# MEGN 544: Project 3

## Introduction

The problem definition for problem 3 is to create a dynamic simulation of a 6-link ABB robotic arm and design a robust controller that allows for the arm's manipulator to follow a desired trajectory, which is to write the letters "CSM" on a 2D plane without lifting the manipulator from the plane. The arm should traverse along the trajectory with joint 6's Z axis orthogonal to the plane of the letters and joint 6's X axis oriented along the trajectory path. The initial swing-up should take 2 seconds, the arm should then pause for 0.5 seconds, and finally traverse the path in 5 seconds. Also, the arm speed while traversing the path should be approximately constant. The controller should be designed in Simulink using the forward kinematics functions from project 1, the inverse kinematics & dynamics functions from project 2, and the two following control methods discussed in class:

1. Configuration Space Inverse Dynamics Controller
2. Operational Space Inverse Dynamics Controller

After settling on a working Kp & Kd for each controller, it is obvious that the second of the two is more robust. The Operational Space Inverse Dynamics Controller is smoother and has a much smaller error throughout the entire trajectory. On the other hand, the Configuration Space Inverse Dynamics Controller deviates from the trajectory when traversing from the letter "C" to "S", the quickly returns to the trajectory. My idea for why this happens will be discussed in detail in the *Controller Performance* section below. Overall though, both controllers followed the desired trajectory while satisfying the constraints given above.

Controller Descriptions

*Part 1: Configuration Space Inverse Dynamics Controller*

The following equation defines the algebraic control law that has been implemented for part 1. Reference Appendix A, Figure 1 for a block diagram of the resulting controlled system and Appendix C for definitions of variables in the equation below.

$$\tau_{cont} = G(\theta_a) + C(\theta_a, \dot{\theta}_a) + F + M(\theta)\ddot{\theta}_{cont}$$

$$where\ \ddot{\theta}_{cont} = \left(\ddot{\theta}_d + k_p(\theta_d - \theta_a) + k_d(\dot{\theta}_d - \dot{\theta}_a)\right)$$

In the block diagram, $\theta_d$, $\dot{\theta}_d$, and $\ddot{\theta}_d$ are calculated in the Interp_Target function. $\theta_a$ and $\dot{\theta}_a$ are outputs of the Simulink model, and are fed back to be subtracted from $\theta_d$ and $\dot{\theta}_d$.Then, $(\theta_d - \theta_a)$ is multiplied by $k_p$ and $(\dot{\theta}_d - \dot{\theta}_a)$ is multiplied by $k_d$. These values are then summed with $\ddot{\theta}_d$ to give $\ddot{\theta}_{cont}$. Feeding $\ddot{\theta}_{cont}$ with $\theta_a$ and $\dot{\theta}_a$ into the newton_euler function calculates $G(\theta_a)$, $C(\theta_a, \dot{\theta}_a)$, and $M(\theta)$ which allows us to solve for $\tau_{cont}$ without $F$. I am assuming that $F$ is included in the ABB Arm Dynamics function, where $\tau_{cont}$ is input giving us $\theta_a$ and $\dot{\theta}_a$.

For this control model, I used the following gain values:

$$k_p = 1000$$

$$k_d = 150$$

After running multiple simulations on my controller with a variety of different gains, these seemed to allow for the manipulator to trace the trajectory as smooth and close as possible. Reference Figures in Appendix B for trajectory plots & position errors.

*Parts 2 & 3: Operational Space Inverse Dynamics Controller*

The following equation defines the algebraic control law that has been implemented for parts 2 & 3. Reference Appendix A, Figures 2 & 3 for a block diagrams of the resulting controlled systems and Appendix C for definitions of variables in the equation below.

$$\tau_{cont} = G(\theta_a) + C\left(\theta_a, \dot{\theta}_a\right) + F + M(\theta)\ddot{\theta}_{cont}$$

$$where\ \ddot{\theta}_{cont} = \left({J_v}^{-1}\right)\ddot{x}_{cont}$$

$$and\ \ddot{x}_{cont} = \left(\ddot{x}_d + k_p(x_d - x_a) + k_d(\dot{x}_d - \dot{x}_a) - \dot{J}_v\dot{\theta}_a\right)$$

In the block diagram, $\dot{x}_d$, and $\ddot{x}_d$ are calculated in the Interp_Target function to be [6x1] vectors where:

$$\dot{x}_d = [\dot{x}_d \quad \dot{y}_d \quad \dot{z}_d \quad w_{x,d} \quad w_{y,d} \quad w_{z,d}]^T$$

$$\ddot{x}_d = [\ddot{x}_d \quad \ddot{y}_d \quad \ddot{z}_d \quad \dot{w}_{x,d} \quad \dot{w}_{y,d} \quad \dot{w}_{z,d}]^T$$

Interp_Target also outputs a [7x1] vector that is of the following form:

$$[x_d \quad y_d \quad z_d \quad Q_d]^T$$

This is fed into the x_err function with $T_{act}$(see $T_{act}$ in Appendix C for difference between parts 2 & 3), which gives us $(x_d - x_a)$. $\theta_a$ and $\dot{\theta}_a$ are outputs of the Simulink model, and are fed back input into the velJac_Actual function, which provides $\dot{x}_a$ and $\dot{J}_v\dot{\theta}_a$. This allows us to find $(\dot{x}_d - \dot{x}_a)$, which is multiplied by $k_d$. Now, we have all of the required terms to find $\ddot{x}_{cont}$. Once calculated, $\ddot{x}_{cont}$ is fed into the Theta_dd_act function with ${J_v}^{-1}$(found in the velJac_Actual function) and we can calculate $\ddot{\theta}_{cont}$. Feeding $\ddot{\theta}_{cont}$ with $\theta_a$ and $\dot{\theta}_a$ into the newton_euler function calculates $G(\theta_a)$, $C\left(\theta_a, \dot{\theta}_a\right)$, and $M(\theta)$ which allows us to solve for $\tau_{cont}$ without $F$. I am assuming that $F$ is included in the ABB Arm Dynamics function, where $\tau_{cont}$ is input giving us $\theta_a$ and $\dot{\theta}_a$.

For both of these control models, I used the following gain values:

$$k_p = 3000$$

$$k_d = 350$$

After running multiple simulations on my controller with a variety of different gains, these seemed to allow for the manipulator to trace the trajectory as smooth and close as possible. Reference Figures in Appendix B for trajectory plots & position errors.

Controller Performance (1-2 pages)

*Part 1: Configuration Space Inverse Dynamics Controller*

The manipulator tracks the desired trajectory reasonably well (reference Appendix B, Figure 19), but it tends to chatter when drawing the "S" and deviates from the desired trajectory when navigating from the letter "C" to the letter "S". I believe this deviation is caused by an issue with my abbInvKine function, and it is selecting the theta value with the wrong sign for one of the joints. I investigated this issue and tried changing the sign on the theta value causing this issue (theta 4 at time 4.5 seconds), but it causes the next point to have the same reaction (theta 4 at time 4.75 seconds). Due to the project due date approaching, I chose to focus on the remaining parts of the project rather than try further to resolve this issue.

Reference Appendix B, Figure 13 for plots of my root-mean-squared errors in X, Y, and Z. As seen, my average y-error is 3.65e-5 and my average x-error is 0.3402. The x-error does not match the Manipulator Trajectory vs Desired Trajectory plot (reference Appendix B, Figure 19), so I do not why my average x-error is so high.

Reference Appendix B, Figure 16 for plots of my joint control torques resulting from my gain values. As you can see, the torque reaches up to a maximum magnitude of 445 Nm in joint 1 at the very beginning of the simulation. This is far too high and would either break the robot or not allow for many repeated robot uses, depending on the size of the robot arm & motors powering it.

Overall, this control scheme will follow the desired trajectory, but it is not as robust as I desired and I would not implement it on an industry level.

*Parts 2 & 3: Operational Space Inverse Dynamics Controller*

The manipulator tracks the desired trajectory very well (reference Appendix B, Figures 20 & 21), without any chatter. However, this is most likely because the gain values are so high.

Reference Appendix B, Figures 14 and 15 for plots of my root-mean-squared errors in X, Y, and Z. As seen, my average y-error is 0.0008 and my average x-error is 0.0043. This appears to match the Manipulator Trajectory vs Desired Trajectory plot (reference Appendix B, Figures 20 & 21). Also, both control schemes from parts 2 & 3 do not seem to have any difference from one another.

Reference Appendix B, Figures 16 & 17 for plots of my joint control torques resulting from my gain values. As you can see, the torque reaches up to a maximum magnitude of almost 65000 Nm in joint 4 at the very beginning of the simulation. This is far too high and would either break the robot or not allow for many repeated robot uses, depending on the size of the robot arm & motors powering it. After this though, you can see the torques all drop to within a reasonable value of under 50 Nm. However, towards the every now and then during the simulation they rise back to extremely high values. Also, both control schemes from parts 2 & 3 do not seem to have any difference from one another.

Overall, this control scheme used in parts 2 & 3 will follow the desired trajectory with high accuracy, but it is not as robust as I desired when it comes to required control torques and I would not implement it on an industry level.

Discussion

*Part 1: Configuration Space Inverse Dynamics Controller*

If I decrease my gains to $k_p = 250$ and $k_d = 50$, it decreases my maximum torque magnitude to 264 Nm while increasing my maximum trajectory error in y to about 0.0002 m. This isn't much, but the plot of the trajectory seems to have more error. Similar to where is deviates from the path between the letters "C" and "S", it also seems to deviate at the second 180° turn in the letter "S". Therefore, lowering my gains provide a lower robust system. If I increase my gains to $k_p = 3000$ and $k_d = 1000$, it increases my maximum torque magnitude to almost 2000 Nm while increasing my maximum trajectory error in y to about 0.0014 m. The plot of the trajectory seems to follow the desired trajectory in a smoother pattern, but does deviate from the path slightly between the letters "S" and "M". It appears increasing the gains resulting in an increase in the joint control torques.

Speeding up the trajectory completion time from 7.5 seconds to 5 seconds causes the robot not only to deviate from the desired trajectory (especially in the y & z), but it also increases my maximum torque magnitude to value to almost 1800 Nm. If I increase the speed to 3 seconds the manipulator just makes a squiggle for the letter "M" and the maximum torque magnitude to value to almost 4500 Nm.

