MCEN90028 Robotics Systems

Assignment 4 - Robot Dynamics & Velocity Control

Thomas Miles, 626263

Semester 1, 2020

Introduction

1 Robot Dynamics Simulation

1.1 Derivation of Robot Dynamics & Equations of Motion

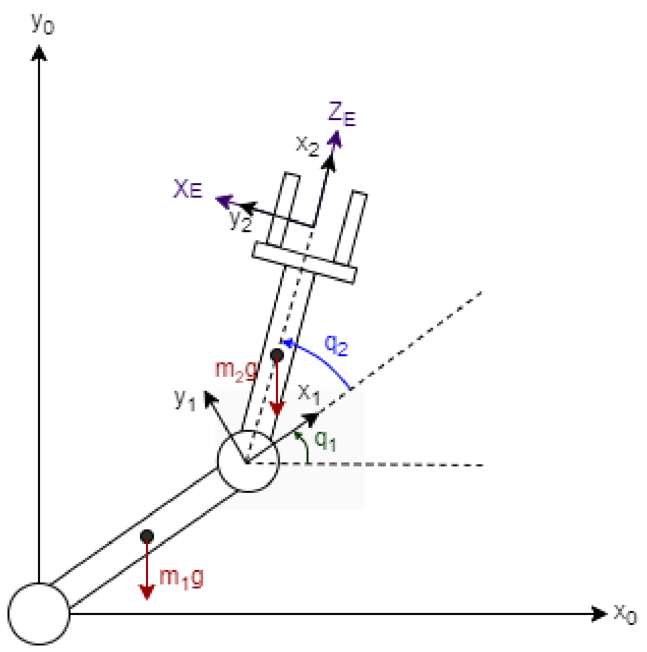

Figure 1: DH Frames for Two link RR Robot

Using the coordinate frames as pictured in Figure 1, and L_1 and L_2 are the lengths of link 1 and link 2 respectively, the DH parameters for the robot are defined as:

$\mathbf{i}$	$\mathbf{a_{i-1}}$	$\alpha_{\mathbf{i-1}}$	$\mathbf{d_i}$	$\mathbf{q_i}$
1	L_1	0	0	q_1
2	L_2	0	0	q_2
3	0	90°	0	90°

For a given joint, i, the angular and linear components of the Jacobian is defined by the relationship between joint torque τ_i and resulting task-space moment and force about frame and point A respectively as follows:

$$\begin{aligned}
\tau_i &= \hat{\mathbf{z}}_i^T \mathbf{M_A} \\
\tau_i &= \mathbf{J}_{\omega i}^T \mathbf{M_A} \\
\mathbf{J}_{\omega i} &= \hat{\mathbf{z}}_i && \textbf{(1)} \\
\tau_i &= (\hat{\mathbf{z}}_i \times \mathbf{p}_{iA})^T \mathbf{f_A} \\
\tau_i &= \mathbf{J}_{vi}^T \mathbf{f_A} \\
\mathbf{J}_{vi}^T &= (\hat{\mathbf{z}}_i \times \mathbf{p}_{iA}) && \textbf{(2)}
\end{aligned}$$

Where $\mathbf{p}_{iA}$ is the vector from joint i to point A.

For link 1 it follows that the Jacobian matrices of the centre of mass are:

$$\begin{aligned}
\mathbf{J}_{\omega 1} &= \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 1 & 0 \end{bmatrix} \\
\mathbf{J}_{v_1} &= \begin{bmatrix} ({}^0\hat{\mathbf{z}}_1 \times {}^0\mathbf{p}_{1,c1}) & (0,0)^T \end{bmatrix} \\
\hat{\mathbf{z}}_0 = \hat{\mathbf{z}}_1 &= \begin{bmatrix} 0 & 0 & 1 \end{bmatrix}^T \\
{}^0\mathbf{p}_{1,c1} &= \begin{bmatrix} -l_{c1}\sin(q_1) \\ l_{c1}\cos(q_1) \\ 0 \end{bmatrix} \\
{}^0\mathbf{p}_{1,c1} &= {}^0_1\mathbf{R} \begin{bmatrix} l_{c1} & 0 & 0 \end{bmatrix} \\
\Rightarrow \mathbf{J}_{v_1} &= \begin{bmatrix} -l_{c1}\sin(q_1) & 0 \\ l_{c1}\cos(q_1) & 0 \end{bmatrix}
\end{aligned}$$

