UNIVERSIDAD TÉCNICA FEDERICO SANTA MARÍA

Departamento de Electrónica

GUÍA DE LABORATORIO 2
FILTROS DIGITALES
FILTROS FIR E IIR

CURSO	LABORATORIO DE PROCESAMIENTO DIGITAL DE SEÑALES
SIGLA	ELO–385
PROFESOR	MARCO E. RIVERA A.
AYUDANTE	IGNACIO LIZAMA A.

Valparaíso, Marzo de 2009

Guía de Laboratorio 2
Introducción Filtros Digitales: Filtros FIR e IIR

En esta guía se presentan los conceptos básicos y pauta para el desarrollo del segundo laboratorio del curso. En el mundo digital existen dos familias de filtros, conocidos por sus siglas en inglés como filtros **FIR** (Finite Impulse Response) e **IIR** (Infinite Impulse Response), estos últimos corresponden a la discretización de los conocidos filtros análogos, mientras que los primeros son exclusivos del mundo digital. En la práctica los filtros IIR se utilizan para aplicaciones de control o aquellas que pretenden replicar esquemas analógicos. Por otra parte, los filtros FIR resultan muy útiles cuando se requieren filtros de alto orden y el tiempo de procesamiento no es un problema, o cuando se requiere de la característica de fase lineal dentro de la banda de paso. Una vez elegido el tipo de filtro a utilizar, el problema se reduce a encontrar los coeficientes que dan al filtro su característica de respuesta en frecuencia. Hoy en día este proceso se facilita enormemente por la disponibilidad de numerosos programas de software como MatLab, el que se utilizará en esta experiencia, mientras que la implementación experimental se realizará con el DSP TMS320C6711 y el codec de audio. **Este laboratorio tiene como objetivo** familiarizar a los estudiantes con el diseño e implementación de filtros digitales FIR e IIR en un procesador digital de señales, empleando las herramientas de MatLab para determinar los coeficientes y posteriormente aplicándolos a un sistema en tiempo real.

1. Teoría básica Filtros FIR.

Un filtro FIR con longitud *M*, entrada *x(n)* y salida *y(n)* se describe por la ecuación de diferencias:

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

donde $\{b_k\}$ son los coeficientes del filtro. Esta estructura es comúnmente conocida como *promedio móvil*, utilizada por ejemplo para determinar el promedio de notas al final de cada semestre.

Alternativamente, es posible representar la secuencia de salida como la convolución de la respuesta a impulso del sistema *h(n)* con la entrada.

$$y(n) = \sum_{k=0}^{M-1} h_d(k) \cdot x(n-k) \tag{2}$$

donde *x(n)* representa la entrada muestreada, $h_d(n)$ representa la respuesta a impulso ideal del filtro, e *y(n)* es la salida filtrada. Es claro que las ecuaciones (1) y (2) son iguales en su forma y, por lo tanto, se puede decir que $b_k = h_d(k)$, $k = 0, 1, 2,...,M-1$.

En general, la respuesta a impulso $h_d(n)$ para un filtro ideal es infinita en duración y debe ser truncada en algún punto para su manejo en sistemas reales. Para producir un filtro FIR de largo M es necesario truncarla en $n = M - 1$. Este truncamiento es equivalente a la multiplicación de la respuesta a impulso ideal del filtro con una ventana rectangular, definida como:

$$w(n) = \begin{cases} 1 & n = 0,1,2,...,M-1 \\ 0 & otro\ caso \end{cases} \tag{3}$$

Con ello se puede rescribir la ecuación (2) de la siguiente forma:

$$y(n) = \sum_{k=0}^{M-1} h(k)x(n-k) \tag{4}$$

Esta truncación de los coeficientes afecta la respuesta del filtro, apareciendo oscilaciones en las bandas de paso y supresión, además de una disminución en la pendiente de caída del filtro. Estas características pueden alterarse utilizando otro tipo de ventanas y entre las más comunes se pueden mencionar las siguientes:

Blackman
$$0.42 - 0.5 \cdot \cos\left(\frac{2\pi n}{M-1}\right) + 0.08 \cdot \cos\left(\frac{4\pi n}{M-1}\right) \tag{5}$$

