Project Report for MAE280A Linear Systems Theory, Fall 2020 (Northern Hemisphere): due electronically by Friday December 11 by midnight via Gradescope (preferred) or Canvas.

Introduction We shall be considering the control of the Mobile Inverted Pendulum (MiP) as developed by UCSD Professor Tom Bewley, his Coordinated Robotics Lab, and commercialized by WowWee Robotics and available for $50 as eduMIP here.

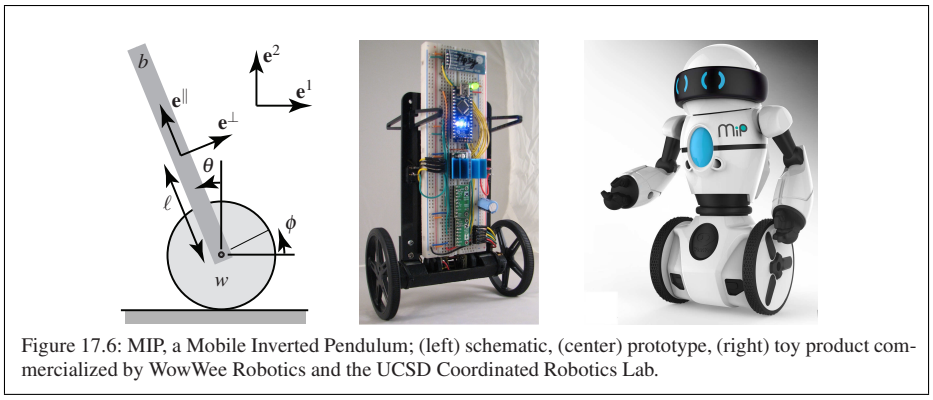

Figure 17.6: MIP, a Mobile Inverted Pendulum; (left) schematic, (center) prototype, (right) toy product commercialized by WowWee Robotics and the UCSD Coordinated Robotics Lab.

Figure 1: MiP figure from Tom Bewley's *Numerical Renaissance* draft on November 22, 2015. This is available on the class Canvas site in the *MIP Files for Final Report* module.

You now are going to try out your controller design skills on a fairly high-fidelity simulation of the MiP with a number of real-world nasty aspects: nonlinearity of the dynamics, saturation of the control signal, noisy measurements. There are a number of files on the class Canvas Module *MIP Files for Final Report.*

These are as follows:

NR.pdf – A version of Tom Bewley's *Numerical Renaissance* with lots of resources for MAE280A and a full model derivation of the MIP dynamic model.

ZhuZhuo_Master_Thesis.pdf – Zhu Zhuo's MS Thesis form 2017; Mauricío de Oliveira Chair, Tom Bewley (Co-Chair), me as his committee.

ZhouMiPparameters.m – Calculation of Zhu Zhuo's model parameters as near as I can do ... and I tried many ways. Thanks to Matthew Pearson for providing his calculations which get closer than most of mine. This file also contains Zhu's second-order discrete state-space controller parameters.

Please note that the nonlinear MiP simulation in Simulink requires the quantities a, b, c, d, e and j to be in the workspace before the simulation can run. Zhu's controller matrices $ALCBKDmIP$, $LLDmIp$, $KLDmIp$ and $DLDmIp$, which define the state-space controller, are also computed by this m-file and are needed for the "digital MiP controller" block in Simulink. Note: his controller has the wrong state dimension to run with our model.

Running this m-file places all of these matrices in the workspace, type `ZhouMiPparameters` after the `>>` prompt. Simulink reads the data from the workspace.

nlMiP.slx is a 2020b simulink file containing the nonlinear MiP block together with my digital state-space feedback controller and initial conditions. I took no care to design this controller. This contains the following items.

- capacity to set the initial conditions on the MiP states (or on the controller states if you knew what was needed). You can do this by (i) clicking on the nlMiP subsystem to open the block, (ii) clicking on an integrator, say 'theta,' and (iii) typing in an initial condition in the appropriate box. The units of states are radians or radians

per second. Then save the system, return to the outer system by clicking on the back arrow in the model browser window.
  - At present, the integrator `th`, whose output is signal theta, has initial condition thetaic in the workspace. I set it to 0 then 0.01 just to check the operation of the model. You can change this in the MATLAB workspace.
  - Integrator `thdot`, whose output is thetadot, is set to 0. You can open the integrator and change it.
  - Integrator `phdot`, whos eoutput signal is phidot, is set to 0. Feel free to alter it.
- zero-order-hold element which converts the discrete signal u_k from the digital controller to a continuous-tie signal by keeping the signal constant across the sample period.
- a saturation element which limits the applied control voltage to $\pm$6V as in the actual MiP power supply.
- noise signals which can be added to either or both measurements going to the controller. The noise power is roughly the amplitude of the noise squared. So be careful not to set it too high.
- scopes to allow you to see the signals from the simulation.
- outputs to the MATLAB workspace.

nlMiP_201X.xls the same Simulink model in versions amenable to Simulink from year 201X. This allows people with earlier versions to simulate the MiP and controllers.

Your task is to design your own discrete linear state-estimate feedback controller for MiP using the things that you have learnt from MAE280A this quarter. This design needs to be documented properly with tests for equilibrium and an investigation of the quality of the control performance that you are seeking and achieving. You may use the MiP state, and control output signals from the Simulink block to examine the quality of your and control.

You should use MAE280A linear control design ideas but test out and compare your resulting controlled performance between linear and nonlinear cases. Feel free to kick the system around using the excitation and/or noises and make sure to look at the estimator performance when you do this.