Through trying out a variety of different trajectory times, 5.5 seconds appears to be the lowest time that my system can handle. Anything faster than this causes the joint torques to get to high in magnitude and deviates too far from the path. Reference Appendix B, Figure 22 for plot of this trajectory.

*Parts 2 & 3: Operational Space Inverse Dynamics Controller*

If I decrease my gains to $k_p = 150$ and $k_d = 60$, it increases my maximum torque magnitude to about 334500 Nm while increasing my average trajectory error in y to about 0.0031 m. This isn't much, but the plot of the trajectory seems to have more error. Therefore, lowering my gains provide a lower robust system. If I increase my gains to $k_p = 5000$ and $k_d = 600$, it decreases my maximum torque magnitude to about 50000 Nm while decreasing my maximum trajectory error in y to about 0.0007 m. The plot of the trajectory seems to follow the desired trajectory in a smoother pattern, but the torques are still too high. Therefore, increasing my gains provides a more robust control.

Speeding up the trajectory completion time from 7.5 seconds to 5 seconds causes the robot not only to deviate from the desired trajectory (especially in the y), but it also increases my maximum torque magnitude to value to almost 65000 Nm. If I increase the speed to 3 seconds the manipulator just makes a squiggle for letters "S" and "M" and the maximum torque magnitude to value to almost 65000 Nm.

Through trying out a variety of different trajectory times, 6.5 seconds appears to be the lowest time that my system can handle. Anything faster than this causes the joint torques to get to high in magnitude and deviates too far from the path. Reference Appendix B, Figures 23 & 24 for plot of this trajectory.

When comparing the controllers used in part 1 versus parts 2 & 3, this controller is definitely the more robust controller. Even though it will require higher joint torque control, is results in the manipulator following the desired trajectory with a higher accuracy.

## Conclusions

Both the Configuration Space Inverse Dynamics Controller (part 1) and the Operational Space Inverse Dynamics Controller (parts 2 & 3) appear to control the manipulator so that it will follow the desired trajectory. The Configuration Space Inverse Dynamics Controller has more positional error, but requires less joint torque control, which will result in less wear & tear to the robotic arm. Also, this controller was more intuitive to when designing. Even though the Operational Space Inverse Dynamics Controller is more complicated to design and requires more computation, it has a much smaller positional error and performs much better. Though, it does require much higher joint torque control. Overall, I would choose the Operational Space Inverse Dynamics Controller over the Configuration Space Inverse Dynamics Controller due to its better trajectory following performance.

If I were to modify this in the future, I would modify my choice of gains values by solving for the systems A & B matrices and using the “place” function in Matlab. I think this would reduce my issue with requiring high control torques for both controllers. I would also fix the issue with the Configuration Space Inverse Dynamics Controller where it deviated from its path when transitioning from the letter “C” to “S”. If I were to fix these issues, I think it would give me a much better idea of how changing each individual gain value affects the system.

Project Impression

I think this project was an excellent way of connecting all of the knowledge we gained from lectures and homework. I think it's great to see how each topic is implemented into a mechanical system and what the advantages & disadvantages are of using different techniques. I also this having us sufficiently document all code will provide a substantial benefit in the future if I come back to this project in the future to refresh myself on these techniques.

I think the most frustrating part is learning the differences between how Simulink accepts code and how MatLab accepts code, such as everything needing to be pre-allocated or how troubleshooting isn't as user friendly (from my experience at least). It would be nice to have a better understanding of these differences ahead of time due to the rigorousness of the project as is so we could avoid wasting time troubleshooting. However, I guess this is all part of the experience in learning a new system.

Overall, I think this is a great project. I recommend you keep doing this it for future classes.

