Harshul Singh
Lawrence Onyango
Anoushka Sengupta

Robotics Capstone Report

DH Parameter Extraction:

During the DH parameter extraction process, we commenced by measuring the individual link lengths between joints and subsequently determined the offsets between each frame. These measurements were then compared against the lecture notes and used to derive the corresponding DH parameters for the robotic system. It's important to note that throughout this process, we assumed that all joints were positioned out of the page, adhering to the conventions commonly used in DH parameterization. The following table shows our DH parameters for the reference configuration:

Link Lengths	Link Twist	Link Offset	Joint Angle
0	$\pi/2$	56.05	0
400	0	94	0
334	$-\pi/2$	3	0
0	$\pi/2$	39	0
0	0	108.05	0

Forward Kinematics:

a_i Link length

α_i Link twist

d_i Link offset

θ_i Joint angle

To construct the homogeneous transformation matrix between frames using the 4 D-H parameters, we utilize the following convention:

$$H_i^{i-1} = Rot_{z,\theta_i} Trans_{z,d_i} Trans_{x,a_i} Rot_{x,\alpha_i}$$

$$= \begin{bmatrix} c_{\theta_i} & -s_{\theta_i} & 0 & 0 \\ s_{\theta_i} & c_{\theta_i} & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & a_i \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & c_{\alpha_i} & -s_{\alpha_i} & 0 \\ 0 & s_{\alpha_i} & c_{\alpha_i} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} c_{\theta_i} & -s_{\theta_i} c_{\alpha_i} & s_{\theta_i} s_{\alpha_i} & a_i c_{\theta_i} \\ s_{\theta_i} & c_{\theta_i} c_{\alpha_i} & -c_{\theta_i} s_{\alpha_i} & a_i s_{\theta_i} \\ 0 & s_{\alpha_i} & c_{\alpha_i} & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

With our dh parameters extracted along with the mapping of DH parameters to a homogenous transformation matrix we can easily generate the forward kinematics of our robot.

```
function frames = forward_kinematics(robot, thetas)

          ...
           frames = zeros(4,4,robot.dof);
           n = robot.dof;

frames(:,:,1)=Homog(thetas(1),robot.dh_parameters(1,1),robot.dh_parameters(1,2),robot.
dh_parameters(1,3));
           for i=2:n

frames(:,:,i)=frames(:,:,i-1)*Homog(thetas(i)+robot.dh_parameters(i,4),robot.dh_parame
ters(i,1),robot.dh_parameters(i,2),robot.dh_parameters(i,3));
           end
```

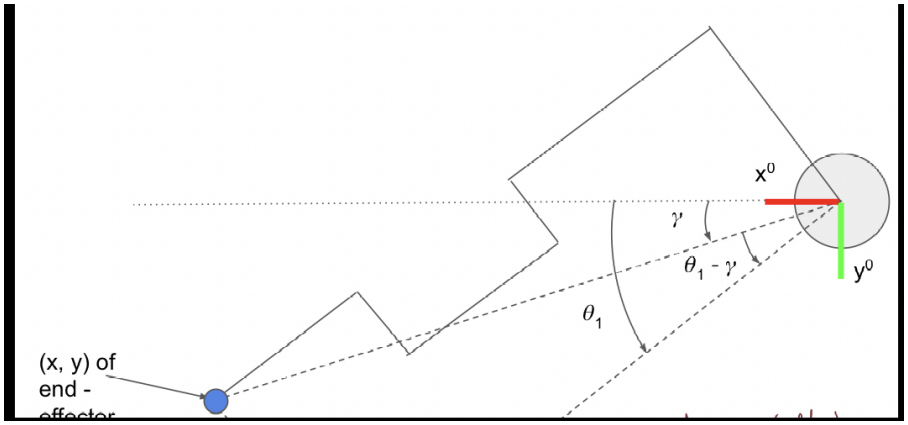
```
        % The transform from the base of link 'i' to the base frame (H^0_i)
        % is given by the 4x4 matrix frames(:,:,i).

        % The transform from the end effector to the base frame (H^0_i) is
        % given by the 4x4 matrix frames(:,:,end).

        % -------------- BEGIN STUDENT SECTION ---------------------------------
        function H=Homog(theta,a,alpha,d)
            H=[cos(theta) -sin(theta)*cos(alpha) sin(theta)*sin(alpha) a*cos(theta)
               sin(theta) cos(theta)*cos(alpha) -cos(theta)*sin(alpha) a*sin(theta)
               0 sin(alpha) cos(alpha) d
               0 0 0 1];
        end
        % -------------- END STUDENT SECTION -----------------------------------
    end
```

