

## EE2823

### Project 1: Simulating Differential Equations

The objective of this project is to examine the means of numerically solving differential equations (for our purposes, we will call this process “simulating the differential equation”).

#### 1 Part 1: First-Order Differential Equations

1.1 Consider a first order differential equation

$$\frac{dy}{dt} + ay(t) = bx(t) \quad (1)$$

The mathematical definition of a derivative for  $y(t)$  is given as

$$\frac{dy}{dt} = \lim_{T \rightarrow 0} \frac{y(t) - y(t-T)}{T}$$

Therefore, an approximation for  $dy/dt$  is given by

$$\frac{dy}{dt} \cong \frac{y(t) - y(t-T)}{T}$$

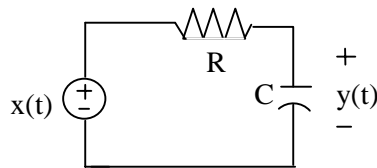
where  $T$  is small. This is known as the backward difference approximation. Use this approximation in the differential equation to obtain an approximate formula for  $y(t)$ . Write this formula in the standard form of a difference equation by letting  $t=nT$  and using the notation  $y[n]=y(nT)$ . Put your difference equation into the generic form

$$y[n] + Ay[n-1] = Bx[n]$$

Give the values for  $A$  and  $B$  in terms of  $a$ ,  $b$ , and  $T$ .

1.2 Now, using the Matlab command `filter` to compute the difference equation, write a Matlab function to simulate a first order differential equation. The input to the function should be the vector  $x$  (defined at the points  $t = nT$ ) and the scalars  $a$  and  $b$ ; the output should be the vector  $y$  defined at the points  $t = nT$ . The resulting approximation for the differential equation is generally known as Euler Integration.

1.3 Consider the RC circuit shown below.



Let  $y(t)$  denote the voltage across the capacitor and  $x(t)$  denote the voltage source. Write the differential equation for  $y(t)$  in terms of the input  $x(t)$ .

1.4 With  $R = 1000\Omega$  and  $C=100\mu f$ , compute the response of the circuit to a step input (assumes zero

initial conditions) by simulating the differential equation. Do this for three different cases:  $T = 1, 0.1,$  and  $0.01\text{sec}$ . Compute the exact solution for a first order equation in the generic form shown in Equation (1):

$$y(t) = e^{-at}y(0) + \int_0^t e^{-a(t-\lambda)}bx(\lambda) d\lambda$$

(The explanation of how to obtain this solution is given in the textbook before Equation (2.5).) Plot all three approximations and the exact answer on one plot. The approximations should be plotted versus time defined only at the points  $t=nT$ . What happens to the approximations as  $T$  gets smaller? What is the time constant of this system? Can you think of a rule for the selection of  $T$  in terms of the time constant in order to get a good approximation?

- 1.5 Now, simulate the circuit with an input of  $x(t) = \cos(2\pi 5t)$  over several cycles of the sinusoid using a time step of  $T = 0.01$ . Plot the resulting  $y(t)$ .

## 2. Solution of Second-Order Equations

Consider the generic second-order differential equation:

$$\frac{d^2y}{dt^2} + a_1 \frac{dy}{dt} + a_2 y(t) = b_1 \frac{dx}{dt} + b_2 x(t)$$

- 2.1 Determine a difference equation that approximates the differential equation. You will need to perform a sequential approximation for  $\frac{d^2y}{dt^2}$ . First, let  $y_1 = \frac{dy}{dt}$ , then use the backward difference approximation to obtain an expression for  $\frac{d^2y}{dt^2}$  in terms of  $y_1$ . Once that approximation is found, use the backward difference approximation again in order to approximate  $y_1$  in terms of  $y$  and  $\frac{dx}{dt}$  in terms of  $x$ . Write the entire approximation as a difference equation in terms of  $y$  and  $x$  with coefficients that depend on  $T$ ,  $a_1$ ,  $a_2$ ,  $b_1$  and  $b_2$ .
- 2.2 Write a Matlab function that simulates the differential equation. The input to the function should be the coefficients, the value of  $T$ , and the vector  $x$  defined at the points  $nT$  where  $n = 0, 1, 2, 3, \dots$
- 2.3 Consider the mechanical system shown in the figure. This diagram can represent a suspension system (like that in cars or that isolates sensitive equipment). Let  $x(t)$  represent the external force applied to the system; this is the input to the system. Let  $y(t)$  represent the vertical position of the mass. The equation of motion can be obtained by applying Newton's Law (sum of the forces = mass  $\times$  acceleration).

$$M \frac{d^2y}{dt^2} + D \frac{dy}{dt} + Ky(t) = x(t)$$

With values of  $M=10\text{ kg}$ ,  $D=1\text{ kg/s}$  and  $k = 1\text{ kg/s}^2$ , use your program to simulate the response to a unit step input in the force  $x(t)$ . Use values of  $T = 1, 10,$  and  $100\text{ sec}$ . Simulate the response long enough so that  $y(t)$  appears to reach a steady-state value. Plot your response for  $y(t)$  versus time (defined at  $t=nT$ ) for all three values of  $T$  on the same plot. Clearly identify each plot according to its value of  $T$ .

2.4 Matlab has a program, `lsim`, for simulating linear differential equations. It uses an approximation that is more accurate than the Euler integration method that you have written. Simulate the response using `lsim` and compare these results to yours (for  $T=1$ ) by plotting them on the same plot. The input to `lsim` is

$$y = \text{lsim}(A,B,x,t);$$

where  $A$  contains the coefficients for  $y$  and  $B$  contains the coefficients for  $x$ . For the generic second-order equation, this would be

$$A = [1 \ a_1 \ a_2]; \ B = [b_1 \ b_2];$$

2.5 There is a Java applet that simulates and animates the suspension system described above. Run the applet for the parameters given above and observe the response. It is located at <http://www.ece.gatech.edu/users/192/book/applets/suspension/MSDdemo.htm>

- Select the system parameters to match those that you used in your simulation.
- Calculate the solution.
- Start the animation.

Note, this Java program calculates the solution of the differential equation using Runge Kutta integration, which is similar but more accurate than the Euler integration method that you wrote.

2.6 Run the Java simulation with a sinusoidal input of  $10\sin(2\pi 0.1t)$ . Sketch the resulting waveform for  $y(t)$ . You can use the "Show Input/Output Summary" button to see the results.

