eman ta zabal zazu

UPV, Facultad de Informática

Dpto. de Arquitectura y Tecnología de Computadores

PROCESADO DIGITAL DE SEÑAL

PROYECTO ESPECÍFICO - LABORATORIO 4 (FSo)

ANALISIS FRECUENCIAL - SONIDO

OBJETIVOS

- Obtener la DTFT de diferentes señales (finitas, infinitas, periódicas, no periódicas) y comprender el problema del enventanado.
- Comprender el uso de la DFT y su relación con la DTFT
- Obtener la DFT de fragmentos de señales de longitud infinita y entender la relación con las series de Fourier.
- Entender qué es un espectrograma.

APLICACIÓN DE LA DTFT A SEÑALES

La transformada de Fourier en tiempo discreto (DTFT) de una señal x(n) es:

$$X(F) = \sum_{n=-\infty}^{+\infty} x(n) e^{-j2\pi Fn}$$

En MATLAB, como en cualquier sistema digital, no puede calcularse, ya que es una función de una variable continua F y en su cálculo intervienen infinitos puntos de la señal $x(n)$. Lo único que podemos hacer es aplicarla a señales de longitud finita y calcularla en un número finito de puntos.

En la carpeta de trabajo encontrarás el fichero **`dtft.m`** que calcula puntos de la DTFT aplicando directamente esta definición. **Edítalo** y trata de comprender cómo funciona. En MATLAB no existe una función análoga porque en la práctica lo que se suele calcular es la DFT que estudiaremos a continuación. Por ejemplo, para visualizar la DTFT de la señal $x(n) = \{0,0,0,1,1,1,1,1,1,0\}$ basta hacer:

```
x = [1 1 1 1 1 1];
dtft(x, 3, 100);
```

(se indican los valores no nulos de la señal x)
A continuación, utiliza esta función para analizar la DTFT de las siguientes señales y rellena la hoja de respuestas:

$x_1(n) = \delta(n)$ $\quad -20 \le n \le 19$

¿Curioso no?

$x_2(n) = \sin\left(2\pi \frac{1}{16} n\right) \quad 0 \le n \le 99$

Usa nF=1000. Si la señal fuera de longitud infinita, el espectro tendría dos impulsos infinitos (en F=1/16 y F=15/16), pero al utilizar un fragmento, se ve claramente el problema del enventanado.

$x_3(n) = x_2(n) w(n)$

Siendo $w(n)$ una ventana de Hanning de longitud 100. Usa la función **`hanning(100)`** de MATLAB.

$x_4(n) = 3 - resto\left(\frac{-n}{7}\right) \quad 0 \le n < 99$

Usa la función **`rem`** para calcular el resto. Aparece el espectro típico de una señal periódica, apreciándose perfectamente la frecuencia fundamental y los armónicos.

$$x_5(n) = \begin{cases} \frac{n}{64}\sin\left(\frac{\pi}{16} n\right) & 0 \le n \le 63 \\ \left(2 - \frac{n}{64}\right)\sin\left(\frac{\pi}{4} n\right) & 64 \le n \le 128 \end{cases}$$

P4 1 dtft x4 x5.m

USO DE LA DFT Y SU INVERSA IDFT

En un sistema digital la operación que se realiza normalmente es la DFT que se define mediante:

$$X(k) \equiv X\left(\frac{k}{N}\right) = \sum_{n=0}^{N-1} x(n) e^{-j\frac{2\pi k}{N}n} \qquad k = 0,1,\ldots,N-1$$

Se aplica a señales de longitud finita (N puntos) y se evalúa en N valores de F distribuidos en el rango (0,1). El cálculo supone que el primer elemento del vector x corresponde a n=0, con lo que si no es así, estamos calculando la DTFT de la señal desplazada en el tiempo, pero en la mayoría de los casos esto no tiene ninguna importancia.

La función MATLAB utilizada para calcular la DFT es **`fft`**, cuya sintaxis es la siguiente:

```
X = fft(x);
```

donde **`X`** es el vector obtenido al aplicar la DFT al vector **`x`**. `X` y `x` tienen la misma longitud. El resultado será el mismo que si utilizásemos la función usada en el primer apartado de la siguiente manera:

```
X = dtft(x, 0, length(x));
```

Sin embargo, la función `fft` no emplea la definición de la DFT en el cálculo, sino que intenta emplear algoritmos rápidos (FFT). Estos algoritmos se basan en trocear la señal en fragmentos de forma iterativa, y por ello, la posibilidad de emplearlos depende de la longitud de `x`, en concreto, de su descomposición en números primos. La velocidad de cálculo es máxima si la longitud es potencia de 2.

La función **`fft`** admite un segundo parámetro:

```
X = fft(x,n);
```

En ese caso se fuerza a `x` a tener `n` elementos (añadiéndole ceros si `n` es mayor que la longitud de `x` truncándolo en caso contrario). Sólo tiene sentido usar `n` mayor que `length(x)`. Esta corrección se hace a veces para que `n` sea potencia de 2 y conseguir así que el cálculo sea más rápido. En versiones anteriores de MATLAB, esta corrección podía ser importante, porque sólo se empleaba un algoritmo rápido en este caso (potencia de 2). A partir de la versión 6, se ha mejorado la eficiencia también en los demás casos.

