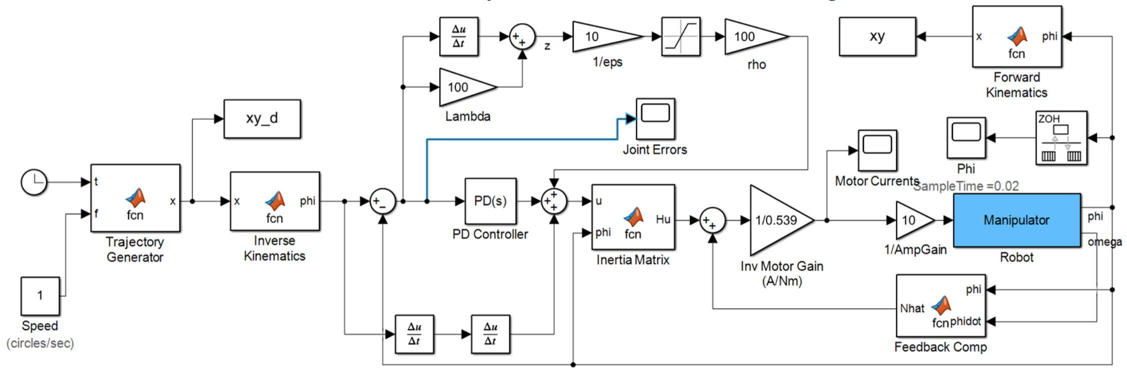
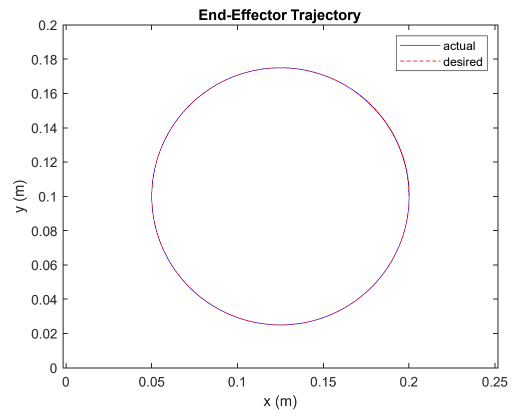
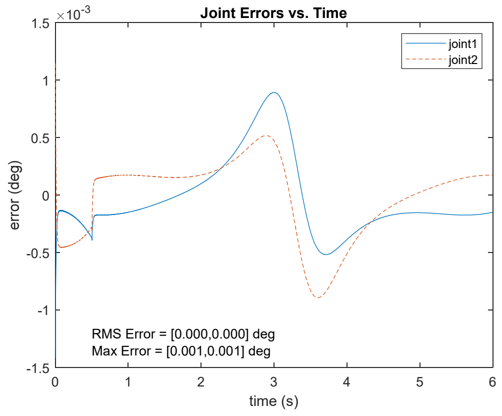
ME 5230/6230, CS 6330, ECE 6651 Intro to Robot Control
Problem Set #6 Solutions
Spring 2023

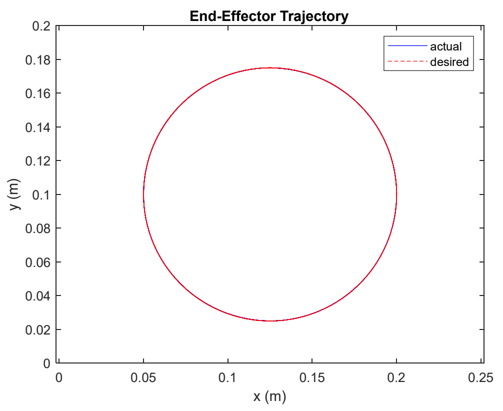
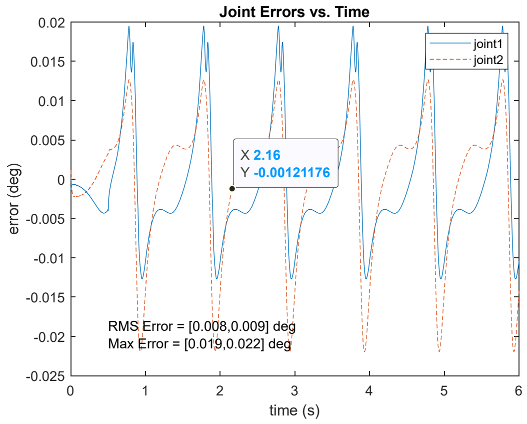
Problem 1: PD Control with inverse dynamics control and sliding mode control



