UNIVERSIDAD PANAMERICANA

ESCUELA DE INGENIERIA

Con estudios incorporados a la
Universidad Nacional Autónoma de México

HERRAMIENTA PARA EL DISEÑO DE FILTROS DIGITALES.

T E S I S

QUE PARA OBTENER EL TITULO DE:

INGENIERO MECANICO ELECTRICISTA
AREA: INGENIERIA MECANICA

P R E S E N T A:

MIGUEL ALEJANDRO CHAVEZ SALAS

DIRECTOR: DR. STANISLAW RACZYNSKI GAWIN

MEXICO, D. F. 1997

A Dios por haberme dado a mi familia. A mis padres Rafael y Susana que con su amor y cariño han sabido demostrarme sus valores a través del ejemplo. A mi abuela Beatríz por haber hecho de su vida, una vida llena de entrega. A mi abuelo Miguel que con su ejemplo de esfuerzo y dedicación hizo de mi familia una familia honrada y trabajadora.

Al héroe anónimo creador de los tacos al pastor.

Miguel A. Chávez Salas

ÍNDICE

INTRODUCCIÓN

En la actualidad y gracias al gran avance que ha tenido en las últimas décadas la tecnología digital y con ésta la computación, resulta fundamental llevar a cabo estudios tendientes a mejorar las herramientas para la implementación de sistemas de procesamiento de señal en áreas en donde aún no se hayan implementado. Una reducción de costo, tamaño y potencia requerida en los sistemas digitales han llevado a un gran incremento en las aplicaciones. Los sistemas digitales se presentan cada vez en un mayor número de aplicaciones en nuestra vida: computadoras , comunicaciones, automóviles e inclusive aparatos domésticos. En estas aplicaciones tan diversas, la información se guarda cada vez con mejores resultados de forma digital. Como resultado, el procesamiento digital de señal se ha convertido en una herramienta indispensable.

Esta tesis tiene como propósito fundamental el desarrollo de un paquete que sirva como herramienta en el diseño de filtros digitales. Estos filtros digitales tienen la finalidad de ayudar a estudiar señales acústicas causadas por fenómenos físicos. El estudio de la señal acústica de algunos fenómenos nos ayudará a comprenderlos más y así modificarlos o utilizarlos a nuestra mejor conveniencia.

Este trabajo de investigación presenta la teoría indispensable para la realización de cualquier procesamiento de señal. En el Capítulo 1 se presenta una breve explicación de cómo debe manipular la señal digital tanto a la entrada como a la salida del procesador digital de señales. En el Capítulo 2 se señala el marco teórico referente a los sistemas de tiempo discreto con el objeto de presentar algunas reglas y propiedades de este tipo de sistemas, esta tesis se enfoca principalmente a sistemas lineales; el estudio de sistemas no lineales requiere de un estudio más profundo y es

tema de investigación aparte. En el Capítulo 3 se indican métodos utilizados para el procesamiento de señales de entrada para producir salidas controladas. En el Capítulo 4 se expone la función de transferencia con la finalidad de ofrecer métodos matemáticamente equivalentes de describir a los filtros. En el Capítulo 5 se enseñan algunos métodos de diseño de filtros digitales utilizados en el desarrollo del software. El Capítulo 6 nos mostrará la realización del paquete, las herramientas utilizadas y el método empleado para su diseño. En el Capítulo 7 se muestra el gran campo de aplicaciones para estos estudios y la necesidad de continuar con ellos en estudios de posgrado para un avance tecnológico de punta en nuestro país.

Capítulo I.

Muestreo y Reconstrucción.

1.1 Introducción.

El procesamiento digital de señal se realiza en tres partes

1. La señal analógica se digitaliza, se muestrea, cada muestra se cuantifica en un número finito de partes (bits). Este proceso se llama conversión A/D análago-digital.
2. Las muestras digitalizadas se manipulan mediante un procesador digital de señal.
3. Las muestras resultantes de salida se convierten de regresa en una señal analógica mediante un reconstructor . Conversión (D/A) digital - analógica.

A continuación se muestra un ejemplo típica.

El procesador digital de señales puede ser programado para desarrollar una variedad de operaciones tales como: filtrado, análisis espectral y algunos otros algoritmos de procesamiento de señal. Dependiendo de la velocidad y requerimientos computacionales de la aplicación, el procesador digital de señal podrá ser realizada por herramientas tales como: computadoras de uso general, minicomputadoras, circuitos especiales de procesamiento de señal y otros equipos digitales dedicados a desarrollar una tarea particular de procesamiento de señal.

1.2 Señales analógicas.

Comenzaremos por revisar algunos temas importantes de la teoría analógica. Una señal analógica se describe como una función en el tiempo, digamos x(t). Su transformada de Fourier X(Ω) de x(t) es el espectro de frecuencia de la señal.

$$X(\Omega) = \int_{-\infty}^{\infty} x(t) e^{-j\Omega t} dt ,$$

en donde Ω es la frecuencia[1] cuyas unidades son [radianes / segundo]. La frecuencia ordinaria f en [hertz] o [ciclos / segundo] se relaciona a Ω por:

$$\Omega = 2\pi f$$

El significado físico de X(Ω) se entiende con más claridad con la transformada inversa de Fourier que nos expresa una señal arbitraria x(t) como una superposición de ondas senoidales de diferentes frecuencias:

$$x(t) = \int_{-\infty}^{\infty} X(\Omega) e^{j\Omega t} \frac{d\Omega}{2\pi}$$

La importancia relativa de cada componente sinusoide está dada por la cantidad de X(Ω). La transformada de Laplace se define como:

$$X(s) = \int_{-\infty}^{\infty} x(t) e^{-st} dt$$

Esta ecuación se reduce a la transformada de Fourier bajo la substitución $s = j\Omega$. El plano s de polos y ceros de las transformadas proveen un conocimiento más profundo de la naturaleza de la señal. Por ejemplo una señal senoidal típicamente exponencial de la forma

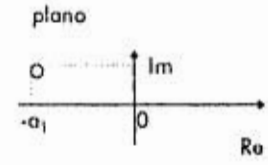
$$X(t) = e^{-a_1 t} e^{j\Omega_1 t} u(t) = e^{s_1 t} u(t)$$

en donde $s_1 = -a_1 + j\Omega_1$, tiene transformada de Laplace

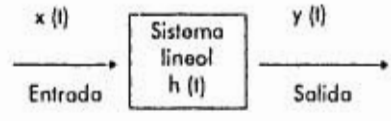
---

[1] Utilizamos la notación Ω para denotar la frecuencia en unidades de [radianes / segundos], y nos reservamos la notación ω para denotar la frecuencia digital en [radianes / muestra].

$$X(s) = \frac{1}{s - s_1}$$



con un polo en $s=s_1$, que coe en la parte izquierda del plano s. Después debemos considerar la respuesta de un sistema lineal a una entrada de señal x(t):



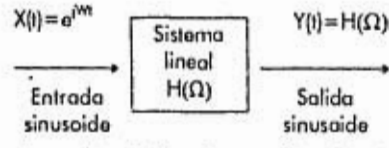
El sistema se caracteriza completamente por la respuesta al impulso unitario de la función h(t). La salida y(t) se obtiene en el dominio del tiempo por la convolución:

$$y(t) = \int_{-\infty}^{\infty} h(t - t')x(t')dt'$$

o por la multiplicación:

$$y(\Omega) = H(\Omega)X(\Omega)$$

en donde $H(\Omega)$ es la respuesta en frecuencia del sistema, definido como la transformada de Fourier de la respuesta al impulso h(t):



La figura anterior ilustra la acción del filtrado para filtros lineales. Una frecuencia dada de componente Ω es atenuada (o amplificada) la cantidad de $H(\Omega)$ por el filtro. Más precisamente, una entrada sinusoide de frecuencia Ω reaparecerá en la salida modificada por un factor $|H(\Omega)|$ y adelantada o atrasada en fase la cantidad de $\arg(H(\Omega))$:

$$x(t) = e^{j\Omega t} \Rightarrow y(t) = H(\Omega)e^{j\Omega t} = |H(\Omega)|e^{j(\Omega t + \angle H(\Omega))}$$

Por superposición, si la entrada consiste de la suma de dos ondas senoidales de frecuencia Ω_1 y Ω_2 y amplitudes relativas A_1 y A_2:

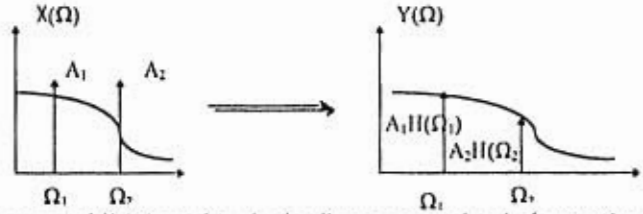
$$x(t) = A_1 e^{j\Omega_1 t} + A_2 e^{j\Omega_2 t},$$

después de filtrar, la respuesta continua o la salida será:

$$y(t) = A_1 H(\Omega_1) e^{j\Omega_1 t} + A_2 H(\Omega_2) e^{j\Omega_2 t}.$$

Cabe notar que el filtro modifica las amplitudes relativas de las sinusoides, pero no sus frecuencias. El efecto del filtro puede observarse también en el dominio de la frecuencia usando la ecuación siguiente.

$$Y(\Omega) = H(\Omega).X(\Omega)$$



La entrada espectral X (Ω) consiste de dos líneas espectrales de frecuencias Ω_1 y Ω_2, como puede observarse al tomar la transformada de Fourier de x(t):

$$x(\Omega)=2\pi A_1 \delta(\Omega-\Omega_1)+2\pi A_2 \delta(\Omega-\Omega_2)$$

el espectro de salida correspondiente Y(Ω) se obtiene de la ecuación:

$$Y(\Omega) = H(\Omega).X(\Omega)$$

en donde,

$$Y(\Omega)=H(\Omega)X(\Omega)=H(\Omega)(2\pi A_1 \delta(\Omega-\Omega_1)+ 2\pi A_1 \delta(\Omega-\Omega_2))$$
$$= 2\pi A_1 H(\Omega_1)\delta(\Omega-\Omega_1)+2\pi A_2 H(\Omega_2)\delta(\Omega-\Omega_2)$$

La importancia de los filtros lineales radica en que el diseñador tiene control absoluto sobre la forma de la respuesta en frecuencia H(Ω) del filtro. Por ejemplo, si la componente senoidal Ω_1 representa una señal deseada y Ω_2 una interferencia

indeseada, entonces un filtro podrá ser diseñado para que Ω_1 pase a través de él mientras que al mismo tiempo filtre y elimine la componente Ω_2. Dicho filtro deberá tener $H(\Omega_1)=1$ y $H(\Omega_2)=0$

1.3 Teorema de Muestreo.

A continuación estudiaremos el proceso de muestreo, ilustrado en la figura 1.3.1, la señal analógica se mide periódicamente cada T segundos. El tiempo en la señal se discretiza en unidades de intervalos de muestreo t:

$$t=nT, \quad n=0,1,2,3,...$$

Considerando la cadena resultante de muestras como una señal analógica, observamos que el proceso de muestreo representa una operación drástica y perturbante de la señal x(t). Este proceso introduciró una gran cantidad de frecuencias altas no deseadas en nuestro espectro de frecuencia de la señal muestreada. Para evitar esta, debemos respondernos dos preguntas:

1. ¿Cuál es el efecto del muestreo en el espectro de frecuencia de la señal original?
2. ¿Cómo escogemos un intervalo de muestreo para obtener una señal?

Trataremos de contestarnos estas preguntas intuitivamente en un principio y después lo haremos más formalmente usando la transforma de Fourier. Observamos que aunque el proceso de muestreo genera componentes con frecuencias altas, estas componentes aparecen de una manera muy singular. La frecuencia de la señal original se replica periódicamente con un solo periodo. Este periodo esta dado por la razón de muestreo:

$$f_s = 1/T$$

Considérese como ejemplo una señal senoidal $x(t)=e^{2\pi ft}$ con frecuencia f. Antes del muestreo, su espectro de frecuencia consiste en una línea espectral simple en f. Después del muestreo el espectro de la señal senoidal será $x(nT)=e^{2\pi fnT}$ que es la replica periódica de la señal original a intervalos de f_s, como se muestra en la figura 1.3.2.

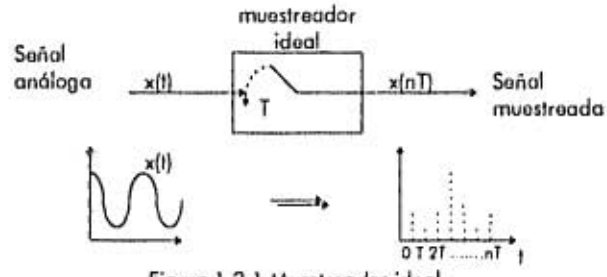


Figura 1.3.1 Muestreador ideal

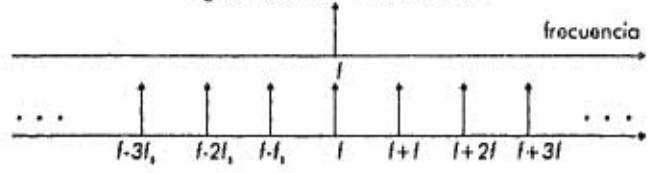


Figura 1.3.2 Réplico del espectro causada por el muestreo.

Nótese también que observando la réplica de la señal patrón, no podemos decir cuál es la frecuencia original. La señal original puede ser cualquiera de las réplicas. Así $f'=f+mf_s$, m=0,±1,±2,±3,... Este fenómeno sucede debido a que cualquiera de estas señales tiene el mismo periodo de réplica . A esta confusión potencial entre la frecuencia original y las otras frecuencias se le llama indistinción (*aliasing*) y podrá evitarse si se satisfacen las condiciones del teorema de muestreo.

El teorema de muestreo nos provee de una respuesta cuantitativa a la pregunta de ¿cómo escoger una frecuencia de muestreo T? Claramente, T deberá ser lo suficientemente pequeña para que la variación de la señal entre muestras no se pierda.

Pero, ¿Cuánto es "suficientemente pequeña"? Sería impráctico escoger T demasiado pequeña ya que tendríamos demasiadas muestras para procesar. Esto se ilustra en la figura 1.3.3 en donde T es demasiado pequeña para resolver el detalle de la señal 2, pero no suficientemente pequeño para resolver el detalle de la señal 1.

En términos de la frecuencia de muestreo f_s se mide en unidades de [muestras / segundo] o [Hertz] y representan la densidad de muestras por unidad de tiempo. Así una señal rápidamente variable deberá muestrearse con una mayor f_s, y una señal lentamente variable deberá muestrearse con una menor razón de muestreo.

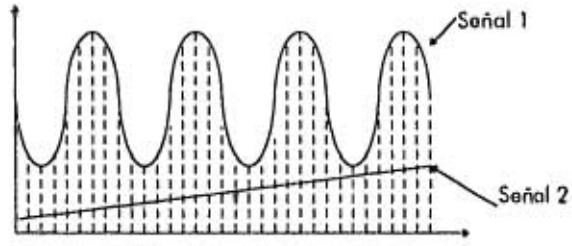


Figura 1.3.3. La señal 2 está sobremuestreada.

1.3.1 Teorema de Muestreo.

Un criterio más cuantitativo al anteriormente mencionado lo provee el teorema de muestreo. El teorema de muestreo demuestra que para una representación más acertada de la señal x(t) en intervalos finitos de tiempo x(nT), dos condiciones deberán cumplirse.

1. La señal x(t) deberá limitarse por bandas de frecuencia. Su espectro de frecuencia deberá estar limitado para contener frecuencias hasta de un valor máximo, f_{max}, y ninguna frecuencia fuera de este límite. El gráfico de la limitación en las bandas se muestra en la figura 1.3.4.

2. La frecuencia de muestreo deberá escogerse de por lo menos el doble de la máxima frecuencia, esta es:

$$f_s => 2f_{max}$$

en términos del intervalo de muestreo: $T \leq \frac{1}{2f_{max}}$

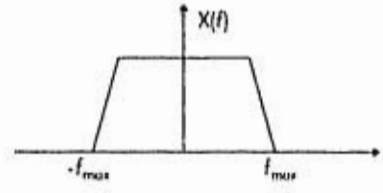


Figura 1.3.4

La frecuencia mínima de muestreo permitida por el teorema de muestreo, o sea $f_s=2f_{max}$, se llama razón de Nyquist. Para valores arbitrarios de f_s, la cantidad $f_s/2$ se llama frecuencia de Nyquist o frecuencia de doblez. Esta frecuencia define los puntos finales del intervalo de frecuencia de Nyquist;

$$\left[-\frac{f_s}{2}, \frac{f_s}{2}\right] = Intervalo\_Nyquist.$$

La frecuencia de Nyquist $f_s/2$ define también las frecuencias de corte del filtro analógico pasabajas y de los filtros posteriores requeridos para operaciones de procesamiento de señales DSP. Los valores de f_{max} y f_s dependen de la aplicación. Algunos valores típicos para aplicaciones comunes de procesamiento de señales se muestran en la tabla 1.

Aplicación	f_{max}	f_s
geofísico	500 Hz	1 kHz
biomédico	1 kHz	2 kHz
mecánico	2 kHz	4 kHz
voz	4 kHz	8 kHz
audio	20 kHz	40 kHz
video	4 MHz	8 kHz

Tabla 1

1.3.2. Prefiltros de distinción (*antiallasing*).

Las implicaciones prácticas del teorema de muestreo son muy importantes. Como la mayor parte de las señales no están limitadas, deberán estarlo mediante un filtro pasabajas antes de muestrearse.

Para muestrear una señal a una frecuencia fs y satisfacer la condición del teorema de muestreo, la señal deberá estar prefiltrada por un filtro analágico conocido como prefiltro antialiasing o de distinción. La frecuencia de corte del prefiltro, fmax, deberá ser tomada a lo más a la frecuencia de Nyquist fs/2, esto es; fmax =< fs/2. Esta operación se muestra en la figura 1.3.5.

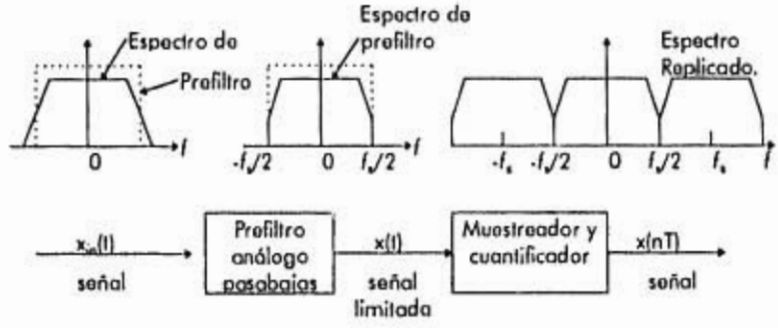


Figura 1.3.5. Filtro antialiasing

Es importante subrayar que f_s deberá escogerse lo suficientemente alta, de tal forma que después de la operación de prefiltrado, el espectro que quede dentro del intervalo de Nyquist $[-f_s/2, f_s/2]$ contenga todos los componentes significantes para la aplicación.

1.4 Muestreo de ondas senoidales.

Las dos condiciones del teorema de muestreo, o sea, que x(t) sea acotada por bandas y el requerimiento de que $fs => 2f_{max}$, pueden derivarse intuitivamente considerando el muestreo de señales senoidales únicamente. La figura 1.4.1 muestra una onda senoidal de frecuencia f, $x(t) = \cos(2\pi ft)$ que ha sido muestreada a tres intervalos de tiempo diferentes: fs=8f, fs=4f, fs=2f. Esto corresponde a tomar 8 ,4 y 2 muestras en cada ciclo de una sinusoide.

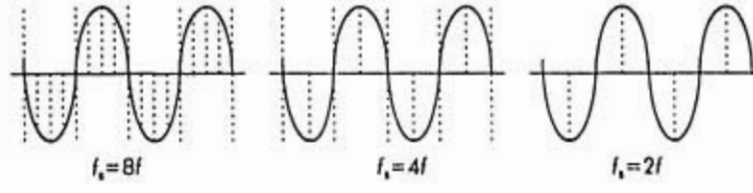


Figura 1.4.1. Muestras de separación de señal senoidal con intervalos de muestreo 8f,4f,2f

Una simple observación de estas figuras nos lleva a la conclusión de que el mínimo de muestras por ciclo es de dos. La representación de una sinusoide por dos muestras por ciclo es apenas adecuada[2] , pero al menos incorpora la naturaleza de subir y bajar de la señal senoidal. El número de muestras por ciclo está dado por la cantidad fs/f:

$$\frac{fs}{f} = \frac{muestras / segundo}{ciclos / segundo} = \frac{muestras}{ciclo}$$

Aún para muestrear una sola señal senoidal requerimos:

$$\frac{fs}{f} \geq 2 muestra / ciclo \Rightarrow fs \geq 2f$$

A continuación considérese el caso de una señal arbitraria x(t). De acuerdo con la transformada inversa de Fourier de la ecuación

$$x(t) = \int_{-\infty}^{\infty} X(\Omega) e^{j\Omega t} \frac{d\Omega}{2\pi}$$

x(t) podrá expresarse como una combinación de ondas senoidales. El muestreo adecuado de x(t) se conseguirá únicamente si todo componente de la onda senoidal x(t) se muestrea según lo antes especificado.

Esto requiere que la señal x(t) sea limitada por bandas, de no ser así, la señal tendrá componentes arbitrarios de frecuencias altas f, y para muestrear estos componentes adecuadamente, necesitaremos frecuencias orbitrariamente altas fs. Si la señal se limita por bandas a una frecuencia máxima f_{max} entonces al escoger $fs => 2f_{max}$, muestrearemos adecuadamente el componente de más rápida variación de x(t) incluyendo con esto todos los componentes de menor variación. Considérese el caso especial:

---

[2] Dependiendo de la fase de la onda senoidal. Ejemplo; muestrear una señal en el punto cero de las ordenadas resultará de valores cero para las muestras.

$$x(t) = A_1\cos(2\pi f_1 t) + A_2\cos(2\pi f_2 t) + \ldots\ldots + A_{max}\cos(2\pi f_{max} t)$$

en donde f_i se lista en orden ascendente. Las candicianes

$$2f_1 \le 2f_2 \le \ldots\ldots \le 2f_{max} \le f_s$$

nos indican que todo componente de x(t), y por la tanto x(t) misma, es muestreada adecuadamente.

1.4.1 Reconstrucción análoga y no distinción (*aliasing*).

A continuación discutiremos las efectos de indistinción (*aliasing*) que resultan si no se cumplen las condiciones del teorema de muestreo. Considérese la versión compleja de una senoidal:

$$x(t) = e^{j\Omega t} = e^{2\pi f t}$$

y su versión muestreada obtenida al hacer t=nT

$$x(nT) = e^{j\Omega Tn} = e^{2\pi fTn}$$

Defínase también la siguiente familia de senoidales, para m=0, ±1,±2,...,

$$x_m(t) = e^{2\pi(f + mf_s)t}$$

y su versión muestreada,

$$x_m(nT) = e^{2\pi(f + mf_s)Tn}$$

usando la propiedad $f_sT=1$ y su identidad trigonamétrica,

$$e^{2\pi mf_sTn} = e^{2\pi mn} = 1$$

encontramos que aún cuando las señales $x_m(t)$ son diferentes entre sí, sus valores del muestreo son iguales; ciertamente,

$$x_m(nT) = e^{2\pi(f + mf_s)Tn} = e^{2\pi fTn}e^{2\pi mf_sTn} = e^{2\pi fTn} = x(nT)$$

en términos de sus valores de muestreo, las señales $x_m(t)$ son indistinguibles (*aliased*). El conocimiento de los valores $x(nT)=x_m(nT)$ no es suficiente para determinar cuál de ambas es la señal original muestreada. Puede ser cualquiera de ambas $x_m(t)$. En otras palabras el conjunto de frecuencias,

$$f, f \pm f_s, f \pm 2f_s, \ldots, f \pm mf_s, \ldots$$

son equivalentes entre sí. El efecto del muestreo es el de reemplazar la frecuencia original f por el conjunto replicado. Esta es la explicación intuitiva de la propiedad del espectro de réplica.

Dado que los valores x(nT) no determinan únicamente a la señal analógica, surge una pregunta, ¿Qué señal se obtendrá si estas muestras se alimentan a un reconstructor analógico, como se muestra en la figura 1.4.2?

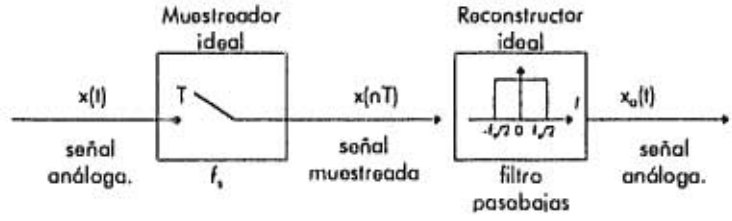


Figura 1.4.2. Reconstructor Ideal

Un reconstructor analógico extrae de la señal muestreada todos los componentes de frecuencia que caen dentro del intervalo de Nyquist $[-f_s/2, f_s/2]$ y remueve todas las frecuencias fuera de este intervalo. En otras palabras, un reconstructor analógico ideal actúa como un filtro pasabajas con una frecuencia de corte igual a la frecuencia de Nyquist $f_s/2$.

