Sistemas y Señales II
Trabajo Práctico 3 – Diseño de Filtros Analógicos y Digitales
Noviembre 2023

1. Objetivos

Incorporar la utilización de `Matlab` y el entorno `Simulink` como herramientas para analizar, verificar y afianzar los conocimientos adquiridos sobre:

a. Diseño de la plantilla e implementación de un filtro antialiasing,

b. Diseño de distintos tipos de filtros clásicos (Butterworth y Chebyshev),

c. Comparación entre las características de los distintos tipos de filtros: selectividad, orden, distorsión.

d. Diseño de filtros digitales (FIR e IIR).

2. Introducción

Se desea digitalizar una señal de audio analógica captada por un micrófono a partir de una canción que suena en un recinto a través de un procesador digital de señales (DSP). Las especificaciones del DSP indican que no es conveniente utilizar una frecuencia de muestreo superior a $f_s = 12$ kHz.

Para poder simular la presencia de la señal de audio analógica mediante Simulink, se utilizará una señal de audio digital de alta calidad. Dicha señal se encuentra en el archivo `musica_tp3.mat`.

a. Cargue los datos del archivo mediante el comando `load('musica_tp3.mat')`.

b. Observe las variables cargadas en el workspace. ¿Cuánto más grande es la frecuencia de la señal que usamos para simular la señal analógica con respecto a la frecuencia máxima que nos permite utilizar el DSP?

c. Escuche la señal de audio mediante el comando `soundsc(Musica.Data,fo,nbits)`[1]. La reproducción puede detenerse ejecutando el comando `clear sound`.

3. Diseño de filtros analógicos

3.1. Muestreo en crudo

a. En Simulink, arme un esquema de filtrado y muestreo básicos mediante los bloques `From Workspace`, `Transfer function` y `To Workspace`. Configure el bloque `From Workspace` de forma de leer los datos contenidos en la variable `Musica`.

b. Configure el bloque `Transfer Function` para que tome los valores de numerador y denominador desde variables en el espacio de trabajo. Defina dichas variables para que la función transferencia consista simplemente en una ganancia unitaria.

c. Configure las opciones del bloque `To Workspace` de manera que muestree la señal con una frecuencia $f_s = 12$ kHz.

d. Simule el sistema y escuche mediante el comando `soundsc` la señal filtrada. ¿Puede utilizarse este muestreo en crudo para digitalizar la señal analógica? ¿Escucha algo en la señal muestreada que indique que esto no es posible? ¿A qué se debe?

Comandos útiles: `pspectrum(Musica.Data,fo)`

[1] El campo `nbits` representa la *profundidad de bits* de la señal. Por ejemplo, para 16 bits de resolución, se tendrán 2^{16} valores para representar la amplitud de la señal.

3.2. Muestreo inteligente

Para evitar el fenómeno de *aliasing* la señal analógica debe filtrarse mediante un filtro antialiasing para ser correctamente muestreada y digitalizada por el DSP.

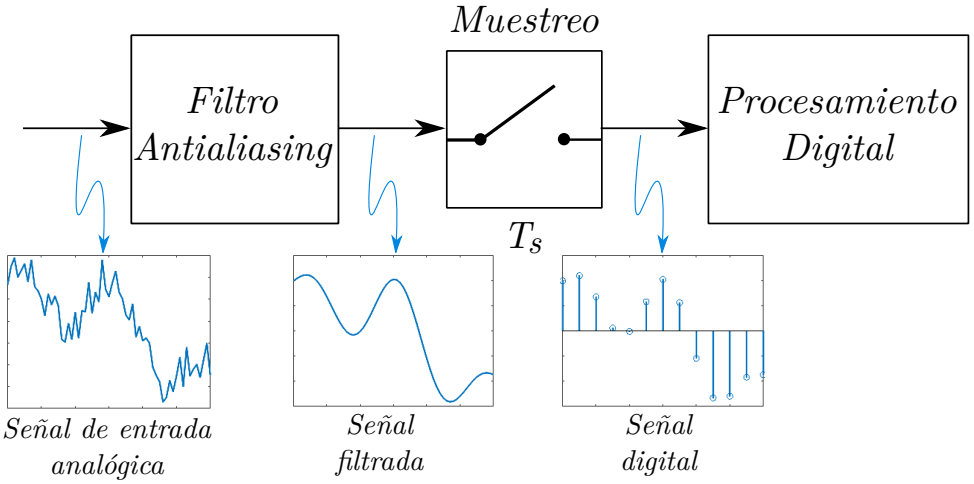

Figura 1: Esquema de filtrado y muestreo.

a. Diseñe una plantilla de un filtro antialiasing para minimizar el aliasing generado al muestrear con una frecuencia $f_s = 12$ kHz. Puede utilizar la función `filterTemplate` provista para verificar el diseño.

