GEORGIA INSTITUTE OF TECHNOLOGY
SCHOOL of ELECTRICAL and COMPUTER ENGINEERING

ECE 2025 Fall 1999
Lab #10: Discrete-Time Simulation of Continuous-Time Systems

Date: 9–15 Nov 1999

Lab Quiz #3 will be given at the beginning of this lab.

This is *the official* Lab #10 description.

The lab report for this lab will be **INFORMAL:** discuss your simulation results from section 4. Staple the **Instructor Verification** sheet to the end of your lab report.

The report will **due during the week of 16–22 Nov. at the start of your lab**.

---

1 Introduction & Objective

The goal of the laboratory project is to show that discrete-time systems can be used to produce the output of continuous-time systems via the process known as *simulation*. The simulation cannot be exact, so it always involves some degree of approximation. In particular, we want to study how closely the behavior of a continuous-time system can be approximated with an IIR filter, and how the approximation depends on the "sampling rate" of the simulation.

2 Background: Simulation

Many times, it is useful to *simulate* the behavior of a continuous-time system by using digital computation. In such cases, it is necessary to deal with sampled signals and discrete-time systems that are derived from the continuous-time system. In this project you will study a simple example of how this can be done.

2.1 Continuous-Time Systems Described by Differential Equations

Many continuous-time systems are described by differential equations. Examples include electrical networks, biological processes, and mechanical systems. In such cases, the input and output are related by a linear, constant-coefficient, differential equation of the general form

$$\sum_{k=0}^{N} a_k y^{(k)}(t) = \sum_{k=0}^{M} b_k x^{(k)}(t), \tag{1}$$

where $y^{(k)}(t) = \frac{dy^k(t)}{dt^k}$. This equation is assumed to be satisfied for every value of t. Thus, if we sample $x(t)$ and $y(t)$ and all the required derivatives at times nT_s, we can obtain a relation involving the samples of the derivatives as in

$$\sum_{k=0}^{N} a_k y^{(k)}(nT_\mathrm{s}) = \sum_{k=0}^{M} b_k x^{(k)}(nT_\mathrm{s}). \tag{2}$$

In general, we would not know any of the derivatives of either $x(t)$ or $y(t)$. Nevertheless, (2) is a useful starting point for our discrete simulation.

In order to obtain a difference equation that we can solve for $y(nT_\mathrm{s})$, we need to express all the sampled derivatives in terms of $y(nT_\mathrm{s})$ and $x(nT_\mathrm{s})$. One simple approach is to recall that the derivative of a function

at a given time is just the slope of the function at that time. For example we could approximate $y^{(1)}(nT_s)$ by the difference between $y(nT_s)$ and its neighboring sample divided by the time increment T_s. If we pick the sample just before time nT_s, we get

$$y^{(1)}(nT_s) = \frac{y(nT_s) - y(nT_s - T_s)}{T_s}. \tag{3}$$

Equation 3 is called a *first backward difference* approximation of the derivative. An alternative approximation is the *first forward difference* approximation

$$y^{(1)}(nT_s) = \frac{y(nT_s + T_s) - y(nT_s)}{T_s}. \tag{4}$$

Now since the derivatives are related as $y^{(k)}(t) = \dfrac{d}{dt}y^{(k-1)}(t)$, with $y^{(0)}(t) = y(t)$, we can successively apply either (3) or (4) to obtain all the approximate sampled derivatives in (2) in terms of either $y(nT_s)$ or $x(nT_s)$. The following example illustrates how this is done.

Example: Suppose that the system is described by the differential equation

$$y^{(2)}(t) + y^{(1)}(t) + 1.25y(t) = x(t).$$

The true impulse response is:

$$h(t) = -j\tfrac{1}{2}e^{-0.5t}e^{jt}u(t) + j\tfrac{1}{2}e^{-0.5t}e^{-jt}u(t) = e^{-0.5t}\sin(t)\,u(t)$$

and this can be verified by taking the first and second derivatives.