De entre las frecuencias en el conjunto replicado, existe una frecuencia única que cae en el intervalo de Nyquist[3]. Esta señal se obtiene al reducir la señal original f módulo f_s, que se consigue sumando o sustrayendo de f los suficientes múltiplos de f_s hasta que caiga en el intervalo simétrico de Nyquist $[-f_s/2, f_s/2]$. Denotamos esta operación como:

$$f_a = f \bmod(f_s)$$

Esta es la frecuencia en el conjunto réplica que será utilizada por el reconstructor analógico. De tal forma que la señal reconstruida será:

---

[3] La única excepción es cuando cae exactamente en el lado izquierdo o derecho del intervalo, $f = +\text{-}f_s/2$.

$$x_a(t) = e^{j 2\pi f_a t}$$

Se puede observar que $f_a = f$, sólo si f cae en el intervalo de Nyquist; sólo si $|f| \leq f_s/2$ que es equivalente al teorema de muestreo. Si f cae fuera del intervalo de Nyquist, es decir, si $|f| > f_s/2$, en donde se viola el teorema de muestreo, la frecuencia indistinguible (*aliased*) f_a será diferente de f y la señal reconstruida $x_a(t)$ será diferente de x(t), aún cuando las dos concuerden en los periodos de muestreo $x_a(nT) = x(nT)$.

Es instructivo también graficar como en la figura 1.4.3 la frecuencia *aliased* $f_a = f \bmod(f_s)$ contra la frecuencia real f. Obsérvese como la línea recta $f_{real} = f$ se reemplaza por señales en segmentos paralelos de translación de periodos de Nyquist múltiplos de f_s.

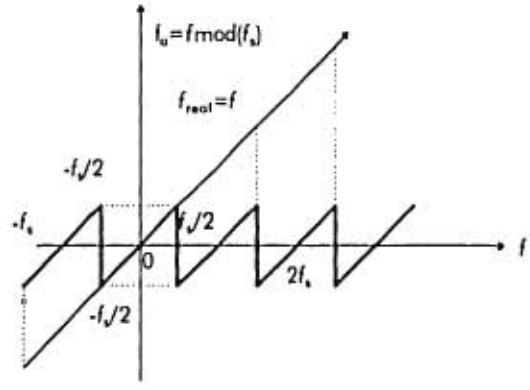


Figura 1.4.3 $f \bmod (f_s)$ vs f.

En resumen, efectos potenciales de indistinción (*aliasing*) que pueden surgir en la fase de reconstrucción de operaciones de DSP (Procesamiento Digital de Señales) pueden evitarse si uno se asegura de que todos los componentes de frecuencia pueden ser muestreados satisfaciendo la condición del teorema de muestreo, $|f| \leq f_s/2$, que todos los componentes de la frecuencia caigan dentro del intervalo de Nyquist. Se debe

asegurar con un prefiltro pasabajas de distinción (*antialiasing*) que remueva todas las frecuencias por debajo de la frecuencia de Nyquist $f_s/2$.

1.5 El proceso de Cuantificación.

El muestreo y la cuantificación son requisitos necesarios para una operación de procesamiento digital de señales analógicas. Un muestreador y un cuantificador se muestran en la figura 1.5.1. El capacitor de espera en el muestreador retiene cada muestra x(nT) al menos T segundos durante las cuales el convertidor A/D lo transforma en una muestra cuantificable $x_Q(nT)$, que se representa por un número finito de bits, digamos *B* bits. La palabra de *B* bits se transfiere al procesador digital de señales.

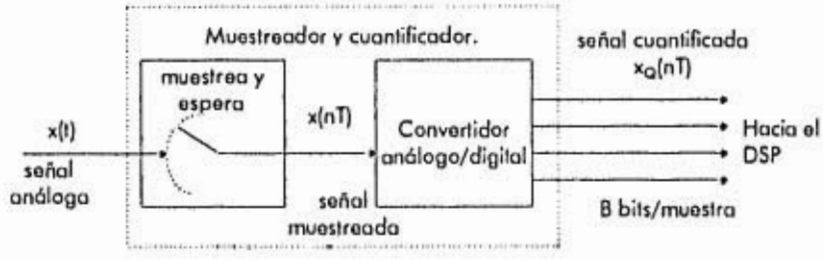


Figura 1.5.1 Conversión análoga a digital.

Después de procesarla digitalmente, las palabras resultantes de *B* bits se alimentan en un convertidor D/A (digital analógico) que las regresa a un formato analógico, generando con esto una salida escalonada. En la práctica, la operación de muestreo/(espera,retiene) (*sample/hold*) y ADC pueden ser módulos separados o pueden residir dentro del mismo microcircuito.

La muestra cuantificada $x_Q(nT)$, representada por *B* bits, puede tomar solo una de 2^B valores posibles. Un convertidor A/D se caracteriza por un rango de escala completa R

que se divide uniformemente (por un cuantificador uniforme) en 2^B niveles de cuantificación como se muestra en la figura 1.5.2. El espacio entre niveles,

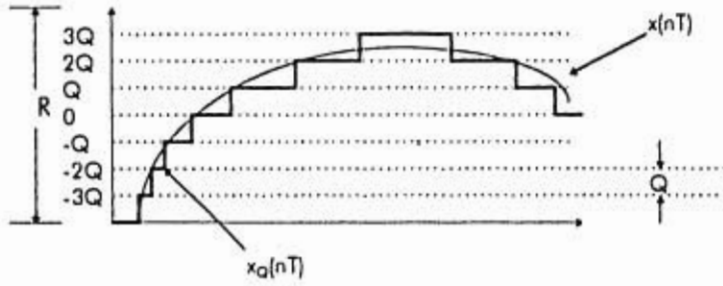


Figura 1.5.2 Cuantificación de señal.

llamado ancho de cuantificación (*quantization width*) o resolución de cuantificado está dado por:

$$Q = \frac{R}{2^B}$$

Esta ecuación puede escribirse también de la forma:

$$2^B = \frac{R}{Q}$$

que nos provee de la cantidad de niveles de cuantificación. Algunos valores típicos de R en la práctica están entre los 1-10 volts. La figura 1.5.2 nos muestra el caso de B=3 o 2^B=8 niveles, y supone un ADC (Circuito analógico digital) bipolar para el cual los valores posibles de cuantificación caen en el rango simétrico.

$$-\frac{R}{2} \le X_Q(nT) < \frac{R}{2}$$

Para un circuito (ADC) tenemos $0 \le x_Q(nT) < R$. En la práctica la señal de entrada x(t) deberá ser previamente acondicionada por métodos analógicos para que caiga en toda la escala del cuantificador, esto es, $-R/2 \le x(t) < R/2$, antes de ser mandada al

muestreador y cuantificador. El límite superior R/2, de toda la escala no se considera dentro de los niveles, así el nivel máximo es R/2 - Q.

En la figura 1.5.2 se muestra que la cuantificación de x(t) se realiza redondeando, es decir, reemplazando cada valor de x(t) por su valor de cuantificación más cercano. También es posible realizar la cuantificación, truncando la señal en el nivel inmediato inferior a ésta. El redondeo es recomendable en la práctica, ya que produce una señal menos sesgada dentro de la cuantificación de la misma.

El *error de cuantificación* es el error que resulta de usar una señal cuantificada $x_Q(nT)$ en vez de la señal original x(nT),[4]

$$e(nT)=x_Q(nT)-x(nT)$$

En general, el error en cuantificar un número x que cae en $\left[-\frac{R}{2},\frac{R}{2}\right]$ es:

$$e=x_Q-x$$

En donde x_Q es el valor cuantificado. Si x cae entre dos niveles, será redondeado hacia arriba o hacia abajo dependiendo de cual sea el nivel más cercano. Si x cae en la mitad superior (inferior) , será redondeado hacia arriba o hacia abajo. Aunque el error e solo pueda tomar los valores[5]

$$-\frac{Q}{2}\le e<\frac{Q}{2}$$

Así el error máximo es $e_{max}=Q/2$ en magnitud. Esto es una estimación extrema para el error típico real. Para obtener un valor más representativo para el error promedio, consideremos la media y el valor de la media cuadrática de una e definida por:

$$\bar{e}=\frac{1}{Q}\int_{-Q/2}^{Q/2}e\,de=0 \quad \text{y} \quad \overline{e^2}=\frac{1}{Q}\int_{-Q/2}^{Q/2}e^2=\frac{Q^2}{12}$$

---

[4] Una definición más natural sería $e(nT)=x(nT)-x_Q(nT)$. La ecuación mostrada es más conveniente para hacer modelos de cuantificación.

[5] Si el punto intermedio entre niveles siempre se redondea hacia arriba, debemos tomar estrictamente $-Q/2<e\le Q/2$.

Ecuación 1.5.1

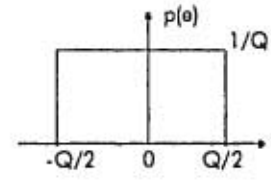
El resultado $\bar{e}=0$ nos dice que, en promedio la mitad de los valores son redondeados hacia arriba y la otra mitad hacia abajo; aunque $\bar{e}$ no pueda utilizarse como un error representativo. Un valor más típico es el error raíz media cuadrática *[root mean square error]* definido por:

$$e_{rms} = \sqrt{\overline{e^2}} = \frac{Q}{\sqrt{12}}$$

Ecuación 1.5.2

La **Ecuación 1.5.1** puede interpretarse de una manera más probabilística, suponiendo que el error de cuantificación es aleatoriamente variable y que se distribuye de una manera uniforme sobre el rango $-\frac{Q}{2} \le e < \frac{Q}{2}$, teniendo con esto una densidad de probabilidad:

$$p(e) = \left\{ \frac{1}{Q} si - \frac{Q}{2} \le e < \frac{Q}{2}, 0 \; de \; otra \; manera. \right\}$$



La normalización 1/Q se requiere para garantizar :

$$\int_{-Q/2}^{Q/2} p(e)de = 1$$

De aquí concluimos que la **Ecuación 1.5.1** representa las expectativas estadísticas:

$$E(e) = \int_{-Q/2}^{Q/2} ep(e)de \quad y \quad E(e^2) = \int_{-Q/2}^{Q/2} e^2 p(e)de$$

Pensando en R y Q como los rangos de la señal y del ruido de cuantificación, la razón en la ecuación $2^B = \frac{R}{Q}$ es la razón de la señal vs el ruido *[signal to noise ratio (snr)]*.

Puede expresarse en dB:

$$2\log_{10}\left(\frac{R}{Q}\right) = 20\log_{10}(2^B) = B * 20\log_{10}(2) \quad \text{ó}$$

$$SNR = 20\log_{10}\left(\frac{R}{Q}\right) = 6B \text{ dB}$$

que se refieren a la regla de 6 dB por bit. La ecuación de SNR se llama "rango dinámico del cuantificador". $Q = \frac{R}{2^B}$ y $e_{rms} = \sqrt{e^2} = \frac{Q}{\sqrt{12}}$ pueden utilizarse para determinar el ancho de palabra B si se proporciona el rango completo y el error *rms* deseado.

La interpretación probabilística del ruido por cuantificación es muy útil para determinar los efectos de la cuantificación en tanto que éstos se propagan a través del resto del proceso digital del sistema. La ecuación $e(nT)=x_Q(nT)-x(nT)$ se puede escribir de la forma:[6]

$$x_Q(n)=x(n)+e(n)$$

Podemos pensar de la señal cuantificada $x_Q(n)$ como una versión de la señal original a la que se le ha añadido un ruido. Un modelo aditivo del cuantificador se muestra en la figura 1.5.3.

---

[6] Por comodidad escribiremos x[nT] como x[n], etc.

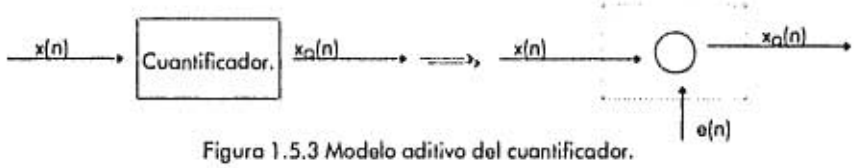


Figura 1.5.3 Modelo aditivo del cuantificador.

En general las propiedades estadísticas de la secuencia de sonido e(n) son muy complejas. Sin embargo para las señales de amplitud y banda amplias (*wide amplitude wide band)* ,que varían a través de todo el rango de la escala R cruzando a menudo todo los niveles de cuantificación; la secuencia e(n) puede suponerse estacionaria con media en cero, sonido blanco con densidad de probabilidad uniforme dentro del rango $[-Q/2, Q/2]$. Además, e(n) puede supanerse no correlacionada con la señal x(n). La potencia promedio de la varianza de e(n) es:

$$\sigma_e^2 = E\left[e^2(n)\right] = \frac{Q^2}{12}$$

La suposición de que e(n) sea sonido blanco significa que tiene una función delta autocorrelacionada (ver anexo A):

$$R_{ee}(k) = E\left[e(n+k)e(n)\right] = \sigma_e^2\delta(k)$$

para todo el último k. Similarmente, ser no correlacionada con x(n) significa que tiene correlación cruzada igual a cero:

$$R_{ex}(k) = E\left[e(n+k)x(n)\right] = 0$$

para todo k.

Este modelo no es adecuado para señales de baja amplitud y de muy lenta variación. Por ejemplo, una señal senoidal que cae exactamente en el punto intermedio de dos niveles y que tiene amplitud menor que Q/2 será cuantificada como una señal cuadrada con todos los saltos redondeados hacia arriba y todos los bajos redondeados hacia abajo. El error resultante e(n) será altamente periódico, correlacionado de

muestra a muestra y no se asemeja al sonido blanco aleatorio. También será altamente correlacianado con entradas senoidales x(n).

En aplicaciones digitales de audio, las distorsiones de cuantificación provenientes de señales de bajo nivel son denominadas ruido granulado *(granulation noise)* y corresponden a sonidos desagradables. Éstos pueden ser virtualmente eliminados, mediante el uso de un excitador *(dither)* que es sanido de bajo nivel agregado a la señal antes de la cuantificación.

El efecto benéfico de la excitación *(dithering)* será el de hacer que toda la cuantificación se comporte como un sonido blanco, que es mucho más aceptable que el tosco sonido granulado de la señal no modificada. En contraparte, realizar este tipo de filtrado reduce la razón de señal contra ruido entre 3 y 6 dB, dependiendo del tipo de excitador *(dither)* usado.

Capítulo 2.

Sistemas de tiempo discreto.

2.1 Introducción.

La relación de entrada/salida (I/O) en un sistema lineal de tiempo invariable (LTI) está dada por la convolución en tiempo discreto de la respuesta al impulso de la señal de entrada del sistema.

Los sistemas lineales de tiempo invariable (LTI) se clasifican en (FIR) respuesta al impulso finito o (IIR) respuesta a impulso infinito, dependiendo de si su respuesta tiene una duración finita o infinita. Nuestro objetivo primordial es el desarrollo de algoritmos prácticos para el caso de respuesta al impulso finito.

Dependiendo de nuestra aplicación y de nuestro hardware, una operación de filtrado para el caso FIR puede organizarse para ser realizada en dos tipos de operación: por muestra o por bloque. En el procesamiento por bloque, la señal de entrada se considera un solo bloque de las muestras de una señal. El bloque se filtra convalucionándolo con el filtro, generando así la señal de salida como otro bloque de muestras.

Si la entrada es muy larga o de duración infinita, este método requiere modificaciones, por ejemplo, rompiendo la entrada en múltiples bloques de tamaño manejable, filtrando los bloques uno a la vez y pegando los bloques a la salida para formar así la salida total. El filtrado de los bloques puede implementarse de varias maneras, tales como con una simple convolución o por una convolución más rápida, vía la Transformada de Fourier.

En el caso de procesamiento por muestras, las muestras son procesadas una a la vez . El filtro opera como una máquina de estados; esto significa que cada muestra que entra se utiliza junto con el estado interno del filtro para procesar así la muestra de salida actual y restablecer el estado del filtro preparándolo para procesar la muestra sucesiva.

Esta aproximación es útil en aplicaciones en tiempo real involucrando señales con una entrada prolongada. Es útil también en aplicaciones de filtros adaptativos, en donde el

filtro cambia después de procesar cada muestra. Además se han implementado eficientemente con la familia actual de microcircuitos DSP tales como el Texas Instrument TMS320, el Bell Labs AT&T DSP16/32, el Motorola DSP56K/96K y la familia de los dispositivos analógicos ADSP2101. La arquitectura e instrucciones de estos microcircuitos se han optimada para tales operaciones de procesamiento de muestra por muestra. En el anexo D se presenta un circuito de procesamiento de señales.

2.2 Reglas de entrada y salida.

1. Un sistema de tiempo discreto, como el que se muestra en la figura 2.1, es un procesador que transforma una secuencia de entrada de muestras de tiempo discreto x(n) en una salida de muestras y(n), de acuerdo con algunas reglas de entrada/salida que especifican la manera de procesar la secuencia de salida y(n) conociendo la secuencia de entrada x(n). En el procesamiento de muestra por muestra, podemos pensar en reglas de entrada salida I/O como el procesamiento de las entradas una a la vez:[7]

$$\{x_0, x_1, x_2, \ldots, x_n, \ldots\} \xrightarrow{H} \{y_0, y_1, y_2, \ldots, y_n, \ldots\}$$

esto es, $x_0 \xrightarrow{H} y_0, x_1 \xrightarrow{H} y_1, x_2 \xrightarrow{H} y_2$, y así sucesivamente. En el procesamiento por bloques, pensamos en la entrada como un bloque o vector de muestras de señal procesadas como un sistema completo que produce un bloque de salida:

$$\mathbf{x} = \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ . \\ . \\ . \end{bmatrix} \xrightarrow{H} \begin{bmatrix} y_0 \\ y_1 \\ y_2 \\ . \\ . \\ . \end{bmatrix} = \mathbf{y}$$

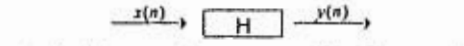
[7] Por simplicidad escribiremos $\{x(0), x(1), x(2)\ldots\}$ con $\{x_0, x_1, x_2\ldots\}$.

Aunque la regla I/O mapea el vector de entrada **x** dentro del vector de salida **y** de acuerdo a un mapeo funcional:

$$y = H[x]$$

Para un sistema lineal se convierte en una transformación lineal por media de una matriz **H**, **y=Hx**. Para sistemas lineales y de tiempo invariable, la matriz **H** tiene una estructura muy especial que se construye en función de la respuesta al impulso del sistema.

Algunos ejemplos de sistemas de tiempo discreto ilustran la gran variedad de reglas I/O como se dan a continuación.



entrada de la secuencia salida de la secuencia

Sistema de tiempo discreto

Ejemplo 1. y(n)=2x(n). Esto corresponde a una simple escalación de la entrada:

$$\{x_0,x_1,x_2,x_3,x_4,\ldots\} \xrightarrow{H} \{2x_0,2x_1,2x_2,2x_3,2x_4,\ldots\}$$

Ejemplo 2. y(n)=2x(n)+3x(n-1)+4X(n-2). Un promedio no ponderado de tres muestras de entrada. A cada instante n, el sistema deberá recordar la muestra anterior x(n-1) y x(n-2) para poder sumarlas.

Ejemplo 3. Aquí la regla I/O se especifica como una operación de procesamiento de bloques por media de una transformación lineal, transformando una entrada de tamaño 4 en un bloque de entrada $\{x_0,x_1,x_2,x_3\}$ en un bloque de tamaño 4.

$$\mathbf{y} = \begin{bmatrix} y_0 \\ y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \end{bmatrix} = \begin{bmatrix} 2 & 0 & 0 & 0 \\ 3 & 2 & 0 & 0 \\ 4 & 3 & 2 & 0 \\ 0 & 4 & 3 & 2 \\ 0 & 0 & 4 & 3 \\ 0 & 0 & 0 & 4 \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \end{bmatrix} = H\mathbf{x}$$

que es equivalente a la forma directa de la convolución del ejemplo 2. El bloque de salida es más grande que el bloque de entrada por dos muestras, debido a que este filtro tiene memoria 2. Las últimas dos muestras son transitorias y están generadas por las muestras remanentes cuando dejan de ingresar muestras. Si tenemos que filtrar bloques de tamaño 5 $\{x_0,x_1,x_2,x_3,x_4\}$, la transformación lineal tendrá una columna y fila de más:

$$\mathbf{y} = \begin{bmatrix} y_0 \\ y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \\ y_6 \end{bmatrix} = \begin{bmatrix} 2 & 0 & 0 & 0 & 0 \\ 3 & 2 & 0 & 0 & 0 \\ 4 & 3 & 2 & 0 & 0 \\ 0 & 4 & 3 & 2 & 0 \\ 0 & 0 & 4 & 3 & 2 \\ 0 & 0 & 0 & 4 & 3 \\ 0 & 0 & 0 & 0 & 4 \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \end{bmatrix} = H\mathbf{x}$$

Ejemplo 4. El ejemplo 2 también podrá ser expuesto en un proceso equivalente de muestra por muestra, descrito por el siguiente sistema de ecuaciones:

$$\begin{aligned} y(n) &= 2x(n) + 3w_1(n) + 4w_2(n) \\ w_2(n+1) &= w_1(n) \\ w_1(n+1) &= x(n) \end{aligned}$$

Las variables auxiliares w_1 y w_2 pueden considerarse como el estado interno del sistema. La muestra actual x(n) junto con el conocimiento de los estados internos $\{w_1(n),w_2(n)\}$ son suficientes para computar la salida actual y(n). La siguiente salida y(n+1) debida a la entrada x(n+1) requiere del conocimiento de los estados actualizados $\{w_1(n+1),w_2(n+1)\}$ pero éstos son disponibles con la muestra n, así en el tiempo n+1 tenemos:

$$\begin{aligned} y(n+1) &= 2x(n+1) + 3w_1(n+1) + 4w_2(n+1) \\ w_2(n+2) &= w_1(n+1) \\ w_1(n+2) &= x(n+1) \end{aligned}$$

Las operaciones son repetitivas de un instante de tiempo al siguiente y pueden resumirse por el siguiente algoritmo I/O de procesamiento de muestra por muestra,

que nos indica cómo procesar cada muestra que llega **x** para producir la muestra de salida **y** y actualizar los estados internos:[8]

Para cada entrada x hacer:

$$y:=2x+3w_1+4w_2$$

$$w_2:=w_1$$

$$w_1:=x$$

Una vez que los valores de los estados internos $\{w_1,w_2\}$ son utilizados en la operación de la salida **y** pueden ser actualizados por las últimas dos ecuaciones de asignación, tomando los valores que deberán tener para procesar la siguiente muestra. Así $\{w_1,w_2\}$ deberán guardarse cada vez que se llame el algoritmo. Para que $\{w_1,w_2\}$ sean actualizados, es importante que w_2 sea actualizado antes que w_1 , así deberá ser para prevenir sobreescribir los valores correctos.

Esto y el ejemplo anterior representan formulaciones equivalentes del mismo sistema discreto en el tiempo. El decidir qué fórmula utilizar depende de la naturaleza del tipo de aplicación, si la secuencia de la señal de entrada finita o infinito; o de si las muestras deben de procesarse una a la vez al tiempo en que llegan.

2.3 Linealidad e invariancia en el tiempo.

En una combinación lineal de dos o más señales de entrada, un sistema lineal tiene la propiedad de que la señal de salida puede obtenerse al realizar la misma combinación lineal de cada una de las entradas individualmente. Si $y_1(n)$ y $y_2(n)$ son las salidas provenientes de las entradas $x_1(n)$ y $x_2(n)$, entonces la salida debida a la combinación lineal de las entradas

$$x(n)=a_1x_1(n)+a_2x_2(n)$$

está dada por la combinación lineal de las salidas

---

[8] El símbolo := denota asignación, no es una ecuación, así, si a:=b significa que "a toma el valor de b".

$$y(n)=a_1y_1(n)+a_2y_2(n)$$

Para probar la linealidad uno deberá determinar separadamente las tres salidas y(n), $y_1(n)$, y $y_2(n)$ y después mostrar que satisfacen la ecuación anterior. Las operaciones requeridas se muestran en la figura 2.3.1

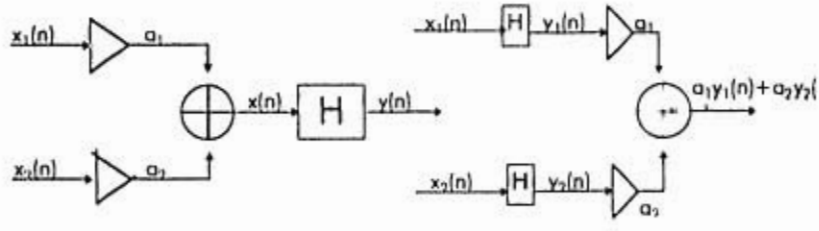


Figura 2.3.1 Prueba de linealidad.

Un sistema invariable en el tiempo es un sistema que permanece sin cambios. Esta quiere decir que si se le aplica una entrada en cualquier instante que ocasiona una salida, la misma salida será producida en cualquier otra instante si se aplica la misma entrada. La operación de espera o retraso de la señal por un tiempo de, digamos D, unidades de tiempo se muestra en la siguiente figura 2.3.2 que representa el traslado de x(n) hacia la derecha.

