The University of Texas at Austin
Department of Electrical and Computer Engineering

EE362K: Introduction to Automatic Control – Fall, 2017

Problem Set 2

Recommended Reading: Chapters 3 (skim except for 3.1, 3.3, 3.6 or other examples in domains of interest to you) and 4.1 of Murray

1. Complete the exercises for Chapters 3 & 4 of the MATLAB Review. You do not need to turn in your answers, but submit an answer that asserts you completed these exercises on your own or completed an equivalent activity.

2. Consider the epidemic disease example from the State-Space Representation Lecture (slide 11). Modify the code (slide 14) to also include plot the total population that is alive on any given day. Turn in a copy of the graph with this modification. Then create a second graph that compares the total population for immunization rates of 1, 10, 100, and 1000 per day. *(Note, there are multiple ways to complete this from very hacky to elegant. All options are okay.)*

3. Consider the following 1st order differential equation.

$$\frac{dx}{dt} = 4e^{0.8t} - 0.5x$$

Determine the value of x at t=2 if $x(0)$= 2 using

(a) Euler's Method with a step size of 0.5s. (by hand or using MATLAB)
(b) Heun's Method with a step size of 1.0 s. (by hand or using MATLAB)
(c) 4th order Runga-Kutta Method with a step size of 2.0s. (by hand or using MATLAB)
(d) Using the application of your choosing (MATLAB, Excel, etc.), graph the analytical solution over the time span of interest if the solution is

$$x = \tfrac{4}{1.3}\left(e^{0.8t} - e^{-0.5t}\right) + 2e^{-0.5t}$$

On the same graph, plot and label you numerical solutions. Be sure to label the graph.

4. Classical 4th order Runga-Kutta. Find the value of $y(t)$ at t=2.5 using a single iteration of the classical 4th order Runga-Kutta formulation. Next, find the analytical solution to the differential equation and determine the absolute error. *(hint: separation of variables)*

$$\frac{dy}{dt} = (1+t)\sqrt{y} \text{ ,where } \quad y(0) = 1$$

5. Convert the following differential equation into state-space form and determine the value of x at t=*4.0* seconds if the input is $u(t)$=*0* using the classical 4th order Runga Kutta if all initial conditions are 1.0. Compare the results using 1 and 4 steps. You can complete this by hand or using MATLAB code of your own devising.

$$4x + 0.5\dot{x} + 3\ddot{x} + \dddot{x} = u(t)$$

6. Use ODE45() to simulate the behavior of a Mass-Spring-Damper system(s) given the parameters k=[2, 4], b=0.25, and m = 4.0. Assume there is no input force, but the system starts with a displacement in the x direction of 0.0 and an initial velocity of 4.0. Turn plots (using subplot to create a 2 x 2 set of plots) of the systems' position and velocity for the following time periods: 1, 5, 30, and 60 seconds that illustrate the changes that occur when using different springs. From the graph estimate the natural frequency of the systems.

Finally turn in one plot showing the position after 30 seconds where you modify other system parameters. Briefly discuss how the changes in physical properties impacted the system behavior.

7. The solution to $\dot{x} = \alpha x$ is $x(t) = x_o e^{\alpha t}$. Stability (the system settling on finite value) as time goes to infinity depends on the real part of α. To demonstrate this, use the Taylor series expansions for sin, cos, and the exponential to show that

$$e^{i\alpha t} = \cos(\alpha t) + i\sin(\alpha t).$$

Based on this proof, discuss why for any complex number α (i.e. $\alpha \in \mathbb{C}$) system stability is only dependent on its real component.

8. From the website for the Astrom and Murray text, download the necessary files to replicate the cruise controller found in Figure 1.13. For your convenience, the necessary files are:

1. cruise_response.m (Links to an external site.)
2. cruisedyn.m (Links to an external site.)
3. aminit.m (Links to an external site.)
4. amprint.m (Links to an external site.)
5. amsetup.m (Links to an external site.)

These should all be in your working directory in order for the simulation to work. First, review the code and briefly describe the purpose of each file. Second, using trial and error, change the parameters of the control law so that *(if possible)* the overshoot in speed is not more than 1 m/s for a vehicle with a mass of 2500kg. Second, for a car with a different weight, determine a different performance parameter than overshoot, and modify the parameters to see if how small you can get the selected performance value.