ROCO218 CONTROL ENGINEERING 2018

ASSESSED COURSEWORK PART 2: SFC OF A REAL INVERTED PENDULUM

IMPORTANT NOTICE

Submit this report to the DLE by: 17th May 2018 (4pm)

Please only submit a single report for this coursework in PDF format. You **must** include your student number in a header on every page of the report and **also include** your student number **in the document filename**! Please use the supplied template format for your report and importantly, please use the section heading used in this coursework sheet.

This coursework should contain a few pages of explanation as required (although it can be more than this is necessary) and a set of images showing the plots requested by the individual parts of the practical exercises, as well as embedded equations and Matlab code that you developed to implement your solutions.

The report ***must*** be a single **stand-alone document**. Please embed images in-line in the report. In order to show video demonstrations if you have any, please use Internet links to videos that you have been uploaded to YouTube.

You can work in pairs for the practical sessions but submit a report **individually**!

Introduction

This assignment system involves the analysis of an inverted pendulum mounted to a motorized cart. Such a system is unstable without control. Balance in the inverted configuration can only be achieved by displacing the cart.

This first part of the coursework is to give you experience of:

- Deriving the state space model of an inverted pendulum
- Designing a direct state feedback controller to stabilize the pendulum in its inverted configuration
- Designing state feedback control using state estimated using an observer.
- Implementing a controller simulation that runs in Matlab.
- Implementing a controller solution using Euler integration that could easily be ported to run on a microcontroller.

[70 marks]

The second part of the coursework is to give you experience of implementing a state feedback controller and observer to control a real inverted pendulum:

- Augment the state space model to include a state to represent cart position and redesign the state feedback controller gains using this extra state
- Implement the augmented system using Euler integration so it can easily be ported to run on a microcontroller.
- Implement the augmented system using an Arduino Mega microcontroller to balance a real inverted pendulum

[30 marks]

8. Augment positional state into your state space model

[10 marks]

- We need to control cart position as well as angle and angular velocity since otherwise it might never stop moving
- Then we can control cart position to remain at its initial location
- To do so we need add a third state x_3 to our state vector represent cart position
- Since the control signal is cart velocity, the differential of x_3 is simply given by the input velocity control signal
- Show that the state space model of the system is now given by

$$\frac{d}{dt}\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 \\ -a_2 & -a_1 & 0 \\ 0 & 0 & 0 \end{bmatrix}\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} + \begin{bmatrix} b_0 \\ -a_1 b_0 \\ 1 \end{bmatrix} v_c$$

$$y = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix}\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}$$

where

$$a_1 = \frac{\mu}{\left(I + ml^2\right)} \qquad a_2 = \frac{-mgl}{\left(I + ml^2\right)} \qquad b_0 = \frac{ml}{\left(I + ml^2\right)}$$

9. Implement the augmented state feedback controller

[10 marks]

- Implement the augmented state feedback controller using Euler integration
- Use the Luenberger observer to only estimate x_1 and x_2. Directly compute the new positional state x_3 by integrating the input control velocity.
- Show that in simulation the system performs as well as it did before.

10. Run the inverted pendulum demos

- A suite of example programs is available in the ROCO218/223 coursework folder.
- Download the zip file ROCO218Arduino.zip
- Expand the file. It contains 6 directories containing Arduino applications
- There is also a libraries directory.
- Make sure the Arduino integrated development environment is installed on your PC or Mac
- Copy all folders in the libraries directory into the default Arduino libraries directory on your PC.

All the Arduino application programs display a help menu if when the character 'h' is entered on the serial monitor. These application programs are in the following directories.

1. ProcessControlTest: This implements a menu driven command manager running over the serial monitor.
2. SimpleHallSensorTest: This implements end stop detection using the Hall sensors.

3. SimpleEncoderTest: This reads to encoder value and print out via the serial output. If you select the plot monitor you can generate a plot of the signal versus time.
4. StepperVelocityTest: This uses a velocity control stepper motor class to drive the carriage back and forth sinusoidally.
5. SFCInvPendTX: This implements SFC control of an inverted pendulum with the state vector augmented with cart position.

6. SFCIPROCO218: This is a special version of the inverted pendulum control program that you are required to finish implementing!

- Connect up the Arduino pendulum unit
- Connect the host PC to the Arduino unit by means of a USB cable.
- Connect up the power lead to the special pendulum bench power supply set to 18V and 2A. Connect the red wire to the positive terminal and the black wire to the 0v terminal. Take care not to connect this leads the wrong way around since this will destroy the controller!
- Please do not use more than a supply of 18V as this will also destroy the controller! Do not use the standard electronic workshop power supply as they can generate voltages that could destroy the unit.

- Run all the demo programs 1 – 5 to gain experience of operation of the inverted pendulum. You will not be marked on this, it's just so you see how it operates!
- Refer to the inverted pendulum manual for more information.

11. Implement the augmented state feedback controller on the Arduino Mega

[10 marks]

Here you will now implement your augmented state feedback controller on a real microcontroller to control a real system!

- The main Arduino program SFCIPROCO218 is an uncomplete version of the SFCInvPendTX demo program.
- Look at lines 93-105 as shown below:

```
93 // ENTER YOUR VALUES FOR THE SYSTEM MATRICES HERE
94 // system matrix definitions
95 double A[3][3] = {{xxx, xxx, xxx}, {xxx, xxx, xxx},  {xxx, xxx, xxx}};
96 double B[3] = {xxx, xxx, xxx};
97 double C[3] = {xxx, xxx, xxx};
98
99 // ENTER YOUR VALUES FOR THE SFC GAINS HERE
100 // SFC gains
101 double K[3] = {xxx, xxx, xxx};
102
103 // ENTER YOUR VALUES FOR THE OBSERVER GAINS HERE
104 // observer gain just for theta and thetadot
105 double L[2] = {xxx, xxx};
```

- You first need to enter your own values for the system matrices A, B, C and D.
- The enter the SFC gains K
- Then enter the Luenberger observer gains L

After you have entered all the values, check the programs compiles before proceeding to the second task.

The main Arduino program SFCIPROCO218 uses the cpp class CSFCROCO218.cpp with header CSFCROCO218.h to implement state feedback control.

- The member function ComputeSFC needs to calculate the motor command on the basis of the pendulum angle. It is provided empty, as shown below between lines 99 to 125:

```
99 ////////////////////////////////////////////////////////////////
100 // compute the SFC
101 // given output angle of pendulum y and the current time
102 // returns the motor command u
103 double CSFCROCO218::ComputeSFC(double y, unsigned long theTime)
104 {
105   // control value - stepper motor speed
106   double   u = 0.0;
107
108   // PUT YOUR OWN ComputeSFC FUNCTIONALITY IN HERE
109
110   // Calculate time since last update
111
112   // compute control variable u
113
114   // calculate observer correction term
115
116   // update the state estimates for theta and thetaHat
117
118   // record variables
119
120   //  use control velocity from input to update position of cart
121
122
123   // return motor command
124  return (u);
125 }
```

- You now need to enter your implementation of state feedback control using a Luenberger observer within the empty ComputeSFC function
- After you have implemented the SFC, upload the program to the Arduino and see if your inverted pendulum works!
- Include a link to video of the balancing pendulum in your report