Appendix A: Simulink Models

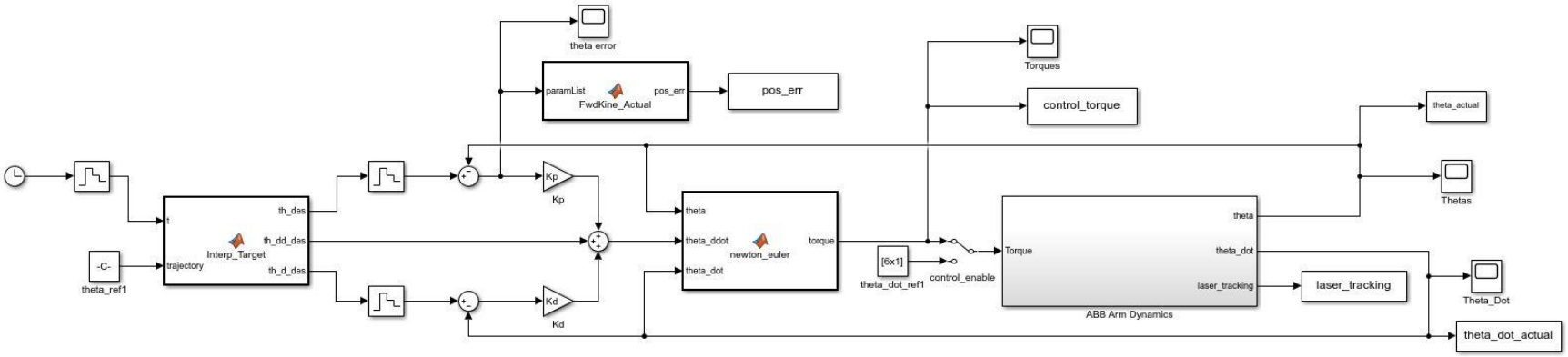


**Figure 1: Part 1 - Configuration Space Inverse Dynamics Controller**

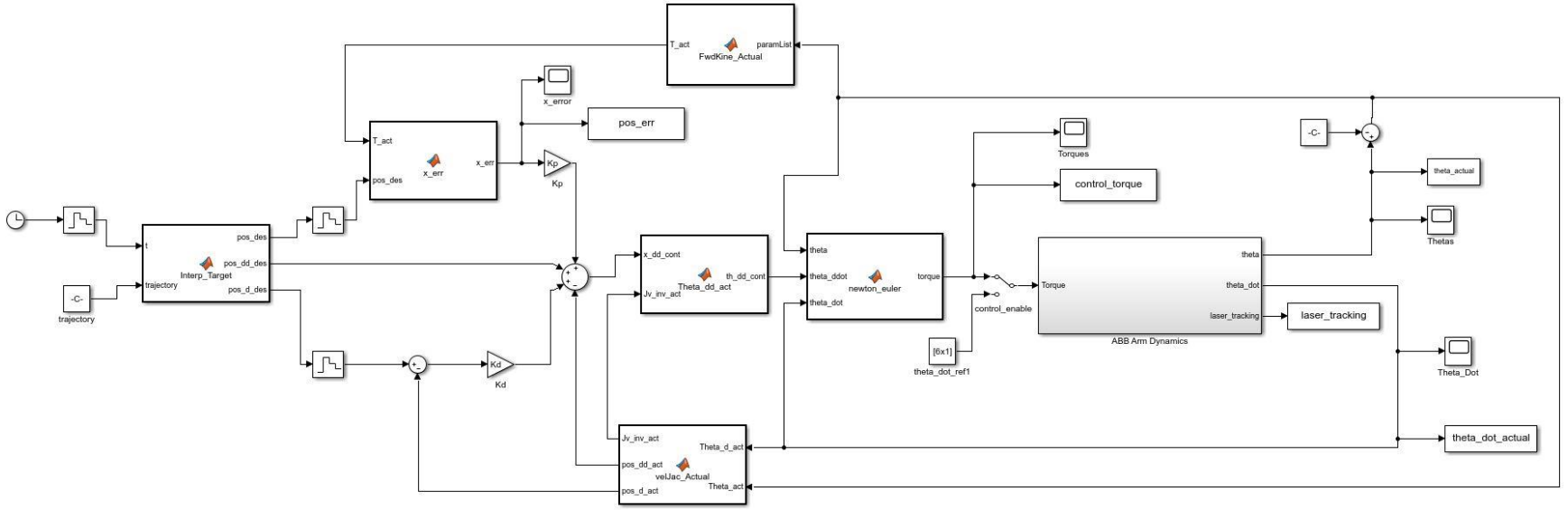


**Figure 2: Part 2 - Operational Space Inverse Dynamics Controller**

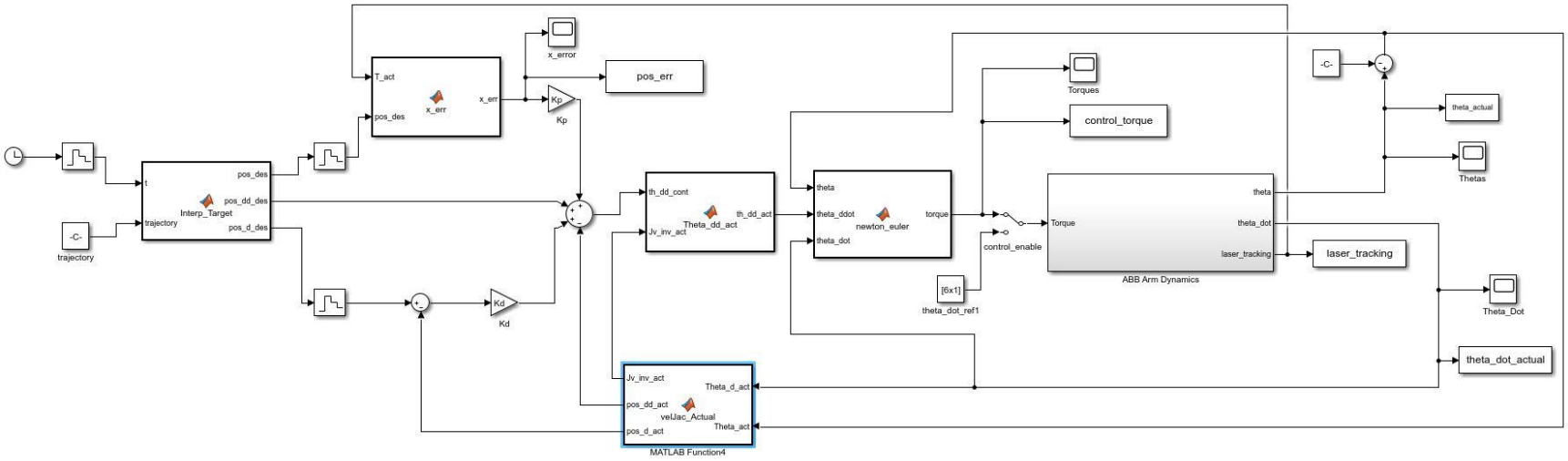


**Figure 3: Part 3 - Operational Space Inverse Dynamics Controller with laser_tracking**

Appendix B: Plots

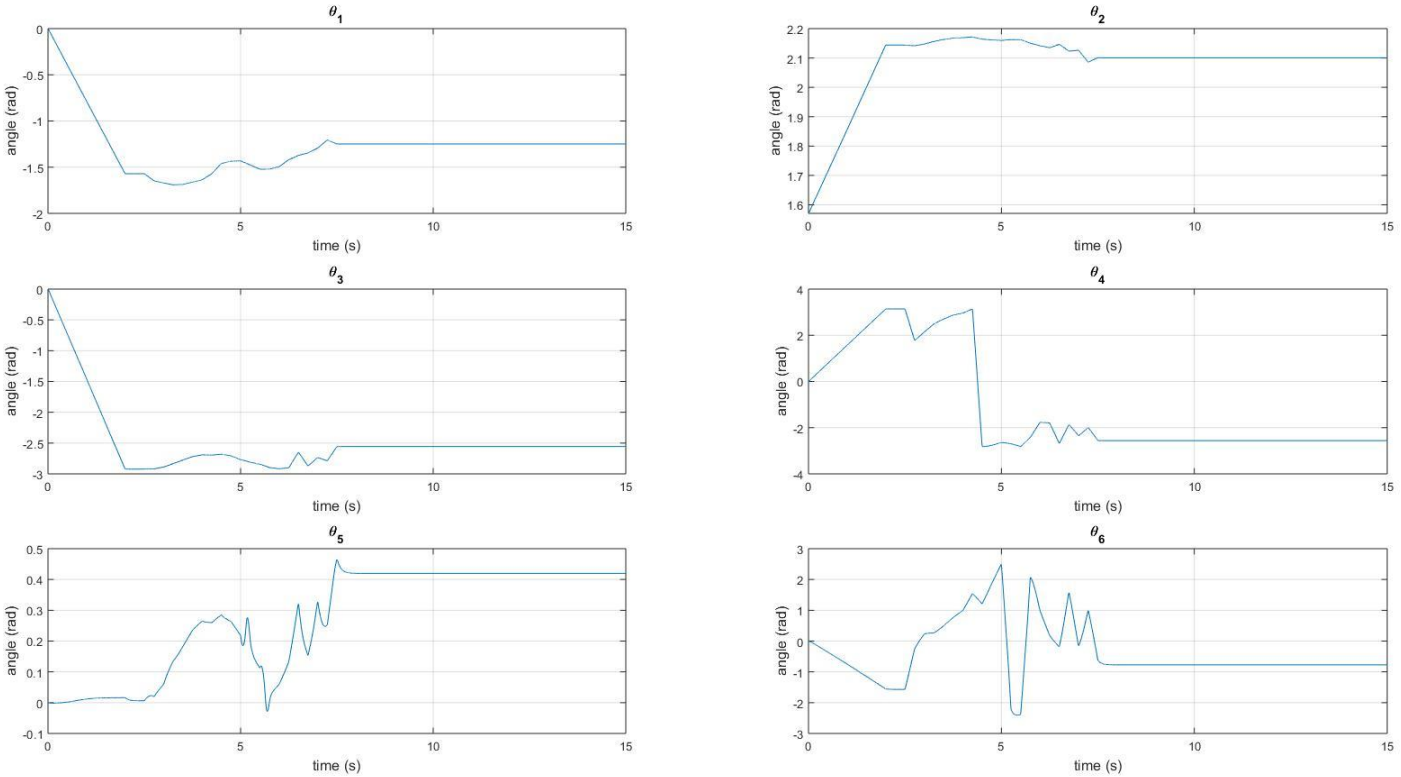


**Figure 4: Part 1 – Thetas (actual) vs Time**

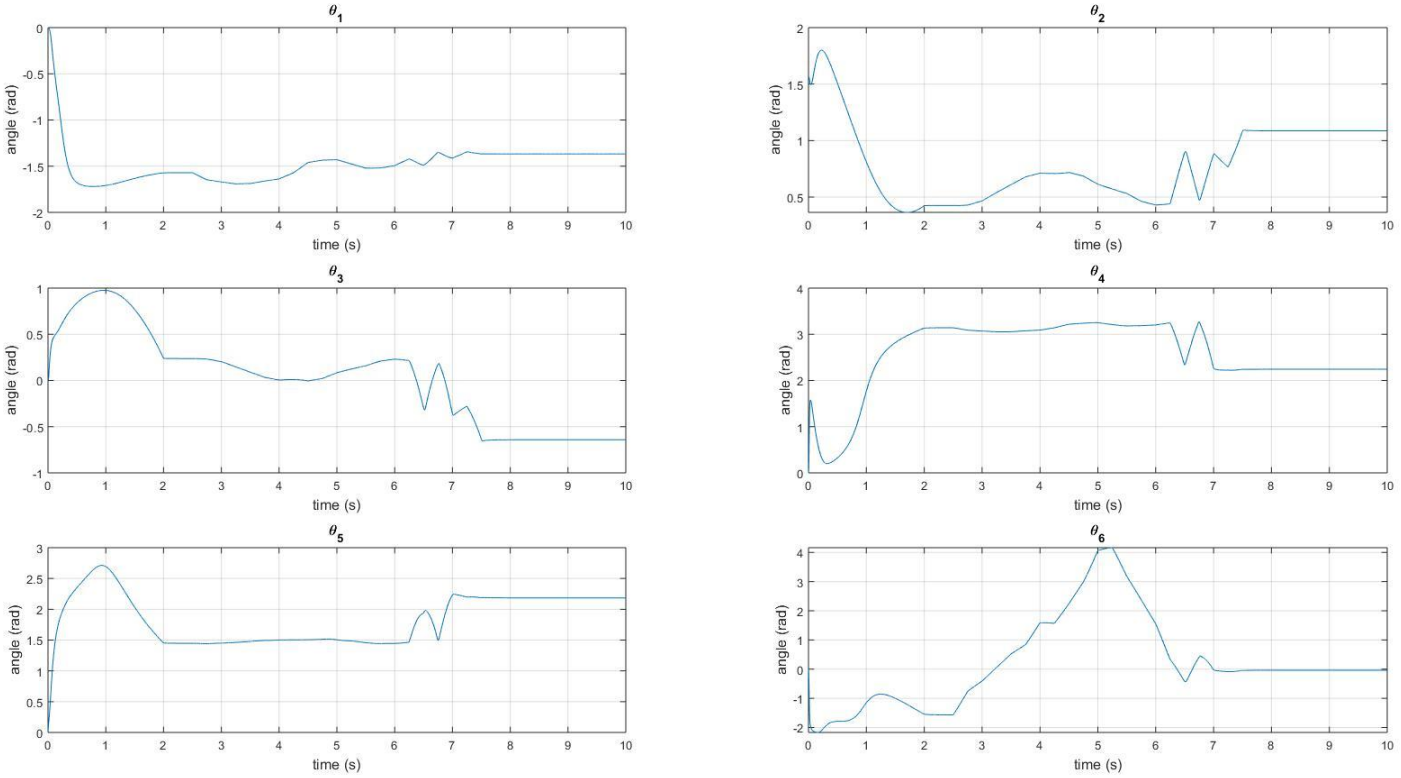


**Figure 5: Part 2 – Thetas (actual) vs Time**

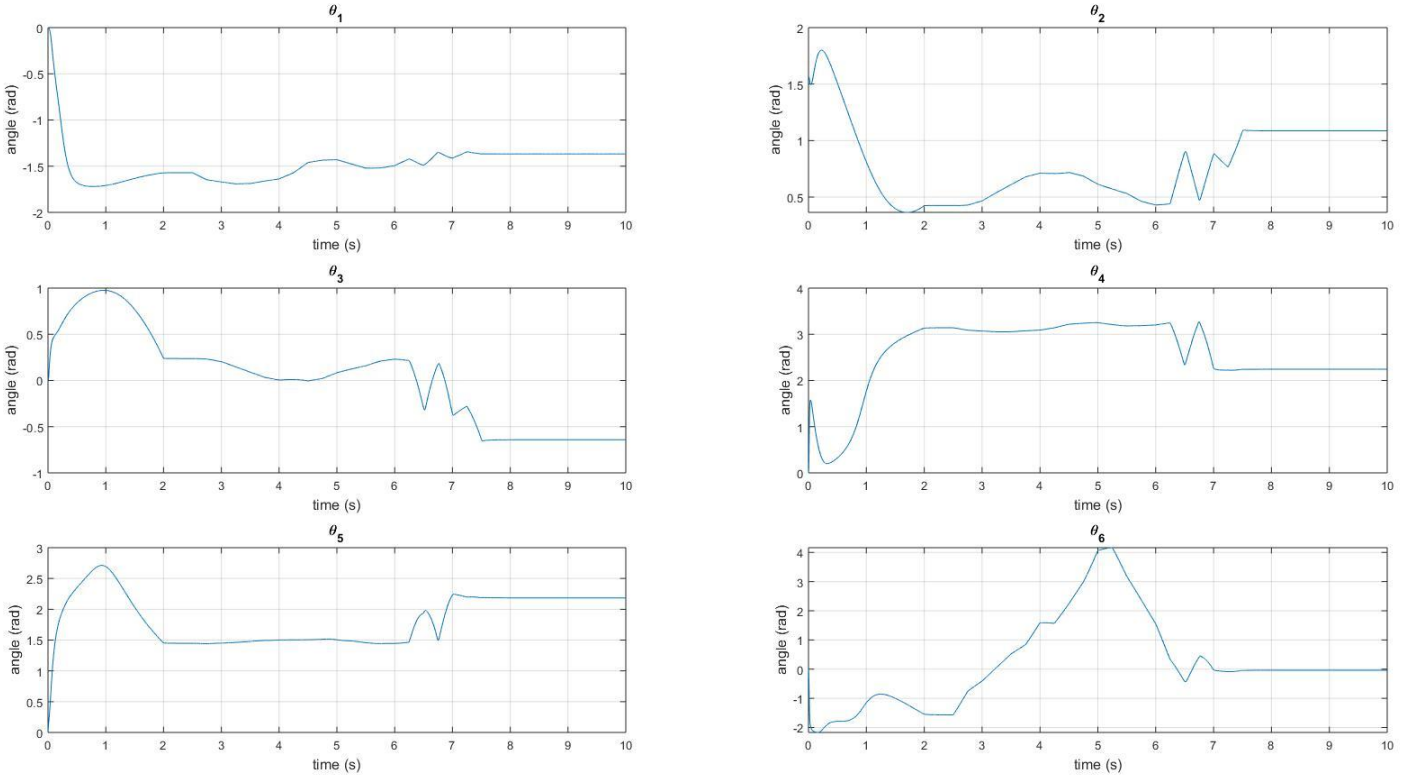


