Different Performance Indices of 4R and RPRP Manipulator

Developed these indices for spiral trajectory, the start point is from the outer end of the spiral and thereon, it moves inwards.

1. **Manipulability Index**

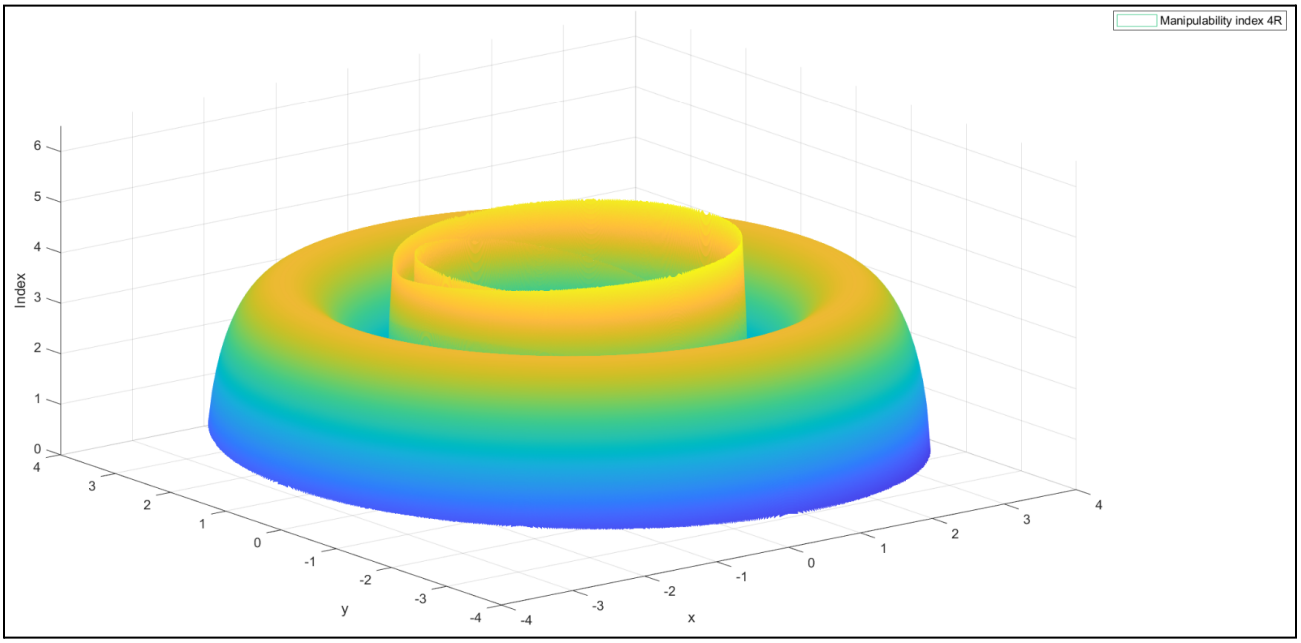
   The manipulability index in robotics is a measure of how well a robotic manipulator can control its end effector in different directions and orientations. It is a scalar value that quantifies the sensitivity of the robot's motion to small changes in its joint configurations.

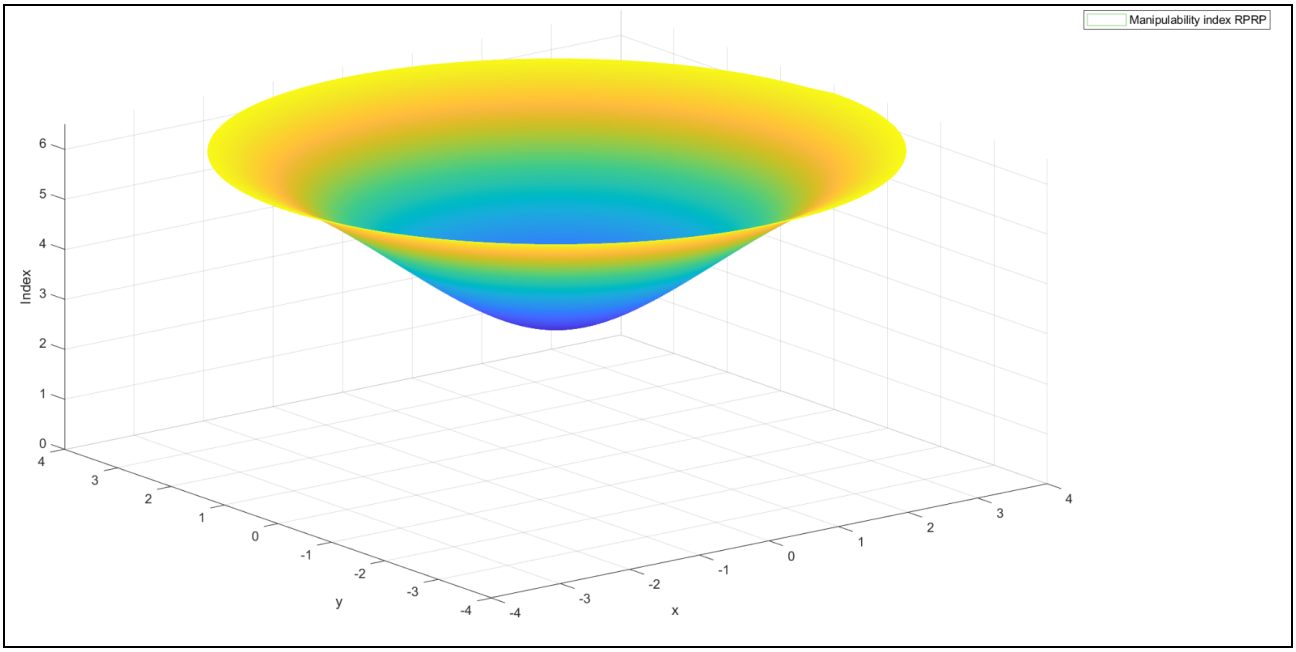
   The manipulability index takes into account the geometry of the robot, the configuration of its joints, and the position and orientation of its end effector. It is calculated based on the Jacobian matrix.

   $$\mu(\boldsymbol{\theta}) = \sqrt{\det(\mathbf{J}\mathbf{J}^T)}.$$

   where J is Jacobian

   A high manipulability index means that the robot can move its end effector in many different directions and orientations, which is desirable for tasks that require a high degree of dexterity. On the other hand, a low manipulability index means that the robot is limited in its ability to reach certain positions or orientations, which can be a disadvantage for certain applications.





Observations: For the 4R manipulator, the manipulability index starts at zero, indicating that the manipulator is unable to move in certain directions due to the configuration of the joints. As the manipulator moves along the trajectory (inwards), the manipulability index increases, indicating that the manipulator becomes more dexterous and able to move in a different direction. However, the index eventually reaches a peak and then starts to decrease, indicating that the manipulator becomes less dexterous as it approaches certain configurations where it is more difficult to move. Finally, the index increases again, indicating that the manipulator becomes more dexterous once it moves past these difficult configurations.

For the RPRP manipulator, the manipulability index initially starts at a peak, indicating that the manipulator is highly dexterous and able to move in a wide range of directions. However, as the manipulator moves along the trajectory, the manipulability index gradually decreases to a minimum point, indicating that the manipulator becomes less dexterous and more constrained in its motion. The index never reaches zero, which may indicate that the manipulator always has some degree of freedom, but that it is limited compared to its initial state. Therefore

