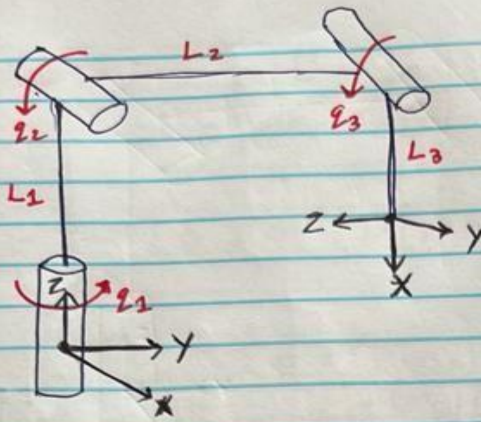
Section2:

Problem-1:

A].

The calculations for defining the home configuration (M) and the screw axis of the given 3-DOF robot are as follows:
Please take note that L1 = L2 = L3 = 0.3m.

MW2 problem 1



$\rightarrow$ $M = \begin{bmatrix} R & P \\ 0 & 1 \end{bmatrix}$

$R = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & -1 \\ -1 & 0 & 0 \end{bmatrix}$ $P = [0 \;\; L_2 \;\; (L_1 - L_3)]^T$

$$M = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & -1 & L_2 \\ -1 & 0 & 0 & (L_1 - L_3) \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$\rightarrow$ $S_1 = [w_1 \;\; v_1]^T$

$w_1 = [0 \;\; 0 \;\; 1]^T$ $P_1 = [0 \;\; 0 \;\; 0]^T$ $V_1 = -[w_2] P_1 = [0 \;\; 0 \;\; 0]$

$\therefore \boxed{S_1} = [0 \;\; 0 \;\; 1 \;\; 0 0 0]^T$

$\rightarrow$ $S_2 = [w_2 \;\; V_2]^T$

$w_2 = [1\ 0\ 0]^T \quad P_2 = [0\ \ 0\ L1]^T$

$v_2 = -[w_2]P_2 = -\begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & 1 & 0 \end{bmatrix}\begin{bmatrix} 0 \\ 0 \\ L1 \end{bmatrix} = -\begin{bmatrix} 0 \\ -L1 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ L1 \\ 0 \end{bmatrix}$

$S_2 = [1\ 0\ 0\ 0\ L1\ 0]^T$

$\rightarrow S_3 = [w_3\ v_3]^T$

$w_3 = [1 0 0]^T \quad P_3 = [0\ L_2\ L1]^T$

$V_3 = -[w_3]P_3 = -\begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & 1 & 0 \end{bmatrix}\begin{bmatrix} 0 \\ L2 \\ L3 \end{bmatrix} = -\begin{bmatrix} 0 \\ -L3 \\ L2 \end{bmatrix} = \begin{bmatrix} 0 \\ L3 \\ -L2 \end{bmatrix}$

$S_3 = [1\ 0\ 0\ 0\ L3\ -L2]^T$

B].

I have calculated the homogenous transformation matrix of the home configuration (M) and used it to calculate the forward kinematics of the robot along with the “fkine” function developed for section-1 of this homework assignment.
**The code for the fkine function can be found in the fkine.m file of the submission.

C].

I calculated the space jacobian using the jacob0 function developed for the section-1 of this homework assignment.
**The code for the jacob0 function can be found in the jacob.m file of the submission.

D].

I have implemented NEWTON-RAPHSON numerical method to find the inverse kinematic solution for the given robotic manipulator. It iteratively calculates detlaQ using the following equation -

deltaQ = pinv(J)*(target_twist – current_twist)

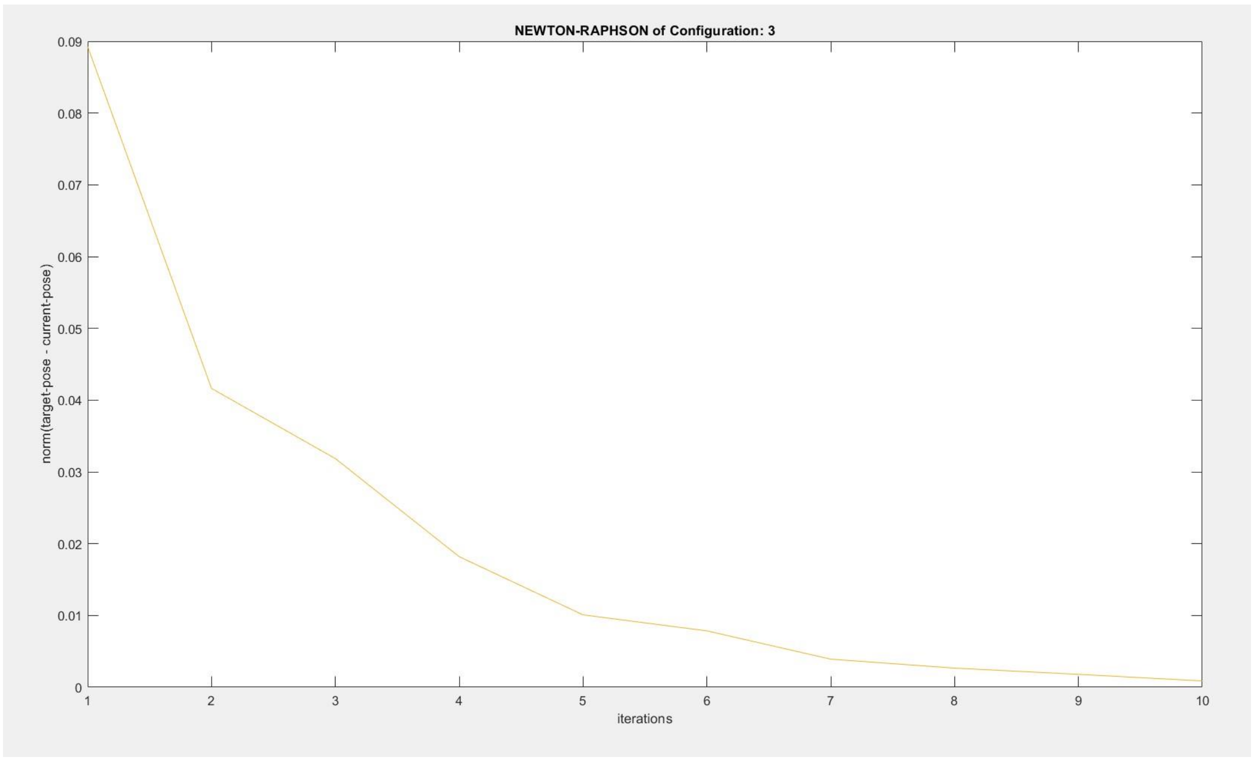
here,

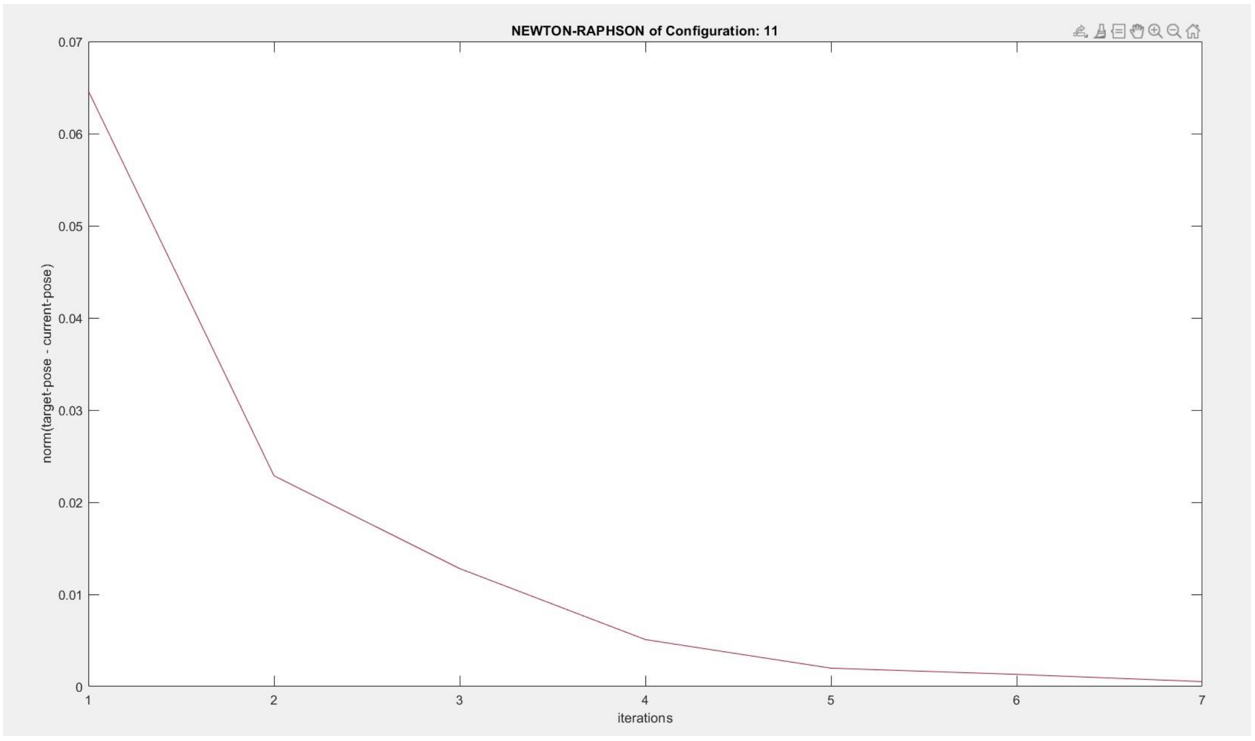
- deltaQ is a vector whose elements represent the change in joint values from the previous joint values.
- pinv(J) is Moore-Penrose pseudo-inverse of the jacobian matrix. Pseudo-inverse is used to check abruptly high joint values.
- target_twist is twist of the final desire pose of the end effector.
- Current_twist is the twist of the current pose (at current iteration) of the end effector.