The sampled version of this differential equation would be

$$y^{(2)}(nT_s) + y^{(1)}(nT_s) + 1.25y(nT_s) = x(nT_s), \tag{5}$$

If we can express both $y^{(2)}(nT_s)$ and $y^{(1)}(nT_s)$ in terms of $y(nT_s)$, then we can write an equation for the output $y(nT_s)$ in terms of the input $x(\cdot)$. One way to do this is to use the *backward difference* formula in (3) to replace the derivatives with differences

$$y^{(1)}(nT_s) = \frac{y(nT_s) - y(nT_s - T_s)}{T_s}, \tag{6}$$

which gives us $y^{(1)}(nT_s)$ in terms of $y(nT_s)$. *Note: this replacement is an approximation that is known to be valid in the limit as* $T_s \to 0$.

Now we can apply the backward difference formula a second time to write the second derivative $y^{(2)}(nT_s)$ in terms of the first derivative $y^{(1)}(nT_s)$:

$$y^{(2)}(nT_s) = \frac{y^{(1)}(nT_s) - y^{(1)}(nT_s - T_s)}{T_s}. \tag{7}$$

Then we can substitute (6) into (7) to write the second derivative in terms of sample values of the output signal, $y(T_s)$.

$$y^{(2)}(nT_s) = \frac{y(nT_s) - 2y(nT_s - T_s) + y(nT_s - 2T_s)}{T_s^2}. \tag{8}$$

Finally, we substitute (6) and (8) into (5) and collect terms to obtain the difference equation

$$\begin{aligned} y(nT_s) \;=\; & \left(\frac{2 + T_s}{1 + T_s + 1.25T_s^2}\right) y(nT_s - T_s) - \left(\frac{1}{1 + T_s + 1.25T_s^2}\right) y(nT_s - 2T_s) \\ & + \left(\frac{T_s^2}{1 + T_s + 1.25T_s^2}\right) x(nT_s). \end{aligned} \tag{9}$$

Now at last we have obtained a recursive difference equation in the form that is an IIR digital filter, if we recognize that $y[n] = y(nT_s)$.

3 Warm-up

In the warm-up you will implement a simulation of the second-order differential equation above and verify that an IIR filter can produce an acceptable simulation.

(a) Use equation (9) to find the IIR filter coefficients (`aa` and `bb`) when $T_s = 0.1$; give numerical values.

Make a plot of the poles of the IIR filter when $T_s = 0.1$. and zeros of the IIR digital filter that is doing the simulation. Use the MATLAB function called `zplane().`

Instructor Verification (separate page)

(b) Do some work by hand to verify that the true step response is

$$s(t) = \frac{4}{5}u(t) + (-0.4 + j0.2)e^{-0.5t}e^{jt}u(t) + (-0.4 - j0.2)e^{-0.5t}e^{-jt}u(t)$$

This involves taking the first and second derivative and adding together the three terms on the left-hand side of the differential equation.Remember that the derivative of the unit-step signal, $u(t)$, is the unit impulse signal, $\delta(t)$.

Plot the true step response by plotting samples of $s(t)$ above at the times $t_n = nT_s$.

(c) We can implement the difference equation in (9) using MATLAB's `filter(bb,aa,xx)` function. In using this function, we would need to sample the input $x(t)$ to obtain the vector `xx`, and we would need to obtain the coefficient vectors `bb` and `aa`. Write a MATLAB function (called `stepsim()`) to compute the unit step response of the system of the above example; i.e., assume an input $x(t) = u(t)$. Your function should take the sampling period `Tsim` as an input variable, so that you can run different cases.

Your MATLAB function `stepsim()` should compute samples of the output over the time interval $-2 \leq nT_s \leq 20$. Your function will have to recompute the vectors `aa` and `bb` when T_s changes. You will also have to generate the sampled input vector `xx`. (Since the unit step is infinitely long, you will have to create its values over the time interval of interest.)