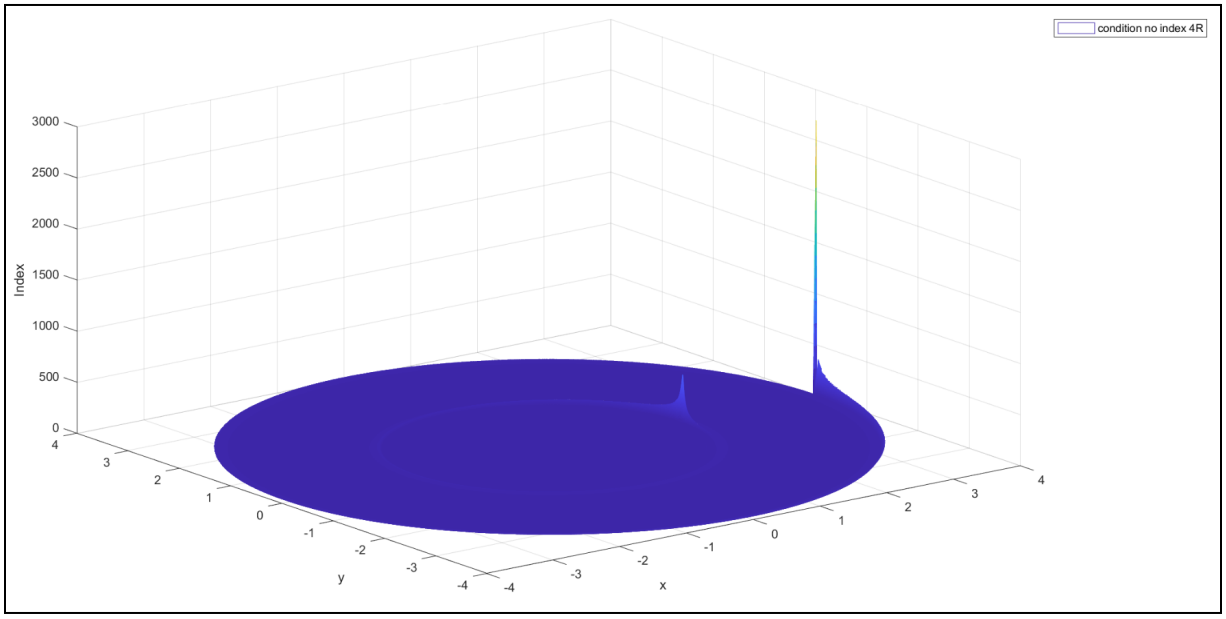
2. Condition Number

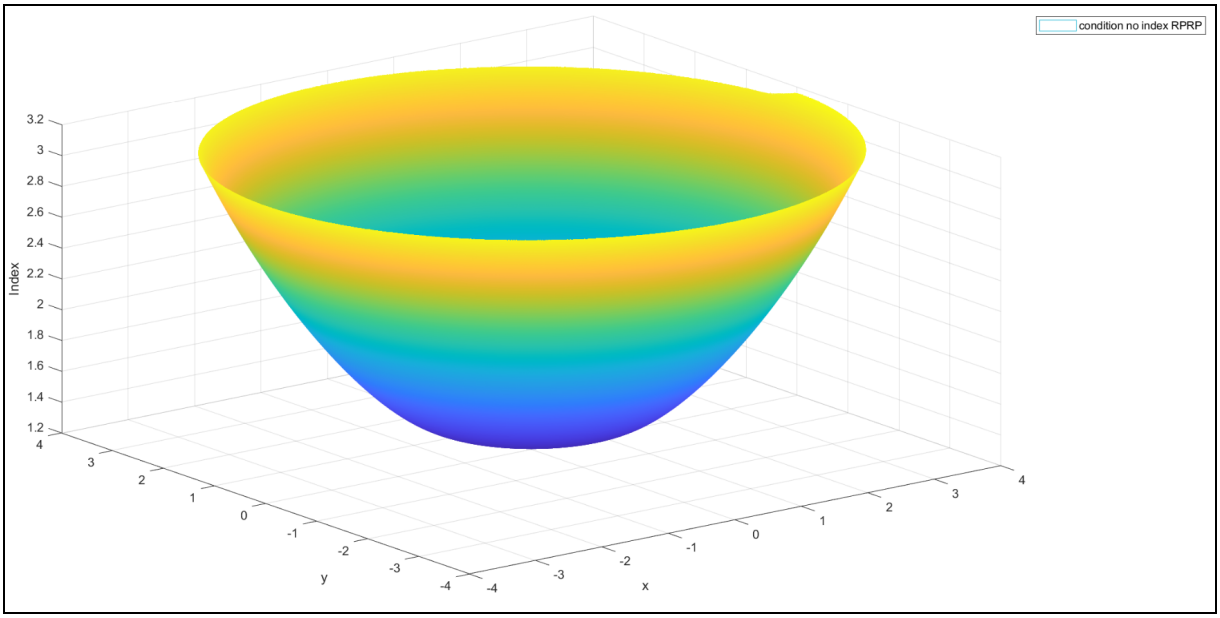
The condition number is a mathematical measure that describes the sensitivity of a system of linear equations to perturbations or changes in its coefficients. In the context of robotics, the condition number of the Jacobian matrix is a measure of the robot's ability to move its end-effector in different directions and at different speeds, while taking into account the limits on joint velocities and joint positions.

$$\kappa = \|\boldsymbol{J}\|_2 \left\|\boldsymbol{J}^{-1}\right\|_2$$

where J is Jacobian

The condition number is calculated based on the norm of the Jacobian matrix where norm signifies how much an end-effector velocity changes on small perturbations in the joint velocity. A high condition number indicates that the Jacobian matrix is ill-conditioned, meaning that small changes in the joint velocity can lead to large changes in the end effector velocity. This can result in instability or poor performance of the robot in certain configurations.Overall, a high condition number in robotics is an indication that the robot may have limited dexterity, limited workspace, and less stable control.





Observations: In the graph for 4R manipulator, at the start of the trajectory we get a comparatively higher value of condition number index than the rest of the values in the graph. The rest of the values are very close to each other (up to 2-10) and do not fluctuate much. The values drop to a great extent after some of the starting values. We can see a small rise in between the graphs too.

In the graph for RPRP manipulator, at the start of the trajectory the value of condition no. index is at its peak at a value slightly lower than 3.2 and then the values decrease gradually and attain a certain minimum value of 1.2.
Overall, the graphs indicate that the RPRP manipulator has a more stable and consistent performance in terms of condition number index (less value) over the trajectory compared to the 4R manipulator, which experiences more fluctuations and a higher initial spike in the index.

3. **Local Conditioning Index**

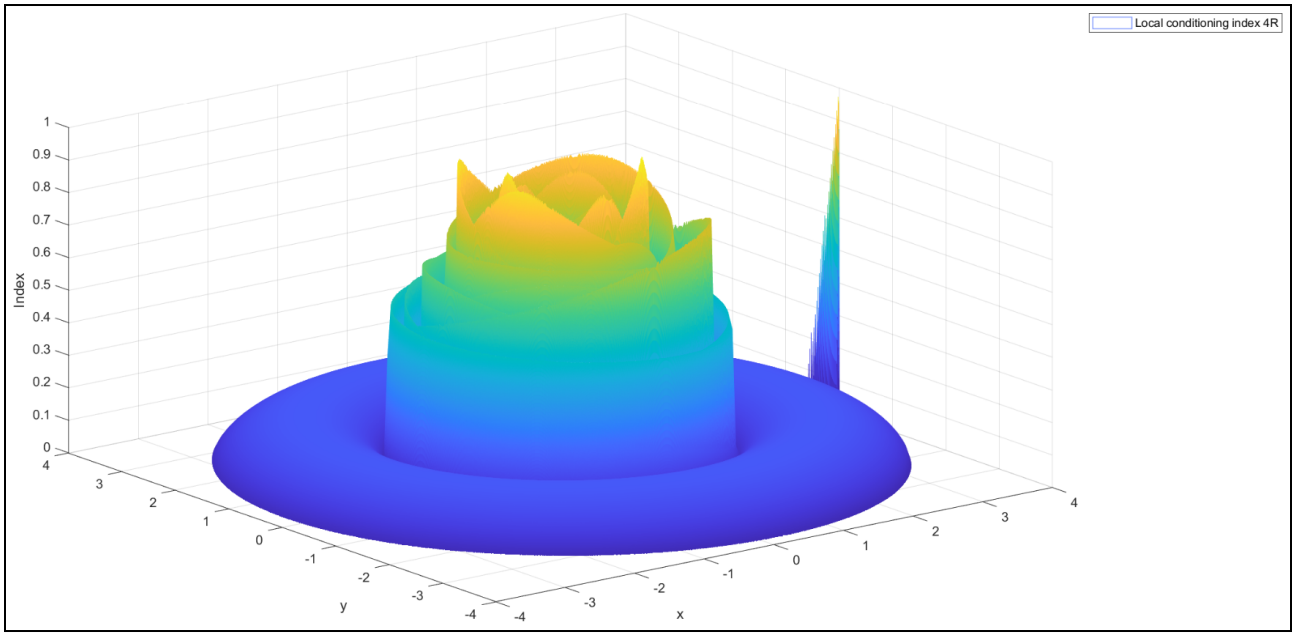
The Local Conditioning Index (LCI) is a metric used in robotics to evaluate the sensitivity of a robot's motion to small changes in its joint angles. Unlike the condition number, which provides a global measure of the sensitivity of the Jacobian matrix to perturbations, the LCI evaluates the local behavior of the matrix around a particular configuration.

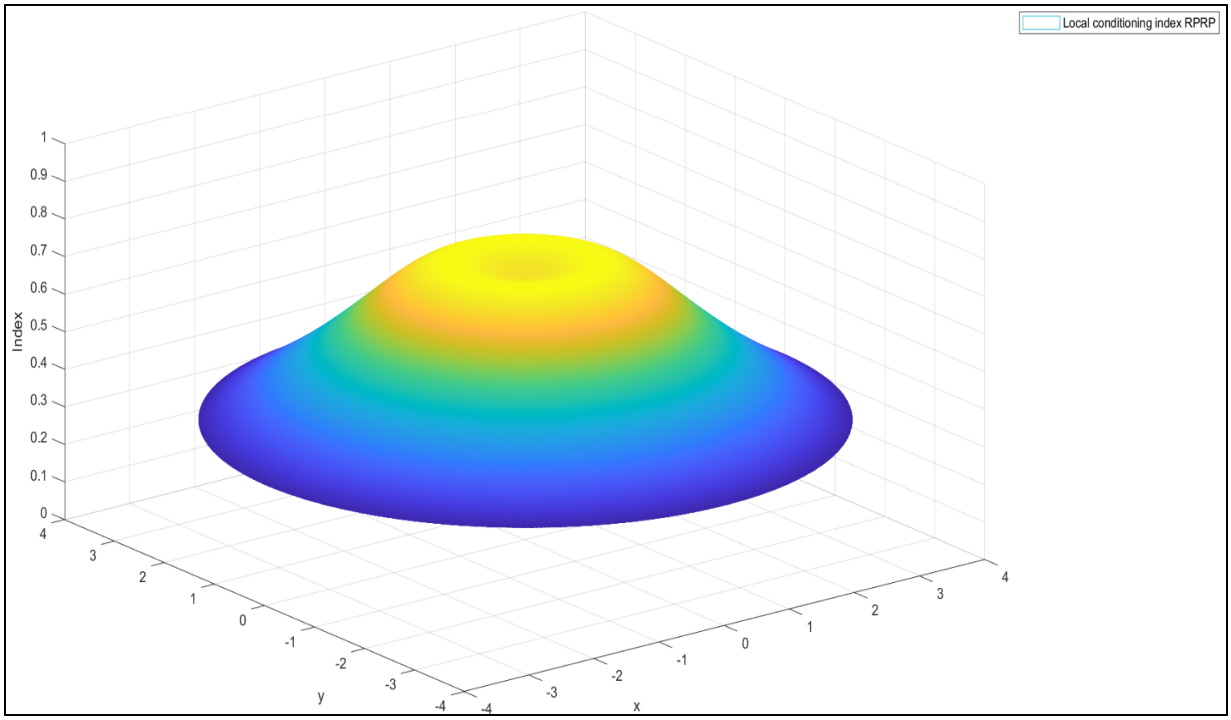
The LCI is calculated by considering the effect of small perturbations in the joint angles on the motion of the end-effector, while keeping the joint velocities constant. More specifically, it measures the ratio of the change in the end-effector velocity caused by a small change in the joint angle to the original end-effector velocity.

$$LCI = \frac{1}{\kappa}$$

where K is the Condition Number

A low LCI value at a specific configuration indicates that the robot is highly sensitive to small changes in its joint angles, which can result in inaccurate or unstable motion of the end-effector. This can be problematic for tasks that require high precision and accuracy, such as assembly, manipulation, or surgery.





Observation: In the graph for 4R manipulator, there is an increase in the values initially and thereafter decreasing to a certain extent. Thereafter the values continue to increase gradually to a certain extent and after that we see a non-uniform increase and decrease in the values.

