Guía N° 2: *Procesamiento de señales con Labview*

Objetivo:

Introducir al alumno al procesamiento de señales aprovechando las herramientas disponibles en Labview.

Fecha de entrega:

10/05/2019

Formato de entrega:

Cada ejercicio se corresponde a un archivo .vi, con nombre e*N_M.vi*, donde N es el nombre de la guía y M el número de ejercicio. El contenido de cada archivo .vi debe contener resuelto el último punto de cada ejercicio, ignorando aquellos puntos intermedios que no sean necesarios para completar el punto final.

Todos los archivos se deberán comprimir en un solo archivo .zip o .rar, con nombre guia2 y enviarse por correo con asunto *Entrega Guía 2*. En el cuerpo del correo agregar los legajos de todos los integrantes del grupo.

Bibliografía recomendada:

LabVIEW Basics I Introduction Course Manual

The Scientist and Engineer's Guide to Digital Signal Processing

Introducción al procesamiento digital de señales

La información en el mundo real es predominantemente analógica. Señales como la luz, el sonido, la temperatura, etc, pueden representarse como funciones de tiempo continuo f(t). Operar con estas señales implica utilizar elementos de procesamiento analógicos como amplificadores operacionales, resistencias, capacitores, etc. Este tipo de operación está condicionado por las tolerancias de los elementos, el rango de trabajo y principalmente, el ruido.

El primer paso en el procesamiento digital de señales es el de convertir señales analógicas en digitales para simplificar la operación. Esto se realiza mediante dos procesos:

1) Muestreo, para digitalizar el eje temporal y trabajar con una cantidad finita de muestras.
2) Cuantización, para digitalizar los valores de la señal.

Existe una enorme cantidad de operaciones disponibles al trabajar con señales digitales. En esta guía vamos a explorar algunas de estas operaciones y nos vamos a centrar en trabajar con señales muestreadas, pero sin cuantizar.

Filtrado

En el procesamiento de señales, el filtrado es un proceso que elimina algunas componentes o características no deseadas de una señal. El filtrado es una clase de procesamiento de señal que se define por la supresión total o parcial de algún aspecto de la señal. Lo más frecuente es quitar algunas frecuencias en lugar de otras para suprimir las señales de interferencia y reducir el ruido de fondo.

1. Reducción de Ruido

Uno de los métodos más utilizados para reducir el ruido en las mediciones es el promediado, es decir la generación muestras de salida compuestas por el promedio de varias muestras de entrada. Matemáticamente:

$$y[n] = \sum_{i=0}^{N-1} \frac{x[n-i]}{N}$$

Se puede observar de la ecuación que cada muestra *y[n]* es el promedio de las últimas *N* muestras de *x*.

En este ejercicio se evaluarán las propiedades del filtro promediador para la remoción de ruido gaussiano.

- Crear un nuevo VI. En el panel frontal agregar un control numérico llamado *N*, otro llamado *Nivel de ruido* y uno llamado *Frecuencia*. Agregar también un visualizador gráfico de tipo **Waveform Graph**.
- Agregar un nodo de generación de ruido desde **Signal Processing → Waveform Generation → Gaussian White Noise Waveform**. Conectar el control *Nivel de ruido* a la entrada *standard deviation*. Esta será nuestra fuente de ruido.
- Agregar un nodo de generación de onda cuadrada desde **Signal Processing → Waveform Generation → Square Waveform**. Conectar el control *Frecuencia* a la entrada *frequency*. Esta será nuestra señal.
- Sumar ambas señales con un nodo de suma.
- Agregar un nodo de convolución continua desde **Signal Processing → Waveform Conditioning → Continuous Convolution**.
  Este nodo implementa la siguiente ecuación:

$$y[n] = x[n] * h[n] = \sum_{i=0}^{N-1} x[n-i]h[i]$$

donde *h[n]* se conoce como kernel de convolución.