Hamming
$$0.54 - 0.46 \cdot \cos\left(\frac{2\pi n}{M-1}\right) \tag{6}$$

Aplicando la transformada Z a (4) es posible observar algunas características interesantes de los filtros FIR:

$$\begin{aligned} Z\{y(n)\} &= Z\left\{\sum_{k=0}^{M-1} h(k) \cdot x(n-k)\right\} = Z\left\{x(n)\right\} \cdot \sum_{k=0}^{M-1} h(k) \cdot z^{-k} \\ &= \frac{\sum_{k=0}^{M-1} h(k) \cdot z^{M-k-1}}{z^{M-1}} Z\left\{x(n)\right\} \end{aligned} \tag{7}$$

De (7) se puede ver que un filtro FIR de orden M posee M-1 polos en cero, es decir M-1 retardos, por lo que es siempre estable, y su característica de filtrado está dada sólo por la ubicación de sus ceros.

2. Buffer Lineal y Circular.

De la ecuación (4) es posible identificar tanto entradas *x(k)* como coeficientes *b(k)*. Para calcular la salida del filtro es necesario mantener en memoria un buffer con los datos de las entradas pasadas (también llamadas "delay line") además de la muestra actual. Existen dos formas de implementar un buffer para almacenar los datos pasados.

- **Buffer Lineal**: Este tipo de buffer contempla el uso de un puntero que siempre apunta al comienzo del arreglo que actúa como buffer. Con este tipo de arreglo es necesario realizar corrimientos sucesivos de todos los datos almacenados de modo que el más antiguo "caiga" del buffer y el más nuevo quede en la primera posición. El inconveniente que se presenta en esta implementación es que el costo en tiempo de ejecución de mover grandes cantidades de datos es alto. En la **Fig. 1** se observa un esquema de la implementación.

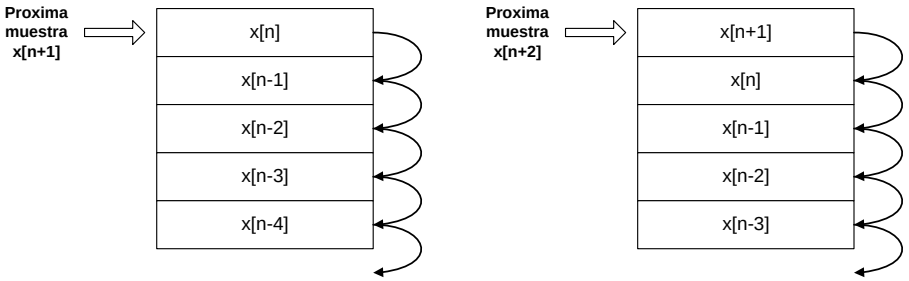

Fig. 1. Esquema del buffer lineal.

- **Buffer Circular**: El direccionamiento por buffer circular se basa en la manipulación de un puntero, el cual es incrementado en cada ciclo. Por este método es posible realizar los cálculos de forma más eficiente. Cada nueva muestra que debe ser almacenada es ubicada en la posición de memoria utilizada por el dato más viejo, y que ya no es necesario para el cálculo de la salida del filtro, reutilizando de esta forma la memoria. Cuando el puntero alcanza el final del buffer debe ser actualizado para que salte al comienzo del buffer. Este hecho da nombre a este tipo de buffer ya que el final del arreglo parece quedar junto al comienzo del mismo en una forma circular. Algunos DSPs implementan los buffer circulares por hardware, lo cual permite conservar memoria y minimizar la sobrecarga del software. Es necesario darse cuenta de que este tipo de implementación requiere del manejo apropiado del puntero tanto de buffer de datos como de coeficientes (**Fig. 2**).