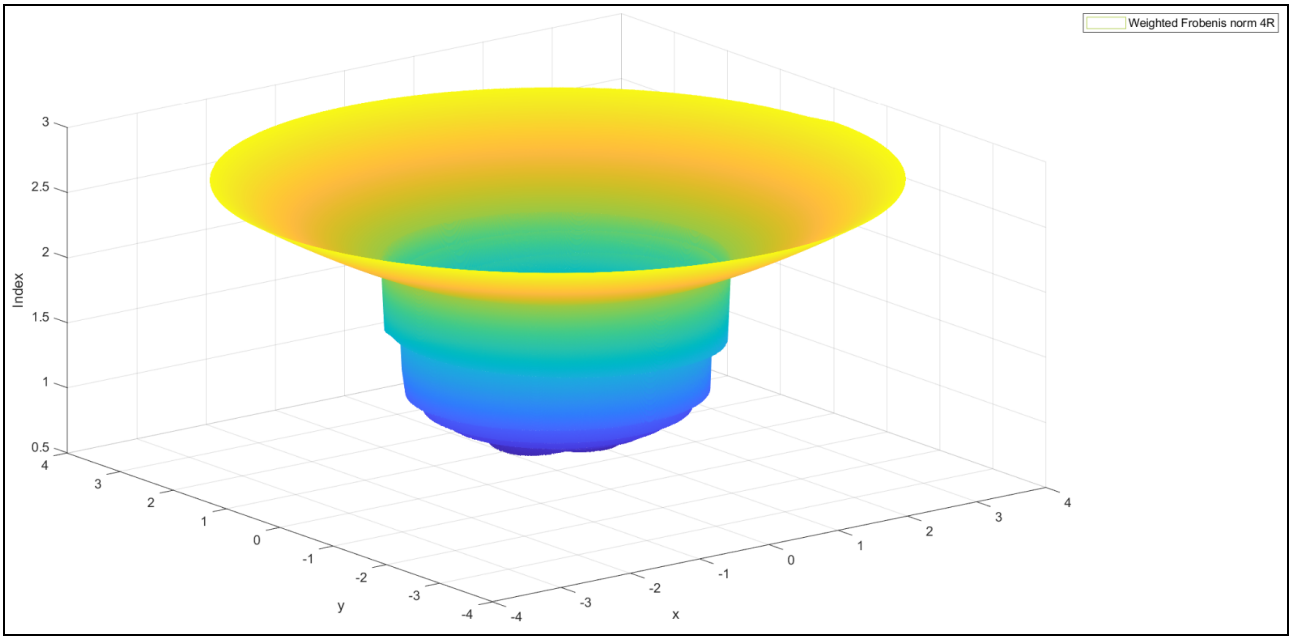
In the graph for RPRP manipulator, the values initially start from a certain value (0.3) and continuously increase up to 0.63 and thereafter start to decrease up to 0.57.
Although the range of values are the same in both RPRP and 4R, there is fluctuation in the case of 4R and it starts with 0 where the change in end-effector velocity is high for the small changes in the joint angles and velocities.

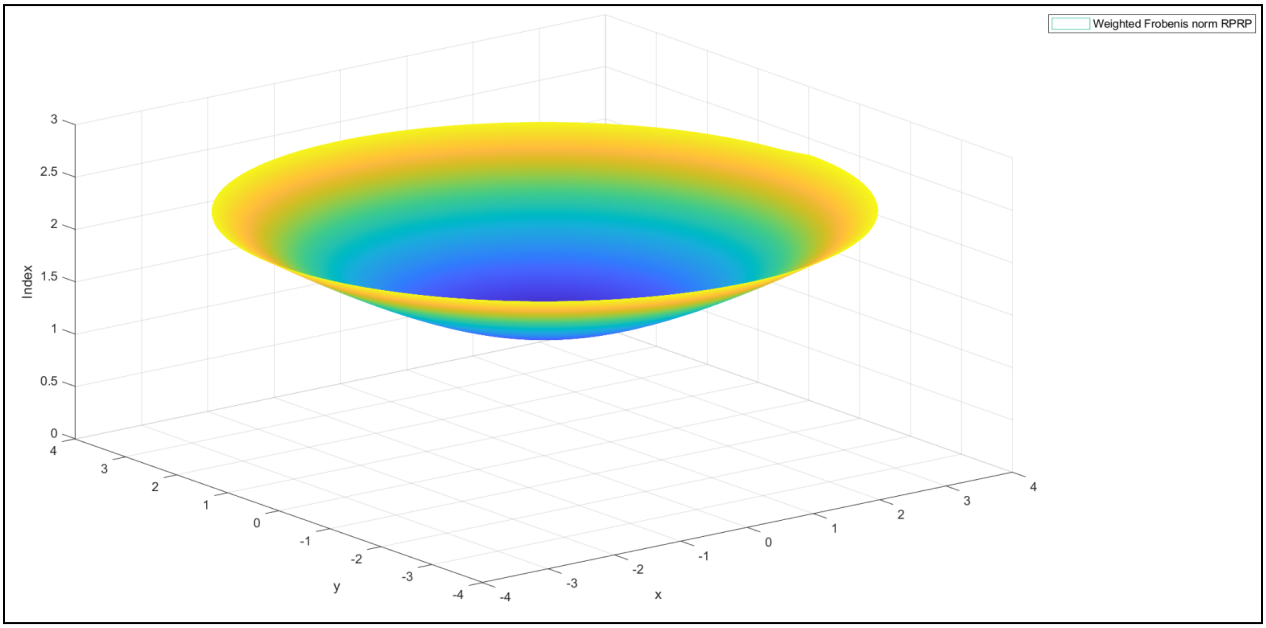
4. **Weighted Frobenius Norm**

The weighted Frobenius norm is a mathematical concept used to measure the difference or error between two matrices. It is commonly used in robot control and motion planning to evaluate the accuracy and performance of a robot's movement.

$$\|\mathbf{J}\|_F(\boldsymbol{\theta}) = \sqrt{\frac{1}{n}\operatorname{tr}(\mathbf{J}\mathbf{J}^T)},$$

The higher the value of the weighted Frobenius norm index, the more sensitive the robot's motion is to changes in the joint velocities. It indicates that the robot is less stable and more susceptible to errors and vibrations. Thus, the robot's motion planning and control algorithms must be designed to minimize the value of the index, ensuring a more stable and accurate robot motion





Observations: In the graph for the 4R manipulator, we see that the values start from a certain maximum value (2.7) and go on decreasing gradually, making a funnel shaped graph. After a certain point in the trajectory, the rate at which the values decrease is increasing to a great extent and at last it reaches a certain minimum value near to 0.5.

In the graph of the RPRP manipulator, we see that initially the values start from a certain maximum value (2.3) and then start to decrease gradually to a certain minimum value near to 1.2.

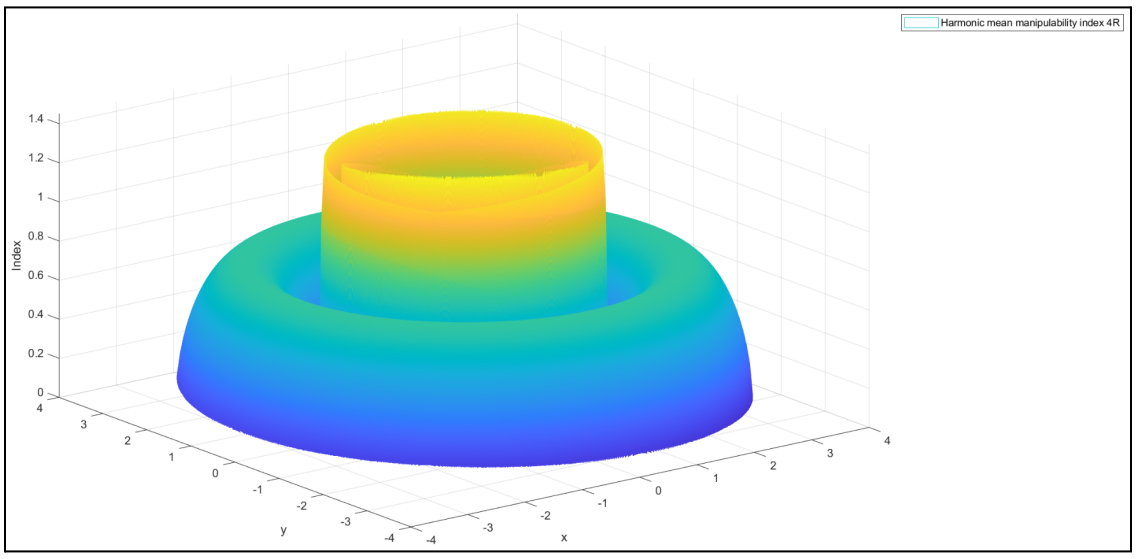
Initially the value of 4R is high and then at last point of trajectory the value of RPRP is higher than the 4R which means 4R becoming more dexterous and and RPRP becoming less dexterous on a comparison basis.

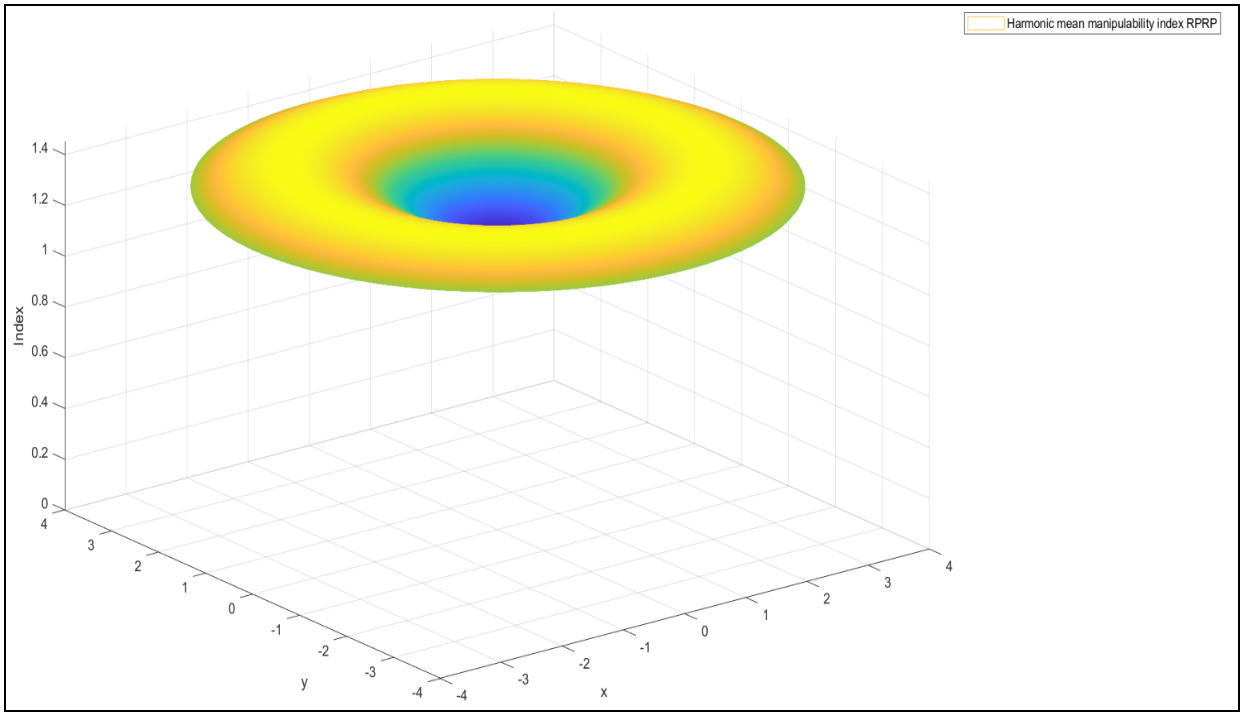
5. **Harmonic Mean Manipulability Index**