And for link 2:

$$\begin{aligned}
\mathbf{J}_{\omega 1} &= \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 1 & 1 \end{bmatrix} \\
\mathbf{J}_{v_2} &= \begin{bmatrix} ({}^0\hat{\mathbf{z}}_1 \times {}^0\mathbf{p}_{1,c2}) & \mathbf{z}_2 \times {}^0\mathbf{p}_{2,c2} \end{bmatrix} \\
{}^0\mathbf{p}_{1,c2} &= {}^0_1\mathbf{R} \begin{bmatrix} L_1 & 0 & 0 \end{bmatrix} + {}^0_1\mathbf{R}^1_2\mathbf{R} \begin{bmatrix} l_{c2} & 0 & 0 \end{bmatrix} \\
\Rightarrow {}^0\mathbf{p}_{1,c2} &= \begin{bmatrix} L_1\sin(q_1) - l_{c2}\sin(q_1+q_2) \\ L_1\cos(q_1) + l_{c2}\cos(q_1+q_2) \end{bmatrix} \\
{}^0\mathbf{p}_{c2} &= \begin{bmatrix} -l_{c2}\sin(q_1+q_2) \\ l_{c2}\cos(q_1+q2) \end{bmatrix} \\
\hat{\mathbf{z}}_0 = \hat{\mathbf{z}}_1 = \hat{\mathbf{z}}_2 &= \begin{bmatrix} 0 & 0 & 1 \end{bmatrix}^T \\
\mathbf{J}_{v_2} &= \begin{bmatrix} L_1\sin(q_1) - l_{c2}\sin(q_1+q_2) & -l_{c2}\sin(q_1+q_2) \\ L_1\cos(q_1) + l_{c2}\cos(q_1+q_2) & l_{c2}\cos(q_1+q2) \end{bmatrix}
\end{aligned}$$

Expressing the equations of motion as:

$$\tau = \mathbf{A}(q)\ddot{q} + \mathbf{B}(q)[\dot{q}\dot{q}] + \mathbf{C}(q)[\dot{q}^2] + \mathbf{G}(q) \quad \textbf{(3)}$$

The 'Inertia Matrix', $\mathbf{A}$ is defined by the sum of kinetic energies of each link. $\mathbf{B}$ and $\mathbf{C}$ are the 'Coriolis' and 'Centrifugal' terms, and **G** is the force due to gravity. These are defined and computed as follows:

$$
\begin{aligned}
\mathbf{A}(q) &= \sum_{i=1}^{N} \left(m_i, \mathbf{J}_{vi}^T \mathbf{J}_{vi} + \mathbf{J}_{\omega i}^T \mathbf{I}_{ci} \mathbf{J}_{\omega i} \right) \quad [1] \\
a_{ijk} &= \frac{\partial a_{ij}}{\partial q_k} \\
a_{ij} &= \mathbf{A}[i, j] \\
b_{i,jk} &\triangleq \frac{1}{2}(a_{ijk} + a_{ikj} - a_{jki}) \\
\mathbf{B}(q) &\triangleq 2 \begin{bmatrix} b_{1,12} & \dots & b_{1,1n} & b_{1,23} & \dots & b_{1,2n} & \dots & b_{1,(n-1)n} \\ b_{2,12} & \dots & b_{2,1n} & b_{2,23} & \dots & b_{2,2n} & \dots & b_{2,(n-1)n} \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ b_{n,12} & \dots & b_{n,1n} & b_{n,23} & \dots & b_{n,2n} & \dots & n_{2,(n-1)n} \end{bmatrix} \\
\Rightarrow \mathbf{B}(q) &= -2L_1 m_2 l_{c2} \sin(q_2) \\
\mathbf{C}(q) &\triangleq \begin{bmatrix} b_{1,11} & b_{1,22} & \dots & b_{1,nn} \\ b_{2,11} & b_{2,22} & \dots & b_{2,nn} \\ \dots & \dots & \dots & \dots \\ b_{n,11} & b_{n,22} & \dots & n_{2,nn} \end{bmatrix} \\
\Rightarrow \mathbf{C}(q) &= \begin{bmatrix} 0 & -L_1 m_2 l_{c2} \sin(q_2) \\ L_1 m_2 l_{c2} \sin(q_2) & 0 \end{bmatrix} \\
\mathbf{G}(q) &\triangleq \sum_{i=1}^{N} \mathbf{J}_{vi}^T (-m_i g \hat{\mathbf{y}}_0) \\
\mathbf{G}(q) &= -g \begin{bmatrix} m_2 l_{c2} \cos(q_1 + q_2) + \cos(q_1)(m_1 l_{c1} + L_1 m_2) \\ m_2 l_{c2} \cos(q_1 + q_2) \end{bmatrix}
\end{aligned}
$$