• In details, write down analytical inverse kinematics for capstone arm

Following is our analytical inverse kinematics for the capstone arm. We used the DH parameter after fine-tuning and measurement of the capstone arm.

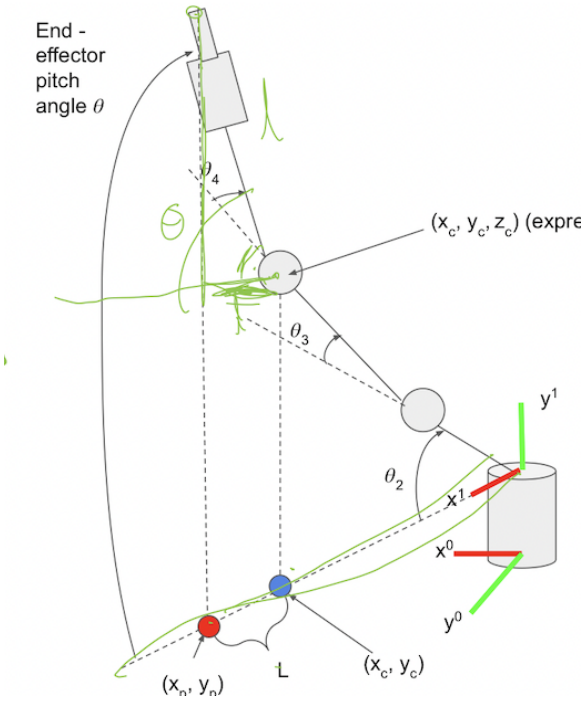


To begin with we considered theta(5) to be a constant value that was given to us, as theta(5) controlled the rotation of the suction cup at the end of the arm.

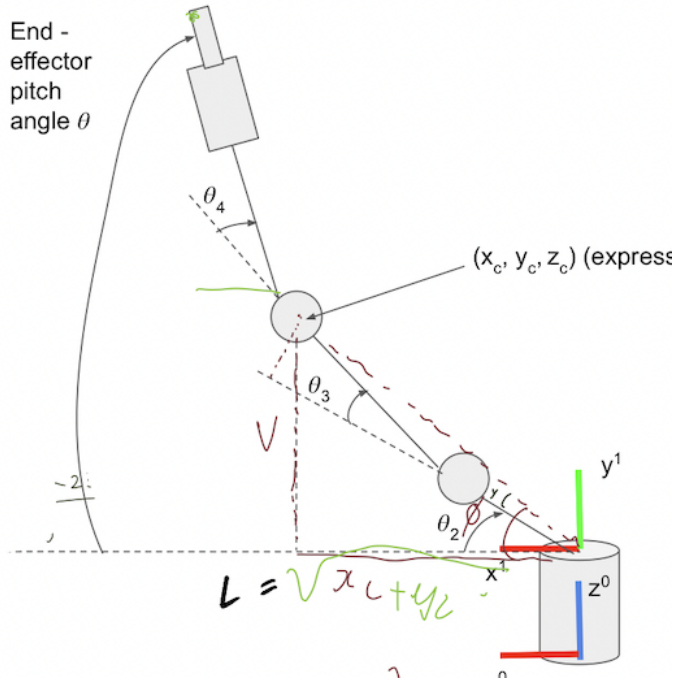
For the equations for theta(4) to theta(2), we considered it to be akin to a RRR inverse kinematics problem.

Following the pattern for an RRR IK, we derived the value for theta(4). This was later tweaked to account for the specific axes on the robotic arm.

Theta(4) = theta + theta(3) - theta(2).