Hashimoto also noted that even though a singular value approaches zero, there may not be a noticeable decrease in the manipulability if the other singular values increase in value. This is an important consideration when using the HMMI to evaluate a robot's manipulability. It is defined as -

$$HMMI = \sqrt{\frac{1}{tr\left[\left(J.J^{T}\right)^{-1}\right]}}$$

A high value of the HMMI indicates that the robot is more manipulable, while a low value indicates that the robot is less manipulable. In the neighborhood of any singular point, the HMMI becomes zero. HMMI is a posture-dependent local index. It can be used to identify configurations where the robot has limited manipulability, and to optimize the robot's joint angles and configurations for better manipulability.





Observations: In the graph for 4R manipulator, the initial values start at the value 0 and then keep on increasing till we have a peak and then decrease to a certain level. Thereafter, the values keep on increasing, we also see a non-uniform behavior for some of the graphs.

In the graph for RPRP manipulator, initially the values start from a certain value of 1.33 and keep on increasing till we get a peak (1.41) and there on starts to decrease until the minimum value of 1.15 is reached.
There is always a higher value of HMMI for RPRP than the 4R manipulator which means RPRP is more manipulable than the 4R on the entire workspace.

6. Stochastic Manipulability Index

The Stochastic Manipulability Index is a measure of a robot's ability to move its end-effector in different directions and at different velocities, while taking into account the probability density function of the manipulator arm in those directions. The index is calculated using the velocity vector and the probability density function of the arm for motion in the specific direction range.

$$w_d = \begin{cases} \left(\frac{n}{\iint_{s_d} P(ds_d)\dot{x}_d (JJ^T)^{-1} \dot{x}_d ds_d} \right)^{\frac{1}{2}} & if \det\left(JJ^T\right) \neq 0 \\ 0 & if \det\left(JJ^T\right) = 0 \end{cases}$$

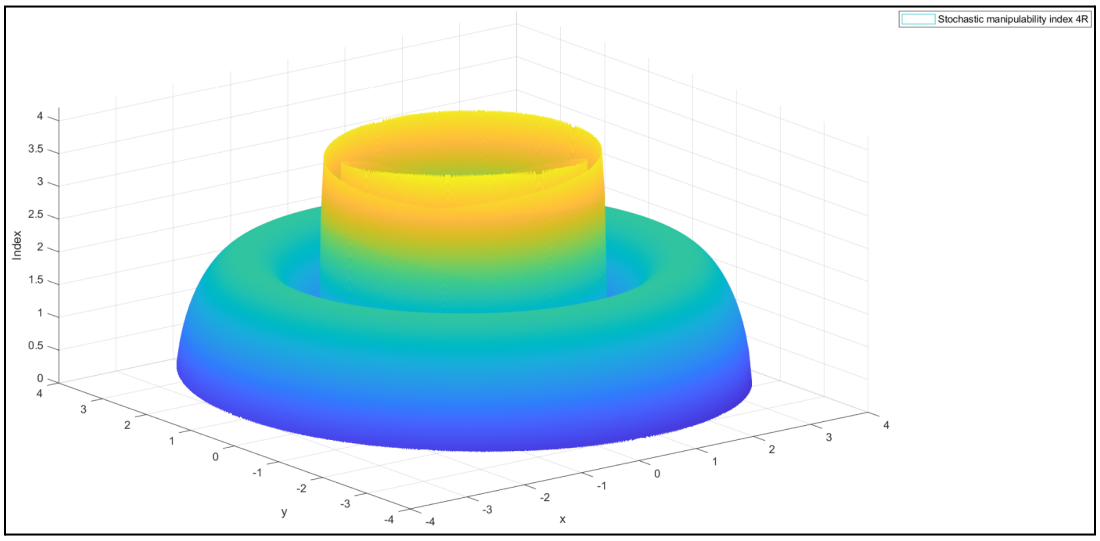
The formula for the index includes the (n) number of degrees of freedom of the manipulator, the probability density function for motion in the direction range ds_d, represented by P(ds_d) and $\dot{x}_d$ is the velocity vector.

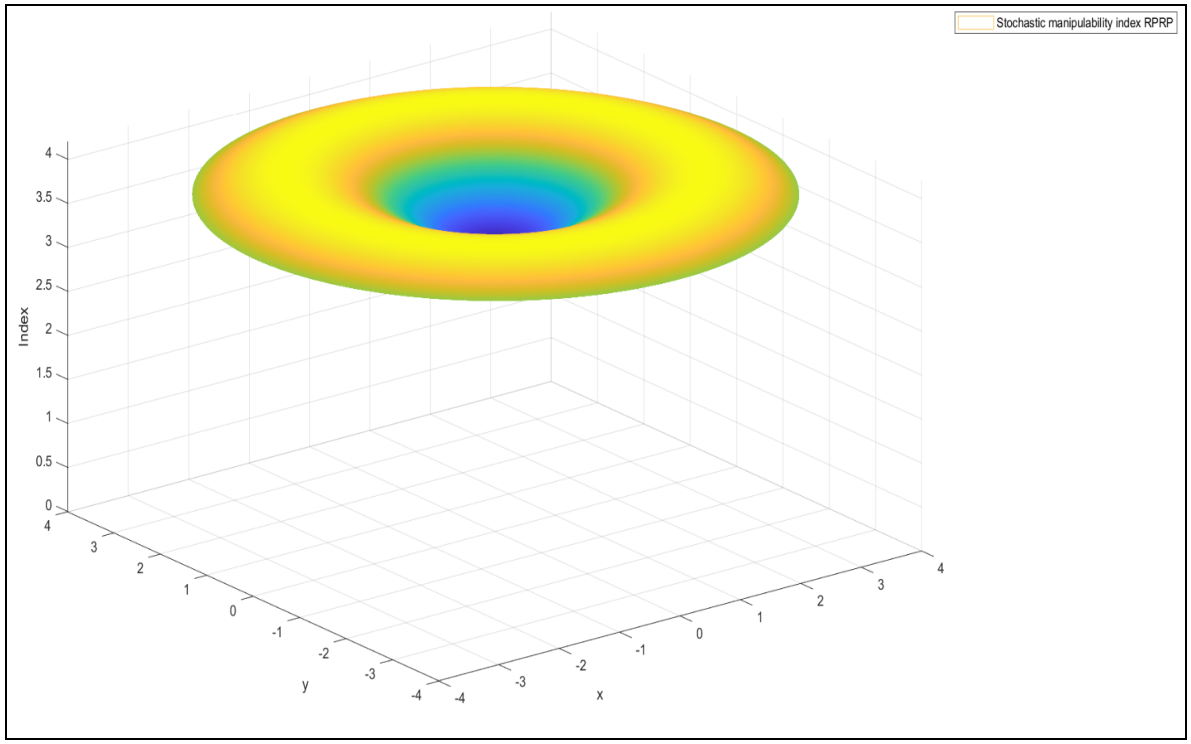
If the probability density function P(ds_d) for a robot's manipulator arm is unknown, one can assume a uniform probability of motion in all directions. In such cases, the Stochastic Manipulability Index can be simplified and expressed in terms of the Harmonic Mean Manipulability Index (HMMI),

$$w_d = \begin{cases} \left(\frac{nt}{tr[(JJ^T)^{-1}]}\right)^{\frac{1}{2}} = \sqrt{nt}(HMMI) & if \det\left(JJ^T\right) \neq 0 \\ 0 & if \det\left(JJ^T\right) = 0 \end{cases}$$

where t is the number of degrees of freedom of the task space.

NOTE: We have solved this index by using the above expression, assuming we didn't know the probability density function.





Observations: In the graph for 4R manipulator, the initial values start at 0 and then keep on increasing till we have a peak (1.75) and then decrease to a certain value. Thereafter, the values keep on increasing, we also see a non-uniform behavior in some parts of the graph.

In the graph for RPRP manipulator, initially the values start from a certain value(3.75) and keep on increasing till we get a peak (3.97) and thereon starts to decrease until the minimum value of 3.2 is reached.

There is always a higher value of stochastic index for RPRP than the 4R manipulator which means RPRP is more manipulable than the 4R on the entire workspace.