Para probar esta función, puedes aplicarla a alguno de los ejemplos del apartado anterior y comprobar que se obtiene el mismo resultado.

La operación inversa de la DFT es la IDFT y se calcula mediante la fórmula siguiente:

$$x(n) = \frac{1}{N}\sum_{k=0}^{N-1} X(k) e^{j\frac{2\pi k}{N}n} \qquad n = 0,1,\ldots,N-1$$

implementada en MATLAB mediante la función **`ifft`**. Esta operación nos permite reconstruir la señal original a partir de su espectro y puede recordar a las series de Fourier para señales periódicas. En realidad los valores de la DFT son salvo un factor 1/N los coeficientes c_K de la serie de Fourier de la señal $x_P(n)$ obtenida extendiendo periódicamente el vector x.

$$c_k = \frac{1}{N}\sum_{n=0}^{N-1} x_P(n) e^{-j\frac{2\pi k}{N}n} = \frac{1}{N}\sum_{n=0}^{N-1} x(n) e^{-j\frac{2\pi k}{N}n} = \frac{1}{N}X(k) \quad k = 0,1,\ldots,N-1$$

EJEMPLOS DE APLICACIÓN A SEÑALES DE SONIDO

La transformada DFT es una herramienta útil para analizar señales de sonido, ya que nos muestra su composición frecuencial que es lo que fundamentalmente percibe nuestro oído.

Ejemplo práctico 1:

Vamos a probar el uso de esta función sobre una señal de sonido ya utilizada en prácticas anteriores. La melodía al piano del fichero **`melodia.wav`** (82528 muestras)

```
[melo, fs] = audioread('melodia.wav');
soundsc(melo, fs)
subplot(2,1,1), plot(melo)
tic, MELO = fft(melo); toc            %tic y toc para medir el tiempo de cálculo
subplot(2,1,2), plot(abs(MELO))
```

Mostramos sólo el espectro de magnitud porque la información de fase no es importante. El cálculo de la DFT es costoso y con señales largas puede llevar bastante tiempo. Como ya se ha mencionado, el tiempo de cálculo mejora si ampliamos el vector con ceros hasta tener una potencia de 2 (de 82528 a 131072=2^{17}). En versiones anteriores de MATLAB la diferencia era mucho mayor:

```
tic, MELO = fft(melo,2^17); toc, plot(abs(MELO))
```

Como la señal se ha obtenido digitalizando una señal analógica con frecuencia de muestreo f_S(muestras/sg) para poder interpretar el espectro, conviene tener en cuenta que se puede calcular la frecuencia f en el dominio de t (Hz o ciclos/sg) a partir de la frecuencia F en el dominio de n (ciclos/muestra) mediante la expresión:

$$F = \frac{f}{f_s} \qquad f = \frac{F}{T_S} = Ff_S = \frac{k}{N} f_S$$

De todos modos, veréis que aplicar la DFT a la señal entera no tiene mucho sentido porque no vemos la variación a lo largo del tiempo de la composición frecuencial de la señal. Lo que se suele hacer es calcular el espectro de **fragmentos** de la señal a lo largo del tiempo. Por ejemplo:

```
subplot(2,1,1),plot(melo(3001:3512))
FRAG = fft(melo(3001:3512));
f=(0:511)/512*fs;
subplot(2,1,2),plot(f,abs(FRAG))
```

o todavía mejor, usando una ventana no cuadrada (por ejemplo: Hanning):

```
FRAG2 = fft(melo(3001:3512).*hanning(512));
plot(f,abs(FRAG2))
```

Vemos claramente que entre las muestras 3001 y 3512 (512 / f_S = 46 ms) la señal es cuasiperiódica y podríamos estimar la frecuencia fundamental fácilmente a partir del espectro. En el fichero **`fftevol.m`** se implementa una función que visualiza la evolución del espectro de la señal a lo largo del tiempo. **Edítalo** para ver cómo funciona y **pruébalo** con la melodía:

```
fftevol(melo,1024,1024,fs)
```

En la práctica, la evolución en el tiempo del espectro de una señal se suele mostrar mediante un gráfico denominado **espectrograma** (Short-Time Fourier Transform, STFT). En él, los espectros obtenidos a lo largo del tiempo se agrupan por columnas formando una matriz que se visualiza como si fuera una imagen, con la frecuencia en el eje vertical y el tiempo en el horizontal. La función correspondiente en MATLAB se llama **`spectrogram`**. Por ejemplo:

```
[SPEC, f, t] = spectrogram(melo,512,256,512,fs,'yaxis');
               %spectrogram (x,window,noverlap,nfft,fs)
```

genera una matriz con los valores del espectrograma correspondiente a la señal **`melo`**. Utiliza, para ello, fragmentos de 512 puntos (`nfft`) con una ventana de Hanning de 512 puntos (`window`), desplazando en cada paso el fragmento de forma que se solape con el anterior en 256 puntos (`noverlap`). La frecuencia

de muestreo permite a la función calcular las frecuencias en Hz y los tiempos asociados, valores que se devuelven en los vectores `f` y `t`.