Friction in each joint is proportional to it's velocity:

$$\mathbf{f_{fr}} = \mu \begin{bmatrix} \dot{q}_1 \\ \dot{q}_2 \end{bmatrix}$$

Where $\mu = 0.1$.

Rearranging **(3)**, and including friction gives the final equations of motion for the robot:

$$\ddot{q} = \mathbf{A}^{-1} \left(\begin{bmatrix} \tau_1 \\ \tau_2 \end{bmatrix} - \mu \begin{bmatrix} \dot{q}_1 \\ \dot{q}_2 \end{bmatrix} - \mathbf{B} \dot{q}_1 \dot{q}_2 - \mathbf{C} \begin{bmatrix} \dot{q}_1^2 \\ \dot{q}_2^2 \end{bmatrix} - \mathbf{G} \right)$$

1.2 Unactuated Motion Simulation

To test the equations of motion, the robot was simulated with $\tau = \mathbf{0}$, with initial conditions:

$$
\begin{aligned}
q_1(0) = q_2(0) = 0 \\
\dot{q}_1(0) = \dot{q}_2(0) = 0
\end{aligned}
$$

The resulting joint angles can be seen in Figure 2.

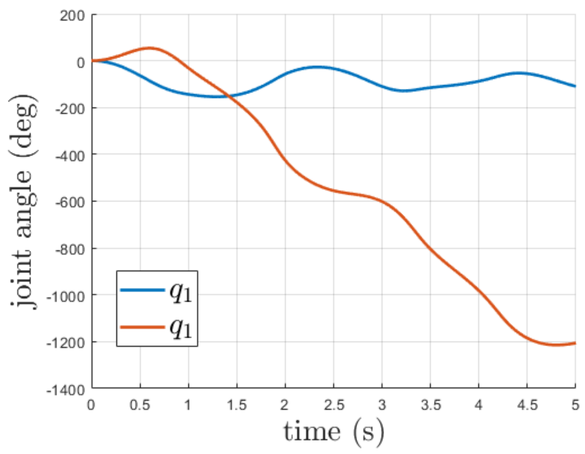

Figure 2: Unactuated Motion

1.3 Robot Vs. Gravity

To compensate for gravity, the robot simply sets $\tau(t) = \mathbf{G}(q(t))$, since the robot is initially at rest, and all other terms in the equations of motion depend only on joint velocity, this results in the robot remaining stationary, as can be seen in Figure 4.

[1]Solving for our robot:

$$\mathbf{A}(q) = \begin{bmatrix} m_2 L_1^2 + 2m_2 L_1 l_{c2} \cos(q_2) + m_1 l_{c1}^2 + m_2 l_{c2}^2 + I_{zz1} + I_{zz2} & m_2 l_{c2}^2 + L_1 m_2 l_{c2} \cos(q_2) + I_{zz2} \\ m_2 l_{c2}^2 + L_1 m_2 l_{c2} \cos(q_2) + I_{zz2} & m_2 l_{c2}^2 + I_{zz2} \end{bmatrix}$$