- Comparando la ecuación del promediador con la ecuación de la convolución, obtener los valores del vector *h[n]*, y crearlo utilizando el control N y elementos de la paleta **Programming → Array** y de la paleta **Programming → Numeric**.
- Conectar la salida del nodo suma a la entrada *signal in* del nodo de convolución y conectar el vector *h[n]* del punto anterior a la entrada *kernel*. Conectar la salida del nodo de convolución al visualizar gráfico.
- Comenzando con *Nivel de ruido=1* y *Frecuencia=1* probar el filtro para *N=1*, *N=10* y *N=100*. Responder de manera cualitativa las siguientes preguntas como comentarios en el diagrama:
  - Cual es el efecto de *N* en la cantidad de ruido de salida
  - Cual es el efecto de *N* en la forma de la señal de salida

2. Caracterización del filtro promediador - SNR

Una manera de determinar cuánto se reduce el nivel de ruido a la salida de un filtro es evaluar la relación señal ruido (SNR) en ese punto.

$$\mathrm{SNR} = 10 \log_{10} \frac{P_s}{P_n}$$

El problema de la ecuación anterior es que en general no se dispone de la señal y el ruido por separado. A los efectos de evaluar el filtro, podemos crear un esquema que nos permita disponer de ambas señales.
Como el promediado es una operación lineal (como lo es la convolución) se cumple la propiedad distributiva en la suma de la señal y ruido.

$$y[n] = (x[n] + u[n]) * h[n]$$
$$y[n] = x[n] * h[n] + u[n] * h[n]$$

Por lo tanto a la salida tendremos la suma de la señal con el ruido, ambas convolucionadas con el promediador.
A partir de esto la SNR a la salida resulta

$$\mathrm{SNR} = 10 \log_{10} \frac{P(x[n] * h[n])}{P(u[n] * h[n])} = 10 \log_{10} \frac{P(x[n] * h[n])}{P(y[n] - x[n] * h[n])}$$

Podemos calcular la SNR generando una nueva señal a partir de la convolución de la entrada sin ruido con el kernel del promediador. Esto lo hacemos para poder caracterizar el filtro y porque disponemos en nuestro entorno de prueba de la señal sin el ruido.

- Crear un nuevo VI tomando de base el VI del ejercicio anterior.
- Agregar otro nodo de convolución y generar la nueva señal convolucionando sólo con onda cuadrada (sin el ruido).
- Obtener la señal de ruido filtrada haciendo la resta *y[n]-x[n]*h[n]*.
- Calcular la potencia de la señal de ruido y de la señal cuadrada filtrada. La potencia de una secuencia finita de muestras se obtiene de la siguiente manera:

$$P(x) = \sum_{i=0}^{M-1} \frac{x[i]^2}{M}$$

  Para el caso de señales alternas, se puede utilizar la varianza de la señal, simplificando el cálculo. Este nodo se encuentra en **Programming →Mathematics → Probability & Statistics →Std Deviation and Variance**.
- Obtener la SNR en decibeles.
- Agrupar todos en los elementos en un bucle FOR, conectar la cantidad de iteraciones a una constante de valor 100. Reemplazar el control N con el índice del bucle FOR más 1 (i+1). Esto nos permitirá variar automáticamente el tamaño de la promediación.
- Agregar un visualizador gráfico por fuera del bucle FOR y conectar el vector de SNRs.
- Ejecutar el VI y comprobar que la SNR aumenta al aumentar la cantidad de puntos del promediador.