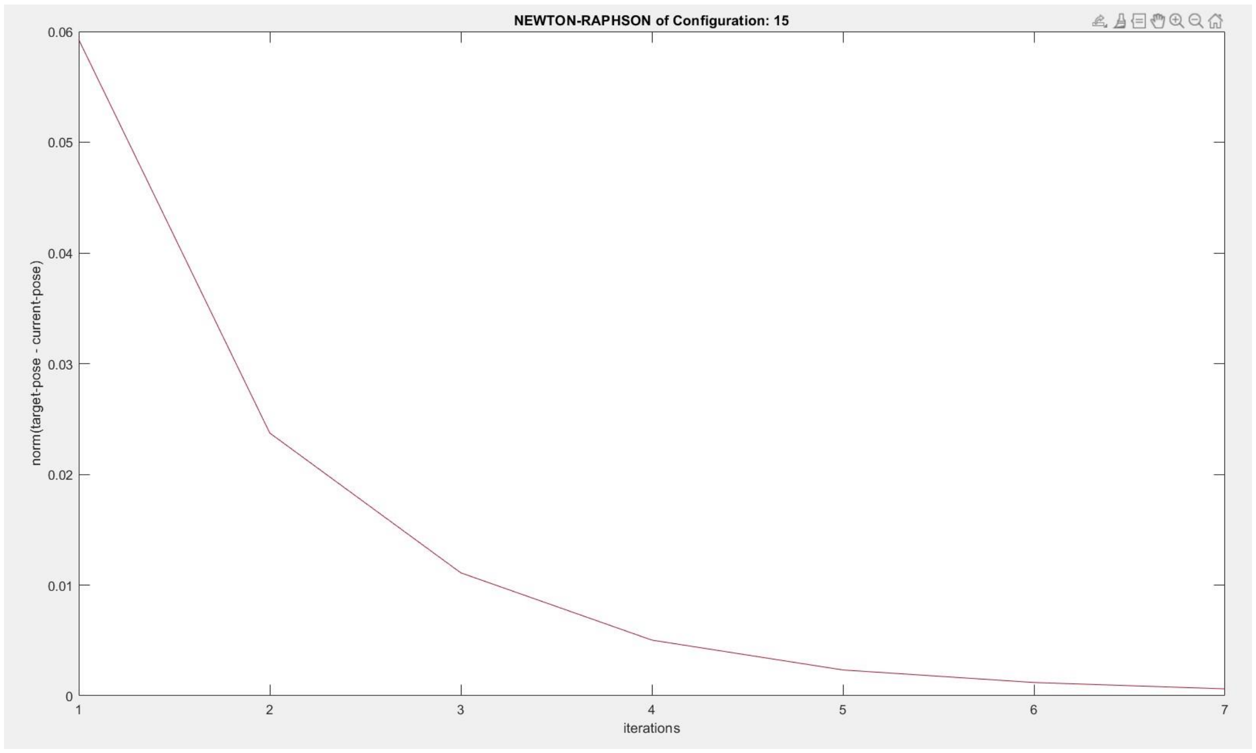
The inverse kinematics algorithm does not fail for any of the target pose configurations for this particular case (robot + initial configuration + range of desired pose.)

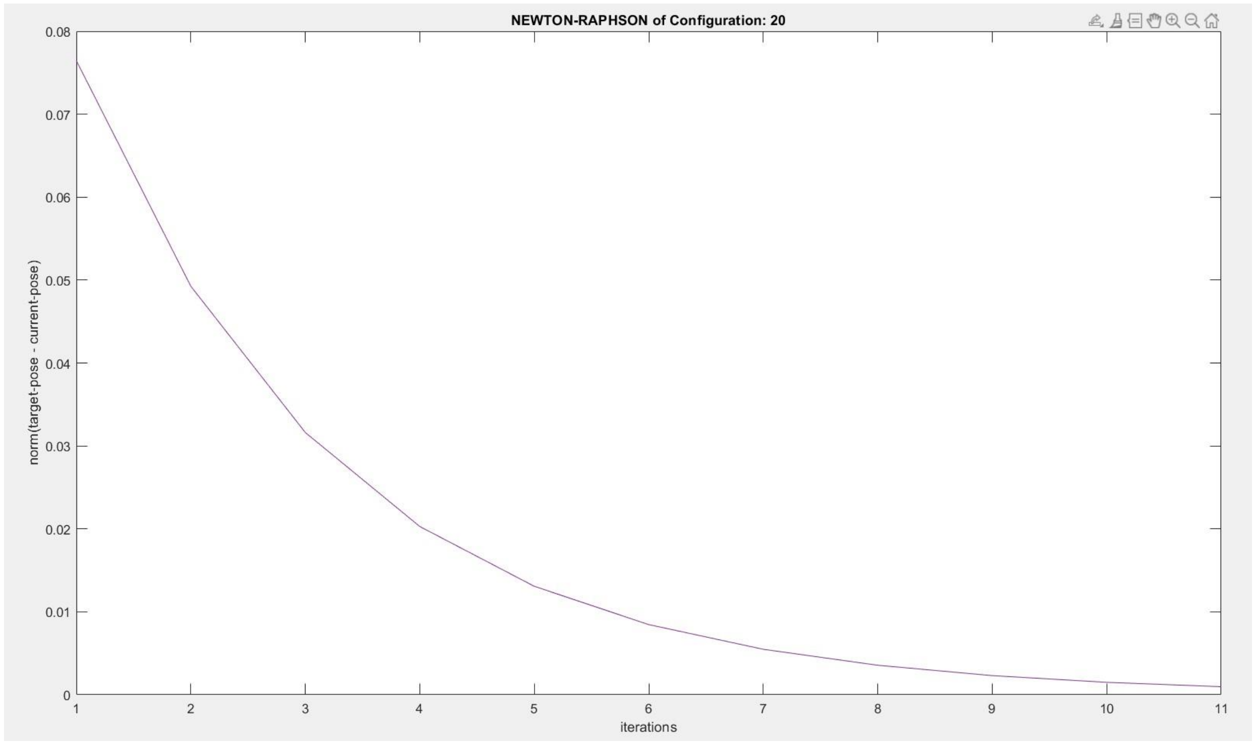
We can visualize that after seeing the graph between the error and number of iterations of each configuration of final pose.
here,

- Error is the norm(target_twist – current_twist)
- Iterations are the number of iterations of Newton-Raphson algorithm to reach the desired (target) twist.


NEWTON-RAPHSON of Configuration: 3
norm(target-pose - current-pose)
iterations
0.09
0.08
0.07
0.06
0.05
0.04
0.03
0.02
0.01
0
1
2
3
4
5
6
7
8
9
10


NEWTON-RAPHSON of Configuration: 11
norm(target-pose - current-pose)
iterations
0.07
0.06
0.05
0.04
0.03
0.02
0.01
0
1
2
3
4
5
6
7





These are the images of error V/S iterations plot for some of the configurations and we can see that for all the configurations error converges to zero eventually and thus simple Newton-Raphson method is sufficient to find inverse kinematics solution for this problem.

HOMEWORK 2

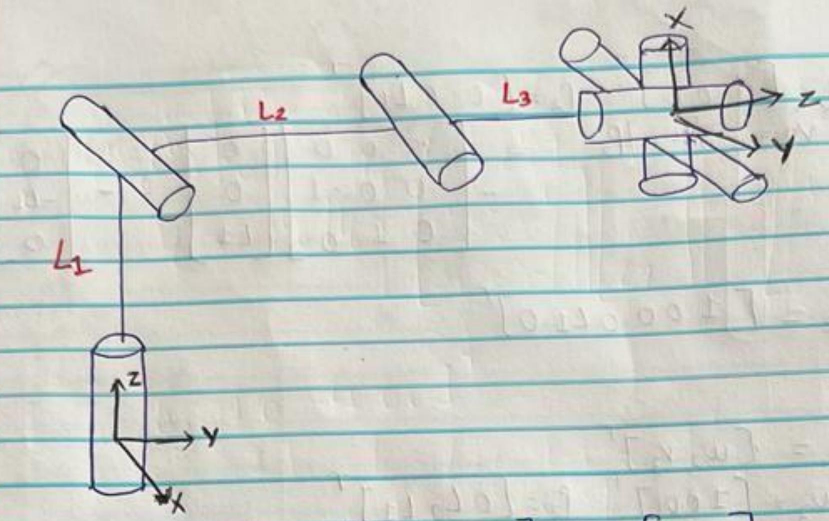
Section2:

Problem-2:

A].

