Técnicas Digitales III

Trabajo práctico: Filtrado digital IIR

1. Filtro Leaking Integrator (LI) con señales senoidales en Python

a) Genere una señal senoidal con frecuencia fundamental de 100Hz.

b) Agregue ruido a la señal senoidal tal que la relación señal a ruido entre la señal senoidal y la señal con ruido sea de 15 dB.

c) Diseñe un filtro *leaking integrator* (LI) con λ igual a 0.7.

d) Grafique la respuesta en frecuencia y fase del filtro LI. Use la función freqz(). Determine la frecuencia de corte fco con:

$$fco = - \ln(\lambda) \cdot fs / \pi$$

e) Determine el cero y el polo del filtro con la función zplane() provista. ¿Es el filtro estable?.

f) Aplique el filtro LI a la señal con ruido. Utilice la función `lfilter()` de scipy.signal. Determine los valores de b y a.

```
from scipy.signal import lfilter

y = lfilter(b, a, x)
```

g) Grafique la respuesta en el tiempo de las señales original y filtrada y compare.

h) Grafique la respuesta en frecuencia de las señales original y filtrada y compare.

```
import numpy as np

frequencies = np.fft.fftfreq(len(x), d=sample_spacing)
import matplotlib.pyplot as plt

plt.plot(frequencies, np.abs(Y))
plt.xlabel('Frecuencia [Hz]')
plt.ylabel('Amplitud')
plt.grid()
plt.show()
```

i) Repita los puntos c) a h) para λ igual a 0.9 y 0.98. Analice el comportamiento de la fco.

2. Filtro IIR en el dominio de la frecuencia con señales de audio en Python

a) Cargue el archivo de audio provisto llamado Tchaikovsky.mat.

```
import scipy.io

data = scipy.io.loadmat('Tchaikovsky.mat')

Fs = data['Fs'][0]
signal = data['signal']
```

Se cargarán dos variables, la matriz signal con dos canales (estéreo) y la variable Fs. Elija 1 de los 2 canales disponibles.

a) Diseñe un filtro IIR elíptico usando la función `signal.ellip()` de la biblioteca `scipy`:

```
from scipy import signal

orden = 6
fs = 44100  # Frecuencia de muestreo
f1, f2 = 300, 3400  # Frecuencias de corte
rp = 0.5  # Ripple en la banda de paso en dB
rs = 60   # Atenuación en la banda de rechazo en dB

# Diseñar filtro pasa-banda Elliptic
b, a = signal.ellip(orden, rp, rs, [f1, f2], btype='bandpass',
fs=fs)
```

b) Grafique la respuesta en frecuencia del filtro IIR.

```
# Respuesta en frecuencia del filtro
w, h = signal.freqz(b, a, worN=2000)

# Graficar respuesta en frecuencia
plt.figure(figsize=(12, 6))

plt.subplot(2, 1, 1)
```

```
plt.plot(w * fs / (2 * np.pi), 20 * np.log10(abs(h)),
color='blue')
plt.title('Respuesta en Frecuencia del Filtro Elliptic
Pasa-Banda')
plt.xlabel('Frecuencia [Hz]')
plt.ylabel('Amplitud [dB]')
plt.grid(True)
plt.axvline(f1, color='green')  # Frecuencia de corte inferior
plt.axvline(f2, color='green')  # Frecuencia de corte superior
plt.axhline(-rp, color='red', linestyle='--')   # Ripple de la
banda de paso
plt.axhline(-rs, color='red', linestyle='--')   # Atenuación de
la banda de rechazo
plt.xlim(0, fs/2)
plt.ylim(-80, 5)

# Graficar respuesta de fase
plt.subplot(2, 1, 2)
plt.plot(w * fs / (2 * np.pi), np.unwrap(np.angle(h)),
color='blue')
plt.xlabel('Frecuencia [Hz]')
plt.ylabel('Fase [radianes]')
plt.grid(True)
plt.xlim(0, fs/2)

# Mostrar gráficos
plt.tight_layout()
plt.show()
```

c) Aumente el orden del filtro a 12. ¿Se modifica la respuesta en frecuencia del filtro?.

d) Transforma el filtro a una arquitectura SOS.

```
import scipy.signal as signal

# Convertir los coeficientes b y a a la representación SOS
sos = signal.tf2sos(b, a)
```

e) Utilice como señal de entrada el archivo Tchaikovsky.mat al filtro.

```
# Filtrar una señal x usando la representación SOS
y = signal.sosfilt(sos, signal)
```

f) Grafique los espectros de la señal original (signal) y filtrada (y) con la función.
g) Examine ambas gráficas. ¿Qué diferencia observa entre ambas señales?.

3. Diseño de Filtros Pasa-Banda FIR e IIR

Ejecute el script `ej_03.py` proporcionado. El mismo implementa el diseño de un filtro FIR y un filtro IIR con respuestas en frecuencia similares, pasa-banda con frecuencias de corte de 300 Hz y 3400 Hz. El objetivo final es comparar la cantidad de coeficientes en cada filtro.

a) Diseño del Filtro FIR:
Utiliza una ventana de Hamming y la técnica de ventaneo para diseñar el filtro FIR.
El orden del filtro es 101.
Utiliza la función `scipy.signal.firwin` para obtener los coeficientes del filtro.

b) Diseño del Filtro IIR:
Diseña un filtro IIR utilizando un filtro Elíptico.
El orden del filtro es 6.
Utiliza la función `scipy.signal.ellip` para obtener los coeficientes del filtro.

c) Visualización:
Grafica la respuesta en frecuencia de ambos filtros y compáralas.

4. Comparación de Filtros Pasa-Bajo FIR e IIR

Crear una señal 50 Hz con ruido de 200 Hz, el ruido debe ser de la mitad de amplitud que la señal de 50 Hz
a) Diseñar un filtro FIR pasa-bajo usando ventana de Hamming con los siguientes datos:
-Frecuencia de corte fc = 50 Hz
-Frecuencia de muestreo fs = 1000 Hz
-Número de coeficientes = 101
Utiliza la función `scipy.signal.firwin` para el diseño del filtro
Aplicar el filtro a una señal con ruido (`scipy.signal.firwin`) y visualizar los resultados

b) Diseñar un filtro IIR de tipo Butterworth de pasa-bajo con los siguientes parámetros
-Frecuencia de corte fc = 50 Hz
-Frecuencia de muestreo fs = 1000 Hz
-Orden = 4
Utiliza la función `scipy.signal.butter` para el diseño del filtro
Aplicar el filtro a la señal con ruido y mostrar los resultados.

c) Comparación de FIR vs IIR
Graficar las señales filtradas y original en la misma gráfica