MATLAB Assistance

If you do not have your own version of MATLAB, then you should be able to use MATLAB-online:
`https://www.mathworks.com/products/matlab-online.html`
This will require you to use your UCSD login credentials to operate and it should include Simulink. You can drag and drop files to the webpage.

If you have difficulty accessing MATLAB and Simulink, especially over the Thanksgiving break, please feel free to limit your investigations to the linear case using `lsim`. But try to get it working.

The grading of this homework will emphasize the extent to which you show that you understand what you are doing as a design problem. So the report will be more important than the results.

Tasks

0. Get used to playing with Simulink by running my controller and playing around with the initial conditions, the process noise and the saturation bound.
1. Start by considering your own state-estimate feedback controllers. You can do this in continuous time and then convert or do the design in discrete time.
2. Convert them to discrete-time using MATLAB's `c2d` command with the zero-order-hold option. Remember the sample rate is 100Hz.

3. Create matrices `Amine, Kmine, Lmine, Dmine` with names corresponding to those used in the Simulink MiP controller block. These are the same as used in my example below. Do not overwrite a-j though.

4. Try to redesign your controller for improved performance in terms of stability, response time, noise rejection, region of attraction, etc.

5. (i) Make sure that the equilibrium is as expected. To do this:
    i. Double-click on the Theta Noise and Phi Noise blocks and set the noise power to zero.
    ii. Double-click on the digital controller box and ensure that the initial condition is zero.
    iii. Double click on the gray NLmIp box to look inside.
    iv. There are three integrators inside – one for each state variable. Double-Click on each and set the initial condition to zero. The controller initial condition is automatically set to zero, unless you feel like changing it.
    v. Run the simulation by clicking the run arrow at the top. Inspect the state variables (which normally are not known to us) and the output of the controller. They should all stay at zero.

   (ii) Start by changing the initial condition on theta (i.e. the theta integrator) – it should be measured in radians – to see how well the linearized model captures the nonlinear behavior. Try to determine a range of allowable theta initial conditions for stability of the nonlinear system.

   (iii) Reset the theta initial condition to zero and turn up the theta noise power. This is the noise in the measurement of theta but not in theta itself. What is the limiting value of this noise power before we lose stability?

   (iv) Vary the observer design to manage the response to the large noise power.

6. It is easy to test many different controller designs using this setup. Try to develop one that looks better than Zhu's.

7. Document, with plots and brief design explanation, your progress to your favorite controller paying particular attention to the state estimator.

8. **[Bonus]**$^{\frac{1}{2}}$ (not a serious bonus) Try to see if you can expand the range of stabilization by setting the initial conditions of the controller state inside the controller block. In particular, since the controller states are estimates of the plant state, try setting their initial conditions to the correct values. Does this stabilize a wider range of MIP initial conditions?

9. **[Bonus]** Explain how you would alter your design to drive the controlled MiP by making ϕ do what we want.

10. **[Bonus Bonus Bonus]** Have fun!

Here is an example of how I got started. This is the controller in the Simulink model file. I have added some comments. each of these commands was typed in the MATLAB workspace window, except for the RUN command in Simulink.

```
>> ZhuoMiPparameters  % load the parameters for the model
>> mipC=ss(A,B,C,D)  % Zhou's model is continuous-time
>> mipD=c2d(mipC,0.01)  % Convert to discrete-time for control design and implementation
>> [AD,BD,CD,DD]=ssdata(mipD)  % Pull out the discrete-time matrices
>> Kmine=place(AD,BD,[.95;.94;.93]) % Design the state feedback
>> Lmine=place(AD',CD',[.95;.94;.93])' % Design the observer. Note the transposes.
>> Amine = AD-BD*Kmine-Lmine*CD % Construct the A-matrix of the digital controller
>> Dmine=[0 0]  % Construct the D-matrix of the controller
>> eig(AD-BD*Kmine)  % Check that I have the signs correct
>> eig(AD-Lmine*CD)
>> eig(Amine)  % Just being nosey
>> thetaic = .01  % try with a small initial angle (radians)
% run the simulation using the RUN button in Simulink
>> plot(out.theta.Time,out.theta.Data);shg % Plot theta versus time
```

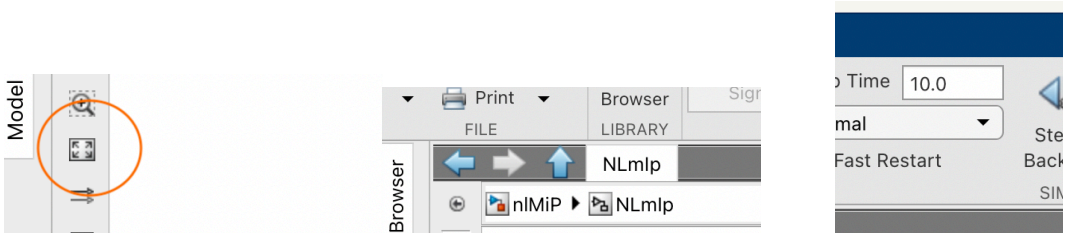

Figure 2: Simulink window showing Recenter/rezoom, return to upper level, and run-time command tools.

Simulink Tricks

If you look at the figure, you will see three things highlighted on the Simulink panel: the button which rescales and recenters the picture when it gets horribly zoomed out or in; where you set the final simulation time, currently set to 10 seconds; and, how to return to the enclosing subsystem.