EE323 Digital Signal Processing
Lab 2 - Discrete-Time Systems

12 Oct. 2017 ~ 25 Oct. 2017

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.

$$y(n) = y[n-1] + x[n] + x[n-1] + x[n-2] \tag{2.1}$$

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 2.1.

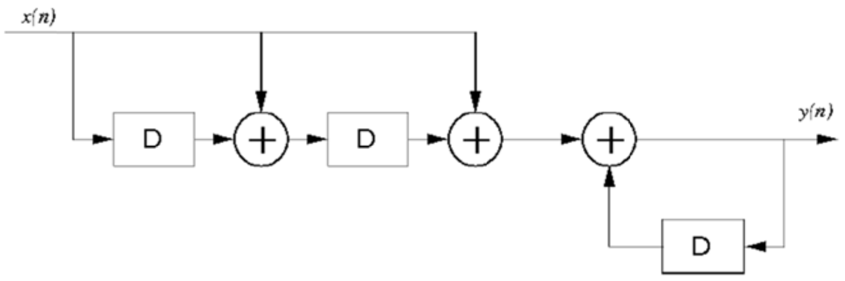

Figure 2.1: Diagram of a discrete-time system. (D = unit delay)

Mathematically, we use the notation $y = \boldsymbol{S}(x)$ to denote a discrete-time system $\boldsymbol{S}$ 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 (2.1) 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.

2.2 Background Exercises

INLAB REPORT: Submit these background exercises with the lab report.

2.2.1 Example 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:

differentiator: $y(t) = \frac{d}{dt} x(t)$

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

(2.2)

For each of these two systems, do the following:

i. Formulate a discrete-time system that approximates the continuous-time function.
ii. Write down the difference equation that describes your discrete-time system. Your difference equation should be in closed form, i.e. no summations.
iii. Draw a block diagram of your discrete-time system as in Figure 2.1.

2.2.2 Stock Market Example

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.

avgvalue[today] = $\frac{1}{3}$ (value[today]+value[yesterday]+value[2 days ago]) (2.3)

avgvalue[today] = 0.8 * avgvalue[yesterday] + 0.2 * (value[today]) (2.4)

avgvalue[today] = avgvalue[yesterday] + $\frac{1}{3}$ (value[today]–(value[3 days ago]) (2.5)

Do the following:

- For each of these three methods: 1) write a difference equation, 2) draw a system diagram, and 3) calculate the impulse response.
- Explain why methods (2.3) and (2.5) are known as moving averages.

2.3 Example Discrete-Time Systems

Write two Matlab functions that will apply the differentiator and integrator systems, designed in the "Example Discrete-time Systems" (Section 2.2.1: Example Discrete-time Systems) section, to arbitrary input signals. Then apply the differentiator and integrator to the following two signals for $-10 \leq n \leq 20$.

- $\delta[\mathrm{n}] - \delta[\mathrm{n}-5]$
- $u[\mathrm{n}] - u[n-(N+1)]$ with $N = 10$

HINT: To compute the function $u(n)$ for $-10 \leq n \leq 20$, first set **n = -10:20**, and then use the Boolean expression **u = (n>=0)**.

For each of the four cases, use the subplot and stem commands to plot each input and output signal on a single figure.

INLAB REPORT: Submit printouts of your Matlab code and hardcopies containing the input and output signals. Discuss the stability of these systems.

2.4 Difference Equations

In this section, we will study the effect of two discrete-time filters. The first filter, $y = \boldsymbol{S}_1(x)$, obeys the difference equation

$$y[\mathrm{n}] = x[n] - x[n-1] \tag{2.6}$$

and the second filter, $y = \boldsymbol{S}_2(x)$, obeys the difference equation

$$y[\mathrm{n}] = \frac{1}{2}y[n-1] + x[n] \tag{2.7}$$

Write Matlab functions to implement each of these filters.

NOTE: In Matlab, when implementing a difference equation using a loop structure, it is very good practice to pre-define your output vector before entering into the loop. Otherwise, Matlab has to resize the output vector at each iteration. For example, say you are using a **for** loop to filter the signal $x[n]$, yielding an output $y[n]$. You can pre-define the output vector by issuing the command **y=zeros(1,N)** before entering the loop, where **N** is the final length of **y**. For long signals, this speeds up the computation dramatically.

Now use these functions to calculate the impulse response of each of the following 5 systems: $\boldsymbol{S}_1$, $\boldsymbol{S}_2$, $\boldsymbol{S}_1(\boldsymbol{S}_2)$ (i.e., the series connection with $\boldsymbol{S}_1$ following $\boldsymbol{S}_2$), $\boldsymbol{S}_2(\boldsymbol{S}_1)$ (i.e., the series connection with $\boldsymbol{S}_2$ following $\boldsymbol{S}_1$), and $\boldsymbol{S}_1 + \boldsymbol{S}_2$.