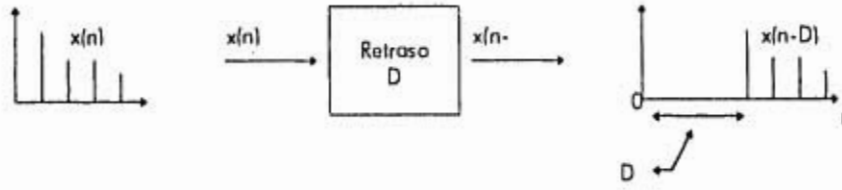


Figura 2.3.2 Atraso en el tiempo D muestras.

Un adelanto en el tiempo tendría un valor de D negativo y correspondería a una translación hacia el lado izquierdo.

La interpretación matemática de la invariancia en el tiempo puede expresarse con la ayuda de la figura 2.3.3. El diagrama superior de esta figura muestra una entrada x(n) que se aplica al sistema, produciendo la salida y(n).

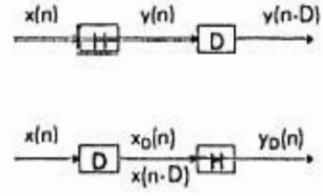


Figura 2.3.3 Prueba de invariancia en el tiempo.

El diagrama inferior muestra la misma entrada demorada D unidades de tiempo, la señal:

$$x_D(n)=x(n-D)$$

que a su vez se aplica al sistema, produciendo la salida; digamos $y_D(n)$.

Para probar si el sistema produce la misma salida a la salida actual en otro instante debemos tomar la salida y(n) producida en un inicio y guardarla; demorarla D unidades de tiempo, como se muestra en la figura 2.3.3. Así puede compararse con la salida $y_D(n)$ que será producida posteriormente. Si

$$y_D(n)=y(n-D)$$

el sistema será invariable en el tiempo. En otras palabras, demorar la entrada causa que la salida sea demorada, durante el mismo intervalo de tiempo. Equivalentemente en término de muestras, si

$$\{x_0,x_1,x_2,\ldots\}\xrightarrow{H}\{y_0,y_1,y_2,\ldots\}$$

entonces

$$\{0,0,0,\ldots,0,x_0,x_1,x_2,\ldots\} \xrightarrow{H} \{0,0,0,\ldots,0,y_0,y_1,y_2,\ldots\}$$
$$D.ceros$$

2.4 Respuesta al impulso.

Las sistemas lineales invariables en el tiempo se caracterizan singularmente por su secuencia de respuesta al impulso h(n), definida como la respuesta del sistema a un impulso unitario δ(n), como se muestra en la figura 2.4. El impulso unitario es el análogo discreto de la función delta de Dirac y se define como

$$\delta(n) = \begin{cases} 1 si\ n = 0 \\ 0 si\ n \neq 0 \end{cases}$$

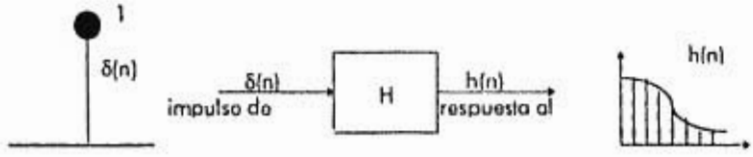


Figura 2.4 Respuesta al impulso unitario de un sistema LTI.

Tenemos que,

$$\delta(n) \xrightarrow{H} h(n)$$

La invariancia en el tiempo implica que si el impulso unitario se demora o se adelanta una cantidad de unidades de tiempo, entonces causará que la respuesta se demore la misma cantidad de unidades de tiempo h(n-D). Así.

$$\delta(n-D) \xrightarrow{H} h(n-D)$$

para cualquier valor positivo o negativo de D. Por otro lado, la linealidad implica que cualquier combinación lineal de las entradas causa la misma combinación lineal de salidas, por ejemplo, la suma de tres impulsos causará la suma de tres salidas;

$$\delta(n)+\delta(n-1)+\delta(n-2)\xrightarrow{H} h(n)+h(n-1)+h(n-2)$$

o generalmente, la combinación lineal ponderada de las tres entradas:

$$x(0)\delta(n)+x(1)\delta(n-1)+x(2)\delta(n-2)$$

causará la misma combinación ponderada de tres salidas:

$$x(0)h(n)+x(1)h(n-1)+x(2)h(n-2).$$

En general, una secuencia de entrada arbitraria {x(0),x(1),x(2),...} puede pensarse como la combinación lineal de impulsos unitarios ponderados atrasados y adelantados:

$$x(n)=x(0)\delta(n)+x(1)\delta(n-1)+x(2)\delta(n-2)+x(3)\delta(n-3)+...$$

Esto se sigue ya que cada término del lado derecho no es cero únicamente en su tiempo de retraso, por ejemplo en n=0, únicamente el primer término no es cero; en n=1 sólo el segundo término no es cero y así sucesivamente. Linealidad e invariabilidad en el tiempo implican que la secuencia correspondiente de salida puede obtenerse reemplazando cada impulso unitario retrasado por su correspondiente respuesta al impulso unitario:

$$y(n)=x(0)h(n)+x(1)h(n-1)+x(2)h(n-2)+x(3)h(n-3)+...$$

o escrito de forma más compacta,

$$y(n)=\sum_{m} x(m)h(n-m)$$

Esta es la convolución en tiempo discreto de la secuencia de entrada x(n) con el filtro h(n). Así los sistemas lineales de tiempo invariable LTI son convolventes.

2.5 Filtros tipo FIR y IIR.

Los sistemas discretos LTI pueden clasificarse en FIR o como se muestra en la figura 2.5.

Un filtro FIR tiene una respuesta al impulso h(n) que se extiende únicamente sobre un intervalo finito de tiempo, digamos $0 \le n \le M$, y es igual a cero después de esto.

$$\{h_0, h_1, h_2, \ldots, h_M, 0, 0, 0, \ldots\}$$

M se refiere al orden del filtro. La longitud de la respuesta al vector de respuesta al impulso $h=\{h_0,h_1,\ldots,h_M\}$ es:

$$L_h = M+1$$

Figura 2.5

Los coeficientes de la respuesta al impulso $\{h_0,h_1,\ldots,h_M\}$ se denominan con diferentes nombres, tales como coeficientes del filtro (*filter coeficients*), pesos del filtro (*filter weights*) o golpes del filtro (*filter taps*), dependiendo del contexto. En la forma directa de la convolución:

$$y(n) = \sum_{m} h(m)x(n-m)$$

todos los términos para m>M y m<0 no existirán, ya que por definición h(m) desaparece para estos valores de m sólo en términos $0 \le m \le M$ están presentes. Así la forma directa de la convolución simplificada o una suma finita forma:

$$y(n) = \sum_{m=0}^{M} h(m)x(n-m)$$

Ecuación de filtro FIR.

o explícitamente:

$$y(n)=h_0x(n)+h_1x(n-1)+h_2x(n-2)+....+h_Mx(n-M)$$

Un filtro IIR, por otro lado tiene una respuesta al impulso h(n) de duración infinita, definida sobre un intervalo $0 \le n < \infty$. La ecuación $y(n) = \sum_{m} h(m)x(n-m)$ tendrá ahora un número infinito de términos:

$$y(n) = \sum_{m=0}^{\infty} h(m)x(n-m)$$

Ecuación de filtro IIR.

La ecuación I/O no es realizable, ya que no podemos lidiar con un número infinito de términos. Por esto, debemos restringir nuestra atención a una subclase de filtros IIR, aquéllos para los cuales el número infinito de coeficientes del filtro $\{h_0,h_1,h_2,....\}$ no sean escogidos arbitrariamente, sino que se encuentren acoplados uno al otro a través de ecuaciones en diferencias lineales con coeficientes constantes.

Para que esta subclase de filtros IIR funcione, se deberán arreglar como una ecuación diferencial, esto permitirá la computación recursiva de la salida y(n).

2.6 Causalidad y Estabilidad.

Tal como las señales analógicas, las señales en tiempo discreto pueden ser clasificados en causales, anticausales o señales mezcladas como se muestra en la figura 2.6.1.

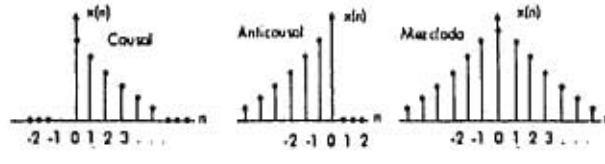


Figura 2.6.1 Señales causales, anticausales y mezcladas.

Una señal causal o de lado derecho x(n) existe sólo para $n \geq 0$ y desaparece para valores negativos $n<0$. Las señales causales son las señales más frecuentemente encontradas, ya que son del tipo que se generan en los laboratorios, por ejemplo cuando encendemos un generador o fuente de señal.

Una señal anticausal o de lado izquierdo existe sólo para $n \leq 1$ y desaparece para todo $n \geq 0$. Una señal mezclada de doble lado tiene ambos lados, un lado derecho y uno izquierdo.

Nuestro origen, n=0, junto con el eje del tiempo es un asunto de convención. Típicamente, se toma el tiempo en donde encendemos nuestro generador de señal o el tiempo en donde comenzamos nuestro procesamiento de señal. Así una señal que tiene ambos lados con respecto a un tiempo elegido es simplemente una señal que ha existido al comenzar nuestro proceso.

Los sistemas LTI pueden clasificarse también en términos de sus propiedades de causalidad, dependiendo de si su respuesta al impulso h(n) es causal, anticausal o mezclada. Para un sistema general de doble lado h(n), que puede extenderse desde $-\infty < n < \infty$, las ecuaciones I/O convolucionales se convierten en

$$y(n) = \sum_{m=-\infty}^{\infty} h(m)\,x(n-m).$$

Estos sistemas no pueden implementarse en tiempo real, como puede observarse al escribir algunos términos de m positivos y negativos.

$$y_n = \ldots + h_{-2}x_{n+2} + h_{-1}x_{n+1} + h_0x_n + h_1x_{n-1} + h_2x_{n-2} + \ldots$$

que indica que para poder computar la salida y(n) en el tiempo n, uno deberá saber las muestras del *futuro* x(n+1), x(n+2),..., que no se encuentran disponibles aún para el procesamiento.

Los sistemas anticausales y de doble lado son contra-intuitivos. Estos sistemas violan nuestro sentido de causalidad. Por ejemplo, en respuesta a un impulso unitario δ(n) que se le aplica al sistema en n=0, el sistema generará su respuesta de salida h(n),

Pero si h(-1) ≠ 0, el sistema habrá producido ya una muestra de salida en el tiempo n=-1, aun antes de que la respuesta al impulso haya sido aplicada a n=0.

Los filtros FIR de atenuación e interpolación se encuentran dentro de una clase de filtros de doble lado que únicamente son finitos y anticausales, esto quiere decir, que su parte anticausal tiene una duración finita, digamos de periodo $-D \le n \le -1$. Estos filtros se muestran en las figura 2.6.2. En general, la parte causal de h(n) puede ser o no finita. La ecuación I/O se convierte para esta clase de filtros en:

$$y(n) = \sum_{m=-D}^{\infty} h(m)x(n-m)$$

Una técnica para lidiar con tales filtros es el hacerlos causales, remplazando h(n) con su versión retrasada D unidades de tiempo, esto es:

$$h_D(n)=h(n-D)$$

Como se muestra en la figura 2.6.2, esta operación traslada h(n) al lado derecho D unidades, haciéndolo causal. La ecuación I/O de filtrado para un filtro causal h_D será

$$y_D(n) = \sum_{m=0}^{\infty} h_D(m)x(n-m)$$

y se podrá implementar en tiempo real. Es fácilmente demostrable que la secuencia resultante $y_D(n)$ es simplemente la versión retrasada de y(n) . Así las muestras de salida se computan correctamente pero salen con un retraso en el tiempo.

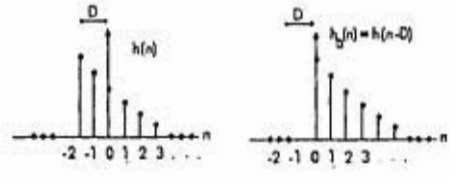


Figura 2.6.2 Filtro anticausal y su versión causal.

Además de las propiedades de causalidad los sistemas LTI pueden clasificarse por su estabilidad. Un sistema LTI estable es aquél que su respuesta al impulso h(n) llega a cero suficientemente rápido en tanto que $n \to \pm\infty$, de tal forma que la salida del sistema y(n) nunca diverge; permanece acotado por algún límite $|y(n)| \leq B$ si su entrada está limitada, digamos $|x(n)| \leq A$. O sea, un sistema es estable si: entradas limitadas siempre generen salidas limitadas.

Puede demostrarse que una condición necesaria y suficiente para que un sistema LTI sea estable, en el sentido de que su entrada/salida esté limitada, es que su respuesta al impulso unitario sea absolutamente adicionable:

$$\sum_{n=-\infty}^{\infty} |h(n)| < \infty$$

condición de estabilidad.

La estabilidad es esencial para aplicaciones de hardware o software de sistemas LTI ya que garantiza que las operaciones numéricas requeridas para procesar la sumas de las convoluciones de I/O o sus ecuaciones en diferencias equivalentes se mantengan estables y nunca crezcan fuera de ciertos límites. En aplicaciones de hardware, estas inestabilidades saturarán rápidamente los registros del hardware y en aplicaciones de software excederán los rangos numéricos de la mayaría de las computadoras, dando como resultado valores numéricos sin sentido.

Debido a consideraciones prácticas, siempre se preferirá la estabilidad sobre la causalidad.

Si la parte anticausal de un sistema estable h(n) tiene una duración finita, entonces será posible hacer al sistema causal mediante un retraso en el tiempo. Si por otro lado la parte anticausal es infinita, entonces h(n) podrá ser manipulado como aproximación por el siguiente procedimiento. Debido a que h(n) es estable, se aproximará a cero para n grandes. Debido a lo anterior, uno podrá escoger números suficientemente grandes de n=--D y truncar el lastre izquierdo de h(n) para n<-D. Uno podrá reemplazar el h(n) verdadero por su aproximación truncada.

$$\tilde{h}(n) = \begin{cases} h(n), para.n \geq -D \\ 0, para.n < -D \end{cases}$$

Esta respuesta truncada será del tipo finito anticausal como en la figura 2.6.2, y puede convertirse en causal con un retraso de D unidades de tiempo, $\tilde{h}_D(n) = \tilde{h}(n-D)$. El error de aproximación podrá hacerse tan pequeño como se quiera al incrementar el valor de D. Para ver esta, supongamos que $\tilde{y}(n)$ sea la salida del sistema aproximado $\tilde{h}(n)$ para una entrada limitada $|x(n)| \leq A$, supongamos también que y(n) sea la salida del sistema original h(n). Es fácilmente demostrable que el error en la entrada está limitado en su parte superior por

$$|y(n) - \tilde{y}(n)| \leq A \sum_{m=-\infty}^{-D-1} |h(m)|$$

para toda n. Al ser una suma parcial, la suma superior es finita y tiende a cero en lo que D se decrementa.

Este tipo de filtros estables para no causales generalmente se implementan en el diseño de filtros inversos. El inverso de un filtro con función de transferencia H(z) tiene función de transferencia:

$$H_{inv}(z) = \frac{1}{H(z)}.$$

Tales filtros inversos se utilizan en varias aplicaciones de ecualización, tales como ecualización por canal, transmisión de datos digitales en donde H(z) puede representar la función de transferencia del canal.

2.7 La transformada z.

La transformada z es una herramienta muy útil para trabajar con sistemas en tiempo discreto. La transformada z de una función x(t) se define por la siguiente ecuación:

$$X(z) = Z[x(t)] = Z[x(kT)] = Z[x(k)] =$$
$$= \sum_{k=0}^{\infty} x(kT)z^{-k} = \sum_{k=0}^{\infty} x(k)z^{-k}$$

En donde t no es negativo. K tiene enteros positivos o cero y T es el periodo de muestreo de la secuencia x(kT) ó x(k).

A esta trasformada z se le denomina transformado z de un solo lado. El símbolo Z denota "la transformado z de". En esta transformado z de un solo lado, suponemos que x(t)=0 para t<0 ó x(kT)=x(k)=0 para k<0. Nótese que es una variable compleja. En ingeniería de control y procesamiento de señales X(z) se expresan frecuentemente como una razón de polinomios en z^{-1}:

$$X(z) = \frac{b_0 z^{-(n-m)} + b_1 z^{-(n-m+1)} + \ldots + b_m z^{-n}}{1 + a_1 z^{-1} + a_2 z^{-2} + \ldots + a_n z^{-n}}$$

en donde z^{-1} se interpreta como el operador unitario de retraso.

La transformada tiene las siguientes propiedades y teoremas:

1. Multiplicación por una constante. $Z[ax(t)] = aX(z)$

2. Linealidad. Sea $x(k) = af(k) + \beta g(k)$ entonces $X(z) = aF(z) + \beta G(z)$.

3. Multiplicación por a^k. Sea X(z) la transformado z de x(k), entonces la transformada z de $a^k x(k)$ está dado por $X(a^{-1}z)$

4. Translación. Si x(t)=0 para t<0 y x(t) tiene transformada X(z) entonces $Z[x(t-nT)] = z^{-n}X(z)$

y otros teoremas que se deducen de éstos.

Capítulo 3.

Filtrado FIR y Convolución.

3.1 Introducción.

Los métodos para el procesamiento digital de señales entran en dos categorías básicas.

- Método de procesamiento por bloques.
- Método de procesamiento por muestras.

En el método de procesamiento por bloques, las datas se recolectan y procesan en bloques. Algunas aplicaciones típicas incluyen: al filtrado FIR de señales de duración finita por convolución, convolución rápida de señales más grandes (las cuales se parten en segmentos más cortas), operaciones de DFT/FFT (Transformada de Fourier Discreta, Transformada de Faurier Rápido), análisis y síntesis de voz y procesamiento de imágenes.

En el método de procesamiento por muestras, la información se procesa una a la vez. Cada muestra es sometida a un algoritmo de procesamiento de señal DSP el cual la transforma en una muestra de salida. Los métodos de procesamiento por muestras se utilizan básicamente en aplicaciones en tiempo real, tales como el filtrado de señales largas, procesamiento digital de efectos de audio, sistemas de control digital y procesamiento adaptativo de señal. Las algoritmos de procesamiento de muestras son esencialmente las relaciones de estado-espacio de los filtros LTI.

En este capítulo se considerará el procesamiento por bloques y el procesamiento por muestras de aplicaciones para filtros FIR. Se discutirán los aspectos computacionales de las ecuaciones de convolución así como su aplicación a filtros FIR y entradas de duración finita.

3.2 Método de procesamiento por bloques.

3.2.1 Convolución.

En muchas aplicaciones prácticas, muestreamos nuestra señal analógica de entrada, de acuerdo con los requisitos del teorema de muestreo. Obtenemos un número finito de muestras, digamos L muestras, que representan un tiempo finito de la señal de entrada. La duración de los datos en segundos será: [9]

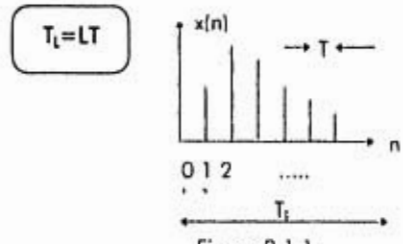


Figura 3.1.1

En donde T es el intervalo de muestreo, y está relacionado con la frecuencia de muestreo con la ecuación $f_s=1/T$. Reciprocamente, podemos resolver el número de intervalos L contenidos en una grabación de duración T_L segundos:

$$L=T_L f_s$$

Las L muestras recolectadas, digamos x(n),n=0,1,2.....,L-1, pueden pensarse como un bloque:

$$x=\left[x_0,x_1,\ldots\ldots,x_{L-1}\right],$$

el cual podrá procesarse posteriormente por un filtro digital. La forma directa y la forma LTI de la convolución

$$y(n)=\sum_m h(m)x(n-m)=\sum_m x(m)h(n-m)$$

describen la ecuación de filtrado de un sistema LTI en general. Una manera alternativa de escribir estas ecuaciones, llamada convolución por tablo, se obtiene al notar que la

[9] De manera más correcta,, $T_L=(L-1)T$, para L grandes esta ecuación es más fácil.

suma de los índices h(m) y x(n-m) es m+(n-m)=n. Entonces, la ecuación anterior puede escribirse de la siguiente forma:

$$y(n)= \sum_{i+j=n} h(i)x(j),$$

que es la suma de todos los productos posibles h(i)x(j) con i+j=n. El rango preciso de la sumatoria con respecto a *m* depende en la naturaleza particular del filtro y de su secuencia de entrada, h(n) y x(n).

3.2.2 Forma directa.

Considérese un filtro FIR causal de orden M con una respuesta al impulso h(n),n=0,1,...,M. Este puede representarse por el bloque

$$\mathbf{h} = [h_0, h_1, \ldots, h_M].$$

Su tamaño es un número mayor que su orden:

$$L_h = M + 1$$

La convolución de la entrada **x** de tamaño L con el filtro de orden **h** de orden M daró como resultado una secuencia de salida y(n). Debemos determinar: (i) el rango de los valores de la salida n y (ii) el rango preciso de sumatoria en m. Para la forma directa tendremos:

$$y(n) = \sum_{m} h(m)x(n-m)$$

El índice de h(m) deberá caer dentro del rango de los índices en la ecuación que $\mathbf{h} = [h_0, h_1, \ldots, h_M]$ o sea, deberá restringirse al intervalo $0 \le m \le M$.

Similarmente, el índice x(n-m) deberá caer en el rango legal de índices, esto es: $0 \le n - m \le L - 1$.

Para determinar el rango de valores de los índices de salida rescribiremos la ecuación de la siguiente forma:

$$m \le n \le L - 1 + m$$

y usaremos la ecuación anterior para extender los límites a:

$$0 \le m \le n \le L-1+m \le L-1+M \text{ ó}$$

$$0 \le n \le L-1+M$$

éste será el ranga de los índices de la secuencia de salida y(n). Es por esto que se representa como un bloque:

$$\mathbf{y} = [y_0, y_1, \ldots\ldots, y_{L-1+M}]$$

de largo

$$L_y = L+M$$

Aunque **y** es más grande que la entrada x por M muestras. Como veremos después, esta propiedad sigue del hecho de que un filtro de orden M tiene memoria M y guarda cada muestra dentro de él M unidades de tiempo. Poniendo $L_x=L$, y $L_h=M+1$ podemos escribir la ecuación anterior de forma más familiar:

$$L_y = L_x + L_h - 1$$

Para cualquier valor del índice n del rango de salida, debemos determinar el rango de adición sobre m en la ecuación de convolución. Para n predeterminados, las inecuaciones deberán estar simultáneamente satisfechas por m. Cambiando el signo de la ecuación $0 \le n-m \le L-1$ obtenemos:

$$-(L-1) \le m-n \le 0$$

y sumando n en ambos lados:

$$n-L+1 \le m \le n$$

Aunque, m deberá satisfacer simultáneamente las ecuaciones:

$$0 \le m \le M$$

$$1-L+1 \le m \le n$$

Podemos concluir que m deberá ser mayor al máximo de los dos lados izquierdos de las inecuaciones y menor que el mínimo de los dos lados derechos de las mismas:

$$\max(0, n-L+1) \le m \le \min(n, M)$$

Debido a esto, para el caso de un filtro FIR de orden M y tamaño de entrada L, la forma directa de convolución es:

$$y(n) = \sum_{m=\max(0,n-L+1)}^{\min(n,M)} h(m)x(n-m)$$

Como ejemplo considérese un filtro de orden 3 y una señal de entrada de 5 muestras. La entrada del filtro y los bloques de salida son:

$$h = [h_0, h_1, h_2, h_3]$$
$$\mathbf{x} = [x_0, x_1, x_2, x_3, x_4]$$
$$\mathbf{y} = h * x = [y_0, y_1, y_2, y_3, y_4, y_5, y_6, y_7]$$

El bloque de salida tiene tamaño $L_y = L+M = 5+3 = 8$ y está indexado como $0 \le n \le 7$. La ecuación de convolución se convierte en:

$$y_n = \sum_{m=\max(0,n-4)}^{\min(n,3)} h_m x_{n-m}, \quad n=0,1,\ldots,7$$

Para n=0,1,2,...,7 la sumatoria de índices m toma los valores:

$$\max(0,0-4) \le m \le \min(0,3) \Rightarrow m = 0$$
$$\max(0,1-4) \le m \le \min(1,3) \Rightarrow m = 0,1$$
$$\max(0,2-4) \le m \le \min(2,3) \Rightarrow m = 0,1,2$$
$$\max(0,3-4) \le m \le \min(3,3) \Rightarrow m = 0,1,2,3$$
$$\max(0,4-4) \le m \le \min(4,3) \Rightarrow m = 0,1,2,3$$
$$\max(0,5-4) \le m \le \min(5,3) \Rightarrow m = 1,2,3$$
$$\max(0,6-4) \le m \le \min(6,3) \Rightarrow m = 2,3$$
$$\max(0,7-4) \le m \le \min(7,3) \Rightarrow m = 3$$

Así por ejemplo, en n=5 la salida y_5 estará dada por

$$y_5 = \sum_{m=1,2,3} h_m x_{5-m} = h_1 x_4 + h_2 x_3 + h_3 x_2.$$

Usando estos valores obtenemos las muestras de salida:

$$y_0 = h_0 x_0$$
$$y_1 = h_0 x_1 + h_1 x_0$$
$$y_2 = h_0 x_2 + h_1 x_1 + h_2 x_0$$
$$y_3 = h_0 x_3 + h_1 x_2 + h_2 x_1 + h_3 x_0$$
$$y_3 = h_0 x_4 + h_1 x_3 + h_2 x_2 + h_3 x_1$$

$$y_2 = h_1x_4 + h_2x_3 + h_3x_2$$

$$y_1 = h_2x_4 + h_3x_3$$

$$y_0 = h_3x_4$$

Existen otras formas de realizar el procesamiento por bloques tales como la forma LTI de convolución (que utiliza las propiedades de linealidad e invariabilidad en el tiempo), la forma matricial, comportamiento transciente y estable, y otras. Su estudio es algo delicado y no es tema de esta tesis.

