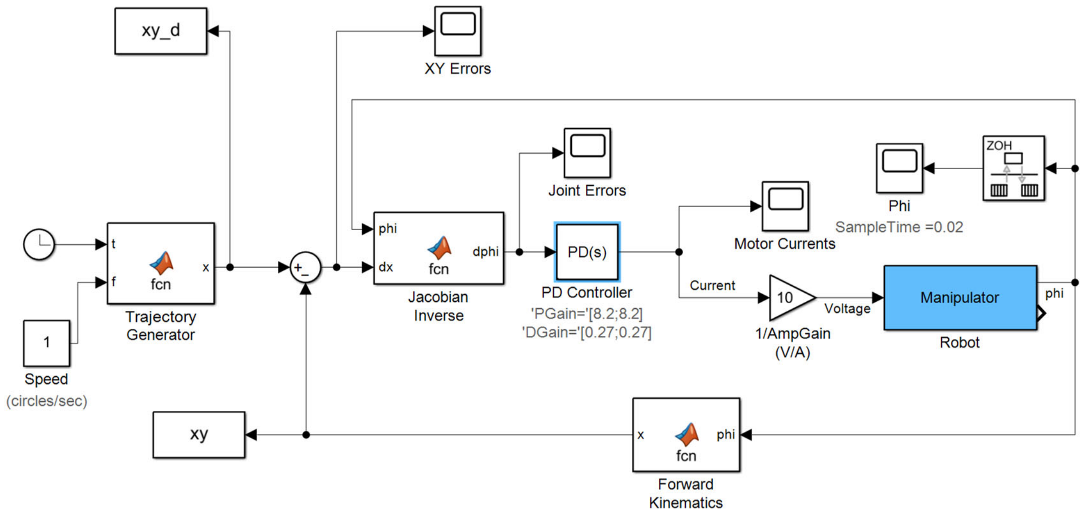
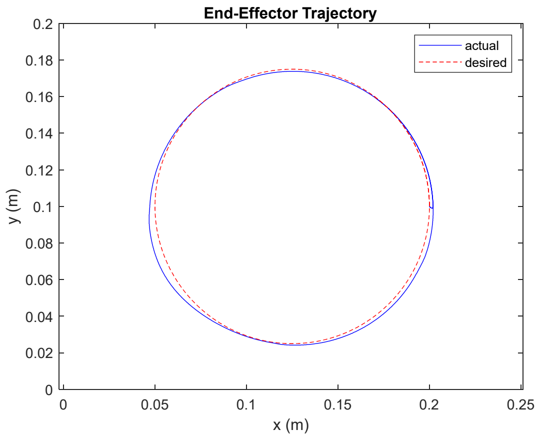
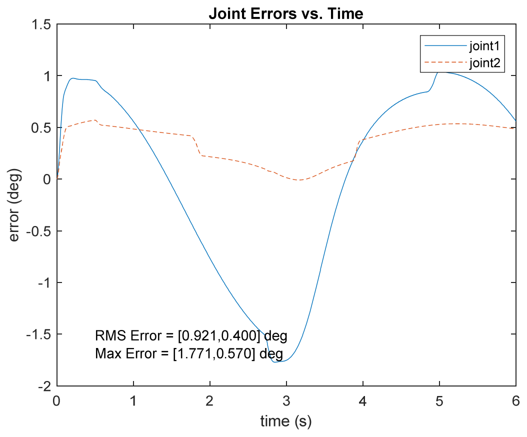
ME 5230/6230, CS 6330, ECE 6651 Intro to Robot Control
Problem Set #7 Solutions
Spring 2023