The calculations for defining the home configuration (M) and the screw axis of the given 6-DOF robot are as follows:
Please take note that L1 = L2 = L3 = 0.3m.

HW2 problem 2



$$\rightarrow M = \begin{bmatrix} R & P \\ 0 & 1 \end{bmatrix} \quad R = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{bmatrix} \quad P = \begin{bmatrix} 0 \\ L_2+L_3 \\ L_1 \end{bmatrix}$$

$$M = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & L_2+L_3 \\ 1 & 0 & 0 & L_1 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\rightarrow S_1 = \quad w_1 = [0\ 0\ 1]^T \quad p_1 = [0\ 0\ 0]^T$$

$$S_1 = [0\ 0\ 1\ 0\ 0\ 0]^T$$

$$\rightarrow S_2 = [w_2\ v_2]^T$$

$$w_2 = [-1\ 0\ 0]^T \quad p_2 = [0\ 0\ L_1]$$

$$v_2 = -[w_2]p_2 = -\begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & -1 & 0 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ L_1 \end{bmatrix}$$

$$= \begin{bmatrix} 0 \\ -L_1 \\ 0 \end{bmatrix}$$

$S_2 = [-1\ 0\ 0\ 0\ -L1\ 0]^T$

$\rightarrow S_3 = [w_3\ V_3]^T$

$w_3 = [-1\ 0\ 0]^T \quad P_3 = [0\ L_2\ L_1]^T$

$$V_3 = -[w_3]P_3 = -\begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & -1 & 0 \end{bmatrix}\begin{bmatrix} 0 \\ L2 \\ L1 \end{bmatrix} = -\begin{bmatrix} 0 \\ L_1 \\ -L_2 \end{bmatrix} = \begin{bmatrix} 0 \\ -L_1 \\ L_2 \end{bmatrix}$$

$S_3 = [-1\ 0\ 0\ 0\ -L1\ L_2]^T$

$\rightarrow S_4 = [w_4\ V_4]^T$

$w_4 = [0\ 0\ 1]^T \quad P_4 = [0\ (L_2+L_3)\ 0]^T$

$$V_4 = -[w_4]P_4 = -\begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}\begin{bmatrix} 0 \\ L_2+L_3 \\ 0 \end{bmatrix} = -\begin{bmatrix} -(L_2+L_3) \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} L_2+L_3 \\ 0 \\ 0 \end{bmatrix}$$

$S_4 = [0\ 0\ 1\ (L_2+L_3)\ 0\ 0]^T$

$\rightarrow S_5 = [w_5\ V_5]^T$

$w_5 = [-1\ 0\ 0] \quad P_5 = [0\ (L_2+L_3)\ L1]$

$S_5 = [-1\ 0\ 0\ 0\ -L1\ (L_2+L_3)]$

$$V_5 = -[w_5]P_5 = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & -1 & 0 \end{bmatrix}\begin{bmatrix} 0 \\ L_2+L_3 \\ L1 \end{bmatrix} = \begin{bmatrix} 0 \\ -L1 \\ (L_2+L_3) \end{bmatrix}$$

$\longrightarrow$ $S_6 = [\omega_6 \; V_6]^T$

$\omega_6 = [0 \; 1 \; 0]^T \qquad P_6 = [0 \; 0 \; L1]$

$$V_6 = -[\omega_6] P_6 = -\begin{bmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \\ -1 & 0 & 0 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ L1 \end{bmatrix} = \begin{bmatrix} -L1 \\ 0 \\ 0 \end{bmatrix}$$

$S_6 = [0 \; 1 \; 0 \; -L1 \; 0 \; 0]^T$

B].

I have calculated the homogenous transformation matrix of the home configuration (M) and used it to calculate the forward kinematics of the robot along with the “fkine” function developed for section-1 of this homework assignment.
**The code for the fkine function can be found in the fkine.m file of the submission.

C].

I calculated the space jacobian using the jacob0 function developed for the section-1 of this homework assignment.
**The code for the jacob0 function can be found in the jacob.m file of the submission.

D].

I implemented NEWTON-RAPHSON numerical method to find the inverse kinematic solution for the given robotic manipulator.
It is iteratively implemented using the following equation -

deltaQ = pinv(J)*(target_twist – current_twist)

here,

- deltaQ is a vector whose elements represent the change in joint values from the previous joint values.
- pinv(J) is Moore-Penrose pseudo-inverse of the jacobian matrix.
- target_twist is twist of the final desire pose of the end effector.
- Current_twist is the twist of the current pose (at current iteration) of the end effector.

This inverse kinematics algorithm fails for many of the target pose configurations for this particular case (robot + initial configuration + range of desired pose.)

We can visualize that after seeing the graph between the error and number of iterations of each configuration of final pose as shown in figure 2.1.
here,

- Error is the norm(target_twist – current_twist)
- Iterations are the number of iterations of Newton-Raphson algorithm to reach the desired (target) twist.

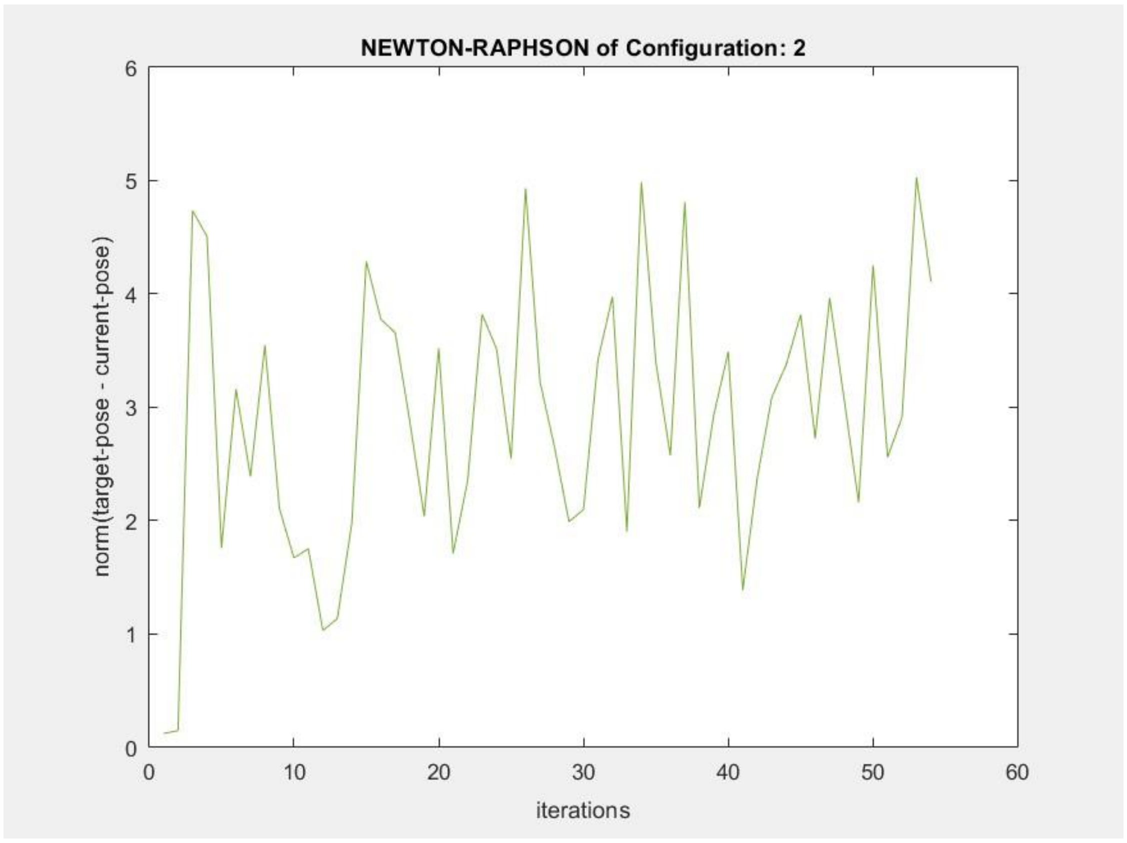
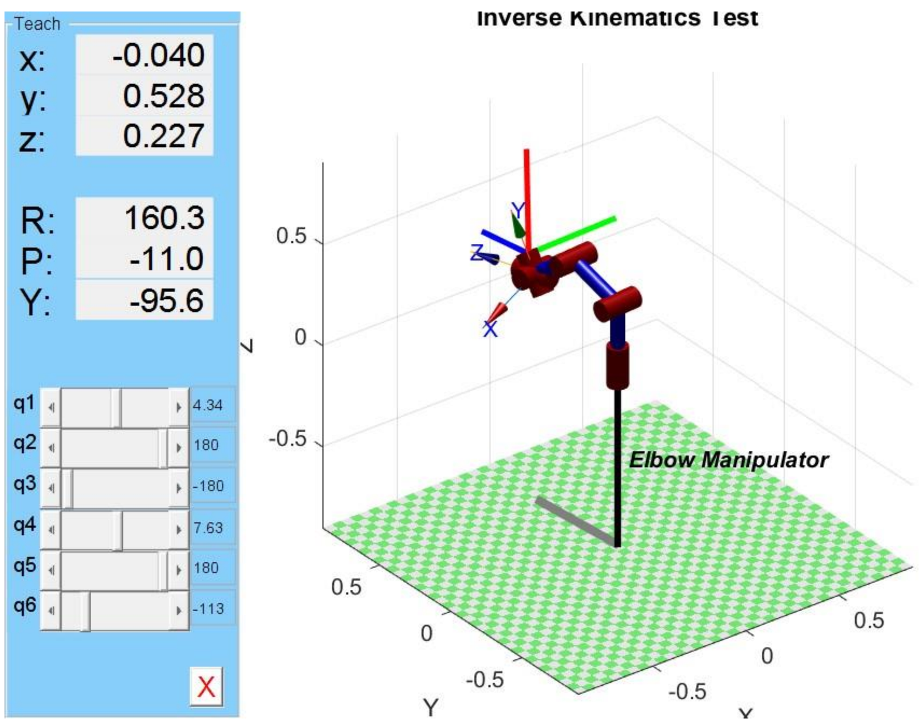