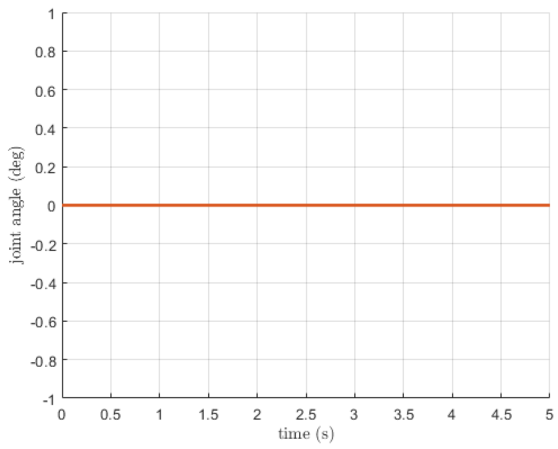

Figure 3: $\tau = \mathbf{G}$

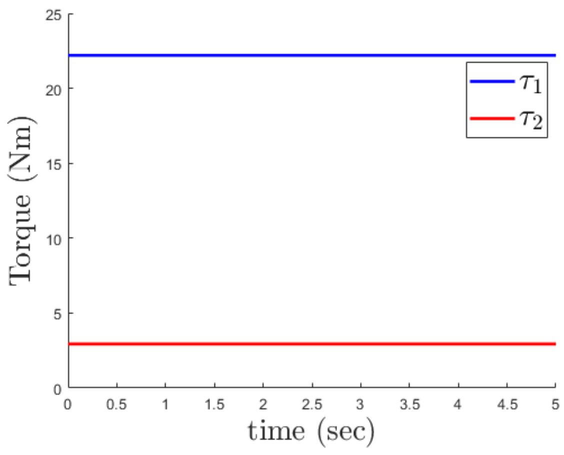

Figure 4: $\tau = \mathbf{G}$

This only works if the robot knows has an exactly perfect model, initializing the Robot object with even the slightest error in physical parameters (see Section 6 for details on matlab code structure) results in complete failure of this open loop approach. Figure 5 shows that even a 0.1% modelling error causes unbounded growth in position and velocity.

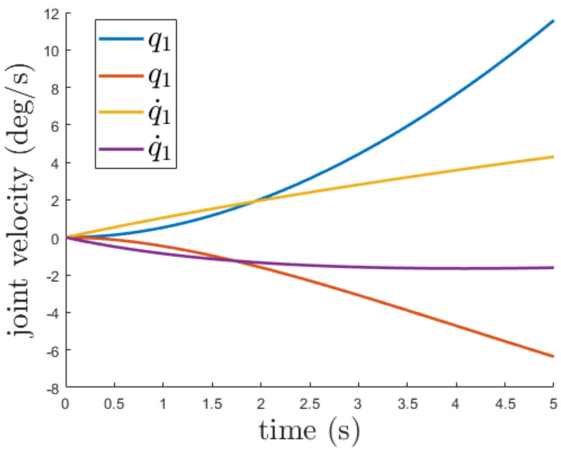

Figure 5: 0.1% model error

2 Joint Space PID Controller

2.1 PID Controller Design

Using MATLAB's pid() function to generate a MIMO PID controller the following controller was implemented:

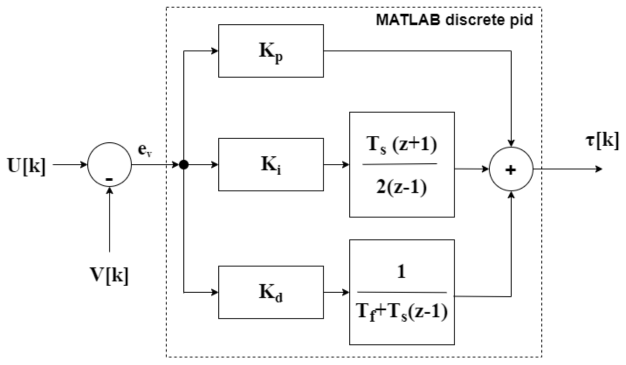

Figure 6: Controller Block Diagram

PID gains were chosen through trial and error to be:

$$\begin{aligned}K_{p1} &= 100\\ K_{p2} &= 20\\ K_{i1} &= 200\\ K_{i2} &= 50\\ K_{d1} &= 0.5\\ K_{d2} &= 0.05\end{aligned}$$

