LABORATORIO DE FILTRADO DIGITAL

Ingeniería Eléctronica (IE).
4º Curso. 2º Cuatrimestre

Marcelino Martínez / Luis Gómez /Antonio J. Serrano Joan Vila / Juan Gómez

Curso 2009-2010

marcelino.martinez@uv.es
luis.gomez-chova@uv.es
antonio.j.serrano@uv.es
joan.vila@uv.es
juan.gomez-sanchis@uv.es

http://www.uv.es/martsobm/

INDICE

BIBLIOGRAFÍA

[Proakis-98] "Tratamiento Digital de Señales: Principios, Algoritmos y Aplicaciones".Proakis, J.G.; Manolakis, D.G.Prentice Hall, 1998. ISBN:84-8322-000-8

[Mitra-01] "Digital Signal Processing: A Computer-Based Approach, 2e".Mitra S. K McGraw-Hill, 2001, ISBN 0-07-232105-9

[Antoniou-93] , "Digital Filters: Analysis, Design, and Applications", 2e. Antoniou A. McGraw-Hill, 1993,ISBN 0-07-002121-X

Bibliografía complementaria

[Oppenheim-00] "Tratamiento de Señales en Tiempo Discreto". Oppenheim, A.V,; Schafer, R.W.;Buck J.R. Prentice-Hall. 2000. ISBN: 84-205-2987-7.

[McClelland-98] "Computer-Based Exercises for Signal Processing using MATLAB ver. 5".McClellan, .H.; Burrus, C.S.; Oppenheim A.V.; Parks, T.W.; Schafer, R.W.; Schuessler, H.W. PrenticeHall. 1998. ISBN: 0-13-789009-5

[Jackson-96] "Digital Filters and Signal Processing : With Matlab Exercises " 3e.Leland B. Jackson Kluwer Academic Publishers, 1996 ,SBN 0-7923-9559-X

[Parks-87] "Digital Filter Design".
Parks, T.W.; Burrus, C.S.
John Wiley & sons. 1987. ISBN: 0-471-82896-3

[Ifeachor-97] "Digital Signal Processing: a practical approach"
Ifeachor, E.C; Jervis, B,W,
Addison-Wesley 1997 ISBN 0-201-54413-X

[Ingle-00] "Digital Signal Processing using Matlab". Ingle V.K., Proakis J.G. Brooks/Cole Thomson Learning. 2000. ISBN:0-534-37174-4

PRÁCTICA 1.- HERRAMIENTAS CAD PADA EL DISEÑO DE FILTROS ANALÓGICOS.

Para un diseñador de sistemas digitales, es imprescindible conocer las técnicas básicas de filtrado analógico que le permitan especificar, diseñar y analizar correctamente los filtros antialiasing y de reconstrucción que van a requerirse en la mayor parte de aplicaciones.

Dado que existe un buen número de paquetes software para el diseño de filtros analógicos optaremos por ilustrar su uso más que entrar en las recetas de diseño.

Paquete Filter Wiz Pro de Schematica Software (http://www.schematica.com)

Se trata de un software muy completo de diseño de filtros del que se dispone de una versión de demostración cuya limitación es que oculta los valores finales de las R.

Permite diseñar filtros de manera guiada o libre. Se inicia el diseño seleccionando el tipo de filtro (forma guiada) o con *user defined* (formato libre).

Diseño de forma guiada:

Diseña un filtro paso-bajo de frecuencia de corte en banda pasante 1kHz, en banda no pasante de 4kHz, 3dB de rizado en banda pasante y 40dB de atenuación.

Observa la respuesta al escalón del filtro de Bessel. Compárala con el resto.

Escoge el filtro de Tchebyshev directo y observa las estructuras recomendadas. Selecciona la que prefieras, observa la salida del filtro, la contribución de cada una de las etapas y analiza el estudio de la sensibilidad del filtro que hace el programa.

Pulsa la opción *strategy* en cada ventana, hay información, a menudo, muy recomendable.

Diseño de forma libre:

Diseña de manera aproximada un filtro pasa-banda con frecuencias de corte en banda pasante de 1kHz y 2kHz y en la banda atenuada de 500Hz y 4kHz, encadenando un paso-bajo y un paso-alto.

Paquete FilterCAD de Linear Technologies (http://www.linear.com)

Se trata de un software de diseño de filtros de la empresa Linear Technologies orientado a diseñar topologías que empleen sus circuitos integrados, bien sean operacionales, bien filtros de capacidades conmutadas.

Presenta también dos formas de diseño, el *Quick design* y el *Enhanced design*.

Diseño con *quick design*:

Diseña un filtro paso-bajo con frecuencia de corte en banda pasante de 500Hz, en banda no pasante de 2kHz, atenuación de 40dB, sin exigir fase lineal en banda pasante, ni asegurar ganancia en DC y con alimentación unipolar de 5VDC.

Escoge un filtro de capacidades conmutadas que implemente un elíptico de orden 4 (encapsulado 1067, F_{CLK}=50kHz). Analiza el circuito resultante. Obtén la respuesta a un pulso, un escalón y un período de una sinusoide.

Escoge un filtro R-C activo que implemente un Butterworth de orden 4. Analiza el circuito, verifica la atenuación en banda no pasante y obtén la caída en la primera y segunda décadas.

Diseño con *enhanced design*:

Diseña un filtro paso-banda con frecuencia central de 20kHz, anchura de la banda pasante de 10kHz, anchura de la banda de atenuación 40kHz, atenuación de 40dB y ganancia en banda pasante de 10dB.

Impleméntalo como filtro de capacidades conmutadas y como R-C activo obteniendo su respuesta en frecuencia real y el esquemático del circuito.

Paquete FilterLab de Microchip (http://www.microchip.com)

Se trata de un software de la empresa Microchip para el diseño de filtros basados en células de Sallen-Key con sus amplificadores operacionales.

Tiene también dos estrategias de diseño. La convencional y el *antialiasing wizard*.

Diseño convencional:

Diseña un filtro Butterworth paso-bajo con frecuencia de corte de 1kHz. Haz una comparación numérica de órdenes y tipos de filtros (opción *numerical comparison*).

