Chapter 8: IIR Filters: Simple Design Example

8.1 Introduction

In this chapter, we will walk through a simple design example of an **IIR filter** (Infinite Impulse Response filter). IIR filters are used widely in signal processing because they can provide efficient solutions to many problems, such as noise removal, frequency shaping, and signal enhancement.

We will design a low-pass IIR filter using both the **Impulse Invariant Method** and the **Bilinear Transform Method**, which are two common methods of transforming analog filter designs into digital IIR filters.

This example will help you understand the practical application of IIR filter design methods and how to implement them in digital signal processing.

8.2 Problem Statement

We want to design a simple **low-pass IIR filter** with the following specifications:

- **Analog Cutoff Frequency**: $f_c = 1 \, \text{Hz}$
- **Sampling Frequency**: $f_s = 10 \, \text{Hz}$
- **Filter Order**: 1st order (to keep the design simple)

Our goal is to design a low-pass filter that attenuates frequencies higher than $1 \, \text{Hz}$ and passes frequencies below this threshold.

8.3 Step 1: Design Analog Low-Pass Filter

The first step is to design the analog low-pass filter. For this, we can use the standard **first-order low-pass filter** transfer function in the s-domain (analog):

$H(s) = \frac{K}{\tau s + 1}$

Where:

- K is the gain (typically $K = 1$).

- τ is the time constant of the filter, related to the cutoff frequency by $\tau = \frac{1}{2\pi f_c}$, where f_c is the cutoff frequency.

Given that the cutoff frequency $f_c = 1 \, \text{Hz}$, we can calculate the time constant τ:

$$\tau = \frac{1}{2\pi \cdot 1} \approx 0.159 \, \text{seconds}$$

So, the transfer function of the analog filter is:

$$H(s) = \frac{1}{0.159s + 1}$$

8.4 Step 2: Apply the Impulse Invariant Method

The **Impulse Invariant Method** involves mapping the continuous-time (analog) filter's **impulse response** to the discrete-time domain. To do this, we apply the transformation $s = \frac{1 - z^{-1}}{T}$ to the s-domain filter.

- **Sampling Period** $T = \frac{1}{f_s} = \frac{1}{10} = 0.1 \, \text{seconds}$.

- The impulse response of the analog filter is $h_{\text{analog}}(t) = e^{-\frac{t}{\tau}}$.

To map the analog filter to the digital domain, we use the formula:

$$H(z) = H(s) \Bigg|_{s = \frac{1 - z^{-1}}{T}}$$

For our 1st-order low-pass filter, the transfer function in the z-domain is:

$$H(z) = \frac{1}{\left(\frac{1 - z^{-1}}{T} \cdot 0.159 + 1\right)}$$

Substitute $T = 0.1$ and simplify the expression:

$$H(z) = \frac{1}{\left(0.159 \cdot \frac{1 - z^{-1}}{0.1} + 1\right)}$$

Simplify further:

$$H(z) = \frac{1}{(1.59(1 - z^{-1})) + 1}$$

Thus, the discrete-time transfer function of the low-pass filter is:

$$H(z) = \frac{1}{(1.59 - 1.59z^{-1}) + 1}$$

8.5 Step 3: Apply the Bilinear Transform Method

The **Bilinear Transform Method** maps the entire s-plane to the z-plane using the transformation:

s=2T · 1−z−11+z−1s = \frac{2}{T} \cdot \frac{1 - z^{-1}}{1 + z^{-1}}

For a 1st-order low-pass filter, we start with the same analog transfer function:

H(s)=1τs+1H(s) = \frac{1}{\tau s + 1}

Substitute ss from the bilinear transform equation:

H(z)=1τ(2T · 1−z−11+z−1)+1H(z) = \frac{1}{\tau \left(\frac{2}{T} \cdot \frac{1 - z^{-1}}{1 + z^{-1}} \right) + 1}

Simplify this expression to get the z-domain transfer function.

For fs=10 Hzf_s = 10 \, \text{Hz} and τ=0.159 seconds\tau = 0.159 \, \text{seconds}, we have T=0.1T = 0.1. Applying the bilinear transformation results in a new z-domain filter:

H(z)=1(20.1 · 1−z−11+z−1 · 0.159)+1H(z) = \frac{1}{\left(\frac{2}{0.1} \cdot \frac{1 - z^{-1}}{1 + z^{-1}} \cdot 0.159 \right) + 1}

Simplifying the expression yields the final form of the digital filter transfer function in terms of zz.

8.6 Step 4: Frequency Response

Once we have the z-domain transfer function, we can calculate the frequency response of the filter. The **frequency response** H(ejω)H(e^{j\omega}) can be found by substituting z=ejωz = e^{j\omega} into the transfer function. This provides insight into how the filter behaves in the frequency domain, showing which frequencies are passed and which are attenuated.

8.7 Step 5: Implementation in Code

Here is a simple Python implementation of the designed IIR low-pass filter using the **Bilinear Transform Method** with **scipy.signal** for filter design.

```
import numpy as np

import scipy.signal as signal

import matplotlib.pyplot as plt
```

```
# Design Parameters
fs = 10  # Sampling frequency in Hz
fc = 1   # Cutoff frequency in Hz
tau = 1 / (2 * np.pi * fc)  # Time constant


# Design the analog low-pass filter (s-domain)
# H(s) = 1 / (s + 1/tau)
b, a = signal.butter(1, fc, fs=fs, btype='low')


# Frequency Response
w, h = signal.freqz(b, a, fs=fs)


# Plot frequency response
plt.figure()
plt.plot(w, np.abs(h), 'b')
plt.title("Frequency Response of the IIR Low-pass Filter")
plt.xlabel('Frequency [Hz]')
plt.ylabel('Amplitude')
plt.grid(True)
plt.show()
```

This Python code uses **scipy.signal.butter** to design a first-order low-pass IIR filter, and it plots the frequency response of the filter. You can modify the `fc` and `fs` parameters to adjust the cutoff frequency and sampling rate for different designs.

8.8 Step 6: Filter Analysis and Results

After obtaining the transfer function, the frequency response of the filter can be analyzed to verify the filter characteristics, such as the cutoff frequency, stopband attenuation, and the shape of the frequency response.

- **Cutoff Frequency:** The cutoff frequency is where the filter attenuates the signal by 3 dB3 \, \text{dB} (half power).
- **Passband and Stopband:** The filter should allow signals below the cutoff frequency to pass through while attenuating frequencies above the cutoff.

For the low-pass filter example, the desired behavior would be:

- Signals below 1 Hz should pass through with minimal attenuation.
- Signals above 1 Hz should be significantly attenuated.

8.9 Conclusion

In this chapter, we walked through the design of a simple **low-pass IIR filter** using the **Impulse Invariant** and **Bilinear Transform Methods**. Both methods are widely used for converting analog filter designs to their digital counterparts, each with its advantages:

- The **Impulse Invariant Method** preserves the time-domain characteristics of the analog filter.
- The **Bilinear Transform Method** ensures that aliasing is avoided and provides a more accurate digital representation of the analog filter's frequency response.