RBE 502 Homework 6

Zachary Serocki

October 17, 2025

Contents

1 PD Controller with Gravity Compensation

1.1 Simulation

To tune the gains for the PD Controller with Gravity Compensation and later the rest of the controllers we used the idealized dynamics to allow the controller to converge with zero error. The controller were tuned to be critically damped which we determined was a good feature for running on the hardware.

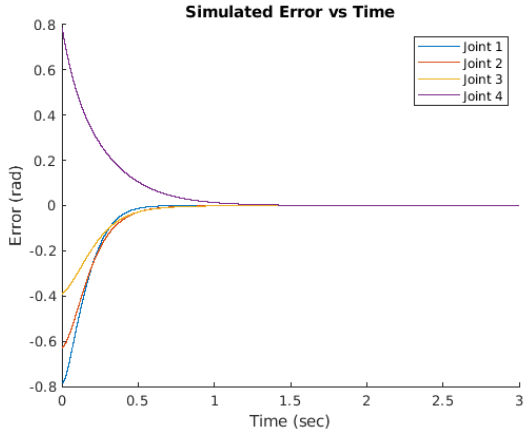

Figure 1: The PD Controller with Gravity Compensation using ideal dynamics. Each joint converges to the reference position with no overshoot.

As seen in Figure 1 with idealized dynamics the controller was able to converge to the setpoint without steady state error. This is what we would expect going off of previous iterations of this controller when given ideal masses and link lengths the controller preforms well. There is no way to control the rate of convergence since the controller is a point to point controller and it is unable to follow a trajectory, this means that for larger gains the controller will oscillate about the reference position.

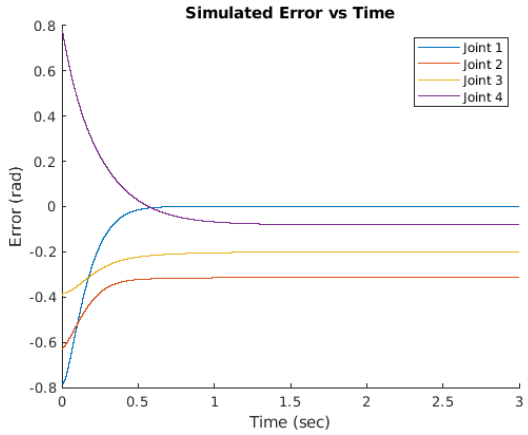

Figure 2: The PD Controller with Gravity Compensation using biased dynamics. The verticle joints are unable to congerge to the referance as it is unable to overcome the gravity term pulling it up; the base joint converges.

As seen in Figure 2 when given Biased dynamics the controller fails to converge to the reference position this is because due to joints applying too much gravity compensation forcing the PD controller portion too try and compensate and since there isn't a way to handle steady state error with this controller the joints never converge to the reference state. The base joint converges since it does not fight gravity and acts as a LTI system when the joints aren't flailing and shifting the moment of inertia.

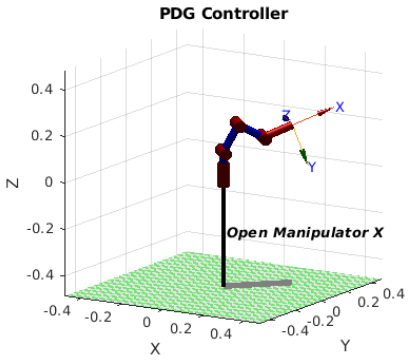

Figure 3: The robots configuration at the end of the movement.

1.2 Hardware

The robot was run with predetermined masses and inertia of the links to give the controller the best chance at convergence. These values were determined in RBE 501, where the robot was disassembled, and the links and motors were weighted. The links were then analyzed in Solidworks to determine the rotational inertia. With the dynamic parameters determined we could then use them to control the robot and get accurate results.