Low speed:

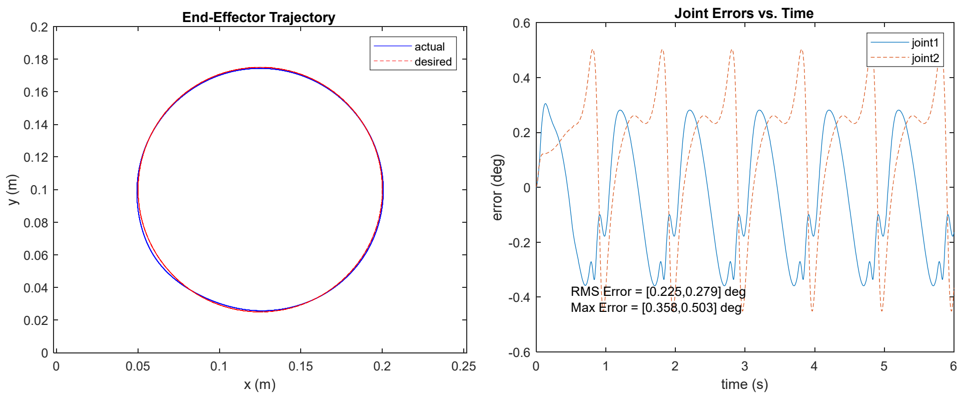
High speed:

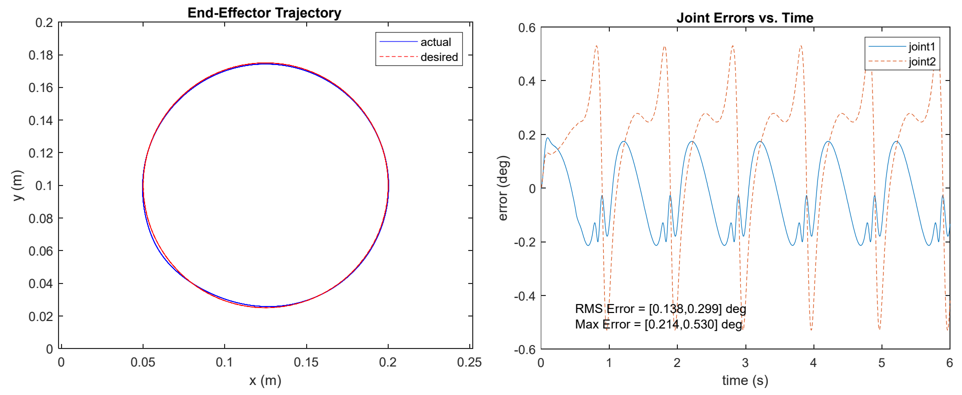
I used the same PD gains from IDC in PS#5: Kp=1200 and Kd = 40. I picked λ=100 to make the time constant of the sliding surface 0.01 (≈10x faster than the time constant of the PD control). 1/eps=10 gives a boundary layer thickness of ε = 0.1, which means the tracking error should be limited to approximately ε/ λ = 0.001 rad ≈ 0.06 deg. The rho = 100 is as large as I could make the sliding mode penalty without saturating the motor current. The sliding mode control works very well at both low and high speeds. The tracking error is significantly less than we achieved on PS#5. These results were achieved using Simulink ODE3 solver and a sample time of 0.002.

Problem 2. Compare the robustness of controllers to disturbances:
(A) Increase the mass of link two by 50% (this implies the inertia is also increased)

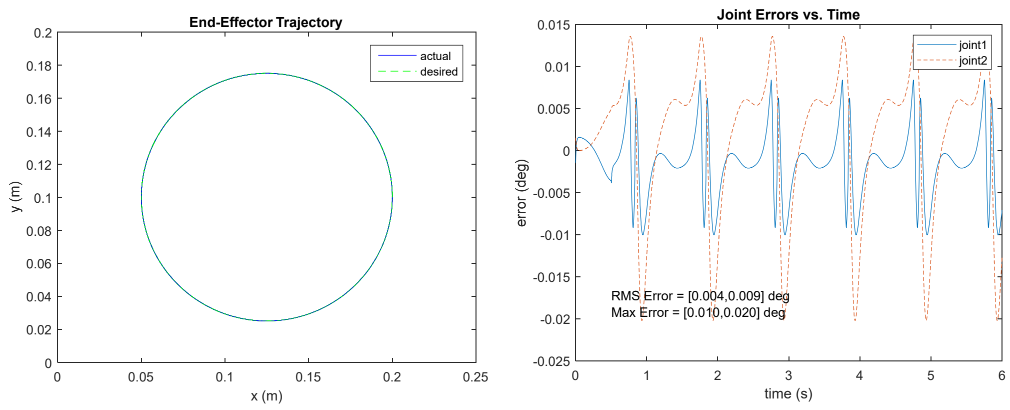
Feedforward Control at high speed with m_2 and I_2 increased by 50%:



Inverse Dynamics Control at high speed with m_2 and I_2 increased by 50%:



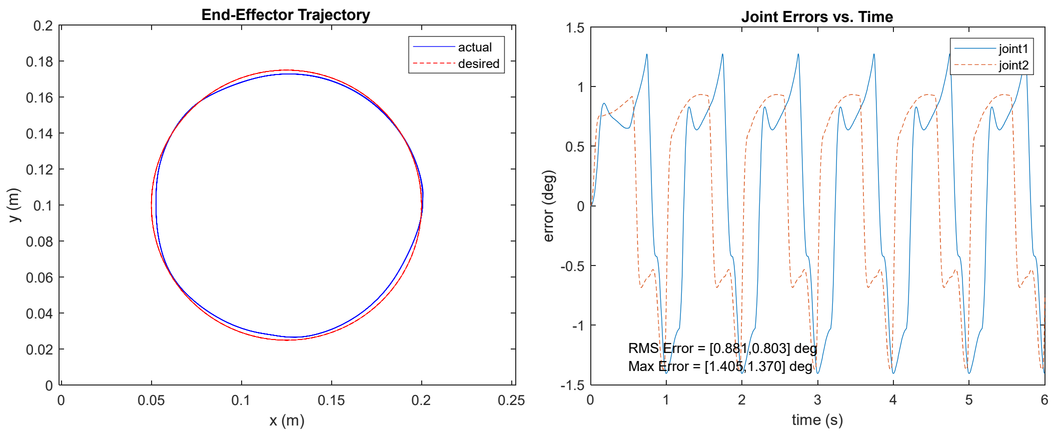
Sliding Mode Control at high speed with m_2 and I_2 increased by 50%:



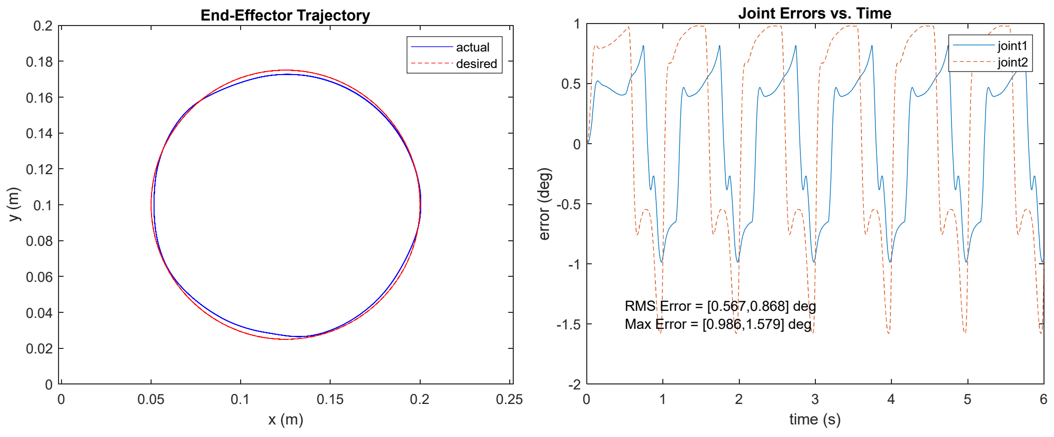
The sliding mode controller still outperforms the other controllers by about the same amount. Interestingly, it appears that the errors actually improved on all three controllers when introducing this particular model error. My guess is that there is something about this particular circle trajectory or the discrete-time implementation that makes it track better when certain mass-parameters are slightly underestimated. We'll see this same phenomenon occur with adaptive control.

