EAS 596, Fall 2019, Final Exam, Part II
Due 11:59 PM Sunday Dec. 15, 2019, submitted to UBlearns
Total Possible Points: 50

Work **all** problems. Please read all directions carefully. You may use course notes, books, internet search, and tools such as Matlab or Python. If you use external references such as journal papers, books, or internet search results, be sure to cite them accordingly. **Consultation, either in-person or virtually, with anyone other than Dr. Dargush or Dr. Salac is STRICTLY FORBIDDEN.** All work submitted must be your own! Submit all code written and one PDF document with the requested answers. Make sure you code is well organized. Points will be deducted if the logic behind your code is not apparent.

Any copying of work, consultation with others, or providing assistance to other students will result in a grade of **ZERO** for the **ENTIRE** exam.

Your code should **not** produce any output other that those specifically requested. Do **not** use any `close all`, `clear all`, or `clc` commands in any of your functions or scripts. *You will be deducted points if these directions are not followed!*

Problem 1 (15 pts.)

Consider the following initial value problem:

$$\dot{y} = \begin{cases} y\left(-2t + \frac{1}{t}\right) & t \neq 0 \\ 1 & t = 0 \end{cases},$$
$$y(0) = 0,$$

which has a solution of $y(t) = te^{-t^2}$. We wish to get an approximation for the solution at $t = 1$.

a) (2 pts) Write a function `yp = ubitname_Final_p1a(t, y)` which takes the time `t` and current solution `y` and returns the derivative `yp`. Submit the file to UBlearns.

b) (5 pts) Write a MATLAB script called `ubitname_Final_p1b` which approximates the solution of the initial value problem using `ode45` in Matlab and the function written previously. Use the defaults of `ode45`. Write the numeric solution to at least 5 decimal places (you may round) and the associated error on the submission sheet and submit the script to UBlearns.

c) (5 pts) Write a function `[t, y] = ubitname_Final_p1c(N)` which takes in the number of time steps `N` and returns the time vector `t` and the solution vector `y` using the explicit Improved Euler method. When creating the time vector use the code `t = linspace(0, 1, N)`. Submit the file to UBlearns.

d) (3 pts) Determine the minimum number of time steps needed to obtain the an error at $t = 1$ no larger than that provided by `ode45`. On the submission sheet compare this number of time steps to that by `ode45`.

Problem 2 (35 pts)

The goal of this problem is to study the behavior of a simple mechanical system. Consider the system shown in Figure 1. The bars are rigid and connected to a piston p. The angle θ_1 will be given a function of time, but it will have noise. You will use the input $\theta_1(t)$ to compute the resulting $\theta_2(t)$ and $x(t)$. Then, using this information, you will compute the velocity and acceleration of the piston. Because the input $\theta_1(t)$ has noise, you will solve the problem two different ways: using the data directly and developing a curve fit for $\theta_1(t)$ and then use the curve fit to solve the problem. The position of the point p is given by the equations

$$x = (L_1 \cos\theta_1 + L_2 \cos\theta_2) \tag{1a}$$
$$h = (L_1 \sin\theta_1 + L_2 \sin\theta_2) \tag{1b}$$

The unknowns are θ_2 and x; you may assume θ_1, L_1, L_2, and h are given. For the remainder of the problem, take $L_1 = 0.6$ m, $L_2 = 0.8$ m, $h = 0.1$ m, and the mass of the piston as 10 kg.