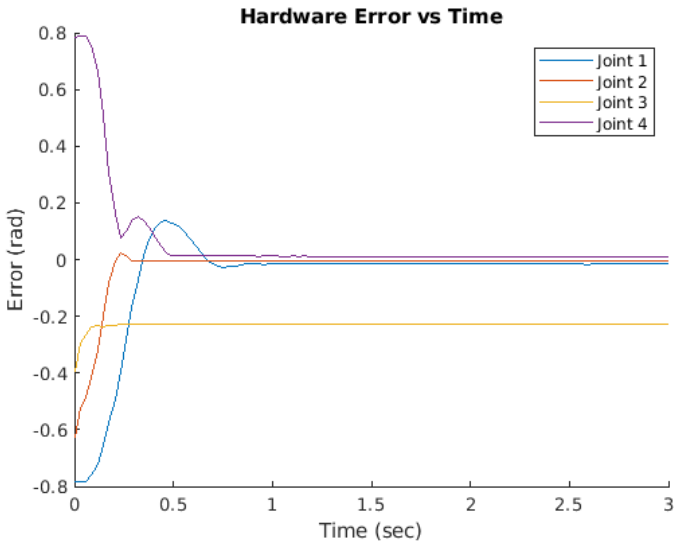

Figure 4: Joints one, two, and four are able to converge to the reference position with minimal overshoot and steady state error. Joint 3 is unable to converge to the reference position.

Video: https://www.youtube.com/watch?v=j9k-BtRgL0c
The for most as seen in Figure 4 the joints the controller was able to converge to very close too the reference value with minimal overshoot and steady state error however, for joint 3 it was unable to converge to the reference position, since the steady state error is negative it implies that the joints are above the reference value implying that the predicted masses are more than the true masses.

Overall, the PD controller with gravity compensation is not a robust controller and will not converge if there is uncertainty in the parameters. The PD controller is a simple controller that took minimal effort to implement and tune which made it a great light weight option. the controllers parameters were easy to tune as the response isn't a convolution of the convergence parameters like in other controllers.

2 Computed Torque Controller

2.1 Simulation

The Computed Torque Controller with overestimated mass could not converge to the target position. With ideal dynamics, the robot was able to converge with negligible error. One of the key advantages of the computed torque controller is that it can follow a trajectory. A quintic trajectory is produced for each joint, starting and ending at rest to highlight this feature. As seen in Figure 5 the controller can follow the trajectory with ideal dynamics.