INLAB REPORT: For each of the five systems, draw and submit a system diagram (use only delays, multiplications and additions as in Figure 2.1). Also submit plots of each impulse response. Discuss your observations.

2.5 Audio Filtering

For this section download the music.au file. For help on how to play audio signals see audio.pdf.

Use the command auread to load the file music.au5 into Matlab. Then use the Matlab function sound to listen to the signal.

Next filter the audio signal with each of the two systems $\boldsymbol{S}_1$ and $\boldsymbol{S}_2$ from the previous section.

Listen to the two filtered signals.

INLAB REPORT: How do the filters change the sound of the audio signals? Explain your observations.

2.6 Inverse Systems

Consider the system $y = \boldsymbol{S}_2(x)$ from the "Difference Equations" (Section 2.4: Difference Equations) section. Find a difference equation for a new system $y = \boldsymbol{S}_3(x)$ such that $\delta = \boldsymbol{S}_3(\boldsymbol{S}_2(x))$ where δ denotes the discrete-time impulse function $\delta[\mathrm{n}]$. Since both systems $\boldsymbol{S}_2$ and $\boldsymbol{S}_3$ are LTI, the time-invariance and superposition properties can be used to obtain $x = \boldsymbol{S}_3(\boldsymbol{S}_2(x))$ for any discrete-time signal x. We say that the systems $\boldsymbol{S}_3$ and $\boldsymbol{S}_2$ are inverse filters because they cancel out the effects of each other.

HINT: The system $y = \boldsymbol{S}_3(x)$ can be described by the difference equation

$$y[\mathrm{n}] = ax[n] + bx[n-1] \tag{2.8}$$

where a and b are constants.

Write a Matlab function $y = \boldsymbol{S}_3(x)$ which implements the system $\boldsymbol{S}_3$. Then obtain the impulse response of both $\boldsymbol{S}_3$ and $\boldsymbol{S}_3(\boldsymbol{S}_2(x))$.

INLAB REPORT: Draw a system diagram for the system $\boldsymbol{S}_3$, and submit plots of the impulse responses for $\boldsymbol{S}_3$ and $\boldsymbol{S}_3(\boldsymbol{S}_2)$.

2.7 System Tests

For this section download the zip file bbox.zip.

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 inputs signals. However, it is possible to show that a system is non-linear or time-varying because only a single instance have to be found where the properties are violated. The zip file bbox.zip contains three "black-box" systems in the files bbox1.p, bbox2.p, and bbox3.p.

These files work as Matlab functions, with the syntax **y=bboxN(x)**, where **x** and **y** are the input and the output signals, and **N = 1, 2 or 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:

1. You should try a variety of input signals until you find a counter-example.
2. 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 MATLAB have 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.
3. 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 random command.

INLAB REPORT: State which system is non-linear, and which system is time-varying. Submit plots of input/output signal pairs that support your conclusions. Indicate on the plots why they support your conclusions.

2.8 Stock Market Example

For this section download stockrates.mat. For help on loading Matlab files see load.pdf.

Load stockrates.mat into Matlab. This file contains a vector, called **rate**, of daily stock market exchange rates for a publicly traded stock. Apply filters (2.4) and (2.5) from the "Stock Market Example" (Section 2.2.2: Stock Market Example) section of the background exercises to smooth the stock values. When you apply the filter of (2.4) you will need to initialize the value of avgvalue(yesterday). Use an initial value of 0. Similarly, in (2.5), set the initial values of the "value" vector to 0 (for the days prior to the start of data collection). Use the subplot command to plot the original stock values, the result of filtering with (2.4), and the result of filtering with (2.5).

INLAB REPORT: Submit your plots of the original and filtered exchange-rates. Discuss the advantages and disadvantages of the two filters. Can you suggest a better method for initializing the filter outputs?

2.9 Exercises:

1. Consider the discrete-time system characterized by the input-output relation

$$y[n] = \frac{1}{2}\left(y[n-1] + \frac{x[n]}{y[n-1]}\right),$$

where $y[n]$ and $x[n]$ are, respectively, the output and input sequences.
(1) Show that the output $y[n]$ of this system for an input $x[n] = \alpha\mu[n]$ with $y[-1]=1$ converges to sqrt(α) as $n \to \infty$ when α is a positive number.
(2) Write a Matlab program to verify.

2. An algorithm for the calculation of the square root of a number is given by

$$y[n] = x[n] - y^2[n-1] + y[n-1].$$

(1) Show that the output $y[n]$ of this system for an input $x[n] = \alpha\mu[n]$ with $0 < \alpha < 1$ converges to sqrt(α) as $n \to \infty$, while $y[-1]$ may be a suitable initial approximation of sqrt(α).
(2) Write a Matlab program to compute the square root using the above algorithm, and verify the above conclusion.
(3) Plot the error as a function of n for several different values of α.
(4) How would you compute the square root of a number α with a value greater than one?

INLAB REPORT: Submit your derivations, Matlab codes, plots of the above two exercises?