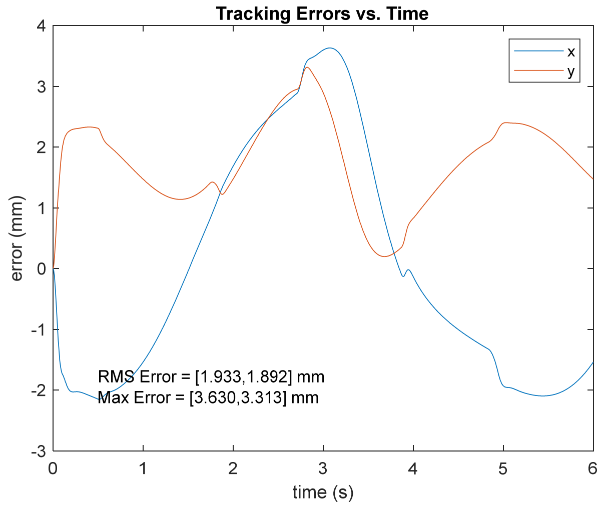
Problem 1: Jacobian Inverse Control



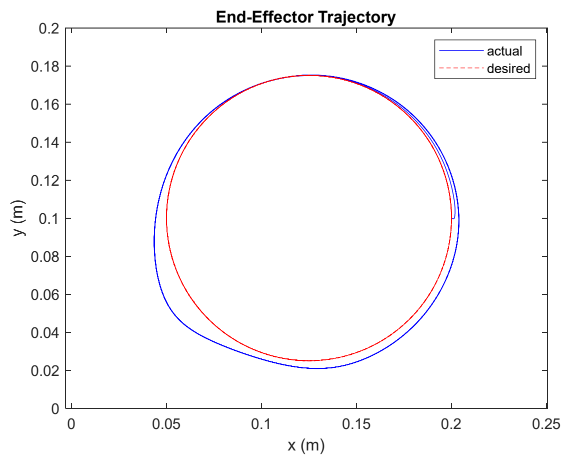
Low speed:

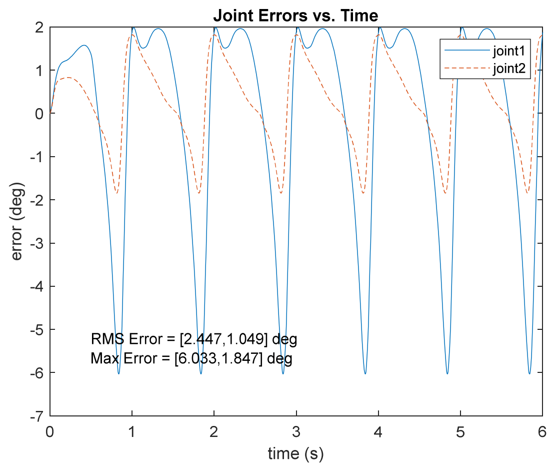


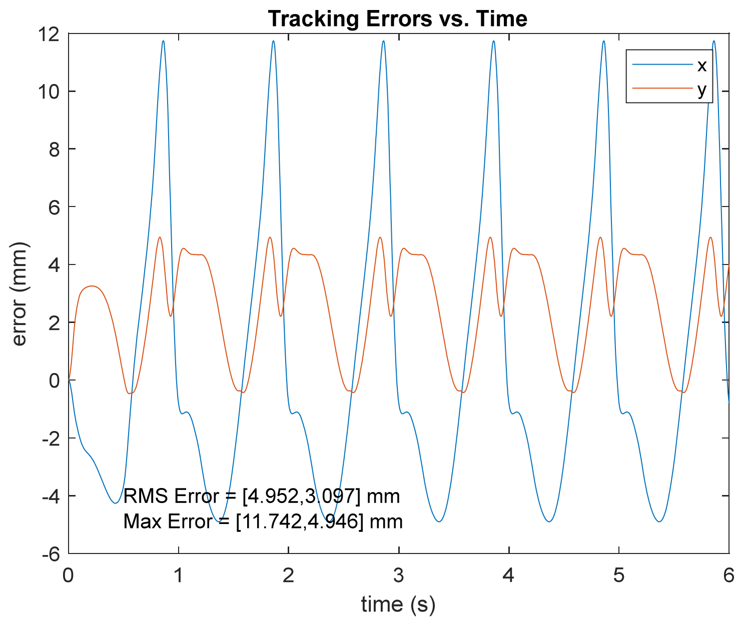




High speed:







The Jacobian Inverse control in operation space produces the exact same joint errors as PD control in joint space (PS#5 problem 1.1) when using the same PD gain values.