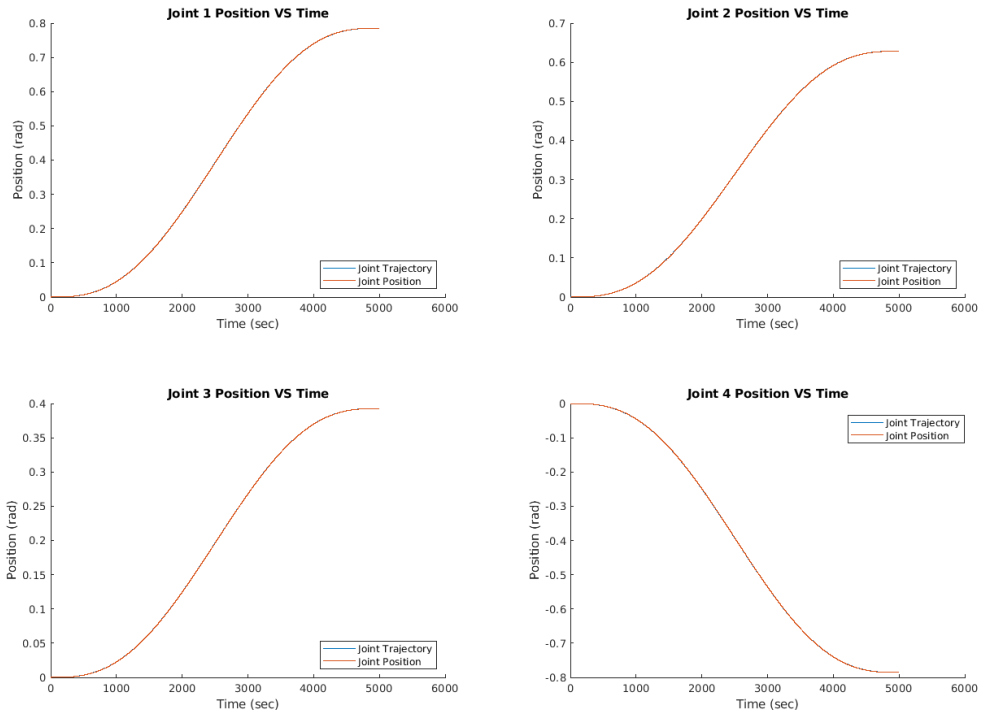

Figure 5: The Computed Torque Controller using ideal dynamics running a quintic trajectory from the start too the end position. Each of the joints follows the reference trajectories very closely as we would expect for ideal dynamics.

The Error Plots show that the robot is able to follow the trajectory throughout the movement as seen in Figure 6 the magnitude of the error is on the order of 10^{-4} which implies that when the controller is given the correct dynamics it is able to converge, implying a working controller.

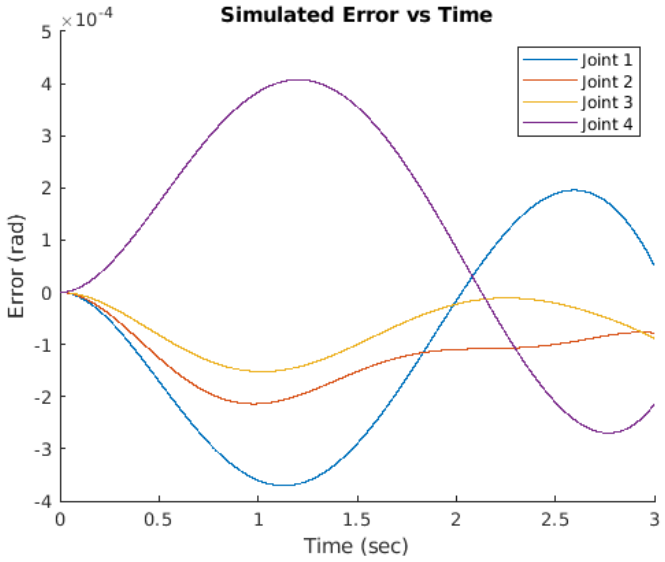

Figure 6: The error in Computed Torque Controller using ideal dynamics running a quintic trajectory from the start too the end position. Each Joint follows the reference closely with the error being on the magnitude of 10^{-4}

The response of the controller without the ideal dynamics is a different story as the controller relies on having correct information about the system to determine the motor torques as it has no mechanism to combat steady state error. The simulation jumps up and down

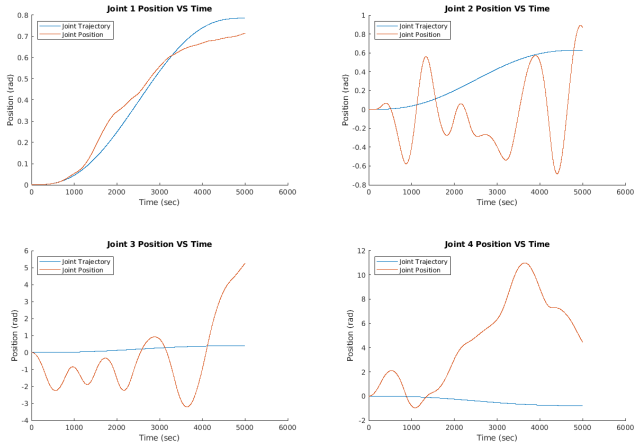

Figure 7: The Computed Torque Controller joint positions using biased dynamics was marginally stable as seen in the difference in the reference and the actual position. The vertical three joints oscillated in a dynamic response with one another.

As Seen in Figure 7 none of the joints made it too the reference position. Joint 1 somewhat follows reference trajectory but has moments of moving too fast or too slow this is caused by the robot using the biased masses to determine the torque on the base motor where to start it moving the motor would predict that it takes more torque to accelerate the heavier links, It would do the same overshoot when decelerating the links which explains the shape of the curve why it overshoots in the beginning and undershoots in the end. It also experiences perturbed factors from the other three links moving and changing the rotational inertia. The other three joints experience an oscillation pattern that is caused by the controller predicting more torque to hold up the controller than is required and thus the links accelerate up it then repeats this on the way down where it predicts more torque to change the

direction of motion.