Para visualizar el resultado basta utilizar la función `spectrogram` sin argumentos de salida. Así se muestra una imagen en la que cada pixel corresponde a un elemento del espectrograma, asignándole un color que se obtiene de comparar su valor con los valores máximo y mínimo de la matriz. Se aplica una paleta de colores por defecto (para lograr una definición mayor se usan la función logaritmo de los valores absolutos del espectro).

Ejemplo práctico 2:

Prueba ahora a obtener el espectrograma de una señal de voz. La encontrarás en el fichero **`e101.wav`**. Se puede observar que:

- La señal de voz formada por una serie de fonemas encadenados tiene un espectro complejo y muy cambiante según los fonemas que se pronuncien.
- En ciertos intervalos se aprecian claramente líneas horizontales que indican la presencia de vibraciones periódicas. Se trata de los fonemas sonoros, es decir, en los que intervienen las cuerdas vocales. Por ejemplo sucede así en todas las vocales.
- Las vocales se pueden diferenciar entre sí por los formantes, que son bandas de frecuencia en las que los armónicos están más marcados; cada vocal tiene sus formantes característicos.
- En algunos puntos aparece ausencia de señal. Son las consonantes oclusivas (p, t, k) que se producen cerrando el tracto bucal en algún punto durante un instante.
- En otros puntos se puede apreciar también la presencia de fricativas (s, z, f) que contienen ruido de altas frecuencias producido al pasar el aire por algún estrechamiento del tracto vocal.

Para ver un poco mejor las diferencias entre vocales, visualiza el espectrograma del sonido grabado en el fichero **`aeiou.wav`**.

Ejemplo práctico 3:

Los tonos DTMF son los sonidos usados en telefonía para la marcación por tonos. En la actualidad prácticamente ha sustituido a la antigua marcación por pulsos. A cada tecla se le hace corresponder un tono DTMF obtenido sumando dos sinusoidales de dos frecuencias determinadas, según la siguiente tabla:

	1209 Hz	1336 Hz	1477 Hz	1633 Hz
697 Hz	**1**	**2**	**3**	**A**
770 Hz	**4**	**5**	**6**	**B**
852 Hz	**7**	**8**	**9**	**C**
941 Hz	*****	**0**	**#**	**D**

Por ejemplo, si se pulsa el 5 se combinan dos señales de frecuencias 770 Hz y 1336 Hz. En los teléfonos la última columna no se utiliza. El estándar especifica que el sonido debe durar entre 65 y 105 ms y, entre tono y tono, debe haber como mínimo un silencio de 200 ms. Se admite un error de +/- 1.5% en las frecuencias.

La generación de estos tonos es muy sencilla, puedes editar la función **`gendtmf.m`** para ver una manera sencilla de hacerlo con MATLAB:

```
gendtmf('495123706');
```

Por otro lado, la DFT nos permite analizar fácilmente los tonos utilizados. Para comprobarlo dibuja el espectrograma de la señal guardada en el fichero **`numero.wav`** y trata de descubrir el número de teléfono al que corresponde.

P4_2_numero.m

Ejemplo práctico 4: **Filtrado de señales**

a) La transformada de Fourier también puede usarse para eliminar ciertas componentes frecuenciales de una señal. Así en el fichero **`frase.wav`** se ha almacenado una señal de voz a la que se ha superpuesto un pitido. Para eliminarlo se os propone diseñar una función de filtrado: **`filtrbf.m`** que anula del espectro el rango de frecuencias comprendido entre finf y fsup (en Hz). Para ello, calcula la DFT de la señal original, anula las componentes frecuenciales correspondientes y obtén la IDFT para restituir la señal original pero filtrada. En la hoja de respuestas se propone parte del contenido de dicha función. Debéis completarla y utilizarla.

b) Vamos a usar esta misma función para eliminar intervalos o bandas de frecuencia de la propia señal. En particular, sea la señal de voz (correspondiente a una frase) almacenada en el fichero '**`e101.wav`**'. Genera usando la función de filtrado **`filtrbf.m`** las tres señales siguientes en las que se eliminan las componentes frecuenciales necesarias para obtenerlas:

   a. Sb, señal de baja frecuencia: sólo contiene componentes de frecuencia menor que 500 Hz.

   b. Sm, señal de de frecuencias medias: con componentes de frecuencia entre 500Hz y 2000Hz.

   c. Sa, señal de alta frecuencia: con componentes de frecuencia mayor que 2000 Hz.

   Escúchalas por separado y sumadas ¿Qué ocurre?

P4_3_filtrado.m