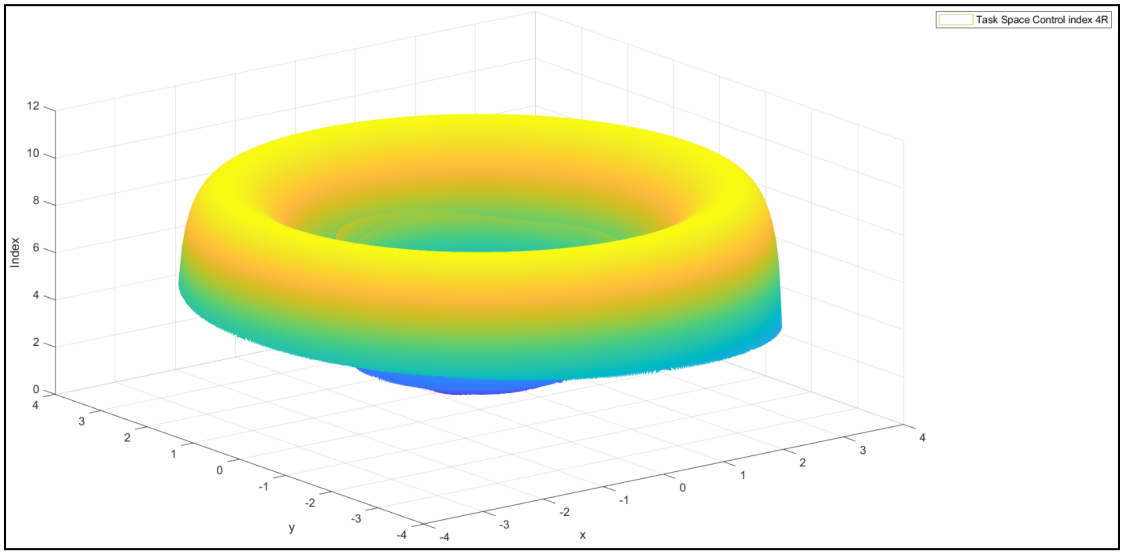
7. **Task Space Control Index**

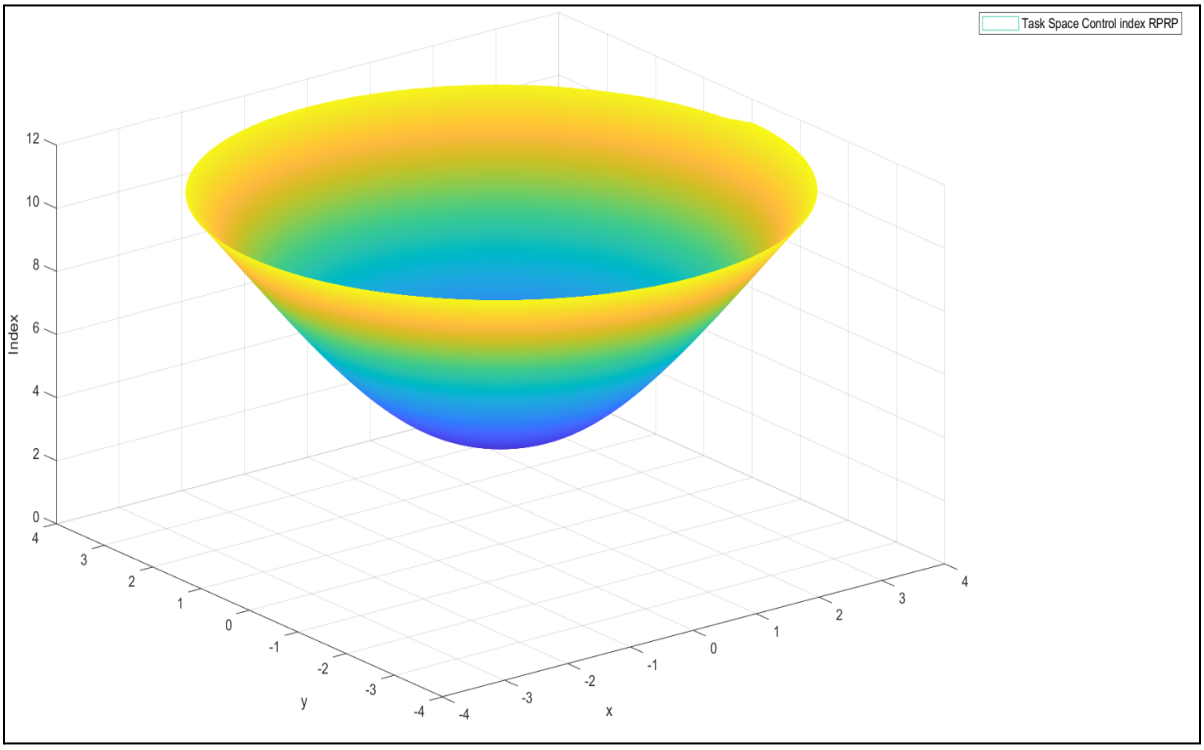
The Task-Space Control Index (TSCI) is a scalar value used to evaluate a robot's control performance in task space. It indicates the robot's ability to accurately follow a desired trajectory in task space. The TSCI is calculated as the square root of the product of the Jacobian determinant and the condition number of the Jacobian matrix.
The formula for the TSCI is given by:

$$\text{TSCI} = \sqrt{(det(JJ^T) \ * \ cond(J))}$$

where J is the Jacobian matrix of the robot, det(J) is the determinant of the Jacobian matrix, and cond(J) is the condition number of the Jacobian matrix.

The TSCI uses both its determinant and condition number to provide a comprehensive assessment of the robot's control performance. A higher value of TSCI indicates better control performance, while a lower value indicates poorer control performance.





Observations: In the graph for 4R manipulator, initially the values start from a certain value (3) and then keep on increasing until we see a peak (9) in the graph and after that the values start to decrease until a minimum value of 0 is reached.
In the graph for the RPRP manipulator, we see that initially the values start from a certain maximum value (11) and then start to decrease gradually to a certain minimum value (3).

There is always a higher value of task space control for RPRP than the 4R manipulator which means RPRP has better control performance than the 4R on the entire workspace.

8. **Dynamic Manipulability**

The Dynamic Manipulability Index is a measure of a robot's ability to generate acceleration based on a joint driving force, taking into account the robot's manipulator dynamics. It quantifies the manipulating ability of a robot's end effector with respect to its dynamic properties. The index is defined mathematically and takes into account the robot's inertia matrix and the Jacobian matrix. By using these matrices, the dynamic manipulability index provides a comprehensive measure of the robot's dynamic manipulability.

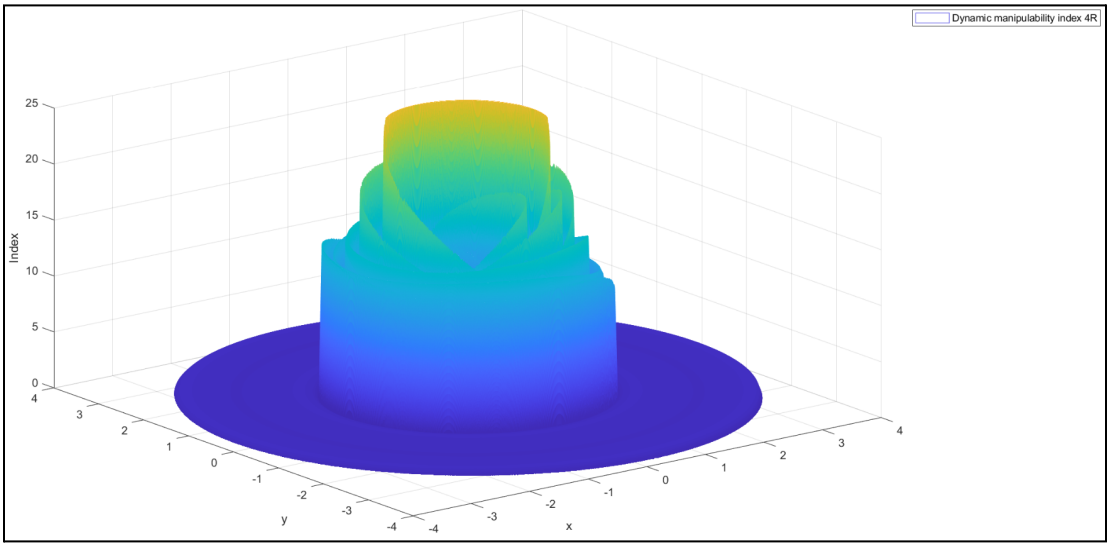
$$\mu_d = \sqrt{\det\left[J\left(M.M^T\right)^{-1}J^T\right]}$$

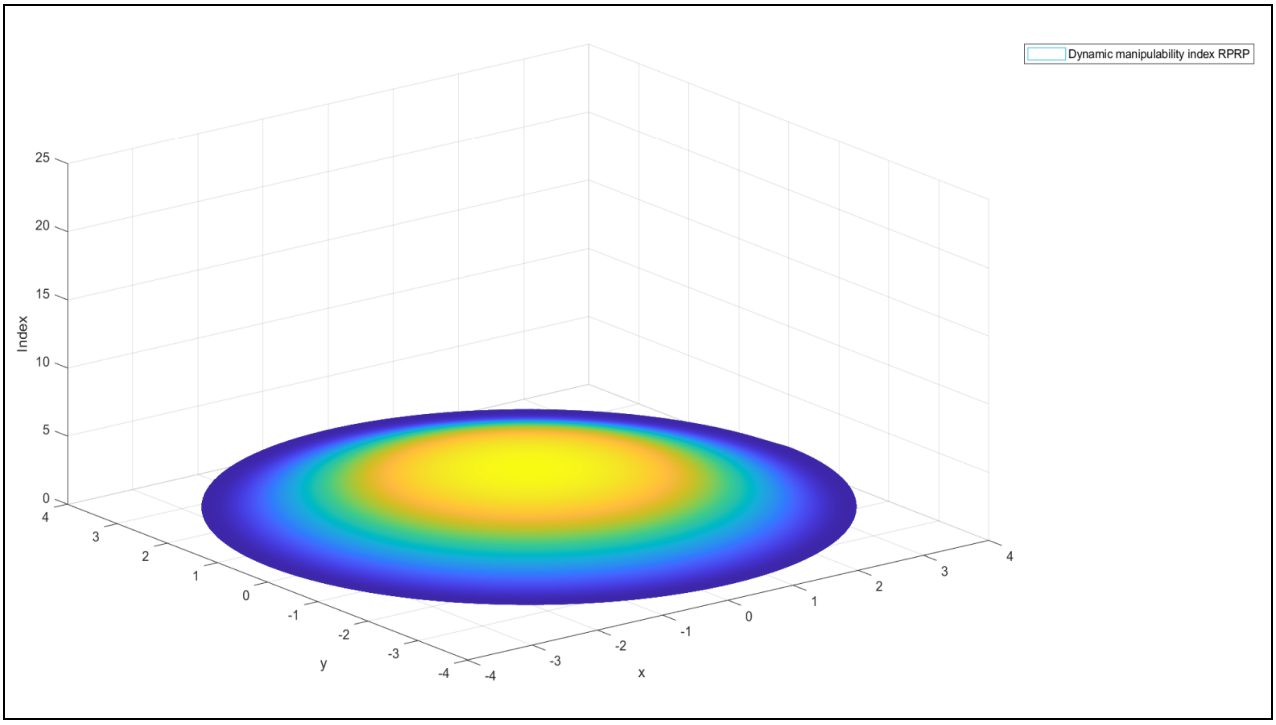
For a non-redundant manipulator, the equation reduces to:

$$\mu_d = \left|\frac{\det(J)}{\det(M)}\right|$$

where M is the inertia matrix.