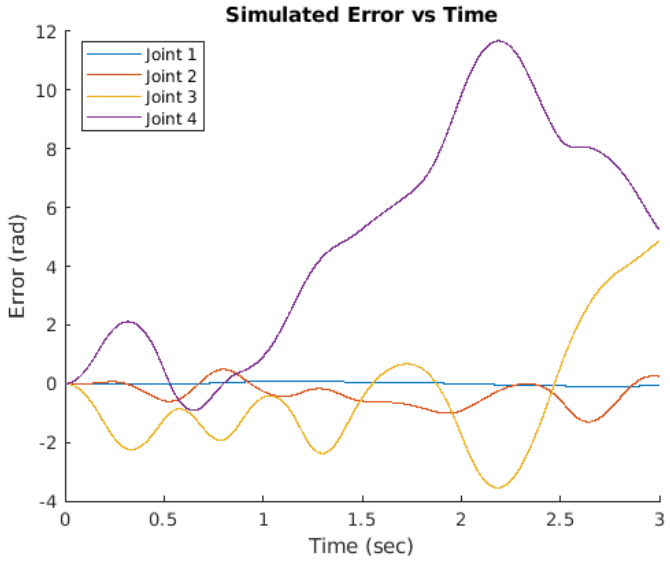

Figure 8: The error in Computed Torque Controller using biased dynamics tended to oscillate about some error value. The oscillations of joints 2 and 3 are almost completely out of phase from one another. This being said the third joint seems as though it is experiencing constructive interference in its motion with the other vertical joints.

The joints' error shows interesting trends where the oscillations of the second and third joints are almost entirely out of phase from one another. The third joint seems to be experiencing constructive interference with the other joints' oscillating, and when the second joint reaches its maximum error, the third joint also reaches its maximum value. The perturbations in the fourth joint seem to follow the oscillations in the second joint. Additionally, the joints tend to be oscillating about a steady state error. This is correlated to the difference in the mass of the links.

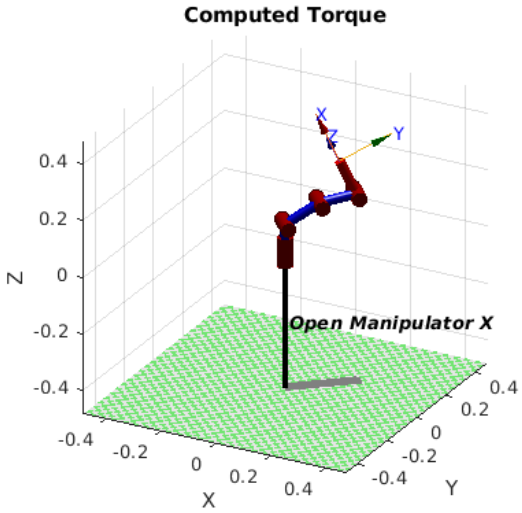

Figure 9: The robots configuration at the end of the movement.

2.2 Hardware

The robot was run with predetermined masses and inertia of the links to give the controller the best chance at convergence. These values were determined in RBE 501, where the robot was disassembled, and the links and motors were weighted. The links were then analyzed in Solidworks to determine the rotational inertia. With the dynamic parameters determined we could then use them to control the robot; these parameters are critical for predicting the torques on the motors when following a trajectory.

