Apply KUO FeedForward Model to Simulated Robotic System.

Introduction

Although initially, we thought that applying Kuo's feedforward model to a trep system would be relatively simple, it turned out that Kuo's model was very idealized in a way that using trep's dynamic simulation doesn't allow.

Real-world Implementation

Trep simulation is much closer to a “real” robotics system that Kuo's model. After initially solving Kuo's feedforward model in mathematica and python, where we could use equations of motion directly out of the paper. We attempted to construct the Kuo model in trep. However, we did not consider the fact that Kuo's implementation and our mathematica and python simulation effectively halts the integration of the dynamics, to instantaneously apply an impulse control and then restart the integration with the new initial conditions. We quickly realized that this was not possible in trep and not realistic for a robotic system. Additionaly, Kuo's model was dimensionless, and we applied dimensions in trep, comparable to that of a human leg. \*\*\*\*additional info about dimensionless problems

Discrete Time issue.

In order to apply the impulse we had to sample to time and write a control law that applied a very high torque at the peaks of the swinging motion. This took much trial and error in describing the tolerance of how close to the specified time the simulation needed to be. We chose to reduce the window such that the control was applied during only one sampling period. This was our 'slop'-timal solution to this issue, but we recognize that other design choice might have also worked.

Impulse

We then turned to the impulse itself. We began with an approximation of what the impulse ought to be based on the velocity of the pendulum, the desired velocity, and the application time of the torque. This starting point was around 600N-m of torque, but was not sufficient to create the trajectory found in Kuo. The torque impulse value was increased to 3195 N-m to follow the trajectory from the Kuo paper. This value was very large, and suggested to us that this model of impulsive control signals was too simplified for an actual system.

SAC

-can follow continuous trajectory(i.e. sine wave)

-can't follow discontinuous/non-differentiable trajectory from Kuo

--ways we tried to implement Kuo trajectory

-in order to implement have to hard code sac/derivatives

Conclusions