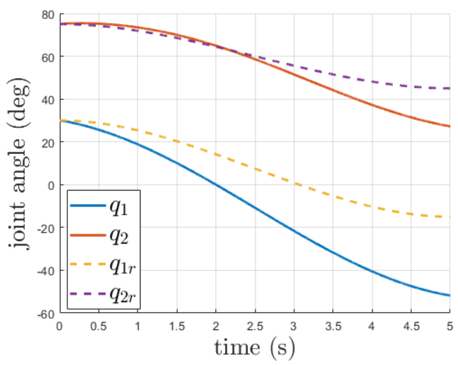

Figure 7: Joint Angles & Reference

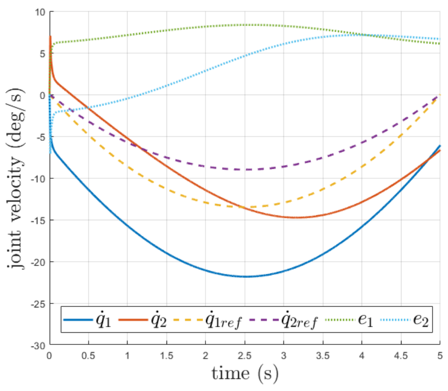

Figure 8: Joint Velocities & Reference

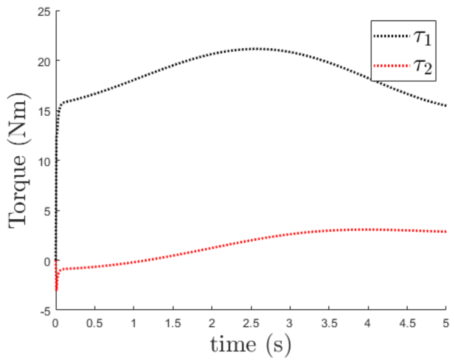

Figure 9: Applied Joint Torques

The resulting motion from the reference joint velocity is shown in Figures 7-8, and the applied joint torque can be seen in Figure ??. Since this controller does not have any consideration for plant dynamics, it is quite poor at tracking the reference velocity; because of this, position is shown to drift away from the reference and fails to arrive at the desired position.

2.2 Improved Controller

To improve to controller's performance, **G** estimation was added to the output of the PID controller to help compensate for gravity while the PID tracks the reference.

