ELO 314 - Procesamiento Digital de Señales
Lab. 4 - Guía complementaria:
Algoritmo de Goertzel

Preparado por

Dr. Gonzalo Carrasco, e-mail: Gonzalo.Carrasco@usm.cl

INTRODUCCIÓN

Esta guía permite complementar la herramienta de análisis Transformada de Fourier Discreta (DFT) que se presenta, analiza y experimenta en la guía Lab.4 Parte I. La DFT es una herramienta que permite obtener un vector de datos de variable compleja, cuyos elementos llamamos *bins*, y que corresponden a una componente en frecuencia de la sucesión de muestras temporales de una señal real o de variable compleja.

En contraste a al algoritmo Transformada de Fourier Rápida con implementación Radix-2 que se desarrolla en la parte I del laboratorio, que entrega el vector de bins en frecuencia a partir del vector de muestras temporales con una menor demanda computacional que implementar la sumatoria doble de la DFT, o usar la matriz de transformación de la DFT, en esta guía se experimentará implementando un algoritmo recursivo que permite obtener solo un bin en frecuencia a partir de las muestras temporales.

La ventaja del algoritmo o filtro de Goertzel es que para señales temporales reales, su implementación requiere mayormente de aritmética real en contraposición a implementar la suma compleja para un bin k directamente de la expresión matemática de la DFT. Debido a su simpleza, es un algoritmo que se puede implementar en procesadores o microcontroladores de baja capadidad de cómputo, y se ha usado durante décadas como método para decodificar las señales DTMF en sistemas de telefonía.

ALGORITMO DE GOERTZEL

A partir de la DFT se busca obtener el valor complejo del bin k, considerando que $W = e^{\frac{-j\,2\,\pi}{N}}$

$$X_k = \sum_{n=0}^{N-1} x[n]\ e^{-j2\pi kn/N} \tag{1}$$

$$X_k = \sum_{n=0}^{N-1} x[n]\ \left(W^k\right)^n \tag{2}$$

$$X_k = \sum_{n=0}^{N-1} \left(W^{-k}\right)^{(N-n)}\ x[n] \tag{3}$$

$$X_k = \left(\ldots\left(\left(\left(W^{-k}\,x[0]+x[1]\right)W^{-k}+x[2]\right)W^{-k}+x[3]\right)\cdots+x[N-1]\right)W^{-k} \tag{4}$$

Una evaluación del esta forma anidada permite reconocer un estructura recursiva:

$$y[-1] = 0 \tag{5}$$

$$y[0] = W^{-k}\,y[-1]+x[0] \tag{6}$$

$$y[1] = W^{-k}\,y[0]+x[1] \tag{7}$$

$$\ldots \tag{8}$$

$$y[N-1] = W^{-k}\,y[N-2]+x[N-1] \tag{9}$$

$$y[N] = W^{-k}\,y[N-1] \tag{10}$$

Donde el último termino es el resultado de la expresión 1, y tiene la forma recursiva:

$$y[n] = W^{-k}\,y[n-1]+x[n] \tag{11}$$

donde $n = 0, \ldots, N-1$. Se puede obtener la función de transferencia discreta para esta recursión:

$$H(z) = \frac{1}{1 - W^{-k}\, z^{-1}} \tag{12}$$

Debido a que esta estructura de filtro tiene parámetros complejos (W^{-k}) hace que en la práctica el posicionamiento del polo en el plano Z sea sensible a errores numéricos de redondeo y por cierto que habría que trabajar con aritmética compleja (equivalente en la práctica a operar con vectores de dos elementos reales), la implementación se realiza considerando la estructura siguiente:

$$H(z) = \frac{1 - W^{k}\, z^{-1}}{(1 - W^{-k}\, z^{-1})\,(1 - W^{k}\, z^{-1})} \tag{13}$$

$$H(z) = \frac{1 - e^{\frac{-j\,2\,\pi\,k}{N}}\, z^{-1}}{1 - 2\,\cos(\omega_0)\, z^{-1} + z^{-2}} \tag{14}$$

donde $\omega_0 = 2\,\pi\,k/N$ es la frecuencia normalizada equivalente al bin k de frecuencias a analizar. Notar que el filtro (12) por ser de estructura IIR se puede ejecutar en tiempo real muestra a muestra recibida, sin embargo solo la salida $y[N-1]$ es de interés (al procesar N muestras) y los pasos intermedios no tienen utilidad. De esta manera, la parte FIR del filtro, esto es el numerador de (14) no tiene que ejecutarse en cada muestra pues la única iteración útil es la última. Consecuentemente, el algoritmo de Goertzel se implememnta en dos etapas:

1) Parte recursiva con condiciones iniciales cero hasta la iteración $N-1$
2) Parte FIR de solo dos taps en la última iteración

La parte recursiva es:

$$\frac{S(z)}{X(z)} = \frac{1}{1 - 2\,\cos(\omega_0)\, z^{-1} + z^{-2}} \tag{15}$$

equivalente a:

$$s[n] = x[n] + 2\,\cos(\omega_0)\, s[n-1] - s[n-2] \tag{16}$$

la que se ejecuta con condiciones iniciales cero, para las N muestras entre la 0 y $N-1$.

