EE 105 Simulink Lab : Numeric Simulation

Jay Farrell
College of Engineering
University of California, Riverside
jay.farrell@ucr.edu

January 18, 2021

1 Objective

In future labs, we will be using the built-in MATLAB simulation facilities. The objective of this lab is to ensure that the students understand the process and pitfalls of the numeric solution of differential equations. In this lab you will learn how to simulate a dynamic system on a computer and will construct such a simulation. Throughout this lab you *must* use vector and matrix notation and implementations.

2 Equipment

A computer and the MATLAB program.

3 Background

The numeric solution of differential equations is referred to as simulation. Simulation can be used both to verify the accuracy of your model and to experiment with the performance of your system. For all but the simplest systems, it will be much easier to solve the differential equations numerically than analytically. Since differential equations are ubiquitous in engineering applications, learning how to simulate a system (e.g., numerically solve its differential equations) will help you in many future situations.

Let's assume that you are interested in a system that can be described by a state space differential equation as

$$\begin{aligned} \dot{\mathbf{x}}(t) &= f\left(\mathbf{x}(t), u(t)\right) && (1) \\ y(t) &= g\left(\mathbf{x}(t), u(t)\right) && (2) \end{aligned}$$

From the definition of a derivative, we know that

$$\dot{\mathbf{x}}(t) = \frac{d\mathbf{x}(t)}{dt} = \lim_{h \to 0} \frac{\mathbf{x}(t+h) - \mathbf{x}(t)}{h}.$$

So that for h suitably small, we can expect,

$$\begin{aligned} \mathbf{x}(t+h) &\approx \mathbf{x}(t) + f\left(\mathbf{x}(t), u(t)\right)\ h && (3) \\ \mathbf{x}(t+2h) &\approx \mathbf{x}(t+h) + f\left(\mathbf{x}(t+h), u(t+h)\right)\ h \end{aligned}$$

to be an accurate approximation. If we denote $\mathbf{x}_k = \mathbf{x}(t + k\ h)$ where $k = 1, 2, 3, \ldots$, then eqn. (3) and the solution of (1) can be written as a recursive program:

$$\begin{aligned} \mathbf{x}_{k+1} &\approx \mathbf{x}_k + f\left(\mathbf{x}_k, u_k\right)\ h && (4) \\ y_k &= g(\mathbf{x}_k) && (5) \end{aligned}$$

In this lab you will use equations (4-5) to simulate the performance of a simple system. Each iteration of these equations advances the solution by h seconds. Simulation from t_o to t_f will require $N = \frac{t_f - t_o}{h}$ iterations.

4 Example

Consider the following example concerning the height of a fluid in a reserve tank. A control system has been designed. Analysis of the controlled system has shown that the dynamics of the fluid height are described by

$$\dot{x} = f(x) = 3 - \sqrt{2x - 1}.$$

The units for x and $\dot{x}$ are m and $\frac{m}{s}$, respectively. Prior to implementation, we would like to test the controlled system in simulation. This is much cheaper than testing the actual system, which may fail and cause damage. Therefore, the goal is to construct a numeric simulation of this dynamic system.

Consider the code for two Matlab functions in Table 1. These two functions simulate the fluid system. Note that this code is a direct implementation of eqn. (3). Since we have approximated the time derivative by a backwards difference, the time step h must be small, for this simulation to be accurate. However, if h is too small, then the number of calculations may be excessively large. Table 2 shows the simulation results (obtained from the function 'lab2') and amount of calculations (FLOPS) for four different values of h. Note that when h is larger than the time constant, the response is oscillatory, and is obviously not accurate. See Figure 1. A good rule of thumb is to select the step-size h to yield at least ten samples for the fastest time constant. In this case, the time constant is 3.0 seconds, which implies $h < 0.3$ seconds. Note that from Table 2, the simulation with $h = 0.1$ is within

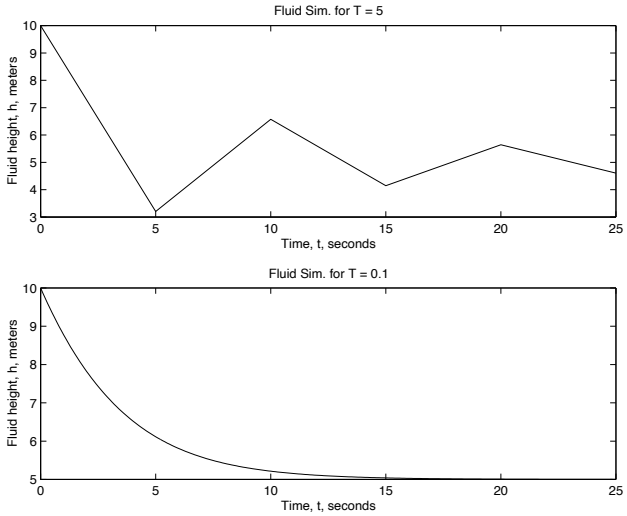

Figure 1: Results of fluid dynamic system simulation for two selected step sizes. The T in the figure title corresponds to h in the text of this document. The h in the figure y-label corresponds to the state x in the text of this document.