3.3 Método de procesamiento por muestras.

Los métodos de convolución procesan la señal de entrada bloque por bloque. Discutiremos fórmulas alternativas para los filtros FIR que pueden operar muestra a muestra. Como se ha mencionado antes, este método es conveniente para aplicaciones en tiempo real que requieren del procesamiento continuo de la señal de entrada.

Los algoritmos de procesamiento por muestras se relacionan íntimamente con la realización por bloques de la ecuación de filtrado I/O. Un diagrama de bloque es una mecanización de la ecuación I/O en términos de tres bloques básicos de construcción: sumadores, multiplicadores y retrasos.

En general, un filtro podrá tener diversos diagramas de bloques equivalentes, dependiendo de cómo se organiza la ecuación I/O. Cada realización produce su propio algoritmo de procesamiento de muestras. Algunos tipos de realización de filtros standard son: *canónico, directo* y en *cascada* con sus versiones transpuestas. Únicamente discutiremos la forma directa para la realización de filtros FIR

3.3.1 Retraso Simple.

Como introducción al concepto de algoritmos de procesamiento de muestras, consideraremos el caso del retraso simple. Sea un sistema LTI con una relación de entrada / salida:

$$y(n)=x(n-1)$$

Puede pensarse en esto como un registro que almacena la muestra anterior x(n-1). A cada instante de tiempo n, dos pasos deberán realizarse: (a) el contenido de x(n-1) se alimenta a la salida y (b) la entrada actual x(n) se almacena en el registro, en donde se mantendrá por un instante de muestreo y se convertirá en la salida del próximo instante n+1.

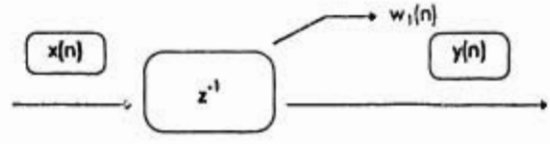


Figura 3.3.1 Retraso Simple.

Podemos pensar que el contenido del registro retrasado en un tiempo n es el estado interno del filtro.

$$w_1(n)=x(n-1) \quad \text{(estado interno en el intervalo de tiempo n)}$$

Aunque la salida sea $y(n)=w_1(n)$. Reemplazando n por n+1, obtendremos el contenido del registro al siguiente instante de tiempo,

$$w_1(n+1)=x(n) \quad \text{(estado interno en el intervalo de tiempo n+1)}$$

Los dos estados de procesamiento (a) y (b) pueden expresarse de la siguiente manera:

$$y(n)=w_1(n)$$

$$w_1(n+1)=x(n).$$

o sea, en el intervalo de tiempo n , el contenido del registro $w_1(n)$ se convierte en la salida, y la entrada x(n) se almacena y se convierte en el nuevo contenido. En el intervalo de tiempo n+1, las dos pasos se repiten:

$$y(n+1)=w_1(n+1)$$

$$w_1(n+2)=x(n+1)$$

El estado interno de w(n+1) está disponible del paso anterior en que fue almacenado. La entrada actual x(n+1) también está disponible y se almacenará para el próximo intervalo de tiempo. Antes de procesar la primera muestra, el retraso se inicializa a cero, esto es, n=0 contiene

$$w_1(0)=0$$

La tabla siguiente muestra los valores de la entrada x(n), el estado interno $w_1(n)$, y la salida y(n) en diversos instantes:

n	x(n)	$w_1(n)$	y(n)
0	x_0	0	0
1	x_1	x_0	x_0
2	x_2	x_1	x_1
3	x_3	x_2	x_2
4	x_4	x_3	x_3
.	.	.	.
.	.	.	.
.	.	.	.

La secuencia de entrada se demora una unidad de tiempo en su totalidad:

$$[x_0,x_1,x_2,x_3,\ldots]\xrightarrow{H}[0,x_0,x_1,x_2,x_3,\ldots]$$

Los dos pasos que siguen a este último diagrama se pueden expresar con el siguiente algoritmo que se aplica repetitivamente a cada muestra.

Para cada muestra de entrada x hacer:

$y := w_1$

$w_1 := x$

Este es el algoritmo muestra a muestra que implementa un solo retraso.

En general, para un retraso de D unidades de tiempo, el contenido de D registros se denotan por $w_i(n), i=1,2,3,4,...,D$. Por conveniencia, la entrada se denota por $w_0(n)$. La salida de cada registro es la versión retrasada de la entrada:

$$w_i(n) = w_{i-1}(n-1), \text{ para } i=1,2,...,D$$

Para el tiempo n, el contenido del registro D sale, $y(n)=w_D(n)$. Después para la preparación del siguiente intervalo de tiempo, el contenido del registro w_{D-1} se convierte en w_D, el contenido de w_{D-2} se convierte en w_{D-1} y así sucesivamente. Finalmente, el contenido de la entrada w_0 se convierte en w_1. Estas actualizaciones pueden ser realizadas al reemplazar n por n+1 y voltear el orden de las ecuaciones. El conjunto completo de ecuaciones I/O que describen D retrasos se convierte en:

$$y(n) = w_D(n)$$

$$w_0(n) = x(n)$$

$$w_i(n+1) = w_{i-1}(n), \quad i=D,D-1,....,2,1$$

El algoritmo de procesamiento será:

Para cada entrada x hacer:

$y := w_D$

$w_0 := x$

Para i=D,D-1,...,1 hacer:

$w_i := w_{i-1}$

o simplemente:

Para cada entrada w_0 hacer:

para cada i=D,D-1,...,1 hacer

$w_i := w_{i-1}$

3.3.2 Filtrado FIR en forma directa

Hemos visto que la ecuación directa de convolución I/O para un filtro FIR de orden M está dada por

$$y(n) = h_0x(n)+h_1x(n-1)+\ldots+h_Mx(n-M)$$

con respuesta al impulso $\mathbf{h} = [h_0, h_1, \ldots, h_M]$. Por ejemplo, un filtro de tercer orden

$$\mathbf{h} = [h_0, h_1, h_2, h_3]$$

tendrá ecuaciones I/O:

$$y(n)=h_0x(n)+h_1x(n-1)+h_2x(n-2)+h_3x(n-3)$$

Para lograr mecanizar esta ecuación, necesitaremos utilizar un sumador para acumular la suma de los productos del lado derecho; necesitaremos multiplicadores para implementar las multiplicaciones del peso de los filtros; y necesitaremos retrasos para implementar los términos de retraso x(n-1), x(n-2),x(n-3).

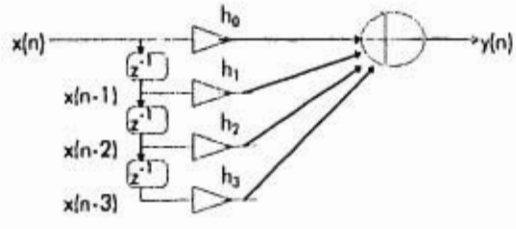


Figura 3.3.2 Forma de realización directa de un filtro de tercer orden.

La figura 3.3.2 muestra la mecanización de la ecuación anterior. Se llama forma directa de realización ya que directamente realiza todos los términos del lado derecho de la misma. Las cuatro entradas al sumador son los cuatro términos del lado derecho de la ecuación, la salida del sumador es el lado izquierdo de la ecuación.

Los tres retrasos son equivalentes al triple retraso y por tanto podremos introducir el mismo número de estados internos $w_1(n)$, $w_2(n)$, $w_3(n)$ para escribir el contenido de los

tres registros. Por esto, definiremos que cada una es una versión retrasada del anterior. Con estas definiciones, podemos reescribir la ecuación a la siguiente forma:

$$y(n)=h_0w_0(n)+h_1w_1(n)+h_2w_2(n)+h_3w_3(n)$$

En la figura 3.3.4 se muestra la realización en este caso. La ventaja de esta ecuación es que todos los términos del lado derecha se refieren al mismo instante en el tiempo. Todas están listas para su procesamiento en el intervalo de tiempo n; esto es $w_0(n)$ será la entrada actual y $w_i(n), i=1,2,3$ son los contenidos actuales de los registros retrasados.

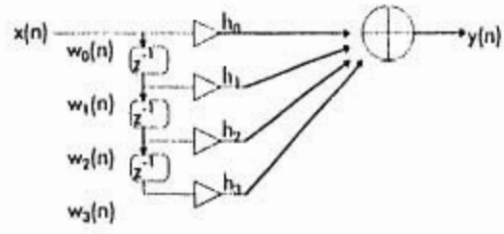


Figura 3.3.3 Forma de realización directa con estados internos.

Una vez que se computa la salida actual, los registros retrasados deben ser actualizados para que contengan los valores necesarios en el instante de tiempo n+1. Las actualizaciones se implementan cambiando de arriba hacia abajo los valores de w: w_2 en w_3, w_1 en w_2 y w_0 en w_1. Aunque la ecuación es equivalente al siguiente sistema:

$w_0(n)=x(n)$

$y(n)=h_0w_0(n)+h_1w_1(n)+h_2w_2(n)+h_3w_3(n)$

$w_3(n+1)=w_2(n)$

$w_2(n+1)=w_1(n)$

$w_1(n+1)=w_0(n)$

Puede mecanizarse por el siguiente algoritmo[10] de procesamiento muestra a muestra.

---

[10] Por razones de simplicidad, usaremos = en vez de :=.

Para cada muestra de entrada x hacer:

$w_0 = x$

$y = h_0 w_0(n) + h_1 w_1(n) + h_2 w_2(n) + h_3 w_3(n)$

$w_3 = w_2$

$w_2 = w_1$

$w_1 = w_0$

Capítulo 4

Función de Transferencia.

4.1 Descripciones equivalentes para filtros digitales.

En este capítulo con la ayuda de la transformada z, desarrollaremos varias métodos matemáticamente equivalentes para describir y caracterizar los filtros FIR e IIR en función de su:

- Función de transferencia H(z),
- Respuesta en frecuencia H(w),
- Realización de su diagrama de bloque y algoritmo de procesamiento por muestra,
- Ecuación diferencial de I/O,
- Patrón polo/cero,
- Respuesta al impulso h(n),
- Ecuación de convolución de I/O,

Lo más importante será la función de transferencia ya que de ella se pueden obtener las demás. La necesidad de tener tantas descripciones es que, cada una provee una perspectiva diferente de la naturaleza del filtro y cada una tiene un propósito diferente. Un usa típico de estas descripciones es, la forma deseada de H(ω). Así, a través de un método de diseño de filtro, se obtiene una función de transferencia H(z) que satisface las especificaciones dadas. A partir de H(z) se puede realizar el diagrama de bloque y su algoritmo de procesamiento muestra por muestra que nos diga cómo operar el filtro diseñado en tiempo real. Para un filtro FIR, se puede obtener la respuesta al impulso h(n) y posteriormente utilizar algún método de convolución para el procesamiento del algún filtro FIR.

4.2 Función de Transferencia.

Se ilustro el papel control de la función de transferencia H(z) de un filtro al mostrarnos cómo puede pasarse entre una descripción y otra.

Dada una función de trosferencia H(z) uno puede obtener (a) la respuesta al impulso h(n), (b) la ecuación diferencial que satisface a la respuesta al impulso, (c) la ecuación I/O que relaciona la salida y(n) con la entrada x(n), (d) el diagrama de bloques del filtro, (e) el algoritmo muestra por muestra, (f) el patrón polo/cero, y (g) la respuesta a la frecuencia H(ω). Dado (e)-(g) como punto de partida, uno puede obtener H(z) y de este, lo restante de (a)-(g).

Como ejemplo, considérese la función de transferencia:

$$H(z)=\frac{5+2z^{-1}}{1-0.8z^{-1}}$$

Para obtener la respuesta al impulso se utiliza una expansión en fracciones parciales para escribirla de la forma:

$$H(z)=\frac{5+2z^{-1}}{1-0.8z^{-1}}=A_0+\frac{A_1}{1-0.8z^{-1}}=-2.5+\frac{7.5}{1-0.8z^{-1}}$$

donde A_0 y A_1 se obtienen por:

$$A_0=H(z)\Big|_{z=0}=\frac{5+2z^{-1}}{1-0.8z^{-1}}\Big|_{z=0}=\frac{5z+2}{z-0.8}\Big|_{z=0}=\frac{2}{-0.8}=-2.5$$

$$A_1=(1-0.8z^{-1})H(z)\Big|_{z=0.8}=(5+2z^{-1})\Big|_{z=0.8}=5+2/0.8=7.5$$

Suponiendo que el filtro es causal, tenemos que:

$$h(n)=-2.5\delta(n)+7.5(0.8)^n u(n)$$

La ecuación en diferencias que satisface a h(n) puede obtenerse de H(z). La aproximación standard es la de eliminar el denominador polinomial de H(z) y transferir el resultado de regreso al dominio del tiempo. Empezando con la ecuación:

$$H(z)=\frac{5+2z^{-1}}{1-0.8z^{-1}}$$

encontramos que :

$$(1-0.8z^{-1})H(z)=5+2z^{-1} \Rightarrow H(z)=0.8z^{-1}H(z)+5+2z^{-1}$$

Al tomar la transformada inversa en ambos lados de la ecuación y utilizando las propiedades de linealidad y de retraso, obtenemos la ecuación diferencial de h(n):

$$h(n)=0.8h(n-1)+5\delta(n)+2\delta(n-1)$$

Se demuestra fácilmente que la ecuación $h(n)=-2.5\delta(n)+7.5(0.8)^n u(n)$ es la solución causal, que significa que es la solución con la condición inicial h(-1)=0. Dada la respuesta al impulso, podemos obtener la ecuación convolucional I/O del filtro:

$$y_n=h_0x_n+h_1x_{n-1}+h_2x_{n-2}+h_3x_{n-3}+\ldots$$
$$=5x_n+7.5((0.8)x_{n-1}+(0.8)^2x_{n-2}+(0.8)^3x_{n-3}+\ldots)$$

Que se puede arreglar como una ecuación en diferencias para y(n), utilizando técnicas en el dominio del tiempo. La ecuación en diferencias puede obtenerse rápidamente utilizando la transformada z con la ayuda del equivalente a la convolución en el dominio z.

$$Y(z)=H(z)X(z)$$

De nuevo, el procedimiento standard será el de eliminar denominadores e ir al dominio del tiempo. Para nuestro caso tenemos:

$$Y(z)=H(z).X(z)=\frac{5+2z^{-1}}{1-0.8z^{-1}}X(z) \Rightarrow (1-0.8z^{-1})Y(z)=(5+2z^{-1}).X(z)$$

Que se podrá escribir:

$$Y(z)-0.8z^{-1}Y(z)=5X(z)+2z^{-1}.X(z)$$

Tomando la transformada inversa en ambos lados, tenemos:

$$y(n)-0.8y(n-1)=5x(n)+2x(n-1)$$

Por lo tanto la ecuación I/O será:

$$y(n)=0.8y(n-1)+5x(n)+2x(n-1)$$

Nótese que la ecuación $h(n)=0.8h(n-1)+5\delta(n)+2\delta(n-1)$ es un caso especial de esta función con $x(n)=\delta(n)$ y $y(n)=h(n)$. Si la ecuación en diferencias, anterior es el punto de partida, podemos obtener H(z) al hacer en sentido inverso los pasos arriba mencionados, esto se hace tomando transformada z en ambos lados de la ecuación

$$Y(z)=0.8z^{-1}Y(z)+5X(z)+2z^{-1}X(z) \Rightarrow$$
$$(1-0.8z^{-1})Y(z)=(5+2z^{-1})X(z)$$

y resolviendo la relación

$$H(z)=\frac{Y(z)}{X(z)}=\frac{5+2z^{-1}}{1-0.8z^{-1}}$$

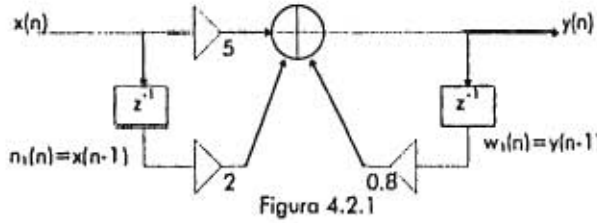


Figura 4.2.1

Una vez que se determina la ecuación en diferencias, se puede mecanizar en un diagrama de bloque. Para nuestro ejemplo la figura 4.2.1 nos muestra la forma directa de realización.

Para el caso de un filtro FIR, el algoritmo de procesamiento por muestras puede obtenerse asignando variables de estado interno a todos los retrasos presentes en el diagrama de bloque. Podemos definir

$$v(n)=x(n-1) \Rightarrow v(n+1)=x(n)$$

en donde $v_1(n)$ es el contenido del retraso x en el tiempo n. Similarmente, definimos

$$w_1(n)=y(n-1) \Rightarrow w_1(n+1)=y(n)$$

de tal forma que $w_1(n)$ es el contenido del retraso de y en el tiempo n. Con estas definiciones, podemos reemplazar la ecuación 6.2.4 por el sistema de ecuaciones:

cómputo de salida	$y(n)=0.8w_1+5x+2v_1$
estados de actualización	$v_1(n+1)=x(n)$
	$w_1(n+1)=y(n)$

Que se puede escribir como el algoritmo repetitivo de procesamiento de muestras:

Para cada entrada x hacer:

$y = 0.8w_1 + 5x + 2v_1$ (forma directa)

$v_1 = x$

$w_1 = y$

La respuesta a la frecuencia de este filtro en particular se obtiene al reemplazar z por $e^{j\omega}$ en H(z). Esta sustitución es válida, ya que el filtro es estable y su región de convergencia $|z| > 0.8$ contiene al círculo unitario. Encontramos:

$$H(z) = \frac{5(1 - 0.4z^{-1})}{1 - 0.8z^{-1}} \Rightarrow H(\omega) = \frac{5(1 + 0.4e^{-j\omega})}{1 - 0.8e^{-j\omega}}$$

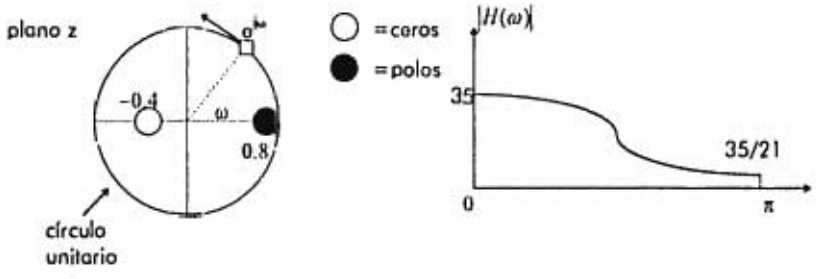
usando la identidad

$$\left|1 - ae^{j\omega}\right| = \sqrt{1 - 2a\cos\omega + a^2}$$

que es válida para cualquier valor real de a, obtenemos la siguiente expresión para la respuesta en magnitud:

$$|H(\omega)| = \frac{5\sqrt{1 + .08\cos\omega + 0.16}}{\sqrt{1 - 1.6\cos\omega + 0.64}}$$

Esta cantidad se gráfica con la ayuda del patrón geométrico polo/cero. El filtro tiene un cero en z=-0.4 y un polo en z=0.8. La siguiente figura muestra la localidad polo/cero relativas al círculo unitario.



Patrón polo/cero y respuesta en magnitud.

Un esquema rápido de la respuesta en magnitud $|H(\omega)|$ se obtiene al permitir el punto $e^{j\omega}$ ubicarse en la perímetro del círculo unitario y con esto graficar sus picos cuando pasan cerca de los polos y sus valles cuando pasa cerca de los ceros.

El punto que se mueve $e^{j\omega}$ está más próximo al polo z=0.8 cuando w=0 y par lo tanto deberá existir un pico ahí. Similarmente en w=π deberá existir un valle, ya que $e^{j\omega}$ se encuentra más cercano al cero z=-0.4. En particular, haciendo que z=1 ó w=0 y z=-1 ó w=π, podremos calcular los valores de la respuesta en frecuencia en los límites del intervalo de Nyquist:

$$H(\omega)_{\omega=0} = H(z)_{z=1} = \frac{5+2}{1-0.8} = 35$$

$$H(\omega)_{\omega=\pi} = H(z)_{z=-1} = \frac{5+2}{1+0.8} = \frac{35}{21}$$

Este filtro actúa como un filtro pasabajas debido a que enfatiza frecuencias bajas y atenúa frecuencias altas. La frecuencia más alta es atenuada por un factor de 21 relativa a la más baja:

$$\frac{|H(\pi)|}{|H(0)|} = \frac{1}{21}$$

o en decibeles:

$$20\log_{10}\left|\frac{H(\pi)}{H(0)}\right| = 20\log_{10}\frac{1}{21} = -26.4dB$$

La realización del diagrama de bloques de una función de transferencia no es único. Funciones de transferencia diferentes pero matemáticamente equivalentes pueden llevarnos a ecuaciones en diferencias, diferentes que se implementan por medio de algoritmos de procesamiento de muestras y diagramas de bloques diferentes.

En general, la función de transferencia de un filtro IIR está dada por la razón de dos polinomios de grado L y M:

$$\mathbf{H}(z) = \frac{N(z)}{D(z)} = \frac{b_0 + b_1 z^{-1} + b_2 z^{-2} + \ldots + b_L z^{-L}}{1 + a_1 z^{-1} + a_2 z^{-2} + \ldots + a_M z^{-M}} \qquad \text{IIR}$$

Nótese que por convención, el coeficiente 0 del polinomio del denominador es $a_0 = 1$. El filtro H(z) tendrá L ceros y M polos. Suponiendo que los coeficientes del numerador y denominador se evalúan con números reales y que cualquiera de los ceros y polos son complejos, entonces deberán venir en pares conjugados.

Para determinar la respuesta al impulso h(n) de tal filtro, debemos usar la transformada inversa z. La colocación relativa de los polos en el plano z dividirá el plano en regiones encimadas que podrán tomarse como la región de convergencia de h(n).

En particular, para obtener una respuesta al impulso estable, debemos tomar la región de convergencia que contenga al círculo unitario. Recuérdese que para que la estabilidad de h(n) sea también causal, todas los polos de H(z) deberán caer estrictamente dentro del círculo unitario. Así la región de convergencia al invertir H(z) deberá ser la del exterior del círculo unitario.

La forma más simple para obtener una respuesta de la función de transferencia es hacer:

$$\mathbf{Y}(z) = H(z).X(z) = \frac{b_0 + b_1 z^{-1} + b_2 z^{-2} + \ldots + b_L z^{-L}}{1 + a_1 z^{-1} + a_2 z^{-2} + \ldots + a_M z^{-M}} X(z)$$

multiplicando por el denominador:

$$(y_n + a_1 z^{-1} + \ldots + a_M z^{-M}) Y(z) = (b_0 + b_1 z^{-1} + \ldots + b_L z^{-L}) X(z)$$

y finalmente, transformando al dominio del tiempo:

$$y_n + a_1 y_{n-1} + \ldots + a_M y_{n-M} = b_0 x_n + b_1 x_{n-1} + \ldots + b_L x_{n-L}$$

Nótese que si los coeficientes del denominador son cero, si, $a_i=0, i=1,2,3,\ldots,M$, el polinomio del denominador es trivial D(z)=1 y H(z) se convierte en el polinomio del numerador H(z)=N(z), o lo que es lo mismo, un filtro FIR:

$$H(z) = N(z) = b_0 + b_1 z^{-1} + b_2 z^{-2} + \ldots + b_L z^{-L} \qquad \text{FIR}$$

En este caso, la ecuación diferencial se convierte en la ecuación convolucional de I/O para un filtro FIR:

$$y_n = b_0 x_n + b_1 x_{n-1} + \ldots + b_L x_{n-L} \qquad \text{Ecuación I/O para un filtro FIR.}$$

4.3 Diseño Polo/Cero.[11]

4.3.1 Filtros de primer orden.

El empleo de polos/cero puede ser utilizado para el diseño de filtros simples, tales como atenuadores de primer orden, filtros de corte o cajas de resonancia. Para ilustrar esta técnica, diseñaremos la función de transferencia

$$H(z) = \frac{5 + 2z^{-1}}{1 - 0.8z^{-1}} = \frac{5(1 + 0.4z^{-1})}{1 - 0.8z^{-1}}$$

discutida anteriormente. Empezaremos con la función de transferencia .

$$H(z) = \frac{G(1 + bz^{-1})}{1 - az^{-1}}$$

en donde ambos a y b son positivas y menores a la unidad. El factor G es arbitrario. El patrón polo/cero se muestra en la figura 4.3.1.