Fig 2.1

The error keeps on increasing and decreasing and does not converge towards zero.

We can also visualize this by seeing the robot's motion in Fig 2.2 and Fig 2.3, despite the target pose being very close to the current pose of the robot, it does not reach there and overshoots to a very far away position.

This implies that the current pose of the robot at the beginning is in singularity.

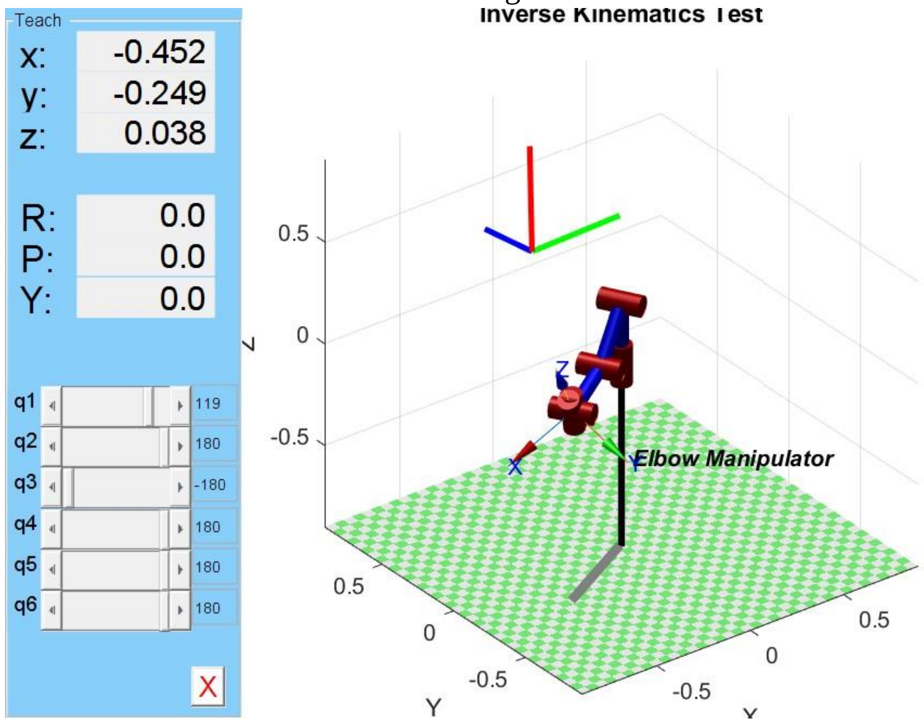


x: -0.040
y: 0.528
z: 0.227

R: 160.3
P: -11.0
Y: -95.6

q1 4.34
q2 180
q3 -180
q4 7.63
q5 180
q6 -113

Fig 2.2



x: -0.452
y: -0.249
z: 0.038

R: 0.0
P: 0.0
Y: 0.0

q1 119
q2 180
q3 -180
q4 180
q5 180
q6 180

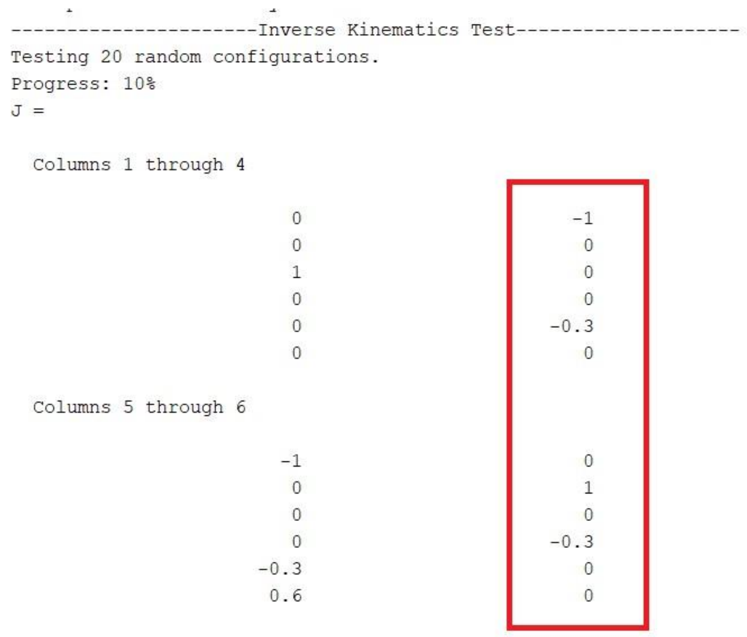
Fig 2.3

After seeing the above images, I began monitoring the jacobians used in each iteration and found out that this problem arises because of the ill-conditioned jacobians that occur during the calculations of Newton-Raphson method.

Drop in the rank (two or more linearly dependent columns) of the jacobians leads to abruptly high joint values and hence this justifies my observations.

Few of the ill-conditioned jacobians that occur during while finding the inverse kinematics solution are as follows –

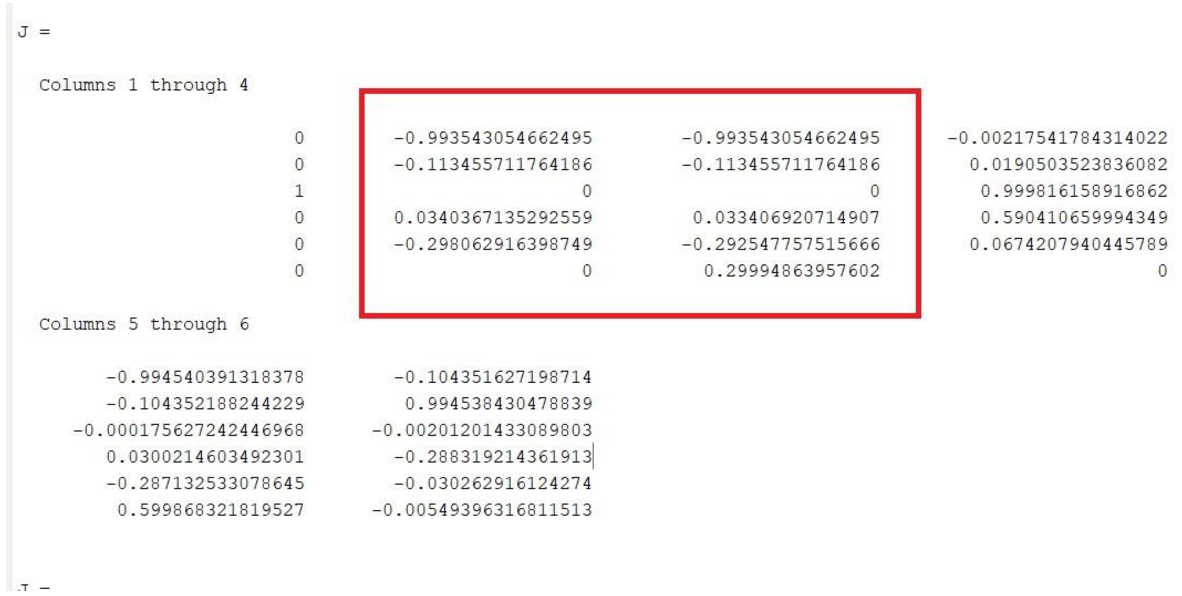
The first jacobian is as follows -



```
----------------------Inverse Kinematics Test--------------------
Testing 20 random configurations.
Progress: 10%
J =

  Columns 1 through 4

                        0                        -1                        -1                         0
                        0                         0                         0                         0
                        1                         0                         0                         1
                        0                         0                         0                       0.6
                        0                      -0.3                      -0.3                         0
                        0                         0                       0.3                         0

  Columns 5 through 6

                       -1                         0
                        0                         1
                        0                         0
                        0                      -0.3
                     -0.3                         0
                      0.6                         0
```