Like the manipulability index, the dynamic manipulability index (μ_d) is also a posture-dependent local performance metric. A higher value of the dynamic manipulability index indicates that the robot can generate a greater acceleration in response to a given joint driving force, indicating better dynamic performance.





Observation: In the graph for 4R manipulator, initially the values start from a certain minima and stay approximately the same for some time and then the values start increasing to a great extent (7). After this we see a non-uniform behavior in which the values are sometimes increasing and sometimes decreasing and it reaches to 25 at max.

In the graph for RPRP manipulator, initially the values start from a value of 1 and then gradually increase till we have a peak of value 3.

Initially the value is higher for RPRP and then the value for 4R increases significantly which means initially RPRP has better dynamic performance and after that 4R has much better dynamic performance.

9. **Minimum Singular Value**

Minimum singular value of a matrix is an important concept in determining the quality of a robot's performance. The minimum singular value is the smallest singular value of a matrix, which is a measure of the spread of the matrix's columns.

Using the Singular Value Decomposition (SVD) theorem, the Jacobian matrix can be represented as a product of three matrices:

$$J = U \Sigma V^T$$

where U is a m x m orthogonal matrix, V is n x n orthogonal matrix; and is an mxn diagonal matrix. The diagonal matrix consists of elements a_{ij} such that: a_{ij} = 0 if i = j, and a_{ij} = σ_i if i = j.

$$\Sigma = \begin{bmatrix} \sigma_1 & & & \\ & \sigma_2 & & \\ & & \ddots & \\ & & & \sigma_m \end{bmatrix}$$

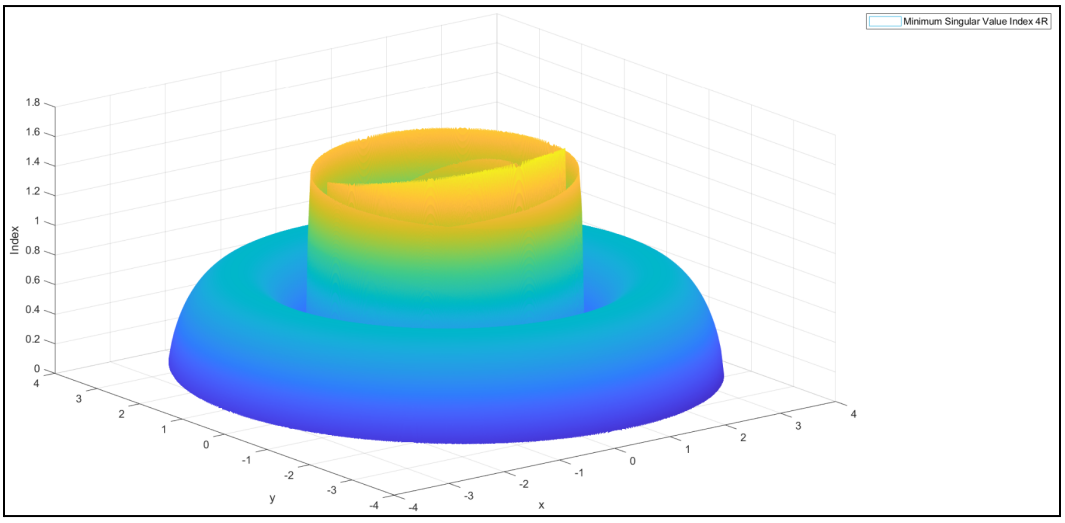
The elements of σi (scalars) are singular values of the matrix, such that:

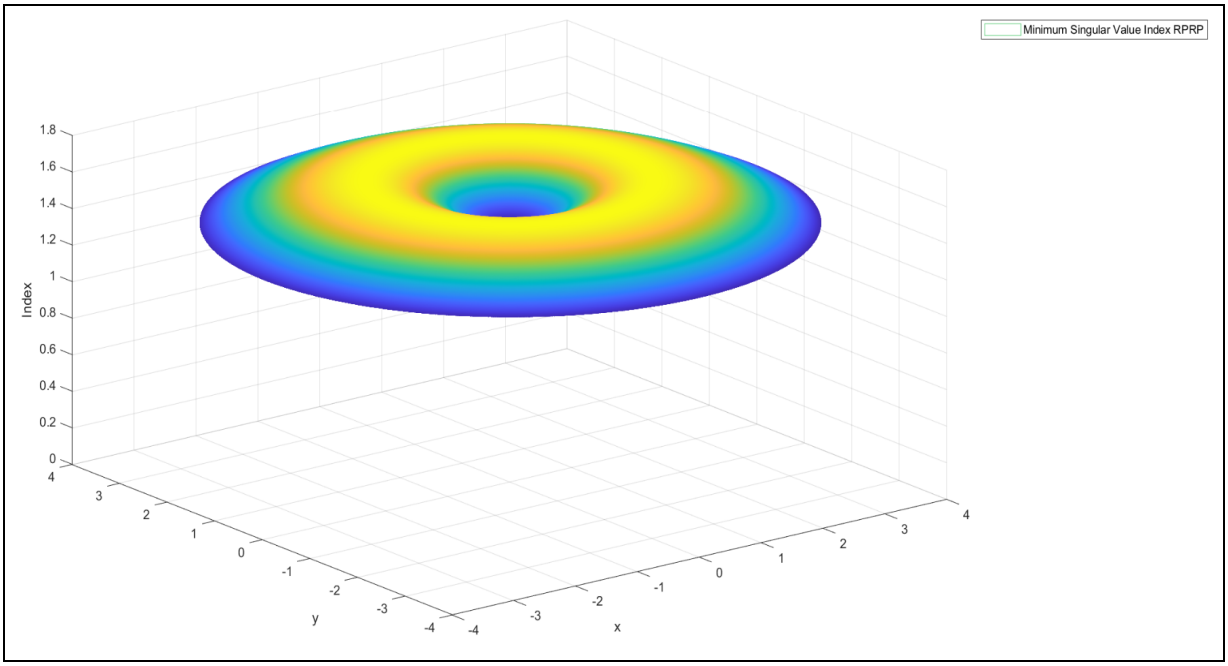
$$\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_m$$

Singular values of Jacobian (J) are defined as the nonnegative square roots of non-zero Eigenvalues of the square matrices (JJ^T) and (J^TJ). Extensive work on the calculation of SVD and it applications can be found in the:

$$\sigma_{\min} = \min(\sigma_1, \sigma_2, \ldots \sigma_m)$$

The minimum singular value represents the minimum transmission ratio, the maximum force transmission, and maximum accuracy. MSV represents the direction in which it is most difficult for the manipulator's end-effector to move, ignoring all other directions. A low minimum singular value indicates that small changes in joint velocities can cause large changes in the end-effector velocity, which can result in inaccuracies in the robot's motion.





Observations: In the graph for 4R manipulator, the initial values start from the minimum value of 0, then keep on increasing till we have a peak of 0.6 and then decrease to a certain level. Thereafter, the values keep on increasing, we also see a non-uniform behavior for some parts of the graphs and value reaches to 1.2 .

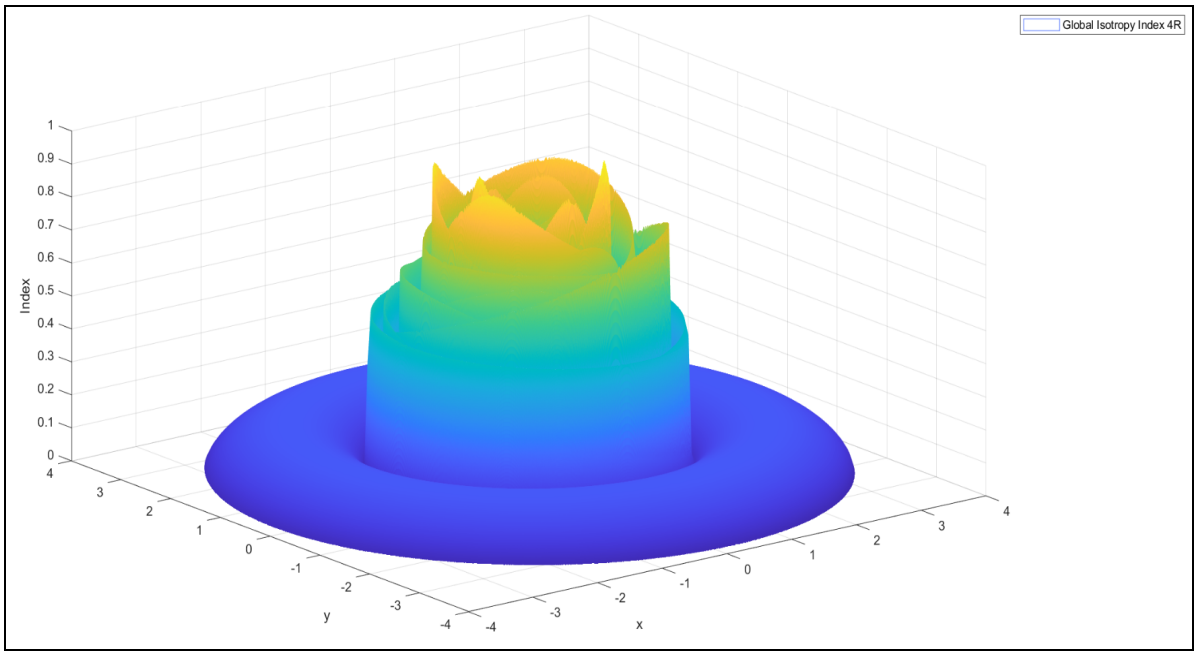
In the graph for RPRP manipulator, initially the values start from a minimum value of about 1.4 and then keep on increasing till we have a peak, approximately 1.6, then the values start to decrease till the trajectory ends.
The value of MSV is always higher for RPRP than the 4R manipulator for the entire workspace which indicates that small changes in joint velocities of 4R can cause large changes in the end-effector velocity. Therefore RPRP is more preferable.