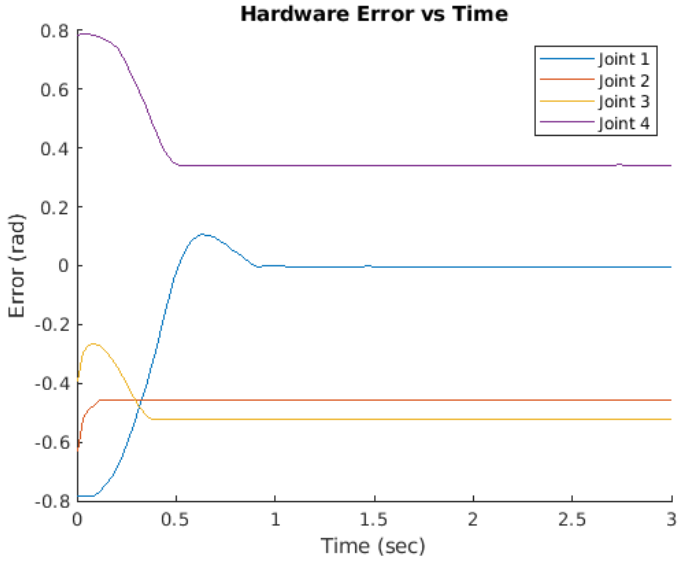

Figure 10: Step response of Computed torque controller

Video: https://www.youtube.com/watch?v=25u30wCJrAw

The robot did not converge to the reference position, the only joint that converged is the base motor. The rest of the robot joints were above the setpoint implying that the estimated parameters were significantly more than the true parameters.

The Computed Torque controller performed well with idealized dynamics but failed when given biased dynamics. The controller was marginally stable. As the margin of error increases in the masses, the error is amplified on the output. When running on the robot, we wouldn't expect it to converge as the simulation shows it is not robust. Running the controller with a trajectory has no overshoot in the idealized case. This is because the controller only has to make minor corrections to torque to keep it on the path in this scenario.

3 Robust Computed Torque

3.1 Simulation

The Robust Computed Torque Controller is similar too the computed torque controller just has a correction applied to it to account for disturbances. The robust computed torque is intended to fix the problem with computed torque controller when there is uncertainty in the parameters. The Robust Computed Torque controllers were run with a quintic trajectory in the joint space starting and ending at rest.

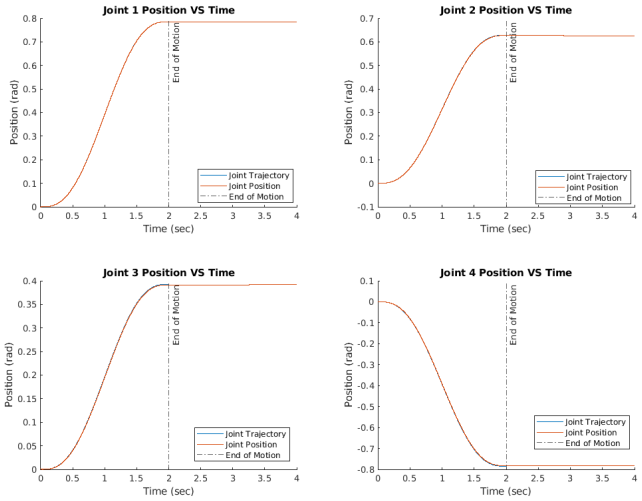

Figure 11: Robust Computed torque controller following a quintic trajectory to the reference. the Joints are able to stay with the trajectory with minimal deviation.

The Robust Computed Torque Controller was able to follow the reference trajectory closely. Each joint was able too converge to the end position without deviating from the path too much. As mentioned during the demonstration the simulation was extended to show what affects it has on the joint positions the trajectory is run from zero to two seconds the lasing affects are better seen in Figure 12.

