ECE 438 - Laboratory 2
Discrete-Time Systems

Last updated on January 17, 2022

In [1]:
```
import numpy as np
import matplotlib.pyplot as plt
import soundfile as sf
import IPython.display as ipd
```

In [2]:
```
# make sure the plot is displayed in this notebook
%matplotlib inline
# specify the size of the plot
plt.rcParams['figure.figsize'] = (16, 6)

# for auto-reloading extenrnal modules
%load_ext autoreload
%autoreload 2
```

1. Introduction

A discrete-time system is anything that takes a discrete-time signal as input and generates a discrete-time signal as output. The concept of a system is very general. It may be used to model the response of an audio equalizer or the performance of the US economy.

In electrical engineering, ***continuous-time*** signals are usually processed by electrical circuits described by differential equations. For example, any circuit of resistors, capacitors, and inductors can be analyzed using mesh analysis to yield a system of differential equations. The voltages and currents in the circuit may then be computed by solving the equations.

The processing of ***discrete-time*** signals is performed by discrete-time systems. Similar to the continuous-time case, we may represent a discrete-time system either by a set of difference equations or by a block diagram of its implementation. For example, consider the following difference equation.

$$y[n] = y[n-1] - 2x[n] + 3x[n-1] \tag{1}$$

This equation represents a discrete-time ***system***. It operates on the input signal $x[n]$ to produce the output signal $y[n]$. This system may also be defined by a system diagram as in Figure 1.

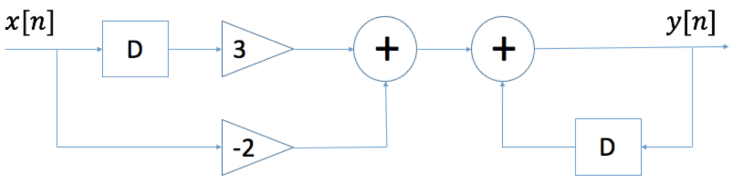

Figure 1: Diagram of a discrete-time system. (D = unit delay)

Mathematically, we use the notation $y = S[x]$ to denote a discrete-time system S with input signal $x[n]$ and output signal $y[n]$. Notice that the input and output to the system are the complete signals for **all** time n. This is important since the output at a particular time can be a function of past, present, and future values of $x[n]$.

It is usually quite straightforward to write a computer program to implement a discrete-time system from its difference equation. In fact, programmable computers are one of the easiest and most cost-effective ways of implementing discrete-time systems.

While Equation (1) is an example of a linear time-invariant system, other discrete-time systems may be nonlinear and/or time-varying. In order to understand discrete-time systems, it is important to first understand their classification into categories of linear/nonlinear, time-invariant/time-varying, causal/noncausal, memoryless/with-memory, and stable/unstable. Then it is possible

to study the properties of restricted classes of systems, such as discrete-time systems, which are linear, time-invariant, and stable.

2. Example of Discrete-time Systems

Discrete-time digital systems are often used in place of analog processing systems. Common examples are the replacement of photographs with digital images and conventional NTSC TV with direct broadcast digital TV. These digital systems can provide higher quality and/or lower cost through the use of standardized, high-volume digital processors.

The following two continuous-time systems are commonly used in electrical engineering:

$$\text{differentiator:} \quad y(t) = \frac{d}{dt}x(t) \tag{2}$$

$$\text{integrator:} \quad y(t) = \int_{-\infty}^{t} x(\tau)d\tau \tag{3}$$

To illustrate how a discrete-time system can be derived from the corresponding continuous-time system, we will show how the above two continuous-time systems can be formulated into corresponding discrete-time systems.

2.1 Formulation of a discrete-time system that approximates the continuous-time differentiator

Since $y(t) = \frac{d}{dt}x(t)$, by the definition of differentiation, $y(t) = \lim_{\Delta t \to 0} \frac{x(t+\Delta t)-x(t)}{\Delta t}$, so sampling $y(t)$ at time nT with sufficiently small T gives

$$y(nT) \approx \frac{x(nT) - x\left((n-1)T\right)}{T} \tag{4}$$

Clearly, this approximation is more accurate when T is sufficiently small. If we define $y[n] \triangleq y(nT)$ and $x[n] \triangleq x(nT)$, then we arrive at a discrete-time system that approximates the continuous-time differentiator: $y[n] = \frac{(x[n]-x[n-1])}{T}$.