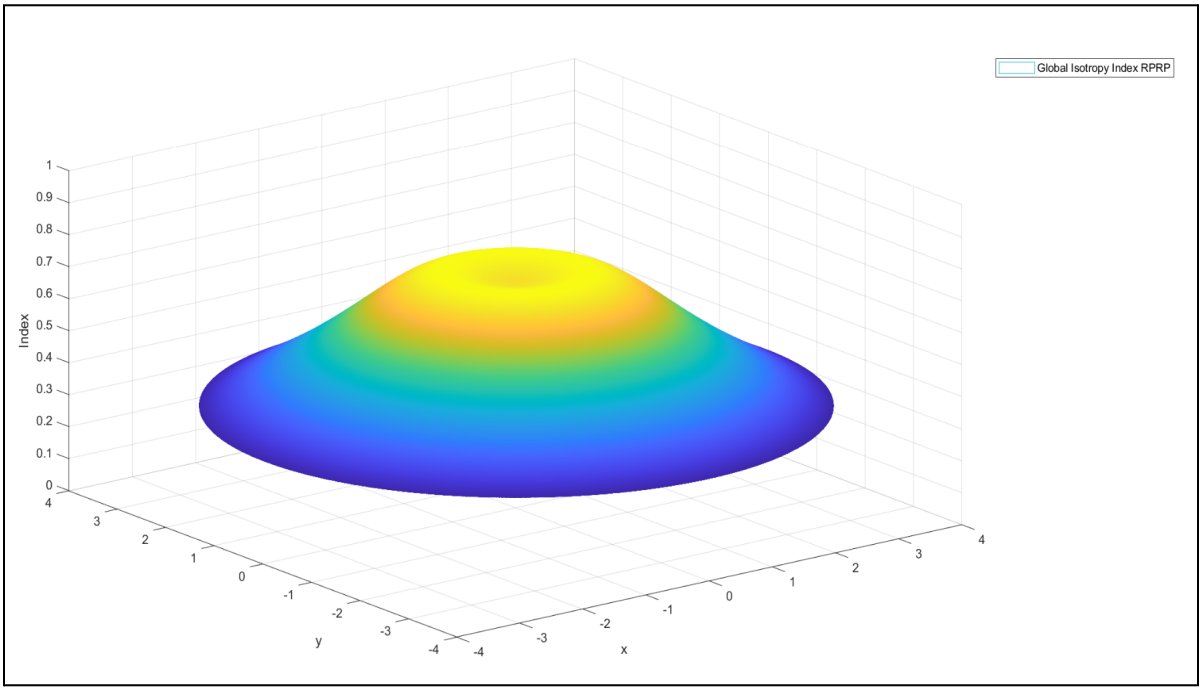
10. Global Isotropy Index

It is defined as the ratio of the minimum to the maximum singular values for the entire workspace

$$GII = \frac{\sigma_{\min}}{\sigma_{\max}}$$

The Geometric Isotropy Index (GII) is a measure of a manipulator's worst performance, it considers the minimum and maximum singular values across the entire workspace.
A lower value of the GII indicates poorer manipulability and worse performance, while a higher value indicates better manipulability and better performance. The GII is a useful tool for evaluating a manipulator's overall performance and identifying areas where improvements can be made.





Observation: In the graph for 4R manipulator, initially the starting value is 0 and then it is increasing till a peak (0.1) and thereafter decreasing to a certain extent(approximately 0). Thereafter the values continue to increase gradually to a certain extent(0.2) and after that we see a non-uniform increase and decrease in the values.

In the graph for RPRP manipulator, the values initially start from a certain value (0.3) and continuously increase up to a value slightly higher than 0.6 and thereafter start to decrease up to 0.67.

Although the range of values are the same in both RPRP and 4R, there is fluctuation in the case of 4R and it starts with 0 indicates poorer manipulability and worse performance at initial points of the trajectory.

11. Global Manipulability Index

The Global Manipulability Index is based on the manipulability index. The global manipulability index (GMI) is defined as the integral of a manipulability index over the whole manipulator workspace, given as

$$GMI = \frac{\mathrm{A}}{\mathrm{B}}$$

where A and B are given as

$$A = \int_W (\mu)\, dW \; and \; B = \int_W dW$$

where W is a specific point in the manipulator workspace, μ is the manipulability at that point in the workspace, and B is the workspace volume. A GMI closer to zero demonstrates poor handleability.

For 4R, the GMI value is 5.5033.
For RPRP, the GMI value is 10.2729.

Although the values for both the manipulators are higher than 0 but since the value of GMI for RPRP is higher than the 4R, therefore RPRP has better handleability than the 4R.

12. Structural Length Index

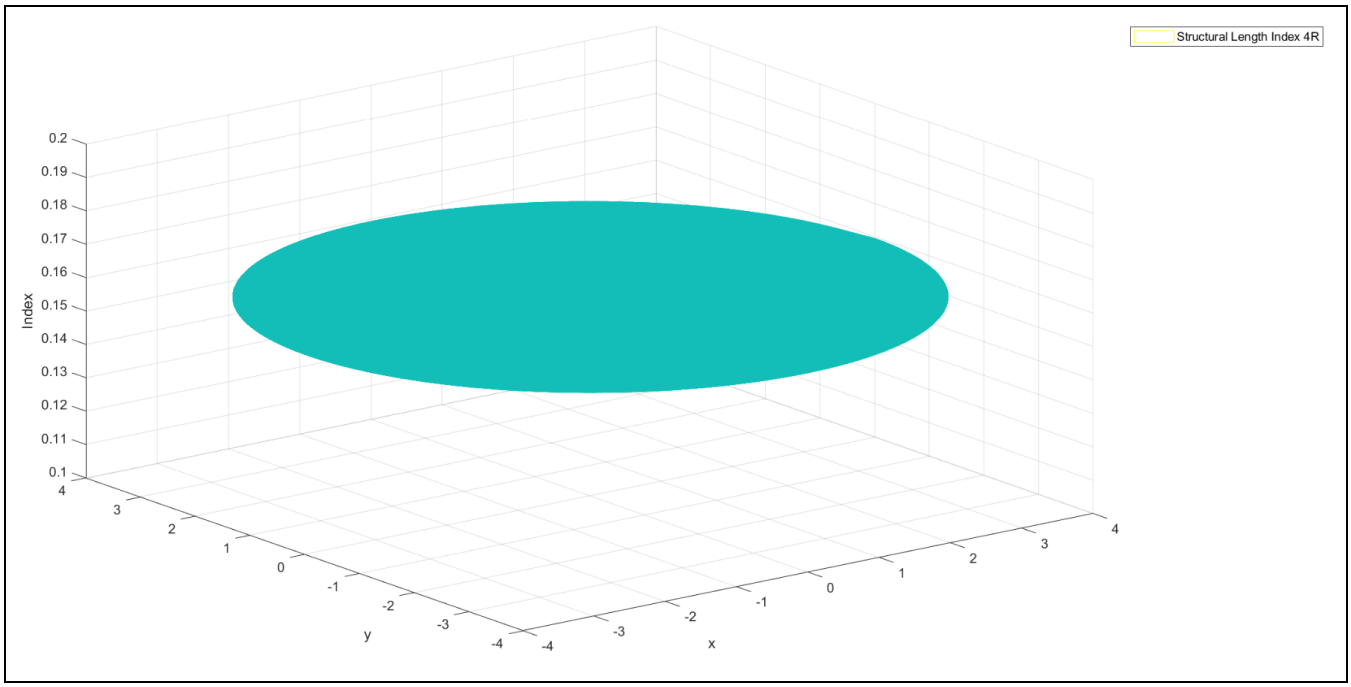
A good robotic design is measured by a small length sum with a large workspace volume. The structural length index is given by -

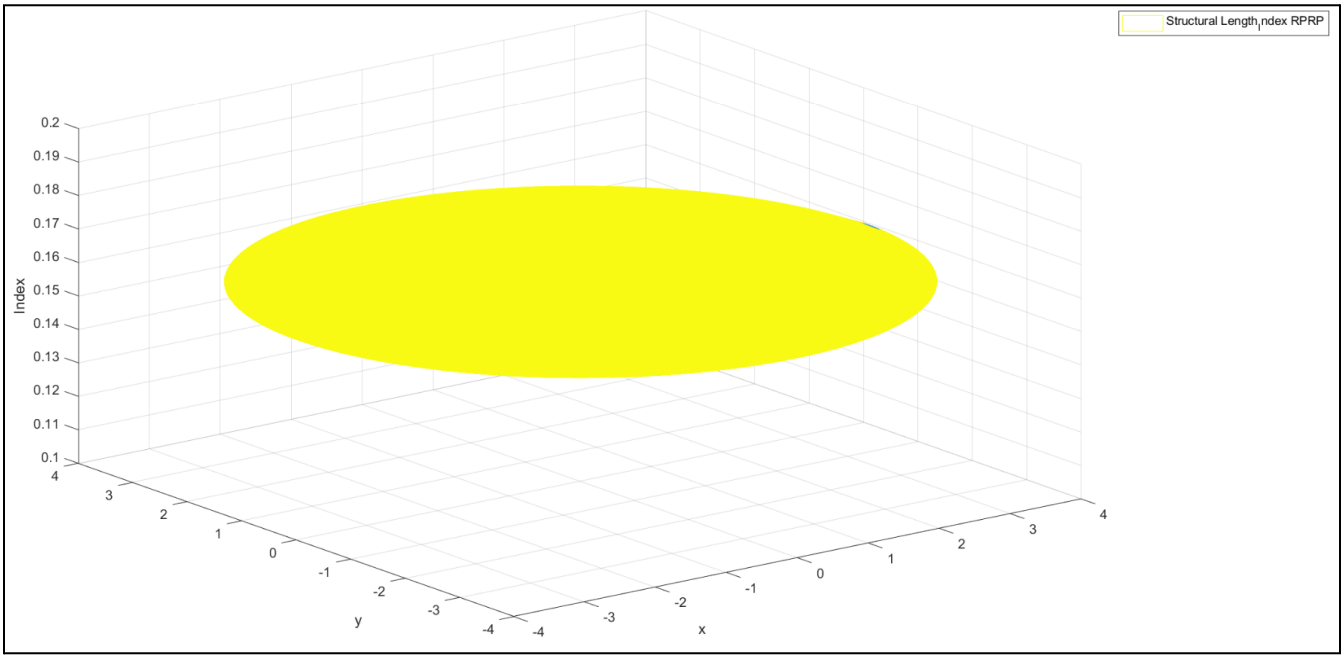
$$Q_L = L / \sqrt[3]{V}$$

where V is the volume of reachable workspace and L is the length sum of the robot manipulator given by

$$L = \sum_{i=1}^{n} (a_{i-1} + d_{i-1})$$

where a_{i-1} and d_i are the link length and joint offset respectively.





