L-Università ta' Malta

CCE3206—Digital Signal Processing

Practical 2

Trevor Spiteri
trevor.spiteri@um.edu.mt

Last updated: 2022-10-17

Discrete-time signals

Objective

The objective of this practical is to implement and analyse an IIR system.

Tasks

1. Figure 1 shows a first-order infinite-duration impulse response (IIR) filter described by the difference equation

   $$y(n) = 0.6y(n-1) + 0.2x(n) + 0.2x(n-1)$$

   Write a Python function that takes a NumPy array as the input and outputs the response of the system in Figure 1 to this input. The output must be a NumPy array of the same length and type as the input.

   The function can look something like this:

   ```
   import numpy as np
   ```

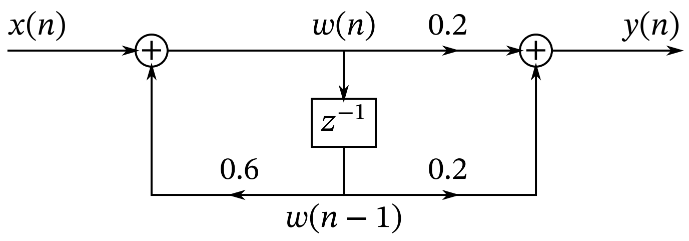

Figure 1: A first-order IIR system.

```
def filter(x):
    # set y to have the same length and type as x
    y = np.zeros_like(x)
    # iterate over all input samples
    for n, x_n in enumerate(x):
        # TODO: compute y(n) and store it in y[n]
```

Hint: At the beginning of iteration $n$, you should have $w(n-1)$ in a variable. At the end of the iteration, you should update this variable with the new value of $w(n)$ to prepare it for the next iteration.

(20 marks)

2. Plot the sample response of the filter. This can be achieved as follows.

   (a) Generate an input sequence $x(n) = \delta(n)$ for a length of 10 samples. You can create a zero array using `np.zeros` and then set its first sample to one.

   (b) Produce an output sequence $y(n)$ by passing the input sequence to the filter from Task 1.

   (c) Plot the output using code similar to:

   ```
   import matplotlib.pyplot as plt
   fig, ax = plt.subplots()
   n = np.arange(len(y))
   ax.stem(n, y)
   ax.set_xlabel('$n$')
   ax.set_ylabel('$y(n)$')
   plt.show()
   ```

(10 marks)

3. Change the input signal to $x(n) = \mathrm{U}(n)$ for a length of 20 samples and

repeat Tasks 2b–2c. You can generate $x(n)$ using `np.ones`. **(10 marks)**

4. Change the input signal to $x(n) = \mathrm{U}(n) - \mathrm{U}(n-10)$ and repeat Tasks 2b–2c. **(10 marks)**

5. Now filter the audio file provided *tones.wav* and observe the output.

   (a) First read the input sequence $x(n)$ from the file.

   ```
   from scipy.io import wavfile
   [fs, x] = wavfile.read('tones.wav')
   ```

   Here, `x` is the sequence and `fs` is the sampling rate.

   (b) Filter the audio using the filter from Task 1.

   (c) Save the output.

   ```
   wavfile.write('output_tones.wav', fs, y)
   ```

   (d) Using a suitable application, listen to and compare the input audio file and the output audio file. Did the system have any audible effect on the sound?

   Hint: Instead of listening to the audio, or in addition to listening, you can open the audio files in an audio editor such as Audacity (https://www.audacityteam.org/) and visually observe the waveform.

   (10 marks)

6. Plot the power spectrum of both the input signal and the output signal. Compare the two plots. Does this correspond with the observed difference, if any, in Task 5d?

   To plot the spectrum of the input `x`:

   ```
   from scipy.fft import fft, fftfreq

   X = fft(x) / len(x)
   # spectrum is symmetric, so remove second half
   X = X[:len(X) // 2]
   Xp = np.abs(X) ** 2

   # frequencies corresponding to the output of fft
   F = fftfreq(len(x), 1 / fs)
   ```

```
F = F[:len(F) // 2]

fig, ax = plt.subplots()
ax.plot(F, Xp)
ax.set_xlabel('$F$')
ax.set_ylabel('$|X(F)|^2$')
ax.grid(True)

# TODO: similarly plot power spectrum of Y against F

plt.show()
```

(10 marks)

7. Generate a sinusoid $x(n) = \cos \omega_0 n$ as the new input, where ω_0 is the signal frequency in rad/sample. Take $\omega_0 = 0.1$ rad/sample. To generate a sinusoid of length L:

   ```
   n = np.arange(L)
   x = np.cos(n * w0)
   ```

   Determine the gain and phase shift of the filter by plotting the input and output on the same axis. Make sure that your length L is long enough, and ignore the transient at the beginning.

   ```
   fig, ax = plt.subplots()
   ax.plot(n, x, label='$x(n)$')
   ax.plot(n, y, label='$y(n)$')
   ax.set_xlabel('$n$')
   ax.legend(loc='best', framealpha=1)
   ax.grid(True)
   plt.show()
   ```

   The delay can be measured at the zero crossings of the sinusoids as shown in Figure 2. Although this delay is *not* an integer, it is measured in samples and must be converted to radians by multiplying by ω_0.

   (10 marks)

8. Generate 10 logarithmically spaced values for ω_0 in the range 0.1 to π rad/sample using `np.geomspace(0.1, np.pi, 10)`, and repeat Task 7 for all the values of ω_0.

   Then, plot the magnitude and phase response of the filter using your

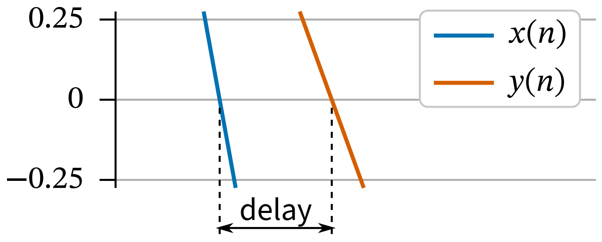

Figure 2: Measuring the phase shift.

measurements, excluding any points that have zero gain. The magnitude should be plotted in dB: if the amplitude gain is G, then the power gain in dB is $20\log_{10} G$. Use a logarithmic scale for the frequency on the x-axis, for example by plotting with the `semilogx` method.

(20 marks)

Report

- If there are answers that require some calculations on plot readings, the report must show both the actual readings and the calculated answers.
- Include any general observations and comments in your report.

Acknowledgements

This practical is based on a similar practical by Victor Buttigieg.