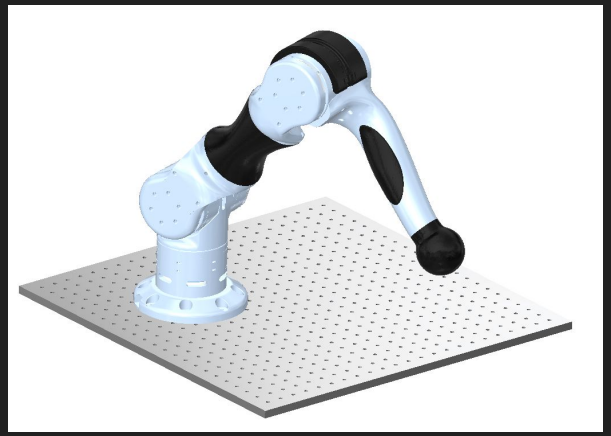
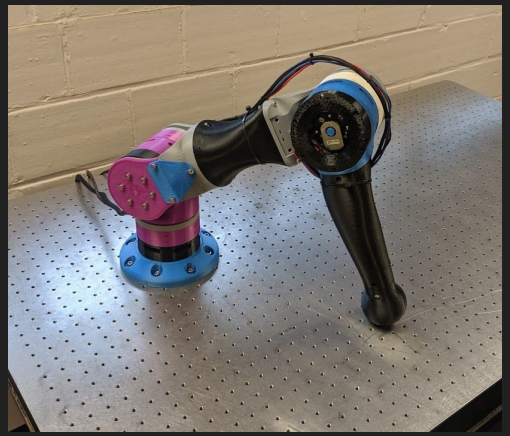
Reinforcement Learning Based Approach for Identification of Friction Parameters of Robotic Arm

Matt McDermott

Goal:

Develop a strategy to estimate the parameters governing the frictional losses for each joint in a 3DOF robotic arm so that they may be compensated via torque control





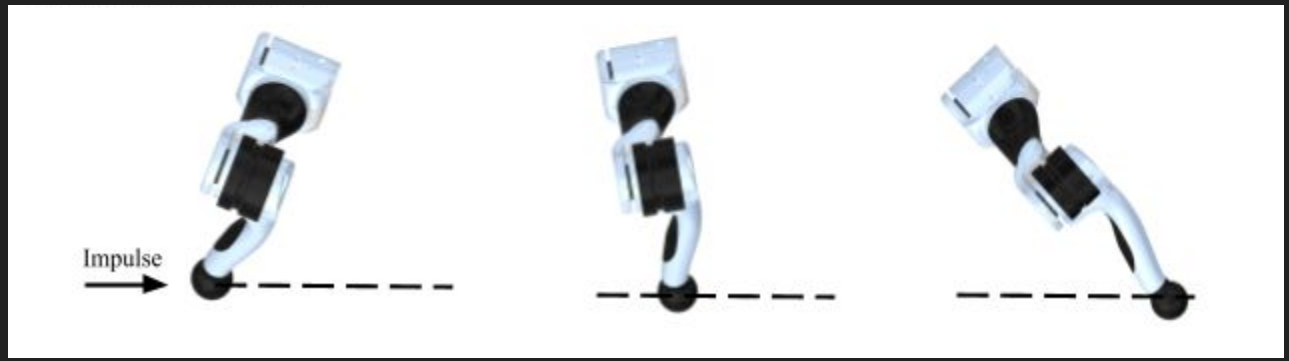
Motivations:

- Saves time in hardware development
  - Unlike mass or length, friction can not be measured directly
- Feedforward Linearization can be used to cancel internal dynamics of robot
  - Perfect system identification allows an actuator to be used as an arbitrary force exertion machine
- Much easier to verify a control policy derived analytically than one achieved solely through a neural network

Gravity, Inertia, and Friction Compensation

No Cancellation

Perfect Cancellation*



*Arm must stay within bounded subregion of workspace

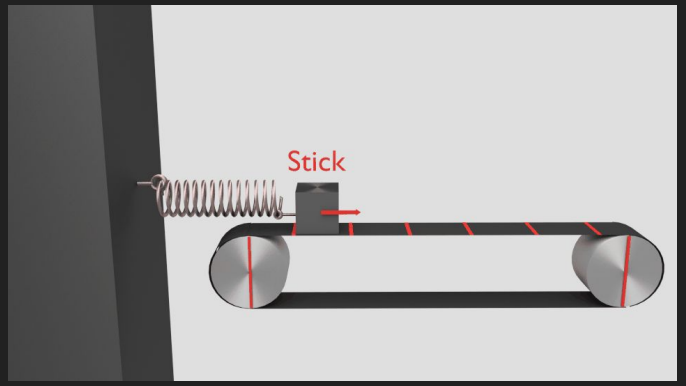
Real world application

Currently, high performance hardware must be manufactured extremely consistently for friction parameters to meet specifications. Implementing RL based system ID could make it easy to determine unique parameters for each joint after production which would allow manufacturers to relax specs and decrease cost while achieving the same performance by customizing friction compensation for each unit.