Diseño con *antialising wizard*:

Diseña un filtro antialiasing con un ancho de banda de 1kHz, una frecuencia de muestreo de 100kHz, 12 bits de resolución, relación señal ruido de 80dB, por seguridad. Escoge una estructura de Tchebyshev. Observa el filtro resultante.

CASO PRÁCTICO

Se desea emplear una salida PWM de un DSP como salida analógica. Es decir, diseñar un conversor D/A a partir de una señal cuadrada entre 0 y 5V con frecuencia de 20kHz y *duty cycle* seleccionable por software con una resolución de 10 bit.

Se desea que el ancho de banda de la señal analógica vaya de 0 a 1 kHz.

Resulta fácil ver que el diseño consiste en incluir un filtro analógico paso bajo que deje pasar la banda baja y atenúe la frecuencia de 20kHz y sus armónicos (emplear un filtro de Butterworth).

La calidad del filtro, en particular la atenuación de la frecuencia del PWM y de sus armónicos, marcará la resolución del sistema.

Hacer un estudio de la resolución de la salida (rizado) dependiendo del orden del filtro (1, 2 y 3) y del ciclo de trabajo del PWM (30%, 50% y 85%).

Obtener las estructuras y coeficientes con el paquete FilterLab y emplear células de Sallen-Key y generar el filtro en PSpice con el paquete MicroSim. Emplear el operacional TL064/301/TI con alimentación simétrica entre +12V y -12V, y el elemento VPULSE para la generación del estímulo.

Empezar con un ciclo del 50% y unos tiempos de subida y bajada del pulso nulos, observar la salida. Modificar los tiempos de subida y bajada a 2µs y observar la salida.

El resultado debe darse en número de bits teniendo en cuenta que el rango dinámico va de 0 a 5 VDC.

Obsérvese y justifíquese el retardo inicial en el sistema.

Asegúrese de emplear una resolución temporal suficiente (opción *print step* del menú de simulación de transitorios) para obtener una medida acertada del ruido.

PRÁCTICA 2.- ESTUDIO DE LAS PROPIEDADES DE LOS FILTROS FIR.

1. Dadas las siguientes ecuaciones en diferencias,

$$y(n) = x(n) + 2x(n-1) + x(n-2) + 2x(n-3) + x(n-4)$$
$$y(n) = x(n) - x(n-5)$$
$$y(n) = x(n) + 6x(n-1) + 11x(n-2) + 6x(n-4)$$

Comprobar,
a)Que se trata de filtros FIR.
b)Analiza la respuesta impulsional (***impz***).
c)Estudia el retardo de grupo (***grpdelay***)

¿Los filtros FIR son filtros de fase lineal ?
(Nota: se puede utilizar la finción ***fvtool*** para el análisis visual de filtros digitales)

2.Diseñar un filtro FIR pasa-baja por el método de ventanas (***fir1***) con estas características:

Fm=20kHz, Fc=5kHz, N=20;

a) Usa la ventana rectangular (***boxcar***), y Hamming (***hamming***). En cada caso, determina aproximadamente el rizado en la banda pasante, la atenuación de la banda eliminada y la anchura de la banda de transición. Observa el efecto Gibbs. (Nota: se puede utilizar la finción ***wvtool*** para el análisis visual de ventanas)

b) Diseña ahora el filtro con los mismos parámetros pero empleando la ventana de Kaiser. Utiliza la función ***kaiser***(N,β) de Matlab para generar las muestras de dicha ventana. Modifica el valor de la atenuación de la banda no pasante a intervalos de 20dB (Atenuación máxima<90dB): A={10,30,50,70}dB.

c) Repite el diseño del apartado anterior pero fijando la atenuación a 60dB. Para estimar los parámetros de la ventana y el filtro emplea ahora la finción ***kaiserord***. Analiza cómo se modifica el orden del filtro en función de la anchura de la banda de transición (Utiliza como valor máximo de la banda de transición 2 kHz): Δf={0.5,1,1.5,2}kHz.

Los parámetros que define la ventana de Kaiser se pueden calcular aproximandamente con las siguientes expresiones.

$$\beta = \begin{cases} 0.1102(A-8.7) & A \geq 50dB \\ 0.5842(A-21)^{0.4} + 0.07886(A-21) & 21 < A < 50 \\ 0 & A \leq 21 \end{cases}$$

A es la atenuación en dB. La relación entre el orden del filtro (N) y la anchura de la banda de transición (Δf) viene dada por:

$$\Delta f = \frac{DF_m}{N-1} \quad siendo \quad D = \begin{cases} \frac{A-7.95}{14.36} & A > 21 \\ 0.922 & A \leq 21 \end{cases}$$

3. Veamos cómo se modifica el orden de un filtro diseñado con la función ***remez*** en función de la anchura de la banda de transición y rizado en la banda pasante y no pasante. Para ello utilizaremos la función ***remezord***. Las características del filtro son: *Fm*=1000Hz, *Fp*=200Hz.
   a) Variación de *N* con el ancho de banda de transición: *Fs*=210:10:300 Hz. *Rp*=3dB, *Rs*=60dB.
   b) Variación de N con la atenuación de la banda pasante: *BT*=20Hz *Rs*=60dB, *Rp*=1:1:5 dB.
   c) Variación de N con la atenuación de la banda eliminada: *BT*=20Hz *Rp*=3dB *Rs*=20:10:100;

   Mostrar en cada caso una gráfica de N en función del parámetro variable.

4. Diseñe mediante el método del muestreo en frecuencias un filtro FIR de fase lineal de Tipo I (simétricos, número de términos impar, siendo *M*+1 es el número de muestras de la respuesta en frecuencia y considerando que la primera muestra está en ω=0) con las especificaciones siguientes:

   Límite de banda de paso: *Fp*=200 Hz,
   Frecuencia de muestreo: *Fm*=1000 Hz,
   Anchura de la Banda de Transición: 50 Hz

   a) Filtro ideal.
   b) Transición entre bandas lineal de 100Hz de anchura.

   Obtenga las respuesta impulsional y respuesta en frecuencia de los filtros FIR con los dos tipos de bandas de transición fijando el módulo de la respuesta en frecuencia en *N*=41 puntos repartidos entre 0 y 2π.