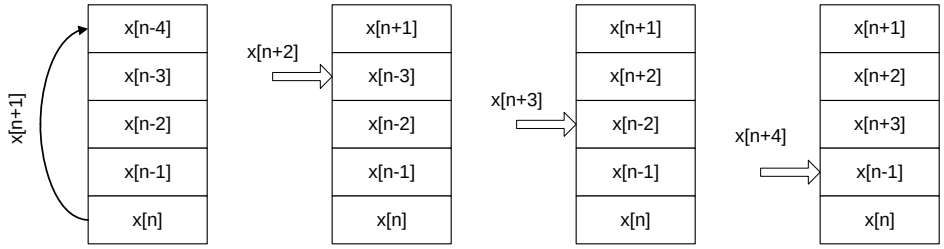

Fig. 2. Movimiento de datos en un buffer circular.

3. Actividades.

3.1 Parte Teórica (FIR).

3.1.1 Estudie y explique detalladamente el funcionamiento de un buffer circular. No copie la definición antes dada, sino que explíquelo con sus propias palabras. ¿Por qué es tan importante utilizarlo en la implementación de un filtro FIR?.

3.1.2 Estudiar los comandos de MatLab fir1, fir2 y remez. Comente las principales características de cada uno de ellos.

3.1.3 Diseñe un filtro pasa bajos de orden 70 con frecuencia de corte en 1.5 kHz y frecuencia de muestreo 8 kHz, empleando el método por ventanas usando el comando fir1, (use ventanas Blackman, y Rectangular). Analice similitudes y diferencias de los ambos filtros empleando el comando freqz.

3.1.4 Compare el resultado obtenido con el filtro Rectangular del punto anterior con uno de las mismas características pero de orden 30. Analice similitudes y diferencias empleando el comando freqz.

3.1.5 Obtenga los coeficientes para un filtro de orden 70 y de orden 150 para el filtro de la Fig. 3.10 utilizando el comando fir2.

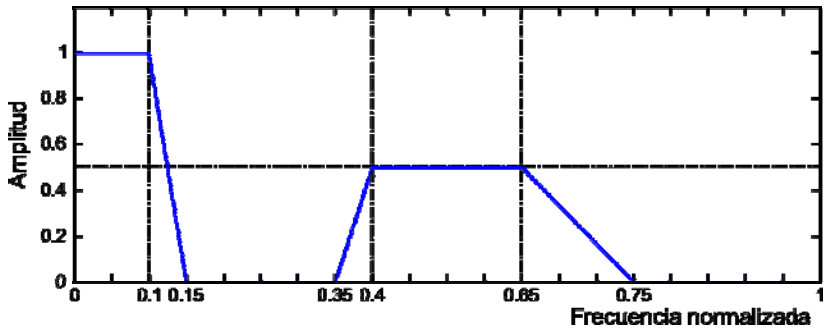

Fig. 3.10. Respuesta del filtro a diseñar con FIR2.

3.1.6 Escriba una función llamada fir_lineal() (en MatLab) y, usando los coeficientes obtenidos en los puntos anteriores, simule los filtros usando la señal rb.m que se encuentra en la página web del curso utilizando un buffer de tipo lineal. Utilice la función *espectro(datos,frec_muest)* para obtener el espectro de los datos de salida.

3.1.7 Escriba una función llamada fir_circ() (en MatLab) que implemente un buffer de tipo circular. Compruebe la correcta implementación de este buffer comparando los resultados con los obtenidos en el punto anterior para al menos uno de los filtros diseñados.

3.2 Parte Experimental (FIR).

Aplique una señal de amplio espectro a través de la tarjeta de sonido del PC. Esta señal se encuentra en el archivo de sonido rb10k.wav. Observe la entrada y salida del CODEC (utilizando los programas de la experiencia 1), verificando la salida del CODEC. Implementar en el DSP, algunos de los filtros diseñados en la parte previa, considerando un buffer del tipo circular.

Segunda Parte Laboratorio 2 – Filtros IIR.

El **objetivo de este laboratorio** es el implementar filtros de respuesta a impulso infinita, aprender a utilizar algunas herramientas de MatLab para el diseño de filtros, lograr comparar los desempeños entre filtros FIR e IIR y comparar diferentes tipos de filtros IIR, tales como Butterworth, Elíptico y Chebyshev.