Thereafter we found our values for (x_p, y_p) and (x_c, y_c) as marked in the diagram. Thereafter, we calculated our values for V and L.



$$\mathsf{x}_p = x_e + DS_1$$
$$y_p = y_e - DC_1$$

$$\mathsf{x}_c = x_p - d_5 C_1$$
$$y_c = y_p - d_5 S_1$$
$$z_c = z_e - d_5 S_\theta$$

Our projection onto the xy plane was: $\mathsf{dist} = \mathsf{sqrtx}_c^2 + y_c^2$

Using the Cosine rule
$$\theta_3 = (v^2 + dist^2 - l_3^2 - l_2^2)/(2 l_3 l_2)$$

$$\theta_2 = atan2(v, dist) - atan2(l_3 S_3/(l_2 + l_3 C_3))$$
$$\theta_1 = \theta_1 - \gamma + \gamma$$
$$\theta_1 = atan2(y_e, x_e) + \arcsin\left(D/\sqrt{(x_e^2 + y_e^2)}\right)$$

While we did not use our IK, we do believe it to be reasonably correct.

• Describe your strategy for robot control using trajectories. What optimizations did you make?

We chose to control with spline trajectories as this smooth interpolation minimizes jerk and leads to a smooth movement which is what we want when trying to repeat a task with high precision.

To optimize the trajectories further we introduced midpoints and approach angles similar to how they were sketched out in pick_place_sample to ensure that we wouldn't make a straight line connection between points. We also introduced an offset in theta_2 to create an approach angle that would result in the place action near the end of the trajectory slowing down to avoid excessive force being applied. We also experimented with the duration of time given between each trajectory to find a good tradeoff of speed and accuracy in terms of joint deviations at high speeds and gravity's effect being more noticeable at low speeds.

We tuned the robot gains to ensure that there were no vibrations happening with motors having their currents/torques saturated. We also stopped commanding velocities and only commanded positions. Qualitatively it seemed that allowing the onboard motor drivers to determine the velocities led to a better trajectory. I think part of why this was the case was because we did not implement gravity compensation (the robot crashed when trying to command torques) which potentially allowed the encoder error experienced by the motors to command a better trajectory.

• Describe challenges you faced and how you would approach this problem differently if you were to do this project again. What worked and what didn't work? What are some improvements you could make? How robust is your solution?

We faced a litany of challenges mostly confined to inconsistencies related to the robot hardware and one core technical challenge.

Asymmetry of the robot setups → this prevented code interoperability of our code between robots which led to downtime when a robot wasn't working/limited the slots we could book.

Being unable to command torques → any attempt at commanding torques on robot B led to issues that continuous troubleshooting on piazza and independently was unable to resolve, having gravity compensation would have helped make our solution more robust from run to run. Maybe this could be resolved by using dedicated lab machines over ethernet to isolate version matlab issues+ ethernet/usbc communication challenges.

