Aerospace Software: Midterm Examination

Due Date: Saturday, March 24 at 11:59 PM

This is a take-home exam. It will be delivered at 11:59 PM on Saturday, March 17, 2018 and will be due at 11:59 PM on Saturday, March 24, 2018. The exam is open-book, and you may reference the class notes, your own homework solutions, the posted homework and quiz solutions, and any additional material included on the course website. You may also reference the MIT OpenCourseWare notes for "Practical Programming in C" and the MATLAB help guide. You may not, however, consult with your classmates, nor may you reference code from external sources (including the internet) other than those previously mentioned. Exam submissions will be compared with submissions from other classmates as well as an online repository to ensure this policy is followed.

Name:

On my honor as a University of Colorado at Boulder student I have neither given nor received unauthorized assistance on this work.

Signature:

Submission Procedure: Your submission should consist of the following:

- The MATLAB code (including all necessary M-files) designed in Part 1.1.
- The MATLAB M- and fig-files associated with the GUI designed in Part 1.2.
- The C code (including all necessary c- and h-files) designed in Part 2.1.
- The Bash script designed in Part 2.2.
- The text file resulting from running the Bash script in Part 2.2.

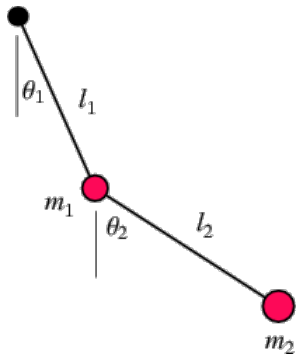
Upload copies of the above items to your private git repository in the directory:

`Midterm/Submission`

In addition, please sign the pledge on the front page of this exam, either scan or take a picture of this signature, and include it with your submission. **An exam will not be graded if this signed pledge is not included with the submission.**

Part 1: MATLAB Programming (60 Points)

Background:
In this part of the exam, you will design a MATLAB program for simulating the trajectory of pendulum bobs attached to rigid massless wires as depicted below, and a corresponding Graphical User Interface (GUI) for user-friendly implementation.



As observed in the diagram, pendulum bobs with masses $m_1 = 2$ and $m_2 = 1$ (kilograms) are attached by rigid massless wires of lengths $l_1 = 1$ and $l_2 = 2$ (meters). The angles the two wires make with the vertical direction are θ_1 and θ_2. The positions of the bobs (x_1, z_1) and (x_2, z_2) are given by

$$x_1 = l_1 \sin(\theta_1) \tag{1}$$
$$z_1 = -l_1 \cos(\theta_1) \tag{2}$$
$$x_2 = l_1 \sin(\theta_1) + l_2 \sin(\theta_2) \tag{3}$$
$$z_2 = -l_1 \cos(\theta_1) - l_2 \cos(\theta_2) \tag{4}$$

In the absence of frictional force, there is only one type of force acting on the bobs, which is a gravity force pointing downward. Recall that the gravitational potential energy of a given mass m (kilograms) at relative height z (meters) is given as mgz where $g = 9.81$ meters per second squared. The gravitational potential energy is essentially the work of a free falling object.

The total potential energy stored in the system of the pendulum bobs is found by combining the potential energy of each mass as follows

$$\begin{aligned} V &= m_1 g z_1 + m_2 g z_2 \\ &= m_1 g(-l_1 \cos(\theta_1)) + m_2 g(-l_1 \cos(\theta_1) - l_2 \cos(\theta_2)) \\ &= -(m_1 + m_2) g l_1 \cos(\theta_1) - m_2 g l_2 \cos(\theta_2) \end{aligned} \tag{5}$$

The total kinetic energy of the system is also found by combining the kinetic energy of each

mass as follows

$$\begin{aligned} K &= \frac{1}{2}m_1 v_1^2 + \frac{1}{2}m_2 v_2^2 \\ &= \frac{1}{2}m_1(\dot{x_1}^2 + \dot{z_1}^2) + \frac{1}{2}m_2(\dot{x_2}^2 + \dot{z_2}^2)^2 \\ &= \frac{m_1 + m_2}{2}l_1^2\dot{\theta_1^2} + \frac{1}{2}m_2 l_2^2\dot{\theta_2^2} + m_2 l_1 l_2 \dot{\theta_1}\dot{\theta_2}\cos(\theta_1 - \theta_2) \end{aligned} \tag{6}$$

where the dot symbol here denotes differentiation with respect to the time t.

