Alex Beyer

ENME743

Project Deliverable 1

This deliverable is written for Standard Project 4 – Acrobot Control.

Data Collection and Processing

Before we can discuss data collection, it's important to understand what the problem is actually asking. To solve this we need to train a model to develop a bang bang optimal control law for the swingup of the Acrobot with no consideration given to the stability of the final state. This means that we have a set number of discrete input states (-1, 0, 1) for the torque with our goal being to get the robot into the configuration $[\theta_1,\theta_2] = [\pi/2,0]$ in the shortest time, with our simulation terminating the instant the robot reaches that state. Additionally, the model in OpenAI gym is built such that any state other than the desired one has a reward of -1 while the desired (and termination) state has a reward of 0. The model also only actuates the 2nd joint; we'll have to achieve the desired swingup behavior by using the inertia of the lower member to move the first.

In order to help with visualizing the system a full model of the dynamics, consistent with the OpenAI Gym implementation, was implemented in MATLAB r2022a. This model allows for not only a dynamics testbed but also gives full control over simulation parameters, making getting and processing data much faster. The collected data for two experiments will be shown in the visualization section, one case with no control input and one with a constant control input of $\tau = +1$ (the constant -1 input case was omitted for being a mirror of the +1 case). MATLABs built in plotting tools were used to display data. No direct processing of the data was done, however it is plotted in both state space ($\theta_1\theta_2$ axes) and cartesian space (xy axes) and is used to calculate quantities like system energy for visualization purposes. Because we are randomizing our initial state this problem is much easier to approach with an unsupervised model, otherwise we'd need to provide an optimal baseline solution across the entire range of initial conditions. In fact, the model lends itself well to reinforcement learning since it's a task with two discrete states (at the objective and not at the objective) with corresponding rewards. To this end we don't need to provide training data but we can get an idea of characteristics of good and bad solutions by looking at the dynamics. In principle all this model will need is a computer fast enough to simulate the coupled nonlinear dynamics of the Acrobot; there is very little data being stored, there are no distributed calculations happening and the coupling between links means system dynamics don't even lend themselves well to being calculated in parallel.

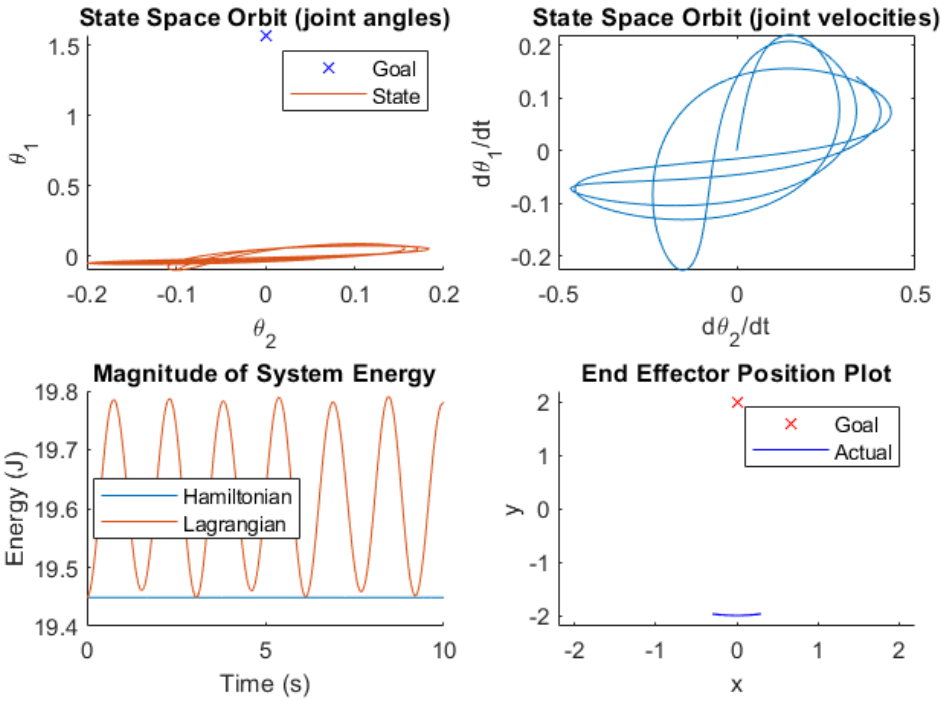
Data Visualization

As before, visualizations were created by simulating the dynamics and plotting model states in MATLAB r2022a. Three types of quantities are visualized here: state space plots, cartesian plots and system energy plots. The cartesian plots are useful to understand the how the system would behave in the real world, but these plots do little to shed light on what an optimal control law looks like for this system. This is down to two facts, first because having two coupled links means that our end effector motion in cartesian space is highly constrained (to within a 1 unit radius circle of the end of the first link) and secondly because our system is prone to chaotic dynamics which often makes these plots hard to