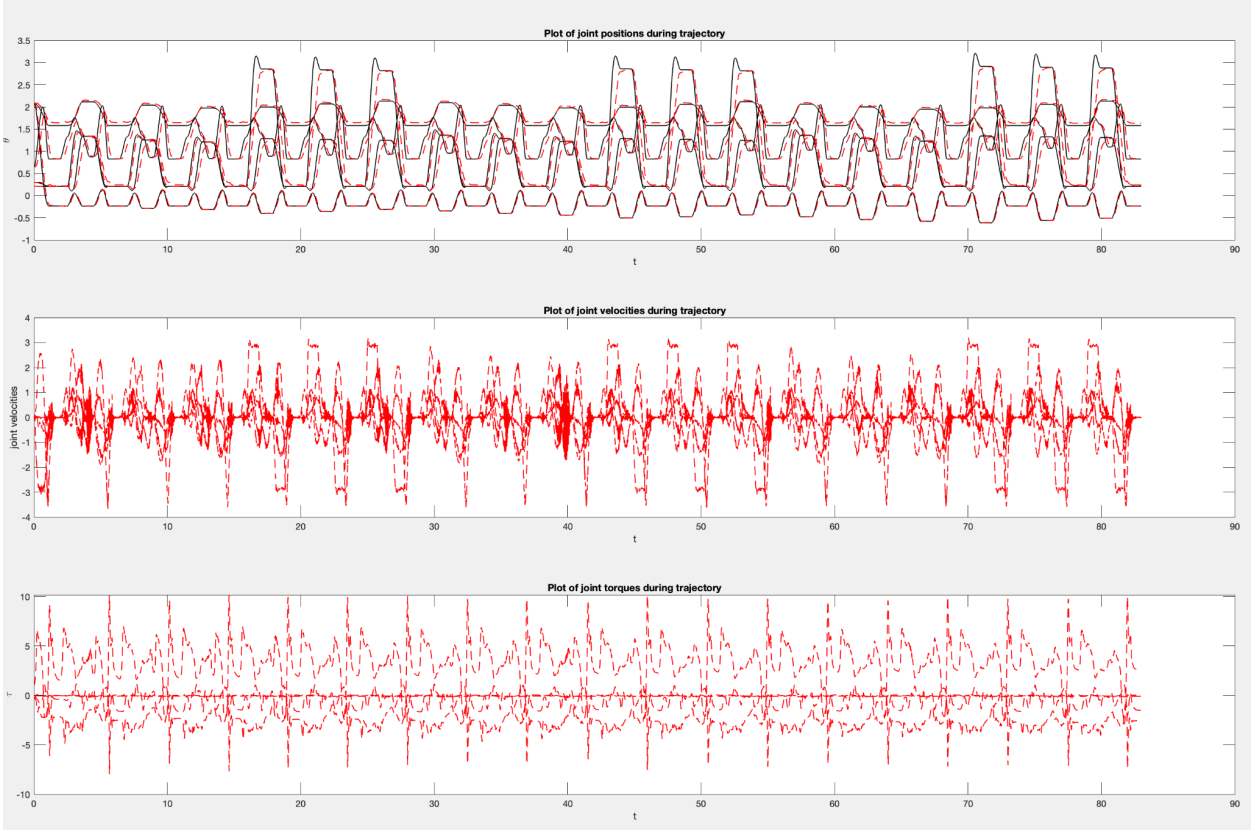
Damage to the jenga block dispenser → this damage led to pick waypoints being inconsistent and … a lot of duct tape managed to mitigate this. Redesigning the dispenser would be a way to make this process more robust.

Inverse Kinematics → We did not manage to successfully implement inverse kinematics after ~15 hours of troubleshooting we decided to cut our losses and teach our waypoints on Tuesday onwards. Getting analytical IK working would go a long way in increasing the robustness of our solution particularly in reducing the overhead required to set waypoints for the six layers since we can just increase z. Spending more time prior to checkpoint to get this working would be worthwhile however this would not be a silver bullet considering the other challenges.

A classmate mentioned using the lagrangian to implement gravity compensation and it appeared to be more robust than the jacobian method which is something that would be interesting to explore in the future.

