Práctica de Laboratorio de Filtros Digitales con Scipy Signal

Curso: Procesamiento Digital de Señales

Universidad de Antioquia

Facultad de Ingeniería

Departamento de Ingeniería Electrónica y de Telecomunicaciones

Ingeniería de Telecomunicaciones

Objetivos:

- Entender cómo utilizar la librería *scipy.signal* para operar con filtros digitales.

Introducción a Filtros Digitales

En esta práctica de laboratorio veremos varias clases de filtros que entran en la categoría de filtros IIR y FIR digitales utilizando la librería *scipy.signal* la cual es una librería de código abierto especializada en el procesamiento de señales y escrita en lenguaje Python (Signal processing with scipy.signal | Scipy Docs).

Empezaremos con algunas nociones básicas, como la definición de los filtros, la utilidad de su comportamiento y cómo Scipy Signal ha definido el diseño de los filtros. Finalmente veremos la caracterización de los filtros a través la comparación entre distintas clases de filtros IIR, entre tipos IIR y FIR y entre métodos de filtrado hacia adelante (*forward*) y hacia adelante y atrás (*forward and backwards*) dados por la librería Scipy Signal.

¿Qué es Scipy Signal?

Scipy es una librería que contiene un compendio de módulos utilizados en múltiples áreas de las ciencias de datos. **Signal** es el módulo que contiene una extensa variedad de herramientas enfocadas en el procesamiento de señales (convoluciones, filtrado y mucho más). Además, Scipy Signal se apoya en Numpy para realizar gran conjunto de operaciones por lo que en algunos casos utilizar una u otra librería resulta indistinguible o con los mismos efectos.

El diseño de las clases de filtros IIR se realizará usando la sección *Matlab-style IIR filter design* y la aplicación de filtros se realizará utilizando la sección *Filtering* donde se encuentran clasificadas las funciones de `lfilter(b, a, x[, axis, zi])` y `filtfilt(b, a, x[, axis, padtype, padlen, ...])`

¿Qué hace realmente un filtro?

Un filtro se diseña para eliminar la potencia de ciertas frecuencias particulares de una señal para un propósito en específico dependiendo de la aplicación. Gracias a Fourier entendemos las señales aleatorias como una suma de infinitas señales, cada una con una potencia y una frecuencia determinadas llamadas **componentes espectrales**. El trabajo de un filtro consiste en eliminar la potencia de ciertas componentes seleccionadas y dejar otras tal con su potencia original o incluso amplificarlas.

Ejemplos de utilidad de filtros hay incontables, por nombrar unos cuantos:

- Un ***ecualizador de audio*** es una aplicación que utiliza un filtro para suprimir la energía de ciertas frecuencias con el objetivo de realzar otras según el género musical o el instrumento a resaltar. El género Jazz puede brillar a través de un filtro que suprime las frecuencias bajas y realce las altas para que el saxofón, la trompeta y el piano resalten sobre el bajo y la batería.
- Otro ejemplo es un ***sistema de control*** que realiza filtrado sobre señales externas al sistema para eliminar el ruido y así analizarlas, optimizarlas y corregirlas para obtener información que no es posible obtener a través de la observación directa.
- Otra utilidad muy común de los filtros es para ***acotar la banda base de la señal*** en el transmisor de un sistema de comunicaciones antes de ser modulada y transmitida. El canal tiene una **capacidad** limitada a causa del ruido, la cual está dada por el Teorema de Shannon-Hartley. Si no existiera el ruido en el canal, en teoría se podría enviar información ilimitada con potencia casi de cero, pero debido al ruido, a mayor frecuencia de la señal, más daños sufre esta. La señal es bien distinguible en el receptor cuando su banda modulada no supera la capacidad del canal.

Función de transferencia de un filtro

Un filtro, al fin y al cabo, es un sistema lineal invariante en el tiempo o LTI. Como todo LTI, tiene una función de transferencia dada por

$$H(Z) = \frac{B(Z)}{A(Z)}$$

Las funciones $B(Z)$ y $A(Z)$ son llamadas "funciones de coeficientes del sistema" y denotan si el filtro es amortiguado, críticamente amortiguado o sobreamortiguado según la distribución de los ceros y polos, es decir, los ceros que anulan $B(Z)$ y $A(Z)$ respectivamente.