2.2 Formulation of a discrete-time system that approximates the continuous-time integrator

Since $y(t) = \int_{-\infty}^{t} x(\tau)d\tau$, so sampling $y(t)$ at time nT gives

$$\begin{aligned} y(nT) &= \int_{-\infty}^{nT} x(\tau)d\tau \\ &= \int_{-\infty}^{(n-1)T} x(\tau)d\tau + \int_{(n-1)T}^{nT} x(\tau)d\tau \\ &\approx y((n-1)T) + x(nT)\cdot T \end{aligned}$$

Clearly, this approximation is more accurate when T is sufficiently small. If we define $y[n] \triangleq y(nT)$ and $x[n] \triangleq x(nT)$, then we arrive at a discrete-time system that approximates the continuous-time integrator: $y[n] = y[n-1] + x[n]\cdot T$.

Exercise 2

1. Draw the block diagram of the discrete-time differentiator as in Figure 1.

insert your diagram here

2. Apply the discrete-time differentiator to the signal $u[n] - u[n-(N+1)]$, with $N = 10$, for $-10 \le n \le 20$, and plot the output. (This assumes a time step of $T = 1$)

In [3]:
```
# write your code here
```

3. Draw the block diagram of the discrete-time integrator as in Figure 1.

write your answer here

4. Apply the discrete-time integrator to the signal $u[n] - u[n-(N+1)]$, with $N = 10$, for $-10 \leq n \leq 20$, and plot the output. (This assumes a time step of $T = 1$)

In [4]:
```
# write your code here
```

5. Use the discrete-time differentiator to numerically evaluate $\frac{d}{dt}x(t)$ of $x(t) = \sin(2\pi t)$ for $t \in [0, 10]$, try $T = 0.1$.

In [5]:
```
# write your code here
```

6. Use the discrete-time differentiator to numerically evaluate $\frac{d}{dt}x(t)$ of $x(t) = \sin(2\pi t)$ for $t \in [0, 10]$, try $T = 0.001$.

In [6]:
```
# write your code here
```

7. Compare the results in Q5 and Q6.

write your answer here

3. Difference Equations

In this section, we will study the effect of two discrete-time filters. The first filter, $y = S_1[x]$, obeys the difference equation

$$y[n] = x[n] - x[n-1]$$

and the second filter, $y = S_2[x]$, obeys the difference equation

$$y[n] = \frac{1}{2}y[n-1] + x[n]$$

Exercise 3

1. For each of these five systems, do the following:

- S_1
- S_2
- $S_1(S_2)$ (i.e., the series connection with S_1 following S_2)
- $S_2(S_1)$ (i.e., the series connection with S_2 following S_1)
- $S_1 + S_2$

i. Draw the system diagram (use only delays, multiplications, and additions as in Figure 1)

insert your diagram here

ii. Write code to implement each of these two filters (S_1, S_2) in the following functions using the following syntax

```
def S1(x):
    """
    Parameters
    ---
    x: the discrete-time signal

    Returns:
    ---
    y: the filtered signal
    """
    pass
```

and

```
def S2(x):
    """
    Parameters
    ---
    x: the discrete-time signal

    Returns:
    ---
    y: the filtered signal
    """
    pass
```

In [7]:
```
# write your code here
```

iii. Write code to calculate the impulse response of each of the five systems, and plot the impulse response of each of these five systems:

In [8]:
```
# write your code here
```

2. Use the command `audio, fs = sf.read("music.au")` to load the file `music.au` into Python. The first variable `audio` is the signal, and the second variable `fs` is the sampling rate.

In [9]:
```
# write your code here
```

3. Play the audio file using the command `ipd.Audio(audio, rate=fs)`.

In [10]:
```
# write your code here
```

4. Next, filter the audio signal with each of the two systems S_1 and S_2 and listen to each of their outputs.

In [11]:
```
# write your code here
```

5. How do the filters change the sound of the audio signals? Explain your observations.

write your answer here

4. Inverse Systems

Exercise 4

1. Consider the system $y = S_2[x]$ from Section 3. Find a difference equation for a new system $y = S_3[x]$ such that $\delta = S_3[S_2[\delta]]$ where δ denotes the discrete-time impulse function $\delta(n)$. Since both systems S_2 and S_3 are LTI, the time-invariance and superposition properties can be used to obtain $x = S_3[S_2[x]]$ for *any* discrete-time signal x. We say that the systems S_3 and S_2 are inverse filters because they cancel out the effects of each other.

