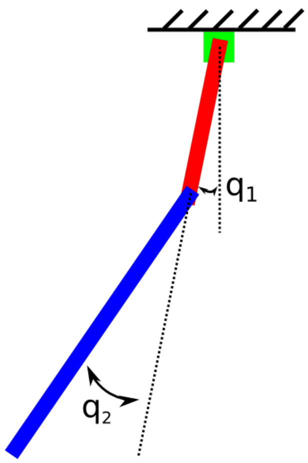
Course › Week 3 › Proble... › Acrobo...

Acrobot Balancing

Acrobot Linearization

0.0/10.0 points (graded)

In this problem, we will develop an end-to-end controller capable of swinging up the acrobot and balancing it at the upright configuration.



First, let's use LQR to design a stabilizing controller for the upright configuration $q_0 = \begin{bmatrix} \pi \\ 0 \end{bmatrix}$.

The first step is to linearize the system about the upright point. Find the matrices A and B such that the linearization is

$$\frac{d}{dt}\begin{bmatrix} q \\ \dot{q} \end{bmatrix} \approx A \begin{bmatrix} (q - q_0) \\ \dot{q} \end{bmatrix} + Bu$$

With the linearization, use MATLAB's **[K,S] = lqr(A,B,Q,R)** function for some symmetric, positive definite cost matrices Q and R of your choosing.

It is recommended that you find the answers using your local copy of MATLAB and paste in the results below.

Drake Acrobot

In the **examples/Acrobot** directory of Drake, you will also find the Acrobot model **Acrobot.urdf**. Note that **AcrobotPlant** class uses different mass and length parameters! Don't use it!

You will find the following lines of code useful:

```
p = PlanarRigidBodyManipulator('Acrobot.urdf');
[f,df] = p.dynamics(t,x,u);
```

Which calculates $\dot{x} = f(t, x, u)$ and its derivative df given by

$$df = \begin{bmatrix} \frac{\partial f}{\partial t} & \frac{\partial f}{\partial x} & \frac{\partial f}{\partial u} \end{bmatrix}$$

Since the acrobot is time-invariant, we expect $\frac{\partial f}{\partial t} = 0$.

Note that the gradients (as are all gradients of vector-valued functions in Drake) are given in the following form,

$$df(:, i) = \frac{\partial f}{\partial x_i}$$

Alternatively, you might inspect the function **linearize** within the **DrakeSystem** class for an even easier approach.

Submission

Enter your answer with A, B, Q, R, K and S below. Please **DO NOT** attempt to use Drake functions in your answer. Calculate A, B in your local copy of MATLAB and then paste the answer below. You may find the mat2str() function helpful for converting a matrix into something suitable to be copied and pasted into the answer box.**lqr()** is a base MATLAB function, and so you can use it in your solution.

```
A =
B =
Q =
R =
[K,S] = lqr(A,B,Q,R);

```

Unanswered

```
% One option to find A and B using Drake is:
% p = PlanarRigidBodyManipulator('Acrobot.urdf');
% [f,df] = p.dynamics(t,x,u);
% A = f(:,2:5);
% B = f(:,6);

A = [0 0 1 0;0 0 0 1;12.6292 -12.6926 -.1721 .3015; -14.75 29.61 .3015 -.6033];
B = [0;0;-3.01;6.033];
Q = eye(4);
R = 1;
[K,S] = lqr(A,B,Q,R);
```

Run Code

LQR is just that easy! Now you have a controller that will stabilize the equilibrium at q_0. In terms of x, x_0, A, B and K *only* write the expression for LQR controller u.

u=

Answer: -K*(x-x_0)

Explanation
Since we linearized about the nominal point x_0, the LQR control law is $-K\left(x - x_0\right)$.

Download the stub code for an acrobot controller here and save it into the **examples/Acrobot** folder within your installation of Drake. When saving the file, make sure to NOT save it as a complete web page, or you will end up with some unwanted HTML headers. Complete the **output** function with your LQR controller, and run the simulation from a few initial states near the upright (see **AcrobotController.run**). Try a few different values for Q and R and note how the performance changes.

Submit You have used 0 of 3 attempts

Answers are displayed within the problem