3. Caracterización del filtro promediador - Ancho de banda

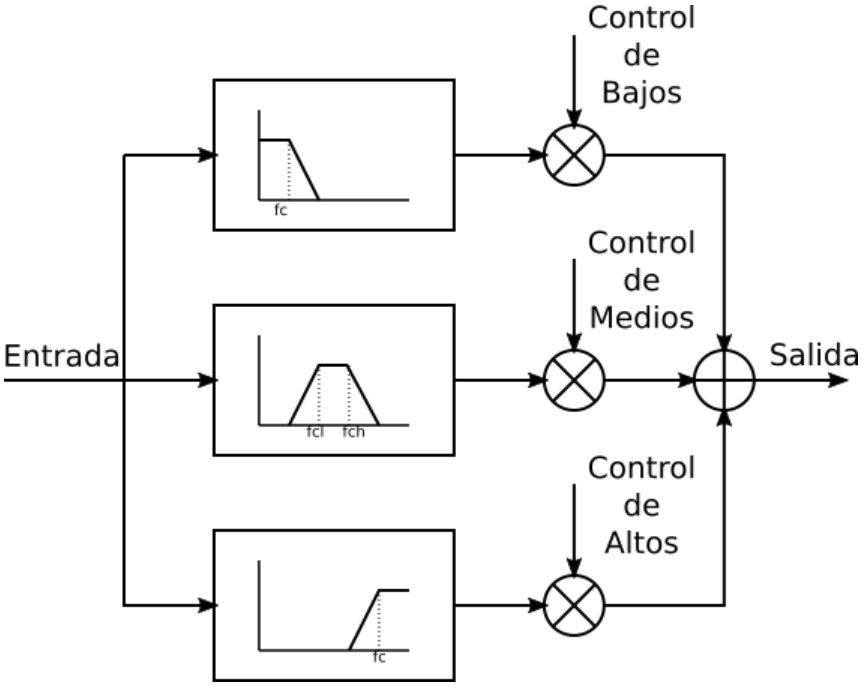
Como se vió en el ejercicio 1, el promediador no sólo afecta al ruido, sino que también afecta a la señal, afectando al ancho de banda de la señal. Para poder evaluar el efecto del promediador en el ancho de banda se estimulará el filtro con tonos de distintas frecuencias y se graficarán las amplitudes de salida.

Una aclaración importante en este ejercicio, es que los nodos de generación de señal permiten configurar la cantidad de muestras de la señal resultante y la frecuencia de muestreo de la misma. Esto se hace mediante un terminal de configuración denominado *sampling info*, ubicado en la parte inferior del nodo. Por defecto los nodos se configuran con 1000 muestras de salida y una frecuencia de muestreo de 1000Hz.

- Partiendo del ejercicio 1, crear un nuevo VI. Remover el agregado de ruido.
- Reemplazar el nodo de generación de señal cuadrada por un nodo de generación senoidal desde **Signal Processing → Waveform Generation → Sine Waveform.**
- Agrupar los elementos en un bucle FOR, conectar la cantidad de iteraciones a una constante de valor 500. Conectar el índice del bucle a la entrada de frecuencia del generador de señales sinusoidales, sumandole +1. La razón de utilizar 500 iteraciones es que el nodo de generación senoidal, al usarse por defecto posee una frecuencia de muestreo de 1Khz, por lo que sólo podremos utilizar frecuencias hasta 500Hz.
- Obtener la varianza de la salida, dividirla por la varianza de la entrada y expresar el resultado en decibeles (ver guía 1).
- Colocar un visualizador gráfico en el panel frontal y conectarlo a la salida del cálculo de decibeles, por fuera del bucle FOR.
- Probar el VI para N=2, 3 y 4. Determinar visualmente el punto de -3dB y colocar las observaciones como comentarios.

4. Ecualización

Otro uso importante de los filtros es la ecualización de señales, para realzar o atenuar determinadas componentes de frecuencia. En este ejercicio vamos a realizar un ecualizador de 3 bandas para audio. Las bandas serán Bajos, Medios y Agudos y las realizaremos con el siguiente esquema:



- Crear un nuevo VI. Agregar 3 nodos de filtrado desde **Signal Processing → Filters → Chebyshev Filter**. Este tipo de filtro se denomina IIR (Respuesta

Infinita al Impulso) y está diseñado para imitar el comportamiento del filtro Chebyshev analógico.

- Configurar cada filtro con las siguientes constantes:
  1. Filtro 1
     - *filter type*: Bandpass
     - *high cut off freq: fh*: 3000Hz
     - *low cut off freq: fl*: 500Hz
     - *order*: 6
  2. Filtro 2
     - *filter type*: Lowpass
     - *low cut off freq: fl*: 500Hz
     - *order*: 3
  3. Filtro 3
     - *filter type*: Highpass
     - *low cut off freq: fl*: 3000Hz
     - *order*: 3