Hint: The system $y = S_3[x]$ can be described by the difference equation

$$y[n] = ax[n] + bx[n-1]$$

where a and b are constants.

insert your answer here

2. Complete the Python function using the following syntax.

```
def S3(x):
    """
    Parameters
    ---
    x: the discrete-time signal

    Returns:
    ---
    y: the filtered signal
    """
    pass
```

In [12]:
```
# write your code here
```

3. Write code to plot the impulse response of both S_3 and $S_3[S_2[\delta]]$ below.

In [13]:
```
# write your code here
```

5. System Tests

Exercise 5

Often it is necessary to determine if a system is linear and/or time-invariant. If the inner workings of a system are not known, this task is impossible because the linearity and time-invariance properties must hold true for all possible input signals. However, it is possible to show that a system is non-linear or time-varying because only a single instance must be found where the properties are violated.

The code in the cell below is the synatx for the three imported black boxes, `bbox1`, `bbox2` and `bbox3`. All of them have the same syntax

```
def bboxN(x):
    return y
```

where `x` and `y` are the input and the output signals, and `N=1,2,3`. **Exactly one of these systems is non-linear, and exactly one of them is time-varying.** Your task is to find the non-linear system and the time-varying system.

Hints:

- You should try a variety of input signals until you find a counter-example.

- When testing for time-invariance, you need to look at the responses to a signal and to its delayed version. Since all your signals in Python have a finite duration, you should be very careful about shifting signals. In particular, if you want to shift a signal x by M samples to the left, x should start with at least M zeros. If you want to shift x by M samples to the right, x should end with at least M zeros.
- When testing for linearity, you may find that simple inputs such as the unit impulse do not accomplish the task. In this case, you should try something more complicated like a sinusoid or a random signal generated with the `x = np.random.rand(N)` command, where `N` here is the length of the signal.

1. Import the three bbboxes from the file `bboxes.py` .

In [14]:
```
# write your code here
```

2. Write code to plot input/output signal pairs that support your conclusions. Create multiple code cells if necessary.

In [15]:
```
# write your code here
```

2. State which system is non-linear and which system is time-varying.

write your answer here

3. Explain how the plots support your conclusion.

write your answer here

6. Stock Market Example

6.1 Moving Averages

One reason that digital signal processing (DSP) techniques are so powerful is that they can be used for very different kinds of signals. While most continuous-time systems only process voltage and current signals, a computer can process discrete-time signals, which are essentially just sequences of numbers. Therefore DSP may be used in a very wide range of applications. Let's look at an example.

A stockbroker wants to see whether the average value of a certain stock is increasing or decreasing. To do this, the daily fluctuations of the stock values must be eliminated. A popular business magazine recommends three possible methods for computing this average.

$$\text{averagevalue(today)} = \frac{1}{3}(\text{value (today)} + \text{value (yesterday)} + \text{value (2 days ago)}) \tag{5}$$

$$\text{averagevalue(today)} = 0.6 \times \text{averagevalue(yesterday)} + 0.4 \times (\text{value (today)}) \tag{6}$$

$$\text{averagevalue(today)} = \text{averagevalue(yesterday)} + \frac{1}{3}(\text{value (today)} - \text{value (3 days ago)}) \tag{7}$$

Exercise 6.1

For each of these three methods:

1. write a difference equation

write your answer here

2. Draw a system diagram

insert your diagram here

3. Calculate the impulse response

write your answer here

4. Explain why methods (5) and (7) are known as moving average.

write your answer here

Exercise 6.2

Load the file `stockrates.npy` into Python. This file contains a vector of daily stock market exchange rates for a publicly-traded stock.

Apply filters (6) and (7) to smooth the stock values. When you apply filter (6), you will need to initialize the value of *averagevalue(yesterday)*. Use an initial value of 0. Similarly, in filter (7), set the initial values of the *value* vector to 0 (for the days prior to the start of data collection).

Note: You will need the following code to load data that is stored in a `.npy` file.

In [16]:
```
rate = np.load("stockrates.npy")
```

1. Plot the original stock values.

In [17]:
```
# write your code here
```

2. Plot the result of filtering with (6).

In [18]:
```
# write your code here
```

3. Plot the result of filtering with (7).

In [19]:
```
# write your code here
```

4. Discuss the advantages and disadvantages of the two filters (6) and (7). Can you suggest a better method for initializing the filter outputs?.

write your answer here