   (Nota: Recuerde que siempre debe aproximar a un número impar de términos siendo el número de coeficientes del filtro $N = 2 \cdot M + 1$)

PRÁCTICA 3.- ESTUDIO DE LAS PROPIEDADES DE LOS FILTROS IIR.

1. Considérese la siguiente ecuación en diferencias correspondiente a un filtro digital.

$$y(n) = y(n-1) - \frac{1}{4}x(n-4) + \frac{1}{4}x(n)$$

Dibujar el diagrama de polos y ceros, y estudia su respuesta en frecuencia (módulo y fase) y la respuesta impulsional. ¿ Se trata de un filtro FIR o IIR ?

2. Diseño de filtros digitales a partir de prototipos analógicos. Los prototipos analógicos que pueden emplearse son butterworth, chebyshev I y II y elípticos. En este ejercicio nos basaremos en un filtro de butterworth para obtener un filtro digital de las siguientes características: fm=1000 Hz , fc=200Hz, Rp=2 dB (valor máximo), Rs=40 dB (valor mínimo) y banda de transición = 100 Hz.

   - Para estimar el orden y frecuencia de corte del filtro en cada caso se emplean las funciones ***buttord, cheb1ord***, ***cheb2ord*** y ***ellipord***.
   - Los prototipos analógicos (con frecuencia de corte unidad) se obtienen con: butterworth (***buttap***), chebyshev tipo I y II (***cheb1ap, cheb2ap***) y elípticos (***ellipap***).
   - Los métodos de transformación que emplearemos son el impulso invariante (***impinvar***) y la transformación bilineal (***bilinear***).
   - Los filtros digitales también se pueden obtener directamente con las funciones ***butter***, ***cheby1***, ***cheby2*** y ***ellip***, que diseñan a partir del prototipo analógico empleando el método de la transformación bilineal y del orden del filtro digital.

   Obtener la respuesta en frecuencia y diagrama de polos/ceros de los filtros en cada caso. Determinar la anchura de la banda de transición a partir de la respuesta en frecuencia.

3. Considerando el filtro del apartado anterior, ver cómo se modifica el orden de los filtros digitales obtenidos a partir de los prototipos de butterworth, chebyshev I y II y elíptico al variar los valores de Rp y Rs.
   - Rs = 40 dB y Rp = 0.1:0.1:4 dB
   - Rp = 2 dB y Rs = 10:1:100 dB

   Indica qué filtro utilizarías en cada uno de los siguientes casos:
   - El orden debe ser el mínimo posible.
   - La respuesta debe ser plana en la banda pasante .
   - Respuesta plana en la banda pasante. Banda transición estrecha. Mínimo orden.

4. Compara los órdenes obtenidos para filtros FIR (equiripple) e IIR (butterworth, chebyshev I y II y elíptico) de las mismas características (frecuencia de corte, atenuación, anchura de la banda de transición, etc): Filtro pasa Baja Fc=200Hz, BT=100Hz, Rp=2dB, Rs=40 dB Frecuencia de muestreo 1000Hz).
   Representa la respuesta en frecuencia para cada caso.

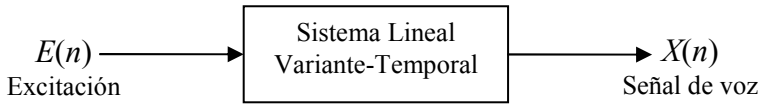
PRÁCTICA 4. MODELIZACIÓN DE LA SEÑAL DE VOZ

En esta práctica vamos a aplicar distintas técnicas del filtrado digital de señales en el tratamiento de señales de voz: preprocesado, modelado, codificación y síntesis. Las señales se leerán y reproducirán empleando las funciones ***wavread*** y ***sound***.

1. Las señales de voz tienen un espectro que decae en altas frecuencias. En algunas aplicaciones resulta deseable que esta caída se compense mediante *pre-énfasis*.

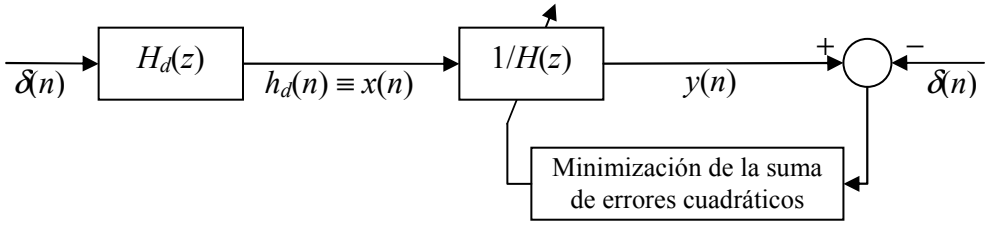
   - *Muestreo*: Si la señal está muestreada a más de 8 kHz, se remuestrea para reducir *Fm* optimizando el rango de frecuencias (***resample***).
   - *Pre-énfasis*: Diseña un filtro de orden 1 que amplifique las frecuencias altas de la señal de entrada mediante una ubicación de sus ceros y polos adecuada.

2. Vamos a modelar la voz como la salida de un sistema lineal cuya entrada es ruido blanco o un tren de impulsos cuasi-periódico. Este sistema, aunque es variante-temporal, $H_d(z,t)$, varia muy lentamente con el tiempo y puede considerarse invariante temporal (LTI) durante periodos cortos de tiempo, $H_d(z)$.



   - *Segmentación*: Divide la señal en segmentos $x(n)$ de 40ms, durante los cuales consideramos constante $H_d(z)$. (*extra*: solapar los segmentos entre si un 50%)
   - *Enventanado*: Aplica una ventana a cada segmento, previa al modelado, para evitar introducir altas frecuencias parásitas (transición abrupta en los bordes).

3. Si describimos el sistema como una función todo-polos, $H(z) = G \,/\, (1+\sum a_k \cdot z^{-k})$, podemos modelarlo mediante el diseño de filtros IIR por aproximación de mínimos cuadrados (*método de la autocorrelación* o *algoritmo de Levinson-Durbin*).



