IE0527 – Ingeniería de Comunicaciones

Taller de simulación: Transformada de Fourier

Introducción

El estudio de la transformada de Fourier es fundamental en muchas aplicaciones que involucran procesamiento de señales, entre ellas la ingeniería de comunicaciones. La visualización en el dominio de la frecuencia de las señales involucradas en un sistema de comunicaciones permite estudiar mejor sus características y diseñar nuevas técnicas de procesamiento.

Mediante software, la transformada de Fourier de una señal se puede aproximar mediante la *transformada rápida de Fourier* (FFT, por sus siglas en inglés) de un vector de muestras de la señal. Además, a partir de este vector de muestras se puede obtener una aproximación de la *densidad espectral de potencia* (PSD, por sus siglas en inglés) de la señal. Adicionalmente, existe software que asiste en diseñar filtros y graficar señales tanto en el dominio del tiempo como de la frecuencia, facilitando su estudio.

Descripción

Este taller busca desarrollar destrezas de software para trabajar con la transformada de Fourier y ejemplificar algunos conceptos relacionados con el estudio de señales en el tiempo y la frecuencia. Así, el taller consiste en utilizar una o vairas herramientas para estudiar la transformada (rápida) de Fourier, la densidad espectral de potencia y el filtrado de señales.

El taller está diseñado para utilizar *Matlab* como herramienta principal, pero es posible desarrollarlo con herramientas libres como librerías de procesamiento digital de señales y comunicaciones para *C*, *C++* o *Python*.

Consideraciones generales

- El taller debe realizarse en grupos de 2 personas.
- Se promueven las consultas al profesor y el asistente; sin embargo, deben ser puntuales y claras.
- Los resultados del taller deben ser presentados en un reporte breve *compuesto* en formato PDF.
- Todo el código fuente utilizado para desarrollar el taller (los archivos `.m`, en el caso de Matlab) debe adjuntarse al reporte.
- Se castigará severamente cualquier intento de copia en el código o los reportes.

1. Estudio preliminar

Considere la señal $x(t) = A_1 \cos(2\pi\, f_1\, t) + A_2 \sin(2\pi\, f_2\, t) + A_3 \cos(2\pi\, f_3\, t)$.

Trabajo previo

1. Escoja valores para A_i y f_i. Para mejorar la calidad de los resultados, se recomienda que f_2 y f_3 tengan un común divisor con f_1. Los valores propuestos se muestran en la siguiente tabla:

variable	valor
A_1	0.7
A_2	0.3
A_3	0.4
f_1	440 Hz
f_2	$3 \cdot f_1$
f_3	$5 \cdot f_1$

2. Obtenga $x(t)$ utilizando los valores escogidos para A_i y f_i.
3. Determine la potencia promedio de $x(t)$, P_x.

Trabajo de simulación

1. Escoja una frecuencia de muestreo `Fs` y una longitud de vector `N` apropiadas. Para obtener resultados agradables, se recomienda que `Fs` esté relacionada con el divisor común de `f1`, `f2` y `f3`. Se propone `Fs = 32*f1` y `N = 256`.
2. Defina un vector para el tiempo t, llamado `t`, que comprenda desde 0 hasta $\frac{(N-1)}{F_\mathrm{s}}$ en intervalos de $\frac{1}{F_\mathrm{s}}$.
3. Genere el vector de $x(t)$, `x`.
4. Grafique el vector `x` en función del tiempo.
5. Determine la potencia promedio de `x`, `Px`, y compárela con P_x.

2. Transformada de Fourier

La *transformada rápida de Fourier* (FFT, por sus siglas en inglés) es un algoritmo eficiente que permite obtener, a grandes rasgos, una aproximación de la transformada de Fourier de una señal $x(t)$, $X(f)$, a partir de un vector de muestras `x` de longitud N con frecuencia de muestreo F_s. El vector resultante `X`, adecuadamente alineado y escalado, contiene valores aproximados de $X(f)$ para valores discretos de frecuencia que van desde $-\frac{F_\mathrm{s}}{2}$ hasta $\frac{F_\mathrm{s}}{2} - \frac{F_\mathrm{s}}{N}$, en intervalos de $\frac{F_\mathrm{s}}{N}$.

