Mini- Project 4: Jacobian, Singularity

Robotics 1

Professor John Wen

Varun Dhir

MANE 4560

10/30/21

https://github.com/Varun-ABC/Robotics1/tree/main/mini_project_4

I, Varun Dhir, certify that the following work is my own and completed in accordance with the academic integrity policy as described in the Robotics I course syllabus.

Table Of Contents

Abstract

The goal of this Mini-Project is to look at Jacobians, Singularities, and iterative methods of computing inverse kinematics. It also looked to explain some of the results we achieved in Mini-Project 3.

For the discussion of the Jacobian, 3 different methods are compared, an iterative method, an analytical method based on the Jacobian of joint 4 in the third frame, and MATLAB's inbuilt function "geometricjacobian" based on the MATLAB's Rigid Body Tree. They will be compared on a basis of computation time and on accuracy.

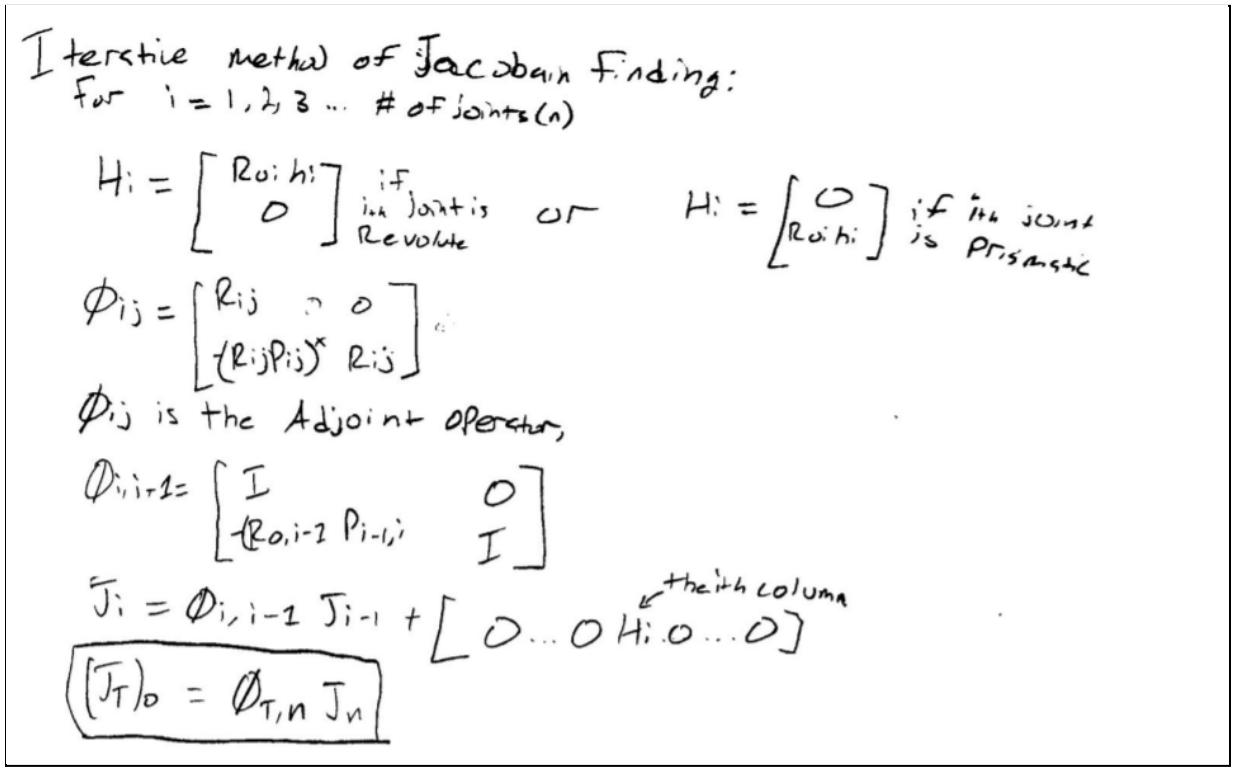
When the robot loses a degree of freedom the arm is in a singular position. This can happen in 3 cases, boundary: where the robot physically can not move past a certain point, wrist: where there are 2 collinear axises, and interior where the robot is susceptible to self motion.

Approach- (What and How)

1. *Jacobian*

The Jacobian provides the relationship between joint velocities and end-effector velocities of the robotic arm. The 3 methods to find the Jacobaian are the iterative method where the column for the current joint is based on previous joints, the analytical method where the spherical wrist is taken advantage of and the Jacobian from the origin to the tool frame $(J_T)_0$ is taken as a function of the Jacobian of joint 4 in the 3 frame $(J_4)_3$. Matlab also has an inbuilt function which computes the jacobian in a similar method to our iterative method.

a) Iterative method



b) Analytical method

Analytical method:

Chosing O_A to be the center of the spherical joint, $O_{4,5,6}$. Thus P_{45} & $P_{46} = 0$

Also chose to look @ J_4 in the third frame, because we can look @ the jacobian in any frame.

$(J_T)_0$ = 6x6 matrix where ith column is

$$\begin{bmatrix} R_{0i} h_i \\ (h_i \times P_{iT})_0 \end{bmatrix}$$

$$(J_T)_3 = \begin{bmatrix} h_3 \\ h_3 \times P_{3T} \end{bmatrix} \quad \& \quad (J_T)_0 = \begin{bmatrix} R_{03} h_3 \\ (R_{03} h_3) \times (R_{03} P_{3T}) \end{bmatrix}$$

$(J_A)_0 = \emptyset_{TA} (J_A)_0$ where $(J_A)_0$ is a 6x6 where the ith column is: $\begin{bmatrix} (h_i)_0 \\ (\vec{h}_i \times P_{iA})_0 \end{bmatrix}$

So the ith column of $(J_{T4})_3 = \begin{bmatrix} h_i \\ (h_i \times P_{i4})_3 \end{bmatrix}$

$$(J_T)_3 = \emptyset_{T4} (J_4)_3 \rightarrow (J_T)_0 = \begin{bmatrix} R_{03} & 0 \\ 0 & R_{03} \end{bmatrix} (J_T)_3$$

$$J_{T0} = \begin{bmatrix} R_{03} & 0 \\ 0 & R_{03} \end{bmatrix} \begin{bmatrix} I & 0 \\ -(P_{4T})^{\times}_3 & I \end{bmatrix} (J_4)_3$$

c) Comparison

To compare different methods of computing the Jacobian, 50 random joint angles are computed and run into each FK solver, then computation time and computation error from the Analytical method is measured.