   Idealmente, si $y(n) = \delta(n)$, tenemos que $H_d(z)/H(z) = 1$, por lo que $H(z)$ modela perfectamente a nuestro sistema. El problema de obtener el modelo del sistema $H_d(z)$ es equivalente a obtener los coeficientes del filtro FIR que predeciría la señal de voz, $x(n)$, que queremos modelar, $\overline{x}(n) = -a_1 \cdot x(n-1) - a_2 \cdot x(n-2) - \ldots - a_p \cdot x(n-p)$. En ambos casos se resuelve el mismo problema de minimización ε (minimización de la suma de errores cuadráticos), por lo que emplearemos técnicas de predicción lineal (LP) para encontrar los coeficientes a_k.

$$\varepsilon = \langle\, (x(n) - \overline{x}(n))^2 \,\rangle = \langle\, (x(n) + \sum a_k \cdot x(n-k))^2 \,\rangle \;\Rightarrow\; \mathfrak{R} \cdot \mathbf{a} = \mathbf{r} \Rightarrow \mathbf{a} = \mathfrak{R}^{-1} \cdot \mathbf{r}$$

- *Codificación*: Obtener los coeficientes a_k del filtro de predicción (***lpc***) de orden 12 que predicen las muestras de la señal $x(n)$ y modelan el denominador de $H(z)$.
- Comparar la señal original, $x(n)$, y la respuesta impulsional, $h_d(n)$, del sistema modelado, $H(z)$, dibujando en una gráfica sus amplitudes en el dominio temporal (en segundos), sus señales de autocorrelación (***xcorr***) y sus espectros (en Hz).
- Comenta la relación entre las dos señales de autocorrelación y el orden de $H(z)$.
- *Extra*: Comparar los coeficientes que devuelve la función ***lpc*** con los obtenidos con la función ***levinson*** y con los obtenidos al resolver el sistema de ecuaciones $\mathbf{a} = \mathfrak{R}^{-1} \cdot \mathbf{r}$ empleando la señal de autocorrelación de $x(n)$, $r_{xx} = \textbf{\textit{xcorr}}(x)$.

4. Los coeficientes LPC del predictor lineal encontrado se pueden emplear para sintetizar la señal original a partir de un modelo todo-polos. La respuesta impulsional del sistema causal todo-polos con los coeficientes de predicción lineal es una nueva síntesis aproximada de la señal original que presenta la misma señal de autocorrelación.

   - Genera las dos señales prueba de excitación, $e(n)$, del sistema: tren de pulsos (1 pulso equivaldrá a la respuesta impulsional) y ruido blanco.
   - Calcula las ganancias G de sistema para que la energía de cada segmento sintetizado sea proporcional a la del segmento correspondiente de la señal original.
   - Combina todos los segmentos sintetizados para obtener una señal completa, $Y=[y_1\ y_2\ ...]$, compara su amplitud en el dominio temporal (seg.) y escucha el resultado (***sound***).
   - Calcula el *factor de compresión* conseguido al codificar la señal con los coeficientes LPC y el *error cuadrático medio* entre la señal original y sintética.

PRÁCTICA 5.- ESTRUCTURAS DE FILTROS DIGITALES

1. Considera un sistema LTI cuya función de transferencia viene dada por la siguiente expresión:

$$H(z)=\frac{\left(1+z^{-1}\right)\left(1+\frac{4}{3}z^{-1}+\frac{9}{8}z^{-2}\right)}{\left(1-\frac{4}{3}z^{-1}+\frac{8}{9}z^{-2}\right)\left(1-\frac{4}{5}z^{-1}+\frac{10}{11}z^{-2}\right)}$$

   Utilizando las funciones ***zp2sos***, ***sos2tf*** y ***residuez*** dibuja el diagrama de bloques para la implementación de este sistema de las siguientes formas:

   a) Forma directa I.
   b) Forma directa II.
   c) Realización en cascada utilizando etapas de primer y segundo orden implementadas mediante la forma directa II
   d) Realización en paralelo con etapas implementadas mediante la forma directa II traspuesta.

2. Escribe una función a la que se le pasen los coeficientes de un filtro IIR en forma directa y una señal de entrada *X*, y realice el filtrado de la misma como etapas de segundo orden en cascada (para el filtrado de cada una de estas etapas se puede utilizar la función ***filter***).

3. Repite el apartado 2 para etapas en paralelo.

4. Dado un sistema digital cuya ecuación en diferencias es:

$$y(n)=x(n)-\sqrt{2}\cdot x(n-1)+x(n-2)+y(n-1)-\frac{1}{4}y(n-2)-\frac{1}{4}y(n-3)+\frac{1}{8}y(n-4)$$

   Programe su realización para el caso de la estructura Directa II y Cascada.
   Compruebe los resultados y comente las diferencias entre estructuras.

PRÁCTICA 6. EFECTOS DE LA CUANTIFICACIÓN DE LOS COEFICIENTES EN FILTROS DIGITALES.

1. En este apartado analizaremos la influencia de la estructura en la estabilidad, como consecuencia de la cuantificación de los coeficientes.
Considera el filtro resonador doble dado en la forma directa por:

$$H(z) = \frac{1}{1 - 3.5020 \cdot z^{-1} + 5.0240 \cdot z^{-2} - 3.4640 \cdot z^{-3} + 0.9790 \cdot z^{-4}}$$

y en cascada por:

$$H(z) = \frac{1}{1 - 1.8955 \cdot z^{-1} + 0.9930 \cdot z^{-2}} \cdot \frac{1}{1 - 1.6065 \cdot z^{-1} + 0.9859 \cdot z^{-2}}$$

donde se ha empleado una precisión de 4 decimales para representar los coeficientes.
Resuelve las siguientes cuestiones:

a) Analice el funcionamiento de las funciones ***round, floor, fix*** y ***ceil*** sobre el vector x=-3:0.2:3. ¿Cuál utilizarías para cuantificar por redondeo?, ¿y para truncamiento?
b) Utilizando ***zplane***, dibuja la posición de los polos del sistema en el plano Z. Representa la respuesta en frecuencia del filtro y su respuesta impulsional.
c) Considera la implementación en cascada y redondea los coeficientes de las dos secciones del filtro a una precisión de 2 decimales. Repite las cuestiones del apartado b), compara la respuesta en frecuencia y analiza la estabilidad del filtro.
d) Considera ahora la implementación directa, redondea los coeficientes del filtro a una precisión de 2 decimales, repite las cuestiones del apartado a), compara la respuesta en frecuencia y analiza la estabilidad del filtro.