**Figure 6: Part 3 – Thetas (actual) vs Time**

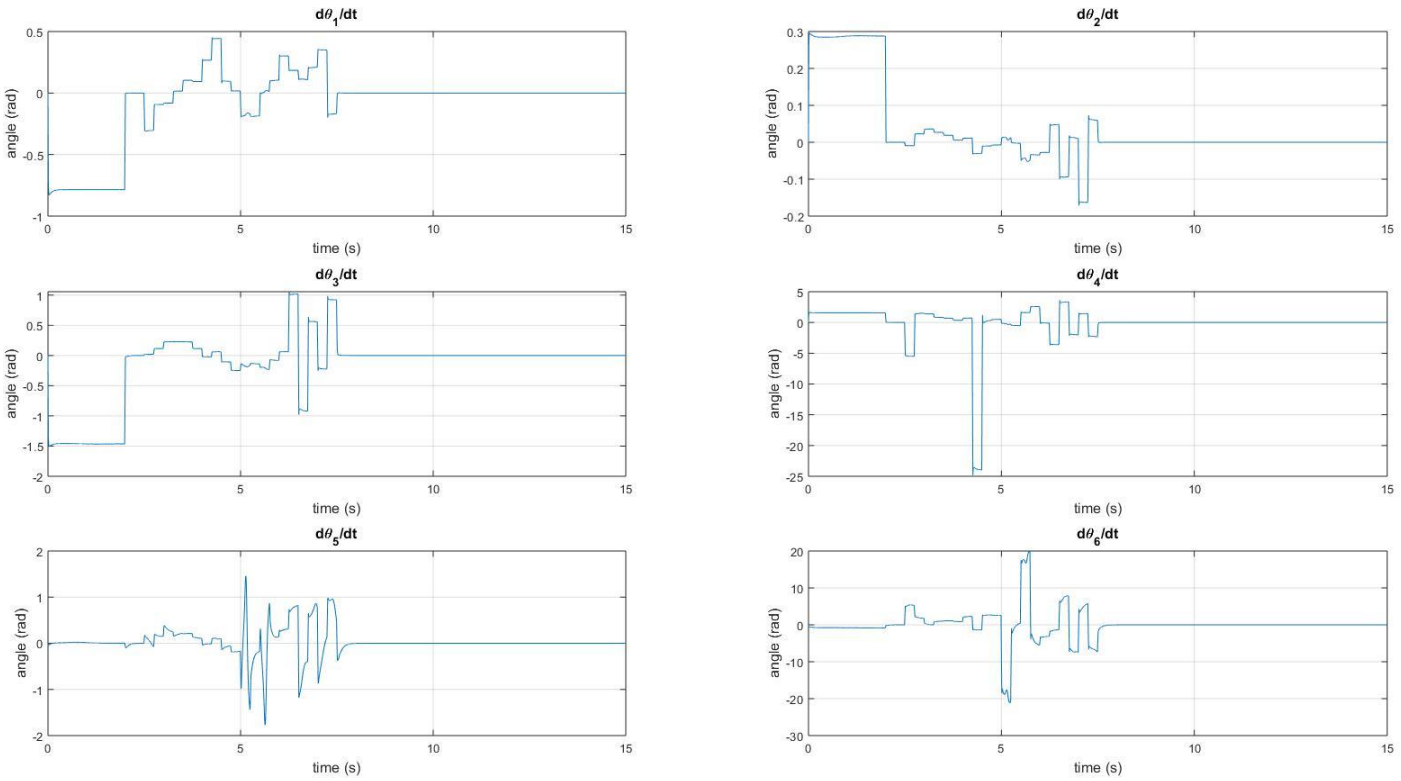


**Figure 7: Part 1 – Theta_dots (actual) vs Time**

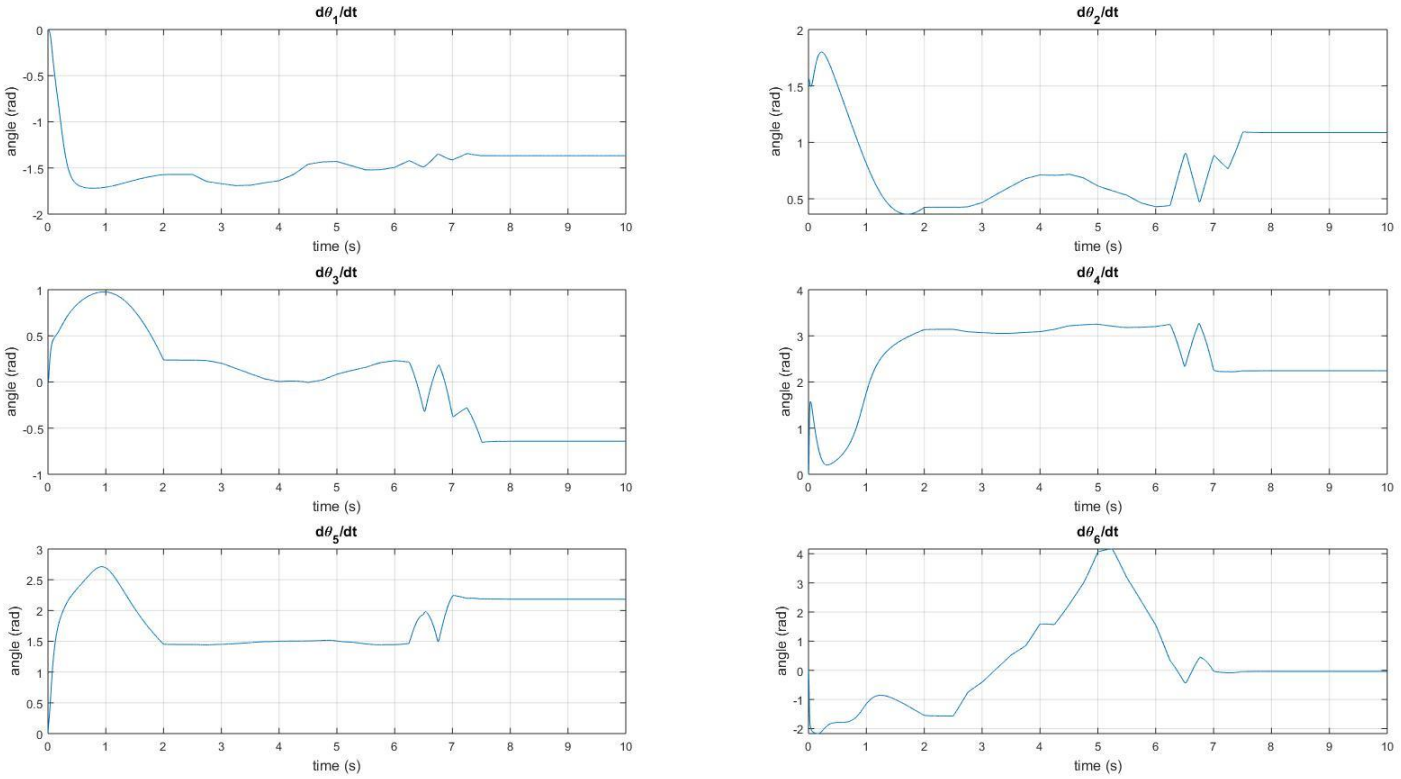


**Figure 8: Part 2 – Theta_dots (actual) vs Time**

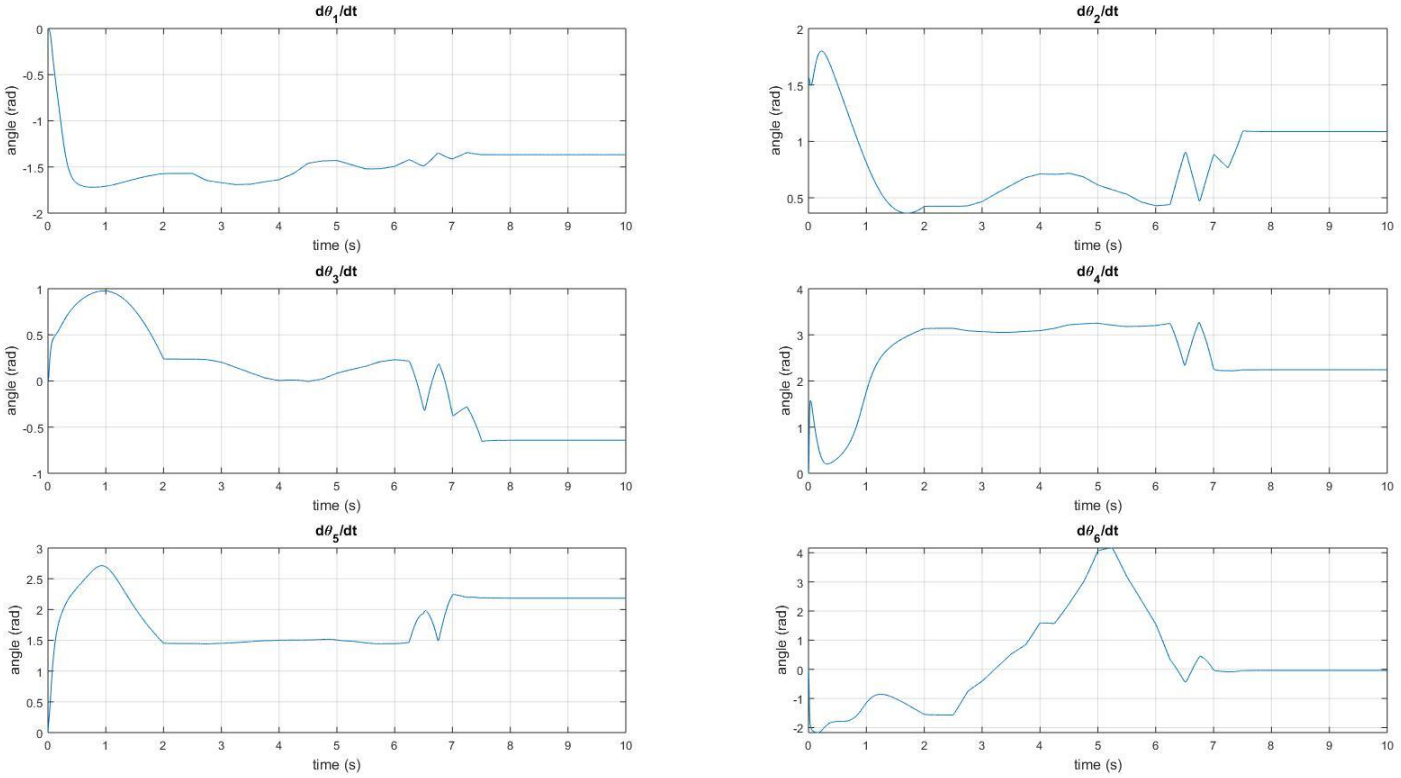


**Figure 9: Part 3 – Theta_dots (actual) vs Time**

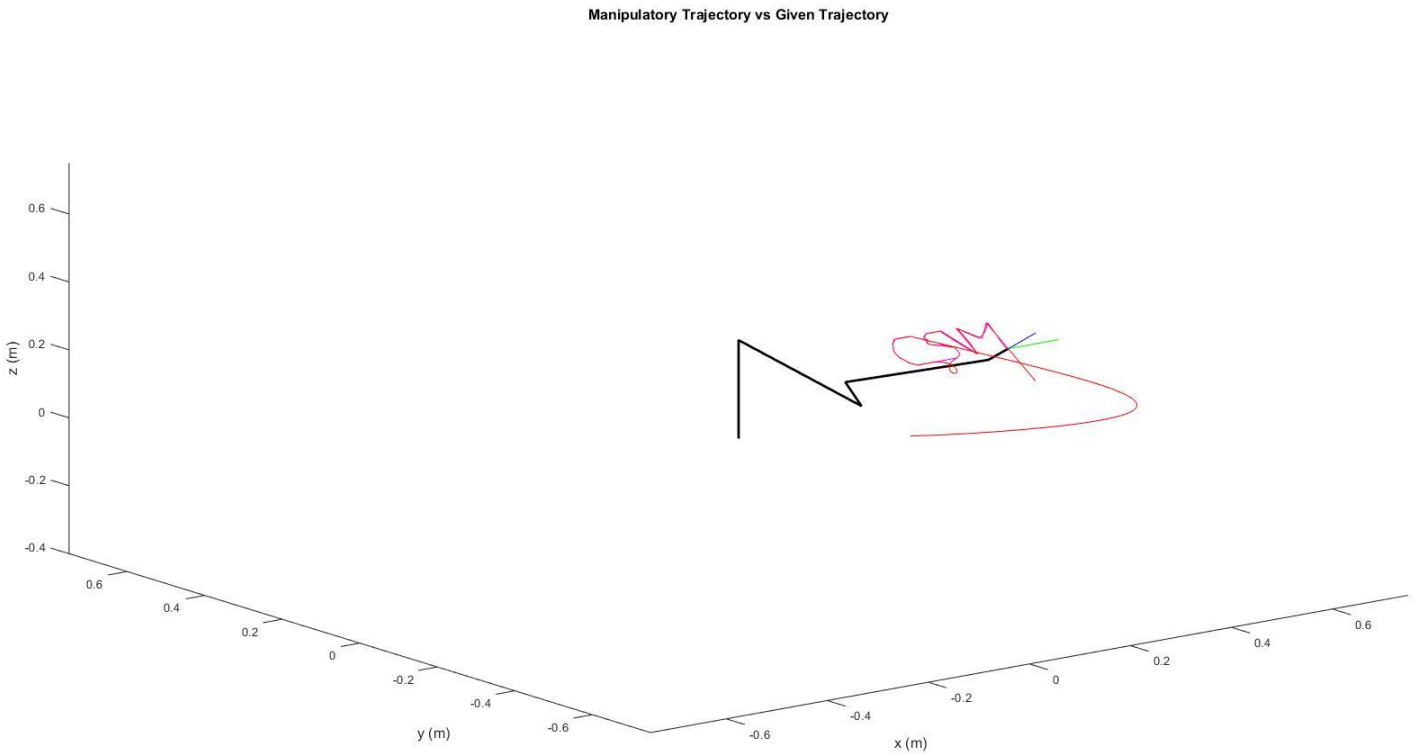


**Figure 10: Part 1 – Manipulator Trajectory (with arm) vs Desired Trajectory**

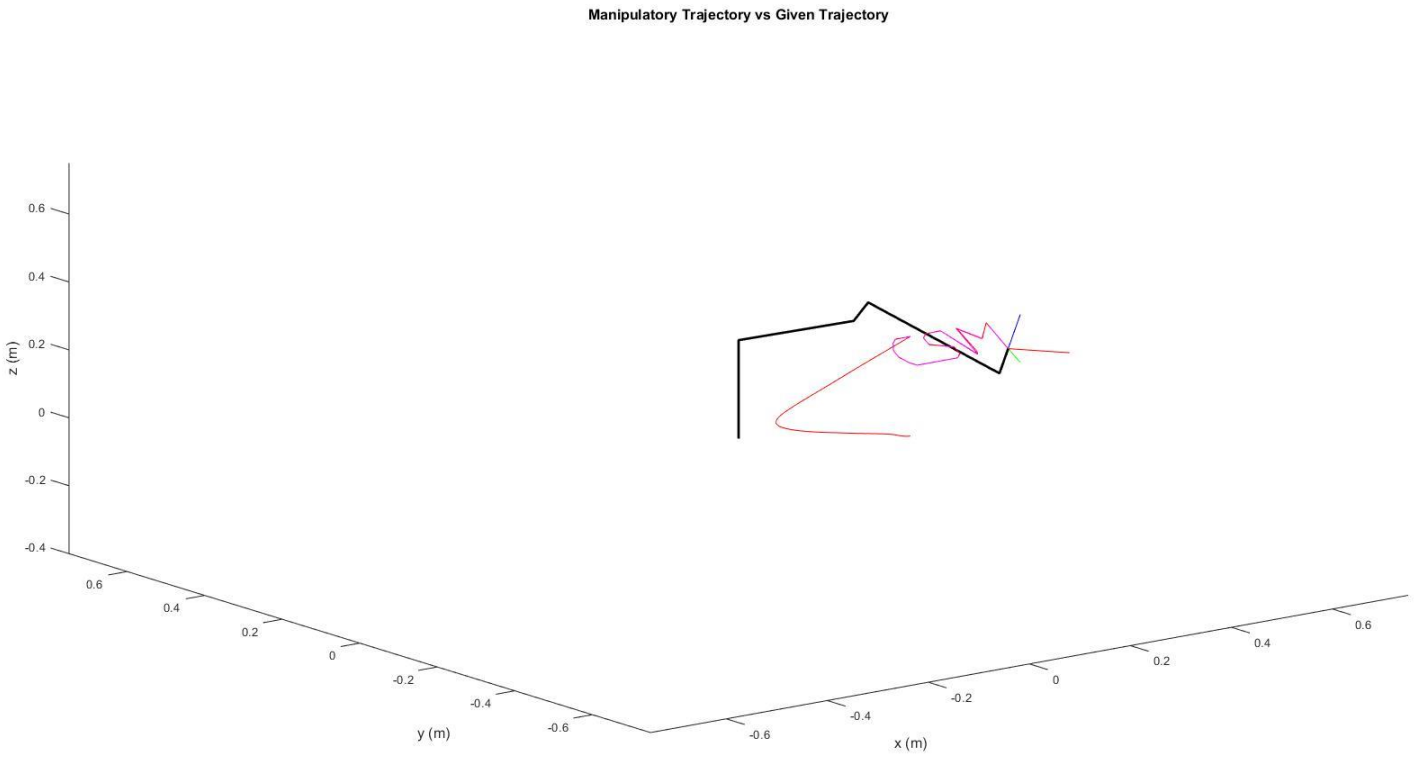


**Figure 11: Part 2 – Manipulator Trajectory (with arm) vs Desired Trajectory**

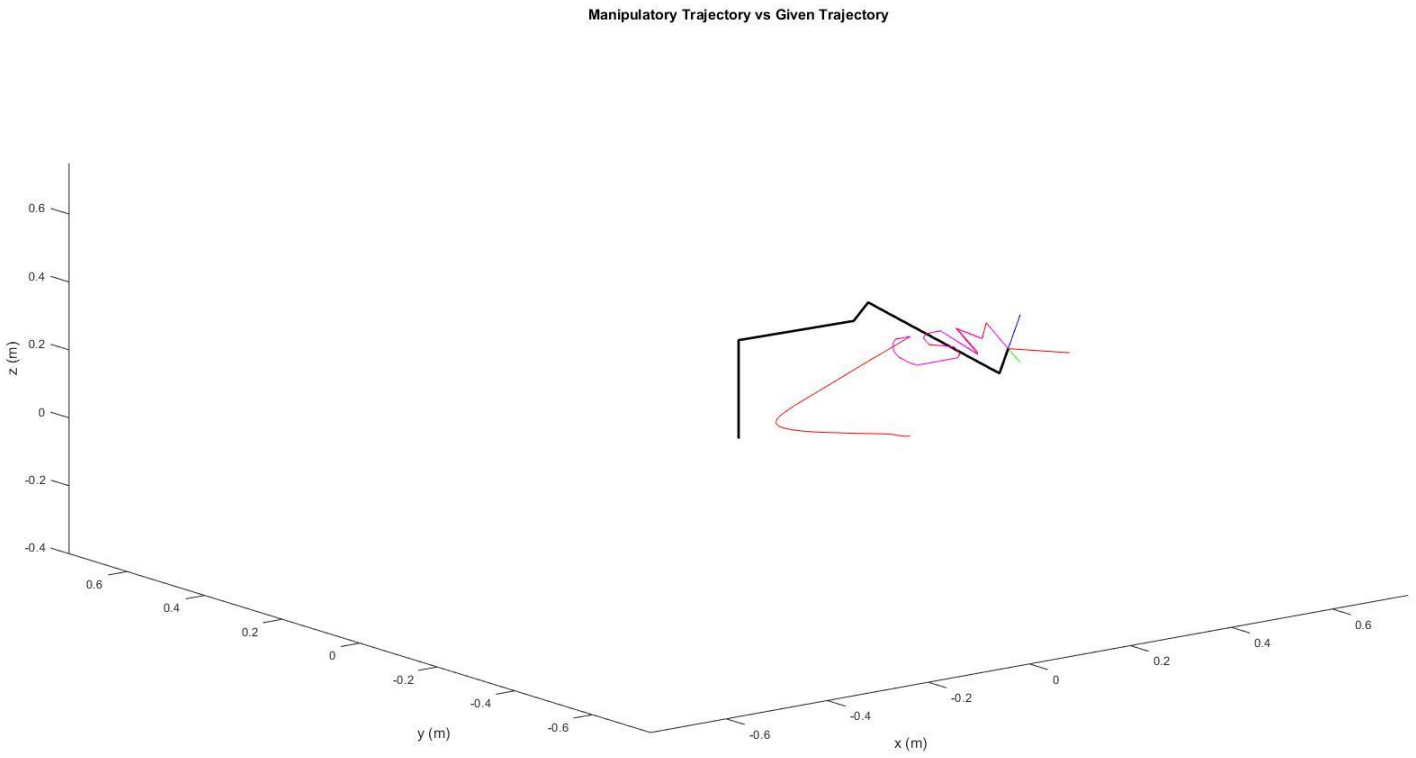