4. Filtros IIR.

Los filtros de Respuesta Infinita a Impulso, o filtros IIR, de la misma manera que los filtros FIR, son sistemas LTI (lineales e invariables en el tiempo) que recrean un amplio rango de diferentes respuestas en frecuencia. A diferencia de los filtros FIR implementados en la experiencia anterior, estos filtros poseen una respuesta a impulso de duración infinita y si se considera un sistema discreto caracterizado por la ecuación de coeficientes invariantes en el tiempo se tiene que su estructura está dada por:

$$y(n) = -\sum_{k=1}^{N} a_k \cdot y(n-k) + \sum_{k=0}^{M} b_k \cdot x(n-k) \tag{8}$$

Al igual que un filtro FIR, los filtros IIR poseen una componente de promedio móvil de las entradas actuales y pasadas, pero además incluyen términos que dependen de los valores pasados de la salida, es decir, una parte autorrecursiva, característica que otorga al filtro su respuesta a impulso infinita. Mediante la transformada *z* se puede obtener la función de transferencia caracterizada por (9):

$$H(z) = \frac{\sum_{k=0}^{M} b_k \cdot z^{-k}}{1 + \sum_{k=1}^{N} a_k \cdot z^{-k}} \tag{9}$$

De esta caracterización se obtienen polos y ceros, los cuales dependen de la elección de los parámetros del sistema $\{b_k\}$ y $\{a_k\}$, y determinan las características de la respuesta en frecuencia del sistema. Note que la estabilidad de este tipo de filtros dependerá de la elección de los coeficientes a_k y de la posibilidad de representarlos con suficiente precisión en un dispositivo digital.

3.1 Estructuras para Sistemas IIR.

Existen varios tipos de estructuras para la implementación de filtros IIR, y realizaciones en forma directa o en cascada, son las que se verán en el laboratorio, puesto que son fáciles de realizar en el DSP y ser conceptualmente similar a las demás existentes.

Una forma alternativa de escribir (9) es por medio de 2 funciones de transferencia individuales en cascada, esto quiere decir:

$$H(z) = H_1(z) \cdot H_2(z) \tag{10}$$

Donde $H_1(z)$ posee los ceros de $H(z)$ y $H_2(z)$ los polos.

$$H_1(z) = \sum_{k=0}^{M} b_k \cdot z^{-k} \tag{11}$$

$$H_2(z) = \frac{1}{1 + \sum_{k=1}^{N} a_k z^{-k}} \tag{12}$$

Según sea el orden en que se ubican las funciones de transferencia (11) y (12) se llegará a uno de 2 tipos de formas directas. La ubicación de $H_1(z)$ antes que $H_2(z)$ llevará a la llamada forma directa I, la cual requiere de $M+N+1$ multiplicaciones y posiciones de memoria, y $M+N$ sumas.

Si por el contrario se ubica $H_2(z)$ antes que $H_1(z)$, la realización obtenida es conocida como forma directa II. Su principal ventaja es que sólo necesita de $max\{M,N\}$ posiciones de memoria, pero el número de multiplicaciones sigue siendo $M+N+1$ y el número de sumas $M+N$. A partir de (12) y (11) se obtienen las ecuaciones de diferencias correspondientes:

$$w(n) = x(n) - \sum_{k=1}^{N} a_k w(n-k) \tag{13}$$

$$y(n) = \sum_{k=0}^{M} b_k w(n-k) \tag{14}$$

Lo anterior supone que

$$H(z) = H_2(z) \cdot H_1(z) \tag{15}$$

y

$$H(z) = \frac{W(z)}{X(z)} \frac{Y(z)}{W(z)} \tag{16}$$

Un ejemplo típico es un filtro de segundo orden. La **Fig. 4.1** muestra uno en su estructura de la forma directa II.