2. En este apartado analizaremos la influencia de la estructura en las prestaciones del filtro y en la estabilidad, como consecuencia de la cuantificación de los coeficientes
Se ha diseñado un filtro digital pasa baja Tchebyshev tipo I con las siguientes características: frecuencia de muestreo de 10 kHz, frecuencia de banda pasante 500 Hz, frecuencia de banda no pasante 1 kHz, rizado en banda pasante 1 dB, atenuación en banda no pasante de 50 dB (se puede comprobar los resultados con ***cheb1ord*** y ***cheby1***).
El filtro resulta ser de orden 6 y se ha descompuesto en cascada utilizando la función ***tf2sos*** (equivalente a ***tf2zp***+***zp2sos***) y de la forma:

$$H(z) = G \cdot \frac{(1+z^{-1})^2}{1 + a_{01} \cdot z^{-1} + a_{02} \cdot z^{-2}} \cdot \frac{(1+z^{-1})^2}{1 + a_{11} \cdot z^{-1} + a_{12} \cdot z^{-2}} \cdot \frac{(1+z^{-1})^2}{1 + a_{21} \cdot z^{-1} + a_{22} \cdot z^{-2}}$$

donde el factor de normalización de ganancia $G=8.07322364 \cdot 10^{-7}$.

Se dan los coeficientes de la implementación en cascada con precisión total[1], entendiéndose por ésta la correspondiente a 8 dígitos:

a_{01}=-1.86711351	a_{11}=-1.84679822	a_{21}=-1.85182222
a_{02}=+0.96228613	a_{12}=+0.89920764	a_{22}=+0.86344488

[1] Matlab muestra los datos con 4 dígitos decimales. Puede modificarse el número de decimales de la representación con la instrucción ***format***. *format short* (4 decimales), *format long* (8 decimales). Este formato no tiene ningún efecto sobre los cálculos que realiza Matlab

La implementación directa del filtro (multiplicando las etapas en cascada) es de la forma:

$$H(z) = \frac{G \cdot (1+z^{-1})^6}{1 + a_1 \cdot z^{-1} + a_2 \cdot z^{-2} + a_3 \cdot z^{-3} + a_4 \cdot z^{-4} + a_5 \cdot z^{-5} + a_6 \cdot z^{-6}}$$

donde los coeficientes a_i pueden obtenerse fácilmente a partir de los a_{kj} (***conv***)

Se pretende observar el efecto que tiene la cuantificación de los coeficientes *a* sobre las prestaciones del filtro cuando se implementa de forma directa y en cascada.

Para ello se van a analizar las siguientes características cuando se redondean los coeficientes del filtro (rededucir el número decimales de uno en uno desde 8 hasta 1):

a) Error relativo de redondeo definido como la relación[2] $\varepsilon = \frac{\|\hat{a} - a\|}{\|a\|}$
b) Posición de los ceros y polos del filtro.
c) Respuesta en frecuencia del filtro en dB verificando si se cumplen las condiciones de rizado en la banda pasante, frecuencia de corte y atenuación.
d) Respuesta impulsional del filtro.
e) Estabilidad.

3. Escribir una función de cuantificación digital a la que se le pase un vector de coeficientes y el número de bits para la cuantificación de los mismos y devuelva los coeficientes cuantificados. Esta función debe aceptar un tercer parámetro que indique si la cuantificación se va a hacer por redondeo o por truncamiento.

Usaremos esta función para ver los efectos de cuantificación sobre diferentes estructuras.
Implementa el filtro IIR con polos en z = 0.8; z = $0.85 \cdot e^{\pm j\pi/4}$; y z = $0.9 \cdot e^{\pm j\pi/6}$; con la estructura directa y con lattice. Analiza su respuesta impulsional (y posición de polos y ceros) en función del número de bits (desde 4 a 16) empleado para representar los coeficientes (a_i y k_i respectivamente).

[2] Dado un vector **x**, se define su norma como $\|x\|^2 = \left(\sum_{n=0}^{N} x^2(n)\right) = x^t x$

PRÁCTICA 7. MODIFICACIÓN DE LA FRECUENCIA DE MUESTREO

1. El diezmado es el proceso de reducir la frecuencia de muestreo con que se ha adquirido una señal. MATLAB dispone del comando ***decimate*** que se ocupa de filtrar la señal y muestrearla a una frecuencia menor. Generar una señal con 1000 puntos muestreada a 8 kHz con las siguientes componentes:

   sinusoide (amplitud 1, frecuencia 480 Hz)
   sinusoide (amplitud 1, frecuencia 1080 Hz)
   sinusoide (amplitud 1, frecuencia 2480 Hz)
   ruido gaussiano de media 0 y desviación típica 0.1.

Realizar la decimación de la señal de orden 4 de las siguientes formas:

a) Con un filtrado previo IIR bidireccional de orden 8 (comando ***decimate***).
b) Con un filtrado previo FIR unidireccional de orden 30 (comando ***decimate***).
c) Con un filtrado previo IIR unidireccional de orden 16 (no usar ***cheby1***).
d) Sin ningún filtro previo.

Observar y comentar la respuesta en los dominios del tiempo y la frecuencia.

2. La interpolación es el proceso de incrementar la frecuencia de muestreo de una señal de tiempo discreto. MATLAB dispone del comando ***interp*** que inserta ceros y posteriormente filtra la señal con un procedimiento adecuado.

Realizar la interpolación de orden 4 sobre la señal original de las siguientes formas:

a) Con un filtrado FIR de orden 8 (l=1,comando ***interp***).
b) Con un filtrado IIR de orden 4.
c) Con un filtrado FIR convencional de orden 8.
d) Sin ningún filtro posterior.
e) Con un filtrado bidireccional con el filtro IIR empleado en b) (***filtfilt***)

Observar y comentar la respuesta en los dominios del tiempo y la frecuencia. Comprobar los resultados dejando el valor por defecto del parámetro α de la función ***interp*** y sustituyéndolo por otro más adecuado. Observa el retardo introducido en las señales en cada uno de los apartados anteriores.