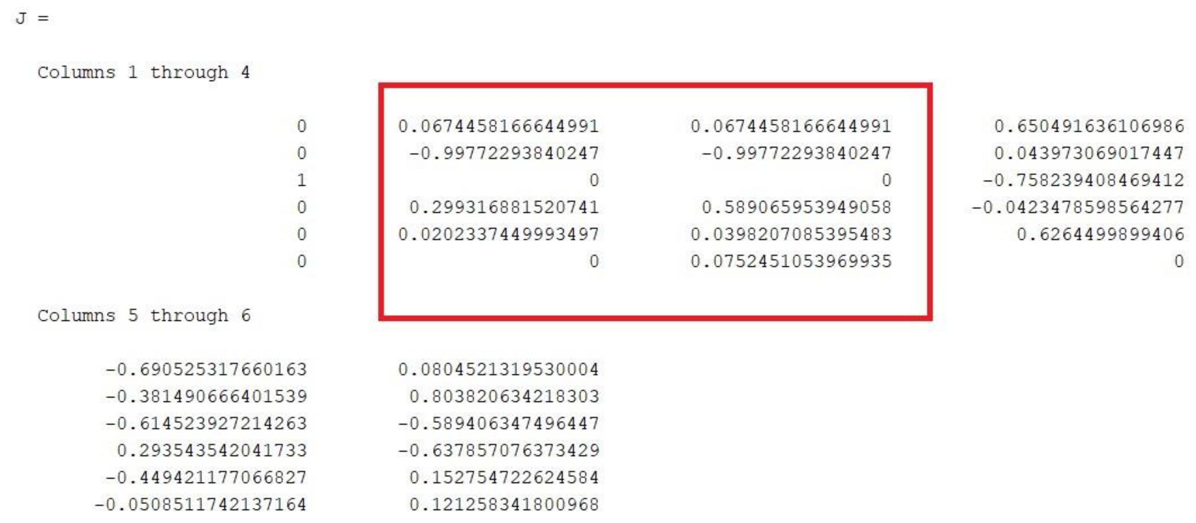
Fig 2.4

This shows that the current pose of the robot (at the start) is in singularity. As two columns of the jacobian matrix are linearly dependent and hence the rank of the jacobian is less than N. Where, N is the number of joints or DOFs of the robot.



```
J =

  Columns 1 through 4

                        0        -0.993543054662495        -0.993543054662495      -0.00217541784314022
                        0        -0.113455711764186        -0.113455711764186       0.0190503523836082
                        1                         0                         0        0.999816158916862
                        0        0.0340367135292559         0.033406920714907        0.590410659994349
                        0        -0.298062916398749        -0.292547757515666       0.0674207940445789
                        0                         0          0.29994863957602                         0

  Columns 5 through 6

       -0.994540391318378        -0.104351627198714
       -0.104352188244229         0.994538430478839
    -0.000175627242446968      -0.00201201433089803
       0.0300214603492301        -0.288319214361913
       -0.287132533078645       -0.0302629161224274
        0.599868321819527      -0.00549396316811513
```

Fig 2.5



```
J =

  Columns 1 through 4

                   0     0.0674458166644991     0.0674458166644991      0.650491636106986
                   0     -0.99772293840247      -0.99772293840247       0.043973069017447
                   1                     0                      0      -0.758239408469412
                   0     0.299316881520741      0.589065953949058     -0.0423478598564277
                   0     0.0202337449993497     0.0398207085395483       0.6264499899406
                   0                     0      0.0752451053969935                     0

  Columns 5 through 6

   -0.690525317660163     0.0804521319530004
   -0.381490666401539      0.803820634218303
   -0.614523927214263     -0.589406347496447
    0.293543542041733     -0.637857076373429
   -0.449421177066827      0.152754722624584
  -0.0508511742137164      0.121258341800968
```

Fig 2.6

The two columns here become almost linearly dependent multiple times during the calculations and gives out large joint variables after the calculations of Newton-Raphson equations and hence the solution is never attained.

Therefore, to tackle this problem, I implemented the Damped-Least-Square modification of Newton-Raphson numerical method to find out the deltaQ and update it iteratively. J* = transpose(J)*pinv(J*transpose(J) + (lamda)^2*I) deltaQ = (J*)*(target_twist – current_twist)

here,

- deltaQ is a vector whose elements represent the change in joint values from the previous joint values.
- pinv(k) is Moore-Penrose pseudo-inverse of the matrix k.
- target_twist is twist of the final desire pose of the end effector.
- Current_twist is the twist of the current pose (at current iteration) of the end effector.
- lambda is the damping factor.
- I is 6*6 Identity matrix

After trial-and-error I found lamda = 2 to best fit the solutions and the error smoothly converges to zero when we use this lambda for all the configurations of the target pose. This is depicted by the error V/S iterations graphs as shown below.

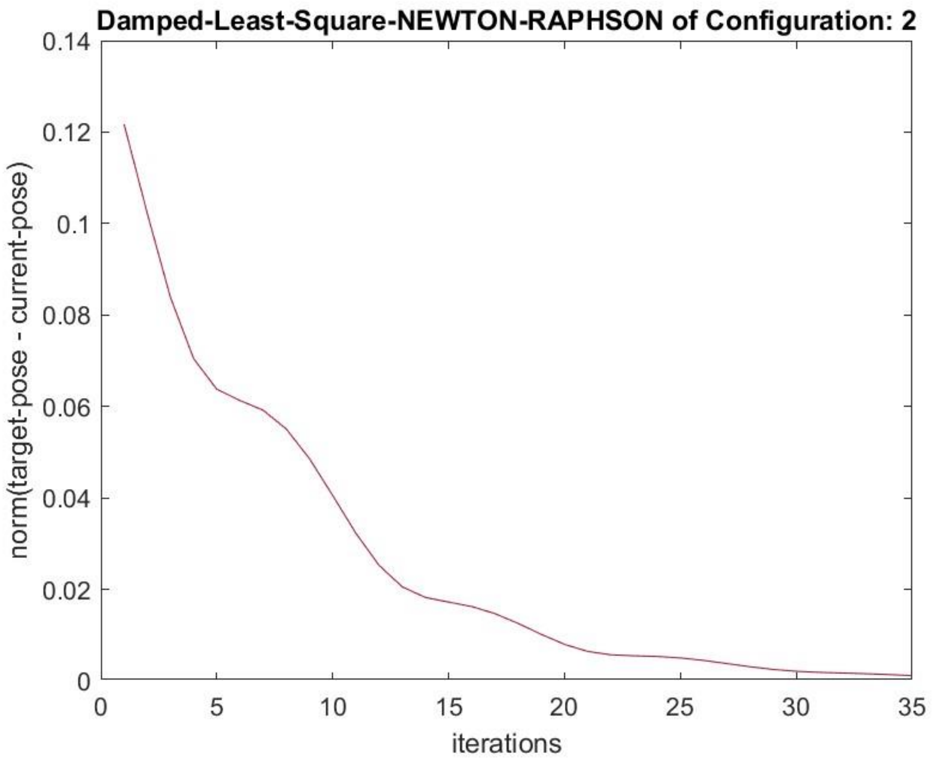

Damped-Least-Square-NEWTON-RAPHSON of Configuration: 2
norm(target-pose - current-pose)
iterations
0
0.02
0.04
0.06
0.08
0.1
0.12
0.14
0
5
10
15
20
25
30
35