---

[11] Los dos criterios de diseño que utilizamos no son los únicos posibles.

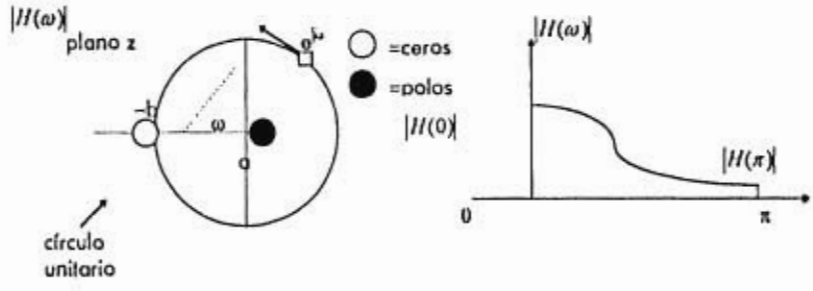


Figura 4.3.1 Patrón de respuesta a la frecuencia polo/cero.

El cero del filtro en z=-b cae en la mitad izquierda (la parte de la frecuencia alta) del círculo unitario, y el polo del filtro en z=a cae en la mitad derecha (la parte de las frecuencias bajas). Debido a esto, el polo enfatiza las frecuencias bajas y el cero atenúa las frecuencias altas; en otras palabras el filtro actúa como un filtro pasabajas.

Las frecuencias mayores y menores se dan en w=0,π, se encuentran al poner z= ±1 en la ecuación $H(z)=\frac{G(1+bz^{-1})}{1-az^{-1}}$:

$$H(0)=\frac{G(1+b)}{1-a}, H(\pi)=\frac{G(1-b)}{1+a}$$

Así, la atenuación de las frecuencias altas relativas a las frecuencias bajas es de:

$$\frac{H(\pi)}{H(0)}=\frac{(1-b)(1-a)}{(1+b)(1+a)}$$

Para determinar los dos parámetros desconocidos a y b se necesitan dos ecuaciones de diseño. Una de estas ecuaciones puede ser la anterior. Si a se conoce, entonces para un nivel de atenuación H(π)/H(0), podemos resolver para b.

Para determinar a, debemos forzar la velocidad de la respuesta del filtro, esto quiere decir que podremos especificar la constante de tiempo efectivo n_{eH}=20 muestras de tiempo y tomando ε=0.01, resolvemos para a:

$$a = \varepsilon^{\frac{-1}{N_p}} = (0.01)^{1/20} \approx 0.8$$

Con este valor de a, con el requisito que H(π)/H(0)= 1/21 la ecuación es

$$\frac{(1-b)(1-0.8)}{(1+b)(1+0.8)} = \frac{1}{21} \Rightarrow b = 0.4$$

que es la ecuación del filtro deseada, con una ganancia de G:

$$H(z) = \frac{G(1+0.4z^{-1})}{1-0.8z^{-1}}$$

Debido a que el parámetro b está restringido en el intervalo $0 \le b \le 1$, podemos observar en los dos diseños extremos, o sea b=0 y b=1. Tomando b=0 en la ecuación $H(z) = \frac{G(1+bz^{-1})}{1-az^{-1}}$ y $\frac{H(\pi)}{H(0)} = \frac{(1-b)(1-a)}{(1+b)(1+a)}$, tenemos que:

$$H(z) = \frac{G}{1-0.8z^{-1}}, \frac{H(\pi)}{H(0)} = \frac{1}{9}$$

y poniendo b= 1,

$$H(z) = \frac{G(1+z^{-1})}{1-0.8z^{-1}}, \frac{H(\pi)}{H(0)} = 0$$

que corresponde a H(π)=0.

4.4. Realización general de filtros digitales.

$$G(z) = \frac{Y(z)}{X(z)} = \frac{b_0 + b_1 z^{-1} + b_2 z^{-2} + \ldots + b_m z^{-m}}{1 + a_1 z^{-1} + a_2 z^{-2} + \ldots + a_n z^{-n}} \quad \text{para } n \ge m$$

Ecuación 4.4.1

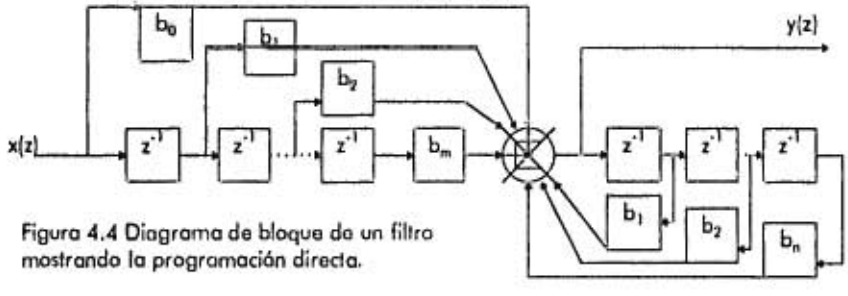


Figura 4.4 Diagrama de bloque de un filtro mostrando la programación directa.

Considérese el filtro digital dado por la ecuación 4.4.1. Nótese que la función de transferencia tiene n polos y m ceros. La figura 4.4 muestra un diagrama de bloque de este filtro. Puede observarse fácilmente que el diagrama de bloque representa a la ecuación 4.4.1 ya que en el diagrama tenemos que

$$Y(z) = -a_1 z^{-1}Y(z) - a_2 z^{-2}Y(z) - \ldots - a_n z^{-n}Y(z) + b_0 X(z) + b_1 z^{-1}X(z) + \ldots + b_m z^{-m}X(z)$$

que rearreglando esta ecuación nos da la ecuación 4.4.1. El tipo de realización mencionada aquí se llama programación directa. La programación directa significa que realizamos el numerador y denominador de la función de transferencia mediante varios elementos de retraso. El numerador usa m elementos de retraso y el denominador usa n elementos de retraso. De tal forma que el número total de elementos de retraso usada en la programación directa es m+n. El número total de elementos de retraso puede reducirse de n+m a n (si n>m) o a m (si n<m). Al método de programación que utiliza el menor número de elementos de retraso se le llama programación standard.

Capítulo 5.

Diseño de Filtros FIR digitales.

El problema de diseño de filtros es el mismo problema de la construcción de una función de transferencia que cumpla con las especificaciones de respuesta a la frecuencia previamente establecida.

Las entradas de cualquier método de diseño de filtros son las especificaciones deseadas y la salida son los coeficientes del vector de respuesta al impulso $\mathbf{h}=[h_0,h_1,\ldots,h_{n-1}]$ para el caso de filtros FIR, o los coeficientes del numerador y del denominador $\mathbf{b}=[b_0,b_1,\ldots,b_m]$, $\mathbf{a}=[a,a_1,\ldots,a_M]$ para el caso de filtros IIR.

Las ventajas más importantes de los filtros FIR son su respuesta lineal y su estabilidad garantizada, debido a la ausencia de polos. Su desventaja potencial es que el requerimiento de filtros muy agudos nos llevará a tamaños de filtro muy grandes (N), dando como resultado un aumento en el costo computacional.

La gran ventaja de los filtros IIR es su bajo costo computacional y su eficiente implementación para obtener filtros específicos. Su más importante desventaja es el potencial que estos filtros tienen para sufrir inestabilidades. Estas inestabilidades son introducidas cuando la cuantización de los coeficientes empujan a los polos fuera del círculo unitario.

5.1 Método de las ventanas.

5.1.1 Filtros ideales.

El método de las ventanas es uno de los métodos más simples de diseño de filtros tipo FIR. Es conveniente para diseñar filtros con respuesta a la frecuencia simple, tales como los filtros pasabajas ideales. Algunas formas típicas que pueden ser diseñadas se muestran en la figura 5.1.1.

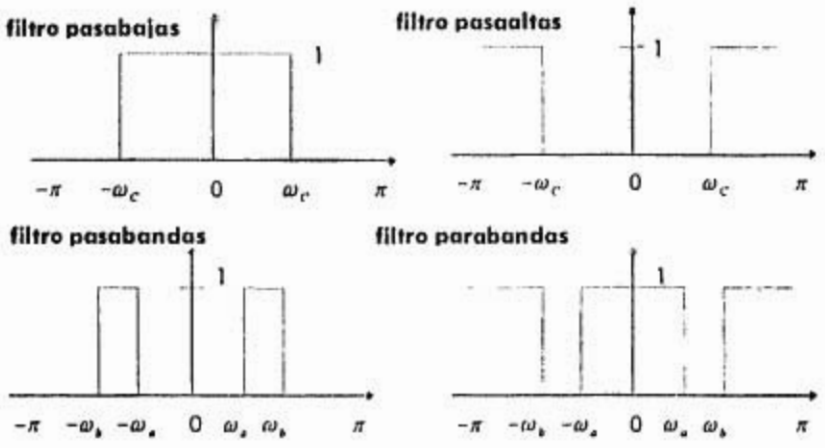


Figura 5.1.1 Filtros ideales

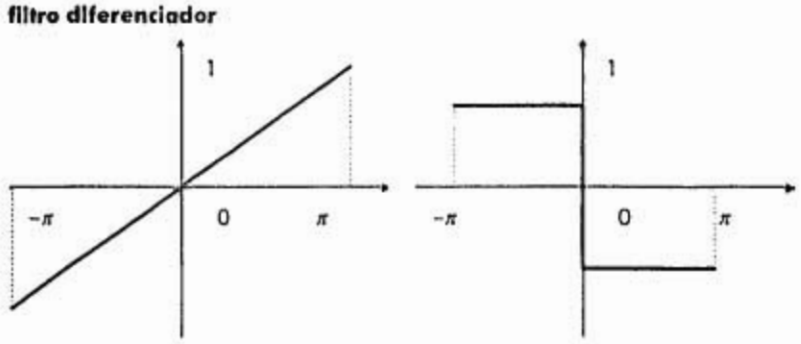


Figura 5.1.2 Filtro diferenciador (ideal) y filtro de Hilbert (ideal).

Una respuesta a la frecuencia ideal dada, digamos $D(\omega)$, siendo periódica en ω con periodo 2π, sólo necesita estar especificada sobre un intervalo de Nyquist $-\pi \leq \omega \leq \pi$. La respuesta al impulso correspondiente digamos $d(k)$, se relaciona con $D(\omega)$ por su Transformada de Fourier en tiempo discreto y su inversa:

$$D(\omega) = \sum_{k=-\infty}^{\infty} d(k) e^{-j\omega k} \Leftrightarrow d(k) = \int_{-\pi}^{\pi} D(\omega) e^{j\omega k} \frac{d\omega}{2\pi}$$

Ecuación 5.1.1

En general, la respuesta al impulso será de doble lado e infinita. Para muchas tipos de filtro, la integración ω puede hacerse en forma cerrada. Por ejemplo, para un filtro pasabajas, la cantidad D(ω) está definida dentro del intervalo de Nyquist como:

$$D(\omega) = \begin{cases} 1, si - \omega_c \le \omega \le \omega_c \\ 2, si - \pi \le \omega \le -\omega_c, o \omega_c \le \omega \le \pi \end{cases}$$

Así, la ecuación 5.1.1 se reduce a:

$$d(k) = \int_{-\pi}^{\pi} D(\omega) e^{j\omega k} \frac{d\omega}{2\pi} = \int_{-\omega_c}^{\omega_c} 1 \cdot e^{j\omega k} \frac{d\omega}{2\pi}$$

$$= \left[\frac{e^{j\omega k}}{2\pi j k} \right]_{-\omega_c}^{\omega_c} = \frac{e^{j\omega_c k} - e^{-j\omega_c k}}{2\pi j k}$$

que puede escribirse también como:

$$\text{(filtro pasabajas)} \quad d(k) = \frac{\sin(\omega_c k)}{\pi k}, \quad -\infty < k < \infty$$

Por motivos computacionales, el caso de k=0 deberá manejarse separadamente. Tomando el límite cuando k tiende a 0 encontramos que :

$$d(0) = \frac{\omega_c}{\pi}$$

Similarmente, encontramos para el filtro pasaaltas, pasabandas y parabandas definidos de $-\infty < k < \infty$

(filtro pasaaltas) $\quad d(k) = \delta(k) - \frac{sen(\omega_c k)}{\pi k}$

(filtro pasabandas) $\quad d(k) = \frac{sen(\omega_b k) - sen(\omega_a k)}{\pi k}$

(filtro parabandas) $\quad d(k) = \delta(k) - \frac{sen(\omega_b k) - sen(\omega_a k)}{\pi k}$

Nótese que para las mismos valores de la frecuencia de corte $\omega_c, \omega_a, \omega_b$, las filtros pasabajas, pasaaltas, pasabandas y parabandas son complementarios. Como consecuencia de la complementariedad, la suma de su respuesta al impulso unitario $\delta(k)$ y su respuesta a la frecuencia es la unidad.

$$d_{LP}(k)+d_{HP}(k)=\delta(k) \quad \Leftrightarrow \quad D_{LP}(\omega)+D_{HP}(\omega)=1$$

$$d_{BP}(k)+d_{BS}(k)=\delta(k) \quad \Leftrightarrow \quad D_{BP}(\omega)+D_{BS}(\omega)=1$$

Como podemos ver, estas propiedades complementarias pueden ser explotadas para simplificar la implementación de sistemas de ecualización y de separación de bandas (cross-over).

El filtro diferenciador ideal de la figura 5.1.2 tiene frecuencia de respuesta $D(\omega)=j\omega$, definida sobre el intervalo de Nyquist. La respuesta del filtro ideal puede expresarse en forma compacta como: $D(\omega)=j^*signo(\omega)$, en donde el signo (ω) es el signo de la función igual a ± 1 dependiendo del signo algebraico de su argumento. Las ω integraciones en la ecuación 5.1.1 nos dan las respuestas al impulso:

(diferenciador) $$d(k)=\frac{\cos(\pi k)}{k}-\frac{sen(\pi k)}{\pi k^2}$$

(transformada de Hilbert) $$d(k)=\frac{1-\cos(\pi k)}{\pi k}$$

Ambos filtros tienen $d(0)=0$, como se puede verificar al tomar cuidadosamente el límite cuando $k\rightarrow 0$. Ambas respuestas al impulso $d(k)$ tienen un valor real y son funciones impares de k (antisimétricas). En contraste, los filtros de la figura 5.1.5 tienen respuestas al impulso real y son funcianes pares en k. Nos referiremos a estas dos clases de filtros como los simétricos y las antisimétricos.

5.1.2 Ventana rectangular.

El método de ventana consiste en truncar la función de ambos lados $d(k)$ a un tamaño finito. Por ejemplo, podemos quedarnos únicamente con las coeficientes:

$$d(k) = \int_{-\pi}^{\pi} D(\omega) e^{j\omega k} \frac{d\omega}{2\pi}, \quad -M \le k \le M$$

Ecuación 5.1.2

Debido a que los coeficientes se toman de igual forma para k positivas y negativas, el número total de coeficientes será impar, esto es, N=2M+1 (valores pares de N son posibles también) El vector de tamaño N es la respuesta al impulso del filtro FIR que aproxima la respuesta infinita ideal:

$$\mathbf{d}=[d_{-M},....,d_{-2},d_{-1},d_0,d_1,d_2,.....,d_M]$$

k=0 se encuentra a la mitad de este vector. Para hacer al filtro causal ponemos el origen al lado izquierdo del vector:

$$\mathbf{h}=\mathbf{d}=[h_0,...,h_{M-2},h_{M-1},h_M,h_{M+1},h_{M+2},...,h_{2M}]$$

En donde definimos $h_0=d_{-M}, h_1=d_{-M+1},....,h_{2M}=d_M$. Así, los vectores **d** y **h** son iguales con la diferencia de su origen. La definición de **h** puede pensarse como un retraso en el tiempo de la secuencia d[k], $-M \le k \le M$, M unidades para convertirla en causal:

$$h(n)=d(n-M), \quad n=0,1,2,.....,N-1$$

Ecuación 5.1.3

El ventaneo y el retraso se muestran en la figura 5.1.3. Resumiendo, los pasos del método de una ventana rectangular son:

1.- Tomar un tamaño impar N=2M+1 y hacer M=(N-1)/2

2.- Calcular los N coeficientes d[k] de la ecuación 5.1.2

3.- Hacerlos causales mediante en retraso h[n]=d[n-M], n=0,1,2,.....,N-1

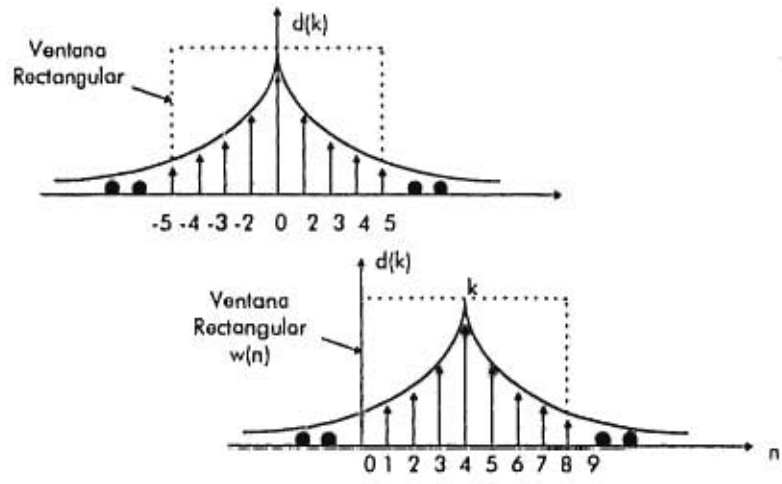


Figura 5.1.3 . Respuesta al impulso de la ventana, con N=9, M=4.

En el plano de la frecuencia, la apraximación D(ω) es equivalente a truncar la expansión de la serie de Fourier completa a una suma finita:

$$\hat{D}(w) = \sum_{k=-M}^{M} d(k) e^{-jwk}$$

Reemplazando $z=e^{jw}$, podemos escribir esta ecuación como una transformada z de doble lado:

$$\hat{D}(z) = \sum_{k=-M}^{M} d(k) z^{-k}$$

El largo total de N obtenido por la ecuación 5.1.1 tiene la función de transferencia:

$$H(z) = z^{-M}\hat{D}(\omega) = e^{-j\omega M} \sum_{k=-M}^{M} d(k) e^{-j\omega k}$$

Ecuación 5.1.3

La propiedad de linealidad en la fase del diseño por ventanas es una consecuencia de la ecuación 5.1.3. $\hat{D}$ tiene las mismas propiedades de simetría/antisimetría que D(ω).

Surge una pregunta. ¿Qué tan buena es la apraximación de $\hat{D}(\omega)$=D(ω)?

Intuitivamente uno pensaría que entre $\hat{D}(\omega)$ es mayor, $\hat{D}(\omega)$ se aproximará mayarmente a D(ω). Esto es verdadera para cualquier punto de continuidad de D(ω), pera na es cierta en las puntas de discantinuidad tales cama las ejes de transición, de pasa o corte de bandas. Alrededar de estas ejes una se tapa con el fenómeno de Gibbs de las series de Faurier, que acasiona que la apraximación sea mala sin impartar el tamaño de N.

En el anexa B se muestran das filtras pasabajas de diferente tamaño de N en dande se alcanza a ver la magnitud de la respuesta a diferentes frecuencias.

La respuesta en magnitud se encuentra al evaluar:

$$H(\omega) = \sum_{n=0}^{N-1} h(n) e^{-j\omega n}$$

La respuesta al impulsa h(n) de tamaña n definida en la ecuación 5.1.3 puede pensarse como una secuencia de la ventana rectangular de dable lada definida por:

$$h(n)=w(n)d(n-M), \quad -\infty < n < \infty$$

en dande w(n) es una ventana rectangular de lamaña N. En el plana de la frecuencia, esto se traduce a la canvolución de las das espectros carrespandientes:

$$H(\omega) = \int_{-\pi}^{\pi} W(\omega-\omega') e^{-j\omega' M} D(\omega') \frac{d\omega'}{2\pi}$$

donde $e^{-j\omega' M}$ surgen del retrasa en d(n-M).

El espectro W(ω) de la ventana rectangular está dada, cuanda L=N. De esta manera, el filtra diseñada H(ω) será una versión apraximada de la farma deseada D(ω). En particular, para el casa del filtra ideal pasabajas, debido a que D(ω') na es cero ni una en el subintervalo $-\omega_c \le \omega' \le \omega_c$, la integral de la canvolución de la frecuencia será:

$$H(\omega) = \int_{-\omega_c}^{\omega_c} W(\omega-\omega') e^{-j\omega' M} \frac{d\omega'}{2\pi}$$

Las arrugas en la respuesta a la frecuencia H(ω) surgen de las arrugas integradas del espectro integrado de ventana rectangular W(ω). En tanto que N se incrementa, notamos tres efectos:

1.- Para ω's que caen en las intervalos de continuidad, las arrugas decrecen en tanto que N se incrementa, resultando una mejor aproximación a D(ω). Tenemos que para una ω tenemos $\hat{D}(\omega) \to D(\omega)$ cuando $N \to \infty$.

2.- El ancho de transición se decrementa cuando incrementamos N. Nótese también que para toda N, la respuesta al impulso H(ω) siempre será de 0.5 en la frecuencia de corte $\omega=\omega_c$. Esta es un propiedad standard en las series de Fourier.

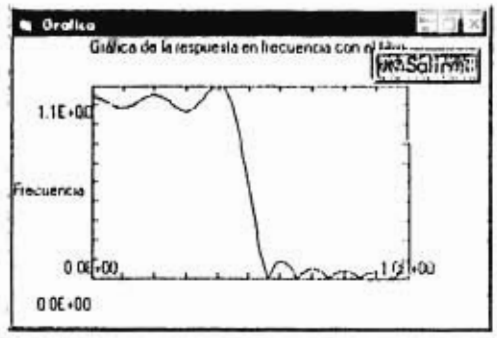
3.- Las arrugas más grandes se encuentran cerca de la banda de corte, en la discontinuidad (en ambos lados), y no se aminoran con N mayores. Su tamaño permanece constante independientemente de N, alrededor de 8.9%. Eventualmente cuando $N \to \infty$ estas arrugas se compactan en una discontinuidad en $\omega=\omega_c$. Esta es conocido como el fenómeno de Gibbs.

5.1.3 Ventana Hamming.

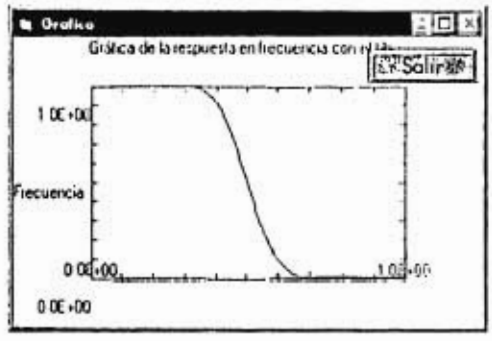
Para eliminar el 8.9% y las arrugas en las bandas de corte, podemos reemplazar la ventana rectangular w(n) por una ventana no rectangular, que gradualmente se aproxime a sus puntos de corte, reduciendo así las arrugas. Existen varios tipos de ventanas, la ventana Hamming es una elección muy común entre ellas.

$$w(n) = 0.54 - 0.46 * \cos\left(\frac{2\pi n}{N-1}\right), \quad n=0,1,2,\ldots,N-1$$

Como ejemplo, considérese el diseño de un filtro pasabajas con N=81, con frecuencia de corte $\omega_c=0.3\pi$. En la figura 5.1.1 siguiente se muestra la ventana rectangular y la ventana Hamming. Nótese como la ventana Hamming se aproxima a cero gradualmente.



Ventana Rectangular. N=21



Ventana Hamming. N=21

Figura 5.1.4 Respuesta en magnitud de la ventana Rectangular y Hamming.

La banda de paso/corte de la ventana rectangular se elimina virtualmente en el diseño de la ventana Hamming. En realidad existen arrugas muy pequeñas con un tamaño máximo de 0.2% pero no son visibles en la escala de la figura 5.1.4. El precio por eliminar las arrugas, una banda de transición más amplia.

5.1.4 Ventana Kaiser.

El diseño de ventanas rectangular y Hamming son muy sencillos, pero no proveen un buen control sobre los especificaciones del filtro. Con estas ventanas, la amplitud de la variación en las arrugas es de 8.9% y 0.2% y no puede cambiarse a un valor menor si así se desea.

Un conjunto de especificaciones flexibles dadas se muestra en la figura 5.1.5 en las que el diseñador puede especificar arbitrariamente la amplitud de la variación de las arrugas δpass, δstop, así como el ancho de la transición Δf.