**Figure 12: Part 3 – Manipulator Trajectory (with arm) vs Desired Trajectory**

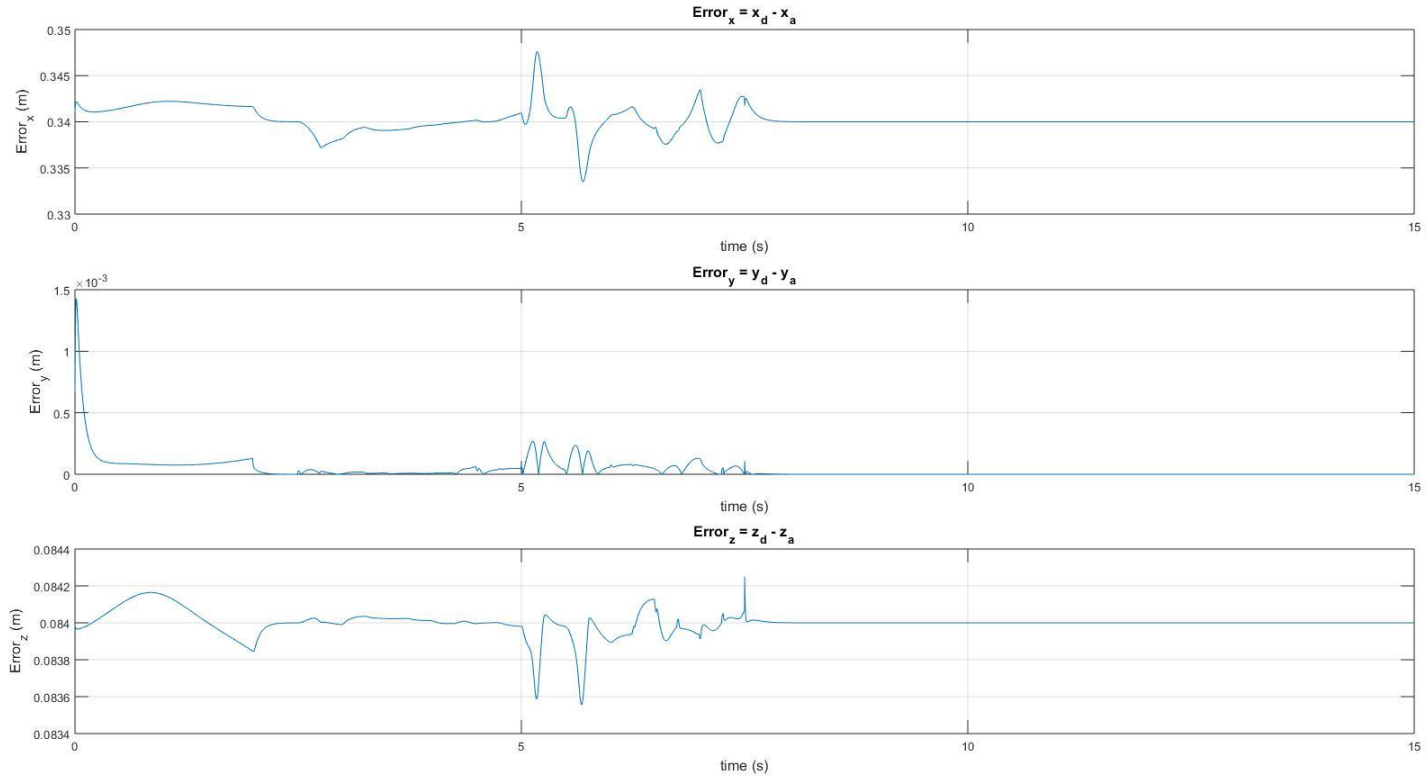


**Figure 13: Part 1 – Position (xyz) Error vs Time**

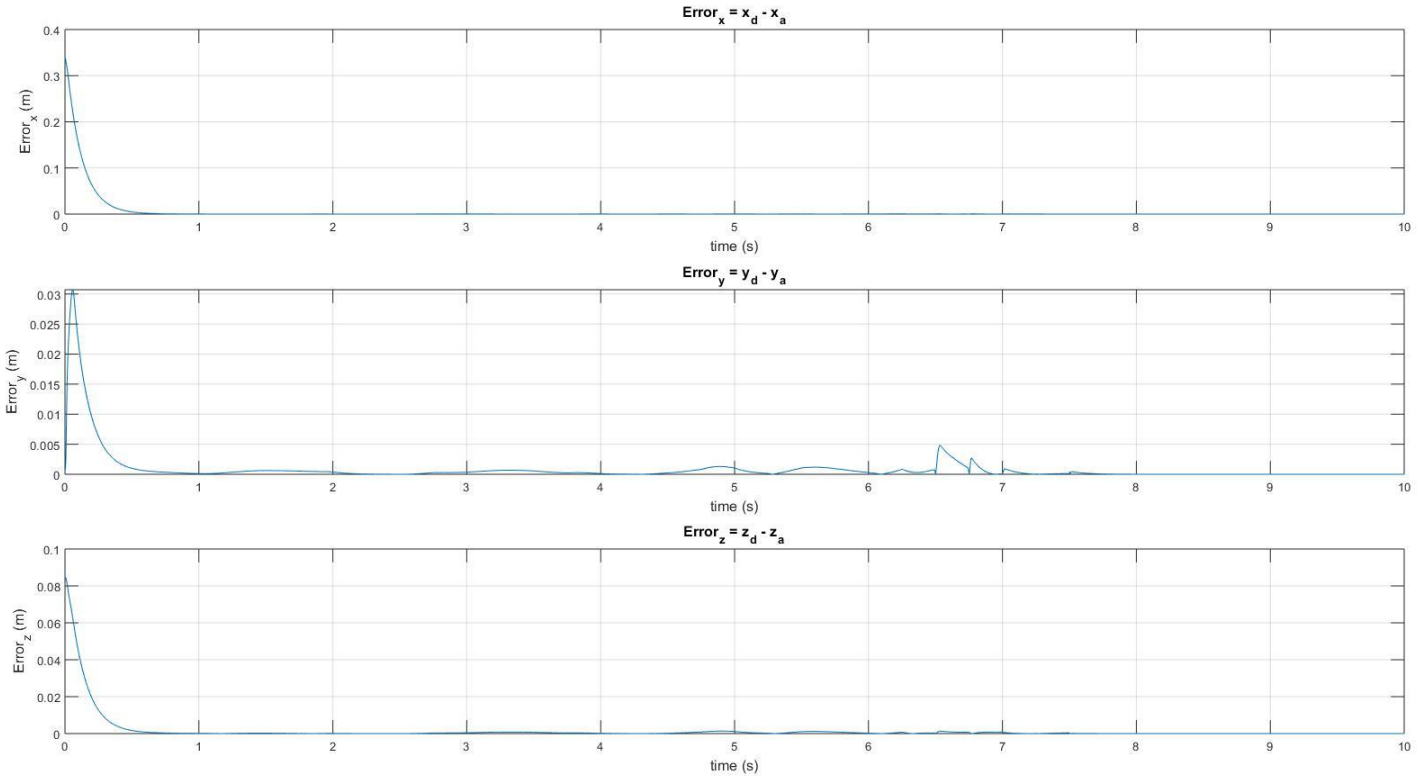


**Figure 14: Part 2 – Position (xyz) Error vs Time**

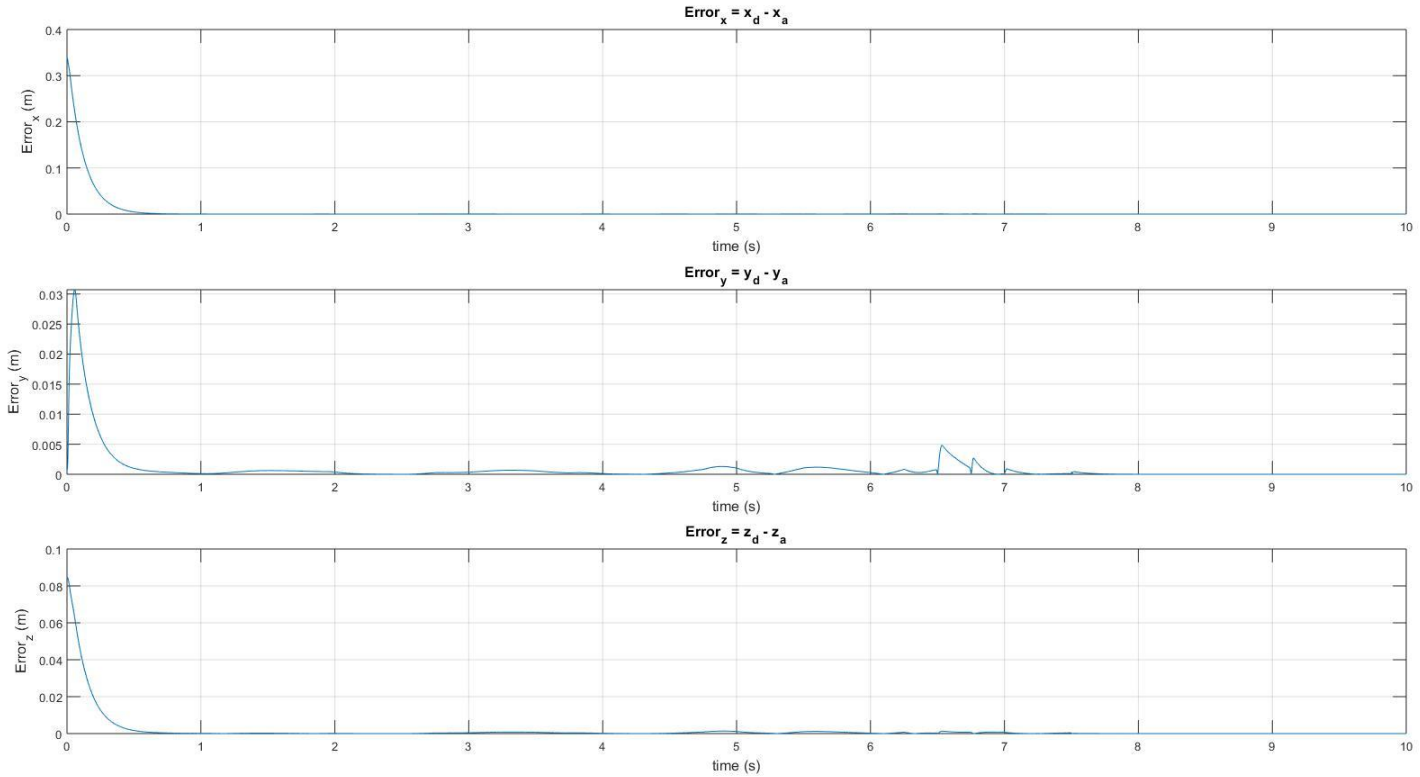


**Figure 15: Part 3 – Position (xyz) Error vs Time**

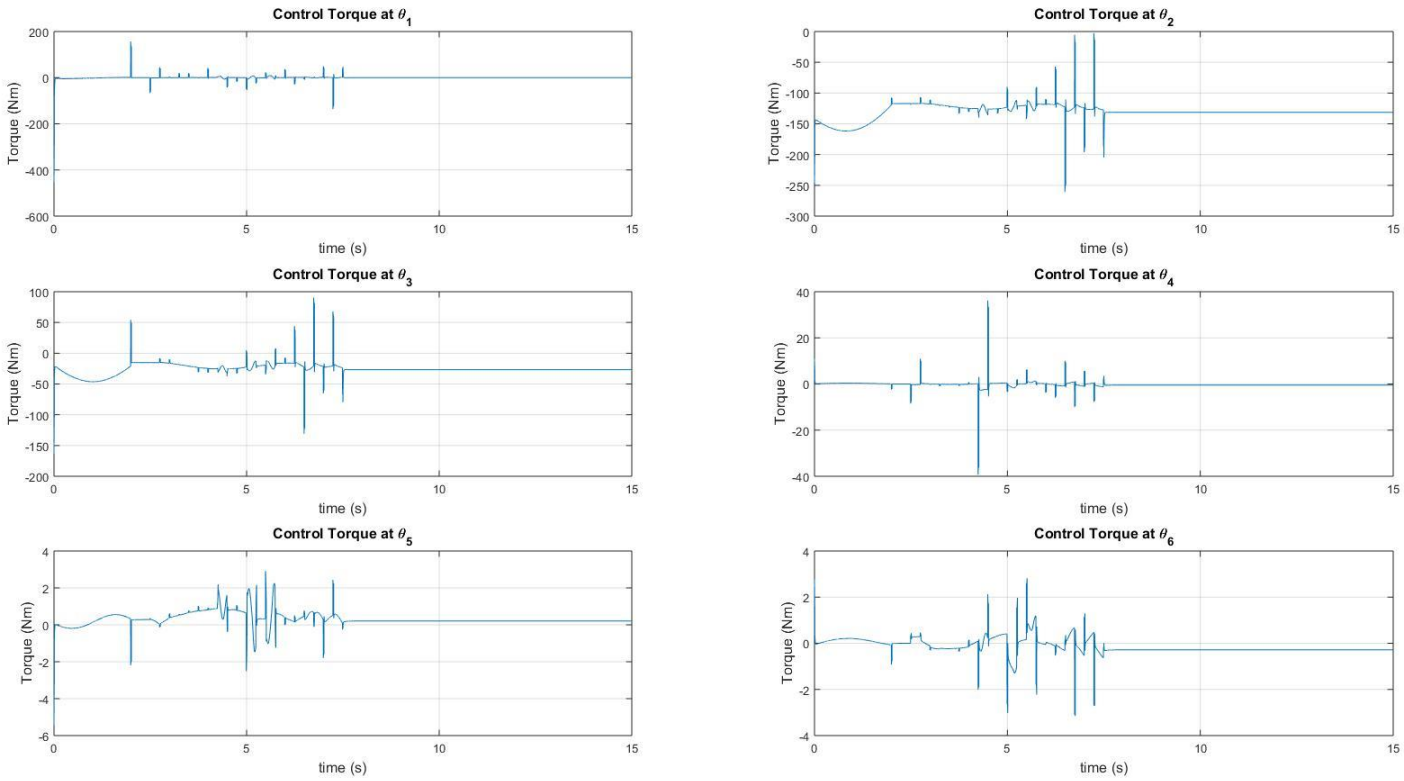


**Figure 16: Part 1 – Control Torque vs Time**

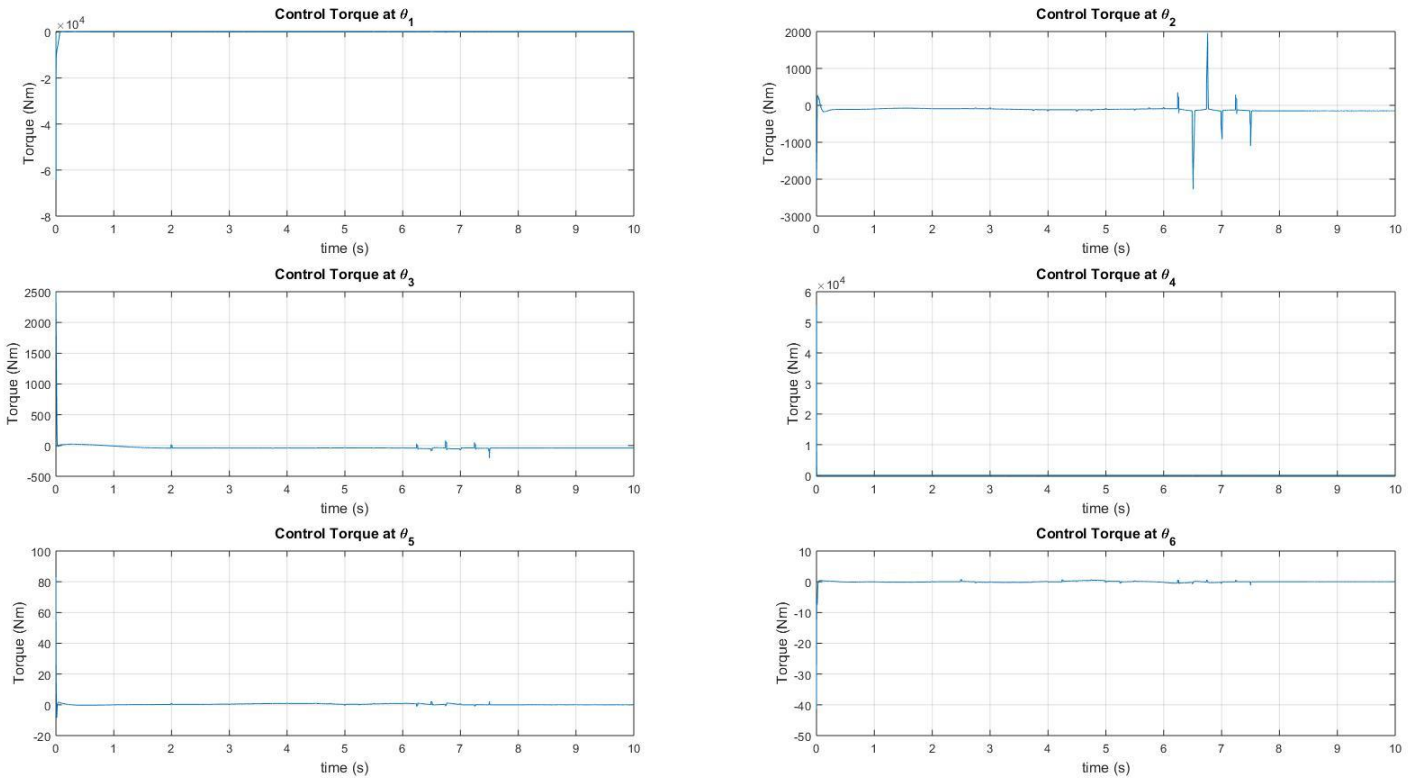


**Figure 17: Part 2 – Control Torque vs Time**

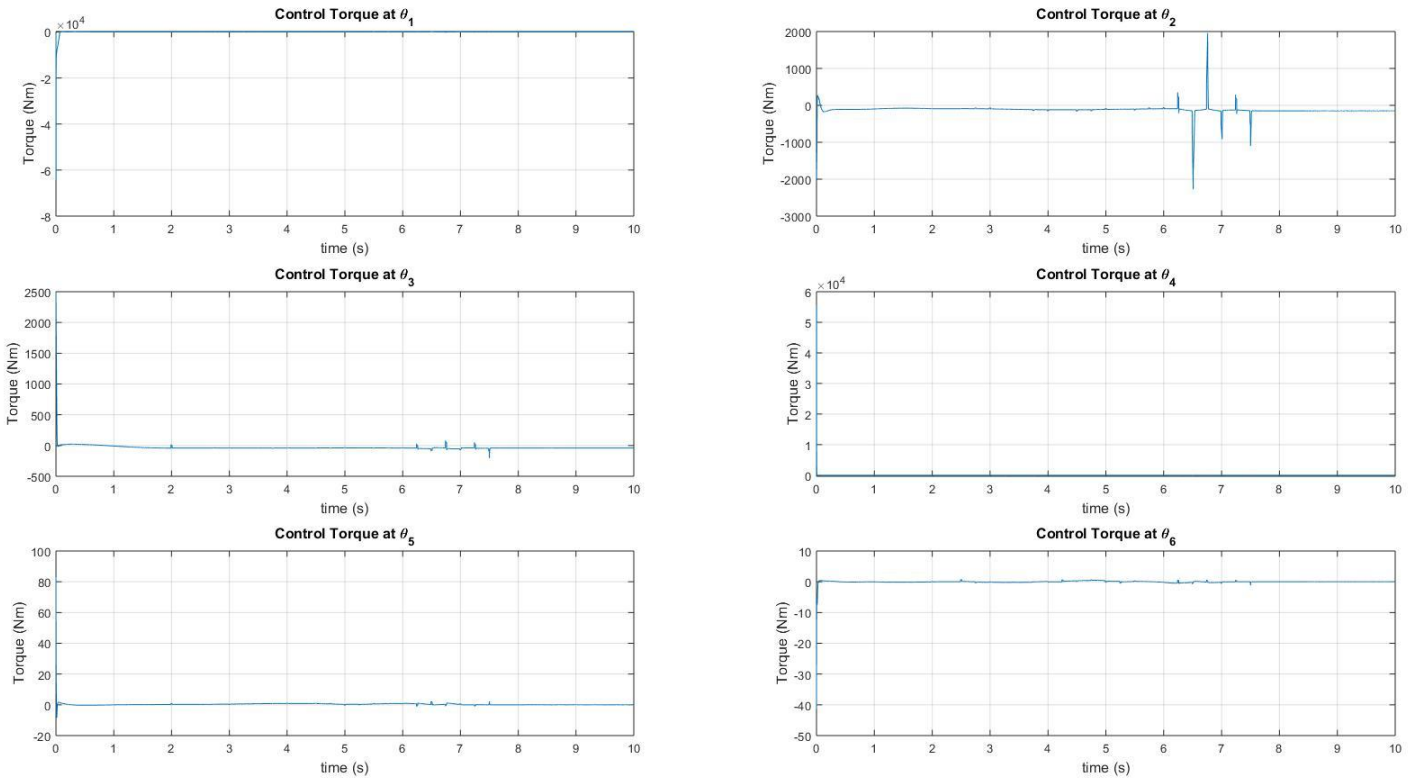


**Figure 18: Part 3 – Control Torque vs Time**

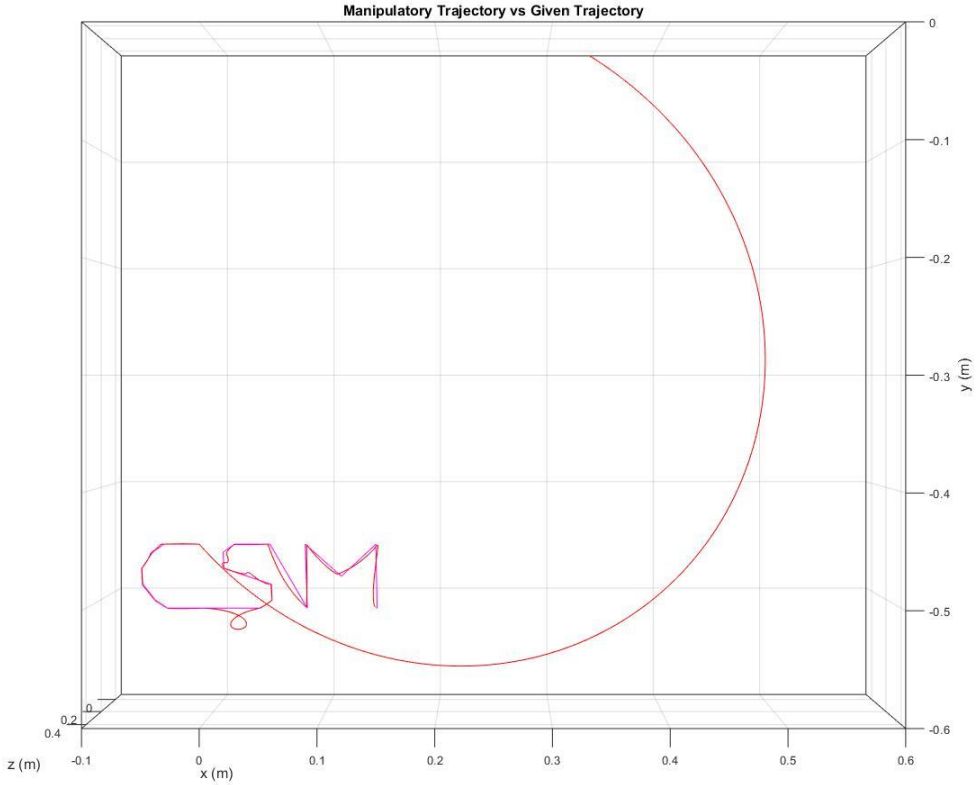