2. Singularities

When the robot loses a degree of freedom the arm is in a singular position. This can happen in 3 cases, boundary: where the robot physically can not move past a certain point, wrist: where there are 2 collinear axises, and interior where the robot is susceptible to self motion.

a) Analytical Expressions

Analytical expressions can be found 2 ways, the determinant based method or the geometric method. When the determinant of a matrix is 0, the matrix loses rank, this means that there are now some linearly dependent columns.
The other method is to first find the geometric conditions that would lead to singularity and then find the analytical expressions that would result in such configurations. This method was not preferred due to the fact that finding poses by inspection may lead to unexpected singularities being missed. The determinant method, while less intuitive, leads to a more consistent answer.
It also follows that when the determinant of $(J_4)_3$ is 0 the determinant of $(J_T)_0$ is also 0.
The approach followed then is, to find the determinant of $(J_4)_3$ then find the zeros of that equation, then find the geometric interpretations of these analytical expressions.

3. Inverse Kinematics

Last project we looked at subproblem decomposition for inverse kinematics. This is a very powerful and very fast way of computing inverse kinematics, however for redundant arms it is not always viable. Iterative IK methods work regardless of how many joints we have.

a) Iterative Inverse Kinematics

For Iterative IK, there are a few methods that can be used, Newton's method, Gradient Descent and Damped Least Squares, shown below. Newton's method and Gradient Descent are both susceptible to failing miserably near singularities. If the configuration is near singularity, the pseudoinverse method (Newton's Method), will generate very large changes in joint angles even for small errors, thus the step size alpha must be small, leading to slow convergence to a solution. For Gradient descent the opposite problem occurs where the transpose of the jacobian is small near singularity, thus alpha must be chosen to be large leading to imprecise convergence. For these reasons, these methods are not preferred to the Damped Least Squares method. Damped Least Squares solves the instability near singularities by introducing a damping term, ϵ, this term should be large enough that the output provides good results near singularities but not too large that that is slows down the convergence to the solution.

Newton:

$$\dot{q} = -KJ^{+}(q) \begin{bmatrix} w\ (f_R(q) - q_{T_d}) \\ f_p(q) - p_d \end{bmatrix}$$

Gradient:

$$\dot{q} = -KJ^{T}(q) \begin{bmatrix} w\ (f_R(q) - q_{T_d}) \\ f_p(q) - p_d \end{bmatrix}$$

Damped Least Square:

$$\dot{q} = -KJ^{T}(q)(J(q)J^{T}(q) + \epsilon I)^{-1} \begin{bmatrix} w\ (f_R(q) - q_{T_d}) \\ f_p(q) - p_d \end{bmatrix}$$

To improve iterative IK, two additional steps were tried. First, a tolerance was set, this cuts off the iterations once the error is below a desired amount. Second, once the error is below a certain amount, the step size is reduced. This allows the solution to converge very fast at first and then allows more precision as it nears the target. The first modification reduces the amount of time it takes for the solution to converge and the second may increase it, so these tolerances can be tuned to give a desired error and convergence time.

Orientation error in SO(3) can be found with 4 different metrics, shown below. e_{R2} and e_{R3} are chosen in this report to determine which error metric is more reliable and converges to a better solution. For e_{R2} , the orientation error, s, is equal to 2 times the unit quaternion of the rotation matrix at the current iteration times the transpose of the desired rotation matrix, as represented here: $s = 2 * q_v(R R_d^T)$. For e_{R3} , s, is equal to 2 times the angle axis representation of the rotation matrix at the current iteration times the transpose of the desired rotation matrix, or represented as $s = 2 * K\theta(R R_d^T)$.

Orientation: $\dot{e}_R = s^T \omega$

$$\dot{e}_{R1} = (4 q_0 q_v)^T \omega$$
$$\dot{e}_{R2} = (2 q_v)^T \omega$$
$$\dot{e}_{R3} = 2(\theta k)^T \omega$$
$$\dot{e}_{R4} = (J_\psi^T \psi)^T \omega$$

Position error is simply found by subtracting the current position by the desired position and taking the 2 norm of it.

To verify the result, 50 randomly generated end effector poses are generated and the error, found using the Frobenius norm of the difference of the desired pose and the achieved pose, is plotted to see how good the convergence is and how long it takes.

b) Parameters finding

To find a good set of parameters; (N, alpha, el and weights) and the same test procedure as described above to see the effect of changing each thing independently. This of course assumes that they are independent variables, when they are likely not, however, it does provide some intuition as to how changing parameters affects the final solution. To keep consistency between trials, a constant seed is placed prior to creating the random numbers.

4. *Jacobian and Trajectory Motion*

Understanding the Jacobian's role in constraining path velocity is critical to understanding why certain solutions are better than others; here we will look at why this is the case when path velocity is constrained to be constant and there is a constraint on maximum joint velocity.

a) Singular Value Decomposition

The singular value decomposition of the Jacobian is related to how close the robot is to singularity. The lower the minimum value is the closer to singularity. Since it's closer to singularity, the robot has less mobility in one DOF, thus movements in that direction will require more displacement from other joints. Thus, with constant path speed the point on the path with the lowest singular value is likely to be the point on the path that constrains the maximum possible path speed.