Los filtros IIR que han sido construidos en la librería *scipy.signal* son ***Butterworth***, ***Chebyshev 1***, ***Chebyshev 2***, ***Elliptic*** y ***Bessel*** usando respectivamente las funciones `butter`, `cheby1`, `cheby2`, `ellip` y `bessel`. Dichas funciones

de diseño de filtros retornan como resultado las funciones de coeficientes $B(Z)$ y $A(Z)$ en forma de vectores numéricos donde el primer elemento del vector es el coeficiente de z^0 y el último de z^{-n} donde n es el orden del filtro.

Por ejemplo, dada la siguiente $H(Z)$:

$$H(Z) = \frac{B(Z)}{A(Z)} = \frac{1 - 0.67z^{-3} + z^{-5}}{1 + 0.9z^{-5}}$$

Las funciones B y A son representadas por los vectores de coeficientes

```
b=array([1, 0, 0, -.67, 0 1])
```

```
a=array([1, 0, 0, 0, 0, .9])
```

Clases de filtros comunes

La clase de un filtro determina varias propiedades, como el rizado en las bandas filtradas y atenuadas o la pendiente de la banda de transición. La clase de un filtro es independiente de su orden o frecuencia de corte.

En Scipy, la función `butter` siver para diseñar un filtro de clase Butterworth y recibe los siguientes parámetros:

```
b, a = butter(N, Wn, btype='low', analog=False, output='ba',
fs=None)
```

Donde: 1. `N` (Entero): El orden del filtro.

2. `analog` (Bool): False si el filtro es digital, True si es analógico.
3. `fs` (Float o None): La frecuencia de muestreo. Sólo puede ser diferente de None si el filtro es digital. Por defecto es `None`.
4. `btype` (String): Dice el tipo de filtro (pasabaja, pasaalta, pasabanda, rechazabanda). Por defecto es `'lowpass'`
5. `Wn` (Escalar o arreglo de 2 escalares): La frecuencia o frecuencias de corte. Depende de los parámetros anteriores. Si el filtro es pasabaja o pasaalta, el filtro tiene una banda de transción y por lo tanto Wn es un escalar; de lo contrario si es pasabanda o rechazabanda, el filtro tiene 2 bandas de transición y Wn es un arreglo de 2 frecuencias de corte. Si el filtro es analógico, Wn está dado en unidades de rad/s; de lo contrario si el filtro es digital y el parámetro `fs = None`, Wn es un número o números entre 0 y 1 (normalizados), pero si el parámetro `fs` es flotante, estas frecuencias están en las mismas unidades de `fs`. En este último caso se debe tener en cuenta el teorema de Nyquist nombrado anteriormente por lo que `fs` debe ser por lo menos 2*Wn.

Pregunta 1

Cada una de las siguientes preguntas finaliza con una gráfica, por lo que se recomienda declarar una figura con `plt.figure(figsize=(ancho, alto))` y graficar las respuestas en diferentes *subplots*.

> Nota: importe las siguientes librerías importantes para todo el proyecto

```
from ipywidgets import interact # Para los widgets de controles
from scipy import signal        # Para el procesamiento de señales
import numpy as np              # Para operaciones con vectores
import matplotlib.pylab as plt  # Para las gráficas
%matplotlib inline
```

Para facilitar la solución de esta pregunta, desarrolle todo el código dentro de una función llamada `filtrar_buffer` donde sus parámetros de entrada están dados por los widgets ingresados al decorador `@interact()`. Utilice el siguiente código inicial:

```
@interact(
    frecuencia_corte=(0.0005, 1, 0.1),
    tipo_banda=["lowpass", "highpass"],
    clase_filtro=["butterworth",
                  "chebyshev1",
                  "chebyshev2",
                  "elliptic",
                  "bessel"],
    orden=(1, 10, 1),
    rizado_dB=(0.0, 5, 0.3)
    )
def filtrar_buffer(frecuencia_corte,
                   tipo_banda,
                   clase_filtro,
                   orden,
                   rizado_dB,
                   ):
```

Pregunta 1.1

Construya una señal sinusoidal con 3 frecuencias normalizadas distintas llamada `buffer` donde $f_1 = 1000 \quad ciclos/muestra, f_2 = 11000 \quad ciclos/muestra, f_3 = 20000 \quad ciclos/muestra$, con 200 muestras en un rango de tiempo normalizado (entre 0 y 1). Por comodidad, se recomienda utilizar la función `np.linspace()`. Grafique la señal y agregue título y rótulos para los ejes.

