EE 222/ME 237 Nonlinear Systems: Analysis, Stability, and Control

Course Project: Ball and Beam Trajectory Tracking - Outline

UC Berkeley

Spring 2022

Issued: 4/13
Due: 5/6 11:59 pm

Motivation: In this class, you have learned a few techniques for nonlinear control. This project is an opportunity to put them to work on hardware and design your own controller! The ball-and-beam is a classic example of a control system. In this project, you'll design a controller for this system that tracks certain trajectories and satisfies other performance criteria (like minimizing control cost and safety violations).

Modeling: The ball and beam setup consists of a track on which the metal ball is free to roll. One side of the beam is attached to a lever arm that can be coupled to the load gear of the rotary servo unit. By controlling the position of the servo, the beam angle can be adjusted to balance the ball to a desired position. A schematic of the setup is below:

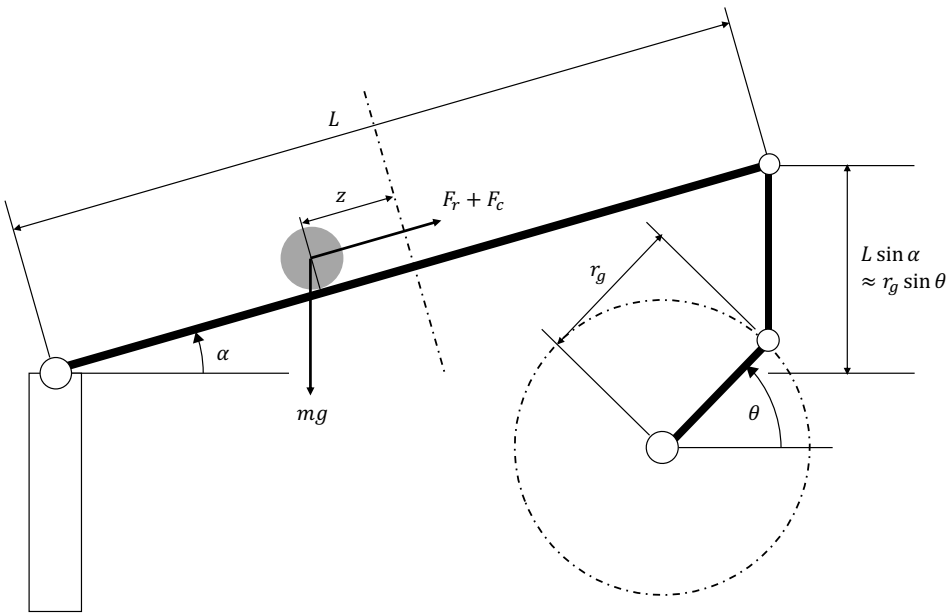

Figure 1: Ball and beam setup

The state of the ball and beam system are given by $x = [z, \dot{z}, \theta, \dot{\theta}]^\top$, where z is the deviation of the ball from the center of the beam, and θ is the angle of the arm of the servo motor,

made with the horizontal axis.

The dynamics of this system can be derived from the kinematics of the links, the equation of motion of the ball, and the motor dynamics of the servo motor. The force applied to the ball consists of the gravity force, centrifugal force $F_c = m\left(\frac{L}{2} - z\right)\dot{\alpha}^2$, and the rolling resistance $F_r = \frac{J_b}{r_b^2}\ddot{z}$, where m is the ball mass, L is the beam length, r_b is the ball radius, and $J_b = \frac{2}{5}mr_b^2$ the moment of inertia of the ball. From this, the equation of motion of the ball is given as

$$m\ddot{z} = mg\sin\alpha - m\left(\frac{L}{2} - z\right)\dot{\alpha}^2 - \frac{J_b}{r_b^2}\ddot{z}. \tag{1}$$

Here, we can assume that the tilt angle of the beam α is small enough so that $\alpha \approx \sin\alpha$. From the kinematics relation $L\sin\alpha \approx r_g\sin\theta$, where r_g is the servo arm length, we can replace $\sin\alpha$ and $\dot{\alpha}$ with

$$\sin\alpha = \frac{r_g}{L}\sin\theta, \quad \dot{\alpha} = \frac{r_g}{L}\cos\theta\,\dot{\theta}. \tag{2}$$

Next, the motor dynamics is given as

$$\tau\ddot{\theta} = -\dot{\theta} + K\cdot V \tag{3}$$

where V is the voltage applied to the motor, τ is the delay constant, and K is the motor constant.