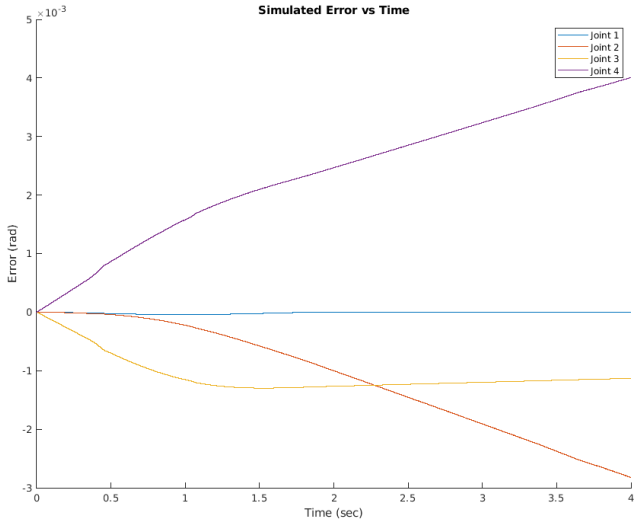

Figure 12: Error of the joint positions throughout the trajectory

The joints were able to converge very close to the reference. The error does appear to keep diverging as time goes on; however, this trend is very minimal and likely has to do with how the simulation works more than the controller as discrete steps are taken in our version of the simulator in order to provide as close of an experience to the actual robot as possible. In contrast, the ODE45 version uses different methods to approximate the actual dynamics of the robot better. This can be said since we expect that a large change in rho would be needed to compensate; however, the trend persisted when changing rho by two orders of magnitude. It was just scaled smaller; changing P, the convergence rate of the controller was able to slow down the divergence but it was not enough to eliminate the trend.

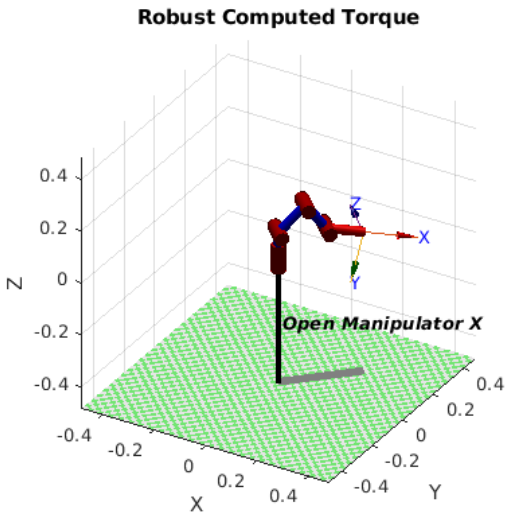

Figure 13: he robots configuration at the end of the movement.

3.2 Hardware

The robot was run with predetermined masses and inertia of the links to give the controller the best chance at convergence. These values were determined in RBE 501, where the robot was disassembled, and the links and motors were weighted. The links were then analyzed in Solidworks to determine the rotational inertia. With the dynamic parameters determined we could then use them to control the robot; these parameters have wiggle room as with the introduction of the correction term allows for the any disturbances to be handled. The disturbances of the controller have to be less than the magnitude of ρ to ensure that the correction can adequately compensate.

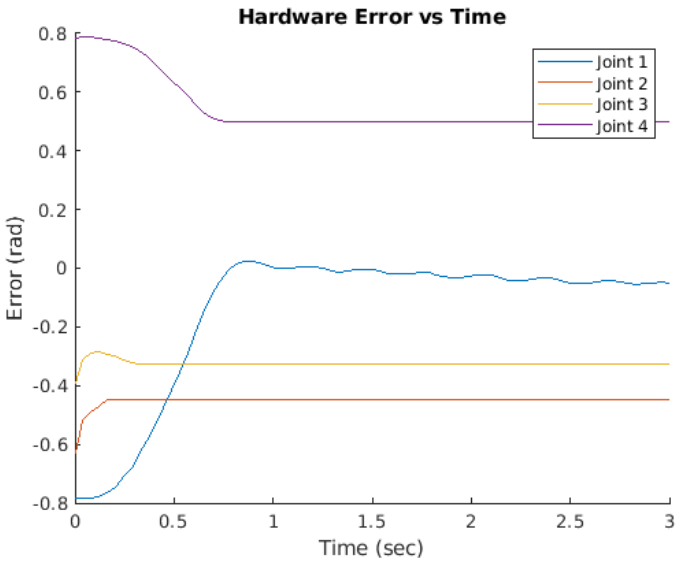

Figure 14: Step response of the Robust Computed torque Controller

Video: https://www.youtube.com/watch?v=sy0_ZAxJ_2o

The robot converged to a position that was closer to the target position than the computed torque. The robust term allowed it to get closer in the end. To overcome the steady state error, it would have had to be modified to individually compensate for the error in each of the joints as upping rho induced jitteriness in the first joint; the more we turned it up, the more we couldn't increase rho anymore. This effect can be seen in joint 1, where the joint associates when it is close to the target.