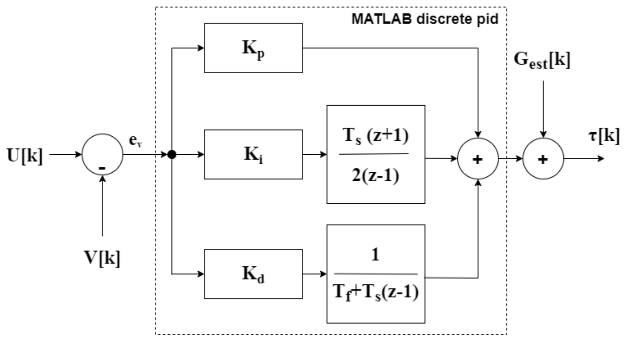

Figure 10: Augmented Controller

2.3 Simulation Results

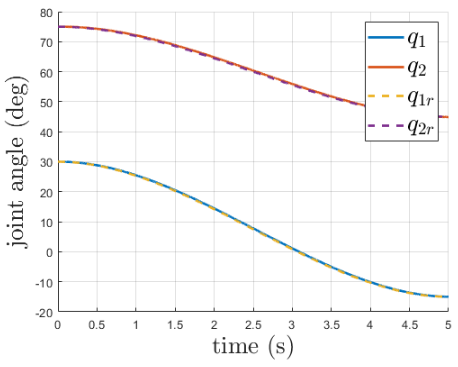

Figure 11: Joint Angles

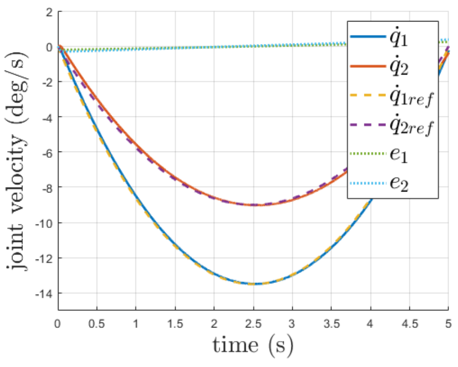

Figure 12: Joint Velocities

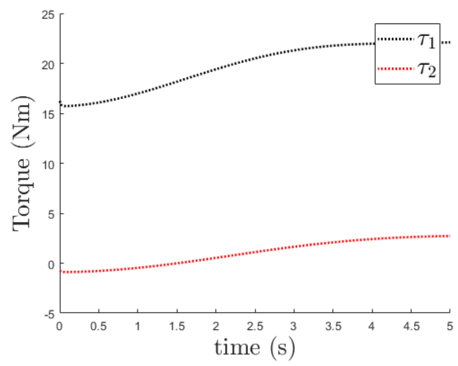

Figure 13: Joint Torques

2.4 Proof of Robustness

Because this controller has velocity feedback, unlike previously, even with all a relatively large model error (applied to every physical parameter in the simulation) the improved controller performs better than the PID alone. The following were simulated with a 5% model error.

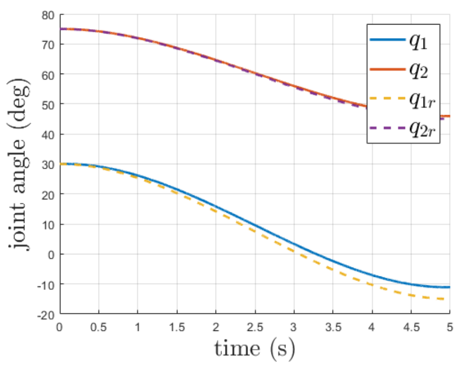

Figure 14: Joint Angles

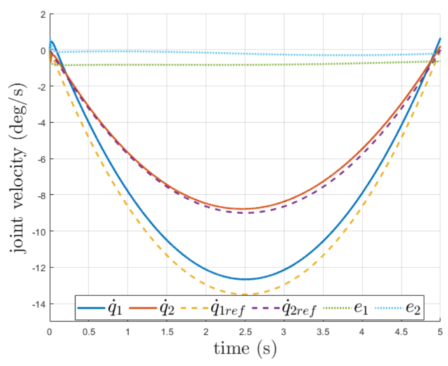

Figure 15: Joint Velocities

2.5 Resulting Motion

Figures 16-18 show a snap-shot of the robot's pose every 0.5 seconds, both with and without model error.

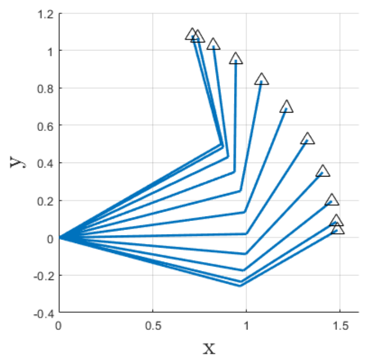

Figure 16: 0% Model Error

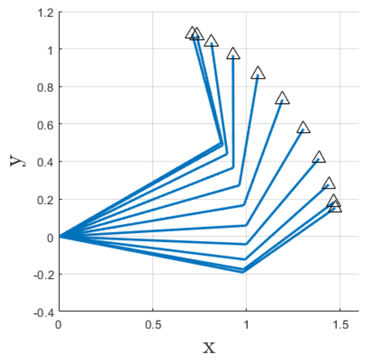

Figure 17: 5% Model Error

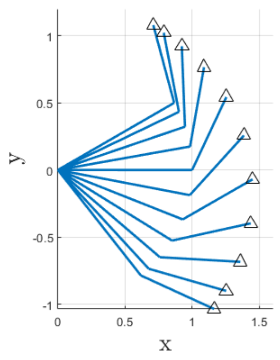

Figure 18: PID Without G Compensation

3 Joint Space Control

3.1 End Effector Paths

Figure 26 shows the robot perfectly follows the reference trajectory. This is not the case however for trajectory B, since the straight line path goes outside the reachable workspace. Because the trajectory and controller at this stage cannot account for this, the robot does not reach it's intended end state.