Putting together, we get the system equations

$$\begin{aligned}
\dot{x}_1 &= x_2 \\
\dot{x}_2 &= \frac{5g}{7}\frac{r_g}{L}\sin x_3 - \frac{5}{7}\left(\frac{L}{2} - x_1\right)\left(\frac{r_g}{L}\right)^2 x_4^2\cos^2 x_3 \\
\dot{x}_3 &= x_4 \\
\dot{x}_4 &= -\frac{x_4}{\tau} + \frac{K}{\tau}u
\end{aligned} \tag{4}$$

where $u = V$ is defined as the control input to the system. We give a table of the relevant model parameters below:

r_g	$0.0254\,\mathrm{m}$
L	$0.4255\,\mathrm{m}$
g	$9.81\,\mathrm{m/s^2}$
K	$1.5\,\mathrm{rad/sV}$
τ	$0.025\,\mathrm{s}$

Finally, note that you do not have measurement of the full state. You only have measurement of the ball position z from the ball position sensor that outputs a voltage signal proportional to z, and the servo angle θ from the encoder attached to the motor. If you wish to use the full state information, consider designing your own state estimator or observers.

For more details of the system, please refer to [1]. Note that the verified dynamics might definitely have some modeling errors due to the approximations, forces that are not considered in the equation of motion, and many other factors. You will be able to see the effect of these modeling errors, or model-plant mismatch, when your controller gives different results in the simulation and in the actual experiment.

Problem Definition: You goal is to design a controller that computes the input voltage to the servo motor V that can drive the ball position to a desired trajectory, given as a function in time. The main objective of this controller is to reduce the tracking error and the energy spent by the motor as small as possible. While doing so, you also want to make sure that the controller is operated in a safe region; the main safety constraint considered here is that the ball does not escape the physical limit of the beam. We next describe the metrics that capture the tracking error, energy consumption, and safety constraint violation.

1. **Tracking a reference trajectory:** In the experiment, you will be given an unknown 90 seconds-long reference trajectory, consisting of arbitrary sinusoidal and square waves of the ball position in time. When its profile is a sinusoidal wave, the period will vary between 6 to 10 seconds and the amplitude will vary between 0 to 15 cm. When the profile is a square wave, the range of the period will be same and the range of the amplitude will be 0-10 cm. In both cases, the wave will be symmetric with respect to the center of the beam (meaning that there will be no offset from $z = 0$.) The initial state can be taken arbitrarily, although we recommend to just leave the ball in rest before you run the controller for the sake of convenience. The average tracking error of the resulting trajectory will be evaluated as

$$J_{tracking} = \frac{1}{T}\int_0^T (z(t) - z_r(t))^2 dt \approx \frac{1}{T}\sum_{k=0}^{N}(z(t_k) - z_r(t_k))^2 \Delta t, \tag{5}$$

   where $z_r(t)$ is the desired ball position trajectory and Δt is the sampling time of the trajectory data.