Las frecuencias de paso/corte {fpass,fstop} se relacionan con la frecuencia ideal de corte fc y su periodo de transición con Δf por:

$$f_c = \frac{1}{2}(f_{pass} + f_{stop}), \Delta f = f_{stop} - f_{pass}$$

Aunque fc se escoge para que caiga exactamente a la mitad de {fpass,fstop}, la ecuación anterior puede expresarse de la siguiente manera:

$$f_{pass} = f_c - \frac{1}{2}\Delta f, f_{stop} = f_c + \frac{1}{2}\Delta f$$

La versión normalizada de las frecuencias son las frecuencias digitales:

$$\omega_{pass} = \frac{2\pi f_{pass}}{f_s}, \omega_{stop} = \frac{2\pi f_{stop}}{f_s}, \omega_c = \frac{2\pi f_c}{f_s}, \Delta\omega = \frac{2\pi\Delta f}{f_s}$$

En la práctica, el exceso en la amplitud de las frecuencias se expresa en dB:

$$A_{pass} = 20\log_{10}\left(\frac{1+\delta_{pass}}{1-\delta_{pass}}\right), A_{stop} = -20\log_{10}\delta_{stop}$$

Ecuación 5.1.4

Una versión simplificada de la ecuación pasabandas puede obtenerse al expandir δ_{pass} a primer orden:

$$A_{pass} \approx 17.372\delta_{pass}$$

que es válida para valores pequeños de δ_{pass}.

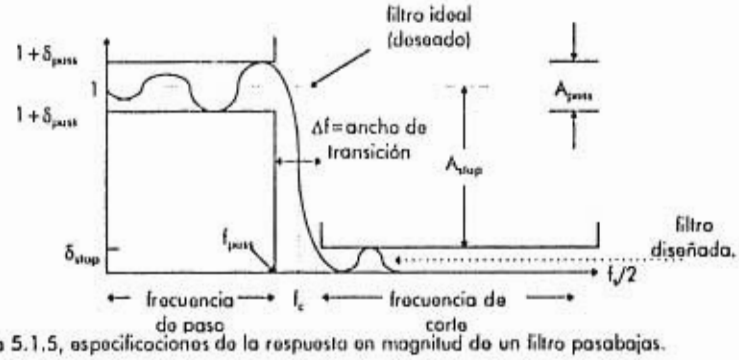


Figura 5.1.5, especificaciones de la respuesta en magnitud de un filtro pasabajas.

Aunque δ_{pass} y δ_{stop} pueden especificarse independientemente una de la otra, una propiedad para todos los diseños de filtros es que el filtro final tenga las mismas arrugas en su porte de paso de frecuencias que en su porte de paro de frecuencias.

$$\delta = \min(\delta_{pass}, \delta_{stop})$$

El filtro diseñado tendrá así arrugas iguales tanto en su periodo de corte como en su periodo de paso de frecuencias. Este valor δ puede expresarse en dB:

$$A = -20\log_{10}(\delta), \text{ ó } \delta = 10^{-A/20}$$

En la práctica el diseño generalmente está basado en las arrugas de corte de frecuencia, esta es debido a que cualquier selección razonable al escoger la atenuación de las frecuencias de corte o de paso resultará (en la mayor parte de los casos) en $\delta_{stop}<\delta_{pass}$, y, $\delta=\delta_{stop}$ y en dB $A=A_{stop}$. Ejemplo. (A_{pass}=0.1 dB y A=60dB). Debido a esto, es útil el considerar a A como la atenuación de corte de frecuencia.

La limitación más importante de la mayor parte de las ventanas es que tienen un valor δ fijo, que depende de la forma particular de la ventana. Tales ventanas limitan los valores de atenuación de paso y de corte $\{A_{pass},A_{stop}\}$ a sólo ciertos valores específicos.

La tabla 5.1.4 muestra las atenuaciones que se pueden alcanzar mediante las ventanas rectangulares y Hamming calculadas con los valores $\delta=\delta_{pass}=\delta_{stop}$=0.089 y

$\delta=\delta_{pass}=\delta_{stop}=0.002$ respectivamente. La tabla muestra también el valor correspondiente de la banda de transición D.

Las únicas ventanas que no sufren de las limitaciones arriba mencianadas son las ventanas: Kaiser, Dolph-Chebyshev y Saramäki. Estas ventanas tienen parámetros de ajuste de forma que permiten a la ventana alcanzar cualquier valor deseado de las arrugas δ o atenuación A.

Ventana	δ	A_{stop}	A_{pass}	D
Rectangular	8.9%	-21dB	1.55	0.92
Hamming	.2%	-54dB	0.03	3.21
Kaiser	variable δ	$-20\log_{10}\delta$	17.372 δ	(A-7.95)/14.36

La ventana Kaiser es única en este tipo, su desempeño es óptima en el sentido de que minimiza la energía del lóbulo externo de la ventana y tiene una implementación más simple. Esta ventana depende de dos parámetros: su tamaño N y el parámetro de su forma α. Suponiendo un tamaño de filtro $n=2M+1$, la ventana se define por $n=0,1,\ldots,N-1$ de la siguiente forma.

$$w(n)=\frac{I_0(\alpha\sqrt{1-(n-M)^2/M^2})}{I_0(\alpha)}$$

En donde $I_0(x)$ es la función de Bessel de clase 1 y orden 0. El numerador de esta ecuación puede rescribirse para que así tengamos una manera de evaluación más conveniente en las evaluaciones numéricas.

$$w(n)=\frac{I_0\left(\alpha\sqrt{n(2M-n)}/M\right)}{I_0(\alpha)},\quad n=0,1,\ldots,N-1$$

Como todas las ventanas, la ventana Kaiser es simétrica en su mitad, $n=M$, en donde tiene un valor $w(m)=1$. En sus límites externos, $n=0$ y $n=N-1$, tiene un valor $1/I_0(0)=1$. Los parámetros de la ventana (N,α) se procesan en términos de las especificaciones del filtro, arrugas δ y su ancho de transición Δf. Las ecuaciones de diseño desarrolladas por Kaiser son:

$$\alpha = \begin{Bmatrix} 0.1102(A-8.7), & si A \geq 50 \\ 0.5842(A-21)^{4} + 0.0786(A-21), & si 21 < a < 50 \\ 0, & A \leq 21 \end{Bmatrix}$$

en donde A son las arrugas en dB. El tamaño de filtro N está inversamente relacionado con su banda de transición:

$$\Delta f = \frac{Dfs}{N-1} \Leftrightarrow N - 1 = \frac{Dfs}{\Delta f}$$

Ecuación 5.1.5

en donde el factor D se computa en términos de A, mediante la ecuación:

$$D = \begin{Bmatrix} \frac{A-7.95}{14.36} & si A > 21 \\ 0.922 & , si A \leq 21 \end{Bmatrix}$$

Los rangos más prácticos de estas fórmulas es A≥ 50dB para la cual se tiene:

$$\alpha = 0.1102(A-8.7), D = \frac{A-7.95}{14.36}$$

Nótese que los parámetros de la ventana {N,α} sólo dependen de las especificaciones {A,Δf} y no de fc. Sin embargo, h(n) si depende de fc.

Los pasos de diseño pueden modificarse fácilmente para diseñar filtros pasaaltas y pasabandas. Para filtros pasaaltas, los papeles de f_{pass} y f_{stop} se intercambian y para su realización, la respuesta al impulso unitario será:

$$h(n) = w(n)d(n-M) = w(n)\left(\delta(n-M) - \frac{sen(\omega_c(n-M))}{\pi(n-M)}\right)$$

El primer término puede simplificarse a w(n)δ(n−M)=w(M)δ(n-M)=δ(n-M) debido a que w(M)=1. A consecuencia de esto, el filtro diseñado será:

$$h(n)=\delta(n-M)-w(n)\frac{sen(w_c(n-M))}{\pi(n-M)}$$

Para el mismo valor de w_c, los filtros pasabajas y pasaaltas son complementarios. La suma de $h_{LP}(n)+h_{HP}(n)=\delta(n-M)$, n=0,1,2,....,N-1. Que se convertirá en el dominio de z en:

$$H_{lp}(z) + H_{hp}(z) = z^{-M}$$

Para filtros pasabandas, las especificaciones deseadas deben darse como en la figura 5.1.6. Existen ahora dos bandas de corte y dos intervalos de transición. El diseño final tendrá intervalos de transición iguales, dados por la ecuación 5.1.5. Por consiguiente, deberemos diseñar el filtro basado en el menor de los dos anchos de transición.

$$\Delta f = \min(\Delta f_a, \Delta f_b)$$

en donde el ancho de transición derecho e izquierdo es:

$$\Delta f_a = f_{pa} - f_{sa}, \Delta f_b = f_{sb} - f_{pb}$$

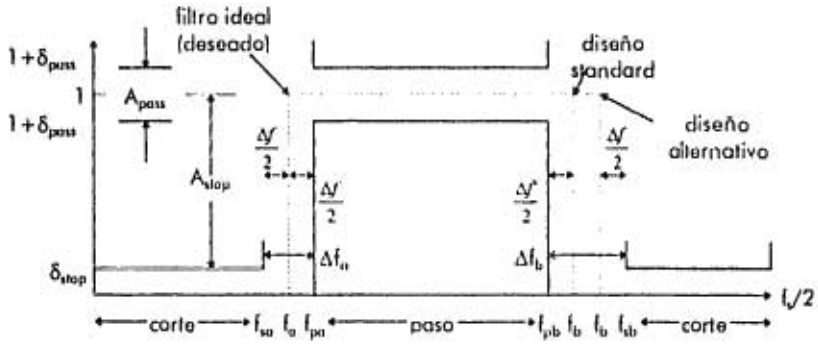


Figura 5.1.6. Especificaciones de un filtro pasabandas

La figura 5.1.6 muestra el caso en donde el lado de transición izquierdo es el menor, así que éste define Δf. La frecuencia de corte ideal fa y fb pueden calcularse al tomarlas como Δf/2 fuera desde la frecuencia de paso o desde la frecuencia de corte. La definición standard se toma con respecto a la frecuencia de paso:

$$f_a = f_{pa} - \frac{1}{2}\Delta f, f_b = f_{pb} + \frac{1}{2}\Delta f$$

Esta elección hace que la frecuencia de paso sea la deseada y la frecuencia de corte sea de mayor espesor. La definición alternativa hace que la frecuencia de corte sea la deseada, y la de paso de mayor espesor.

$$f_a = f_{ca} - \frac{1}{2}\Delta f \,,\; f_b = f_{cb} + \frac{1}{2}\Delta f$$

Una vez que las frecuencias de corte {fa,fb} y los parámetros de la ventana {N,α} se calculan, la respuesta al impulso de un filtro pasabandas puede definirse para n=0,1,...,N-1 como:

$$h(n) = w(n)d(n-M) = w(n)\frac{sen(\omega_b(n-M)) - sen(\omega_a(n-M))}{\pi(n-M)}$$

en donde $h(M) = (\omega_b - \omega_a)/\pi$ y $\omega_a = 2\pi f_a/f_s$, $\omega_b = 2\pi f_b/f_s$.

Capítulo 6.

Construcción de la herramienta.

6.1 Requerimientos.

Para llevar a cabo esta tesis se requirieron los siguientes componentes y software:

1.- Computadora personal con procesador 80486 DX .

2.- Sistema de 32 bits como plataforma.

3.- Visual Basic 4.0 adición profesional a 32 bit para plataforma de 32 bits.

4.- Tarjeta de sonido de 16 bits.

5.- Memoria instalada de 16 MB.

6.- Equipo multimedia.

7.- Controladores OCX para manipulación de sonido y control de tarjeta de sonido.

Para todo control digital de señales es necesario obtener la señal digitalizada; ante esta necesidad y la existencia de una herramienta que realiza este proceso, se ha escogido el uso de la computadora personal como herramienta de desarrollo. El procesamiento digital de señales es un tema de investigación que requiere una gran cantidad de operaciones matemáticas y de control de señal realizables sólo con el uso de computadoras. Otras opciones son los controladores digitales pero resultan ser más complicados y costosos para el desarrollo aunque en ocasiones resulten tener menor tiempo de respuesta y mejores resultados para aplicaciones específicas.

La tecnología actual propone la utilización de plataformas de 16, 32 y hasta 64 bits. Es por esto, y con la finalidad de que el desarrollo de esta tesis no sea obsoleta en poco tiempo, por lo que el sistema utilizado para su realización es de 32 bits. Un tamaño de palabra de 32 bits es actualmente un standard a nivel mundial para computadoras personales.

Para la implementación de los algoritmos computacionales se seleccionó el uso del paquete computacional Microsoft Visual Basic 4.0 en su edición profesional. Esta selección se tomó debido a que este paquete de desarrollo proporciona muchas herramientas computacionales para desarrollo en ambiente Windows. Gracias a que Visual Basic es un lenguaje orientado a objetos y que su programación es

suficientemente intuitiva, (si se tienen nociones de programación), es posible comenzar a programar fácilmente.

La tarjeta de sonido utilizada es una tarjeta *Sound Blaster* 16 SCSI-2. Esta tarjeta tiene la capacidad de muestrear señales a 16 bits y con una frecuencia de muestreado de 44.1 kHz. Para nuestra aplicación estaremos subutilizando esta tarjeta ya que nuestro diseño de filtros se hará analizando la señal a 11,025 Hz.

El equipo multimedia es necesario, ya que toda señal procesada con un filtro debe poder escucharse y con esto comprobar su funcionalidad.

Una ventaja muy importante del Visual Basic (en su edición 4.0 ó mayores) es que posee extensión de control de objetos de enlace encajonado (*object link embeding control extensions*) llamados también OCX. La tecnología OCX es una extensión de la arquitectura abierta del sistema Windows diseñada para correr en cualquier aplicación de Windows que posee objetos de enlace encajonado habilitado. Estos controladores dan la habilidad al diseñador de codificar métodos y propiedades. Esta arquitectura permite al diseñador crear controles que se interrelacionen con cualquier parte del sistema, mediante la comunicación directa con la plataforma.

Los controladores OCX utilizados en esta investigación fueron adquiridos de la compañía TogoSoft Inc. vía correo electrónico a su dirección togosoft@msn.com.

6.2 Realización de herramienta.

Dentro de la investigación se desarrollaron herramientas complementarias para el procesamiento digital de señal:

1.- Lector de señal de audio.

2.- Reproductor de señal.

3.- Graficador de señal

4.- Diseñador de filtro.

5.- Filtrado de señal.

6.- Análisis de Frecuencia, vía la transformada de Fourier.

6.2.1 Lector de señal de audio.

En esta subrutina lo que se pretende es que el usuario tenga un acceso inmediato a la señal acústica que se está escuchando. Esto se realiza mediante el llamado a la función TegoWav1 que llama al OCX correspondiente. Nótese que la subrutina llamada propone una frecuencia de muestreo de 11025 Hz. y un tiempo de muestreo de 65534 muestras El valor de 65536 fue seleccionado debido que 2^N=65536, con N=16.

$$2^{16}=65536$$

y si tenemos 2 canales, tenemos: $2^{17}=131072$ muestras.

El tiempo que tardan estas muestras en reproducirse será:

65536 muestras /11025 (muestras/segundos) =5.944 segundos.

En esta investigación se considera que un tiempo aproximado de 6 segundos es suficiente para fines de diseño de filtros y de pruebas acústicas.

Esta consideración se tomó para simplificar el cálculo de la transformada de Fourier mediante la transformada rápida de Fourier FFT que requiere un número de muestras de 2^n para implementarse en el algoritmo.

Nótese también que la señal se toma de una señal de referencia a.wav. Esta señal de referencia contiene los parámetros necesarios para que la señal que se genere sea grabada en dos canales. Si cambiamos esta señal de referencia, podríamos estar cambiando el número de canales que se obtienen del muestreo.

```
Private Sub Command1_Click()
TegoWav1.Open "C:\Matlab\Mitesis\a.Wav"
TegoWav1.SetSamplingRate 11025
TegoWav1.Record -1, 65534, True
TegoWav1.Delete 65535, -1
TegoWav1.SaveAs "D:\Señales de la tesis\" + Text1 + ".wav"
Numbits = TegoWav1.GetNumBits
 If Numbits = 8 Then
  Label1.Caption = "Archivo de 8 Bits"
 End If
 If Numbits = 16 Then
  Label1.Caption = "Archivo de 16 Bits"
 End If
```

```
numCanales = TegoWav1.GetNumChannels
If numCanales = 1 Then
  Label2.Caption = "Monoaural"
End If
If numCanales = 2 Then
  Label2.Caption = "Stereo"
End If
End Sub
```

6.2.2 Reproductor de señal.

Una vez obtenida la señal es necesario escucharla[12] . El procedimiento para que la señal se escuche utiliza también los controladores OCX. Cabe remarcar que la señal escuchada es la que el usuario seleccione en el controlador de archivos denotado como File1 en el procedimiento.

```
Private Sub Comm3_Click()
archivo = File1
TegoWav1.Open "d:\Señales de la tesis\" + archivo
TegoWav1.Seek 0
TegoWav1.Play -1, -1, True
TegoWav1.Close
End Sub
```

6.2.3 Graficador de señal.

La gráfica de la señal nos ayudará a analizar la estructura de los datos de la misma. Esta visualización nos da una mejor comprensión de la forma que pueden presentar las muestras dentro de un intervalo de tiempo.

El algoritmo utilizado nos muestra con líneas la estructura de la señal en su canal derecho. Para efectos de visualización es suficiente mostrar un solo canal de la señal.

```
Private Sub Command3_Click()
If File1 <> "" Then
archivo = File1
TegoWav1.Open "d:\Señales de la tesis\" + archivo
TegoWav1.Seek 0
tamañoarr = TegoWav1.GetLength * TegoWav1.GetNumChannels
ReDim arreglo(tamañoarr)
TegoWav1.WavFileToArray arreglo(0), tamañoarr
TegoWav1.Close
```

---

[12] para señales de audio.

```
Dim tamañoarreglo As Double
Dim arreglo1() As Double
tamañoarreglo = tamañoarr
L = tamañoarr / 2
ReDim arreglo1(L)
Dim C As Long
  C = 0
  For i = 1 To tamañoarr Step 2
   arreglo1(C) = arreglo(i)
   C = C + 1
  Next i
Dim x() As Double
ReDim x(L)
Title$ = "Gráfico de la señal"
XTITLE$ = ""
YTITLE$ = "2 ^ 8 bits"
FormGrafica.Show
FormGrafica.ForeColor = RGB(0, 0, 255)
FormGrafica.BackColor = RGB(255, 255, 255)
Call SETLINEAR(FormGrafica, -1, tamañoarreglo / 2, -1, 256, Title$, XTITLE$, YTITLE$)
For i = 0 To L - 1
x(i) = i
Next i
 Call LINGRAPH(FormGrafica, x(), arreglo1(), tamañoarreglo / 2, 0)
End If
End Sub
```

El procedimiento anterior llama al módulo **graphs**. Dentro de estos módulos se encuentran los procedimientos para la grófica. Los procedimientos para generación de gráficas se encuentran en el anexo C.

6.2.4. Diseñador de filtros.

Existen una gran cantidad de filtros factibles para el diseño. La herramienta diseña filtros con los siguientes tipos de ventanas:

1.- Rectangular: subtipo pasabajas
subtipo pasaaltas
subtipo pasabandas
subtipo parabandas.

2.- Hamming: mismos subtipos que la ventana rectangular.

3.- Diferenciador

4.- Transformada de Hilbert

5.- Kaiser: mismos subtipos que la ventana rectangular.

El siguiente procedimiento nos muestra el algoritmo de las ventanas arriba expuestas excepto la ventana Kaiser cuyo formato se expone más adelante.

```
Private Sub TegoCheck1_MouseDownOnImage(ByVal Transparent As Boolean, ByVal x As Integer, ByVal
y As Integer, ByVal Button As Integer)
Text3.Visible = True
Dim M As Long
Dim wc As Double
 Select Case Combo1
 Case "rectangular pasabajas"
N = Text1
ReDim h(N)
M = (N - 1) / 2
wc = Text2
For i = 0 To N - 1
  If i = M Then h(i) = wc Else h(i) = (Sin((wc * Pi) * (i - M))) / (Pi * (i - M))
Next i
 Case "rectangular pasaaltas"
N = Text1
ReDim h(N)
M = (N - 1) / 2
wc = Text2
For i = 0 To N - 1
  If i - M = 0 Then delta = 1 Else delta = 0
  If i = M Then h(i) = delta - wc Else h(i) = delta - (Sin((wc * Pi) * (i - M))) / (Pi * (i - M))
Next i
  Case "rectangular pasabandas"
N = Text1
ReDim h(N)
M = (N - 1) / 2
Wb = Text2
Wa = Text4
For i = 0 To N - 1
  If i = M Then h(i) = Wb - Wa Else h(i) = ((Sin((Wb * Pi) * (i - M))) - (Sin((Wa * Pi) * (i - M)))) / (Pi * (i - M))
Next i
 Case "rectangular parabandas"
N = Text1
ReDim h(N)
M = (N - 1) / 2
Wb = Text4
Wa = Text2
For i = 0 To N - 1
  If i - M = 0 Then delta = 1 Else delta = 0
```

```
 If i = M Then h(i) = delta - Wb + Wa Else h(i) = delta - ((Sin((Wb * Pi) * (i - M))) - (Sin((Wa * Pi) * (i -
M)))) / (Pi * (i - M))
Next i
 Case "Diferenciador"
N = Text1
ReDim h(N)
M = (N - 1) / 2
For i = 0 To N - 1
 If i = M Then h(i) = 0 Else h(i) = Cos(Pi * (i - M)) / (i - M) - Sin(Pi * (i - M)) / (Pi * (i - M) ^ 2)
Next i
 Case "Transformada de Hilbert"
N = Text1
ReDim h(N)
M = (N - 1) / 2
For i = 0 To N - 1
 If i = M Then h(i) = 0 Else h(i) = (1 - Cos(Pi * (i - M))) / (Pi * (i - M))
Next i
 Case "Hamming pasabajas"
N = Text1
ReDim h(N)
M = (N - 1) / 2
wc = Text2
For i = 0 To N - 1
 If i = M Then h(i) = wc * (0.54 - 0.46 * Cos(Pi * i / M)) Else h(i) = (0.54 - 0.46 * Cos((2 * Pi * i) / (N - 1
- 1))) * (Sin((wc * Pi) * (i - M))) / (Pi * (i - M))
Next i
Case "Hamming pasaaltas"
N = Text1
ReDim h(N)
M = (N - 1) / 2
wc = Text2
For i = 0 To N - 1
 If i - M = 0 Then delta = 1 Else delta = 0
 If i = M Then h(i) = delta - wc * (0.54 - 0.46 * Cos(Pi * i / M)) Else h(i) = delta - (0.54 - 0.46 * Cos((2 *
Pi * i) / (N - 1 - 1))) * (Sin((wc * Pi) * (i - M))) / (Pi * (i - M))
Next i
Case "Hamming pasabandas"
N = Text1
ReDim h(N)
M = (N - 1) / 2
Wb = Text2
Wa = Text4
For i = 0 To N - 1
  If i = M Then h(i) = (Wb - Wa) * (0.54 - 0.46 * Cos(Pi * i / M)) Else h(i) = (0.54 - 0.46 * Cos((2 * Pi * i)
/ (N - 1))) * ((Sin((Wb * Pi) * (i - M))) - (Sin((Wa * Pi) * (i - M)))) / (Pi * (i - M))
Next i
Case "Hamming parabandas"
N = Text1
ReDim h(N)
M = (N - 1) / 2
Wb = Text4
```

```
Wa = Text2
For i = 0 To N - 1
  If i = M Then delta = 1 Else delta = 0
  If i = M Then h(i) = (delta - Wb + Wa) * (0.54 - 0.46 * Cos(Pi * i / M)) Else h(i) = (delta - ((Sin((Wb * Pi)
* (i - M))) - (Sin((Wa * Pi) * (i - M)))) / (Pi * (i - M))) * (0.54 - 0.46 * Cos(Pi * i / M))
Next I
End Select
Label5.Visible = True
Image1.Visible = True
End Sub
```

Las fórmulas utilizadas en cada uno de los filtros son las fórmulas expuestas en el capítulo 5.