  En todos los casos conectar la entrada *ripple(dB)* a una constante de valor 0.5. Esta es la variación permitida en la ganancia en la banda de paso, medida en decibeles.
- En el dominio digital, no existe unidad de frecuencia, por lo tanto, siempre que se especifiquen valores en Hz, es necesario proveer la frecuencia de muestreo para obtener las correspondientes frecuencias digitales. Para obtener la frecuencia de muestreo haremos la inversa del período del waveform que usaremos de entrada. Colocar un nodo **Programming → Waveform → Get Waveform Components**, crear un control a su entrada llamada *Entrada*, seleccionar *dt* y con la salida calcular la frecuencia de muestreo. Conectar el valor resultante a las entradas *sampling freq: fs* de los nodos de filtrado.
- Conectar las entradas y salidas de los 3 filtros como el esquema, agregando desde el panel frontal los controles para la ganancia de cada banda, que deberán llamarse *Bajos*, *Medios* y *Agudos*. Luego de realizar las operaciones necesarias convertir la salida en un waveform utilizando el nodo **Programming → Waveform → Build Waveform**. Conectar la entrada *waveform* de este nodo al waveform de entrada. Esto copiará los demás parámetros de la forma de onda. A la salida del nodo crear un indicador llamado *Salida*.
- Colocar un nodo de comparación desde **Programming → Comparison → Not Equal?,** conectar una de las entradas a 0 y en la otra crear un control llamado *Índice de Bloque*. La salida conectarla a las entradas de init/cont de los nodos de filtrado. Esta conexión nos permitirá inicializar la memoria del filtro al comenzar a operar.
- Crear un subVI con lo realizado. Descargar el siguiente archivo y descomprimirlo:

  equ_test.zip

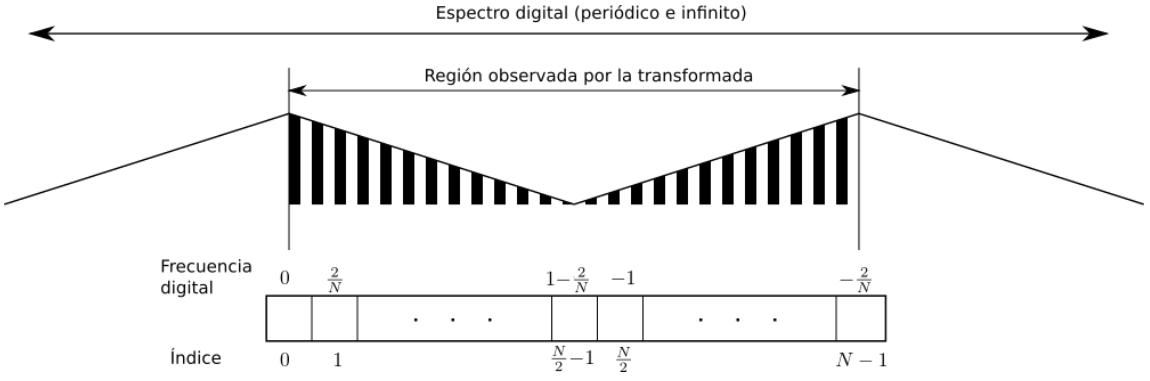
En el VI equ_test.VI, reemplazar los ecualizadores vacíos por el nuevo subVI y probar el funcionamiento del ecualizador.

5. Remoción de interferencia

La interferencia difiere del ruido en que en general se conocen detalles de la señal que facilitan su remoción. Un ejemplo concreto es la interferencia generada por un tono senoidal de frecuencia conocida, como por ejemplo los 50Hz de la red eléctrica o alguna de sus armónicas. En este caso, se puede eliminar la interferencia utilizando un filtro de tipo elimina banda o *notch*. De los diferentes tipos de filtros existentes vamos a utilizar un filtro de tipo FIR (Finite Impulse Response) . Este filtro que se implementa a partir de la convolución de la señal de entrada con una secuencia de coeficientes (kernel). La particularidad de este filtro es que sus coeficientes representan la respuesta al impulso del filtro, y por tratarse de una convolución, que es una operación lineal, esta respuesta contiene toda la información necesaria para caracterizar el comportamiento del filtro.
Para este ejemplo tenemos una señal de audio muestreada a 44100Hz contaminada con un tono senoidal de 1KHz