3. Diseñar una simulación de un sistema que permita combinar señales de audio correspondientes a conversaciones telefónicas (muestreadas a 8 kHz) y sintonías almacenadas en cintas DAT (muestreadas a 48 kHz). El resultado debe almacenarse en formato de televisión de alta definición (HDTV) que funciona con una frecuencia de muestreo de 32 kHz.

PRÁCTICA 8.- Diseño y análisis de filtros digitales con *FDATool*, representación de estructuras en *Simulink* e implementación del algoritmo en un *DSP Starter Kit* (DSK) de Texas Instruments.

1. **Diseño y análisis de filtros digitales empleando el entorno de FDATool** (*MATLAB > Filter Design Toolbox > Filter Design & Analysis Tool*: FDATool). La función de MATLAB fdatool.m consiste en un entorno gráfico para el diseño y análisis de cualquier tipo de filtro digital. Vamos a realizar un repaso de las posibilidades que ofrece esta función siguiendo unos cuantos pasos que plasmen las distintas fases del diseño de un filtro. En primer lugar tendrás que abrir la ventana de diseño (fdatool). Al ejecutarse esta función, durante la inicialización comprueba las librerías de MATLAB instaladas y muestra las funciones que estén disponibles. En principio el proceso comienza con la ventana de *Design Filter*.

   Filter Type: Lowpass
   Design Method: IIR, Butterworth
   Filter Order: Minimum order
   Frequency Specifications:
      Units: Hz
      Fs (frecuencia de muestreo): 8000
      Fpass (frecuencia de corte de la banda pasante): 1500
      Fstop (frecuencia de corte de la banda no pasante): 2000
   Magnitude Specifications:
      Units: dB
      Apass (máxima pérdida en la banda pasante): 2
      Astop (mínima pérdida en la banda no pasante): 35
   Pulsa *Design Filter*.

   Pulsa en los iconos de la barra superior para ver:
      Filter Specifications
      Magnitude Response
      Phase Response
      Magnitude and Phase Response
      Group Delay
      Impulse Response
      Step Response
      Pole/Zero Plot
      Filter Coefficients

   Dentro de la parte de *Current Filter Information*, empleando el menú contextual del botón derecho, selecciona *Convert structure*….
   Selecciona *Direct Form II* con la opción de *Use second order sections*.
   Selecciona *None* para el escalado.
   Pulsa en *Filter coefficients* para ver los coeficientes de las etapas de 2º orden.

   Para exportar los coeficientes de la implementación del filtro diseñado, *File > Export*…. En esta ventana podremos seleccionar entre guardar los coeficientes en un archivo de texto o en variables de MATLAB

   Para copiar/manipular las gráficas se hace desde *Analysis > Full View Analysis*.

2. **Representación y simulación de filtros digitales en Simulink empleando el entorno de *Filter Realization Wizard*** (*MATLAB > Blocksets > DSP > Filter Realization Wizard*). La herramienta de MATLAB *Filter Realization Wizard* abre el entorno gráfico para la representación de filtros digitales FDATool. Como hemos comentado antes, la función fdatool comprueba las librerías de MATLAB instaladas e incluye funciones que estén disponibles. *Filter Realization Wizard* pertenece a la librería de *Simulink DSP Blockset*, pero si se tienen todas las librerías el resultado al ejecutarla es el mismo que al ejecutar fdatool, pero se accedería directamente al panel de *Realize Model* (barra de iconos vertical de la izquierda). Este panel está optimizado para su uso con las herramientas de *DSP Blockset* para simular el comportamiento numérico de los filtros en dispositivos DSP, FPGA, o ASIC con coma fija y coma flotante de distinta precisión.

   Comenzamos el proceso en la ventana de *Import Filter*. En esta ventana se puede seleccionar la estructura del filtro e indicar los valores de los coeficientes:
   Filter Structure: Direct Form II
   Numerator: [1, 0.2, 0.7]
   Denominator: [1, -0.5, 0.9]

   Una vez importado el filtro pasamos a la ventana *Realize Model.*
   Optimization: Marcar todos
   Model:
   Destination: New model
   Block name: MiFiltro

   Al pulsar en *Realize Model* (automáticamente genera la estructura del filtro en simulink)

   Con un doble click en el bloque del filtro aparecerá el diagrama detallado de la realización del filtro (sumas, retardos y ganancias).

   Obten las estructuras equivalentes del mismo filtro para los siguientes casos:
   Direct Form I
   Direct Form II
   Lattice (ARMA)

   Comprueba para cada caso los valores de los coeficientes en las estructuras.

3. **Implementación de modelos de Simulink en un *DSP Starter Kit* (DSK) de Texas Instruments.** El objetivo de este apartado es mostrar la integración de diferentes herramientas para el diseño, análisis, representación, simulación, implementación y ejecución de filtros digitales. Se va a explica brevemente como utilizar un DSK de la familia de DSP C6000 de TI junto con Matlab para realizar rutinas de procesado y generar el código para este DSP automáticamente. Los elementos indispensables para realizar todo el proceso son:

   Diseño y análisis: *Matlab* con las librerías *Filter Design* y *Signal Processing*.
   Representación y Simulación: *Simulink* con el *DSP Blockset* y el *ET C6000DSP*.
   Implementación: *Matlab Real-Time Workshop* y *Code Componer Studio* de TI.
   Ejecución: Placa del *DSP Starter Kit*.
   Sistema Operativo: *Windows XP* (imprescindible para compilar con *Matlab*)

   En general, el proceso a seguir es:

   1. Comprobamos la versión de Matlab y las librerías.
   2. Instalamos el Code Componer Studio.
   3. Ponemos en marcha Matlab y generamos un modelo de Simulink empleando las librerías estándar y las específicas del DSK empleado (Simulink > Blockset > Embedded Target for TI C6000 DSP).
   4. Pulsando CTRL+B (*Build All*) se inicia todo el proceso de compilación. Una vez generados por Matlab los ficheros fuente se hace una llamada al Code Composer Studio (CCS) para realizar la compilación para el DSP.
   5. Si no tenemos conectado el DSK al PC (o no está alimentado), CCS nos dará un mensaje indicándolo. Se puede seguir el proceso de compilación pero no se podrá volcar el código del programa (esta tarea se podrá hacer a posteriori, sin necesidad de arrancar Matlab, únicamente con el CCS con la opción File > Load Program y eligiendo el fichero de extensión .out generado anteriormente. Si el DSK está conectado el programa se vuelca automáticamente por el puerto de comunicaciones.
   6. Posteriormente elegimos ejecutar (Run). y se inicia la ejecución, del programa previamente cargado en el punto 5.

