Laboratorio de Procesado Digital de Señal - 3º GITT

Práctica 6: implementación de filtros LTI utilizando DFT

En esta práctica se va a filtrar una señal de audio utilizando la Transformada Discreta de Fourier (DFT) mediante el método de solape-almacenamiento visto en clase.

Se facilita al alumno un archivo de audio y los valores de los parámetros del filtro FIR con el que se van a trabajar más delante.

Para el desarrollo de esta práctica, el alumno necesita unos altavoces o auriculares.

Programe en Matlab los siguientes apartados, comentando el código.

Al finalizar la práctica, suba a la plataforma web de la asignatura (Moodle) un archivo PDF, en el que se responda a los apartados de la práctica y en el que se presenten las figuras que se piden, y un archivo .m, con el código de Matlab con el que se obtienen los resultados.

Diseño de un filtro FIR

En este bloque de apartados, el alumno analizará la señal de audio facilitada y diseñará un filtro FIR para eliminar un molesto tono que tiene la señal.

A partir de los parámetros facilitados al alumno, realice los siguientes apartados.

a) Lea el archivo de audio que le ha facilitado el profesor. La señal obtenida se denominará x[n]. Reproduzca esta señal de audio prestando mucha atención al volumen. Empiece con volumen bajo para evitar dañarse los oídos.

Como habrá podido comprobar, sobre la persona que habla hay un tono de sonido muy molesto, el cual se desea eliminar mediante un filtro FIR.

b) Indique los valores de la frecuencia de muestreo y de la frecuencia de corte que debe tener el filtro. Justifique de forma clara y contundente dichos valores.
c) Usando la herramienta `fdatool`, obtenga los coeficientes `b` de un filtro FIR **paso bajo** con las siguientes características, con el objetivo de atenuar notablemente dicho tono:
   - Tipo de respuesta: Lowpass
   - Método de diseño: FIR – Constrained Equiripple
   - Orden del filtro: *M*
   - Especificación de frecuencias:
     - Fs: *frecuencia de muestreo* (a especificar por el alumno)
     - Especificación: cutoff
     - Fc: *frecuencia de corte* (a especificar por el alumno)

- Especificación de magnitudes:
  - Unidades: dB
  - Apass: Apass
  - Astop: Astop

d) Represente la respuesta en frecuencia del filtro diseñado, y justifique que se cumple el objetivo solicitado.

Implementación de un filtro FIR utilizando DFT

Como bien sabe el alumno, la operación de filtrado puede implementarse mediante una convolución lineal. No obstante, la operación de convolución es computacionalmente compleja (tiene una complejidad de $O(N^2)$, siendo N la longitud de la señal a filtrar). Debido a que el filtrado es un proceso muy común en comunicaciones, sería deseable optimizar los recursos dedicados a esta operación. Tal y como se ha estudiado en clase, la Transformada Discreta de Fourier (DFT) puede ser utilizada para filtrar señales discretas con una salvedad: la DFT produce convoluciones circulares que, por regla general, no son equivalente a las convoluciones lineales. No obstante, una de los claros beneficios del uso de la DFT es su rápida capacidad de cálculo: el filtrado con DFTs supone una complejidad de $O(N \cdot \log_2(N))$, que es mejor (es decir, menos complejo) que $O(N^2)$ para valores de N bajos. De manera que, lo único que hay que hacer para acceder a las ventajas en la rapidez de cálculo es conseguir hacer la convolución lineal equivalente a la convolución circular. Una vez conseguido esto, tendremos una manera rápida de filtrar señales discretas. Tal y como veremos en la sesión de hoy, el algoritmo de solape y almacenamiento consigue igualar el resultado de la convolución circular con el de la convolución lineal mediante una determinada gestión de las muestras de entrada y salida.

En este bloque de apartados el alumno implementará el algoritmo del método de solape-almacenamiento visto en clase para aplicar el filtro diseñado en el bloque anterior sobre la señal de audio facilitada por el profesor. Puede consultar la documentación subida a Moodle sobre dicho algoritmo.