A continuación se diseñarán dos tipos de filtros analógicos clásicos, Butterwoth y Chebyshev.

3.2.1. Muestreo inteligente 1. Filtro de Butterworth.

a. Diseñar filtros de Butterworth para minimizar el aliasing para una frecuencia de muestreo de $f_s = 12$ kHz. Asumir una atenuación mínima $\alpha_{\text{mín}} = 20$ dB para una frecuencia de corte de $f_a := f_s/2$. Seleccionar una atenuación máxima para la banda de paso $\alpha_{\text{máx}}$ y varias selectividades $k = \frac{f_p}{f_a}$.

Funciones útiles: `[n,`ω_n`]=buttord(`$\omega_p, \omega_a, \alpha_{\text{máx}}, \alpha_{\text{mín}}$`,'s')`
`[num,den]=butter(n,`ω_n`,'low','s')`

Para comprobar que se cumple la plantilla propuesta puede superponer la gráfica de atenuación del filtro sobre la plantilla dada por `filterTemplate` por medio de los comandos: `[H,w]=freqs(num,den)`, `plot(w,-20*log10(abs(H)))`.

b. Graficar amplitud y fase de la respuesta en frecuencia del filtro para las distintas selectividades seleccionadas en una misma gráfica.[2] Identificar ventajas y desventajas de un filtro antialiasing con mayor o menor selectividad. ¿Qué selectividad resulta la más adecuada?

c. Configurar el bloque `Transfer Function` del modelo Simulink para que implemente alguno de los filtros de Butterworth diseñados. Simular el sistema de filtrado y muestreo. Escuchar el resultado y evaluar las diferencias con el muestreo en crudo. En caso de no ser satisfactorio el resultado, rediseñar el filtro para lograr un mejor resultado.

[2] Para ello puede utilizar los comandos `[H,w]=freqs(num,den)`, `plot(w,abs(H))`, `plot(w,phase(H))`.

3.2.2. Muestreo inteligente 2. Filtro de Chebyshev Tipo 1.

a. Para los mismos parámetros de atenuaciones y selectividades anteriores, diseñar filtros de Chebyshev.

Funciones útiles: `[n,`ω_n`]=cheb1ord(`$\omega_p, \omega_a, \alpha_{\text{máx}}, \alpha_{\text{mín}}$`,'s')`
`[num,den]=cheby1(n,`$\alpha_{\text{máx}}$`,`ω_p`,'low','s')`

b. Graficar amplitud y fase de la respuesta en frecuencia del filtro para las selectividades seleccionadas en una misma figura. Comparar el orden y las características de los filtros obtenidos con lo obtenido para los filtros de Butterworth para cada una de las selectividades. ¿Qué ventajas y desventajas encuentra al comparar ambos filtros?

c. Simular el sistema de filtrado y muestreo correspondiente. Escuchar el resultado y comparar con el filtro Butterworth seleccionado anteriormente.

d. ¿Resulta adecuada la plantilla del filtro antialising diseñada en el punto 3.2.a. para el diseño de filtros de Butterworth y Chebyshev? Justifique.

4. Diseño de un ecualizador básico mediante filtros digitales

Una vez obtenida la señal adecuadamente muestreada a la frecuencia f_s se desea dividirla en 3 bandas: de 0 a 600 Hz, de 600 a 1500 Hz y de 1500 Hz en adelante. Se pide diseñar entonces el ecualizador básico mediante filtros FIR y filtros IIR.

1. **Filtros FIR.**

   a. Diseñar filtros FIR de fase lineal para obtener la separación en bandas deseada empleando ventanas de Hamming. Funciones útiles: `fir1`, `window`.

   b. Graficar las respuestas en frecuencia de los filtros diseñados. Funciones útiles: `fvtool, freqz`.

   c. Filtrar la señal de audio mediante cada uno de los filtros diseñados. Escuchar las señales resultantes. Funciones útiles: `filter`.

   d. Evaluar los efectos del empleo de distintas ventanas y distintos órdenes del filtro.

2. **Filtros IIR.**

   a. Diseñar filtros IIR para obtener la separación en bandas deseada, mediante el diseño de filtros analógicos y aplicación de la transformación bilineal[3].

   b. Filtrar la señal de audio mediante cada uno de los filtros diseñados y escuchar las señales resultantes.

[3] Esto es realizado automáticamente por `Matlab` mediante las funciones `butter, cheby1`. Ver la ayuda para la sintaxis.