Fig 2.7

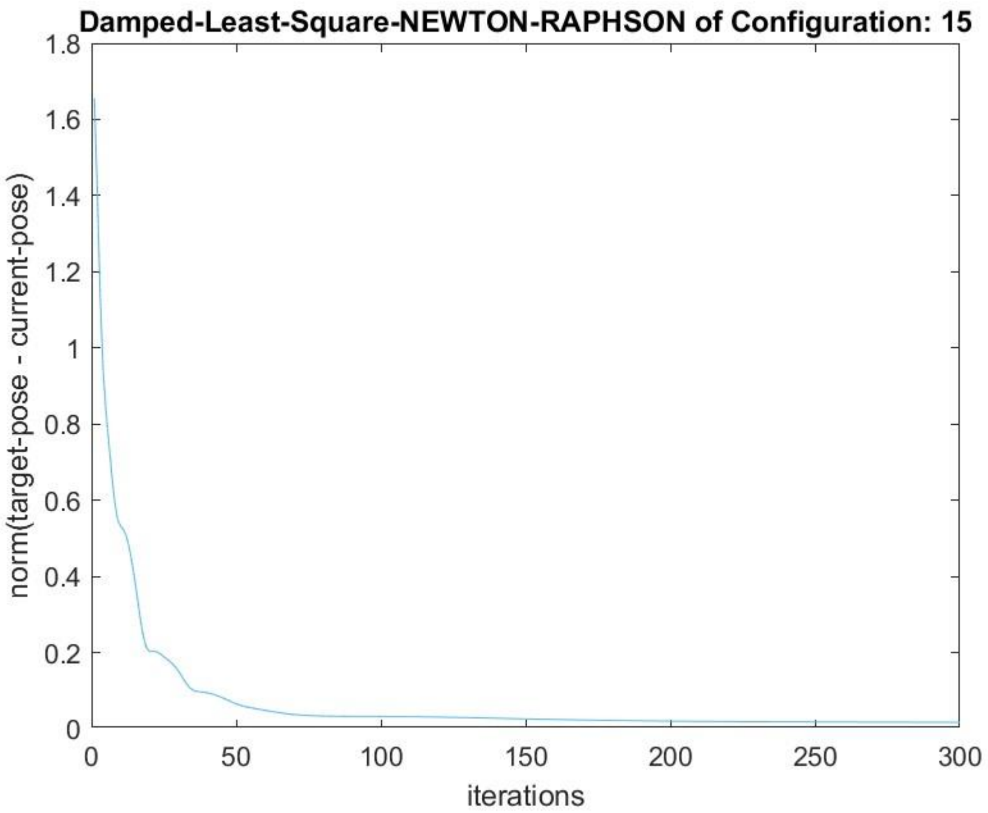

Damped-Least-Square-NEWTON-RAPHSON of Configuration: 15
norm(target-pose - current-pose)
iterations
0
0.2
0.4
0.6
0.8
1
1.2
1.4
1.6
1.8
0
50
100
150
200
250
300

Fig 2.8

Therefore, we can observe that The Damped-Least-Square method solves the inverse kinematics problem for the give case (robot + initial configuration + range of desired pose.)

** Here, I chose the range of the desired (target) pose after keeping in mind that even the DampedLeast-Squared version of Newton-Raphson largely depends on the initial condition (twist) of the robot to reach a target pose (target twist) and cannot find solution for all the combination of the initial and final twist.

HOMEWORK 2

Section2:

Problem-3:

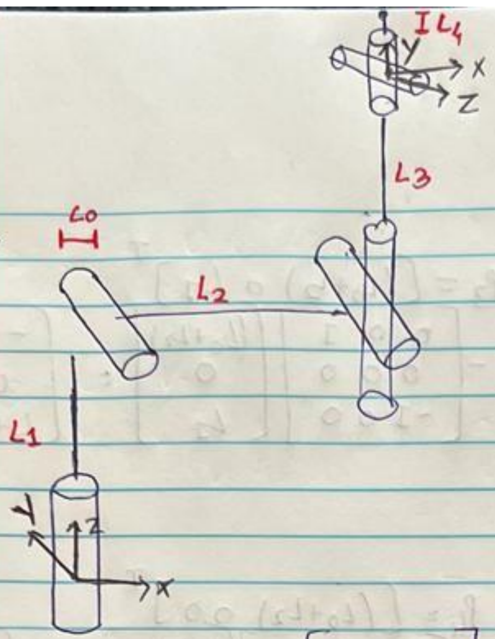
A].

The calculations for defining the home configuration (M) and the screw axis of the given 6-DOF ABB IRB140 robot are as follows:

Please take note that

L0 = 70mm
L1 = 352mm
L2 = 360mm
L3 = 380mm
L4 = 65mm

HW2 problem3 →



→ $M = \begin{bmatrix} R & P \\ 0 & 1 \end{bmatrix}$ $R = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & 1 & 0 \end{bmatrix}$ $P = \begin{bmatrix} L_0 + L_2 \\ 0 \\ L_1 + L_3 + L_4 \end{bmatrix}$

$$M = \begin{bmatrix} 1 & 0 & 0 & (L_0 + L_2) \\ 0 & 0 & -1 & 0 \\ 0 & 1 & 0 & (L_1 + L_3 + L_4) \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

→ $S_1 = [w_1 \ v_1]^T$

$w_1 = [0 \ 0 \ 1]^T$ $p_1 = [0 \ 0 \ 0]^T$ $S_1 \to [0 \ 0 \ 1 \ 0 \ 0 \ 0]^T$

→ $S_2 = [w_2 \ v_2]^T$

$w_2 = [0 \ 1 \ 0]^T$ $P_2 = [L_0 \ 0 \ L_1]$

$V_2 = -[w_2] P_2 = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \\ -1 & 0 & 0 \end{bmatrix} \begin{bmatrix} L_0 \\ 0 \\ L_1 \end{bmatrix} = -\begin{bmatrix} L_1 \\ 0 \\ -L_0 \end{bmatrix} = \begin{bmatrix} -L_1 \\ 0 \\ L_0 \end{bmatrix}$

$S_2 = [0 \ 1 \ 0 \ -L1 \ 0 \ L_0]^T$

$\rightarrow S_3 = [w_3 \ v_3]^T$

$w_3 = [0 \ 1 \ 0]^T \quad p_3 = [(L_0+L_2) \ 0 \ L_1]^T$

$$v_3 = -[w_3]p_3 = -\begin{bmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \\ -1 & 0 & 0 \end{bmatrix}\begin{bmatrix} (L_0+L_2) \\ 0 \\ L_1 \end{bmatrix} = \begin{bmatrix} -L_1 \\ 0 \\ L_0+L_2 \end{bmatrix}$$

$\rightarrow S_4 = [w_4 \ v_4]^T$

$w_4 = [0 \ 0 \ 1]^T \quad p_4 = [(L_0+L_2) \ 0 \ 0]^T$

$$v_4 = -[w_4]p_4 = -\begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}\begin{bmatrix} L_0+L_2 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ -(L_0+L_2) \\ 0 \end{bmatrix}$$

$S_4 = [0 \ 0 \ 1 \ 0 \ -(L_0+L_2) \ 0]^T$

$\rightarrow S_5 = [w_5 \ v_5]^T$

$w_5 = [0 \ 1 \ 0]^T \quad p_5 = [(L_0+L_2) \ 0 \ (L_1+L_3)]^T$

$$v_5 = -[w_5]p_5 = -\begin{bmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \\ -1 & 0 & 0 \end{bmatrix}\begin{bmatrix} L_0+L_2 \\ 0 \\ L_1+L_3 \end{bmatrix} = \begin{bmatrix} -(L_1+L_3) \\ 0 \\ (L_0+L_2) \end{bmatrix}$$

$S_5 = [0 \ 1 \ 0 \ -(L_1+L_3) \ 0 \ (L_0+L_2)]^T$

$\rightarrow S_6 = [w_6 \ v_6]^T$

$w_6 = [0 \ 0 \ 1]^T \quad p_6 = [(L_0+L_2) \ 0 \ 0]^T$

$$v_6 = -[w_6]p_6 = -\begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}\begin{bmatrix} L_0+L_2 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ -(L_0+L_2) \\ 0 \end{bmatrix}$$

B].

I have calculated the homogenous transformation matrix of the home configuration (M) and used it to calculate the forward kinematics of the robot along with the "fkine" function developed for section-1 of this homework assignment.
**The code for the fkine function can be found in the fkine.m file of the submission.

C].

I calculated the space jacobian using the jacob0 function developed for the section-1 of this homework assignment.
**The code for the jacob0 function can be found in the jacob.m file of the submission.

D].

I implemented NEWTON-RAPHSON numerical method to find the inverse kinematic solution for the given robotic manipulator.
It is iteratively implemented using the following equation -

deltaQ = pinv(J)*(target_twist – current_twist)

here,
- deltaQ is a vector whose elements represent the change in joint values from the previous joint values.
- pinv(J) is Moore-Penrose pseudo-inverse of the jacobian matrix.
- target_twist is twist of the final desire pose of the end effector.
- Current_twist is the twist of the current pose (at current iteration) of the end effector.

This inverse kinematics algorithm fails for many of the configurations for this particular case (robot + initial configuration + range of desired pose.)

We can visualize that after seeing the graph between the error and number of iterations of each configuration of final pose as shown in figure 3.1.
here,
- Error is the norm(target_twist – current_twist)
- Iterations are the number of iterations of Newton-Raphson algorithm to reach the desired (target) twist.