A partir de los resultados obtenidos en el bloque anterior, realice los siguientes apartados:

a) Implemente el algoritmo de solape-almacenamiento, con una longitud de $L = 500$ muestras para cada trozo de señal, y aplíquelo a la señal original x[n]. La señal resultante final se denominará y[n]. Reproduzca esta señal.
b) Incluya el código del algoritmo en este apartado del informe de la práctica, comentando las líneas más significativas.
c) Indique y justifique el valor del parámetro P.

Análisis de resultados

En esta última parte de la práctica se van a analizar los resultados del proceso anterior frente a los resultados que se obtendrían implementando el filtro con la función de Matlab.

Realice los siguientes apartados, a partir de los resultados de los bloques anteriores:

a) Aplique el filtro FIR diseñado, a la señal original x[n], con la función `filter` de Matlab. La señal resultante se denominará g[n].
b) Represente en el tiempo y superpuestas en una misma figura las señales y(t) y g(t).
c) Calcule el error cuadrático medio entre y[n] y g[n], e indique el valor obtenido.
d) Superponga el espectro en frecuencia de x[n], y[n] y g[n].
e) Exponga las conclusiones que obtiene al analizar los resultados obtenidos en los apartados anteriores. Justifique adecuadamente dichas conclusiones.

Funciones de Matlab

Consulte la ayuda de Matlab para conocer el significado de cada uno de los argumentos.

Miscelánea	
Borra todas las variables del Workspace	`clear`
Cierra todas las figuras abiertas	`close all`
Limpia la ventana de comandos	`clc`
Carga en el Workspace las variables almacenadas en el archivo **nombre_archivo.mat**	`load 'nombre_archivo.mat'`
Devuelve la longitud del vector **x**	`y = length(x);`
Devuelve el tamaño de **x** expresado en filas y columnas	`y = size(x);`
Redondea el valor **x** al número entero inferior más cercano	`y = floor(x);`
Devuelve una matriz de ceros de tamaño **a** filas y **b** columnas	`y = zeros(a, b);`
Muestra el mensaje de **texto** por pantalla	`disp('texto');`
Convierte en string (cadena de caracteres) el número **x**	`y = num2str(x);`
Representación	
Abre una nueva ventana para representar gráficamente	`figure;`
Representación en ejes x-y uniendo los puntos	`plot(ejex, ejey, '-.');`
Congela la figura activa para poder superponer más representaciones	`hold on;`
Activar la rejilla de la representación	`grid on;`
Poner título a la representación	`title('Texto');`
Poner etiqueta en eje x. La función **ylabel** hace lo mismo en el eje **y**	`xlabel('Texto');`
Muestra una leyenda en la figura, donde se muestran los textos indicados	`legend('Texto1', 'Texto2', ...);`

Filtrado	
Aplica el filtro definido por sus coeficientes **a** y **b** a la señal **x**	`y = filter(b, a, x);`
Audio	
Lee el archivo de audio **file**	`[y, fs] = audioread('file');`
Reproduce la señal de audio **y** a una frecuencia de muestreo **fs**	`sound(y, fs);`
Vectores	
Generar el vector de valores **v** entre **ini** y **fin** equiespaciados **delta**	`v = ini : delta : fin;`
Generar el vector **v** de **N** valores equiespaciados entre **ini** y **fin** (inclusive)	`v = linspace(ini, fin, N);`
Genera el vector **v** con los elementos del vector **x** desde el **a** hasta el **b** (inclusive)	`v = x(a:b);`
Matemáticas	
Calcula el valor absoluto del número **x**, o el módulo de éste si es un número complejo	`y = abs(x);`
Calcula la transformada de Fourier del vector **x**	`X = fft(x, length(x)) / length(x);`
Calcula la transformada inversa de Fourier del vector **X**	`x = ifft(X, length(X)) * length(X);`
Trasposición del espectro en frecuencias **X**	`X = fftshift(abs(X));`
Calcular el sumatorio de todos los valores del vector **x**	`y = sum(x);`
Calcula el logaritmo decimal de **x**	`y = log10(x);`
Calcula el logaritmo neperiano de **x**	`y = log(x);`
Respuestas	
Calcula **num** puntos de la respuesta en frecuencia **H** de un filtro caracterizado por los coefs. **a** y **b**	`H = freqz(b, a, num);`