⚠ En Matlab, la función `fft()` se utiliza junto con la función `fftshift()` para obtener la transformada rápida de Fourier de `x`, `X`, de la siguiente forma:

```
X = 1/N*fftshift(fft(x,N));
```

Trabajo previo

1. Obtenga la transformada de Fourier 'teórica' de $x(t)$, $X(f)$.
2. Grafique las partes real e imaginaria de $X(f)$.
3. Grafique el espectro de Fourier de $x(t)$ (la magnitud de $X(f)$).

Trabajo de simulación

1. Obtenga `X`, la transformada rápida de Fourier de `x`.

2. Defina un vector de frecuencias `f` apropiado.
3. Grafique las partes real e imaginaria de `X`.[1]
4. Grafique el espectro de Fourier de `x` (la magnitud de `X`).[2]
5. Compare los resultados obtenidos para `X` con $X(f)$.

3. Densidad espectral de potencia

La *densidad espectral de potencia* (PSD, por sus siglas en inglés) de una señal es una forma popular de representar cómo se distribuye su potencia sobre su espectro. Existen diferentes formas de estimar la PSD de una señal $x(t)$ a partir de un vector de muestras `x` de longitud N con frecuencia de muestreo F_s, que incluyen su *periodograma* (la transformada de Fourier del vector de autocorrelación) y el cuadrado de la magnitud de su FFT. El vector resultante de estas aproximaciones normalmente se presenta en unidades de dBW/Hz para valores discretos de frecuencia que van desde 0 hasta $\frac{F_\mathrm{s}}{2} - \frac{F_\mathrm{s}}{N}$ en intervalos de $\frac{F_\mathrm{s}}{N}$.

⚠ En Matlab, la función `periodogram()` se puede utilizar para estimar la PSD de un vector `x`, `Sxx`, de la siguiente forma:

```
Sxx = periodogram(x,rectwin(N),N,Fs,'onesided');
Sxx = Fs/N*Sxx(1:N/2)';
```

aunque también es posible estimarla a partir de su FFT, `X`, de la siguiente forma:

```
X = 1/N*fftshift(fft(x,N));
Sxx = 2*abs(X(N/2+1:N)).^2;
```

Además, `Sxx` se puede expresar en dBW/Hz de la siguiente forma:

```
SxxdB = 10*log10(Sxx);
```

Trabajo previo

1. Intente obtener la función de densidad espectral de potencia 'teórica' de $x(t)$, $S_{xx}(f)$[3].
2. Intente graficar $S_{xx}(f)$ en dBW/Hz.

Trabajo de simulación

1. Obtenga `Sxx`, la estimación de la PSD de `x` utilizando el periodograma.
2. Grafique `Sxx` en dBW/Hz.
3. Compare los resultados obtenidos para `Sxx` con S_{xx}, de ser posible.
4. Determine `Px` a partir de `Sxx` y compárela con P_x.
5. Obtenga `Sxx1`, la estimación de la PSD de `x` a partir de su FFT `X`.
6. Grafique `Sxx1` en dBW/Hz.
7. Compare los resultados obtenidos para `Sxx1` con `Sxx`.
8. Determine `Px` a partir de `Sxx1` y compárela con P_x.

[1] ⚠ Considere las funciones `real()` e `imag()`.
[2] ⚠ Considere las funciones `abs()` y `stem()`.
[3] Considere que el resultado puede llegar a ser complicado.

4. Filtrado

El filtrado de una señal analógica normalmente se modela mediante la convolución de la señal con la respuesta al impulso del filtro ($y(t) = x(t) * h(t)$). Sin embargo, cuando se utilizan vectores de muestras `x`, `h` y `y` para $x(t)$, $h(t)$ y $y(t)$, respectivamente, el vector de salida del filtro se puede representar de distintas formas si se consideran diferentes factores, como la cantidad de muestras requeridas y las variaciones transitorias introducidas por el filtro.