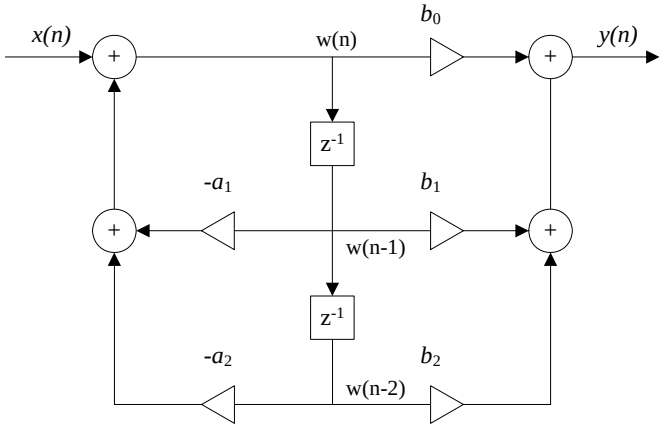

Fig. 4.1. Forma directa II de un filtro de segundo orden IIR.

El nombre de “realización en forma directa” se desprende del hecho de que aparecen directamente de la función de transferencia (10) sin ningún tipo de reordenamiento. Desafortunadamente, este tipo de estructura, de orden I y II, tienen la principal desventaja de ser altamente sensibles a la cuantificación de sus coeficientes cuando N es muy grande.

3.2 Estructura en Forma de Cascada.

Considere un sistema IIR de orden superior a 2 con la función de transferencia (9). El sistema puede ser factorizado en la cascada de subsistemas de segundo orden tal que H(z) puede ser expresado como

$$H(z) = \prod_{k=1}^{K} H_k(z) \tag{17}$$

donde K es la parte entera de (N+1)/2. H_k tiene la forma general

$$H_k(z) = \frac{b_{k0} + b_{k1} \cdot z^{-1} + b_{k2} \cdot z^{-2}}{1 + a_{k1} \cdot z^{-1} + a_{k2} \cdot z^{-2}} \tag{18}$$

Como existen muchas formas de manejar los polos y ceros de (18) en secciones de segundo orden, habrá muchas realizaciones en cascada y varias maneras de ordenar los subsistemas resultantes. Aunque todas las realizaciones en cascada son equivalentes en aritmética infinita, las diferentes realizaciones difieren significativamente cuando se implementan con aritmética de precisión infinita.

La forma general de la estructura en cascada se muestra en la **Fig. 4.2**.

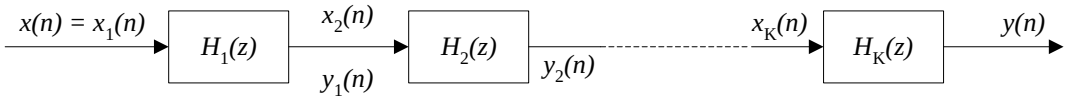

Fig. 4.2. Estructura en cascada de sistemas de segundo orden.

4.1. Parte Teórica (IIR)

Para todos los casos asuma una frecuencia de muestreo de 8 kHz.

4.1.1. Utilizando la función “*ellip*” de MATLAB diseñe un filtro de segundo orden para cada uno de los siguientes casos:

a) Pasa Bajos (fc=1KHz).

b) Pasa Altos (fc=2KHz).

c) Pasa Banda (f1=1KHz, f2=2KHz).

d) Elimina Banda (f1=1KHz, f2=2KHz).

4.1.2. Diseñe con el comando Cheby1 y Cheby2 dos filtros Pasa Banda (f1=1KHz, f2=2KHz) de segundo orden y:

a) 2dB para la banda de paso (Cheby1).

b) 20dB de atenuación (Cheby2)

Utilize el comando freqz para obtener la respuesta en frecuencia de ambos filtros.

4.1.3. Diseñe un filtro Butterworth pasabanda con frecuencias entre 400[Hz] y 800[Hz], y de orden 4, 10 y 20. ¿Qué sucede con los polos y ceros del sistema a medida que aumenta el orden del filtro? ¿Qué implicancias tiene esto en relación con la implementación en el DSP? Si es un problema ¿Cómo lo solucionaría?

4.1.4. Escriba un programa en MatLab que permita simular los filtros IIR diseñados en el punto 4.1.2 utilizando el esquema de la **Fig. 4.1**, para la entrada de amplio espectro.

4.2. Parte Experimental (IIR)

4.2.1. Implemente los filtros pasabanda y elimina banda del punto 4.1.1.

4.2.2. Implemente los filtros Chebyshev obtenidos en el punto 4.1.2.