The equations of motion for these two pendulum bobs can be obtained by employing the Lagrangian formulation of classical mechanics, specifically by using the Lagrangian function $L = K - V$ defined to be the difference between the kinetic and potential energies expressed in terms of locations and velocities of objects in the angular coordinate system (θ_1, θ_2) as

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{\theta_1}}\right) - \frac{\partial L}{\partial \theta_1} = 0 \tag{7}$$

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{\theta_2}}\right) - \frac{\partial L}{\partial \theta_2} = 0 \tag{8}$$

Evaluating (7) and (8) and reorganizing terms yields the following pair of second-order, ordinary differential equations governing the motion of pendulum bobs.

$$(m_1 + m_2)l_1\ddot{\theta_1} + m_2 l_2 \ddot{\theta_2}\cos(\theta_1 - \theta_2) + m_2 l_2(\dot{\theta_2})^2\sin(\theta_1 - \theta_2) + g(m_1 + m_2)\sin(\theta_1) = 0 \tag{9}$$

$$m_2 l_2 \ddot{\theta_2} + m_2 l_1 \ddot{\theta_1}\cos(\theta_1 - \theta_2) - m_2 l_1(\dot{\theta_1})^2\sin(\theta_1 - \theta_2) + m_2 g\sin(\theta_2) = 0 \tag{10}$$

Note that these two equations depend on four unknowns, so these can be reformulated as a system of four first-order ordinary differential equations.

$$\frac{d\mathbf{y}}{dt} = \mathbf{f}(t, \mathbf{y}) \tag{11}$$

where $\mathbf{y} = \begin{bmatrix} \dot{\theta_1} \\ \dot{\theta_2} \\ \theta_1 \\ \theta_2 \end{bmatrix}$, and the right-hand-side (RHS) of this equation $\mathbf{f}(t, \mathbf{y})$ can be obtained by reorganizing (9) and (10) into the form given by (11).

Part 1.1: MATLAB Program Design (30 Points)
Based on the preceding discussion, write a MATLAB function that numerically integrates the system of four first-order ordinary differential equations given in (11) for a given set of initial conditions `y0` from $t = 0$ for an interval specified by `tspan`. Your MATLAB function should take the following form:

```
function [t,y] = simulate_pendulum(tspan, y0, l1, l2, m1, m2)
```

which takes as input the masses of two bobs `m1`, `m2` and the length of wires `l1`, `l2`.

Part 1.2: MATLAB GUI Design (30 Points)
Using GUIDE, write a MATLAB GUI to implement the function given in Part 1.1. Your GUI should allow the user to set:

- The initial angles of the pendulum bobs in radians (θ_1, θ_2 at $t = 0$).
- The initial angular velocities of the pendulum bobs in radians per second ($\dot{\theta_1}, \dot{\theta_2}$ at $t = 0$).
- The mass of the pendulum bobs in kilograms (m_1, m_2).
- The length of the pendulum wires in meters (l_1, l_2).
- The integration time in seconds.

and then compute the vectors `t` and `y` as detailed in Part 1.1. Following the aforementioned computation, your GUI should also allow the user to plot either:

- The angle of the first bob θ_1 (in radians) versus time (in seconds).
- The angle of the second bob θ_2 (in radians) versus time (in seconds).
- The vertical position z (in meters) versus horizontal position x (in meters).

The plot should have proper axis labels.

Some suggested parameter values for testing are $l_1 = 1, l_2 = 2, m_1 = 2$ and $m_1 = 1$, and zero initial angular velocities.