Results

1. *Jacobian*

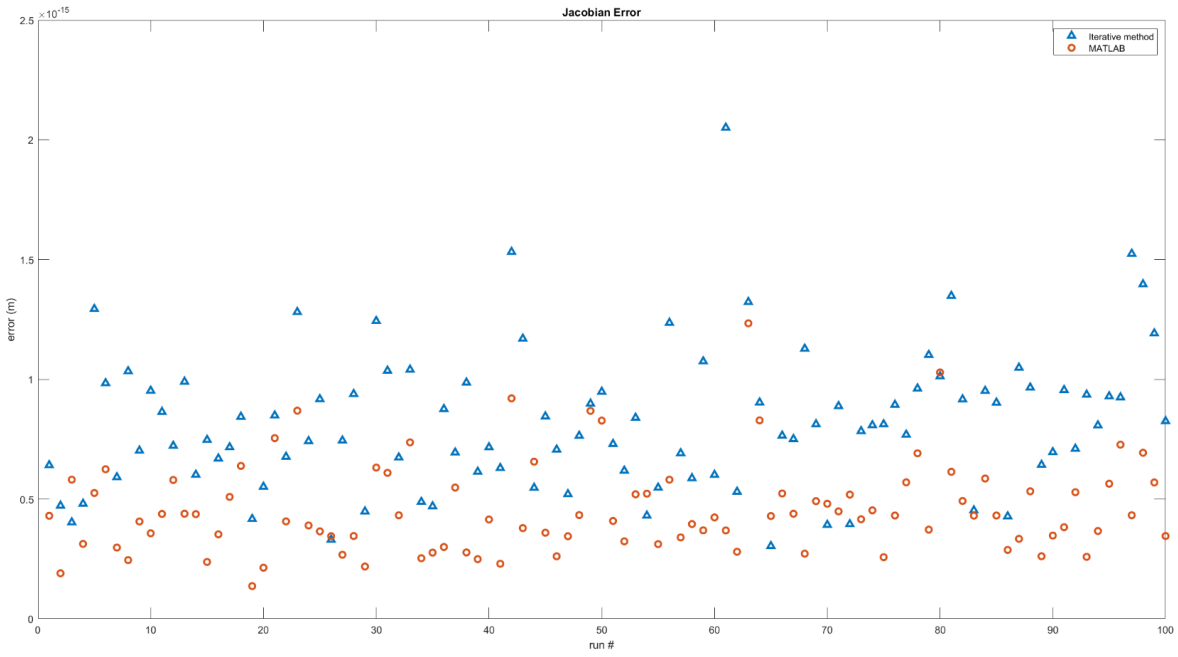
a) Analytical Jacobian

Shown below is $(J_4)_3$

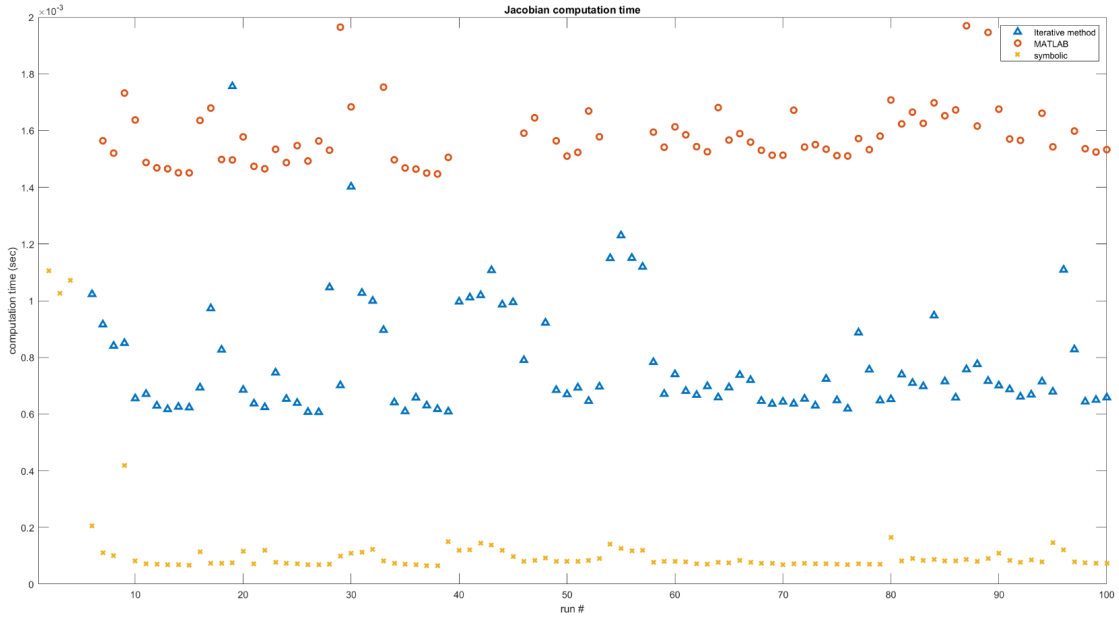
```
/                   -sin(q2 + q3),                          0,          0,  1,    0,          cos(q5)     \
|                                                                                                         |
|                         0,                                1,          1,  0, cos(q4),   sin(q4) sin(q5) |
|                                                                                                         |
|                   cos(q2 + q3),                           0,          0,  0, sin(q4), -cos(q4) sin(q5) |
|                                                                                                         |
|                         0,                          L3 + L2 cos(q3),  L3, 0,    0,            0         |
|                                                                                                         |
| L4 cos(q2 + q3) + L3 sin(q2 + q3) + L2 sin(q2),         0,          0,  0,    0,            0         |
|                                                                                                         |
\                         0,                          L2 sin(q3) - L4, -L4, 0,    0,            0        /
```

b) Comparison

The comparison in for computation error from the analytical solution:



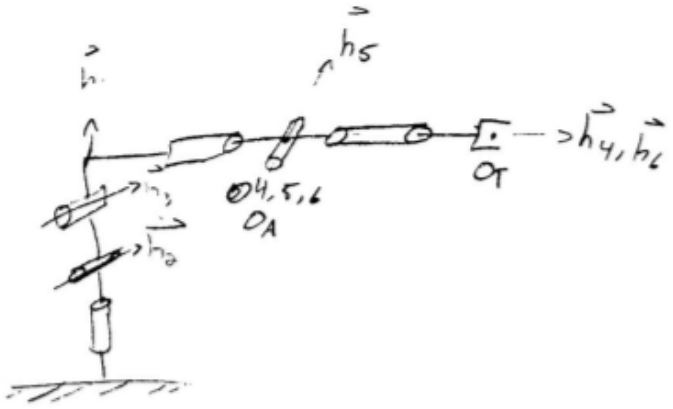
The comparison in for computation time:



The difference in computation time between the iterative and MATLAB's solution may have to do with the precision it carries out, this can be seen with the smaller error and longer computation time. The difference in computation time between the iterative methods and symbolic is because the iterative methods do matrix multiplication once for every joint, where the symbolic does it twice to convert from $(J_4)_3$ to $(J_4)_0$ to $(J_T)_0$, resulting in a much lower total computation time.

2. Singularities

a) <u>Analytical Expressions</u>



$$(J_4)_3 = \begin{bmatrix} -\sin(\xi_2+\xi_3) & 0 & 0 & 1 & 0 & \cos(\xi_5) \\ 0 & 1 & 1 & 0 & \cos(\xi_4)\sin(\xi_5) & \sin(\xi_5)\cdot\sin(\xi_4) \\ \cos(\xi_2+\xi_3) & 0 & 0 & 0 & \sin(\xi_4) & -\cos(\xi_4)\sin(\xi_5) \\ 0 & L_3+L_2\cos(\xi_3) & L_3 & 0 & 0 & 0 \\ L_4\cos(\xi_2+\xi_3)+L_3\sin(\xi_2+\xi_3)+L_2\sin(\xi_2) & 0 & 0 & 0 & 0 & 0 \\ 0 & L_2\sin(\xi_3)-L_4 & -L_4 & 0 & 0 & 0 \end{bmatrix}$$

$$\det(J_4)_3 = L_2\sin(\xi_5)\left(L_4\cos(\xi_3)+L_3\sin\xi_3\right)\left(L_4\cos(\xi_2+\xi_3)+L_3\sin(\xi_2+\xi_3)+L_2\sin(\xi_2)\right)$$

Singularity when $\det(J_4)_3 = 0$:

1) $L_2\sin(\xi_5) = 0$

2) $L_4\cos\xi_3 + L_3\sin\xi_3 = 0$

3) $L_4\cos(\xi_2+\xi_3) + L_3\sin(\xi_2+\xi_3) + L_2\sin(\xi_2) = 0$

eq. 1)

$L_2 \sin(\xi_5) = 0$

$\xi_5 = 0, -\pi, \pi$ } Since $\xi_5 = -\pi, \pi$ is not Physically Possible due to joint limits & body collision $\xi_5 = 0$ for singularity.