2. **Minimizing the energy consumption:** The power consumption of the motor is approximately proportional to V^2. Therefore, we approximate the average energy consumption of the motor as

$$J_{energy} = \frac{1}{T}\int_0^T u(t)^2 dt \approx \frac{1}{T}\sum_{k=0}^{N} u(t_k)^2 \Delta t. \tag{6}$$

3. **Safety constraint:** The physical limit of the position of the ball on the beam is given as $-0.2\text{m} \leq z \leq 0.2\text{m}$. At each side of the beam, a stopper makes sure that the ball does not escape the beam. Ideally, we want the ball to avoid collision with the stoppers because the impact will make the system unstable and unpredictable (as your system equations do not model this effect). Also, the servo motor has a physical limit on its

angle range. Therefore, we consider the system to be unsafe when the state satisfies one of the following conditions:

$$z(t) \leq -0.19\text{cm}, \quad \text{collision imminent at the right edge} \tag{7}$$
$$z(t) \geq 0.19\text{cm}, \quad \text{collision imminent at the left edge} \tag{8}$$
$$\theta(t) \geq 60^\circ \text{ or } \theta(t) \leq -60^\circ \quad \text{servo angle out of operation range.} \tag{9}$$

The exception to this condition is allowed for the first one second of the trajectory, to allow the ball to start from the boundary at its initial state. We consider the following indicator

$$J_{safety} = \begin{cases} 1 & \exists t, z(t) \text{ unsafe}, \\ 0 & \text{else} \end{cases}, \tag{10}$$

which is 1 if the trajectory is unsafe and 0 if the trajectory is safe.

Score & Evaluation:

For each experiment run, given the recorded data of $\{z(t_k)\}_{k=0}^N$ and $\{u(t_k)\}_{k=0}^N$, $J_{tracking}$, J_{energy} and J_{safety} can be evaluated. Your goal is to **minimize** the **score** of your controller, computed as:

$$J_{score} = w_t J_{tracking} + w_e J_{energy} + w_s J_{safety}, \tag{11}$$

where w_t, w_e, w_s are the weight of each cost terms. When you report your data to the leaderboard, the leader board will save the best (minimum) score among your reported runs, and use it as your final score. (Instructions on how to report your data will be announced next week with the experiment guideline.) Students who achieve better (lower) scores will receive more credits for the course project. You will have to write a short report on what controllers you tried when the competition ends.

Code Instructions:

1. Install MATLAB and Simulink by using the Berkeley academic license.
2. Download the starter code from https://github.com/HybridRobotics/ball-and-beam-project.
3. Run **setup.m** or add the repository and the subfolders to the matlab path.
4. You only need to change one file for developing your controller: **studentControllerInterface.m**.

Development guideline:

1. Develop your controller in **studentControllerInterface.m**. The function **setupImpl** can be used to do any one-time setup at the initialization of the controller, and the

function **stepImpl** is used to compute the control at each instant in time. For each timestep, this function takes the ball position (p_ball) and the servo angle (theta) measurement, and decides the voltage applied to the motor (V_servo). Feel free to do whatever you want inside the class, but make sure you do not modify the signature of the two functions **setupImpl** and **stepImpl**. Call the function **get_ref_traj.m** to get the reference trajectory $z_r(t)$ at a specific time instance. We gave you an implementation of a very simple proportional feedback controller for your reference. For more details, please read the instructions in the code.

2. Test the performance of your controller by running **run_matlab_ball_and_beam.m**. Play around with different reference trajectory profile under various values of amplitude and period by modifying **get_ref_traj.m**, and debug and improve the controller.

3. Finally, before the experiment, make sure that your controller can run in the Simulink. This is very important to make sure that your controller can interface with the real hardware for experiments. Run **run_simulink_ball_and_beam.m** to test this. If there are issues in your implementation, building the code will fail and it will output errors in the Simulink Diagnostic Viewer.

4. Once your controller works well in the simulation, be ready to test it on the real hardware! **(Instructions for the experiment and the lab sessions will be announced next week.)**

5. Debug your controller on the real hardware. When the reference trajectory for the competition is released, you will be able to report the score of the controller to the leader board.

Note: The simulation result from **run_simulink_ball_and_beam.m** and from **run_matlab_ball_and_beam.m** may be different from each other, due to the difference in the used models. The dynamics used in latter is implemented in **ball_and_beam_dynamics.m**, which is the implementation of (4).

References

[1] Quanser. Ball and beam - quanser. `https://www.quanser.com/products/ball-and-beam/`. Accessed: 2022-04-12.