Modes of Friction:

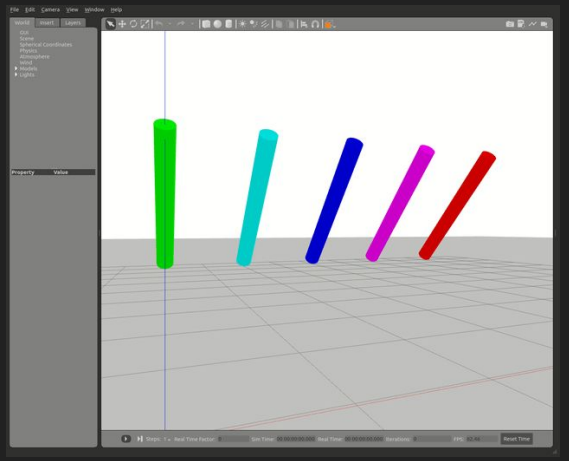
Dry Friction: (constant)

Static & Kinetic Friction Coefficients (F_s,F_K)



Viscous Damping: (velocity dependant)

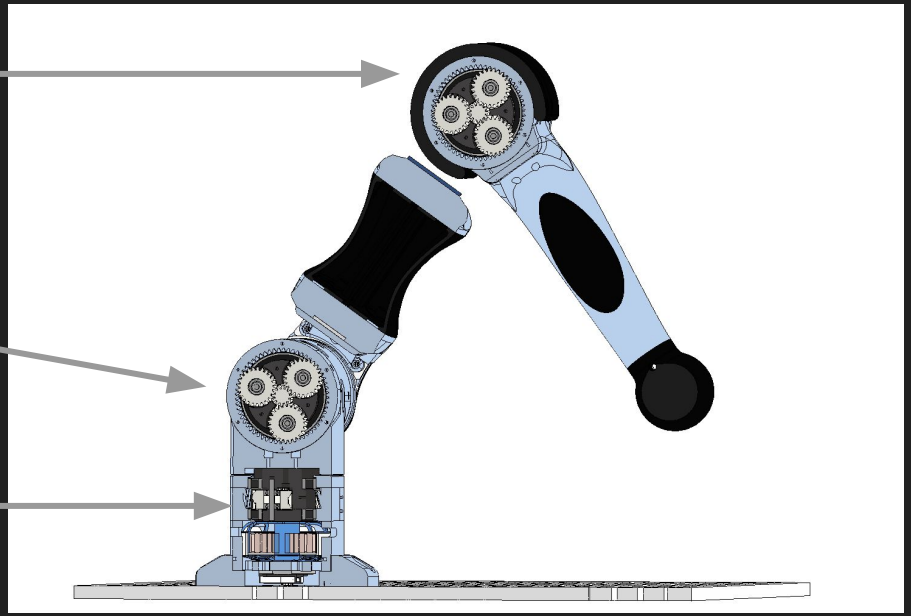
Viscous Damping Coefficient (β)



(Air resistance and other higher order terms ignored)

Values for F_s F_K and β are STATE DEPENDANT

- Flexing of plastic links may increase contact pressure in some positions
- Irregularities in gearbox will cause periodic changes in resistance
- Cantilever effect by reaching to points far from base will increase radial loading and cause changes in performance of bearings

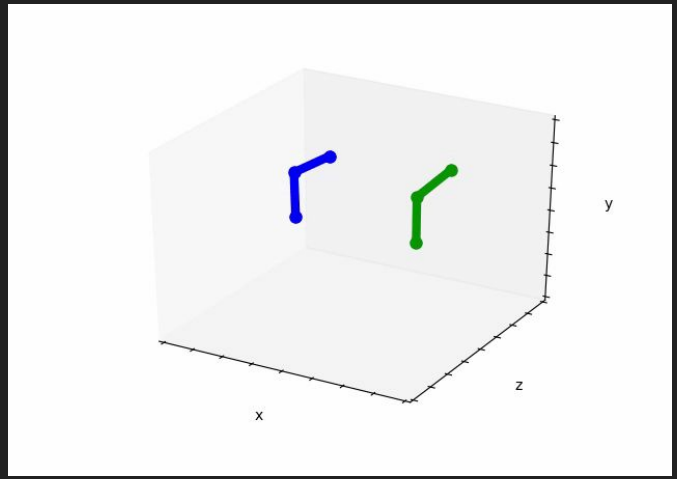


Environment:

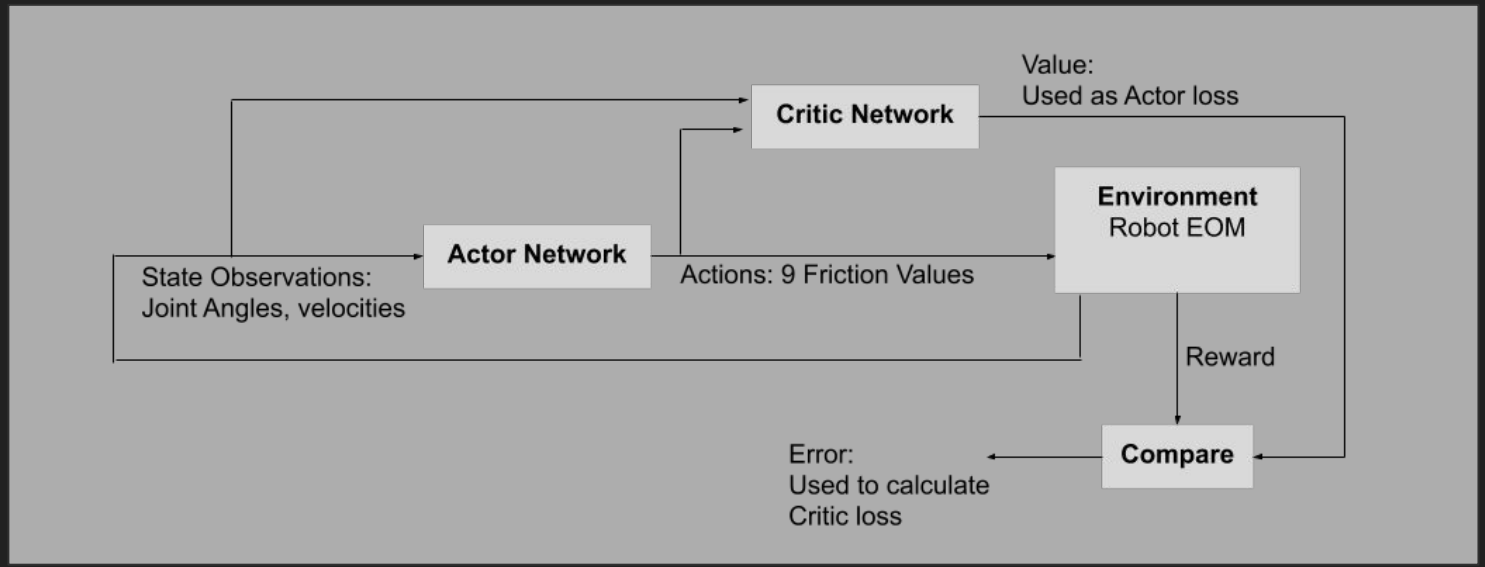
Modified EOM of robot to include the three friction parameters in forcing function

$$\mathbf{M}(q,t)\dot{u} = \mathbf{f}(q,\dot{q},u,t)$$

A 3D double pendulum system is HIGHLY chaotic. Shown here the blue and green arms have identical parameters for all but one friction variable (joint 2 kinetic).