   Empleando los bloques específicos de Simulink para el DSK utilizado, que permiten acceder a la entrada de audio AD, la salida de audio DA, los interruptores y los LEDs podemos construir diagramas complejos que utilicen otros bloques de la librería de DSP Blockset o Simulink (NO todos los modelos de bloques serán implementables).

   Vamos a utilizar la herramienta de análisis de filtros FDAtool para diseñar 3 filtros FIR de audio y generar su diagrama de bloques con la estructura seleccionada para su implementación. Colocaremos los filtros directamente en el diagrama de bloques que emplee el módulo para acceder al AD (entrada de audio), el bloque de los Switch para seleccionar el filtro empleado para obtener la señal de salida y el bloque de conversor DA para sacar la salida de audio filtrada (y el bloque de Reset para activarlo).

Las especificaciones de los filtros a implementar son las siguientes:

Filtro 1: Filtro Pasa Baja de Fp=500Hz, Fs=1000Hz, Rp=1dB, Rs=40dB.
Filtro 2: Filtro Pasa Banda de Fs=500Hz, Fp=1000Hz, Fp'=2500Hz, Fs'=3000Hz, Rp=1dB, Rs=40dB.
Filtro 3: Filtro Pasa Alta de Fs=2500Hz, Fp=3000Hz, Rp=1dB, Rs=40dB.
Sistema: Señales de audio mono muestreadas a 8kHz.

Las módulos de Simulink a emplear son los siguientes:

Simulink > Sources > In
Simulink > Sinks > Out
Simulink > Signal Routing > Switch y Selector
Embedded Target for TI C6000 DSP > Todos

Las señales de prueba a emplear son las siguientes:

El fichero de audio de tu ordenador que quieras.
Una señal x con armónicos en 250Hz, 2000Hz y 3500Hz. Matlab: sound(x,Fm)

PRÁCTICA 9 (2 sesiones). PROGRAMACIÓN DE FILTROS EN TIEMPO REAL EN C6713 DSK MEDIANT CODE COMPOSER.

En esta práctica se introducirá la utilización de Code Componer para la programación de filtros digitales sencillos en la placa de evaluación C6713 DSK. Los filtros se programarán en lenguaje C. Los coeficientes se obtendrán mediante la herramienta FDATool de Matlab, que nos permitirá también cuantificar los coeficientes para una implementación en coma fija. El entorno de desarrollo Code Componer es común a todas las familias de DSP de Texas Instruments, por lo que la mayoría de las opciones disponibles para esta placa de desarrollo son también validas para otras familias.

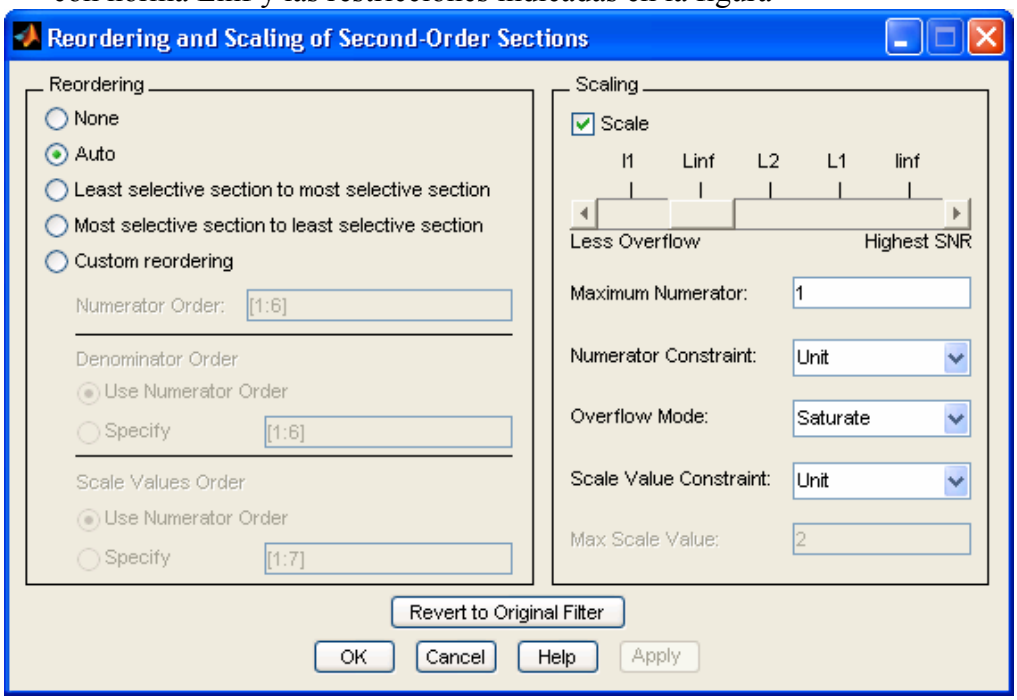
- **Copia los programas proporcionados en la carpeta \myprojects, ubicada dentro del directorio donde esté instalado el Code Componer.**
- **Conecta la salida de auriculares de tu PC o reproductor MP3 con la entrada LINE-IN de la DSK y la salida LINE-OUT o headphone con unos auriculares.**

Para centrarnos en la implementación de los filtros se proporcionarán ejemplos de programas ya realizados sobre los que incluiremos los bloques de filtrado