- Crear un nuevo VI. Agregar un nodo de diseño de filtro desde **Signal Processing → Filters → Advanced Filters → FIR Windowed Coefficients**. Conectar las entradas *high cutoff freq: fh*, *low cutoff freq: lf*, *taps* y *window* a controles llamados *Frecuencia de Corte Alto*, *Frecuencia de Corte Bajo*, *Cantidad de Taps* y *Ventana*, respectivamente. Conectar la entrada *filter type* a una constante y seleccionar *Bandstop*. Conectar la entrada *sampling freq: fs* a una constante de valor 44100 (Hz). Será la frecuencia de muestreo con la que trabajará nuestro filtro. La salida del nodo es un vector con los coeficientes del filtro (kernel).
- Establecer los siguientes valores defecto para los controles
    1. *Frecuencia de Corte Alto:* 1050
    2. *Frecuencia de Corte Bajo:* 950
    3. *Cantidad de Taps:* 3001
    4. *Ventana:* Rectangular
- Agregar un nodo de transformada de Fourier desde **Signal Processing → Transforms → FFT**. Conectar el vector de coeficientes a la entrada del nodo. La transformada de Fourier de los coeficientes nos dará la respuesta en frecuencia del sistema, pues los coeficientes son exactamente la respuesta al impulso.
- Conectar una constante a la entrada *size* y establecer el valor 16384. Este número representa la cantidad de puntos de frecuencia que calculará la transformada, y por ende nos dan la resolución que tendremos en frecuencia. A mayor cantidad de puntos, mayor resolución.

El vector de salida tendrá *size* (*N* a partir de aquí) elementos complejos y representa la transformada discreta de Fourier de la secuencia de entrada. Este vector puede relacionarse con el espectro de frecuencias digital de la siguiente manera:



- Teniendo en cuenta la descripción anterior, vamos a graficar el espectro de frecuencias en dB. Para señales reales como nuestros coeficientes, la salida es simétrica (simetría hermitiana), por lo que alcanzará con graficar los primeros *N/2* elementos. Para armar el gráfico vamos a expresar el eje de frecuencias en Hertz. Para ello armar un vector de *N/2* elementos de 0 a casi *fs/2* utilizando el nodo **Signal Processing** → **Signal Generation** → **Ramp Pattern**. Recordar que *fs* se definió como constante en 44100Hz.
- Con el nodo **Programming** → **Array** → **Array Subset** tomar las primeras *N/2* muestras del vector de la transformada y convertirlas en decibeles (tomando primero el valor absoluto de las muestras).
- Armar un cluster con el vector de frecuencias y con el vector de la transformada, utilizando el nodo **Programming** → **Cluster, Class & Variant** → **Bundle**. Ubicar en el panel frontal un visualizador gráfico de tipo **XY Graph** y conectar el cluster a la entrada. Al ejecutar el VI se debería visualizar la respuesta de tipo notch del filtro generado.
- Buscar y elegir la ventana (window) que nos dé la mayor atenuación en la banda de rechazo.
- Descargar el archivo *audio_play.zip*. El VI reproduce una secuencia de audio con interferencia de 1KHz. Reproducir para escuchar la interferencia, luego utilizando el VI de generación de filtro, conectar el vector de coeficientes siguiendo los comentarios en el diagrama en bloques. Con el filtro encendido se debería escuchar el audio sin la señal de interferencia.
  Link a archivo .zip:
  audio_play.zip

6. Correlación

En muchas aplicaciones es necesario detectar la presencia o no de una señal

determinada, por ejemplo el pulso de reflejo de una onda de radar o el tono correcto en la afinación de una guitarra. En estas aplicaciones la correlación es una función que nos da una indicación de similitud entre dos señales y la ubicación de la similitud.

Matemáticamente la función de correlación es muy similar a la convolución, la única diferencia es que no es necesario invertir el eje temporal de una de las secuencias. Más allá de este detalle, la diferencia principal el uso y la interpretación que se dá a la operación.

$$y[n] = x[n] \star q[n] = \sum_{i=0}^{N-1} x[i-n]q[i]$$

En este caso lo que se busca es comparar dos señales, siendo el valor de la correlación un indicador de similitud y el índice *n* de la secuencia de salida el indicador de posición.

