# Acrobot Energy Shaping

## Acrobot Energy Shaping

0.0/5.0 points (graded)

The last piece of the puzzle is to implement an energy shaping algorithm for swing-up.

Setting aside (for now), your partial feedback linearization, use a similar energy shaping controller to the one which worked for the pendulum. For some user-defined gain $k_1$ and the total potential energy at the upright position $E_d$, use the control input $u = k_1 \left(E_d - E\right) \dot{q}_2$ so that $\dot{E} = k_1 \left(E_d - E\right) \dot{q}_2^2$.

Implement this controller in the stub code **AcrobotController.m**, and try a few different values for $k_1$. Simulate the system from states near the downright (passively stable) equilibrium. What is the typical behavior? Answer all that apply.

- [ ] Energy approaches the desired energy
- [ ] Energy does not approach the desired energy
- [ ] The state approaches the upright equilibrium
- [ ] The state does not approach the upright equilibrium

Submit You have used 0 of 1 attempt

## Acrobot Swingup

0.0/30.0 points (graded)
At last, we're going to put it all together.

First, combine the energy shaping controller with your partial feedback linearization. Choose $y = -k_2 q_2 - k_3 \dot{q}_2$ to stabilize the $q_2$ coordinate. If $u_p$ is the resulting partial feedback linearization controller, and $u_e$ is the energy shaping controller above, implement the control law $u = u_p + u_e$.

Simulate this control law for a few trajectories, starting from near the downward position, trying a few values for $k_1, k_2$ and $k_3$. Unlike with the pendulum, we do not have a proof that we will reach an orbit that passes near to the upright position. However, if you've chosen reasonable gains, you should find that the acrobot does, in fact, swing up--but that it then quickly falls back down. We're still missing stabilization!

This is where the LQR controller from earlier comes into play. The tough question here is when to switch from the swingup controller to the stabilization controller? Ideally, we would like to switch whenever the current state is in the region of attraction created by the LQR solution. However, determining this region is quite hard in practice, and something we will talk about later on in the course.

For now, the LQR cost-to-go function, $(x - x_0)^T S (x - x_0)$ gives a good notion of how "far" away any state is from the equilibrium $x_0$. In your MATLAB controller, implement a case statement to switch from the swing-up controller to the LQR controller, whenever this cost-to-go is within some threshold.

Now, you are good to go! There are a few values to tune: $Q, R, k_1, k_2, k_3$ and the switching threshold. You may have to play around with these values to get a stabilizing controller, but once you have something reasonable, the solution should be quite robust.

Once you are satisfied with your controller, record the output from one run by executing

```
[t,x,x_grade]=AcrobotController.run;
mat2str(x_grade)
```

Copy and paste the state trajectory **x_grade** into the answer window below. Your solution should result in **x_grade** being a 100x4 matrix. The matlab function

```
mat2str
```

just converts the value of x_grade into a string which, when pasted into matlab, results in x_grade being correctly constructed. It is recommended that you put your answer into a local .m file on your computer and ensure that it will run locally.

```
x_grade = [
% paste here
];

```

Unanswered

**Run Code**

Submit You have used 0 of 3 attempts