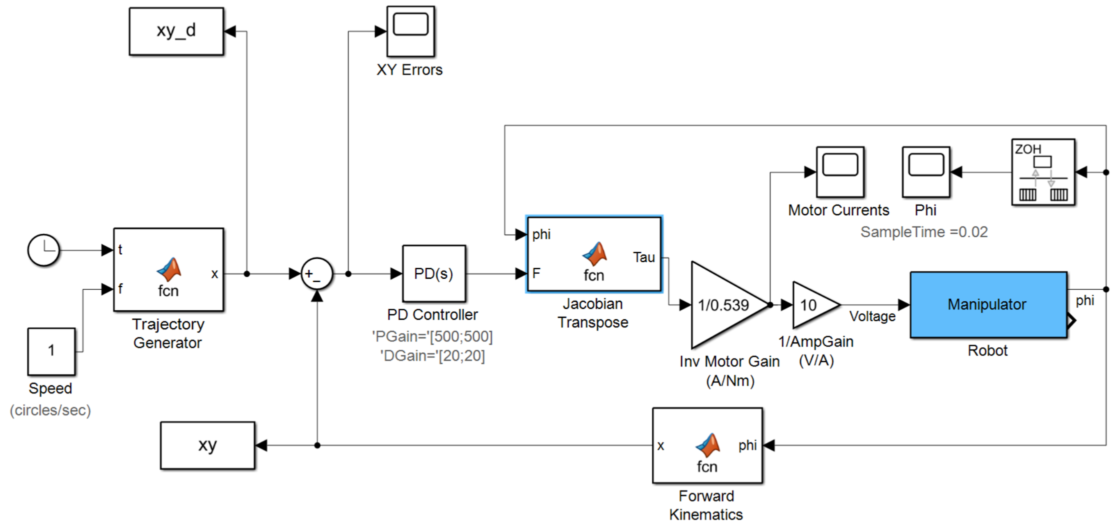
Note the Jacobian used in this controller is the one that maps between encoder space and op-space:

$$J = \begin{bmatrix} -a_1\sin(\emptyset_1) & -a_2\sin(\emptyset_2) \\ a_1\cos(\emptyset_1) & a_2\cos(\emptyset_2) \end{bmatrix}$$

We can get this by compounding the manipulator Jacobian with the pulley Jacobian, or by differentiating the forward kinematics as a function of encoder angles.