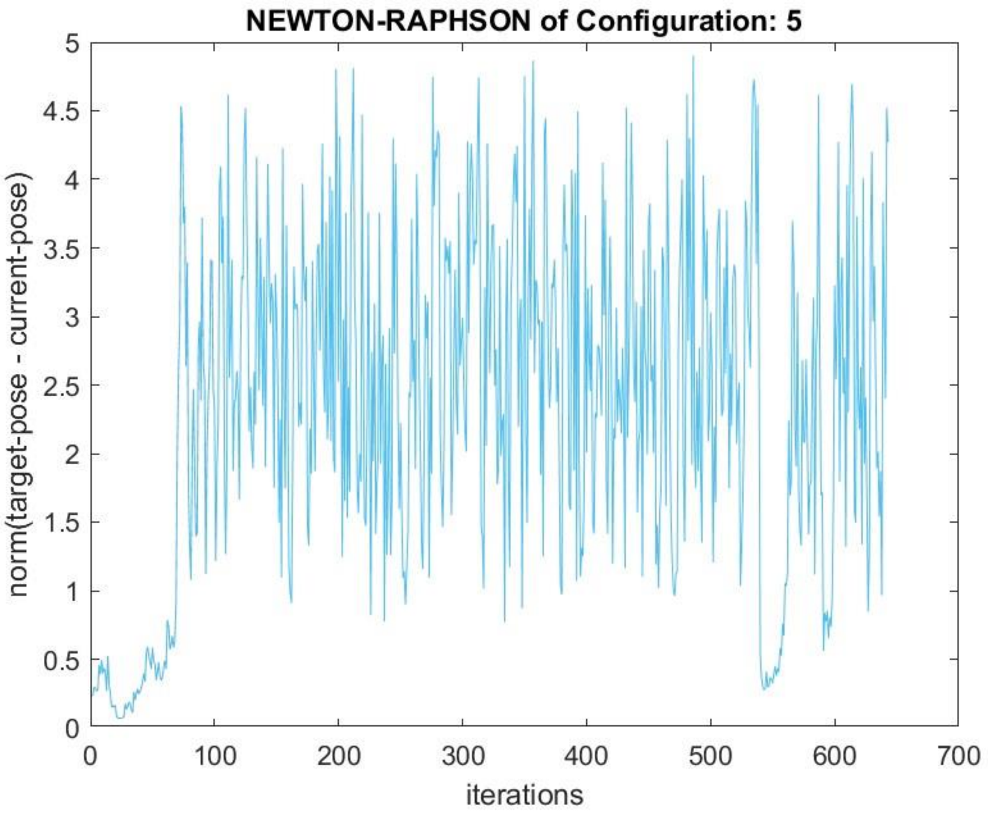


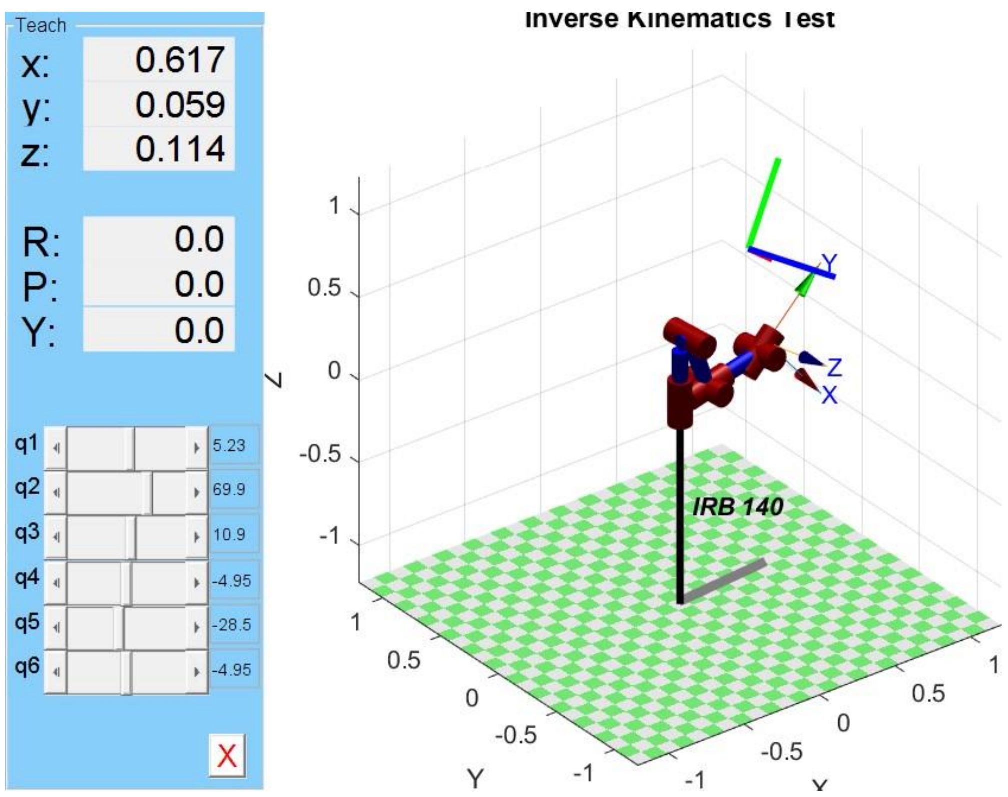
Fig 3.1

The error keeps on increasing and decreasing and does not converge towards zero.

We can also visualize this by seeing the robot's motion in Fig 3.2 and Fig 3.3, despite the target pose being very close to the current pose of the robot it does not reach there and overshoots to a very far away position.

This implies that the current pose of the robot at the beginning of its endeavor to reach the 5th target configuration is in singularity.

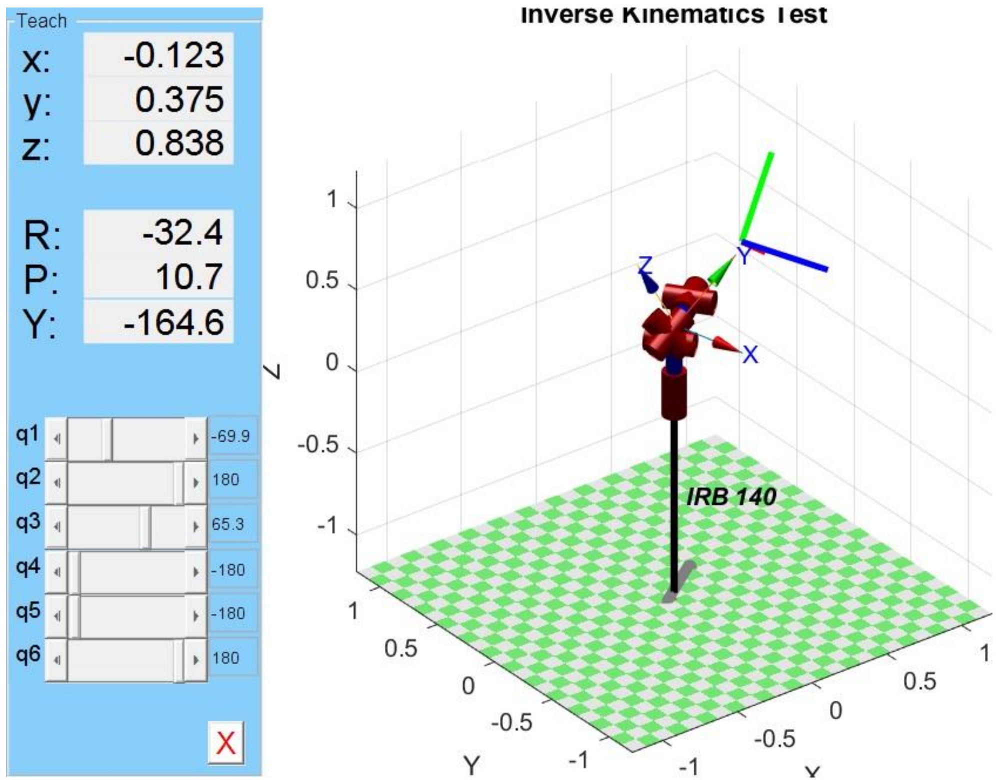
This also implies that the current pose of the robot at the beginning (home configuration) is not in singularity as above observation was not made there.


Teach

x: 0.617
y: 0.059
z: 0.114

R: 0.0
P: 0.0
Y: 0.0

q1 5.23
q2 69.9
q3 10.9
q4 -4.95
q5 -28.5
q6 -4.95


Teach

x: -0.123
y: 0.375
z: 0.838

R: -32.4
P: 10.7
Y: -164.6

q1 -69.9
q2 180
q3 65.3
q4 -180
q5 -180
q6 180

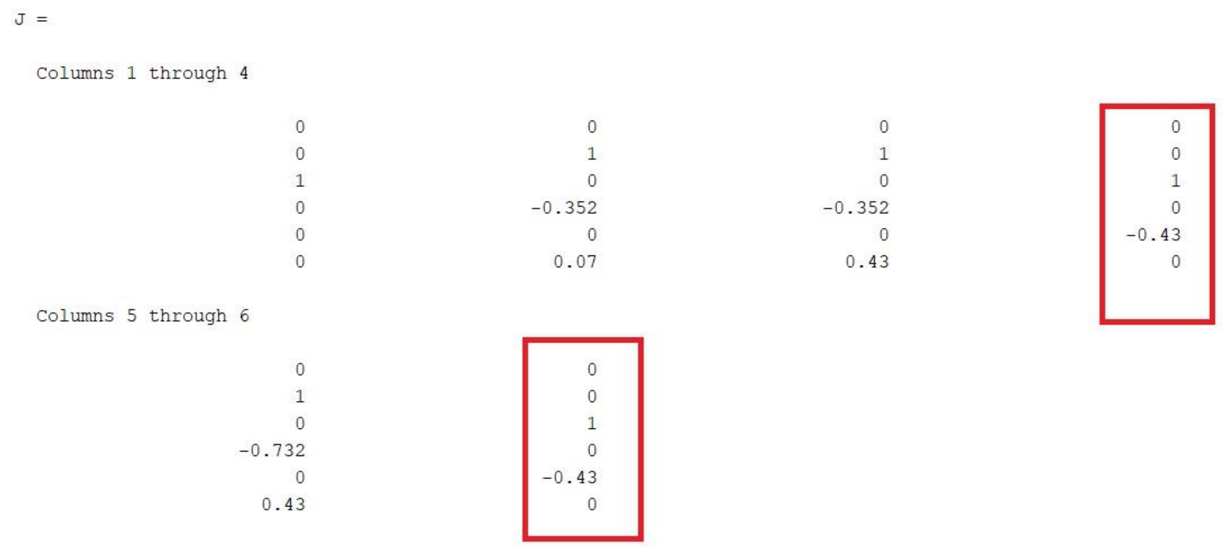
Fig 3.2 and 3.3

After seeing the above images, I began monitoring the jacobians used in each iteration and found out that this problem arises because of the ill-conditioned jacobians that occur during the calculations of Newton-Raphson method.