Mientras que la parte FIR solo es:

$$y[N] = s[N] - e^{\frac{-j\,2\,\pi\,k}{N}}\, s[N-1] \tag{17}$$

que se ejecuta solo una vez al finalizar la parte IIR. Notar que requiere $s[N]$ de manera que se implmenta:

$$y[N] = (2\,\cos(\omega_0)\, s[N-1] - s[N-2]) - e^{\frac{-j\,2\,\pi\,k}{N}}\, s[N-1] \tag{18}$$

$$y[N] = e^{\frac{j\,2\,\pi\,k}{N}}\, s[N-1] - s[N-2] \tag{19}$$

Considerando aritemética real finalmente la parte FIR se puede separar en su parte real e imaginaria:

$$real\,\{y[N]\} = \cos(\omega_0)\, s[N-1] - s[N-2] \tag{20}$$

$$imag\,\{y[N]\} = \sin(\omega_0)\, s[N-1] \tag{21}$$

VII. Implementación en s-functions: Algoritmo de Goertzel

Descargue los archivos necesarios para esta experiencia desde el repositorio del laboratorio de la carpeta `lab4p1` (`https://bitbucket.org/Gonzalo_AC3E/labdsp_elo314_utfsm`).

Usando los archivos entregados para esta experiencia como base para trabajar, debe completar las funciones faltantes que permiten usar el algoritmo de Goertzel en Simulink mediante s-functions.

1) Se enterga el proyecto `goertzel.slx` para Simulink, y se debe trabajar sobre el archivo `goertzel.c` donde **debe crear la función** `computeGoertzel()` que recibe como argumentos el puntero al estado del filtro de Goertzel y una muestra de entrada.

```
/*
 * Estructura de estado de filtros goertzel
 */
typedef struct goertzelStateTag {
    uint64_t samplesCounter;
    double cosW;
    double sinW;
    double A1;
    double outputs[3];
    double binReal;
    double binImag;
    double binMag;
} goertzelState_t;
```

   El objetivo de su función es procesar cada muestra que recibe como entrada de forma continua, pero que actualize sus estados `.binReal`, `.binImag` y `.binMag` (valor absoluto) solo cada N muestras, para este caso $N = 256$. En Simulink, se graficará `.binMag` junto con las señales de entrada al filtro de Goertzel, y se espera que al cabo de recibir las primeras N muestas entregue el primer valor y que se retenga hasta haber recibido las siguientes N muestas para entregar su segundo valor, y así cada N muestras. Se ha entregado la función `resetGoertzel()` para invocarla cuando haya completado su algoritmo y así reinicializar el filtro para las siguientes N muestras.
   Debe también completar la función `initGoertzel()` la que será invocada solo una vez al inicio de la simulación mientras sea llamada dentro de la función `initialization()` ya entregada como parte del proyecto base (existe la definición `def.StartFcnSpec = 'initialization()';` en el archivo `creeate_sFunction.m`).
   (**10pto**) Use el archivo `dtmfSequence_16_16.wav` para validar su codigo, **muestre los segmentos de codigo relevantes**, y muestre en su informe **una comparación** entre la magnitud obtenida por su algoritmo para las primeras N muestras del archivo simulando para los bins 8, 9 y 10, y las correspondientes magnitudes obgenidas con `abs(fft())` de matlab para las mismas N muestras. Puede usar `stem` para los primeros 30 bins del comando `fft`.

2) (**10pto**) Instancie un total de 7 filtros de Gortzel correctamente inicializados para lograr obtener los bins de frecuencia más cercanos a las frecuencias de los DTMF de la experiencia 3. Considere que el archivo de audio usado es de 16 ksps que se usarán $N = 256$ muestras. ¿**Cuales bins k a elegido y por qué**?.
   Muestre los segmentes de código relevantes y un gráfico usando `subplot` y el comando `bar` (en vez de `plot`) con los resultados simulados para las 7 salidas de los filtros Goertzel junto a la señal de pruebas `dtmfSequence_16_16.wav`. Debiera observarse que la magnitud de los coeficientes de salida sean altos solo para dos de los siete en todo momento y correspondientes a los DTMF.

Informe de Laboratorio:
Presente sus diseños, ecuaciones y los gráficos requeridos. Comente sus observaciones al implementar cada filtro. Los códigos generados y el informe deberán ser enviados vía email siguiendo las instrucciones.