$L_4 \cos\xi_3 + L_3 \sin(\xi_3) = 0$ eq. 2

$-L_4 \cos(\xi_3) = L_3 \sin(\xi_3)$

$\frac{-L_4}{L_3} = \tan(\xi_3)$

$\tan^{-1}\left(\frac{-L_4}{L_3}\right) = \xi_3$

eq. 3 $L_4 \cos(\xi_2+\xi_3) + L_3 \sin(\xi_2+\xi_3) + L_2 \sin(\xi_2) = 0$

$L_4 \cos\xi_2 \cos\xi_3 - L_4 \sin\xi_2 \sin\xi_3 + L_3 \sin\xi_2 \cos\xi_3 + L_3 \cos\xi_2 \sin\xi_3 + L_2 \sin\xi_2 = 0$

$\cos\xi_2(L_4 \cos\xi_3 + L_3 \sin\xi_3) + \sin\xi_2(-L_4 \sin\xi_3 + L_3 \cos\xi_3 + L_2) = 0$

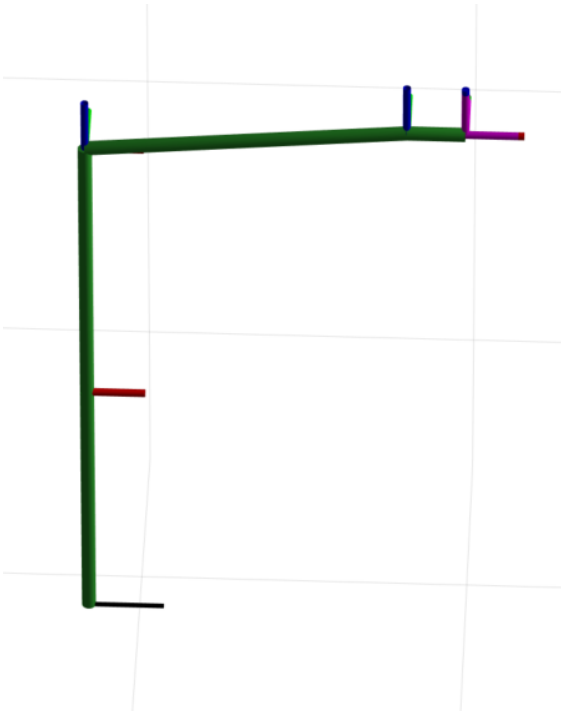
$\cos\xi_2(L_4 \cos\xi_3 + L_3 \sin\xi_3) = -\sin\xi_2(-L_4 \sin\xi_3 + L_3 \cos\xi_3 + L_2)$

$-\tan(\xi_2) = \frac{L_4 \cos\xi_3 + L_3 \sin\xi_3}{-L_4 \sin\xi_3 + L_3 \cos\xi_3 + L_2}$

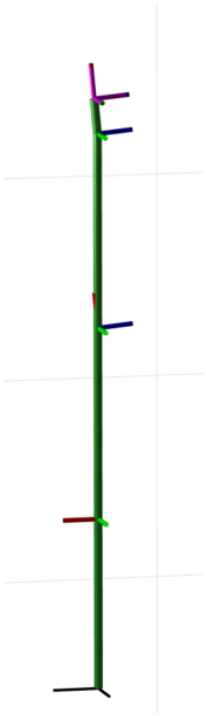
$\xi_2 = \text{atan}\left[\frac{-L_4 \cos\xi_3 - L_3 \sin\xi_3}{-L_4 \sin\xi_3 + L_3 \cos\xi_3 + L_2}\right]$

Geometric conditions from:

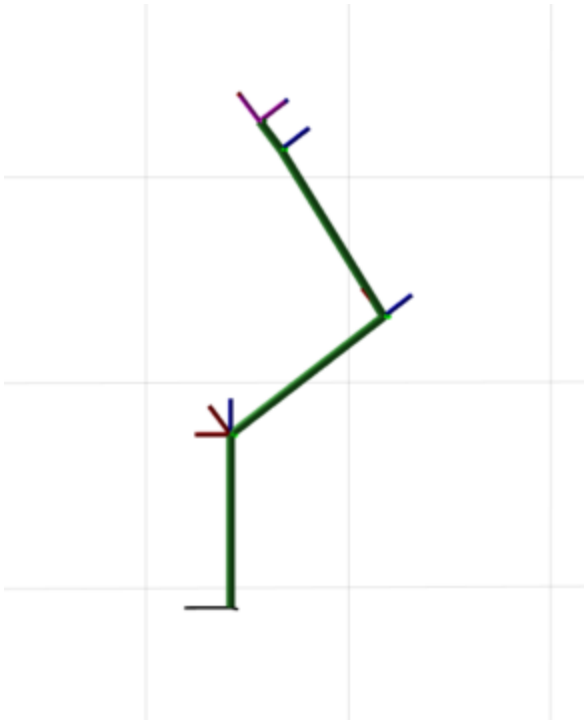
- $q5 = \pi, -\pi,$ *or* 0: since $q5 = \pi, -\pi$is not possible, as it means the arm would be folded in on itself, q5 = 0 means that h6 and h4 can not be collinear. When they are, the wrist can roll and it can pitch but it can't yaw. Essentially, a rotation of joint 4 does the same thing as joint 6 rotating the same amount. On the ABB arm, it looks like this:



$tan^{-1}(\frac{-L_4}{L_3}) = q3$: this is a boundary condition and it means the arm is stretched all the way up. On the ABB arm it looks like this:



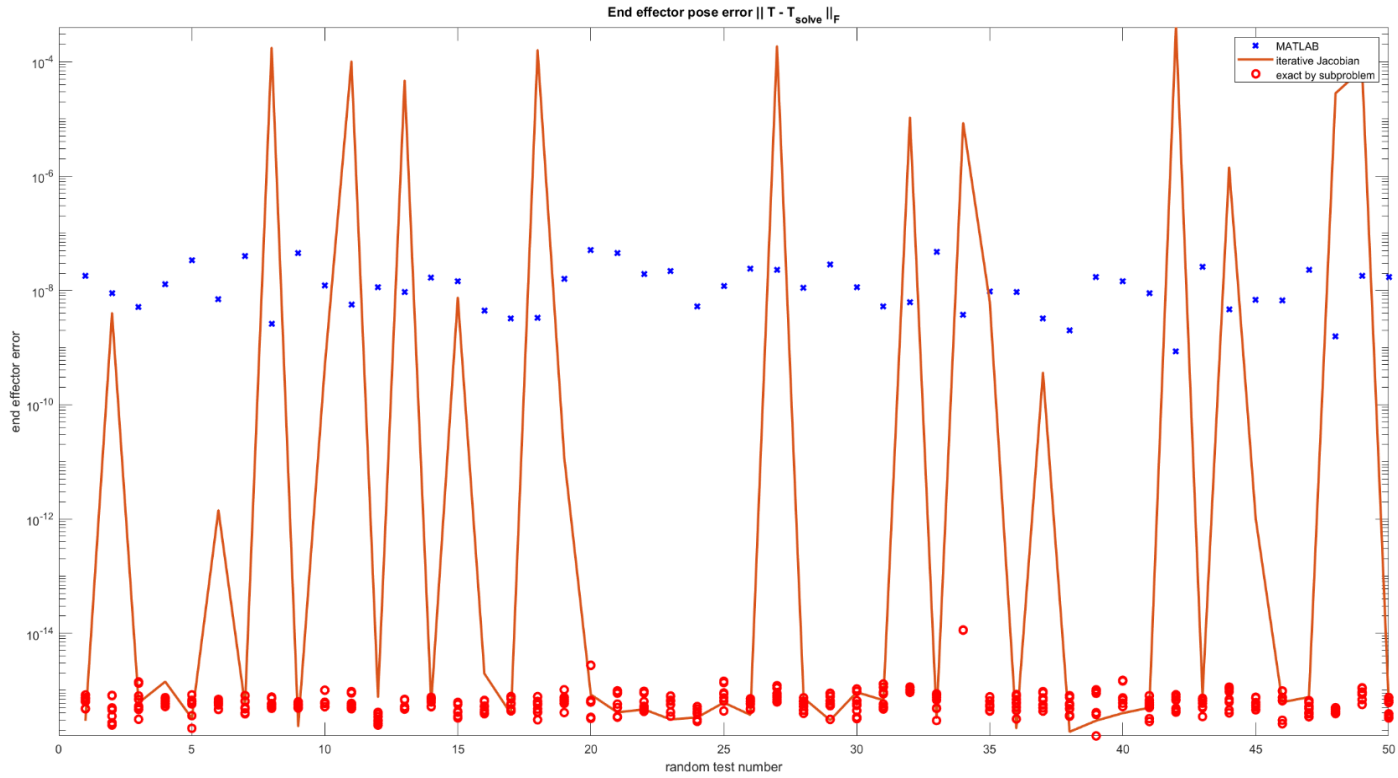
- $q2 = \frac{-L4 * cos(q3) - L3sin(q3)}{-L4sin(q3)+L3cos(q3)+L2}$: This is when the tool frame is vertically aligned with h1. Any rotation of Joint 1 will result in no translation of the tool frame in x, y, or z. On the ABB arm it looks like this:



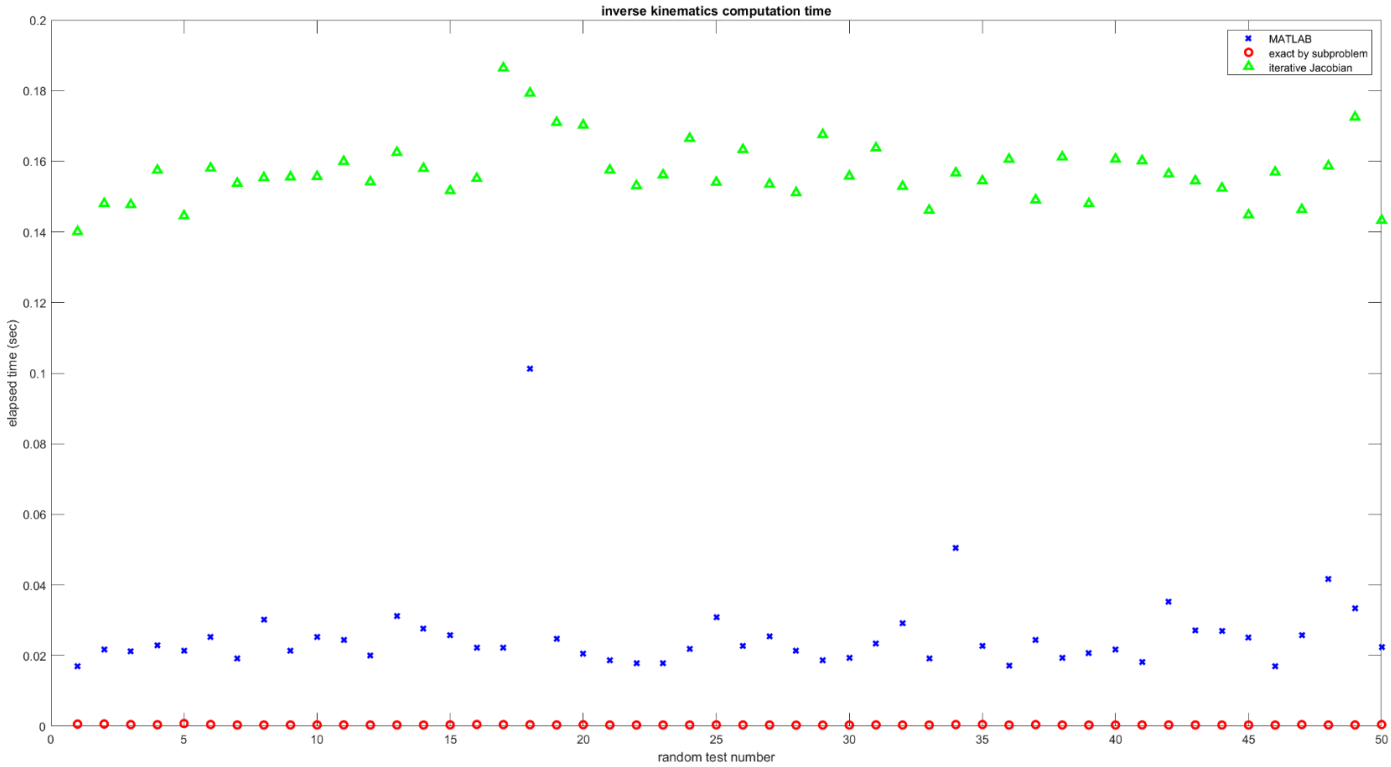
3. *Inverse Kinematics*

a) Iterative Inverse Kinematics

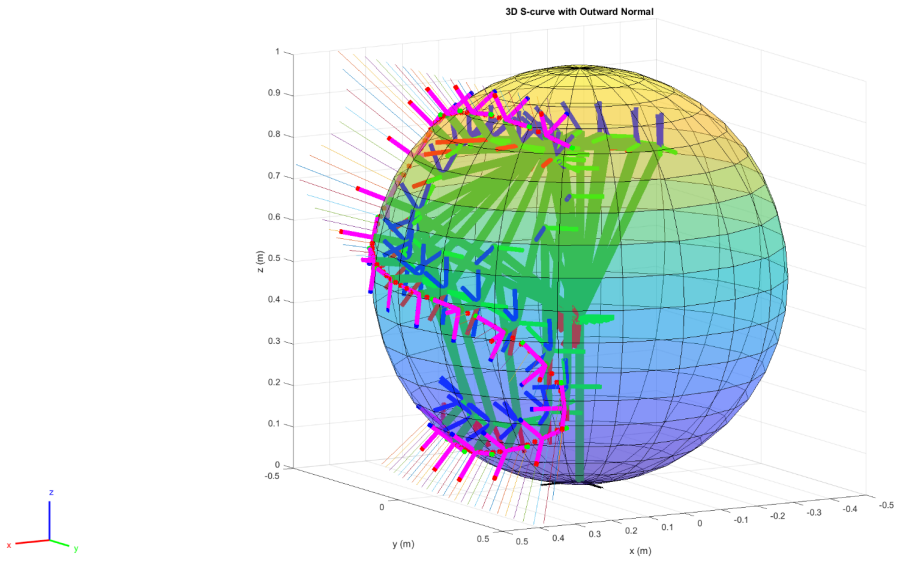
With N = 300, weights = [1.5,1.5,1.5,1,1,1], alpha = .2 and el = .005 for the iterative method the error plot, please note that it is a semi log plot:



The computation time plot:



The Iterative IK is also implemented to ensure that the tests are accurate.

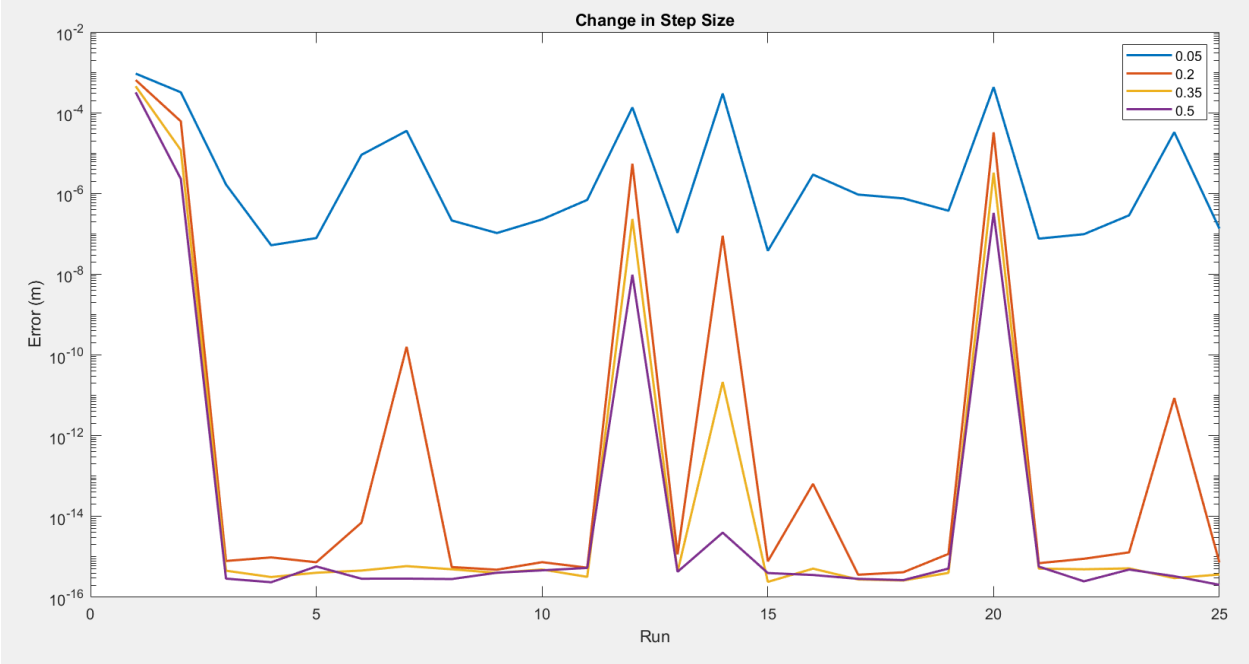


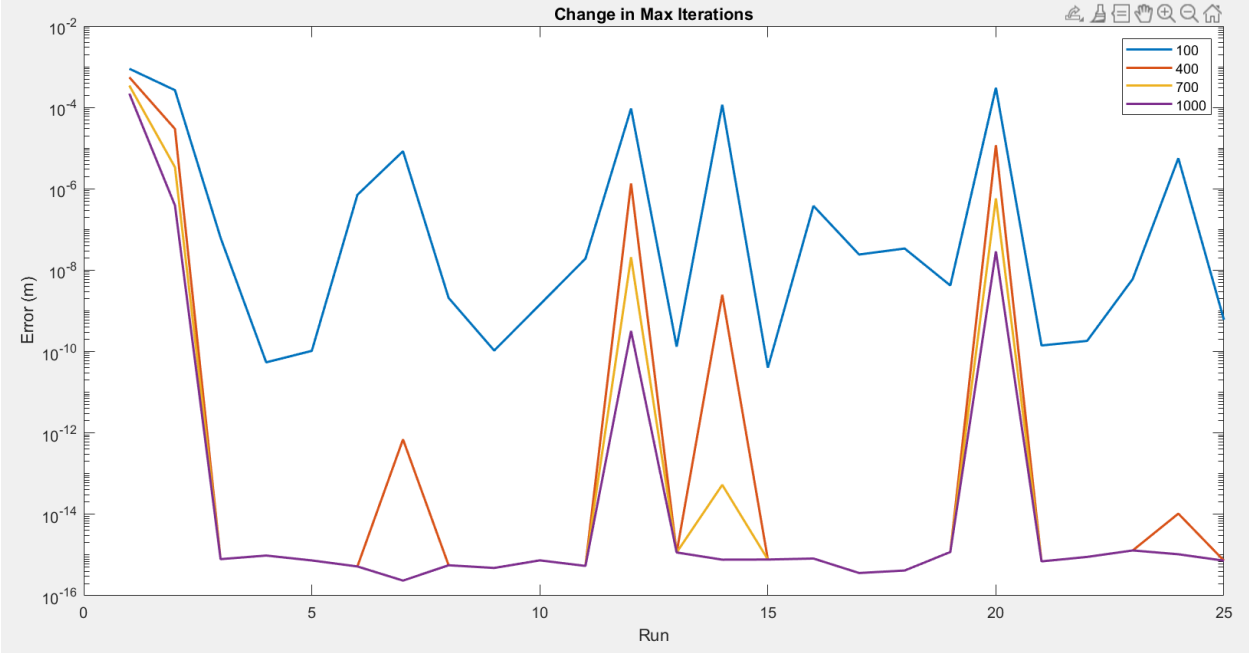
(Full video on Github “SixDOFArm_iter.mp4”)