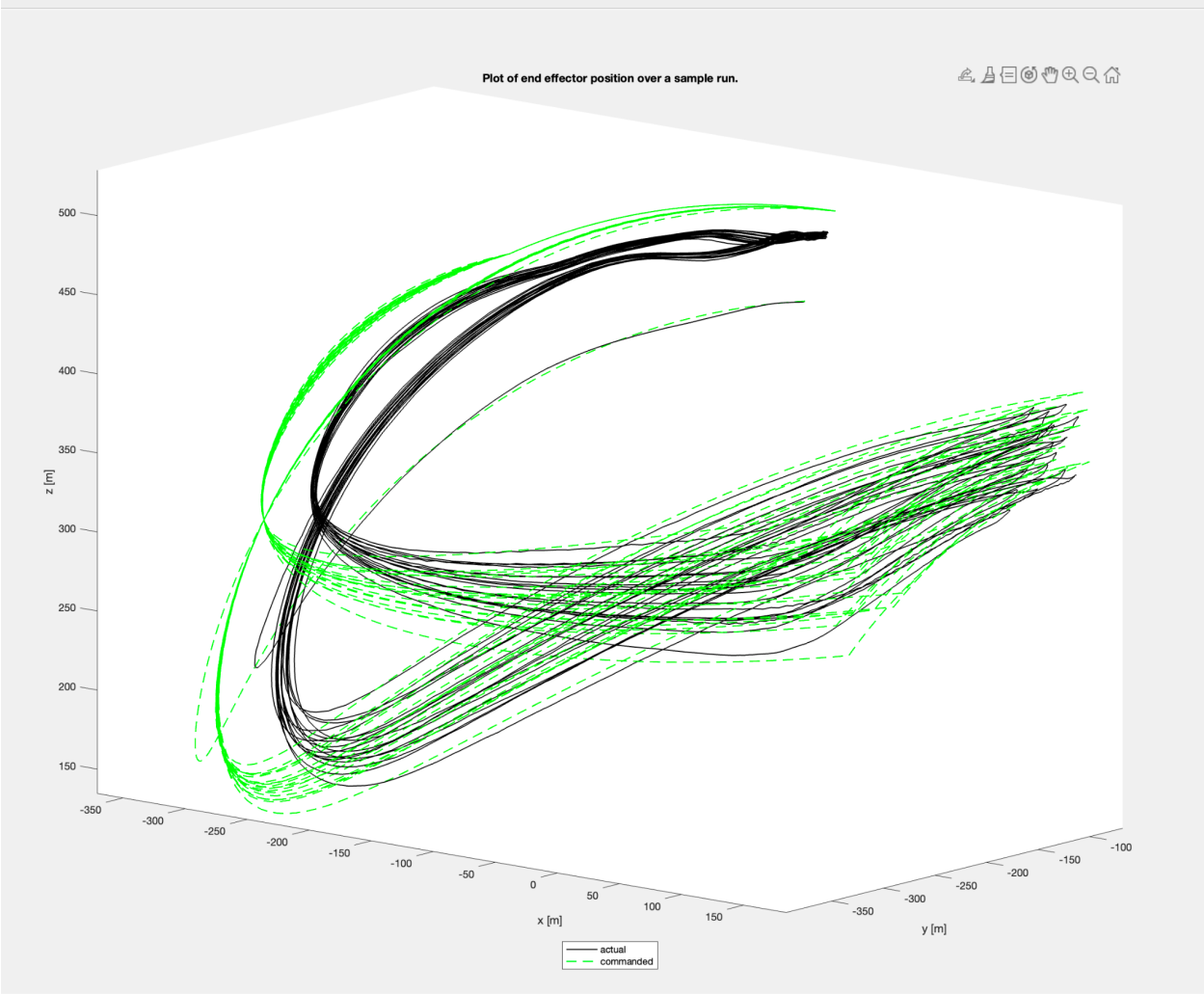
• Include plots of configuration and workspace positions, velocities, and joint torques during the demo run. Describe observations regarding these plots.

Configuration space:

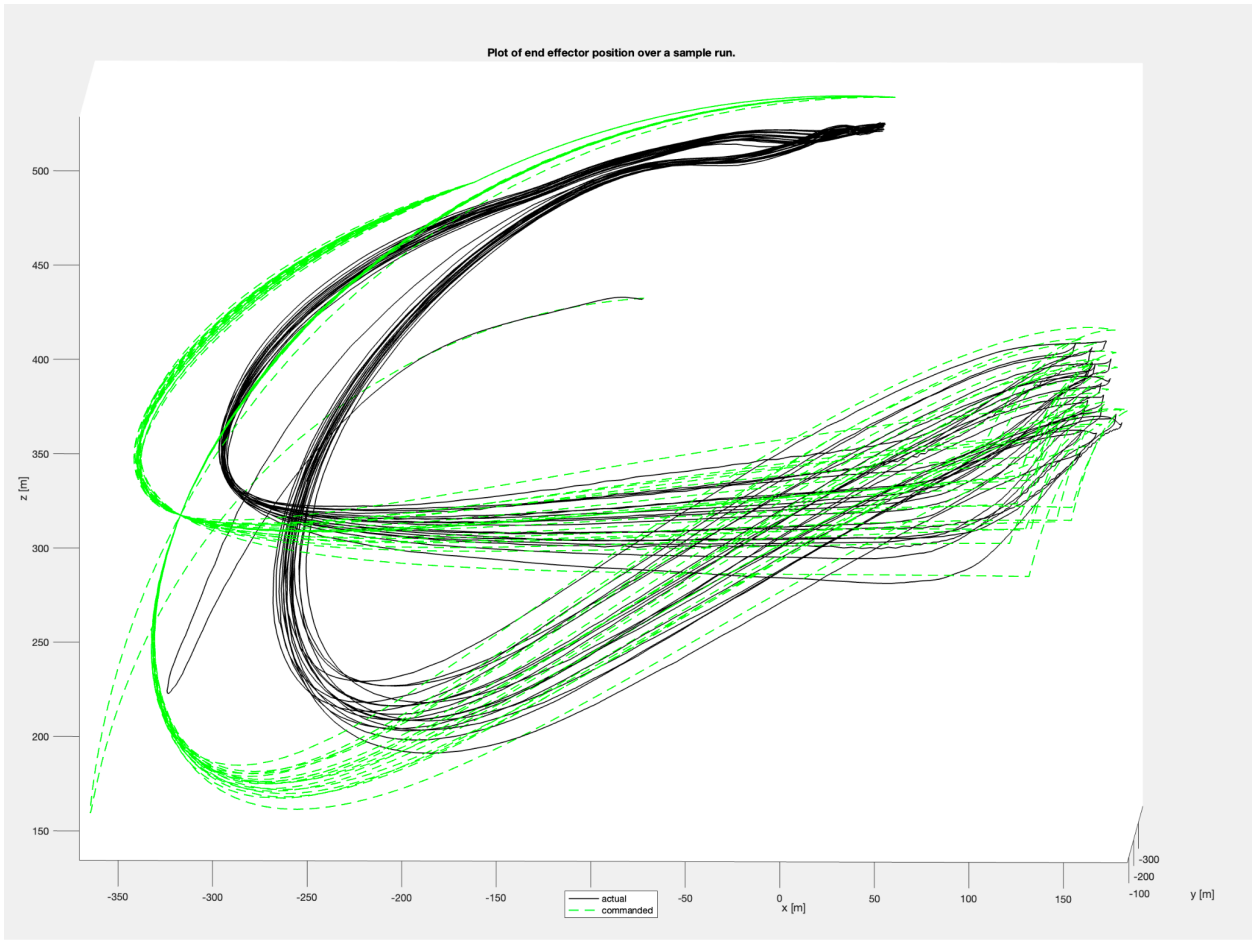


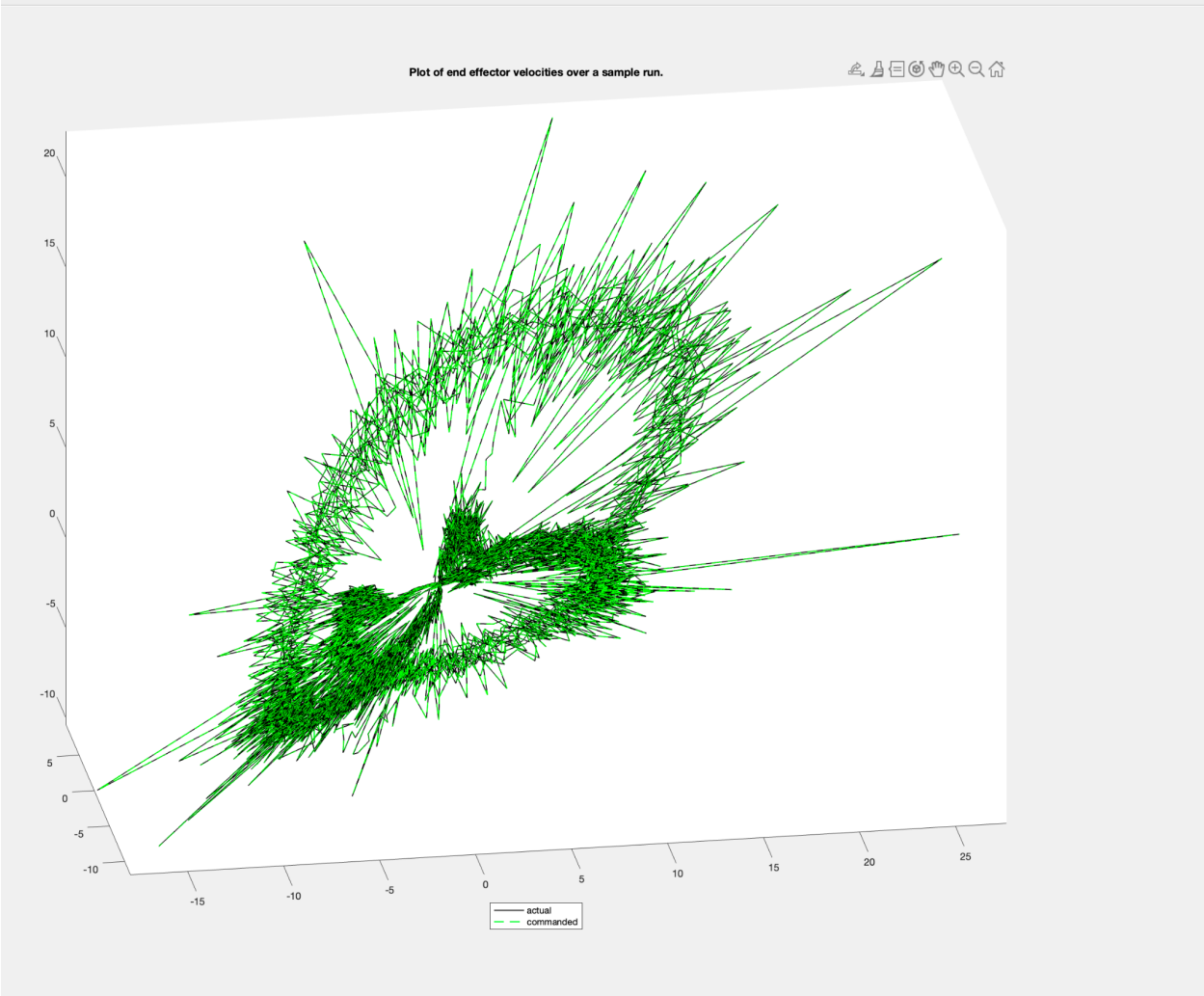
The velocity curve is really noisy, but across all of these we can observe this approximately periodic pattern to position, velocities and torques. The torque curves are rather peaky which some gain tuning/trajectory optimisation could reduce, we can see the change in joint angles when we rotate 90 degrees to place the third layers in those tall peaks. However we can observe in the configuration space that our tracking of joints is not very good, there appears to be both a phase to how closely we are following along with some outright deviations.

Workspace positions:



Here we can observe a challenge discussed earlier. The absence of gravity compensation leads to these deviations from our commanded and actual velocities. Further PID tuning in scope could help alleviate this. Looking at the z axis head on we can see this in clearer detail below: Our hypothesis is that the x, y deviations stem from this lack of gravity comp affecting the curve of the trajectory in z and consequently the x,y but we would have thought there would have been better tracking. A conclusion we can draw from this is that there is either the approach of teaching waypoints + gravity compensation or going with inverse kinematics and adding a sufficient z offset to compensate for gravity would add robustness


Plot of end effector position over a sample run.
z [m]
500
450
400
350
300
250
200
150
-350
-300
-250
-200
-150
-50
0
50
100
150
x [m]
-300
-200
-100
y [m]
actual
commanded



The workspace velocities are very peaky partly because we did not specify them and partly because 5 joints moving transformed into 3 dimensions leads to a complicated curve. The velocity tracking is relatively spot on.

• Give feedback on the course; what did you enjoy and what could be improved?

The lectures were excellent, I think using an ipad would help streamline notes in the future if room logistics place rkd in the same classroom. The homeworks were useful, however a bulk of the work in the homeworks comes from typesetting the matrices rather than actual brain power spent on the problem. The oli pages were super useful to consolidate learning and review along with the videos, but I didn't find the quiz' very helpful especially because some lack answer explanations.

I think one point of improvement with this class would have to be the capstone. It would greatly benefit from clearer logistics, a more robust lab environment with symmetrical robot setups and steps to make sure that the issues we are troubleshooting stem from our own implementation

details rather than attempting to debug the hardware itself. The capstone was engaging to work on practical robotics but it felt like it could have been a lot more fulfilling if we were able to get inverse kinematics working a little more robustly.