The Robust Computed Torque Controller converged very close to the reference value in the simulation and on the hardware. This is expected behavior; one thing that stands out is the behavior of the computed torque controller is very erratic; however, when the correction is applied, the robot is very controlled even when put under the same disadvantages as the others. The robust controller is a lightweight way for compensating for biased parameters and still overcome the challenges.

4 Adaptive Control

4.1 Simulation

The adaptive controller is a controller that aims to parametrize the system during the trajectory. The controller was given biased parameters at the start, and it would then revise the parameters to converge to zero error. This requires significant computation for each iteration; however, it can adapt to changing dynamics, which is very valuable in many systems.

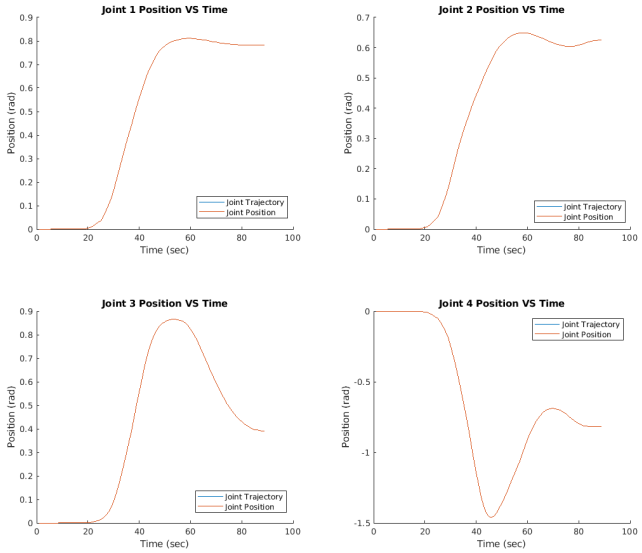

Figure 15: Joint positions of the robot as it converges to the reference value using the adaptive controller

The paths of the joints to the endpoints are fairly unusual as they change the robot's parameters along its trajectory, which has its own dynamic effects. Remnants of the biased dynamics can be seen where every joint overshoots the target and then converges to the values that allow the robot to converge to zero error.

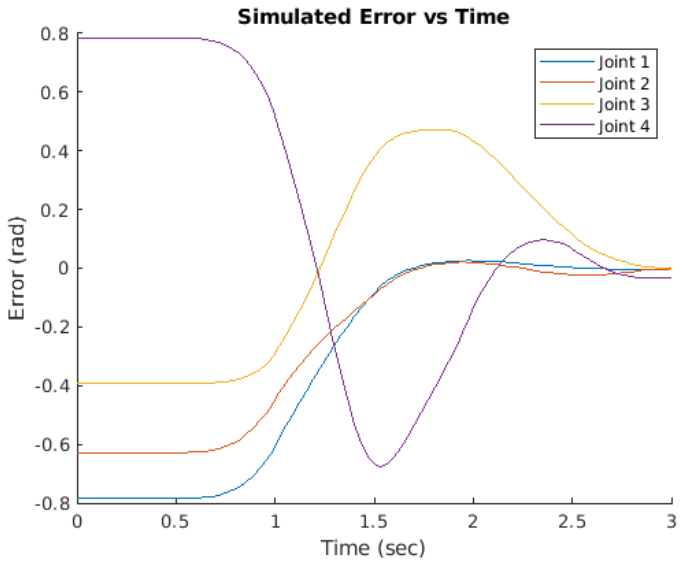

Figure 16: Step response of adaptive controller plotting the error vs time. Each of the joints overshot the target position initially then converged to the reference as the parameters were revised.