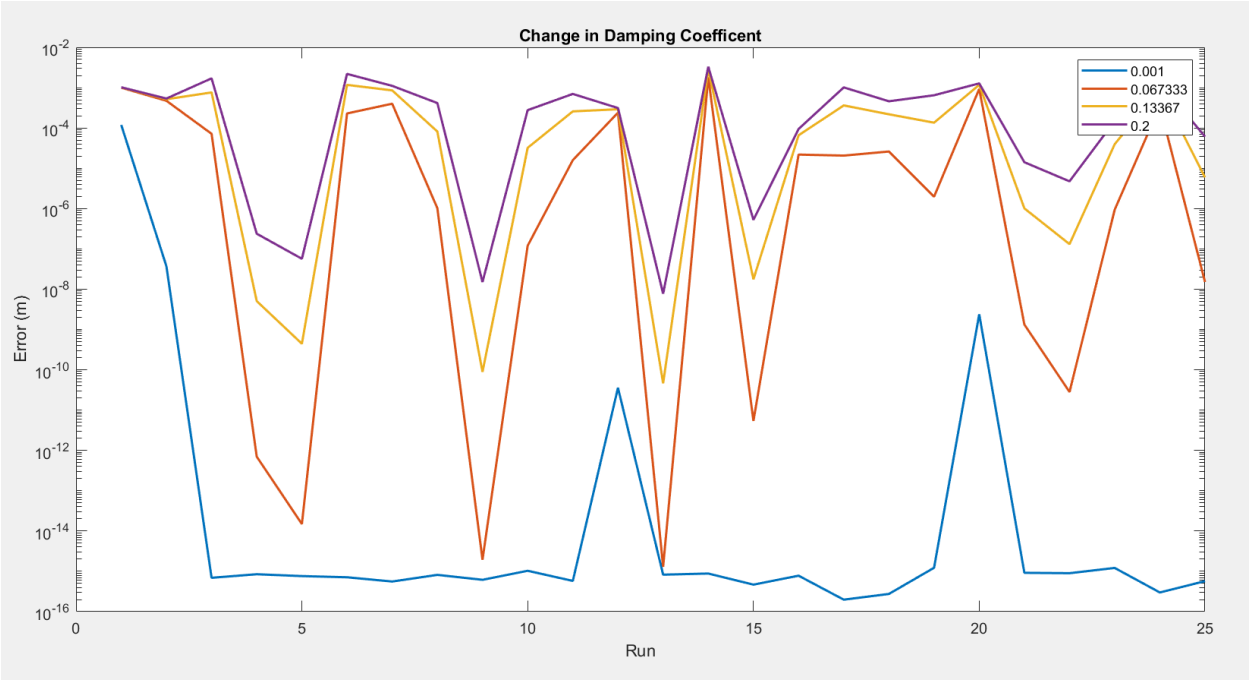
b) Parameters

The results of the parameter finding exercise are shown below.
The general parameters used are:
Maximum iterations = 300
Step size = .2;
Damping coefficient = .005;
Weight = [1,1,1,1,1,1];
For this exercise, only the orientation weight is changed, so it would look like this:
[w,w,w,1,1,1]


Change in Weights
Error (m)
Run
0.1
0.73333
1.3667
2


Change in Step Size
Error (m)
Run
0.05
0.2
0.35
0.5





From this we can see that reducing the step size doesn't always reduce the error, which is something that runs counter to intuition. The best step size found is around .5. I expected that reducing step size would converge to a better answer, but it is possible that I did not give it enough iterations to converge. This makes sense as the smaller alpha requires longer to converge and larger values of alpha converge faster with the risk of not converging at all.

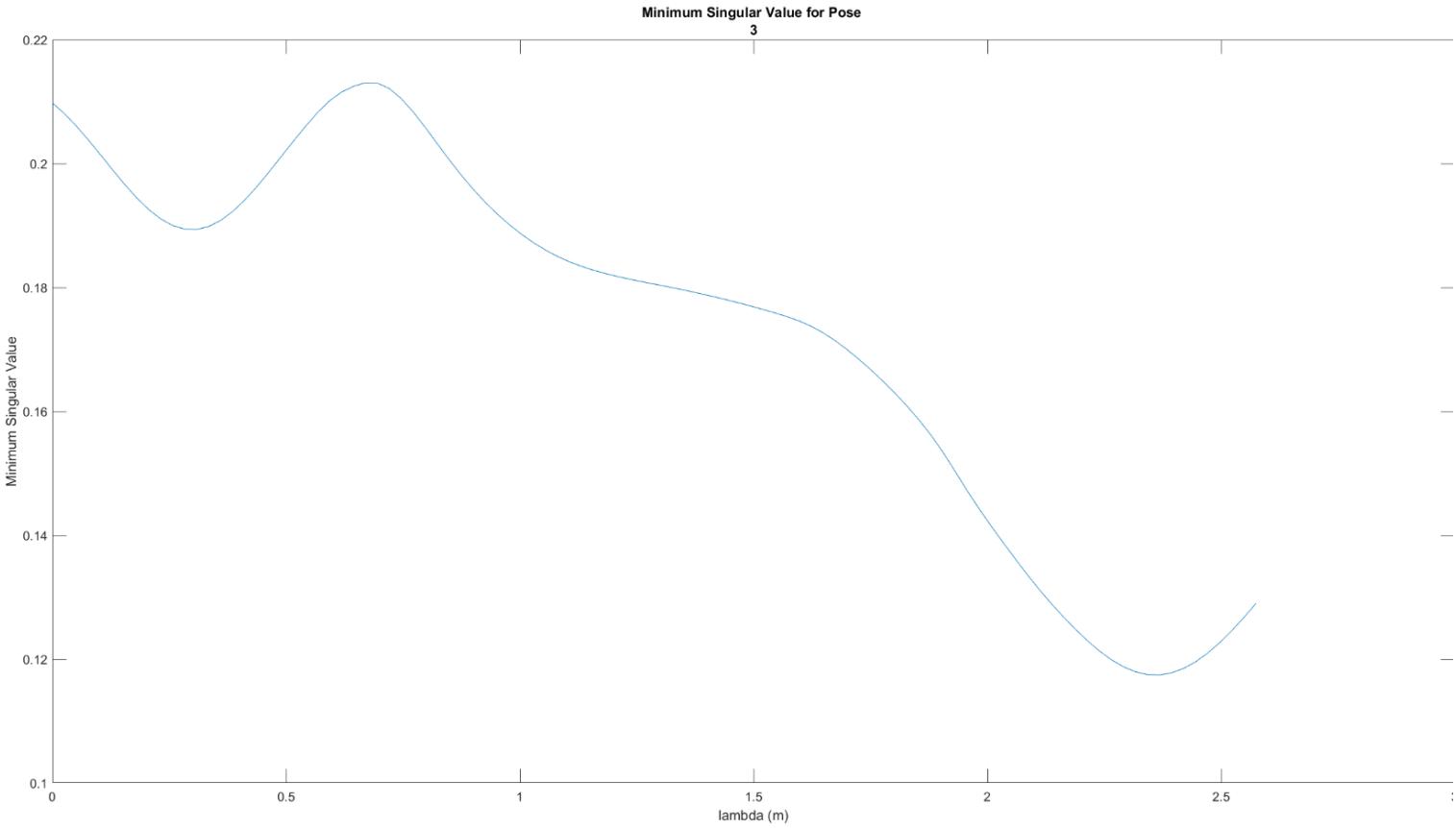
The weight has a massive effect on the final solution, in general, a weighting of between 1.3 2 converges reasonably well.
It was expected that increasing the maximum iterations would reduce error as we are giving more iterations for the solution to converge, however, this is at the expense of computation time.
Changing the damping coefficient also has a massive effect on error, very low damping coefficient seems to result in much lower error, near .001.