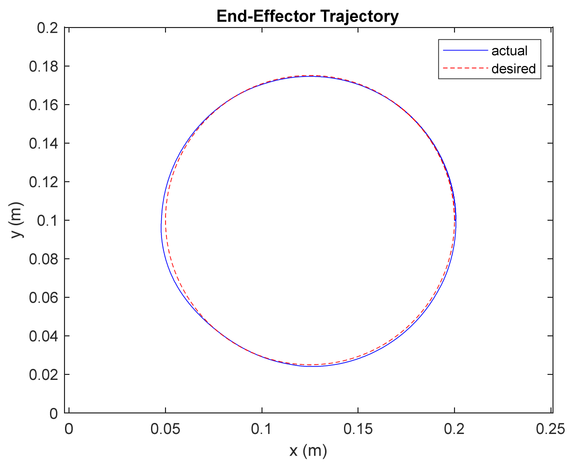
Problem 2

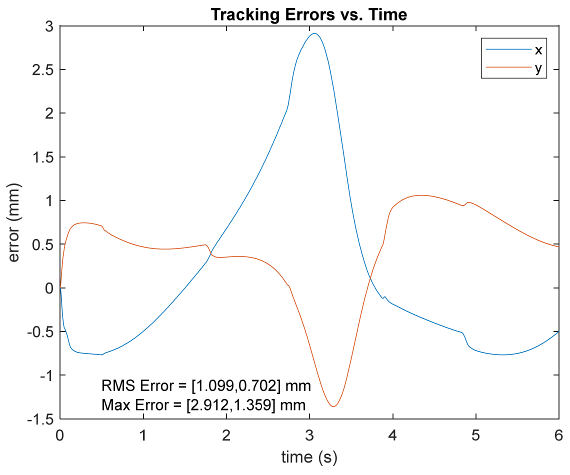
2.1 Jacobian Transpose Control



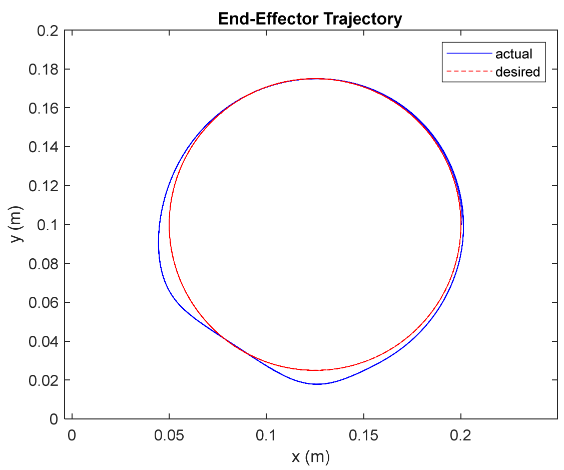
We use the same Jacobian here that we used in the previous problem (in this case the Jacobian transpose maps the penalty force in op-space to a penalty torque in encoder space. The inverse motor gain then converts this to a penalty current to the motors.

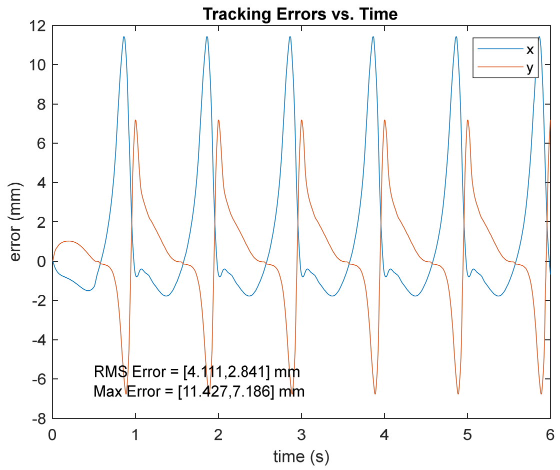
Low speed:





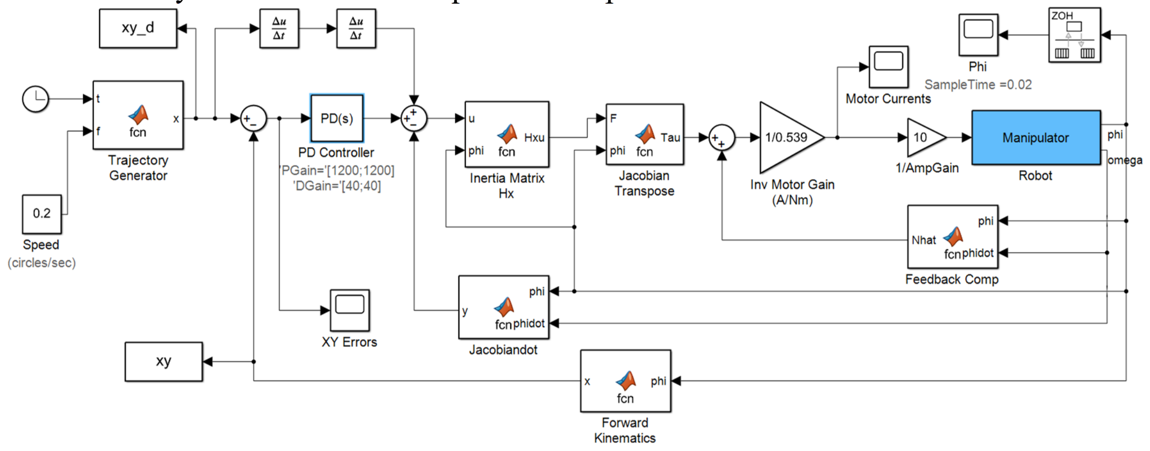
High speed:



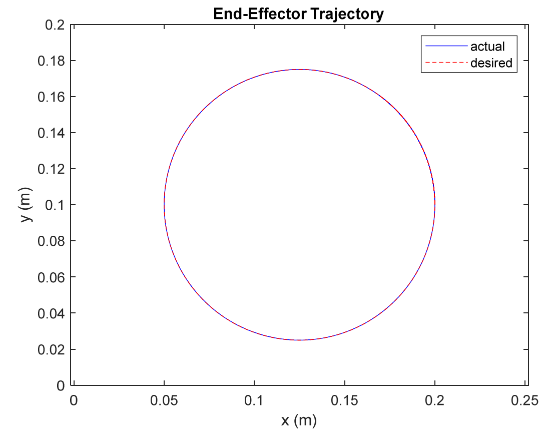


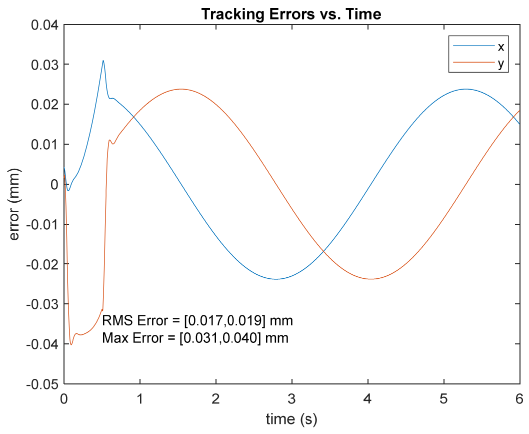
Jacobian Transpose Control is similar to Jacobian Inverse Control, but the PD gains act on the op-space error rather than the joint-space error. As long as we are using constant PD gains, we can't make the tracking performance exactly match that of Jacobian Inverse control. However these set of op-space gains result in comparable levels of RMS error in op-space. It should be no surprise that the gains are much larger, since they are now getting multiplied by the Jacobian transpose instead of the Jacobian inverse, and the values in the Jacobian are less than 0.15.

2.2 Inverse Dynamics Control in Operational Space

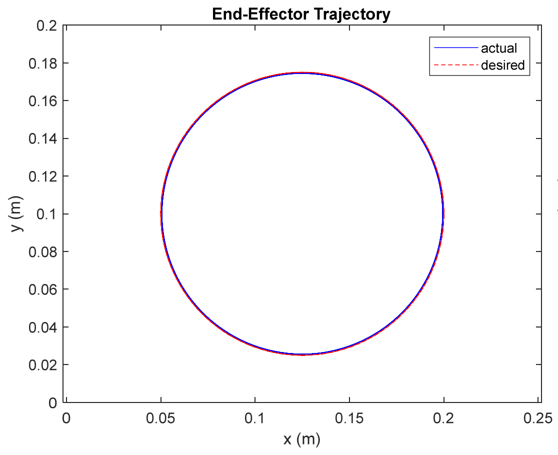


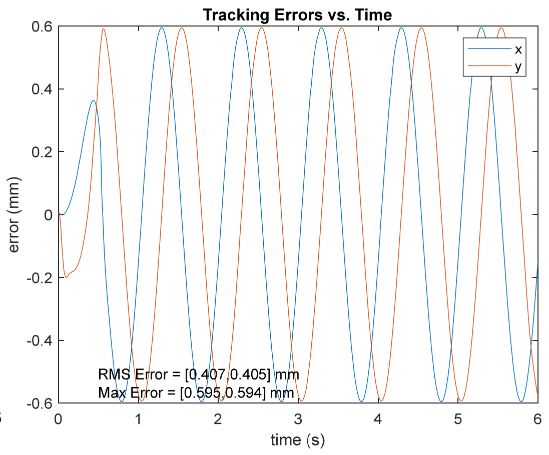
Low speed:





High speed:





The Inverse Dynamics Control does significantly better than the J^T controller. Note that since IDC reduces the plant dynamics to $\frac{1}{s^2}$ in op-space, we can use the same PD gains the we designed for IDC in joint space in PS#5, problem 2.1.

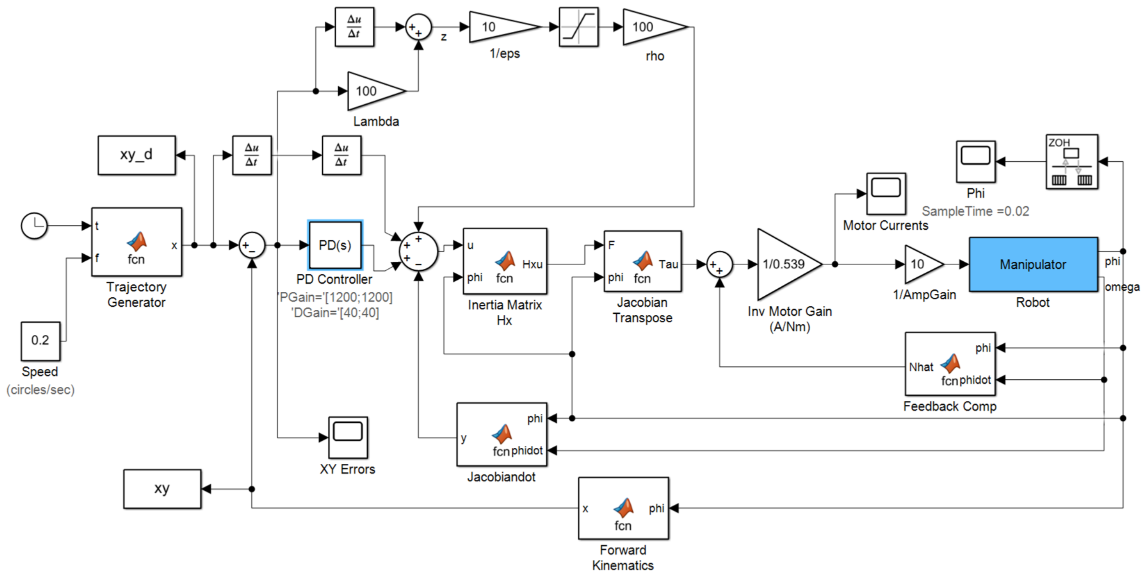
Note that the inertia matrix in this controller is the op-space inertia matrix, which is given by: ${}^{x}H=J^{-T}\left({}^{\Theta}H\right)J^{-1}$, where J is the same Jacobian as we used for J^{-1} and J^{T}.

Also, the Jacobiandot block in this controller requires one to take the derivative of the Jacobian:

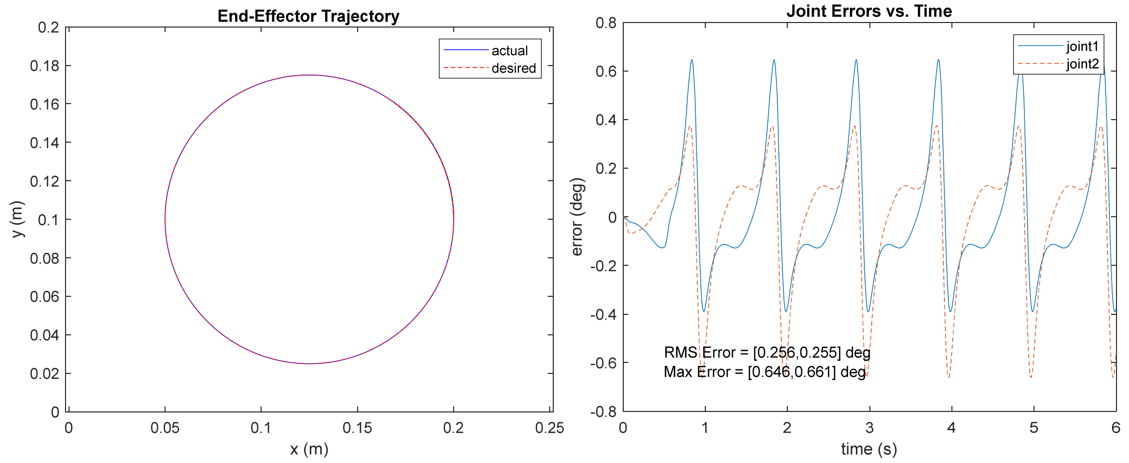
$$\dot{J} = \begin{bmatrix} -a_1\cos(\emptyset_1)\dot{\emptyset}_1 & -a_2\cos(\emptyset_2)\dot{\emptyset}_2 \\ -a_1\sin(\emptyset_1)\dot{\emptyset}_1 & -a_2\sin(\emptyset_2)\dot{\emptyset}_2 \end{bmatrix}$$

Note that $\dot{J}$ is itself a function of the encoder velocities and is also multiplied by the encoder velocities such that the output of the block is $y = \dot{J}\dot{\Phi}$.

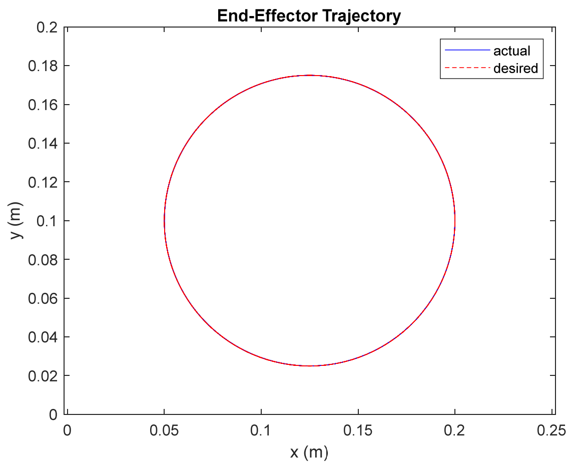
2.3 Robust Control in Operational Space

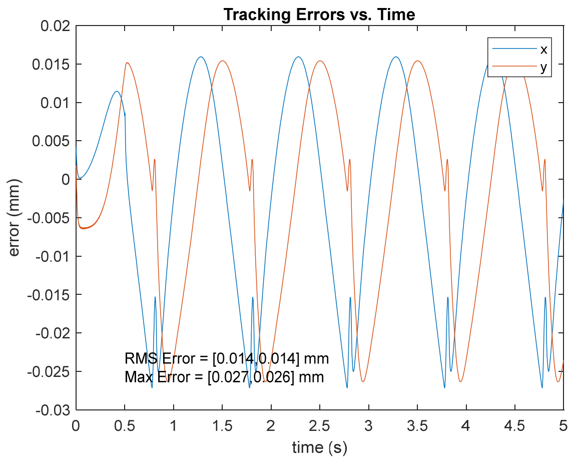


Low speed:



High speed:





The sliding mode control works the best of all at both low and high speeds. Again, since the IDC reduces the plant dynamics to $\frac{1}{s^2}$ in both joint space and op-space, and we're using the same PD gains as we did with IDC in joint space, we should be able to use the same sliding mode parameters as we did in joint space.