Instructor Verification (separate page)

(d) Run your MATLAB function `stepsim()` for values of $T_s = 1,\ 0.1$, and 0.01, each time computing the output at sample times in the interval $-2 \leq nT_s \leq 20$. Plot all three simulations on the same graph and show the plot to your lab TA. What can you say about the effect of the sampling parameter T_s? Which value of T_s gives a simulation that is closest to the theoretically correct answer?

Instructor Verification (separate page)

(e) *Optional:* Modify your `stepsim()` function to do the simulation for a new input signal that is a pulse: $x(t) = u(t) - u(t - 10)$. Call this function `pulsesim().` Again evaluate the output over the interval $-2 \leq nT_s \leq 20$. Test your program for the value of T_s determined to give the best results in the previous part. Plot the output as a function of time. Using the principles of superposition and time-invariance, explain why the plot has similar behavior at the beginning and end of the pulse.

4 Laboratory: Simulation with a Non-Linear System

The power of numerical simulation lies in the arena of simulating systems that cannot be solved by the analytical methods of differential equations. Thus the preceding example is not that interesting because we already know the solution. However, when part of the system is described by a non-linear operation and part by a LTI system, the method of numerical simulation is quite handy. The following example shows one common case in EE.

4.1 Simulation of a Power Supply Circuit

A common problem is that of converting an AC voltage to a DC voltage; no doubt you have in your possession many of these little "power packs" for modems, calculators, phones, etc. An alternating current (AC) voltage waveform is a sinusoid at 60 Hz (in the US); a direct current (DC) voltage is a constant, or zero frequency waveform. Figure 1 depicts a circuit for creating a DC voltage from an AC voltage. The diode bridge circuit implements what is called a "full-wave rectifier," and the RC circuit provides lowpass filtering to produce a (nearly) constant DC output.

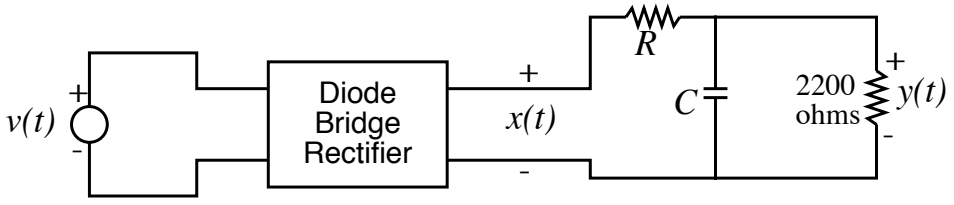

Figure 1: Power Supply circuit: AC to DC voltage converter

The block diagram shown in Fig. 2 represents the two operations performed by the different parts of the circuit. The input signal $v(t)$ is a sinusoid, typically $v(t) = A\cos(120\pi t)$ when the AC supply voltage is

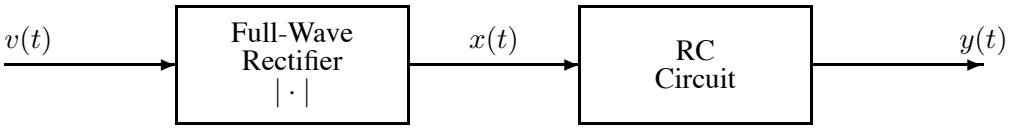

Figure 2: Block diagram representation of power supply.

60 Hz. The full-wave rectifier "takes the magnitude" by doing an absolute value operation that changes the negative values in the sinusoid to positive values. Since the absolute value operator is *non-linear,* its output $x(t)$ contains frequencies other than 60 Hz. The purpose of the lowpass filter is to remove most of the high frequencies in the output of the rectifier, producing a final output that is nearly constant with time, i.e., DC.

The full-wave rectifier is defined by the equation

$$x(t) = |v(t)| \tag{10}$$

where $v(t)$ is the input and $x(t)$ is the output of the full-wave rectifier. The output of the rectifier, $x(t)$, is the input to the RC circuit and $y(t)$ is the output of that circuit. Once you have learned circuits, it would be easy to write the differential equations that describes the R-C circuit in Fig. 1. The following differential equation gives the relationship between the input and output voltages of the circuit:

$$y^{(1)}(t) + \left(\frac{2200 + R}{2200RC}\right) y(t) = \frac{1}{RC}x(t) \tag{11}$$

In a real power supply, the signal $v(t)$ would be the powerline voltage, which would be represented mathematically as $v(t) = 120\sqrt{2}\cos(120\pi t)$. The units of $v(t)$ are *volts*.