¿A qué hace referencia el tiempo normalizado y cuál es su ventaja sobre el tiempo medido en segundos?

Pregunta 1.2

Obtenga los vectores de coeficientes `b, a` de cada uno de los tipos de filtro ***Butterworth***, ***Chebyshev 1***, ***Chebyshev 2***, ***Elliptic*** y ***Bessel*** usando respectivamente las funciones `butter()`, `cheby1()`, `cheby2()`, `ellip()` y `bessel()`. Los filtros deben poder cambiar entre pasabajos y pasaaltos a través del widget de control llamado `tipo_banda`, tener una frecuencia de corte normalizada 0.5, y en los casos que aplique, un rizado máximo permitido controlado por el parámetro `rizado_dB` dB en la banda filtrada y uno de 30 dB en la banda atenuada.

¿En cuáles filtros aparece el rizado y a qué hace referencia este? ¿Por qué es importante tenerlo en cuenta?

Pregunta 1.3

Obtenga la respuesta en amplitud al impulso de cada uno de estos filtros utilizando al función `freqz()`, con 8000 muestras. Luego, en base a dicho resultado obtenga la respuesta en potencia logarítmica al impulso. Grafique ambas versiones de la respuesta.

- ¿Por qué es útil tener en cuenta ambas versiones?
- ¿Qué observa en los filtros Butterworth y Bessel en común con las demás clases de filtros y qué diferencia puede observar entre ellos? ¿Cuál es su utilidad?
- Para la versión logarítmica, marque "el punto de -3 dB" con una línea horizontal usando `plt.axhline()` ¿Por qué es importante este punto?
- Varíe la potencia del rizado ¿qué puede notar en la respuesta al impulso de los filtros que son afectados por este?

  Nota 1: La función `freqz()` retorna 2 parámetros: el eje de la frecuencia angular normalizada en *rad/muestra* (entre 0 y π) y la respuesta compleja cuya magnitud está entre 0 y 1.

  Nota 2: Se recomienda usar `plt.ylim([-30,1])` en la gráfica logarítmica para visualizar mejor el efecto

Pregunta 1.4

Obtenga la señal filtrada pasando el *buffer* por todos los filtros utilizando el método de filtrado `lfilter()`. Grafique la salida del filtro.

¿Qué diferencias encuentra con la señal de entrada?

Pregunta 2

FIR vs IIR

Los algoritmos de filtrado pueden implementarse como respuesta al impulso finita o como respuesta al impulso infinita. Los filtros FIR son muy costosos desde el punto de vista computacional, pero normalmente superan a los diseños de filtros IIR o analógicos, ya que pueden lograr una transición muy pronunciada (denominada ***rolloff***) entre la banda filtrada y la banda atenuada. También son de fase lineal.

Los filtros IIR o de Respuesta Infinita al Impulso son tipos de filtros que a menudo se basan en filtros analógicos y se han convertido en sus equivalentes digitales. Se suelen utilizar en el procesamiento digital de audio así como en sistemas de control y otras aplicaciones.

Un filtro de respuesta al impulso finito (FIR) es un tipo de filtro en el procesamiento de señales cuya respuesta al impulso (o respuesta a cualquier entrada de longitud finita) es de duración finita, ya que se establece en cero en un tiempo finito. Esto contrasta con los filtros de respuesta al impulso infinito (IIR), que pueden tener retroalimentación interna y pueden continuar respondiendo indefinidamente (generalmente disminuyendo)