En este ejercicio sólo evaluaremos la similitud, pero no la posición, por ello podemos obviar el índice n y calcular la correlación sólo para la posición 0, por lo que la ecuación se simplifica a

$$y[0] = \sum_{i=0}^{N-1} x[i]q[i]$$

- Crear un nuevo VI tomando de base el ejercicio 8 de la guía 1. Dejar únicamente lo relacionado a la lectura de la placa de sonido. Crear un control llamado *Longitud de Correlación*, y conectarlo a la entrada de *number of samples/ch* del nodo de configuración de entrada de audio y del nodo de lectura de audio. Establecer por defecto el valor de 5000.
- Agregar un nodo de indexado de arreglo desde **Programming → Array → Index Array** y conectar la constante 0 a la entrada *index* y la salida del nodo de lectura de audio a la entrada *array*. Esto nos permitirá quedarnos con el canal 0.
- Vamos a normalizar el bloque de salida de audio, de manera que la señal esté siempre entre -1 y 1. Para ello realizar los siguientes pasos:
  - Extraer la componente Y del waveform.
  - Pasar los elementos a valor absoluto mediante el nodo **Absolute Value**.
  - Obtener el máximo mediante el nodo **Array Max & Min** de la paleta **Array**.
  - Dividir el waveform del canal 0 por la salida *max value* del nodo anterior.
- Agregar dos nodos de generación senoidal desde **Signal Processing → Waveform Generation → Sine Waveform**. Agregar un control numérico llamado *Frecuencia* y conectarlo a las dos entradas *frequency*. Conectar la

entrada *phase* de uno solo de los dos nodos a la constante 90, de manera que mientras un nodo genera un seno, el otro genere un coseno.

- Crear un cluster mediante el nodo **Programming → Cluster, Class & Variant → Bundle**. Conectar el primer elemento al sampling rate de la placa de sonido, obtenido mediante el nodo **Programming → Cluster, Class & Variant → Unbundle by Name**. Conectar el segundo elemento al control *Longitud de Correlación*. Conectar el cluster resultante a la entrada *sampling info* de los nodos de generación senoidal.
- Correlacionar la señal senoidal con la señal normalizada, como se indica en la última ecuación de la introducción del ejercicio. Para ello multiplicar las señales y luego usar el nodo de suma de arreglo **Add Array Elements** de la paleta **Numeric**.
- Repetir el punto anterior para la señal cosenoidal.
- Normalizar ambas correlaciones dividiendo por el control *Longitud de Correlación*. En este punto tenemos dos indicadores de similitud, uno para el coseno y otro para el seno. Como la señal real es muy probablemente una combinación de ambos, los vamos a combinar utilizando la suma de los valores al cuadrado y tomando raíz cuadrada de dicha suma.
- A la salida de la raíz cuadrada conectar un indicador de tipo Meter, y cambiar el límite superior a 0.1.
- Con un generador de tonos (Disponibles online, o app para celular, en algunos casos puede usarse el VI del ejercicio 7 corriendo en paralelo), aplicar un tono de 440Hz y correr el VI, configurando la misma frecuencia en el control correspondiente. El indicador debería moverse al máximo. Cambiar la frecuencia del tono externo hasta que el indicador se mueva al mínimo. ¿Cual es la sensibilidad del correlador? Indicarlo como comentario.
- Opcional: Aumentar el tamaño de Longitud de Correlación, hasta que la sensibilidad sea menor a 0.5Hz.

7. Remuestreo

Quizás el tópico más importante en el procesamiento digital de señales es el muestreo, operación que permite pasar del dominio del tiempo al dominio de las muestras. El teorema del muestreo establece que si la frecuencia de muestreo es mayor al doble de la componente de mayor frecuencia presente en la señal analógica, entonces toda la información necesaria para representar la señal original está presente en la secuencia muestreada.