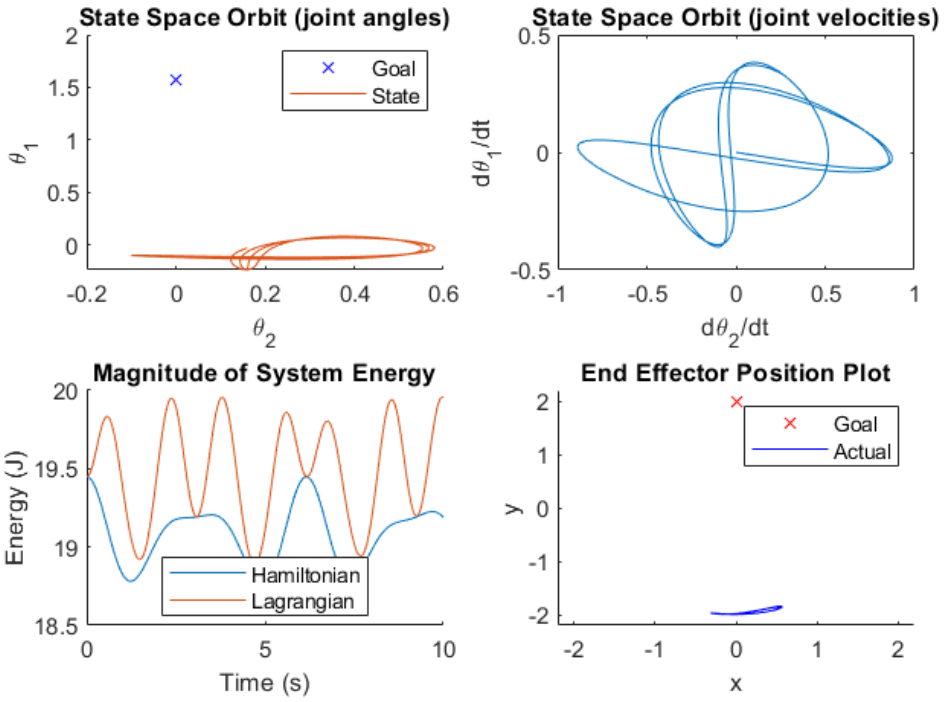
understand. State space plots, by contrast, are more abstract but allow us free motion which is convenient because many times optimal solutions show up as orbits that either grow or decay to the desired point, depending on the control law as opposed to the complicated irregular paths traced by the chaotic dynamics of the end effector in cartesian space. Finally energy plots are included as an aid to go with the state space plots. Our system is inertia driven, relying on the motion of the lower link to swing the upper link into position which will appear in the energy plots as the kinetic and potential energy experiencing alternating periods of growth and decay.

The below visualizations are for the two torque cases discussed above ($\tau=1$ and $\tau=0$) with a preset initial condition of [-.1;-.1]. The IC was chosen to give a consistent comparison between the two cases as well as to attempt to generate the most interesting behavior from the simulation while staying within the bounds of the randomly generated ICs; we'd see similar results from any ICs but this particular set should exaggerate them the most for discussions sake. Neither visualization will capture the full behavior of a state space orbit which approaches the desired point. Realizing that kind of result would require manually deriving the control law which largely defeats the purpose of training a model to do the same. Because our reward is a reduction in penalty that occurs at a single point the reward surface also becomes a single point, and as such is visualized by a black 'x' on the cartesian end effector position plot.

τ = 0 Plots



τ = 1 Plots



Data Interpretation

Due to the nonlinearity present in the system dynamics our data is also nonlinear, however those same dynamics also guarantee continuity of the data since we're physically unable to take large jumps between simulation steps. Our data in these two test cases is very smooth because both cases involve a constant control input, corresponding to no energy addition in the $\tau=0$ case and energy addition at a fixed rate in the $\tau=1$ case. This particular dataset doesn't show strong heteroscedasticity but the double inverted pendulum (the physical system we're controlling here) experiences chaos – and thus has the potential to undergo bifurcation of equilibria – for many control laws. In our visualizations the optimal control law would appear as growing or decaying orbits through state space, which we actually begin to see in the $\tau=1$ case and to a lesser extent in the $\tau=0$ case. We have a 2 degree of freedom system leading to a 4 dimensional state space (joint angles and corresponding angular velocities), although for a control law this simple we can focus primarily on controlling the joint angles (we don't need to stabilize the system upright so the joint velocities at simulation end do not matter). Outliers are little bit harder to quantify with this system. Physically we're constrained to staying within a 2 unit wide circle centered at the origin, any point outside of that is unreachable. However our joint angles are uncapped meaning they could conceivably grow to positive or negative infinity, but because we're looking at a rotational system there's no practical difference between the dynamics at one set f joint angles and any other provided they are separated by a multiple of 2π. However, similar to continuity it's hard to have an outlier in a classical sense, each state evolves into every following state so having a single state far from all the others shouldn't be possible.

A nonconstant control input would also allow us to see the inertial movement of the system in the energy plot as it builds up kinetic energy from the control input which it uses to swing itself around, converting some to potential energy and losing some to the control law trying to wind the robot up for another push. Here, the $\tau=0$ law has no input with which to build up the inertia needed for this and the $\tau=1$ case can build it up, but then fights the inertia throughout the swing as it keeps trying to spin the lower member. If we were developing this control law in a more classical way one method we could use is passivity, in which conservation of energy would be exploited to help achieve the swingup behavior.

Considering model tradeoffs the main one we'd have to make is on how much we want to inform the model of the system dynamics. For example, if we could somehow disincentivize the model from passing the same control input for sustained periods of time we'd likely converge on a solution much faster since this system behaves fairly poorly by our metrics under sustained, unchanging input (the system is unable to explot its inertia well under constant force). Our final control signal will likely be noisy, owing to there only being 3 discrete input states, causing the system to not be able to make fine adjustments. In an extreme case we could try to have the model be able to forward predict what will happen for a given input by giving it a full picture of the dynamics, but this is incredibly expensive and likely wouldn't help much unless we provided a very long forecasting horizon. The opposite extreme, providing no information on the dynamics, will likely preform much better at the cost of taking longer to find a solution.

Full simulator code can be accessed at ter.ps/743ACRO