Diferencia	Filtro digital FIR	Filtro digital IIR
1. Duración del impulso	La respuesta al impulso del sistema FIR tiene duración finita, es decir, tiene un valor diferente de cero para un determinado número de muestras y es cero para el resto. i. e. $h[n] = 0$ para n<3 y n>=M. Así, las muestras de respuesta unitaria existen para una duración entre 3 y M-1.	El sistema IIR presenta una respuesta de muestra unitaria de duración infinita. i. e. $h[n] = 0$ para n < 0. Así la respuesta existe para n entre 0 a infinito.
2. Recursividad	Los sistemas FIR son no recursivos por lo que la salida del filtro depende no más que de las entradas pasadas y la entrada actual.	Los sistemas IIR son recursivos, es decir, tienen retroalimentación (feedback) por lo que la salida del filtro depende no solo de las entradas pasadas y la actual sino también de las salidas pasadas.

Diferencia	Filtro digital FIR	Filtro digital IIR
3. Ecuación de diferencias del sistema LSI (Lineal Invariante de Desplazamiento)	$y(n) = \sum_{k=0}^{M} b_k x(n-k)$	$y(n) = \sum_{k=0}^{M} b_k x(n-k) - \sum_{k=0}^{N} a_k x(n-k)$
4. Costo computacional	Los filtros FIR son muy costosos desde el punto de vista computacional	Los filtros IIR son mucho más eficientes desde el punto de vista computacional
5. Retardo de fase	Los filtros FIR son de fase lineal	Los filtros IIR no son de fase lineal
6. Desempeño	Los filtros FIR pueden lograr una transición muy pronunciada en la banda de transición	Los filtros IIR a menudo no consiguen una atenuación tan pronunciada como los filtros FIR

Pregunta 2.1

Diseñe dos filtros FIR pasabajas de orden 5 y de orden 55 utilizando `firwin()` con la frecuencia de corte normalizada de 0.5 ciclos por muestra.

> Nota: para configurarlo como pasabajas, debe dejar el parámetro `pass_zero=True`

La función retorna la respuesta al impulso, cuya longitud es igual al orden del filtro.

¿Qué observa de las respuestas al impulso?

Pregunta 2.2

Filtre y la señal `buffer` de la pregunta 1 con los dos filtros usando el método `filtfilt()`. Grafique la señal original y la filtrada.

¿Qué diferencias existe entre las respuestas de los dos filtros FIR diseñados y por qué las señales muestran el comportamiento visualizado?

> Nota: la señal `buffer` fue creada dentro de una función que no la retornó, por lo que no existe en el ámbito global. Debe crearla de nuevo desde cero como lo hizo en la pregunta 1.

Pregunta 3

Retardo de grupo y retardo de fase

Un filtro puede impartir un **retardo de tiempo** tanto en la **amplitud** de la señal como en su **fase**. El tamaño de estos retardos puede depender de la frecuencia de la señal. El retardo de tiempo de la amplitud se conoce como **retardo de grupo**, y el de la fase se conoce como **retardo de fase**.

Un **filtro tiene respuesta de fase lineal** si para todas las fases de todas las componentes frecuenciales se observa el mismo retardo.

Pregunta 3.1

Construya un filtro Butterworth pasabajas de orden 4 y frecuencia de corte normalizado de 0.03 ciclos/muestra. Construya una señal de impulso unitario de 1000 muestras usando la función `np.zeros()` donde la muestra 500 tenga valor 1.

Obtenga la respuesta al impulso usando `filtfilt()` y usando dos veces `lfilter()`.

Grafique las respuestas de potencia en dB al impulso.

¿Observa diferencia entre ambas gráficas de potencia?

Pregunta 3.2

Cree una señal aleatoria de ruido Browniano utilizando `np.cumsum(randn(800))` y nómbrela como `sig`.

Filtre a `sig` utilizando una vez `filtfilt()`.

Paralelamente, filtre a `sig` dos veces usando `lfilter()` de tal manera que se pueda observar el efecto de este método de filtrado.

Grafique en una misma gráfica a `sig` y el resultado de las dos filtraciones con rótulos y colores en las curvas para diferenciarlas.

- ¿Qué nota de característico entre ambos métodos de filtrado?
- ¿Encuentra alguna utilidad en la diferencia entre ambos métodos?

Conclusiones

Escriba conclusiones significativas del laboratorio