0.3% of the simulation with $h = 0.01$. For most practical purposes, the increased accuracy obtained with $h = 0.01$ (i.e., 0.3%) would not be worth the extra calculations (i.e., 10 times); and $h = 0.1$ would be a suitable simulation step size.

The numeric solution method described above is the most straightforward to explain; however, there are other numeric integration techniques available that are more accurate. MATLAB has several numeric integration functions (e.g., 'ode23', 'ode34') and a block diagram simulation language SIMULINK built into it. You will also use these routines in this lab.

MATLAB also contains several utility routines for state space to/from transfer function transformations (i.e., ss2tf, tf2ss); series, parallel, and feedback combinations of linear systems (i.e., series, parallel, cloop); linear system simulation (i.e., linsim); Runga-Kutta simulation (i.e., rk23, rk45); and, linear system step response calculation (i.e., step). The format for using these and any other MATLAB function 'fname' can be found by typing 'help fname.'

5 What to turn in.

Your report and conclusions should show that you understand how to write a simulation, how simulation parameters (e.g. h) effect the results, and the tradeoffs that are involved. Clearly number and indicate (circle/highlight) answers to any questions. The TA should not have to search to find them. Try to organize your report to minimize the time that the TA must take to understand it. Also, turn in your Matlab code using the method defined by the TA.

```
function [dx] = fld_lvl(t,x,u);
%
% This function represents the right hand side
% of the fluid level system dynamics. In general,
% dx will have the same dimension as
% the state vector x

dx = (3-sqrt(2*x-1)); % rhs of dynamic equation
```

==========

```
function lab2(x0,T);
% For the given initial condition x0 and step size
% T this function uses Euler integration to
% numerically solve the differential equation
% of the fluid level system.

N = round(25/T) +1;
t = zeros(1,N);
x = zeros(1,N);
x(:,1) = x0;
t = T*(0:N-1);
flops(0);

for i=1:N,
      dx = fld_lvl(t(i),x(:,i),0);
      x(:,i+1) = x(:,i) + dx*T;
end
flops

clf;
plot(t(1:i)',x(:,1:i)');
str = sprintf('Fluid Sim. for T = %g',T);
title(str);
xlabel('Time, t, seconds');
ylabel('Fluid height, h, meters');
```

Table 1: Matlab code for the fluid level example.

6 System

The transfer function for a system of interest is

$$\frac{Y(s)}{U(s)} = \frac{16}{s^2 + 6s + 16} = H(s) \tag{6}$$

where $y(t)$ is the output and $u(t)$ is the input.

7 Prelab

The standard form for the transfer function of a low pass second order system is

$$H(s) = \frac{G\omega_n^2}{s^2 + 2\zeta\omega_n s + \omega_n^2}$$

where s is the Laplace variable, G is the DC gain, ω_n is the undamped natural frequency, and ζ is the damping factor. These parameters are useful for relating the transfer function to the time domain response. Two additional important parameters are the decay rate $\sigma = \zeta\omega_n$ and the damped natural frequency $\omega_d = \omega_n\sqrt{1-\zeta}$.

1. For the transfer function in eqn. (6), what are the values of G, ζ, ω_n, σ, and ω_d?

2. Use the ideas from frequency response, i.e.,

   $$Y(s) = H(s)|_{s=j\omega}\, U(s)$$

   calculate the steady state output signal $y(t)$ when the steady state input signal is $u(t) = sin(0.1t)$.

3. Let $\mathbf{x} = [y, \dot{y}]^T$. Find the state space equations for the system. Define n, A, B, C, D, and $f(x)$. Use vector notation and be certain that all dimensions work out for the state space implementation.

8 Experimental Procedure

In the following, you will be comparing results from software that you develop with results from built in MATLAB functions. You should get nearly identical answers. If not, you should trace down the source of the errors.