Matemáticamente:
Si x_a es una señal analógica, podemos obtener la secuencia x_d haciendo

$$x_d[n] = x_a(nT_s)$$

Donde T_s es el período de muestreo igual a $1/F_s$. Luego si se cumple el criterio de Nyquist:

$$F_s > 2F_{max} \equiv 2B$$

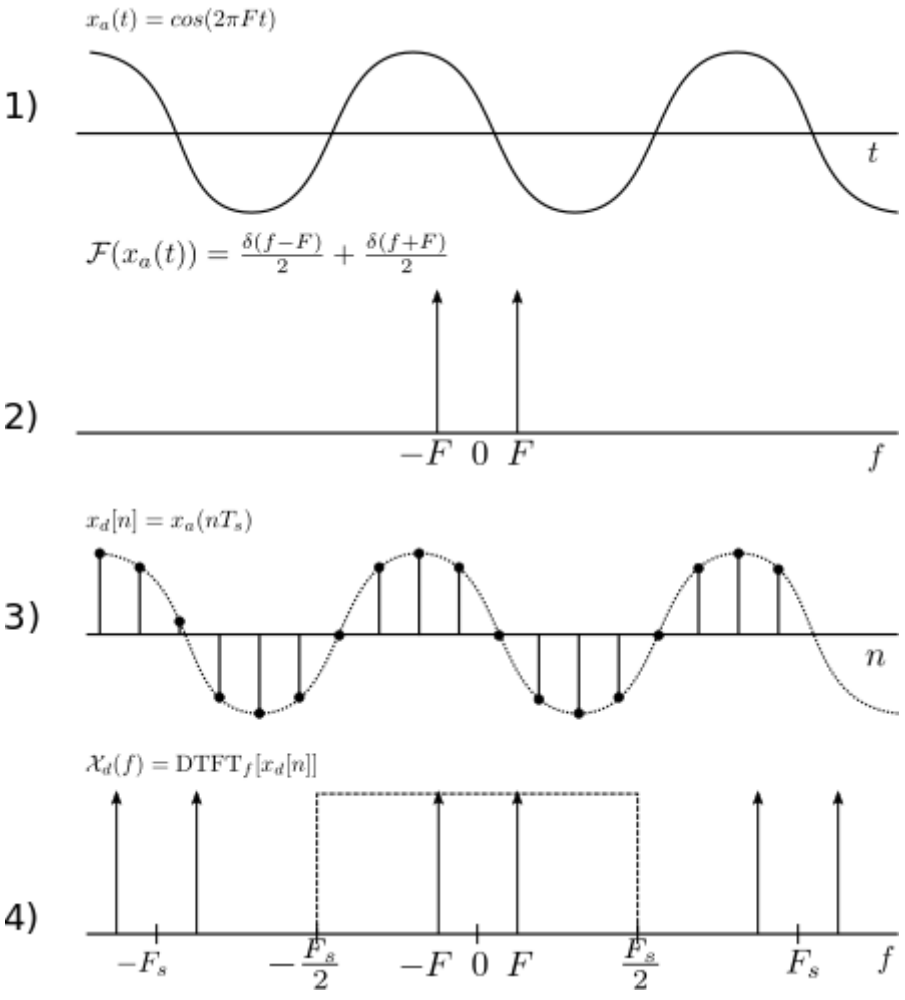
Entonces la señal puede reconstruirse a partir de sus muestras mediante

$$x_a(t) = \sum_{n=-\infty}^{\infty} x_d[n]\, g\left(t - \frac{n}{F_s}\right)$$

donde

$$g(t) = \frac{\sin 2\pi Bt}{2\pi Bt}$$

Podemos ilustrar lo que sucede en el tiempo y en frecuencia al muestrear una señal cosenoidal



1) Señal analógica cosenoidal de frecuencia F.
2) La transformada de Fourier de la señal anterior resulta en dos impulsos ubicados

en F y -F, con valor 0.5.
3) Señal muestreada con frecuencia de muestreo ~8 veces mayor a F.
4) DTFT de la secuencia muestreada. El espectro de una señal muestreada es periódico con período F_s. La línea de puntos representa la transformada de Fourier de la función de reconstrucción *g(t)*.

La función g(t) es conocida como si función *sinc* y nos permite reconstruir señales con componentes tan cercanas a $F_s/2$ como se quiera, pues su respuesta es la de un perfecto filtro pasabajos. Pero si la señal original tiene componentes con frecuencias mayores o iguales a F_s2, estas serán representadas con frecuencias menores y por ende la señal ya no podrá reconstruirse.