La ventana Kaiser tiene diferente implementación y debido a esto requiere de un Formato en Visual Basic diferente al de los demás filtros. A continuación se presenta el código de diseño de las ventanas tipo Kaiser.

```
Private Sub TegoCheck1_Click()
Dim fpass, fpassb As Double
Dim fstop, fstopb As Double
Dim wc, wa, wb, fc, fa, fb As Double
Dim dpaso As Double
Dim dcorte As Double
Dim delta As Double
Dim Apass As Double
Dim Astop As Double
Dim A As Double
Dim alfa As Double
Dim D As Double
Dim M, F As Integer
Dim incf As Double
Dim n1, n2 As Double
Dim paroimpar As Double
Dim numerador As Double
Dim denominador As Double

Select Case Combo1
Case Is = "pasobajas"
fs = 11025
fpass = Text1
fstop = Text2
fc = 0.5 * (fpass + fstop)
wc = (2 * Pi * fc) / fs
dpaso = Text3
dcorte = Text4
Apass = 20 * (Log((1 + dpaso) / (1 - dpaso))) / Log(10)
Astop = -20 * (Log(dcorte)) / Log(10)
Select Case dpaso - dcorte
```

```
Case ls < 0
delta = dpaso
Case ls >= 0
delta = dcorte
End Select
A = -20 * Log(delta) / Log(10)
Select Case A
 Case ls >= 50
 alfa = 0.1102 * (A - 8.7)
 Case ls > 21 And A < 50
 alfa = 0.5842 * (A - 21) ^ 0.4 + 0.07886 * (A - 21)
 Case ls <= 21
 alfa = 0
End Select
Select Case A
 Case ls > 21
 D = (A - 7.95) / 14.36
 Case ls <= 21
 D = 0.922
End Select
incf = fstop - fposs
n1 = 1 + round(D * fs / incf)
n2 = 1 + D * fs / incf
Select Case (n1 - n2)
 Case ls >= 0
 N = n1
 Case ls < 0
 N = n1 + 1
End Select
paroimpar = N / 2
Select Case paroimpar - Int(paroimpar)
 Case ls = 0.5
 N = N
 Case ls = 0
 N = N + 1
End Select
M = (N - 1) / 2
ReDim h(N)
For i = 0 To N - 1
numerador = BsiO(alfa * Sqr(i * (2 * M - i)) / M)
denominador = BsiO(alfa)
Select Case i
Case ls = M
h(i) = wc / Pi
Case ls <> M
h(i) = (numerador / denominador) * (Sin(wc * (i - M)) / (Pi * (i - M)))
End Select
Next i

Case ls = 'pasoaltas'
fs = 11025
```

```
fpass = Text1
fstop = Text2
fc = 0.5 * (fpass + fstop)
wc = (2 * Pi * fc) / fs
dpaso = Text3
dcorte = Text4
Apass = 20 * (Log((1 + dpaso) / (1 - dpaso))) / Log(10)
Astop = -20 * (Log(dcorte)) / Log(10)
Select Case dpaso - dcorte
Case Is < 0
delta = dpaso
Case Is >= 0
delta = dcorte
End Select
A = -20 * Log(delta) / Log(10)
Select Case A
  Case Is >= 50
  alfa = 0.1102 * (A - 8.7)
  Case Is > 21 And A < 50
  alfa = 0.5842 * (A - 21) ^ 0.4 + 0.07886 * (A - 21)
  Case Is <= 21
  alfa = 0
End Select
Select Case A
  Case Is > 21
  D = (A - 7.95) / 14.36
  Case Is <= 21
  D = 0.922
End Select
incf = fstop - fpass
n1 = 1 + round(D * fs / incf)
n2 = 1 + D * fs / incf
Select Case (n1 - n2)
  Case Is >= 0
  N = n1
  Case Is < 0
  N = n1 + 1
End Select
paroimpar = N / 2
Select Case paraimpar - Int(paroimpar)
  Case Is = 0.5
  N = N
  Case Is = 0
  N = N + 1
End Select
M = (N - 1) / 2
ReDim h(N)
For i = 0 To N - 1
numerador = Bsi0(alfa * Sqr(i * (2 * M - i)) / M)
denominador = Bsi0(alfa)
Select Case i
```

```
Case Is = M
delta = 1
h(i) = delta - wc / Pi
Case Is <> M
delta = 0
h(i) = (numerador / denaminador) * (delta - Sin(wc * (i - M)) / (Pi * (i - M)))
End Select
Next i

Case Is = "pasabandas"
fs = 11025
fpass = Text1
fstop = Text2
fpassb = Text5
fstopb = Text6
If (fstopb - fpassb) < (fstop - fpass) Then incf = fstopb - fpassb Else incf = fstop - fpass
fa = fpass - 0.5 * incf
fb = fpassb + 0.5 * incf
wa = (2 * Pi * fa) / fs
wb = (2 * Pi * fb) / fs
dpaso = Text3
dcorte = Text4
Apass = 20 * (Log((1 + dpaso) / (1 - dpaso))) / Log(10)
Astop = -20 * (Log(dcorte)) / Log(10)
Select Case dpaso - dcorte
Case Is < 0
delta = dpaso
Case Is >= 0
delta = dcorte
End Select
A = -20 * Log(delta) / Log(10)
Select Case A
  Case Is >= 50
  alfa = 0.1102 * (A - 8.7)
  Case Is > 21 And A < 50
  alfa = 0.5842 * (A - 21) ^ 0.4 + 0.07886 * (A - 21)
  Case Is <= 21
  alfa = 0
End Select
Select Case A
  Case Is > 21
  D = (A - 7.95) / 14.36
  Case Is <= 21
  D = 0.922
End Select
n1 = 1 + round(D * fs / incf)
n2 = 1 + D * fs / incf
Select Case (n1 - n2)
  Case Is >= 0
  N = n1
  Case Is < 0
```

```
 N = n1 + 1
End Select
paroimpar = N / 2
Select Case paroimpar - Int(paroimpar)
 Case Is = 0.5
 N = N
 Case Is = 0
 N = N + 1
End Select
M = (N - 1) / 2
ReDim h(N)
For i = 0 To N - 1
numerador = BsiO(alfa * Sqr(i * (2 * M - i)) / M)
denominador = BsiO(alfa)
Select Case i
Case Is = M
h(i) = (wb - wa) / Pi
Case Is <> M
h(i) = (numerador / denominador) * ((Sin(wb * (i - M)) - Sin(wa * (i - M))) / (Pi * (i - M)))
End Select
Next i

Case Is = "parabandas"
fs = 11025
fpass = Text1
fstop = Text2
fpassb = Text5
fstopb = Text6
If (fstopb - fpassb) < (fstop - fpass) Then incf = fstopb - fpassb Else incf = fstop - fpass
fa = fpass - 0.5 * incf
fb = fpassb + 0.5 * incf
wa = (2 * Pi * fa) / fs
wb = (2 * Pi * fb) / fs
dpaso = Text3
dcorte = Text4
Apass = 20 * (Log((1 + dpaso) / (1 - dpaso))) / Log(10)
Astop = -20 * (Log(dcorte)) / Log(10)
Select Case dpaso - dcorte
Case Is < 0
delta = dpaso
Case Is >= 0
delta = dcorte
End Select
A = -20 * Log(delta) / Log(10)
Select Case A
 Case Is >= 50
 alfa = 0.1102 * (A - 8.7)
 Case Is > 21 And A < 50
 alfa = 0.5842 * (A - 21) ^ 0.4 + 0.07886 * (A - 21)
 Case Is <= 21
 alfa = 0
```

```
End Select
Select Case A
  Case Is > 21
  D = (A - 7.95) / 14.36
  Case Is <= 21
  D = 0.922
End Select
n1 = 1 + round(D * fs / incf)
n2 = 1 + D * fs / incf
Select Case (n1 - n2)
  Case Is >= 0
  N = n1
  Case Is < 0
  N = n1 + 1
End Select
paroimpar = N / 2
Select Case paroimpar - Int(paroimpar)
  Case Is = 0.5
  N = N
  Case Is = 0
  N = N + 1
End Select
M = (N - 1) / 2
ReDim h(N)
For i = 0 To N - 1
numerador = Bsi0(alfa * Sqr(i * (2 * M - i)) / M)
denominador = Bsi0(alfa)
Select Case i
Case Is = M
delta = 1
h(i) = delta - (wb - wa) / Pi
Case Is <> M
delta = 0
h(i) = (numerador / denominador) * (delta - ((Sin(wb * (i - M)) - Sin(wa * (i - M))) / (Pi * (i - M))))
End Select
Next i
End Select
End Sub
```

Finalmente, es importante hacer notar que la función Bsi0 es la función modificada de Bessel del clase 1 y de orden cero. Esta función es una solución de la ecuación diferencial:

$$x^2y'' + xy' - (x^2 + n^2)y = 0$$

cuya solución es:

$$y = AI_n(x) + BK_n(x)$$

El cómputo de esta función $I_0(x)$ está dado por la expansión en series de Taylor de:

$$I_0(x) = \sum_{k=0}^{\infty}\left[\frac{(x/2)^k}{k!}\right]^2$$

La ventana Kaiser requiere la evaluación de $I_0(x)$ sobre el rango de argumentos $0 \le x \le \alpha$. El algoritmo utilizado en la evaluación de esta expresión se presenta en el anexo C.

6.2.5 Filtrado de la señal.

Una vez diseñado un filtro podemos utilizarlo mediante el módulo de filtrado de señal. Este módulo está basado en la forma directa de la convolución:

$$y(n) = \sum_{m=\max(0,n-L+1)}^{\min(n,M)} h(m)x(n-m)$$

y su algoritmo es:

```
Private Sub Command1_Click()
Form3.MousePointer = 11
'esta porción asigna los valores de h de un archivo
archivoh = File1
Open "D:\filtros para la tesis de Miguel C\" + archivoh For Random As #1 Len = Len(M)
 Get #1, 1, M
 Mfiltroh = CInt(M)
 ReDim h(Mfiltroh)
 For i = 1 To Mfiltroh
 Get #1, i + 1, h(i - 1)
 Next i
Close #1
 archivodeseñal = File2
 TegoWav1.Open "D:\Señales de la tesis\" + archivodeseñal
tamañoarr = TegoWav1.GetLength * TegoWav1.GetNumChannels
ReDim arreglo(tamañoarr)
TegoWav1.WavFileToArray arreglo(0), tamañoarr
TegoWav1.Close
' este codigo divide lo señal de estero a mano.
L = tamañoarr / 2
ReDim arregl1(L)
ReDim arregl2(L)
Dim C As Long
  C = 0
 For i = 1 To tamañoarr Step 2
  arregl1(C) = arreglo(i)
```

```
  arregl2(C) = arreglo(i - 1)
  C = C + 1
 Next i
' Este es el filtrado de la señal
 Dim arriba As Long
 Dim abajo As Long
 Dim temporal As Double
ReDim arregl1fil(L + Mfiltrah)
ReDim arregl2fil(L + Mfiltrah)
Dim maxima As Integer
Dim minimo As Integer
 maxima = 0
 minimo = 0
For N = 0 To (L + Mfiltrah - 1)
  temporal = 0
  temporal2 = 0
  If N < Mfiltrah Then arriba = N Else arriba = Mfiltrah
  If 0 > N - L + 1 Then abajo = 0 Else abajo = N - L + 1
    For mdesumatoria = abajo To arriba
    temporal = temporal + h(mdesumatoria) * arregl1(N - mdesumatoria)
    temporal2 = temporal + h(mdesumatoria) * arregl2(N - mdesumatoria)
    Next mdesumatoria
  arregl1fil(N) = CInt(temporal)
  If arregl1fil(N) > maxima Then maxima = arregl1fil(N)
  If arregl1fil(N) < minimo Then minimo = arregl1fil(N)
  arregl2fil(N) = CInt(temporal2)
  If arregl2fil(N) > maxima Then maxima = arregl2fil(N)
  If arregl2fil(N) < minima Then minimo = arregl2fil(N)
Next N
' restablece la señal a wav y a un arreglo estereo
C = 0
 For i = 1 To tamañoarr Step 2
   arreglo(i) = ((Abs(minimo) + arregl1fil(C)) / (maximo - minimo)) * 255
   arreglo(i - 1) = ((Abs(minimo) + arregl2fil(C)) / (maximo - minimo)) * 255
   C = C + 1
 Next i
 Form3.MousePointer = 1
 Command3.Visible = True
 Text1.Visible = True
End Sub
```

El filtrado de la señal se realiza en ambos canales. La señal resultante puede grabarse a un archivo tipo wav mediante el siguiente procedimiento:

```
Private Sub Command3_Click()
 archivo = "D:\Señales de la tesis\" + Text1 + ".wav"
 TegoWav1.Open "D:\Señales de la tesis\filtrada.wav"
 TegoWav1.Delete -1, -1
 TegoWav1.Seek 0
 TegoWav1.ArrayToWavFile arreglo(0), tamañoarr
 TegoWav1.SaveAs archivo
```

```
TegoWav1.Close
End Sub
```

El algaritma anterior es utilizado en tadas las procedimientas en dande se requiere grabar la señal.

6.2.6. Análisis de Frecuencia.

Para un estudio más detallada de cualquier tipo de señal, es necesario conocer las frecuencias incluidas dentro de las mismas. En el plano de la frecuencia hay una gran cantidad de operaciones que na se pueden realizar en el daminio del tiempa, que ayudan a modificar la señal. Una herramienta muy útil para el análisis de frecuencia es la transformada de Fourier explicada a cantinuación:

La **transformada discreta de Fourier** es una operación básica usada en muchas aplicaciones de pracesamiento de señal. Se utiliza para transfarmar una secuencia de datos ordenada usualmente en el daminio del tiempo, en una frecuencia cuyo daminio es la frecuencia, así que la infarmación de su espectra puede canacerse explícitamente. Describiremos la transformada discreta de Faurier antes de presentar las algaritmos para calcularla.

La transformada discreta de Fourier es una función compleja de la frecuencia, esto quiere decir que es una secuencia ordenada de números. Usualmente la secuencia original de datos contiene números reales, pero no necesariamente, en cualquiera que sea el caso la transfarmada discreta de Faurier es campleja. Supongamos que tenemas una secuencia de datos de una señal x(t) con números reales, el número de datos de la secuencia es N.

$$\text{Muestras de } x(t) = [x_k]$$
$$= [x_0, x_1, x_2, \ldots, x_{N-1}]$$

La transformada discreta de Fourier es:

$$\hat{X}(f) = \sum_{n=0}^{N} x(nT) e^{-2\pi j f n}$$

que proviene de la transformada de Fourier:

$$\hat{X}(f) = \int_{-\infty}^{\infty} \tilde{x}(t) e^{-i\omega t} dt$$

La transformada rápida de Fourier (FFT) no es otro tipo de transformada diferente a la transformada discreta de Fourier. Por el contrario, es un algoritmo más simple de implementar para la transformada discreta de Fourier, su producto es exactamente el mismo que la transformada discreta de Fourier. El algoritmo de la FFT elimina la mayor parte de los productos repetidos de la transformada discreta de Fourier, así que su tiempo de ejecución es mucho menor, especialmente si el tamaño de la frecuencia es una potencia de 2. La razón de cómputo entre la transformada discreta de Fourier DFT y la transformada rápida de Fourier es aproximadamente:

$$\frac{FFT\ tiempo\ de\ computo}{DFT\ tiempo\ de\ computo} = \frac{1}{2N}\log_2(N)$$

La transformada FFT es más compleja que la transformada DFT y se convierte en más lenta y tardada cuando N, no es potencia de 2.

El algoritmo utilizado en esta tesis para el análisis de frecuencias es el siguiente.

```
Private Sub Command1_Click()
If File1 <> "" Then
archivo = File1
TegoWav1.Open "d:\Señales de la tesis\" + archivo
TegoWav1.Seek 0
tamañoarr = TegoWav1.GetLength * TegoWav1.GetNumChannels
ReDim arreglo(tamañoarr)
TegoWav1.WavFileToArray arreglo(0), tamañoarr
TegoWav1.Close
Dim arreglo1() As Complex
Dim arreglo2() As Complex
Dim fftgrafica() As Double
tamañoarreglo = tamañoarr
L = tamañoarr / 2
ReDim arreglo1(L + 1)
ReDim arreglo2(L + 1)
ReDim fftgrafica(L + 1)

Dim C As Long
  C = 0
 For i = 1 To tamañoarr Step 2
  arreglo1(C).x = arreglo(i)
  arreglo1(C).y = 0
```

```
arreglo2(C).x = arreglo(i - 1)
arreglo2(C).y = 0
C = C + 1
Next i

arreglo1(65536).x = 0
arreglo1(65536).y = 0
arreglo2(65536).x = 0
arreglo2(65536).y = 0
Dim arr1 As Double
Dim arr2 As Double
Call FFT(arreglo1(), 65536)
For i = 0 To 65535
arr1 = arreglo1(i).x
arr2 = arreglo1(i).y
fftgrafica(i) = Sqr(Abs(arr1) + Abs(arr2))
Next i
fftgrafica(0) = 0
fftgrafica(1) = 0
Dim x(65536) As Double
Dim ymin As Double
Dim ymax As Double
Call MAXMIN(fftgrafica(), 65536, ymin, ymax)
Title$ = "Gráfica de la señal"
XTITLE$ = ""
YTITLE$ = "Frecuencia"
FormGrafica.Show
FormGrafica.ForeColor = RGB(0, 0, 255)
FormGrafica.BackColor = RGB(255, 255, 255)
Call SETLINEAR(FormGrafica, -1, 65536 / 2, ymin, ymax, Title$, XTITLE$, YTITLE$)
For i = 0 To 65535
x(i) = i
Next i
Call LINGRAPH(FormGrafica, x(), fftgrafica(), 65536 / 2, 0)
End If
End Sub
```

Se observa que este procedimiento llama al procedimiento de cálculo de la transformada de Fourier FFT. El algoritmo de cálculo de la transformada rápida de Fourier es:

```
Sub FFT(YAR() As Complex, NN As Long)
  Dim i As Long
  Dim ii As Long
    N = 2 * NN: ReDim y(N)
    NTEST = Log(NN) / Log(2)
  If Abs(NTEST - Int(NTEST)) > 0.01 Then MsgBox ("ARRAY DIMENSION SHOULD BE 2 ^ M"): Exit Sub
    For i = 1 To NN
      ii = 2 * i - 1
```

```
    y(ii) = YAR(i).x: y(ii + 1) = YAR(i).y
  Next i
  Call BFFT(y(), NN, 1)
  For i = 1 To NN
    ii = 2 * i - 1
    YAR(i).x = y(ii): YAR(i).y = y(ii + 1)
  Next i
End Sub
```

que a su vez hace un llamado a la función BFFT que es una subrutina unidimensional (compleja) .

```
Sub BFFT(YAR(), NN As Long, iflag)
  Dim j As Long
  N = 2 * NN: j = 1
  Dim i As Long
  Dim MMAX As Long
  Dim istep As Long

  For i = 1 To N Step 2
    'If J > I Then SWAP YAR(J), YAR(I): SWAP YAR(J + 1), YAR(I + 1)
    If j > i Then
      Swap YAR(j), YAR(i)
      Swap YAR(j + 1), YAR(i + 1)
    End If
    M = N / 2
1     If M >= 2 And j > M Then j = j - M: M = M / 2: GoTo 1
    j = j + M
  Next i
'------------------------------>
  MMAX = 2
2   If N > MMAX Then
    istep = 2 * MMAX
    THETA = 6.28318530717959 / (iflag * MMAX)
    SN = Sin(0.5 * THETA)
    WPREAL = -2# * SN * SN
    WPIMAG = Sin(THETA)
    Wr = 1#: Wi = 0#
'------------------------------------------>
    For M = 1 To MMAX Step 2
'------------------------------------>
      For i = M To N Step istep
        j = i + MMAX
        TEMPR = Wr * YAR(j) - Wi * YAR(j + 1)
        TEMPI = Wr * YAR(j + 1) + Wi * YAR(j)
        YAR(j) = YAR(i) - TEMPR
        YAR(j + 1) = YAR(i + 1) - TEMPI
        YAR(i) = YAR(i) + TEMPR
        YAR(i + 1) = YAR(i + 1) + TEMPI
      Next i
'------------------------------------>
```

```
        WTEMP = Wr
        Wr = Wr * WPREAL - Wi * WPIMAG + Wr
        Wi = Wi * WPREAL + WTEMP * WPIMAG + Wi
      Next M
'=================================================>
      MMAX = istep
    GoTo 2
    End If
End Sub
```

Capítulo 7.

Aplicaciones.

7.1. Tipo de aplicaciones.

El procesamiento digital de señales lidia con la representación de señales como una secuencia de números y el procesamiento de éstos. Algunas razones típicas para el procesamiento de señales incluyen: estimación de parámetros característicos, eliminación o reducción de interferencias no deseados, transformación de una señal a un formato que nos proporcione más información.

Un beneficio inmediato del procesamiento digital sobre el procesamiento análogo es la flexibilidad. En general, un sistema digital se adapte más fácilmente al presentarse cambios en el problema. De hecho, simulaciones digitales se utilizan frecuentemente para analizar diseños con la finalidad de identificar errores muy costosos antes de que algún equipo sea construido.

Existen aplicaciones del procesamiento digital de señal en diversos campos tales como: acústica, geofísico, comunicaciones , medicina y en otro tipo de industrias tales como la industria metal-mecánica en donde actualmente se realiza una gran cantidad de estudios para detectar fenómenos tales como: desgaste de dientes en máquinas, defectos en el corte de materiales, análisis metalográficos, etc.

7.2 Efectos digitales de audio.

Efectos de audio tales como: retraso, eco, reverberación, filtrado parabandas, realce (flanging), modificación de velocidad, distorsión, compresión, expansión, compuertas de ruido, y ecualización son indispensables para la producción y ejecución musical.

La mayor parte de estos efectos se realizan usando procesadores digitales de señal.

En todos los sistemas digitales, el muestreo/reconstrucción puede ser eliminado y la señal original de entrada puede permanecer en forma digital, de aquí en adelante, las etapas del procesamiento se harán mediante procesadores de señal digital.

7.2.1 Retrasos, ecos y filtros parabandas.

Quizás el efecto más básico de todos es el retraso en el tiempo, debido a que éste es utilizado para generar efectos más complicados como los reverberadores digitales.

En cualquier espacio acústico, como un cuarto o una sala de concierto, las ondas sonoras que llegan a nuestros oídos consisten en un sonido directo que proviene del instrumento generador como de ondas reflejadas de las paredes y de los objetos en el cuarto que llegan con gran cantidad de retrasos y atenuaciones.

Las reflexiones repetidas resultan de las características de reverberación del espacio.

Una simple reflexión o eco puede implementarse por el siguiente filtro:

$$y(n)=x(n)+\alpha x(n-D) \quad \text{(filtrado con eco)}$$

El retraso D representa el viaje de ido y vuelta desde la fuente a la pared reflejante y de regreso y el coeficiente α es una medida de las pérdidas de propagación de tal forma que $|\alpha| \leq 1$. La función de transferencia y su respuesta al impulso son:

$$H(z)=1+\alpha z^{-D} \quad , \quad h(n)=\delta(n)+\alpha\delta(n-D)$$

Su respuesta a la frecuencia se obtiene mediante $z=e^{j\omega}$

$$H(\omega)=1+\alpha e^{-j\omega D}, \quad |H(\omega)| = \sqrt{1+2\alpha\cos(\omega D)+\alpha^2}$$

7.2.3 Realce.

El valor de D en muestras o en segundos $T_D=DT$, puede tener una cambio drástico en el sonido percibido. Por ejemplo, si el retraso es mayor que 100 milisegundos, la señal retrasada puede escucharse como una repetición rápida. Si el retraso es menor que 10 milisegundos, el eco se junta con la señal. Debido a que sólo ciertas frecuencias se enfatizan, la señal resultante aparecerá como con un vacío (ciertas frecuencias estarán atenuadas y algunas otras desaparecerán). Los retrasos pueden utilizarse para enfatizar el estéreo en los señales y son indispensables para la mezcla estéreo. Por ejemplo, un retraso de pocos milisegundos aplicado a un canal causará un cambio y

una mejor distribución del estéreo. Similarmente una señal monoaural a la que se le aplica un retraso tan pequeño se escuchará en estéreo.

Efectos más interesantes tales como el realce pueden crearse al hacer que D varíe con el tiempo. Por ejemplo:

$y(n)=x(n)+ax(n-d(n))$ Procesador de realce (flanging procesor)

Un efecto de realce puede crearse al variar periódicamente el retraso d(n) entre 0 y 10 msec con una frecuencia muy baja tal como 1Hz. Por ejemplo, una onda senoidal que varíe entre los límites $0 \le d(n) \le D$ será:

$$d(n)=(D/2)(1-\cos(2\pi F_d n))$$

en donde F_d es una frecuencia muy baja, en unidades de (ciclos/muestra).

En tiempos pasados, el efecto de realce se lograba al poner dos canciones a la vez y manualmente variar la velocidad de una de las dos.

7.2.4 Reverberación.

La reverberación en un espacio acústico se caracteriza típicamente por tres periodos distintivos: el sonido directo, reflexiones inmediatas y las reflexiones tardías. Las reflexiones inmediatas son las primera reflexiones de las paredes. Mientras que las ondas siguen reflejándose a través de las paredes, su densidad se incrementa y se dispersan llegándose a escuchar por todas las direcciones; estas son las reflexiones tardías. El tiempo de reverberación es la constante de tiempo que toma a la respuesta al impulso del cuarto, decrementarse 60dB. Las salas de concierto tienen típicamente una constante de tiempo de alrededor de 1.8-2 segundos.

La calidad de la sala de conciertos depende de la respuesta al impulso en su reverberación, que depende a su vez de la localidad relativa del emisor y receptor de la señal. Debido a esto, la simulación digital de las características de reverberación de cualquier sala es una tarea casi imposible. Los reverberadores digitales pretenden simular la respuesta al impulso típica de una sala de conciertos, dando al usuario la

flexibilidad de cambiar cualquiera de sus parámetros, tales como la duración de las reflexiones inmediatas o el tiempo de reverberación total.

Otros efectos de reverberación que pueden conseguirse digitalmente son virtualmente imposible de conseguirlos por medios analógicos. Como por ejemplo, la reverberación con compuertas, conseguida al truncar la respuesta de un filtro IIR con un FIR.

7.2.5. Compresores, limitadores, expansores y compuertas.

Los compresores, limitadores, expansores y compuertas tienen una gran cantidad de usos en procesamiento de señales de audio. Los compresores atenúan señales muy fuertes; los expansores realzan señales débiles. Debida a que ambos afectan el rango dinámico de la señal se les denomina procesadores dinámicos.