(B) Multiply the coulomb friction in both joints by a factor of 10

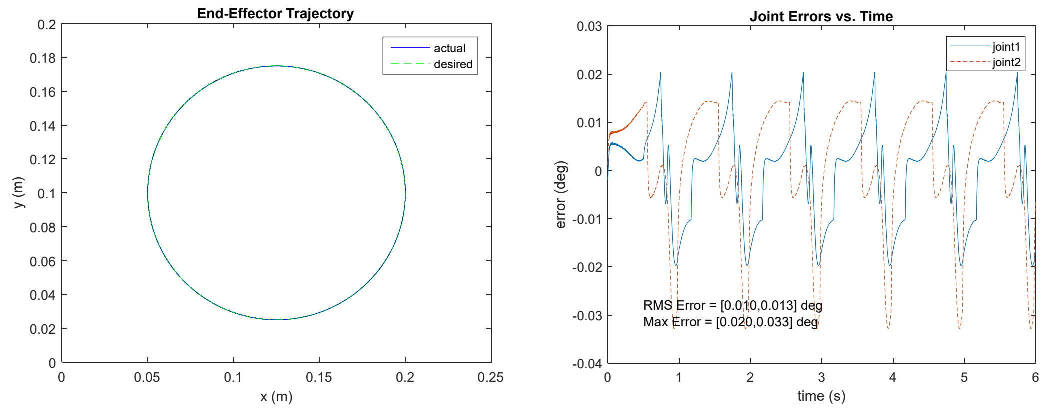
Feedforward Control at high speed with x10 friction:



Inverse Dynamics Control at high speed with x10 friction:

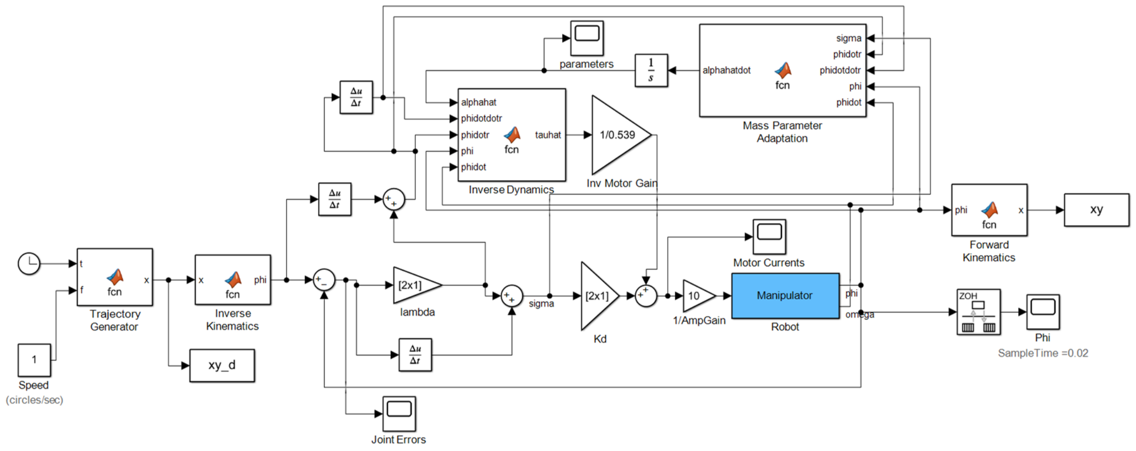


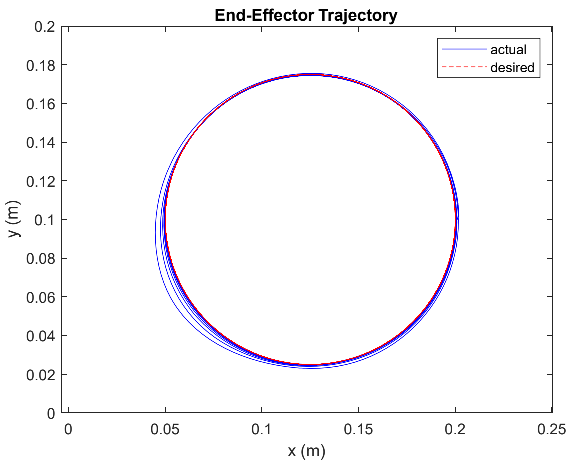
Sliding Mode Control at high speed with x10 friction:

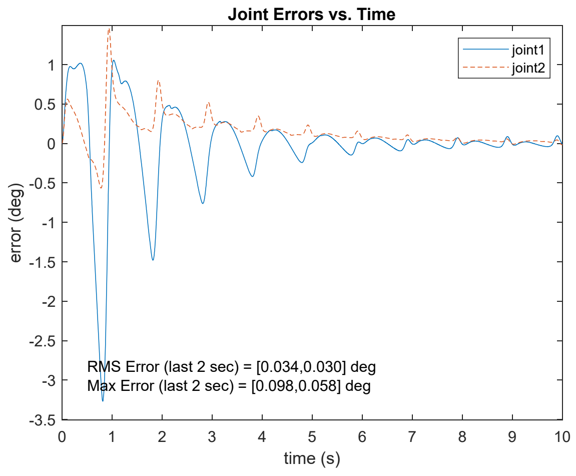


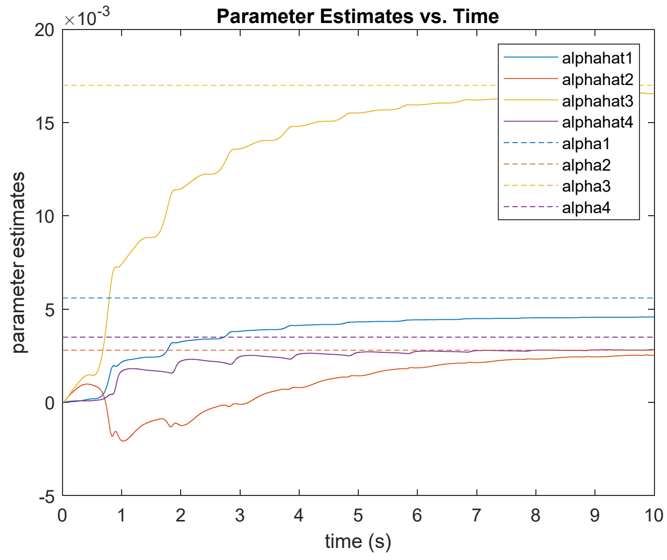
Disturbing the friction of the joints causes the error to get significantly worse for the feedforward controller and the inverse dynamics controller, but not the sliding mode controller. Sliding mode control does a great job of tracking the circle in the presence of all kinds of disturbances.

Problem 3. Adaptive Control









Since Adaptive Control is a form of Feedforward Computed Torque Control, I used the same PD gains as I did for Feedforward Computed Torque control in PS#5: Kd = 0.27, Lambda = Kp/Kd = 8.2/0.27 = 30. I had to tune the adaptation gains by trial and error. The speed of adaptation is inversely proportional to the gamma values. Start with large gamma values, which results in slow but stable adaptation. Then you can selectively decrease the gamma values until the adaptation of each parameter converges with a reasonable time constant. With a gamma matrix: `gamma = diag([10000,800,400,5000])`, the adaptive control converges quite rapidly (within three circles) to tracking errors of less than 0.1 degrees at a speed of 1 circles/s.

```
% parameters
% alphahat1: blue --> 0.0046 vs. alpha1 = N1^2*Jm1 + I1 + m2*a1^2
=0.0056
% alphahat2: green --> 0.0027 vs. alpha2 = r12*m2 = 0.0028
% alphahat3: red  --> 0.017 vs. alpha3 = r01*m1+a1*m2 = 0.017
% alphahat4: cyan --> 0.0028 vs. alpha4 = N2^2*Jm2+I2 = 0.0035
```

Notice that only two of the mass parameters (alphahat2 and alphahat3) converged to their true values, while the other two converged to something smaller than their true values. Unless the trajectory sufficiently excites all the dynamics of the robot, there is no guarantee that the parameters will converge to their true values, only that they will converge to something that minimizes the tracking error.
Once the adaptation converges, this controller is essentially the same as feedforward computed torque control, however notice that the tracking error is much better than when we implemented feedforward computed torque control with the real parameter values. Somehow the adaptive control has found a set of parameter values that works even better than the true parameter values! That might explain why underestimating the mass by 50% in the previous problem actually resulted in less tracking error. Perhaps this is something particular to our circling trajectory and/or the discrete-time implementation of adaptive control. It would be interesting to experiment with other trajectories and see what happens.

```
For reference, here is the Y matrix for the adaptive controller:

Y = zeros(2,4);
Y(1,1) = phidotdotr(1);
Y(1,2) = a1*cos(phi(2)-phi(1))*phidotdotr(2) - a1*sin(phi(2)-
phi(1))*phidot(2)*phidotr(2);
Y(1,3) = g*cos(phi(1));
Y(1,4) = 0;
Y(2,1) = 0;
Y(2,2) = a1*cos(phi(2)-phi(1))*phidotdotr(1) + a1*sin(phi(2)-
phi(1))*phidot(1)*phidotr(1)+g*cos(phi(2));
Y(2,3) = 0;
Y(2,4) = phidotdotr(2);
```