**Figure 19: Part 1 – Manipulator Trajectory (without arm) vs Desired Trajectory**


**Figure 20: Part 2 – Manipulator Trajectory (without arm) vs Desired Trajectory**


**Figure 21: Part 3 – Manipulator Trajectory (without arm) vs Desired Trajectory**

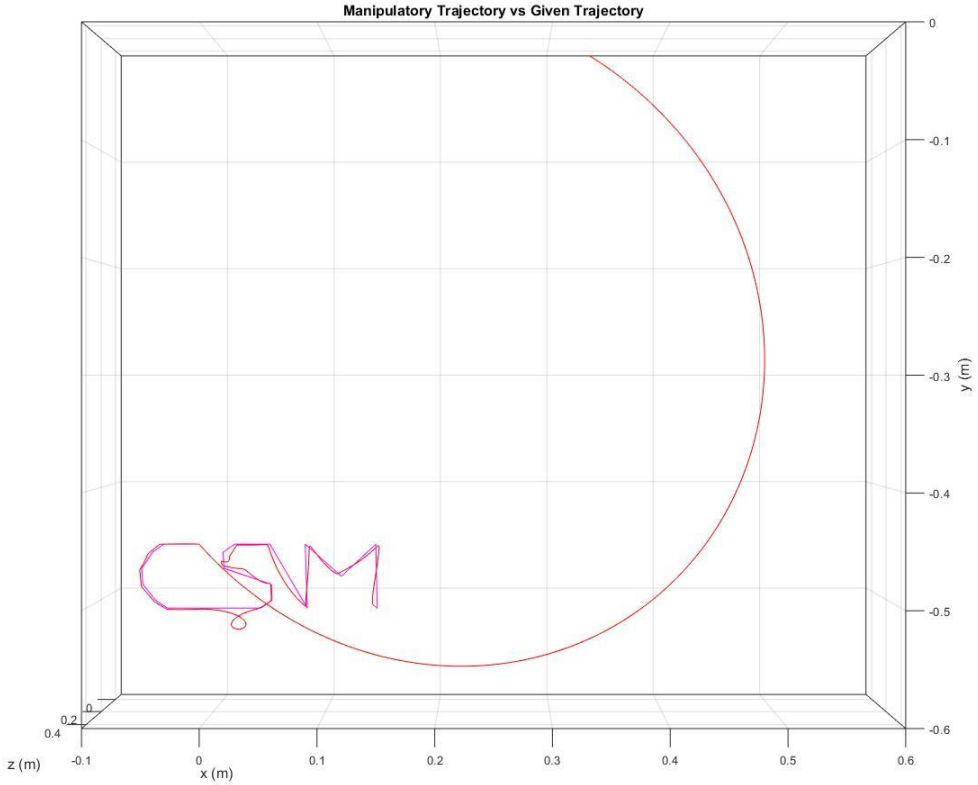


**Figure 22: Part 1 – Manipulator Trajectory (without arm) vs Desired Trajectory with t_end = 5.5 seconds**


**Figure 23: Part 2 – Manipulator Trajectory (without arm) vs Desired Trajectory with t_end = 6.5 seconds**


**Figure 24: Part 3 – Manipulator Trajectory (without arm) vs Desired Trajectory with t_end = 6.5 seconds**

Appendix C: Variable Definitions