Además, la teoría sobre el diseño de filtros es extensa y diversa, por lo que es difícil definir un procedimiento de diseño único para obtener un vector `h` adecuado. En general, existen dos grandes categorías de filtros digitales: de respuesta al impulso finita (FIR, por sus siglas en inglés) y de respuesta al impuslo infinita (IIR, por sus siglas en inglés), y las especificaciones, consideraciones, procedimientos de diseño y resultados pueden variar según el tipo de filtro.

⛰ En Matlab, la función `designfilt()` permite diseñar un filtro digital, ya sea FIR o IIR, y almacenarlo en un objeto `D` de tipo `digitalFilter`, de la siguiente forma:

```
D = designfilt(<tipo>,<especificaciones>);
```

Por ejemplo,

```
D = designfilt('lowpassfir','FilterOrder',16,'HalfPowerFrequency',4000,'SampleRate',44100);
```

crea un objeto `D` con el diseño de un filtro FIR pasa-bajos de orden 16, con frecuencia de media potencia de 4 kHz para vectores de muestras con frecuencia de muestreo de 44.1 kHz.

Correspondientemente, la función `filter()` se puede utilizar para, a partir del filtro `D` y el vector de muestras de entrada `x`, obtener el vector de muestras de salida `y` de la siguiente forma:

```
y = filter(D,x);
```

Trabajo previo

1. Explique qué sucede con $|X(f)|$ cuando $x(t)$ pasa por un filtro pasa-bajos con frecuencia de media potencia $f_\mathrm{c} = f_2$.

Trabajo de simulación

1. Defina un filtro FIR pasa-bajos de orden 32 con frecuencia de media potencia de $f_\mathrm{c} = f_2$.
2. Grafique la respuesta en frecuencia del filtro diseñado, $|H(f)|$, para el vector de frecuencias utilizado para calcular la FFT. Considere la función `freqz()`.
3. Obtenga `y`, el resultado de filtrar `x` con el filtro diseñado.[4]
4. Obtenga `Y`, la FFT de `y`.
5. Grafique el espectro de magnitud de `y`.
6. Compare los resultados obtenidos para `y` y `Y` con `x` y `X`.

[4] ⛰ Considere la función `filtfilt()`.

Reto: Filtrado de una señal de audio

Considere una señal de audio $w(t)$ representada mediante un archivo `.wav` (de unos 3 segundos, aproximadamente).

Trabajo de simulación

1. Obtenga el vector de muestras *mono*[5] de $w(t)$, `w`, y su frecuencia de muestreo, `Fs`, así como su longitud `N`. Considere que si el vector `w` es muy grande, es posible que algunas operaciones tomen un tiempo considerable. [6]
2. Obtenga `Sww`, la estimación de la PSD de `w` y grafíquela en dBW/Hz.
3. Comente sobre el ancho de banda de `w`.
4. Defina un filtro IIR pasa-bajos de orden 8 con frecuencia de media potencia de $f_\mathrm{c} = 2000$ Hz.
5. Obtenga `v`, el resultado de filtrar `w` con el filtro diseñado.
6. Obtenga `Svv`, la estimación de la PSD de `v` y grafíquela en dBW/Hz.
7. Compare los resultados obtenidos para `Svv` con `Sww`.
8. Escriba un nuevo archivo `.wav` con $v(t)$, la señal formada a partir del vector de muestras `v`. [7]
9. Escuche el archivo generado y comente sobre el efecto del filtro en $w(t)$.

[5] En aplicaciones de audio *estéreo*, normalmente se utilizan dos canales: izquierdo (L) y derecho (R). Es posible que el vector de muestras `w` presente dos series de muestras, correspondientes a estos canales. Por convención, en procesamiento de audio *mono* normalmente se utiliza el promedio de ambos canales: $\frac{L+R}{2}$.

[6] ⛰ Considere la función `audioread()`.

[7] ⛰ Considere la función `audiowrite()`.