The errors of the controllers tell a similar story where they all overshoot the target and then converge too the reference value. The robot dosn't converge to zero error in the time that the controller runs for but when the controller runs for longer it has more opportunity to converge to zero positional error. The robot starts out by overshooting the target then realizing that it doesn't have to apply that much force to get it to accelerate bu the time that it gets back up it has almost fully converged.

4.2 Hardware

Running the adaptive controller on the hardware we have it the ball and stick dynamics that assumes all the mass is at the end of the links giving it an over estimate of torques required. The regressor was then used to contentiously approximate the parameters of the robot in trials where the robust computed torque is used to steer the motion and adaptive is used to determine the parameters we see that the parameters fluctuate along the motion this is unexpected as we would assume that the parameters stay constant as the robot completes the motion. This is contributed to the regressor being a simplification of the dynamics at play and therefore it has to adapt to the differences in the dynamics.

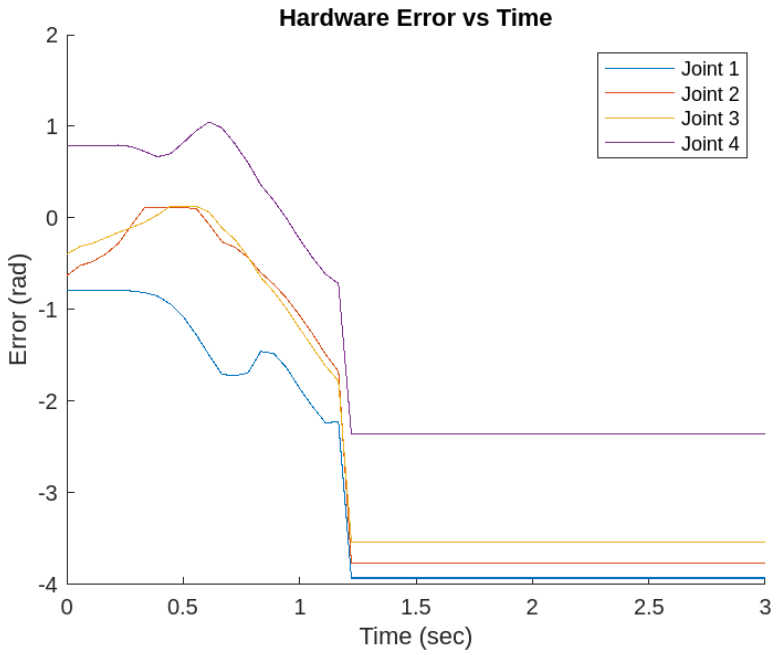

Figure 17: Step response of adaptive controller running on the robot. The robot precede to crash into the floor at the 1.25 second mark. Before that the robot did not apply enough torque to overcome gravity.

Video:https://www.youtube.com/watch?v=wjD57fxVsvU

On the hardware the robot would either crash into the floor as seen in Figure 17 or would swing about wildly. We never were able to get the adaptive controller to work successfully on the robot due to physical limitations about the robot and software limitations as in simulation under the right circumstances the parameters would approach infinity and become unstable.

Adaptive Control worked in simulation but on the hardware was unstable. This could be attributed to a number of things one of which is that the torques calculated by the regressor function were significantly different from known torque convergent values. In efforts to fix this we attempted to make our own regressor but never got the regressor to properly compute the torque and such the provided regressor was used. The potential of this controller is very intriguing as it can be used in applications where the parameters that you are trying to estimate and compensate for are linear with resect to the output.

5 Simulation Framework

To simulate the robot, we created a simulation software that uses the masses and inertia we measured from the links to make it more realistic. The dynamics were used to find the forward dynamics then take a small timestep simulating the joint accelerations and velocities. Using spacial inertia matrices that were determined prior to the project allowed our on robot controllers to preform very well out of the box. The software is designed to use the same script for the robot and a simulated response by just changing one variable. Creating this interfacing pattern makes porting code extremely simple and allows code reuse. The simulation was used to try different gains before going to the robot; we found that, in general, the gains tended to undershoot the experimentally determined gains. Using the framework we were able to program in an environment that was easily convertible to the robot allowing for minimal repeat of code.