$\theta_d \rightarrow$ Desired joint angle

$\dot{\theta}_d \rightarrow$ Desired joint angular velocity

$\ddot{\theta}_d \rightarrow$ Desired joint angular acceleration

$x_d \rightarrow$ Desired joint position

$\dot{x}_d \rightarrow$ Desired joint linear velocity

$\ddot{x}_d \rightarrow$ Desired joint linear acceleration

$\theta_a \rightarrow$ Actual joint angle

$\dot{\theta}_a \rightarrow$ Actual joint angular velocity

$x_a \rightarrow$ Actual joint position

$\dot{x}_a \rightarrow$ Actual joint linear velocity

$k_p \rightarrow$ Proportional Gain

$k_d \rightarrow$ Derivative Gain

$\tau_{cont} \rightarrow$ Control Torque

$\ddot{\theta}_{cont} \rightarrow$ Control joint angular acceleration

$\ddot{x}_{cont} \rightarrow$ Control joint linear acceleration

${J_v}^{-1} \rightarrow$ Inverse of velocity Jacobian $J_v$

$\dot{J}_v \rightarrow$ Derivative of velocity Jacobian $J_v$

$M(\theta) \rightarrow$ Mass Matrix

$G(\theta_a) \rightarrow$ Gravity Torques

$C(\theta_a, \dot{\theta}_a) \rightarrow$ Coriolis Torques

$F \rightarrow$ Friction Torques

$Q_d \rightarrow$ Desired quaternion where $Q_d = [q_{0,d} \quad q_{1,d} \quad q_{2,d} \quad q_{3,d}]$

$T_{act} \rightarrow$ Transformation matrix generated from $\theta_a$ & link parameters (part 2) or created from laser_tracking in Simulink model (part 3)