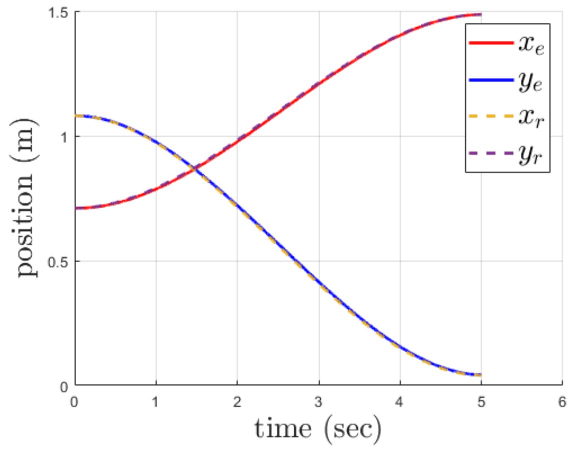

Figure 19: Trajectory A

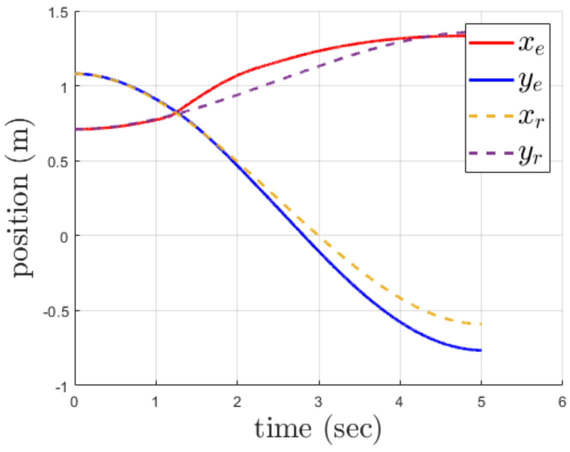

Figure 20: Trajectory B

3.2 Resulting Motion

Despite the start and end positions for trajectory A being the same as in Section 2, the robot takes a very different path because the reference trajectory was computed in task space. This is because the cubic reference polynomials create affine paths in the positional frame in which they are generated, i.e. the reference trajectory in Figure 21 is affine in the (x,y) plane, and the reference trajectory is affine in the (q_1, q_2) plane. Since q_1 and q_2 are revolute, the resulting path is arched.

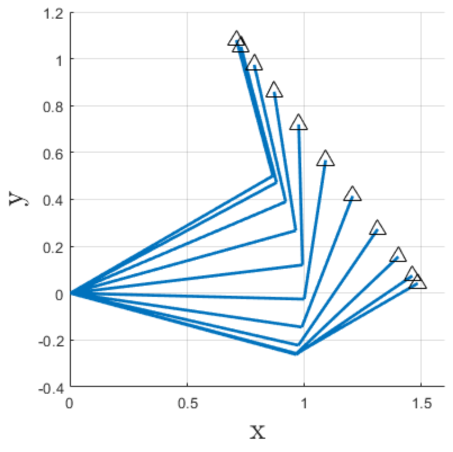

Figure 21: Trajectory A

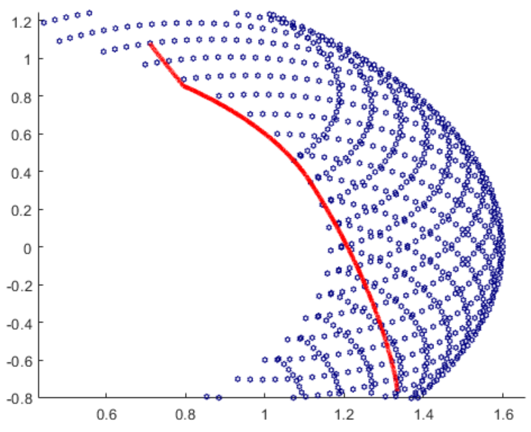

Figure 22: Reachable Workspace

4 Task Space Control

As expected, the robot still follows trajectory A perfectly under task space control. It also successfully recovers in trajectory B and manages to reach very close to it's intended position, since it has position feedback. If the robot was still using PID without the gravity estimator, this would be the first time the robot perfectly followed any path, because the PID isn't particularly well tuned on it's own.