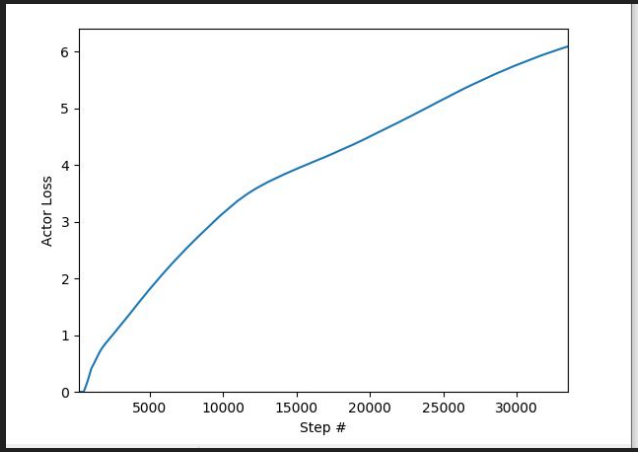


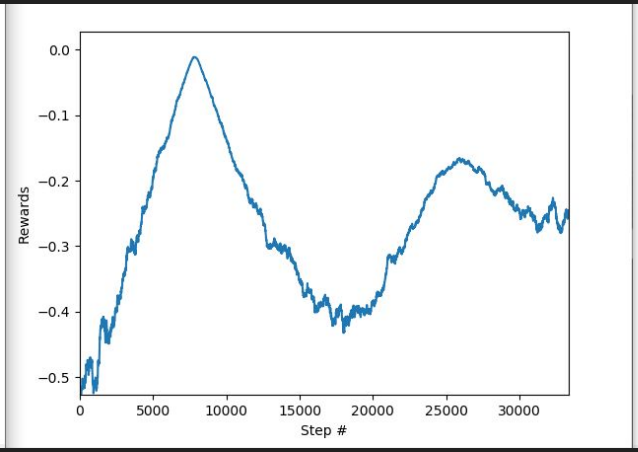
Actor-Critic Network

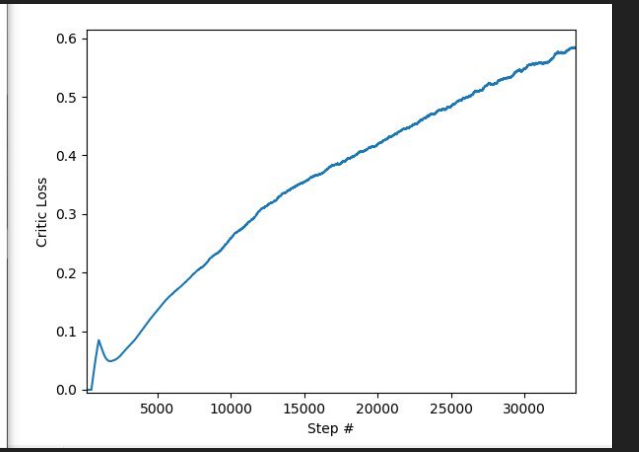


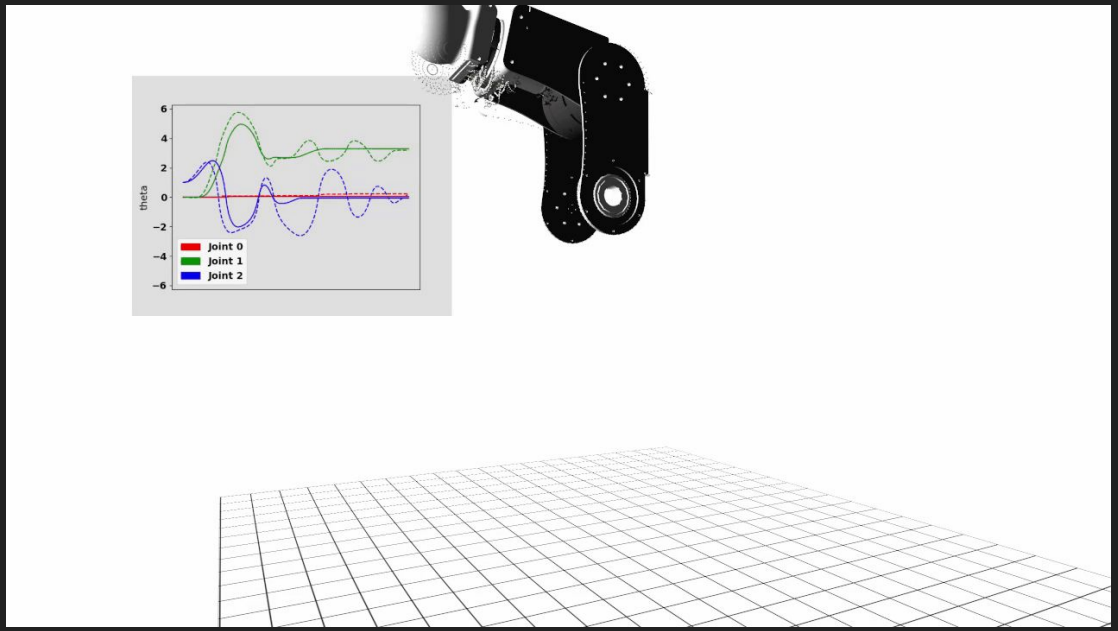
Algorithm

```
gt = ground truth friction enviornment
ef = current estimated friction enviornment

for each trial:
    goal = rand(3)       %theta0,theta1,theta2
    states = rand(3)
    for each step:
        action = actor(states)
        ef.friction_values = action
        next_states_ef = ef.predict(states)
        next_states_gt = gt.predict(states)
        reward = -abs(next_states_gt - next_states_ef)**2
        if last_step:
            done = 1
        memory_buffer.add(state, action, reward, next_state, done)

        if total steps taken > memory_buffer.size
            randomly select batch of n experiences from memory buffer
            for each experience:
                Qvals = critic(states,actions)
                next_actions = actor_target(next_states)
                next_Q = critic_target(next_states, next_actions)
                Qprime = rewards + discount_factor*next_Q*(1-dones)
                critic_loss = closs(Qvals,Qprime)
                critic_loss.backward()

                actions_pred = actor(states)
                actor_loss = -critic(states, actions_pred)
                actor_loss.backward()

                θ_target = τ*θ_local + (1 - τ)*θ_target  #soft update policy
```


