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

Diseñar un filtro FIR y un filtro IIR con respuestas en frecuencia similares y que sean pasa-banda con frecuencias de corte de 300 Hz y 3400 Hz.

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.

Diseño del Filtro IIR:
Diseña un filtro IIR utilizando un filtro Butterworth.
El orden del filtro es 6.
Utiliza la función scipy.signal.butter para obtener los coeficientes del filtro.

```
# Diseñar filtro FIR pasa-banda
numtaps = 101  # Número de coeficientes
fir_coefs = signal.firwin(numtaps, [f1, f2], pass_zero=False, fs=fs)

# Diseñar filtro IIR pasa-banda (Butterworth)
iir_coefs    =    signal.iirfilter(6,    [f1,    f2],    btype='band',
ftype='butter', fs=fs)

# Respuesta en frecuencia de los filtros
w_fir, h_fir = signal.freqz(fir_coefs, worN=2000, fs=fs)
w_iir,  h_iir  =  signal.freqz(iir_coefs[0],  iir_coefs[1],  worN=2000,
fs=fs)
```

Visualización:
Grafica la respuesta en frecuencia de ambos filtros y compáralas.
Asegúrate de que ambos filtros tengan respuestas en frecuencia similares.

```
# Graficar respuesta en frecuencia de los filtros
plt.figure(figsize=(12, 6))

plt.subplot(2, 1, 1)
plt.plot(w_fir, 20 * np.log10(abs(h_fir)), label='FIR')
```

```
plt.plot(w_iir,    20    *    np.log10(abs(h_iir)),    label='IIR
(Butterworth)')
plt.title('Respuesta en Frecuencia de los Filtros Pasa-Banda')
plt.xlabel('Frecuencia [Hz]')
plt.ylabel('Amplitud [dB]')
plt.grid(True)
plt.legend()
plt.axvline(f1,  color='red',  linestyle='--')   #  Frecuencia  de
corte inferior
plt.axvline(f2,  color='red',  linestyle='--')   #  Frecuencia  de
corte superior
plt.xlim(0, fs/2)

# Graficar fase de los filtros
plt.subplot(2, 1, 2)
plt.plot(w_fir, np.angle(h_fir), label='FIR')
plt.plot(w_iir, np.angle(h_iir), label='IIR (Butterworth)')
plt.xlabel('Frecuencia [Hz]')
plt.ylabel('Fase [radianes]')
plt.grid(True)
plt.legend()
plt.xlim(0, fs/2)

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