COMP 765 – Winter 2019

Assignment 1: Dynamics and Control

Due Feb 11th, 6pm

Question 1: Create or Modify a Robot Simulation

In order to work with robots in software, it's essential that we can accurately predict their motions over time. In software, this requires specifying the robot's physical properties and executing ODE forward simulation, for which there are many libraries (some of them are good).

Find or create a robot simulation by using one of the available libraries, and use it to follow the following questions:

a) Briefly name and describe the robot you simulated. Include a descriptive figure to show its body geometry. State your process for creating this from "scratch" if you did so. If you have started from an existing robot, cite the source you started from and describe the modification you made (it can be quite minor, but must change the dynamics equations in some visible way).
b) Write the state and action vectors for your robot using meaningful symbols or English descriptions for each entry.
c) Write the robot's dynamics equations in vector form.
d) Provide a plot for the time evolution of your software simulation under passive dynamics or some simple feed-forward control. Comment on how you checked if this matches the dynamics equations you specified.

In order to get started, it may be helpful to browse this list of simulated robots that are very commonly used in robotics research today. Note that they use a wide range of ODE simulation libraries and methods to specify the robot's geometry and dynamics:

- PyBullet's User's Manual
- https://github.com/openai/gym/blob/master/docs/environments.md
- https://github.com/deepmind/dm_control
- https://github.com/DartEnv/dart-env
- https://github.com/stepjam/RLBench
- https://meta-world.github.io/

Honor for considering:

- 3D, especially if orientation is meaningful in the dynamics
- Anything underactuated
- Anything space-inspired (some links to help with this):
    - https://www.nasa.gov/feature/questions-nasas-new-spaceship
    - https://er.jsc.nasa.gov/seh/robots_in_space.htm
- High dimensional bodies
- Non-standard actuators (fans, hydraulics, flippers, sound, rocket engines, bungie cables)

Question 2: LQR on a Cart Pole System

We have provided code to simulate a cart and pole system in 2D using scipy's ODE solver:

- https://github.com/dmeger/COMP417_Fall2019/tree/master/lqr_question

Initially, this system has no control and falls from near the balance point. The file "cartpole_learn.py" holds starter code for implementing LQR that can make the cartpole balance. Note, of course this system is non-linear, so the idea is to find the linear approximation near the balance point, which we expect to only succeed for "close enough" starting conditions.

a) Show your math and explain the procedure you used to determine A, B, Q and R.
b) Keep an experimental notebook. What sequence of A, B, Q and R did you try, and what was the performance for each?
c) Write out the final A, B, Q and R you found and provide a figure that shows your balancing success.

Some math to get you started:

The state vector of this cart pole is: $x = [x \quad \dot{x} \quad \dot{\theta} \quad \theta]^T$.

Its dynamics equations are:

$$\ddot{x} = \frac{2ml\dot{\theta}^2 \sin\theta + 3mg\sin\theta\cos\theta + 4(u - b\dot{x})}{4(M+m) - 3m\cos^2\theta}$$

$$\ddot{\theta} = \frac{-3(ml\dot{\theta}^2 \sin\theta\cos\theta + 2((M+m)g\sin\theta + (u - b\dot{x})\cos\theta))}{l(4(M+m) - 3m\cos^2\theta)}$$

The relevant physical constants are:

- Pole length, l=0.5
- Pole mass, m=0.5
- Cart mass, M=0.5
- Friction constant, b=0.1
- Force due to gravity, g=9.82

Question 3: Non-linear control for swing-up of a Cart Pole

Try one of the non-linear control methods we've mentioned in class to swing-up and balance the cartpole from an initial downwards pole angle. Recommendations are:

- Value Iteration
- iLQR
- DDP
- A trajectory optimization package such as GPOPS

For this question, you must focus on model-based control techniques. Although methods like Deep RL can be used to find good controllers with trial-and-error, we'll look at those in a later assignment, so please don't use them here.