Actor Loss
Step #
0
1
2
3
4
5
6
5000
10000
15000
20000
25000
30000


Rewards
Step #
0.0
−0.1
−0.2
−0.3
−0.4
−0.5
0
5000
10000
15000
20000
25000
30000


Critic Loss
Step #
0.0
0.1
0.2
0.3
0.4
0.5
0.6
5000
10000
15000
20000
25000
30000

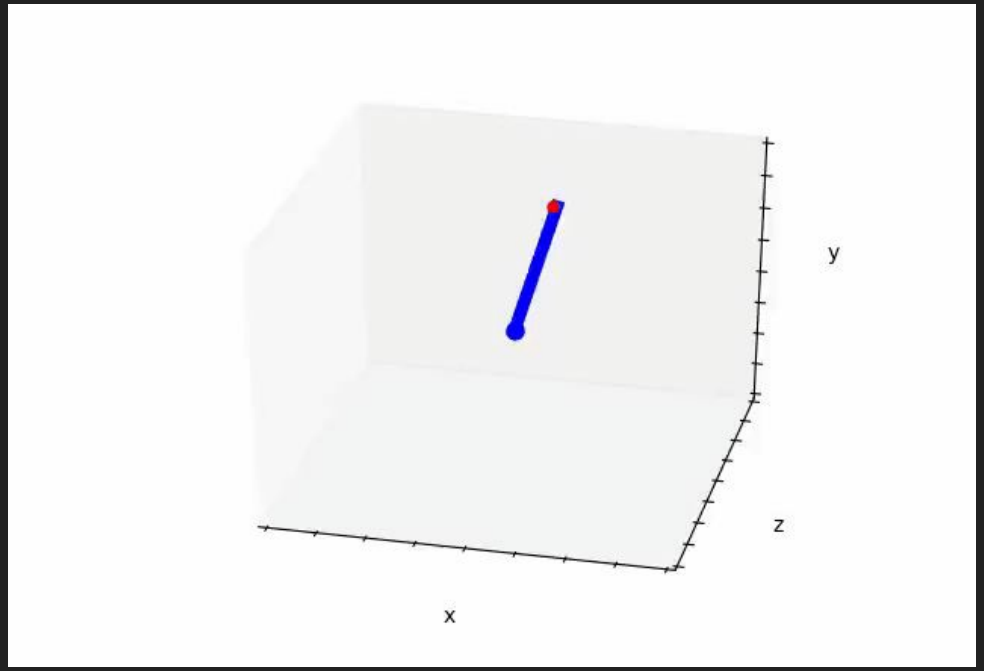

6
4
2
0
−2
−4
−6
theta
Joint 0
Joint 1
Joint 2

Back to the drawing board...