c) Compare to Analytical Solution

In terms of error, the analytical solution beats the iterative methods almost every time. It is also not susceptible to failing near singularities in a similar fashion to the iterative methods. Likewise, in computation time the analytical solution is orders of magnitude smaller than the iterative solution. The iterative can be better, in computation time but only at the expense of precision.

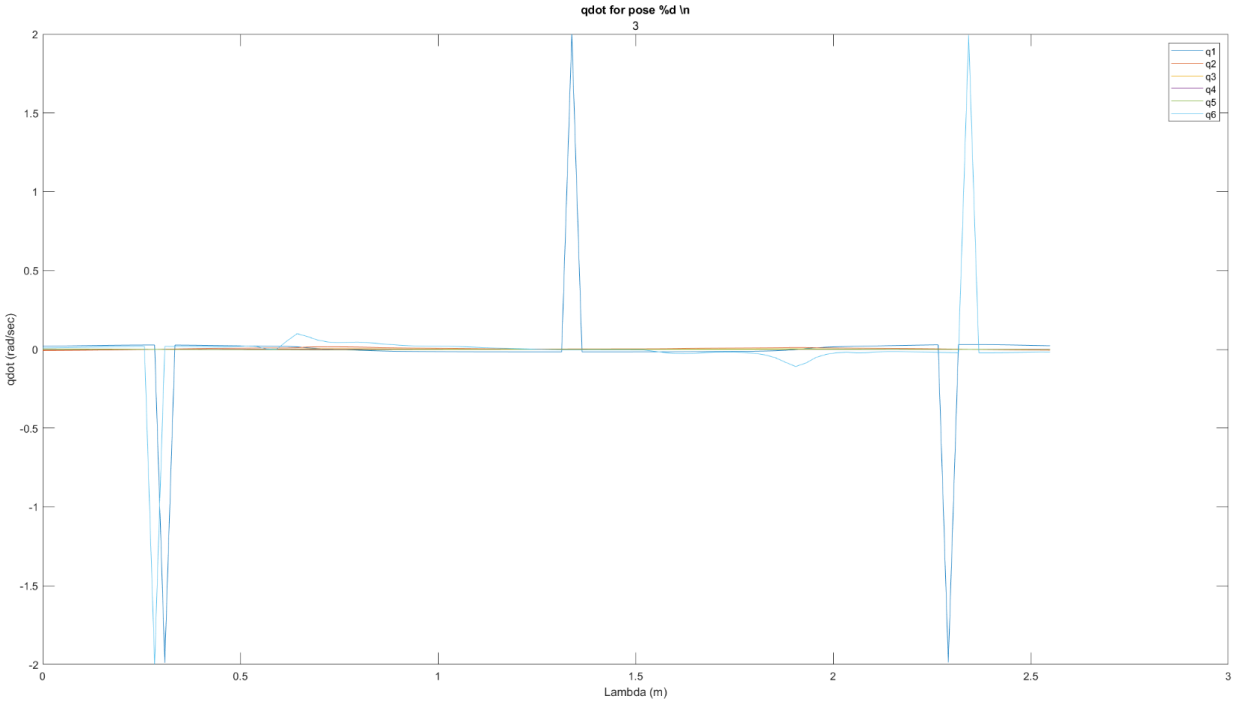
4. *Jacobian and Trajectory Motion*

a) Minimum Singular Value



b) Relation to path speed

The singular value decomposition of the Jacobian is related to how close the robot is to singularity. The lower the minimum value is the closer to singularity. Since it's closer to singularity, the robot has less mobility in one DOF, thus movements in that direction will require more displacement from other joints. Thus, with constant path speed the point on the path with the lowest singular value is likely to be the point on the path that constrains the maximum possible path speed. This can be demonstrated by looking at the qdot for pose 3 from mini-project 3. It can be seen that when we want path speed to be constant, when the jacobian goes close to zero, there are peaks in the joint speed, this may be to compensate for the loss of a DOF.


qdot for pose %d \n
3
q1
q2
q3
q4
q5
q6
qdot (rad/sec)
Lambda (m)
2
1.5
1
0.5
0
-0.5
-1
-1.5
-2
0
0.5
1
1.5
2
2.5
3

Conclusion

This paper looks at a lot of concepts regarding jacobian, singularities and iterative inverse kinematics. The most important conclusion from our results is to use the analytical solution whenever possible. It will save computation time and effort.

The project also taught that a good understanding of singularities will aid especially, when finding solutions iteratively. Finding jacobian singularities also aids in path speed, if it is possible to avoid singularity to get from point A to point B it should be done.

References

Wen, J. T. (2021). ”MANE 4560 - Mini-Project 4: Jacobian, Singularity” Rensselaer Polytechnic Institute

Buss, Samuel R. “Introduction to Inverse Kinematics with Jacobian Transpose, Pseudoinverse and Damped Least Squares Methods.” Carnegie Mellon University, 2009. http://www.cs.cmu.edu/~15464-s13/lectures/lecture6/iksurvey.pdf.