4.2 Simulation

Now we are ready to do the numerical simulation.

(a) Using the first backward difference approximation for the derivative, determine (by hand) the IIR difference equation for simulating the RC circuit. Your answer should be in terms of the resistor R and the capacitor C, as well as the simulation time $T_{\rm s}$.

(b) Write a MATLAB function to simulate the entire system in Fig. 2. Your program should again use first backward difference approximation to the derivative and should take the sampling period $T_{\rm s}$, the resistance R, and the capacitance C as input parameters. Provide the code in your lab report.

(c) Test your simulation for the conditions $R = 33000$ ohms and $C = 0.000005$ farads. Try different values of $T_{\rm s}$ to determine a value of $T_{\rm s}$ that gives a faithful approximation.[1] Hint: What does the sampling theorem tell us about the *maximum* value of $T_{\rm s}$ that we could use to sample the input sinusoid? Why should $T_{\rm s}$ be much smaller than this? Does the rectifier output $x(t)$ have frequency components at frequencies higher than 60 Hz?

*Note: when running your simulation, evaluate the output over 20 periods of the 60-Hz power line input voltage $v(t)$. Plot the input $v(nT_{\rm s})$, the rectified signal $x(nT_{\rm s})$, and the output $y(t)$ over the time interval of the last ten periods of $v(t)$. By ignoring the first 10 periods, you should not be affected by the transient response of the system which always occurs when you have a starting time.

(d) Determine the *largest* value of $T_{\rm s}$ that you can use and still get a faithful simulation. Even though you don't know the true answer, you can start with a very small value of $T_{\rm s}$, say $T_{\rm s} = 10^{-5}$ sec., and keep making $T_{\rm s}$ larger until the output starts to change noticeably.

4.3 Output Voltage and Ripple

The RC circuit cannot smooth the signal perfectly so we have to choose the component values for R and C to trade off the output voltage level against the residual ripple that is not removed by the RC circuit.

(a) Using the values of $T_{\rm s}$, R and C from the previous part, measure the DC (i.e., average) value of the output voltage and also the "peak-to-peak" ripple in volts. Make this measurement on signal in its steady-state region.

(b) Using the value of $T_{\rm s}$ determined in the previous part, experiment with keeping $R = 33000$ ohms and varying C. How should C be changed to make the output more like a constant (smaller ripples)? What value of C is necessary to achieve less than 0.5 volts of peak-to-peak ripple about the constant output value (i.e., ± 0.25 volts)? What is the approximate DC output voltage of this power supply design? Does the DC output voltage depend on C?
Remember that the input powerline sinusoid has a peak value of $120\sqrt{2}$ volts.

(c) Now let both R and C vary. Use your simulation to determine R and C such that the DC output voltage is 3.3 volts with less than 0.3 volts of peak-to-peak ripple, i.e., between 3.15 and 3.45 volts. This will involve a little bit of trial and error.[2]

[1] The word approximation is important because you can never get a perfect answer; but you do know that the correct answer is obtained in the limit as $T_{\rm s} \to 0$.

[2] In the next lab we will analyze the DC component mathematically using the method of Fourier Series.

Lab #10
ECE-2025 Fall-1999
Instructor Verification

Name: _____ Date of Lab: _____

Part 3(a) Determine filter coefficients of IIR simulation filter. Plot the poles and zeros.

Verified:_____ Date/Time:_____

Part 3(c) Write the `stepsim()` function.

Verified:_____ Date/Time:_____

Part 3(d) Run the `stepsim()` function for several different values of T_s and make a plot showing the approximation. Then determine how small T_s has to be to give a near perfect match to the true step response.

Verified:_____ Date/Time:_____