Drop in the rank (two or more linearly dependent columns) of the jacobians leads to abruptly high joint values and hence this justifies my observations.

Few of the ill-conditioned jacobians that occur during while finding the inverse kinematics solution are as follows –

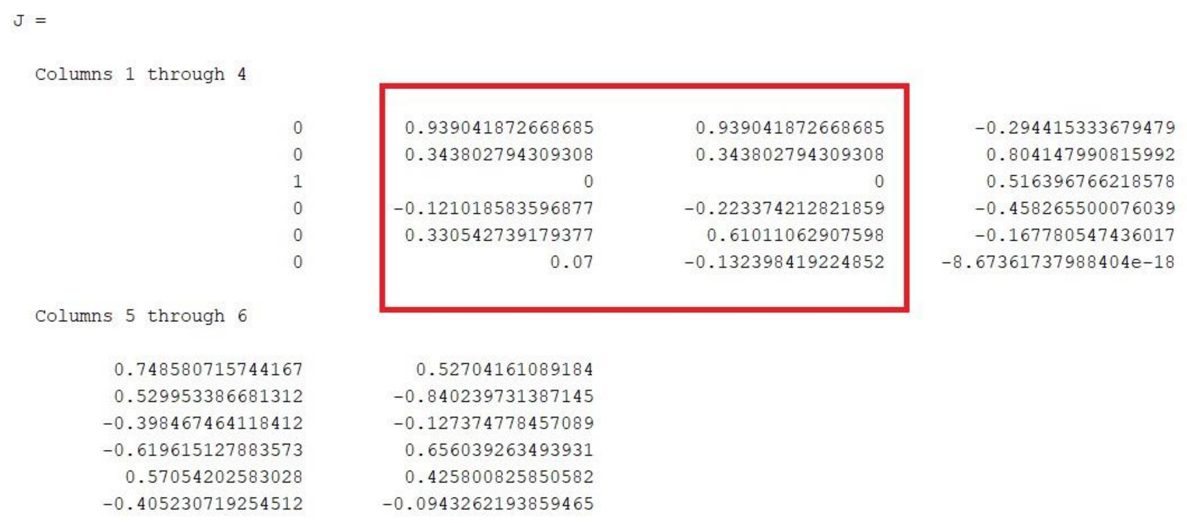
1st jacobian for attaining the 5th target pose is as follows -



```
J =

  Columns 1 through 4

                        0                         0                         0                         0
                        0                         1                         1                         0
                        1                         0                         0                         1
                        0                    -0.352                    -0.352                         0
                        0                         0                         0                     -0.43
                        0                      0.07                      0.43                         0

  Columns 5 through 6

                        0                         0
                        1                         0
                        0                         1
                   -0.732                         0
                        0                     -0.43
                     0.43                         0
```

Fig 3.4

This shows that the current pose of the robot (at the start of its endeavor for target pose 5) is in singularity. As two columns of the jacobian matrix are linearly dependent and hence the rank of the jacobian is less than N. Where, N is the number of joints or DOFs of the robot.



```
J =

  Columns 1 through 4

                     0    0.939041872668685      0.939041872668685     -0.294415333679479
                     0    0.343802794309308      0.343802794309308      0.804147990815992
                     1                    0                      0      0.516396766218578
                     0   -0.121018583596877     -0.223374212821859     -0.458265500076039
                     0    0.330542739179377       0.61011062907598     -0.167780547436017
                     0                 0.07     -0.132398419224852   -8.67361737988404e-18

  Columns 5 through 6

     0.748580715744167     0.52704161089184
     0.529953386681312   -0.840239731387145
    -0.398467464118412   -0.127374778457089
    -0.619615127883573    0.656039263493931
      0.57054202583028    0.425800825850582
    -0.405230719254512  -0.0943262193859465
```

Fig 3.5

The two columns here become almost linearly dependent multiple times during the calculations and gives out huge values of joint variables, and hence the solution is never attained.

Therefore, to tackle this problem, I implemented the Damped-Least-Square modification of Newton-Raphson numerical method to find out the deltaQ and update it iteratively. J* = transpose(J)*pinv(J*transpose(J) + (lamda)^2*I) deltaQ = (J*)*(target_twist – current_twist)

here,

- deltaQ is a vector whose elements represent the change in joint values from the previous joint values.
- pinv(k) is Moore-Penrose pseudo-inverse of the matrix k.
- target_twist is twist of the final desire pose of the end effector.
- Current_twist is the twist of the current pose (at current iteration) of the end effector.
- lambda is the damping factor.
- I is 6*6 Identity matrix

After trial-and-error I found lamda = 2 to best fit the solutions and the error smoothly converges to zero when we use this lambda for all the configurations of the target pose. This is depicted by the error V/S iterations graphs as shown below.

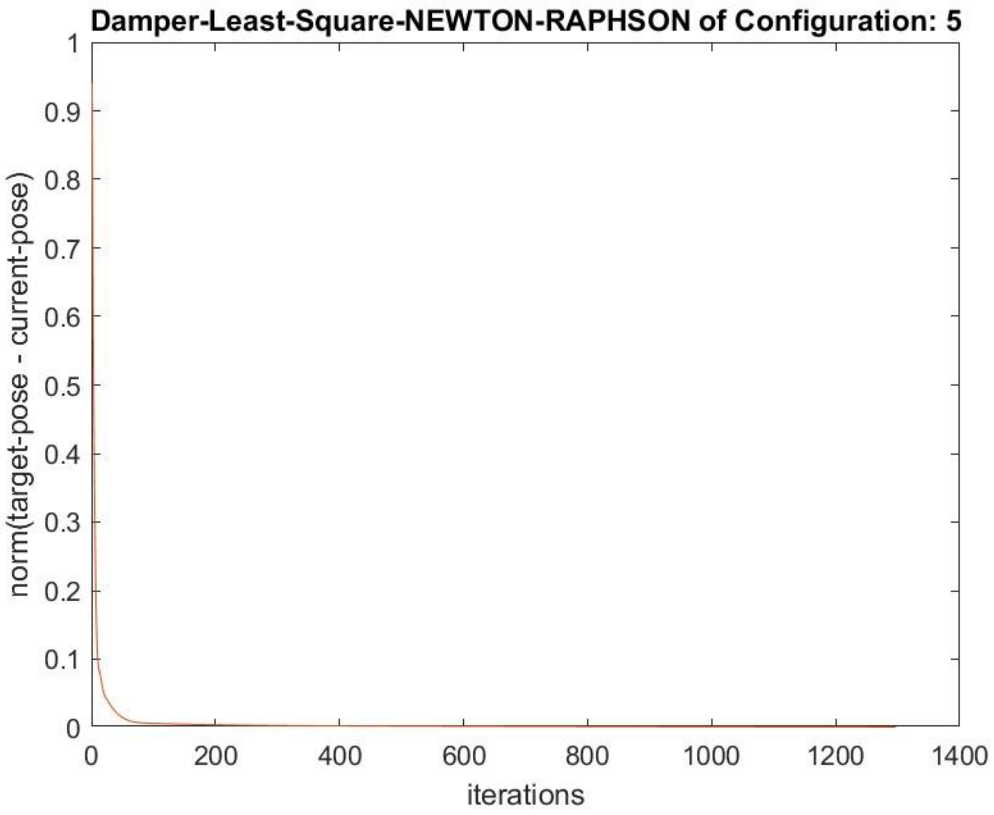


Fig 3.6

Therefore, we can observe that The Damped-Least-Square method solves the inverse kinematics problem for the give case (robot + initial configuration + range of desired pose.)

** Here, I chose the range of the desired (target) pose after keeping in mind that even the DampedLeast-Squared version of Newton-Raphson largely depends on the initial condition (twist) of the robot to reach a target pose (target twist) and cannot find solution for all the combination of the initial and final twist.