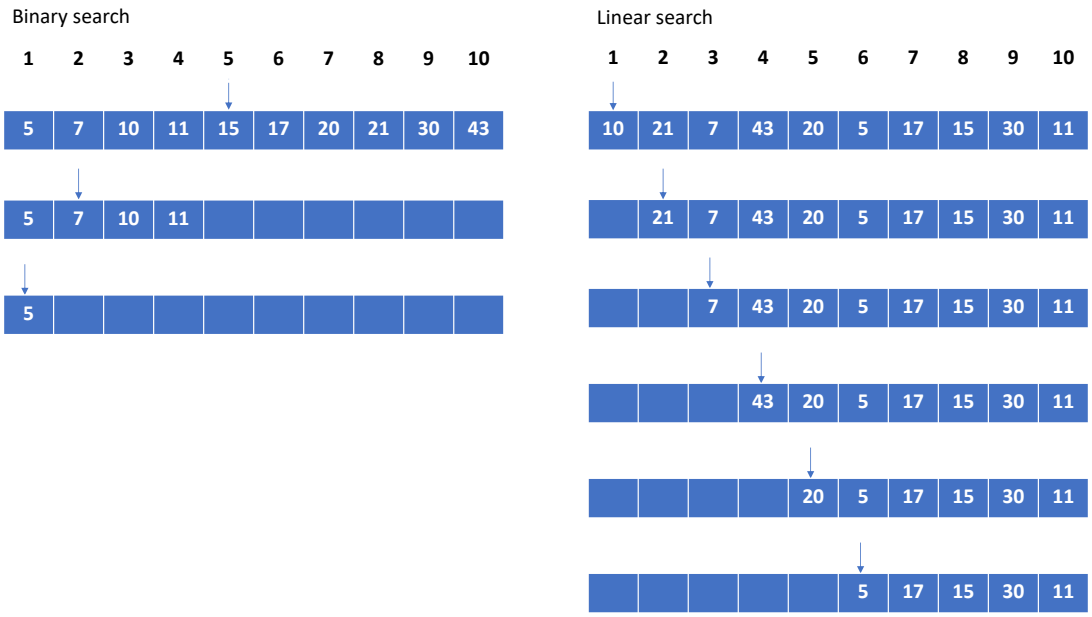
Part 2: C Programming (40 Points)
Background:
In this part of the exam, you will design a C program that searches through a list of integer numbers in an array and finds a target number by using one of the following two algorithms.

- **Linear Search** is a sequential search algorithm that scans one item at a time, starting from the first item in the list, until the position of a target value is found. The algorithm works for both sorted and randomly sequenced arrays.
- **Binary Search** is a search algorithm that finds the position of a target value within a sorted array. Binary search compares the target value to the middle element of the array. If the target value does not match the middle element, the half in which the target value cannot lie is eliminated and the search continues on the remaining half until it is successful.

For instance, the search steps to find the target number of `5` in the following arrays

```
S[10] = {5, 7, 10, 11, 15, 17, 20, 21, 30, 43}
R[10] = {10, 21, 7, 43, 20, 5, 17, 15, 30, 11}
```

can be visualized as follows.



In this example, the position of the target number is `1` in the randomly sequenced array `R`, and the position of the target number is `6` in the sorted array `S`. The position of the target number is here different from an index of the array element for `5` in C code.

Part 2.1: C Program Design (30 Points)
Write a C program named `search.c` which (i) takes as command-line arguments a file name and a target number, (ii) opens the file and reads in one-dimension data array, (iii) searches for the specified target number in the array, and (iv) writes out the search result (position) in an output file. If the output file exists, the search result output should be appended to the end of file without overwriting the previous search result. Your C program will have to display (and write out) an error message if the target number is not found in the array. Furthermore, your C program should be able to handle arbitrarily sized arrays stored in the heap memory segment through ***dynamic memory allocation***. Use the linear search algorithm for a randomly sequenced array, and the binary search algorithm for a sorted array.

A call to your program should take the form:

```
./search Array1.in 5
```

where `Array1.in` contains 1-dimensional array data. The output file of your program should be named `Array1.out`, containing the output string:

```
The position of 5 is 6
```

You will be provided odd-numbered files `Array1.in Array3.in Array5.in` containing randomly sequenced arrays, and even-numbered files `Array2.in Array4.in Array6.in` containing sorted arrays.

Part 2.2: Bash Scripting (10 Points)
Write a Bash script named `run_search.sh` to batch-process all the input array files in a user-specified directory for target number `17`, `56` and `68`. For instance, if a call to your script takes the form:

```
./run_search.sh /home/username/midterm/
```

the directory `/home/username/midterm/` should contain the `Array*.in` files stored within the zip file `Arrays.tar.gz` found in the Shared git repository in the directory:

```
Midterm/Materials
```

Include the resulting output file with your submission.

Bonus: Bash Scripting (10 Points)
Write a Bash script named `pyramid.sh` using a for-loop to print the following pattern for a given user-specified command line argument. You can assume that the user input is integer. For instance, if a call to your script takes the form:

```
./pyramid.sh 9
```

it should display the following pattern:

```
        1
       2 2
      3 3 3
     4 4 4 4
    5 5 5 5 5
   6 6 6 6 6 6
  7 7 7 7 7 7 7
 8 8 8 8 8 8 8 8
9 9 9 9 9 9 9 9 9
```

The Bash script should terminate with an error message `Enter number between 1 and 9`, if the user-specified number is smaller-than-or-equal-to 0 or greater-than-or-equal-to 9. Furthermore, the script should sum up all the numbers displayed in the pattern and append it to a file named `output.txt`. Run the script for an input from 1 to 9, and submit `output.txt` with your submission.