4.1 End Effector Path

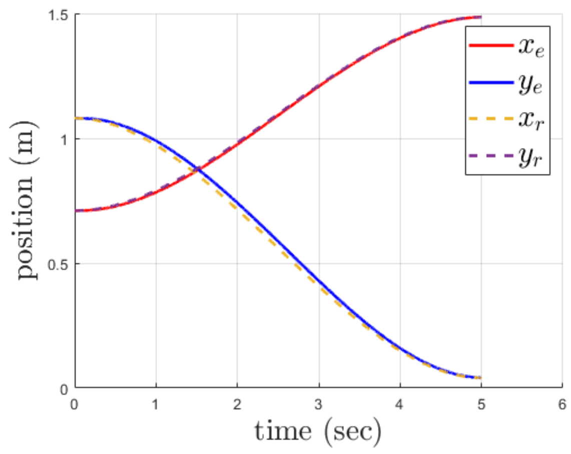

Figure 23: Trajectory A

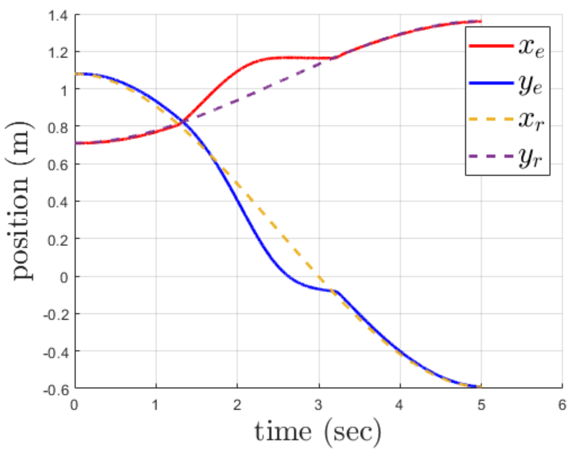

Figure 24: Trajectory B

4.2 Resulting Motion

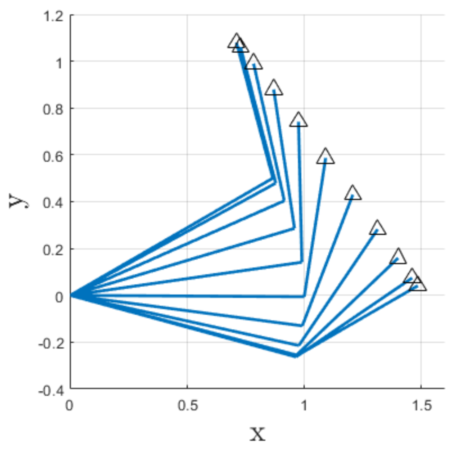

Figure 25: Trajectory A

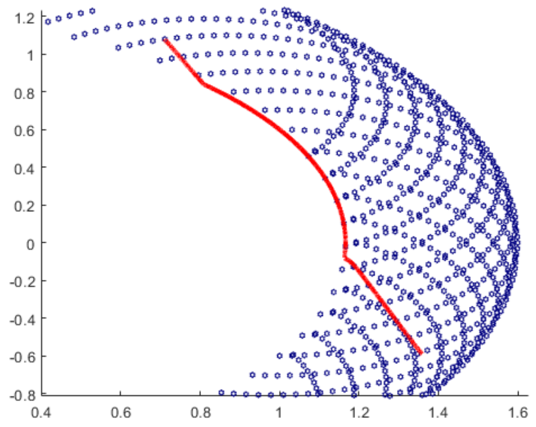

Figure 26: Trajectory B

5 MATLAB File Structure

Please note:

- Every task uses formulteDynamics.m to generate A,B,C,G matrices
- Task 1&2 use Task1_dynamics.m for ode45
- Task 3&4 use Task3_dynamics.m for ode45
- Every task uses the same Robot.m and State.m class file, so modifications to these files propagate
- Task 3&4 use trajectoryGen.m to generate anonymous functions of time for reference trajectory
- Task 2's trajectory coefficients are hard coded.
- Set anim to 1 to generate the plots the pose every 5 sec