Lab 5: "Segway" Inverted Pendulum Robot.

Overview

In this lab, you will modify provided MATLAB files to enable your Lego inverted pendulum robot to balance in place. This requires several tasks, listed below, many of which you should already have completed:

- deriving equations of motion,
- measuring or estimating various model parameters,
- finding offsets (biases) for the gyro and accelerometers, and (finally)
- determining a gain vector, K, for state space control.

We expect most people will find gains via MATLAB's lqr function, which solves for the optimal gains to minimize a quadratic cost function of errors in states (weighted by values in Q) and of actuator output (weighted by values in R).

Deliverables: **Each lab group must submit the items listed below in a short report for Labs 5/6 by Dec. 6. A worksheet is provided for most information, but you will still need to provide output from MATLAB, as well, to compare the actual and theoretical closed-loop system responses.**

1. Estimate model parameters (measured by hand) listed on the worksheet on the next page.

2. Complete the m-file provided to derive the equations of motion using the symbolic toolbox in MATLAB and the function fulldiff.m.

3. List the resulting A and B matrices (numeric values) for the modeled plant on your worksheet. Document the open-loop poles: eig(A).

4. Now, run the system with no motor outputs, to estimate offsets (biases) for the accelerometer and gyro by recording outputs when the system is at rest. (i.e., all velocities at zero.) Use these value in your mdl-file, for more accurate sensor readings.

5. Use the lqr function in MATLAB to solve for a vector of gains, K, to stabilize the system. Test K values on the real system. Iterate to improve stability by adjusting Q and R and observing the response. **Once tuned, document the following:** Q and R matrices, K gains, and predicted closed-loop poles, eig(A-B*K). (Hint: You should be able to stabilize the system with Q and R as diagonal matrices. Recall the Q gives costs on squared errors in state; R sets the penalty on control effort squared.)

6. After the real robot is working well, record data for an "impulse response". To do so, quickly tap the robot to provide the impulsive perturbation. Estimate the second-order poles of the true closed-loop robot system. Compare these with the prediction from eig(A-B*K).

   Extra Credit:
   Test how sensitive your LQR solution is to various model parameters. In particular, how does wheel radius affect K?

   Improve model parameters so predicted and actual closed-loop poles match more closely.