S2,s);

    cla(app.mezclaS1S2_fft,"reset");
    nfft = fq;
    espectro = fft(s,nfft);
    ff = espectro(1:nfft/2);
    ff = ff/max(ff);
    f = (0:nfft/2-1)*fq/nfft;
    my = abs(ff).^2;

   plot(app.mezclaS1S2_fft,f,my)
end
```

Fig. 5. Gráficas de la señal

Estos procedimientos se repiten de manera idéntica en el canal 1 y en el canal 2.

El el canal de mezclado se procede a realizar la suma, pero al almacenar las respectivas señales en vectores la suma no es posible a menos que estos tengan las mismas dimensiones. Para resolver este problema se toman las longitudes individuales de cada vector y se calcula el mínimo valor. Posteriormente se genera un vector de 1 con la extensión mínima de los vectores, de esta manera se multiplican las señales originales por dicho vector resultando en dos señales nuevas de la misma longitud que pueden sumarse sin problemas.

Aquí también se recupera el valor de los sliders de volumen, el cual genera un valor entre 0 y 1, y se multiplica por cada una de las señales individuales para ajustar el volumen de cada canal antes de mezclarlo.

```
function MezclarButtonPushed(app, event)
    if isempty(app.s1) || isempty(app.s2)
        disp('Falta un audio para mezclar');
    else
        app.N = min([app.ns1,app.ns2]);
        app.Y = app.s1(1:app.N);
        app.Z = app.s2(1:app.N);

        app.v1 =app.volS1.Value/10;
        app.v2 =app.volS2.Value/10;

        app.mix = (app.Y * app.v1) + (app.Z * app.v2);

        mezclar(app,app.mix,app.Fs);


    end
end
```

Fig. 6. Mezcla de señales

Para reproducir las pistas se genera un objeto reproductor y en cuanto se presiona el botón, se ejecuta el audio.

```
function playS1ButtonPushed(app, event)
    if isempty(app.s1)
        disp('Audio no seleccionado');
    else
        app.v1 =app.volS1.Value/10;

        cla(app.S1,"reset");

        plot(app.S1,app.s1*app.v1);

        app.song1 = audioplayer(app.s1*app.v1,app.Fs);
        app.song1.play();
    end
end
```

Fig. 7. Reproducción de la señal

Una vez que se tienen los audios mezclados se hace una llamada al presionar el botón llamado "Filtrar", dicha llamada abre la segunda ventana donde se puede configurar el filtro.

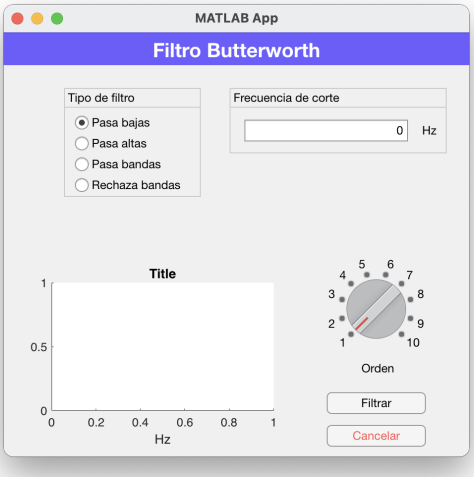

Fig. 8. Ventana de configuración del filtro

En esta nueva ventana se puede escoger el filtro deseado mediante el menú de selección de filtro.

```
function TipodefiltroButtonGroupSelectionChanged(app, event)
    selectedButton = app.TipodefiltroButtonGroup.SelectedObject.Text;
    disp(selectedButton);

    switch selectedButton
        case 'Pasa bajas'
            app.FrecuenciadecortebajaPanel.Visible = 'Off';
            app.FrecuenciadecortealtaPanel.Title = 'Frecuencia de corte';
            app.filtro = 'low';
        case 'Pasa altas'
            app.FrecuenciadecortebajaPanel.Visible = 'Off';
            app.FrecuenciadecortealtaPanel.Title = 'Frecuencia de corte';
            app.filtro = 'high';
        case 'Pasa bandas'
            app.FrecuenciadecortebajaPanel.Visible = 'On';
            app.FrecuenciadecortealtaPanel.Title = 'Frecuencia de corte alta';
            app.filtro = 'bandpass';
        case 'Rechaza bandas'
            app.FrecuenciadecortebajaPanel.Visible = 'On';
            app.FrecuenciadecortealtaPanel.Title = 'Frecuencia de corte alta';
            app.filtro = 'stop';
    end
```

Fig. 9. Selección de filtro

Una vez que se ha seleccionado el filtro deseado se puede modificar el orden del filtro mediante la perilla y también se especifican las frecuencias de corte.

Las frecuencias de corte dependen el filtro seleccionado, si es pasa altos o pasa bajos se define una sola frecuencia, si es pasa bandas o rechaza bandas se escogen dos frecuencias de corte. También cabe mencionar que estas frecuencias deben ser normalizadas a 1, esto es, que debe existir entre valores de 0 y 1, para esto se toma la frecuencia deseada y se divide entre la frecuencia de muestreo entre dos: $\frac{Fc}{\frac{Fs}{2}}$.

```
function KnobValueChanged(app, event)
    value = app.Knob.Value;

     switch app.filtro
         case 'bandpass'
             app.wn = [app.Fcbaja.Value/(app.Fs/2) app.Fcalta.Value/(app.Fs/2)];
         case 'stop'
             app.wn = [app.Fcbaja.Value/(app.Fs/2) app.Fcalta.Value/(app.Fs/2)];
         otherwise
             app.wn = app.Fcalta.Value/(app.Fs/2);
     end


    app.n = str2num(value);
    [app.num,app.den] = butter(app.n,app.wn,app.filtro);
    [app.h,app.w] = freqz(app.num,app.den);
    cla(app.bode,"reset");
    plot(app.bode,app.w,abs(app.h));

end
```

Fig. 10. Configuracion del filtro

El filtro utilizado es un filtro Buterworth mediante la función "butter" la cual toma como parámetros el orden, las frecuencias de corte, y la señal a filtrar. Esta función devuelve los polos y ceros de la función de transferencia del filtro. A partir de estos datos se obtiene los valores para la gráfica del filtro.

Finalmente se aplica el filtro a la señal mezclada y se devuelve la función para que las gráficas respectivas se actualicen con los valores de la nueva señal.

```
function FiltrarButtonPushed(app, event)
    sFiltrada = filter(app.num,app.den,app.mix);
    mezclar(app.CallingApp,sFiltrada,app.Fs)
end
```

Fig. 11. Aplicación del filtro

III. Conclusión

Durante la realización de este proyecto se pudo analizar y observar los efectos que tienen los diferentes tipos de filtros digitales sobre las señales. También pudimos comprender varios conceptos relacionados al procesamiento digital de señales

Referencias

[1] “comprender ffts y funciones ventana.” [Online]. Available: https://www.ni.com/es-mx/innovations/white-papers/06/understanding-ffts-and-windowing.html

[2] “Transformada rápida de fourier,” Nov 2020. [Online]. Available: https://es.wikipedia.org/wiki/Transformada_r$\%C3\%A1pida_de_Fourier$