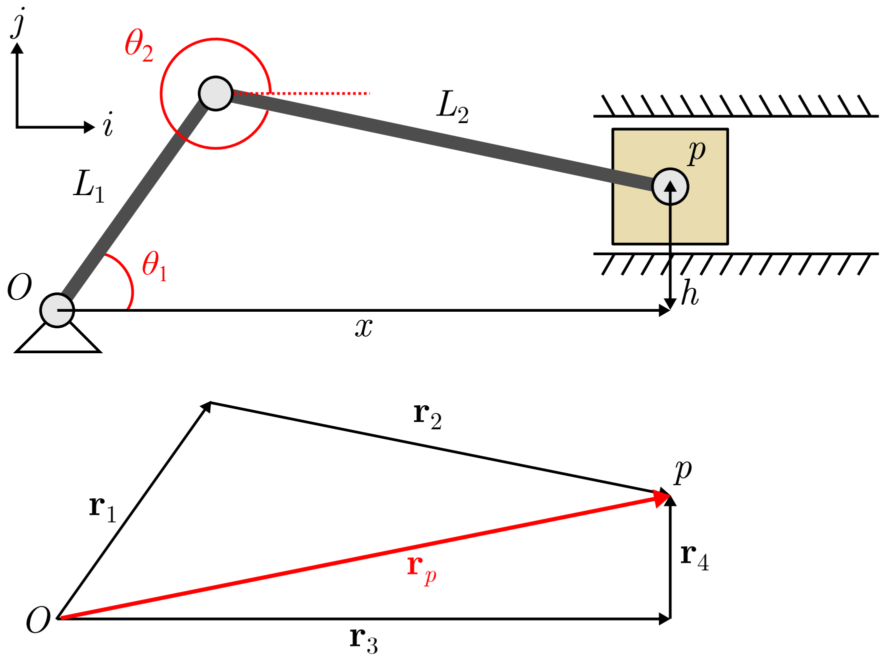

Figure 1: A schematic of a simple piston mechanism. The vectors correspond to the location of the piston p is illustrated.

a) (2 pts) Cast this problem as a root problem. Write this root problem on the submission sheet.

b) (5 pts) Write a function `[x, theta2] = ubitname_Final_p2b(theta1, L1, L2, h)` that takes in a vector `theta1` and scalars `L1`, `L2`, and `h` and returns the values `x` and `theta2` for each value in `theta1`. You must use a nonlinear Newton solver that you have written yourself to solve for each `x` and `theta2`.
Comment: You are allowed to use the \ operator.
Suggestion: in the file ubitname_Final_p2b.m include at least two functions, the main function `ubitname_Final_p2b` and another function which takes as input a specific value of `theta1` along with `L1`, `L2`, and `h` and performs the Newton solve, returning the value `x` and `theta2` for that specific `theta1`. You may include additional functions in that single file if you would like.
Hint: Make sure that your initial solutions for the Newton iteration and the results are physically realistic!
Submit your file to UBlearns.

c) Write a MATALB function `ubitname_Final_p2c(t, theta1, L1, L2, h)` that takes in vectors `t` and `theta1` and scalars `L1`, `L2`, and `h`. This function will perform the following tasks:

   i) (2 pts) Uses the function you previously wrote to determine x and θ_2.

   ii) (2 pts) Computes the velocity, $v(t)$ of the piston using the computed x values. Use a forward finite difference method when applicable and backward finite difference when it is not applicable.

   iii) (2 pts) Computes the acceleration, $a(t)$ of the piston using the computed v values. Use a forward finite difference method when applicable and backward finite difference when it is not applicable.

   iv) (2 pts) Create and display a single MATLAB figure with the following five subplots:

   - θ_1 vs. t,
   - θ_2 vs. t,
   - x vs. t,
   - $v(t)$ vs. t, and
   - $a(t)$ vs. t.

   All subplots must be properly titled and have appropriate axes labels.

   Submit your file to UBlearns.

d) Write a MATLAB script `ubitname_Final_p2d` that does the following:

   i) (2 pts) Load the time series of $\theta_1(t)$ given in `theta1.mat` (use the command `load('theta1.mat')`). This will load two one-dimensional arrays: `t` contains the time while `theta1` contains the values of θ_1.

   ii) (2 pts) Uses your prior function to produce plots of θ_1 vs. t, θ_2 vs. t, and x vs. t, along with $v(t)$ vs. t and $a(t)$ vs. t.

   Submit your file to UBlearns.

e) (5 pts) Now, assume that $\theta_1(t)$ should obey a function of the form

$$f(t) = c_1 \frac{t}{t + c_2}$$

   Write a MATLAB function `[f] = ubitname_Final_p2e(t, theta1)` that takes in a time-vector `t` and θ_1 vector `theta1`. Internally the function will determine the coefficients c_1 and c_2 and then uses those coefficients to return the fitted values `f`, which is the function $f(t)$ evaluated at the times given by `t`. Submit your file to UBlearns.

f) Write a MATLAB script `ubitname_Final_p2f` that does the following:

   i) (2 pts) Load the time series of $\theta_1(t)$ given in `theta1.mat` (use the command `load('theta1.mat')`). This will load two one-dimensional arrays: `t` contains the time while `theta1` contains the values of θ_1.

   ii) (2 pts) Uses your function `ubitname_Final_p2e` to determine the fitted values for θ_1.

   iii) (2 pts) Uses the fitted values and your prior function to produce plots of θ_1 vs. t, θ_2 vs. t, and x vs. t, along with $v(t)$ vs. t and the $a(t)$ vs. t.

   Submit your file to UBlearns.

g) (5 pts) On the submission sheet, comment on the differences between result using the original data and the fitted data, particularly with respect to the velocity and acceleration.