5. Lee el Documento de Introducción (*Introduccion Code Composer 6713SK.pdf*) y comprueba el funcionamiento mediante el ejemplo 1.1 (proyecto sine8_LED). No os detengáis, en las opciones de compilación ya que estarán especificadas en el proyecto proporcionado. El objetivo de este apartado es aprender a compilar, ejecutar, monitorizar variables, etc. **No es necesario considerar los apartados 1.7 y 1.8**.
6. Ejecuta el programa del ejemplo 2.1 (proyecto *loop_intr*). Prueba varios valores de la frecuencia de muestreo (8 kHz, 16 kHz, 24 kHz, 32 kHz, 44 kHz, 48 kHz, 96 kHz). Trabajaremos con un solo canal de audio para aplicar los filtros. Este programa representa el sistema digital y(n)=x(n), dentro de esta función incluiremos nuestro procesado. (En el ejemplo loop_stereo, puedes ver como se trabajaría con 2 canales)
7. A partir del proyecto *flash_sine* modifica el fichero *flash_sine.c* para que el sistema genere una oscilación sinusoidal de frecuencia 1000kHz para una Fm=8kHz (escucha la señal por el altavoz). El oscilador se debe generar como la respuesta impulsional de un sistema de 2º orden con polos sobre la circunferencia unidad. **Nota:** Aunque las operaciones las realices en coma flotante el resultado que se envíe al DAC deber ser un entero con signo de 16 bits. La amplitud determina el volumen del sonido.
8. Abre el proyecto *fir.pjt* e inserta las líneas de código necesarias, en el fichero fuente *fir_alumno.c*, para calcular la salida del filtro, sabiendo que los coeficientes se almacenan en el vector h[], y el buffer con los datos en dly[]. dly[0] es la entrada actual, x(n),y dly[k] es la entrada x(n-k).
9. Diseña un filtro FIR en Matlab (usa FDAtool) que elimine la banda de 1000 a 4000 Hz con bandas de transición de 500 Hz y una atenuación de 40dB. El rizado en la banda de paso es de 0.5 dB y la Fm=44.1 kHz. Cuantifica los coeficiente en formato Q15. Exporta el fichero de coeficientes (Tarjet->Generate C Header …). **Cambia el nombre de las variables en las que se almacenan los coeficientes y el número de términos para que coincidan con las del programa ejemplo. Edita el fichero**

generado y cambia la constante que define el número de términos por un #define N 147 Verifica el funcionamiento del filtro.

10. Usando como ejemplo *sine8_LED*, permite que el filtro actúe o no sobre la señal dependiendo del valor del microinterruptor 0. (Nota: para que el programa funcione correctamente cuando no actúe el filtro también es necesario sacar un valor por el DA)
11. Carga el proyecto *IIR.pjt* y estudia el fichero *IIR.c*. El sistema implementa un filtro IIR en cascada de un número arbitrario de etapas usando la forma directa II. Es importante tener en cuenta las operaciones producto de datos de tipo short (16 bits), cuyo resultado (32 bits) se redondea tomando la parte alta, teniendo en cuenta la duplicidad del bit de signo. Los coeficientes escalados en formato Q15 se almacenan en el archivo *bs1750.cof*. Veamos el proceso para obtener este fichero de coeficientes con Matlab.
    a. Diseña un filtro Elíptico con las mismas especificaciones que el filtro del apartado 6.
    b. E el apartado de cuantificación especifica coma fija con 16 bits. Usa formato fraccional con 15 bits para numerador y denominador.
    c. Vamos a calcular los coeficientes de escalado y reordenar las secciones. (Edit→Reordering and scaling second order sections). Activa la opcion *scale* con norma Linf y las restricciones indicadas en la figura



Observa como la respuesta en frecuencia del filtro obtenido al cuantificar los coeficientes, aunque es estable, dista mucho de la deseada. (Deberíamos optar por una implementación en coma flotante).

8. Repite los apartado del ejercicio 7 pero para un filtro de Butterworth con las siguientes especificaciones: Fp=1000Hz, Fs=2000Hz, Rp=1dB, Rs=40dB, frecuencia de muestreo 8kHz. Exporta los coeficientes y verifica su funcionamiento con el proyecto *IIR.pjt*
9. Mofica el proyecto IIR.c para que el filtrado ser realice en coma flotante y verifica el resultado con el mismo filtro sin escalar ni cuantificar los coeficientes.

Nota: En Matlab podemos ver qué ocurre con la respuesta en frecuencia del filtro cuando se cuantifican los coeficientes. Para este filtro con la frecuencia de muestreo de 44kHz, el filtro IIR obtenido sufre una gran degradación al cuantificar los coeficientes en formato Q15. La respuesta en frecuencia obtenida no es la deseada. Deberíamos implementarlo en coma flotante. En *iir_float* se ha implementado con aritmética de coma flotante.

Nota: Cuando se exportan los coeficientes a formato C, para los filtros IIR tenemos las etapas de segundo orden y también unos factores de ganancia a la entrada de cada etapa y también a la salida. El formato generado es distinto al de los ficheros originales del programa ejemplo por lo que hay que arreglarlo a mano (se podría automatizar)

Programas incluidos en las soluciones.

FIR_IIR.pjt
Optimización de código –o3

Fir.c Filtro FIR en coma fija. Se activa/desactiva el filtrado con el microinterruptor 0.

Fir_iir.c Filtros pasa banda (apdo5). IIR en coma flotante. IIR y FIR en coma fija. Se conmuta con interruptor 0

Fir_iir_float.c Filtros pasa banda (apdo5) en coma flotante. IIR y FIR. Se conmuta con interruptor 0

Loop_stereo.pjt (No usado en la práctica) Muestra como leer datos stereo de AD

Loop_stereo_delay.c Introduce un retardo entre los canales izquierdo y derecho del una señalde audio stereo.

Flash_sine.pjt

Flash_sine_marsel.c Oscilador sinusoidal a partir de un sistema de 2º orden
Flash_sine_marsel_enterol.c Oscilador sinusoidal a partir de un sistema de 2º orden en coma fija

ES INTERESANTE VER LOS EFECTOS DEL REDONDEO SOBRE LA RESPUESTA EN FRECUENCIA CON FDATOOL. Hay que tener en cuenta que con Fdatool, al cuantificar se tiene en cuenta la estructura mientras que con *filter* o *freqz* siempre se usa la forma directa II por lo que en ocasiones, los resultados no son buenos en este último caso, ya que es necesaria una implementación como etapas de segundo orden. En Fdatool se puede observar la respuesta en frecuencia del filtro cuando se implementa con etapas de segundo orden en cascada y se cuantifican cada una de estas etapas. (View→ SOS View Settings)