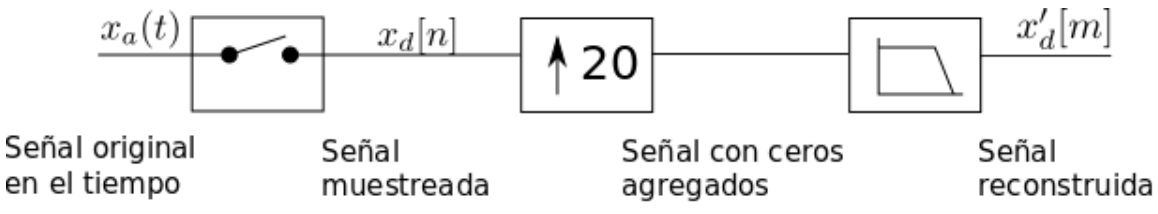
Una extensión del teorema de muestreo es que es posible a partir de una secuencia digital $x_d[n]$ obtener otra secuencia $x_d'[m]$ con más o menos muestras, siempre que se mantenga el criterio de Nyquist original. Esta operación de convertir una secuencia en otra con distinta cantidad de muestras se denomina *Remuestreo* o *Resampling*.

La operación de remuestreo tiene 3 etapas: Agregado de muestras extras, Filtrado y Remoción de muestras excedentes. Dependiendo de la relación entre la cantidad de muestras a la entrada y la salida, puede que la primera o última etapa no estén presentes.

- Descargar el siguiente archivo, descomprimirlo y abrir el VI llamado *Muestreo y Reconstrucción.VI*

  muestreo_y_reconstruccion.zip

- El VI implementa el siguiente esquema:



La señal analógica se muestrea desde una función senoidal, que podrá ser seno o coseno dependiendo de la posición del interruptor. Comenzando con los valores por defecto, correr el VI y observar los resultados. En la ventana *Muestreo Inicial*, veremos a la señal original en línea blanca y a las muestras como puntos rojos unidos por rectas. En la ventana *Tono y Aliasing vs Filtro* se observa la señal muestreada luego del agregado de ceros y antes del filtro de reconstrucción, y corresponde la porción del gráfico 4) que se encuentra entre

0 y F_s. Inicialmente el filtro de reconstrucción es un pasatodos porque solamente tiene 1 tap.

Por último la ventana *Reconstrucción* muestra nuevamente a la señal original, pero comparada con la señal reconstruida.

- Aumentar la cantidad de taps del filtro de reconstrucción hasta lograr que la señal reconstruida sea similar a la original (No es necesario que sea perfecta). Anotar la cantidad de taps en la sección de comentarios del panel frontal.
- Aumentar la frecuencia de la señal a 40Hz.
    1. ¿Qué sucede con la señal reconstruida?
    2. ¿Porqué cree que sucede esto? Anote las impresiones en la sección de comentarios.
- Cambie la frecuencia a 49Hz. Ajuste la cantidad de taps del filtro de reconstrucción hasta obtener una señal reconstruida similar a la original. (Tip: son muchos taps) Anotar la cantidad de taps como comentario.
- Cambie la frecuencia a 50Hz.
    1. ¿Qué sucede con la señal reconstruida?
    2. ¿Y si usamos una señal senoidal?
    3. ¿Cuál es la frecuencia máxima que podemos reconstruir con una frecuencia de muestreo de 100Hz?
    4. ¿Qué sucede si utilizo 51Hz? Anote las respuestas.
- Ahora vuelva la frecuencia a 49Hz y cambie la frecuencia de muestreo a 200Hz. Como hemos cambiado la frecuencia de muestreo, tendremos que ajustar el factor de sobremuestreo de manera que obtengamos la misma cantidad de muestras de salida. También cambie la frecuencia de corte del filtro de reconstrucción a 100Hz.
    1. ¿Con cuantos taps obtenemos una reconstrucción aceptable de la señal?
    2. ¿Qué podemos concluir de usar mayores frecuencias de muestreo? Anote las conclusiones

       Importante! Establezca el texto como valor por defecto para que los comentarios sean guardados.