Alternate Approach - DDPG Torque Control

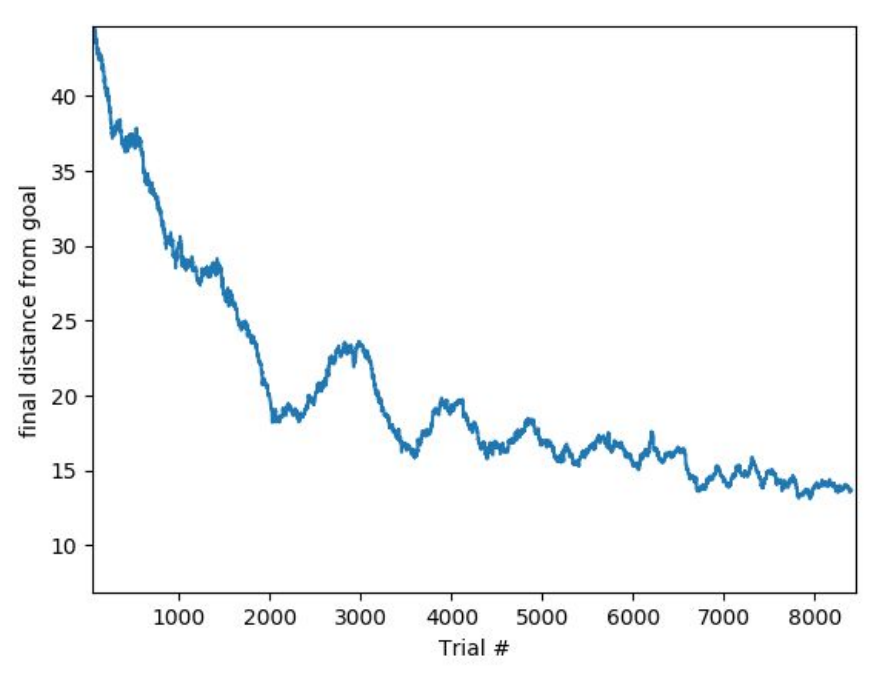
- Kept existing Actor and Critic Layers
- Changed Actor inputs to position and velocity of each joint as well as goal positions for each joint
- Actions taken by agent are torque commands for each joint
- Lose benefits of having EOM since actions change every timestep
  - Must reevaluate every timestep
  - Essentially doing inefficient Numerical Integration

1DOF Test

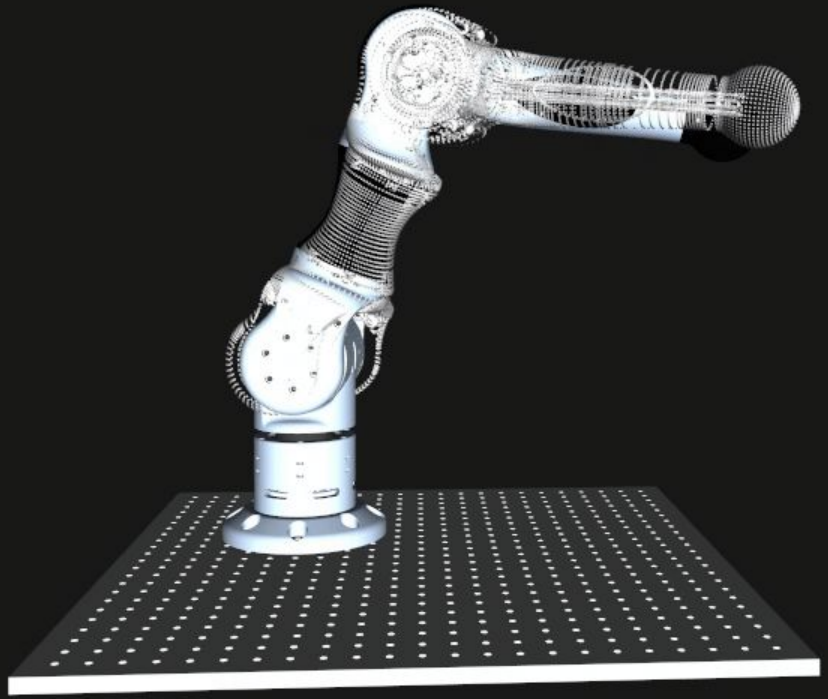


Moving on to 3DOF system

- Removed gravity from EOM (can be compensated separately in a stacked controller)
  - This helps the system converge much more quickly


final distance from goal
40
35
30
25
20
15
10
1000
2000
3000
4000
5000
6000
7000
8000
Trial #

Partially Trained Model- (6 hours on Nvidia Gtx 1060)



Future Work

- Retry original task via continuous contextual bandit approach
- Improving current network to allow for Friction and Inertia cancellation
  - Make arm approach nonzero target velocities
  - Actor input layer: 9 -> 12 nodes
  - Use manipulator jacobian to determine joint velocities at goal positions required to continue moving end effector in a straight trajectory

References

1. https://learnchannel-tv.com/wp-content/uploads/2018/12/Anima%C3%A7%C3%A3o-efeito-Stick-Slip.gif
2. https://www.allisonthackston.com/assets/img/jointdamping_gazebo.gif