Los compresores se utilizan básicamente para decrementar el rango dinámico de señales de audio de tal forma que encajen en el rango dinámico del sistema de reproducción o de transmisión; por ejemplo, para poner una grabación en una cinta de audio. Existen otras aplicaciones tales como, eliminación de música de fondo, eliminación de vicios en los micrófonos, etc.

Los expansores se utilizan para incrementar el rango dinámico de las señales, para reducción de ruido, y otros efectos, como la reducción del tiempo de sostén de algún instrumento.

7.2.6 Diseño de sistemas de ecualización y filtrado de bandas (*crossover filters*).

Usando la ventana Kaiser podemos diseñar un filtro digital con las requeridas. Supongamos una frecuencia de muestreo de 44.1KHz, una atenuación de 65dB. Supongamos también una región de transición de 2khz. Todos los filtros tendrán el mismo tamaño N y los mismos parámetros de Kaiser D y α. Con $A=A_{stop}=65dB$ tenemos que:

$$D = \frac{A - 7.95}{14.36} = 3973, \alpha = 0.1102(A - 8.7) = 6.204$$

$$N - 1 = \frac{Df_s}{\Delta f} = \frac{3973 * 44.1}{2} = 87.6 \Rightarrow N = 89, M = \frac{1}{2}(N - 1) = 44$$

Nótese que, dada un valor A=65dB que corresponde a $\delta_{pass}=\delta_{stop}=10^{-65/20}=0.00056$, que resulta en una atenuación de $A_{pass}=0.0097$ dB.

Este filtro se encuentra dentro de la capacidad de los microcircuitos modernos DSP. Suponiendo una velocidad de procesamiento de 20MIPS, que significa 20 instrucciones por microsegundo, o $T_{instr}=50$ nseg por instrucción y suponiendo que el procesamiento requiere N MAC's por cada salida, el tiempo total de cómputo para procesar cada muestra será de 89*50 microsegundos, que encaja perfectamente en la separación de cada muestra, $T=1/f_s=1/44.1=22.68$ microsegundos.

Varios filtros pueden implementarse simultáneamente en el misma microcircuito DSP. 22.68/4.45=5.1 ó alrededor de 5 filtros de tamaño 89 pueden incluirse en un solo microcircuito DSP. Asimismo el filtro más grande que puede implementarse tendrá un tamaño tal que $NT_{instr}=T$ o $N=T/T_{instr}=f_{instr}/f_s=20000/44.1\approx 453$, que resulta en una frecuencia de transición mínima de:

$$\Delta f_{min} \approx \frac{Df_s}{N} = \frac{Df_s^2}{f_{instr}} = 0.386kHz$$

En la actualidad, la gran mayoría de las bocinas contienen un separador de frecuencias (*crass over*) analógico que divide la señal análoga de entrada en sus frecuencias altas y sus frecuencias bajas que activan al baja (*woafer*) y al agudo (*tweeter*). Equipos más avanzados pueden tener un tercer componente que divide las frecuencias medias.

Los bocinas digitales operan con una entrada digital de audio y utilizan filtros FIR e IIR para separar la señal en las frecuencias requeridas. Estas frecuencias se convierten posteriormente a un formato analógico, se amplifica y activan a los componentes del equipo. Estos equipos digitales han existido desde hace ya algún tiempo en estudios

profesionales y actualmente se pueden conseguir comercialmente para uso domástico (típicamente la salida de los CD se conecta a la entrada digital de la bocina).

7.3 Reducción de ruido y definición de señal.

Uno de los problemas más comunes en el procesamiento de señales es el extraer la señal deseada, digamos x(n), de una señal con ruido.

$$x(n)=s(n)+v(n)$$

en donde v(n) es una componente no deseada de la señal.

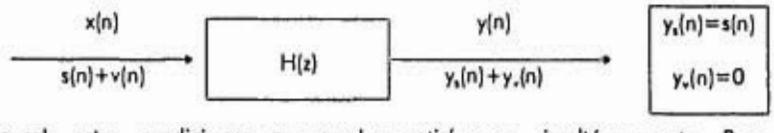
La señal del ruido v(n) depende de la aplicación. Por ejemplo, (1) puede ser una señal blanca, que es típicamente la señal que se produce en el proceso de muestreo; (2) una señal periódico de interface tal como la de los 60Hz de la señal de entrada de la línea de corriente; (3) una frecuencia muy baja, como la de un radar; (4) cualquier otro señal, no necesariamente ruido, que deba separarse de s(n).

El método standard para extraer s(n) de x(n) es el diseñar un filtro apropiado H(z) que elimine los componentes del ruido v(n) y que al mismo tiempo permita pasar la señal s(n) sin modificación. Usando la propiedad de linealidad, podemos expresar la señal de salida producida por la entrada:

$$y(n)=y_s(n)+y_v(n)$$

en donde $y_s(n)$ es la salida producida por s(n) y $y_v(n)$ producida por v(n)

Las dos condiciones de diseño para el filtro son: que $y_v(n)$ sea lo más pequeña posible y que $y_s(n)$ sea lo más parecido a s(n) . Idealmente requerimos



En general, estas condiciones no pueden satisfacerse simultáneamente. Para determinar cámo pueden ser satisfechas, las expresamos en términos de su frecuencia y de su espectro de frecuencia correspondiente como sigue: $Y_s(\omega)=S(\omega)$ y $Y_v(\omega)=0$.

Usando la ecuación de filtrado $Y(\omega)=H(\omega)X(\omega)$ separadamente en las componentes de la señal y en las del ruido

$$Y_s(\omega)=H(\omega)S(\omega)=S(\omega)$$

$$Y_v(\omega)=H(\omega)V(\omega)=0$$

El primera requiere que $H(\omega)=1$ para toda ω en donde el espectro de la señal no sea cero, $S(\omega)\neq 0$. El segunda requiere que $H(\omega)=0$ para todo ω en donde el espectro del ruido no sea cero, $V(\omega)\neq 0$.

Estas condiciones deberán cumplirse simultáneamente sólo si el espectro de la señal y el ruido no se traslapan. En estos casos, el filtro $H(\omega)$ deberá tener filtros pasabandas que coincidan con la frecuencia de la señal y filtros parabandas en donde se coincida con la frecuencia del ruido.

Si la señal y el ruido se traslapan en alguna frecuencia, que es el caso más usual, las condiciones especificadas no podrán satisfacerse simultáneamente debido a que existirán valores de w tales que $S(\omega)\neq 0$ y $V(\omega)\neq 0$, en donde las condiciones requieren que $H(\omega)=1$ y que $H(\omega)=0$ para el mismo valor de ω.

7.4 Limpieza de líneas y bordes para televisión digital.

Otra aplicación para los filtros pasabajas y pasabandas es el caso en donde ambas señales s(n) y v(n) son periódicas y deben separarse una de la otra.

Para extraer s(n), una puede utilizar ya sea un filtro de pico (pasabandas aguda) con picos en las frecuencias armónicas de s(n) o de ranura (parabandas aguda) con valles en las armónicas de v(n). Similarmente, para extraer v(n), uno deberá usar un pico en las frecuencias armónicas de v(n) o un filtro de ranura en las frecuencias armónicas de s(n). Para que este método funcione, las frecuencias armónicas no deberán coincidir una con la otra.

Una aplicación muy importante de esta idea se da en la televisión digital a color, discos de video y sistemas de televisión de alta resolución. Los filtros ranura se utilizan para

separar: la luminiscencia (colores blanco y negro) de las frecuencias cromáticas (colores) y para reducir el ruido o interferencia.

Bibliografía.

BLOCK, SHAKIR Y SMITH, Promath 2.0 and Promath/VB 2.0 dual version, Rockville MD, U.S.A. ,TeraTech,Inc, 1987. http: // www.teratech.com / teratech, ftp.teratech.com/pub/teratech.

BURDEN Y FAIRES, Análisis Numérico., México D.F, Grupo Editorial Iberoamérica, 1991.

FLOTOW AND MERCADAL, "The measurement of noise and vibration transmitted into aircraft cabins", The Moth Works Inc, MA, U.S.A. 1995

FORESMAN, Scott, Intermediate Dictionary, Glenview, Illinois, U.S.A, Ed. Scott, Foresman and Company, 1979.

OGATA, Katsuhiko, Discrete Time Control Systems,New Jersey, U.S.A., Prentice Hall,1987.

OGATA, Katsuhiko, Ingeniería de Control Moderno, México D.F, Ed. Prentice Hall Hispanoamericana, S.A., 1993, segunda edición.

ORFANIDIS , Sophocles J. Introduction to Signal Processing. , New Jersey U.S.A., Prentice-Hall Signal Processing Series, 1996, segunda edición.

PITA RUIZ, Claudio de Jesús, Ecuaciones Diferenciales, Una introducción con aplicaciones, México D.F, Ed. Limusa, 1989.

STEARNS, DAVID, Signal Processing Algorithms In Matlab, New Jersey, U.S.A, Prentice Hall Signal Processing Series, 1996.

"Enciclopedia BARSA", Nuevo Diccionario de Pronunciación de las Lenguas Inglesa y Española, New York, U.S.A, Ed. Meredith Publishing Company, 1967, 4ta edición.

"Developer Kit for Sound Blaster Series Programmer's Guide", U.S.A., Ed. Creative Labs, 1993, segunda edición

"Documentation for the TegoSoft Wave OCX Control", NY, U.S.A. ,TegoSoft Inc. 1996. http://www.tegosoft.com

"Documentation for the TegoSoft OCX Control", NY, U.S.A. ,TegoSoft Inc. 1996. http://www.tegosoft.com

"Microsoft Visual Basic, Language Reference", U.S.A., Microsoft Corporation, 1995.

"Microsoft Visual Basic, Programmer's Guide", U.S.A., Microsoft Corporation, 1995.

Anexos.

Anexo A. Función de autocorrelación.

La función de autocorrelación de una señal con promedio cero[13] se define como la correlación entre dos muestras x(n) y x(n+k) separadas por un tiempo k. Es una medida de la dependencia de las muestras sucesivas:

$$R_{xx}(k) = E\left[x(n+k)x(n)\right] \quad \text{(función de autocorrelación)}$$

Para señales estacionarias, Rxx(k) depende sólo en el distanciamiento relativo k y no del tiempo absoluto n. Nótese que Rxx(k) es una secuencia de dos lados. Al ser estacionaria, es simétrica en k, que quiere decir que Rxx(-k)=Rxx(k).

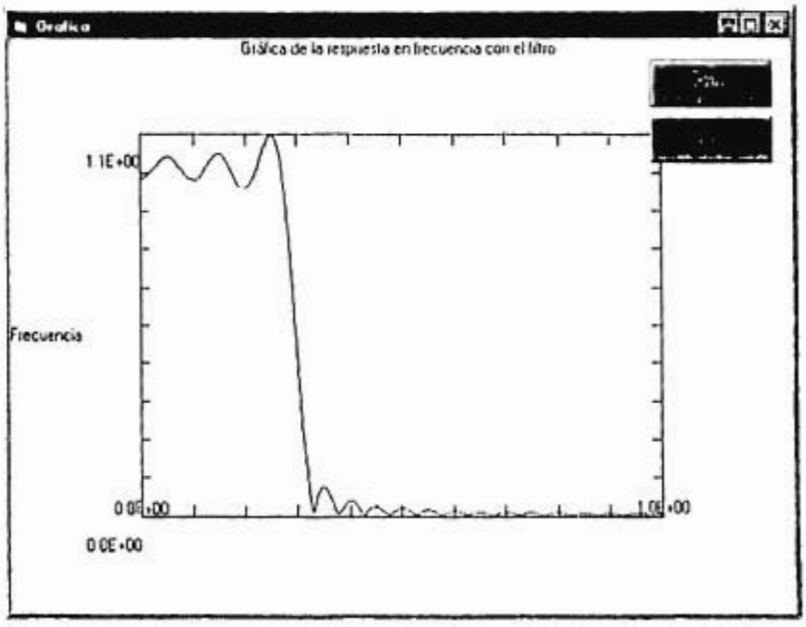
El espectro de potencia (*power spectrum*) se define como la transformada de Fourier en tiempo discreto de la función de autocorrelación Rxx(k). Representa el contenido en frecuencia de una señal aleatoria x(n) visto como un promedio:

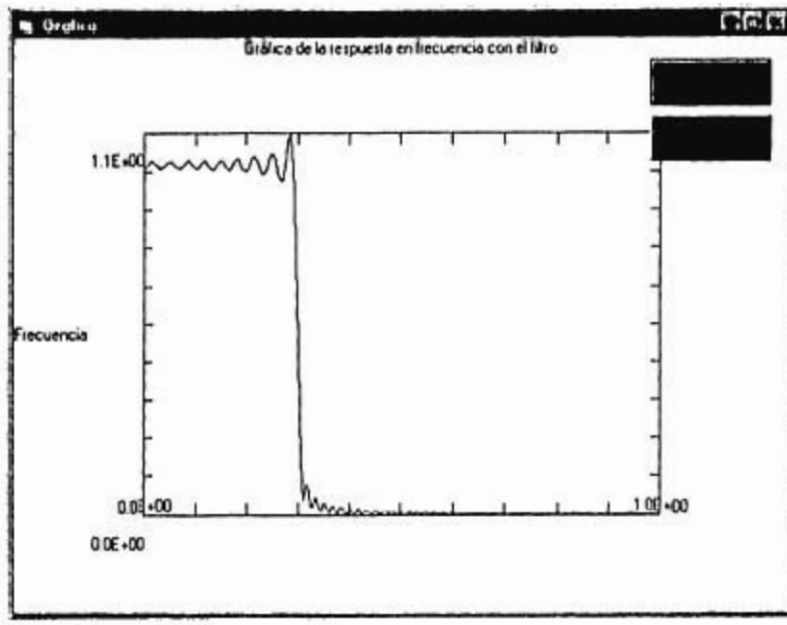
$$Sxx(\omega) = \sum_{k=-\infty}^{\infty} Rxx(k)e^{-j\omega k} \quad \text{espectro de potencia.}$$

en donde $\omega = 2\pi f/f_s$ es la frecuencia en radianes por muestra.

[13] Si la media no es cero, reemplazaremos x(n) por su versión en con media-cero x(n)-m.

Anexo B. Gráficas de la respuesta en frecuencia de un filtro pasabajas. Ventana rectangular, filtros pasabajas con diferente tamaño de N.




Gráfica de la respuesta en frecuencia con el filtro
1.1E+00
Frecuencia
0.0E+00
1.0E+00

Anexo C. Procedimientos en Visual Basic.

Gráficas:

```
Sub LINGRAPH(FormName As Form, x(), y(), N As Long, pattern%)
   FormName.PSet (x(1), y(1))
   Dim i As Long
   For i = 2 To N
     savestyle = FormName.DrawStyle
     FormName.DrawStyle = pattern%
     FormName.Line -(x(i), y(i))
     FormName.DrawStyle = savestyle
     i = i + 1
   Next i
End Sub

Sub SETLINEAR(FormName As Form, Xp0, Xp1, Yp0, Yp1, Title$, xtit$, ytit$)
   Xr = Abs(Xp1 - Xp0)
   Yr = Abs(Yp1 - Yp0)
   Sr = 0.25
   gx0 = Xp0 - Sr * Xr: gx1 = Xp1 + Sr * Xr
   gy1 = Yp1 + Sr * Yr: gy0 = (Yp0 - Sr * Yr)
   FormName.Scale (gx0, gy1)-(gx1, gy0)
   FormName.Cls

 FormName.Line (Xp0, Yp0)-(Xp1, Yp1), , B
   Ncyclesx = 10: Ncyclesy = 10
   Dx = (Xp1 - Xp0) / Ncyclesx: xx = Xp0
   Xtic = (Yp1 - Yp0) * 0.03
   Dy = (Yp1 - Yp0) / Ncyclesy: yy = Yp0
   Ytic = (Xp1 - Xp0) * 0.02
   For i = 1 To Ncyclesy
     yy = yy + Dy
     FormName.Line (Xp0, yy)-(Xp0 + Ytic, yy)
     FormName.Line (Xp1, yy)-(Xp1 - Ytic, yy)
   Next i
   For i = 1 To Ncyclesx
     xx = xx + Dx
     FormName.Line (xx, Yp0)-(xx, Yp0 + Xtic)
     FormName.Line (xx, Yp1)-(xx, Yp1 - Xtic)
   Next i
   FormName.CurrentX = gx0 + FormName.ScaleWidth / 2 - FormName.TextWidth(Title$ + " ") / 2
   FormName.CurrentY = gy1    ' Top line '
   FormName.Print Title$
 ' Print axis titles
   FormName.CurrentX = gx0
   FormName.CurrentY = Yp1 - Yr / 2
   FormName.Print ytit$
   FormName.CurrentX = gx0 + FormName.ScaleWidth / 2 - FormName.TextWidth(xtit$ + " ") / 2
   FormName.CurrentY = Yp0 - 1.5 * FormName.TextHeight(xtit$)
```

```
    FormName.Print xtit$
    NL$ = Format$(Yp0, "0.0E+00")
    FormName.CurrentX = Xp0 - FormName.TextWidth(NL$)
    FormName.CurrentY = Yp0 + 1 * FormName.TextHeight(NL$)
    FormName.Print NL$
    NL$ = Format$(Yp1, "0.0E+00")
    FormName.CurrentX = Xp0 - FormName.TextWidth(NL$)
    FormName.CurrentY = Yp1 + 1 * FormName.TextHeight(NL$)
    FormName.Print NL$
    NL$ = Format$(Xp0, "0.0E+00")
    FormName.CurrentX = Xp0 - FormName.TextWidth(NL$) / 2
    FormName.CurrentY = Yp0 - 1 * FormName.TextHeight(NL$)
    FormName.Print NL$
    NL$ = Format$(Xp1, "0.0E+00")
    FormName.CurrentX = Xp1 - FormName.TextWidth(NL$) / 2
    FormName.CurrentY = Yp0 - 1 * FormName.TextHeight(NL$)
    FormName.Print NL$
    FormName.CurrentX = 0#
    FormName.CurrentY = 0#
End Sub
```

Evaluación de la función $I_0(x)$, función modificada de Bessel clase 1 y orden 0.

```
Function Bsi0(x)
  Dim aa, bb, cc As Double
    par1 = 1#: par2 = 3.5156229: par3 = 3.0899424: par4 = 1.2067492
    par5 = 0.2659732: Par6 = 0.0360768: Par7 = 0.0045813
    red1 = 0.39894228: red2 = 0.01328592: red3 = 0.00225319: red4 = -0.00157565
    red5 = 0.00916281: Red6 = -0.02057706: Red7 = 0.02635537: Red8 = -0.01647633
    Red9 = 0.00392377
    If Abs(x) < 3.75 Then
      y = (x / 3.75): y = y * y
      aa = y * (Par6 + y * Par7)
      bb = y * (par5 + aa)
      cc = y * (par4 + bb)
      yy = par2 + y * (par3 + cc)
      Bsi0 = par1 + y * yy
    Else
      Ax = Abs(x): y = 3.75 / Ax
      yy = y * (Red6 + y * (Red7 + y * (Red8 + y * Red9)))
      aa = y * (red5 + yy)
      bb = y * (red4 + aa)
      cc = y * (red3 + bb)
      yy = red1 + y * (red2 + cc)
      Bsi0 = yy * Exp(Ax) / Sqr(Ax)
    End If
End Function
```

Variables globales:

```
Public arreglo() As Integer
Public arregl1() As Integer
Public arregl1fil() As Integer
Public arregl2() As Integer
Public arregl2fil() As Integer
Public tamañoarr As Long
Public M As Double
Public N As Long
Public h() As Double
Public Milltrah As Integer
```

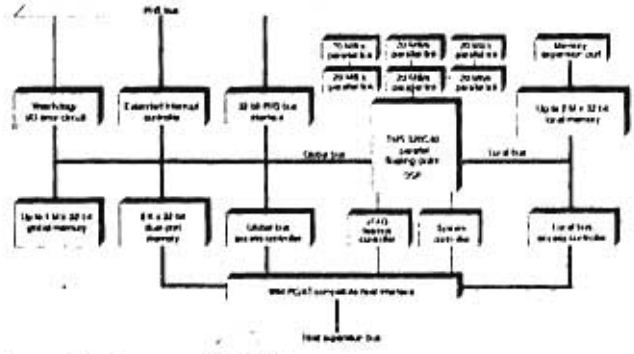
Función de redondeo.

```
Function round(xy As Double) As Double
Select Case xy - Int(xy)
Case Is > 0.49
xy = Int(xy) + 1
Case Is <= 0.49
xy = Int(xy)
End Select
round = xy
End Function
```

Declaración de números complejos y constante pi.

```
DefDbl A-H, O-Z: DefInt I-N
Type Complex
   x As Double
   y As Double
End Type
Global Const PmArrayBase = 1
Global Const Pi = 3.14159265358979
```

Anexo D. Tarjeta para procesamiento digital de señales.



Processor: Texas Instruments TMS320C40

- 50 MHz clock rate, 40 ns cycle time
- 32/32-bit floating-point/integer multiplier and ALU
- single-cycle floating-point multiply/accumulate
- 2 K words of 32-bit on-chip RAM
- 128 words of 32-bit instruction cache
- 6 parallel 8-bit wide communication ports with 20 MB/s transfer rate supported by on-chip DMA controller
- boot on chip timers

Local memory

- up to 2 M words of 32-bit on-board SRAM, zero wait states
- DSP and host access arbitrated by bus controller
- connector for expansion to up to 5 M words local memory
- intended to be used as instruction memory

Global memory

- up to 1 M words of 32-bit on-board SRAM, zero wait states
- DSP- and host access arbitrated by bus controller
- intended to be used as data memory

Dual-port memory

- 8 K words of 32-bit true dual-port RAM
- parallel DSP and host access

Safety provisions

- programmable watchdog
- I/O error controller

Interrupts

- programmable interrupt controller with 8 interrupt inputs, expandable to up to 61 inputs via PHS-bus
- communication port interrupts
- DMA interrupts
- DSP to host and host to DSP interrupt
- watchdog interrupt
- DSP internal timer interrupts
- I/O error interrupt/NMI

PHS-bus interface

- 32-bit synchronous I/O bus
- 20 MB/sec sustained transfer speed
- 256 32-bit memory mapped I/O channels
- host access to all I/O channels even while the DSP is running
- synchronization lines for parallel sampling and update on all I/O boards

PC/AT-bus interface

- eight 8-bit I/O ports in host I/O space
- 1024 pages of 64 KB or
- 256 pages of 256 KB in 16 MB host memory space
- bus-width converter guarantees data integrity during 32-bit transfers

Power supply

- 5 V, max. 2.5 A

Physical size

- 340 x 111 x 23 mm, 16-bit PC/AT slot

We are continually improving our products, and reserve the right to alter the above specifications at any time, without notice.

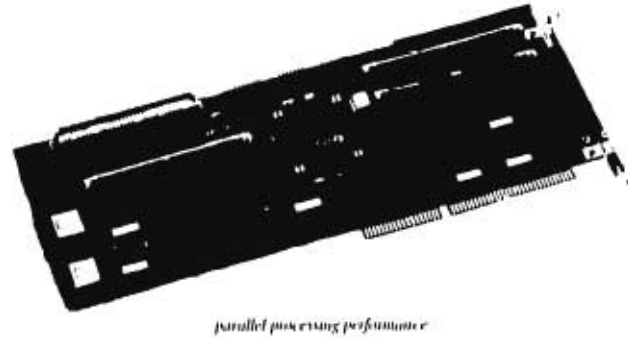
dSPACE

digital signal

Technologiepark 25

dSPACE

DS1003 Parallel DSP Board



parallel processing performance

The high-performance DS1003 processor board is based on the Texas Instruments TMS320C40 parallel processing DSP and is designed for a standard IBM®-PC/AT environment, but can also be connected to a workstation via the network kit.

With 50 MFlops peak performance of the CPU, the board guarantees high-speed floating-point operation and simultaneous interprocessor communication for sophisticated real-time control and hardware-in-the-loop simulation tasks in various fields such as robotics, vehicle dynamics, and many others. The DSP's six communication ports with a 20 MB/sec peak transfer rate are user accessible via standard link connectors to add further TMS320C40-based boards for sharing very demanding computation tasks.

The board is equipped with up to 3 M words dual-ported zero-wait-states memory and 8 K words of true dual-port memory for simultaneous DSP and host access. Two data busses on-board enable the DSP to perform concurrent data and instruction fetches. An expansion connector allows memory expansion to up to 5 M words local memory.

The DS1003 extends the flexibility of dSPACE's hardware line and can be connected most effectively to all dSPACE I/O boards via the 32-bit wide PHS bus interface supporting the full word length of the TMS320C40 at 20 MB/sec transfer speed.

Watchdog and I/O-error logic are safety features required by many control applications.

Setup and loader software for PC/AT or Sun® SPARC® workstations is included. Host-DSP Interface Libraries are available in C language.

IBM is a registered trademark of International Business Machines.
Sun is a registered trademark of Sun Microsystems Inc.
SPARC is a registered trademark of SPARC International Inc.