Ball and Beam

Objectives

The objective of this lab is to gain experience in the design of control algorithms, taking the ball and beam system as an example. Using an animation program in Matlab, the challenges of controlling the system manually will be observed. Next, a proportional-derivative controller will be designed and evaluated.

Introduction

The ball and beam system is an educational experiment that is fun to watch and play with. A diagram is shown in Fig. 1. A beam is attached to a motor so that its angle θ with respect to the horizontal can be controlled at will. A ball is placed on the beam and is free to roll under the action of gravity (a small channel in the beam may keep the ball from rolling sideways). The distance of the ball from the center of the beam is denoted x. The ball can be placed at any location on the beam, and will stay there if its velocity v is zero and the beam angle is zero.

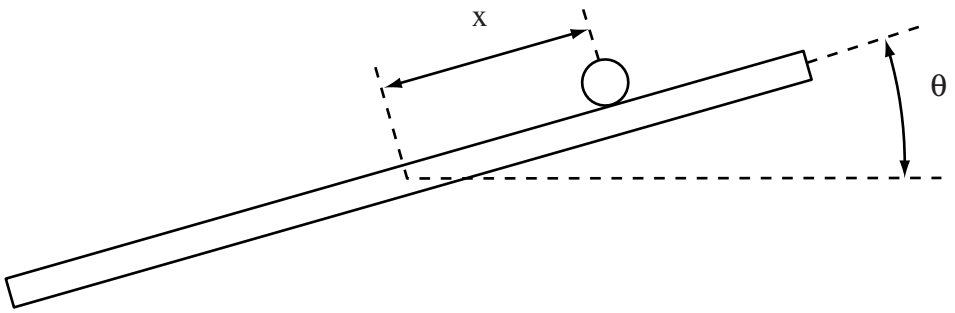

Figure 1: Ball and Beam System

Assuming that the only force acting on the ball is gravity, the movement of the ball is determined by Newton's law

$$m\frac{d^2x}{dt^2} = -mg\sin(\theta) \tag{1}$$

where g is the acceleration of gravity and m is the mass of the ball. A more careful analysis shows that the rotational inertia of the ball adds itself to the translational inertia, resulting in a factor of $5/7$ in the right-hand side of the equation of motion. The resulting state-space

model of the ball and beam system is then

$$\begin{aligned} \frac{dx}{dt} &= v \\ \frac{dv}{dt} &= -\frac{5}{7} g \sin(\theta). \end{aligned} \tag{2}$$

where x is the position of the ball and v is its velocity. In order to apply linear control theory, one assumes that the angle θ is small and one replaces $\sin(\theta)$ by θ. The transfer function of the system is then

$$P(s) = \frac{X(s)}{\Theta(s)} = \frac{k}{s^2}, \qquad \text{where } k = -\frac{5}{7} g \tag{3}$$

The transfer function is the so-called *double integrator*, which is often encountered in control applications. Newton's law, $F = m.a$, generally yields this transfer function if force is the control variable and position is the output variable. Moving a spacecraft with thrusters is a practical example of such a system. Note that the model of the ball and beam system is only approximate. Also, a laboratory testbed may only permit to specify the torque of the motor, rather than its position. The model for such a system will have four states, instead of two. Nevertheless, the simplified model (3) exhibits the most interesting part of the dynamics of the ball and beam system, and is tractable for manual control.

Real-time simulation and visualization

The real-time simulation comes in the form of a Matlab macro (m-file). Visualization is provided through a Matlab figure, and a joystick is used for manual control. The file *bbeam.m* contains the m-file with the simulation of the system. A file called *jstick.dll* must be placed in the working directory of Matlab to provide readings of the joystick commands. It may be necessary to first calibrate the joystick by clicking on *Start/Settings/Control Panel/Game Controllers*. After calibration, typing *ref=jstick* in Matlab returns a Matlab variable *ref* with value between -1 (full left) and 1 (full right). *Please contact the instructor if you want to work on the labs at home, and have trouble using the joystick function.*

The model of the ball and beam system is implemented in *bbeam.m* using an Euler approximation

$$\begin{aligned} x(i) &= x(i-1) + dt \cdot v(i-1) \\ v(i) &= v(i-1) + dt \cdot (-\frac{5}{7} g \sin(\theta(i-1)) \end{aligned} \tag{4}$$

where i is the time instant and dt is the sampling period. The sampling period is set to 0.05s, or a frequency of 20Hz. Timing is obtained in Matlab using the *tic* and *toc* commands. The sampling period is close to the resolution of these commands, so that the joystick input and the visualization do not occur at exactly 20Hz. However, deviations are not normally noticeable.

Visualization is provided in a window using Matlab graphic functions. The window is scaled so that the unit lengths of the x and y axes are identical on the computer screen used

to develop the labs. It may be necessary to adjust the window size to your own computer, and information is given in the code about the parameters that may need to be modified.

The simulation program gives the option of manual control or automatic control. Manual control is immediately available. For automatic control, two m-files must be written: a file called *bbeamc.m* containing the control algorithm, and an initialization file called *bbeamcinit.m.* The initialization file is called once before the simulation starts, while the control algorithm is called at the same rate as the ball and beam simulation. As provided, the program does not store the time histories of the signals. You will need to define arrays in the initialization macro, and store relevant variables in the control macro in order to plot the results of your experiments.

The control signal is the angle of the beam (called *theta* in the code), and is limited to ± 5 degrees. Measured variables are the position of the ball (called *xball* in the code) and the velocity of the ball (called *vball* in the code). The variable t in the code gives the time. In the simulation program, the position of the ball is limited to the length of the beam ($\pm 0.4m$) by setting the velocity to zero when the end of the beam is reached. Friction of the ball rolling on the beam is simulated by setting the velocity to zero if the ball velocity and the beam angle are small enough.

Manual control

The animation program is a good opportunity to have fun and get a sense for the challenges of controlling the double integrator. As an objective, try to move the ball from the left line to the right line as fast as possible. You will notice the importance of accounting for the ball velocity in your strategy. An automatic controller also needs to use this information. For the report, explain the challenges of the manual control problem and describe the strategies that you have developed for control.

Automatic control

The objective of this part of the lab is to develop an automatic controller, and to appreciate its performance compared to manual control. Choose a proportional-derivative controller, so that

$$\theta = k_p e + k_v \frac{de}{dt}, \qquad \text{with } e = x_{REF} - x \tag{5}$$

Assuming that both the ball position and the ball velocity can be measured, and neglecting dx_{REF}/dt, the control law may be implemented simply with

$$\theta = k_p e - k_v v \tag{6}$$

Note that the closed-loop system is a second-order system whose poles can be placed arbitrarily. Choose the parameters k_p and k_v so that the two closed-loop poles are real and equal, with an associated time constant of 0.3 seconds. Explain what would happen if the derivative gain k_v was set to 0.

Implement the control algorithm in the real-time simulation, and generate steps of position reference alternating between $0.3m$ and $-0.3m$ every 6 seconds. Plot the responses of x, v, and θ for 20 seconds. Repeat the experiment with a time constant of 0.2 seconds and explain the overshoot observed in the responses. Show your code and demonstrate the real-time operation to the TA.

Report at a glance

Be sure to include:

- Description of challenges and strategies for manual control.
- Description of PD controller and evaluation. Plots of responses with two controller settings.
- Observations and comments.
- Listing of *bbeamc.m* and *bbeamcinit.m* files.