1. Since eqn. (6) is a linear system, its solution can be found using the 'lsim' function. Do this first, using a zero initial condition with $u(t) = 1.0sin(.1t)$, so that you know what the correct response is for the subsequent steps. [1]

2. Manual selection of step-size h:

   (a) Write an m-file (call it f.m in the following) that implements the state space representation of the system. At least initially, assume that $u(t) = 0$. After you get your simulation running, then redefine $u(t) = 1.0sin(.1t)$.

   (b) Write an m-file to implement the recursion in (4-5) by calling 'Euler.m'.

   (c) Using an initial condition of [3.0;0], simulate the system dynamics until steady state is (approximately) achieved. You will have to iterate on the simulation step size to get an accurate but reasonable length simulation. Be careful how you measure accuracy. Be sure to describe your metric.

   In your report include for this section[2] Tables similar to Tables 1 and 2, and Figure 1. Include discussion of what step size you selected and how it was selected. Compare the time that it took to achieve 1% steady-state (i.e., setting time T_s) with the value predicted by the pole locations. One of the Figures should be for your selected step size. Discuss the physical interpretation of the state variable plots. Be reasonable in the amount of data you include in your tables. Your report should show that you thought about what you were writing and did not clutter it with anything not essential to getting your point across. All figures and tables that you include must be labeled and discussed in the body of your report. To measure the computation time consider using the functions 'tic' and 'toc'.

3. Automatic step-size selection: The step size selection procedure from part 2 is tedious. Fortunately, many people before you have felt the same. In addition, the Euler integration routine that you implemented does not have a good trade-off between accuracy and required computations. Its accuracy converges at a rate proportional to T. Alternative numeric integration routines have been derived that converge at rates proportional to higher powers of T. Since T is normally small, this can lead to significant computational savings. Matlab (and other numeric packages) includes several advanced simulation routines.

   (a) Type 'help ode23'.

   (b) Use ode23 with input function 'f.m' to simulate the system using the same initial condition as for Part 2. Discuss and compare the computation time required for each approach. Look at the time vector that is returned. You might

[1] Note that in real problems, the correct response will not be available. Therefore, you must begin to build a set of reasonableness checks so that you can evaluate whether or not a simulated solution is reasonable.

[2] Since there are two state variables, you should use the 'subplot' command to fit two plots-each of one variable-on a single page. Use 'title', 'xlabel', and 'ylabel' to label each plot. It may also be helpful to plot position (x_1) versus velocity (x_2).

notice that the integration step-size is not constant. Instead, the algorithm automatically adjusts the step size to try to maintain accuracy without requiring excessive calculations.

(c) Plot the output of ode23 versus time and compare with the simulation you wrote in Part 2. Is there any difference? How much CPU time was required and how does it compare with that required in Part 2.

NOTE: Ode23 will not necessarily have fewer flops, since this is dependent on the tolerance that is specified. In addition, there is some overhead associated with calculating the appropriate step-size. The main advantage of this type of routine is not necessarily computational savings, but automatic step-size selection. This is even more important for nonlinear problems.

4. Simulate the system when the input is u(t) = sin(0.1 t). Quantitatively compare the simulated response with the answer from the prelab. How well do they compare?

5. Simulate the system for $u(t) = sin(\omega\, t)$ for at least three different values of ω that you choose. One value of ω should be much less than ω_n, one value approximately the same as ω_n and one value much larger than ω_n. Discuss and compare the amplitude and phase shift of the output relative to the input. How do the steady state values compare with those predicted by frequency response (try using the 'Bode' function)?

	$h = 5.0$ s.		$h = 1.0$ s.		$h = 0.1$ s.		$h = 0.01$ s.	
Time, t	k	x_k	k	x_k	k	x_k	k	x_k
0.0	0	10.0000	0	10.0000	0	10.0000	0	10.0000
2.5					25	7.4284	250	7.4491
5.0	1	3.2055	5	5.8774	50	6.1149	500	6.1366
7.5					75	5.4943	750	5.5101
10.0	2	6.5747	10	5.1227	100	5.2152	1000	5.2248
12.5					125	5.0928	1250	5.0983
15.0	3	4.1467	15	5.0163	150	5.0399	1500	5.0428
17.5					175	5.0171	1750	5.0186
20.0	4	5.6436	20	5.0021	200	5.0073	2000	5.0081
22.5					225	5.0031	2250	5.0035
25.0	5	4.6068	25	5.0003	250	5.0013	2500	5.0015
FLOPS		56		236		2261		22511

Table 2: Euler simulation result for the example fluid system for various values of h. Note: *On some machines, FLOPS is now an obsolete MATLAB function. If this is true, then use CPU time instead of FLOPS.*