Observations: In both cases 4R and RPRP, we get the same results as the workspace is the same and the lengths are not changing in 4R and in RPRP the lengths are not changing to a great extent.

13. Kinematic performance index

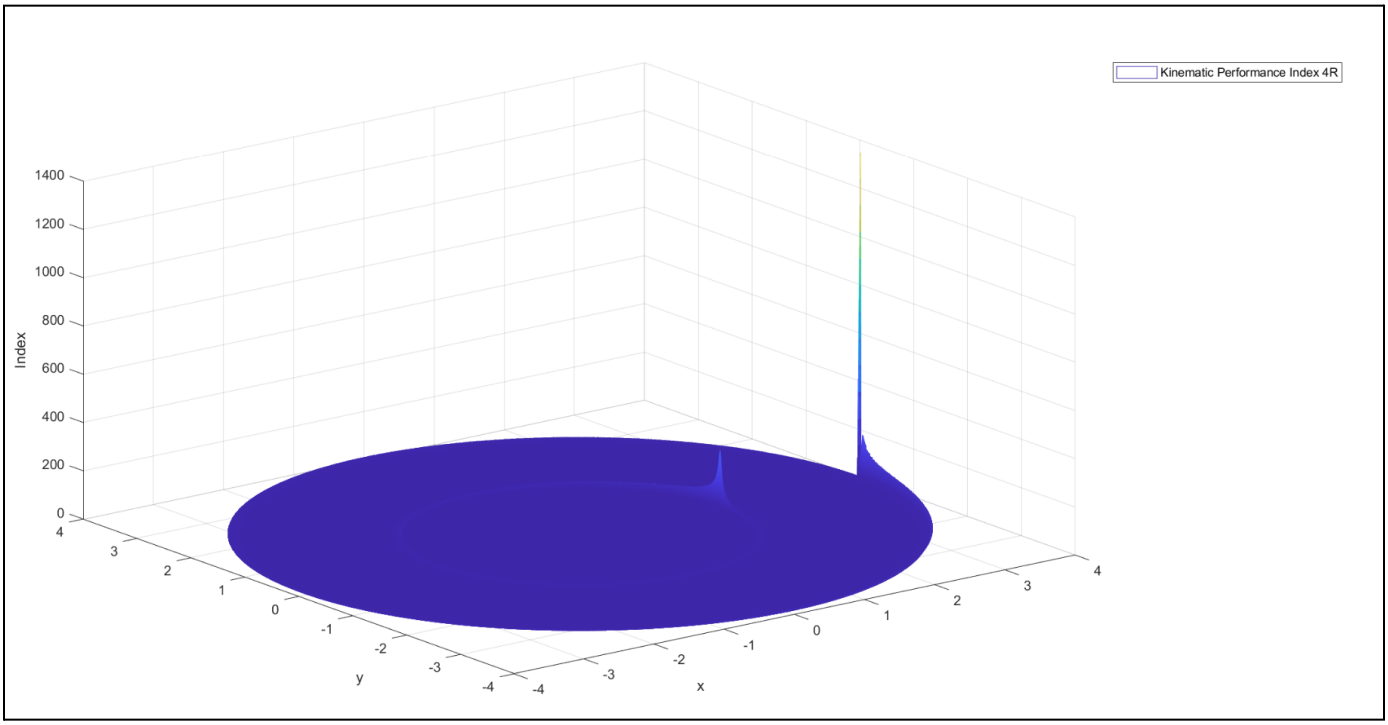
A kinematic performance index of a robotic mechanical system is a scalar quantity that measures how well the system behaves with regard to the force and motion transmission. Nowadays, the reciprocal of the condition number of the Jacobian matrix,

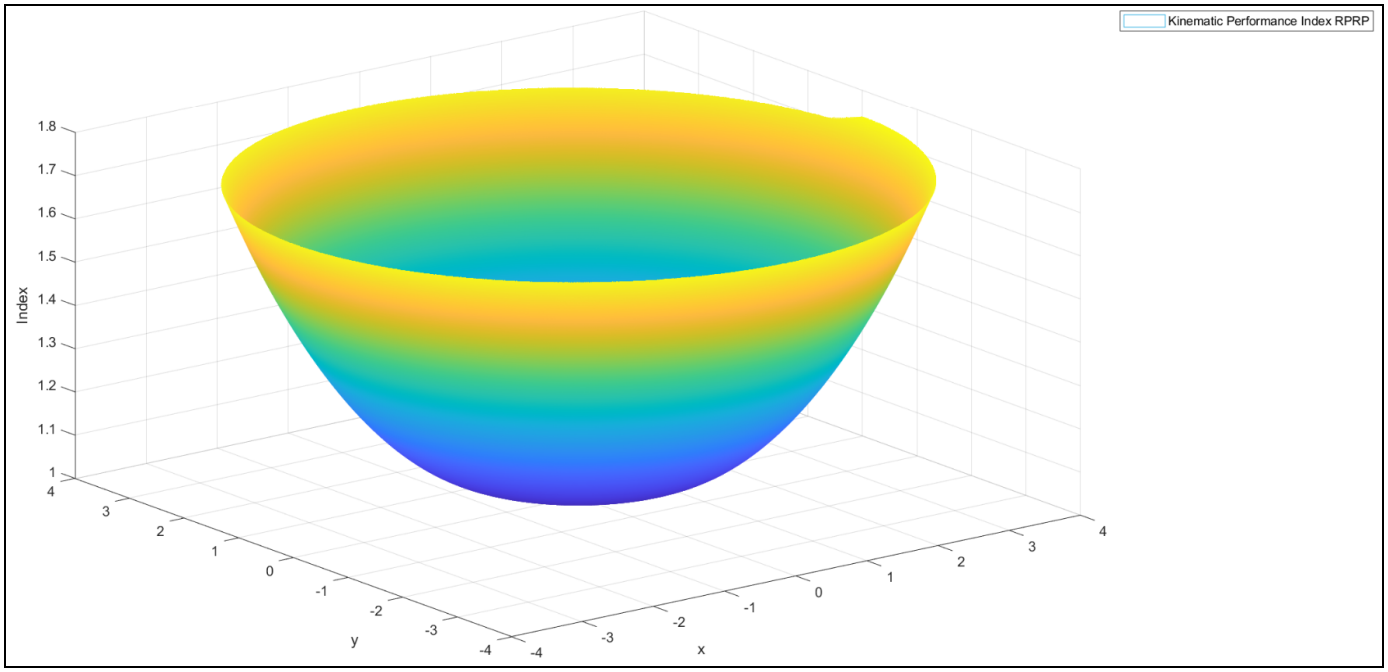
which can quantify the distance to singularities, is used as a kinematic performance index called kinetostatic conditioning index (KCI), defined as:

$$\mathrm{KCI} = \frac{1}{k(\boldsymbol{J}_N)} \times 100\%, \quad k(\boldsymbol{J}_N) = \frac{1}{m}\sqrt{tr(\boldsymbol{J}_N \boldsymbol{J}_N^T) tr((\boldsymbol{J}_N \boldsymbol{J}_N^T)^{-1})}$$

where $k(J)$ is the condition number

The kinematic performance index typically takes into account several factors, including the speed of the system, the acceleration and deceleration rates, and the amount of jerk (the rate of change of acceleration). A higher kinematic performance index indicates that the system is able to move more smoothly and accurately, with less jerky motion and fewer errors in its trajectory.





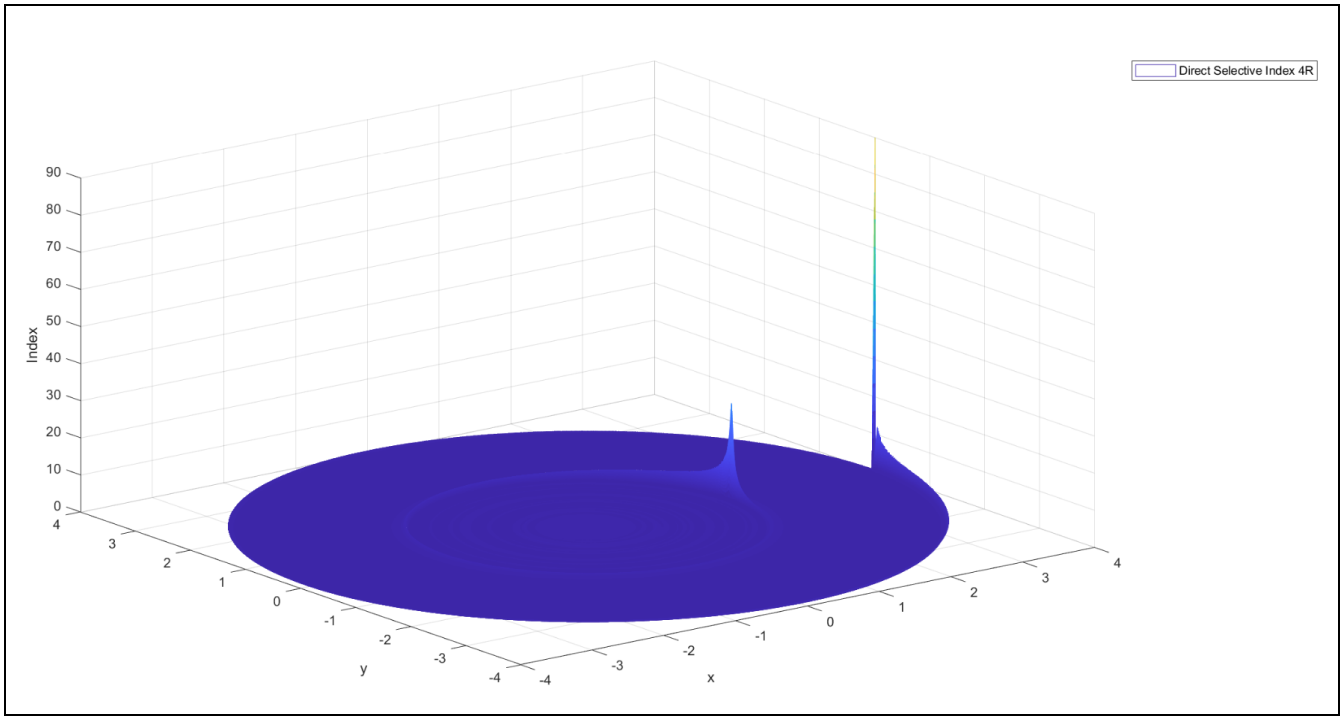
The value of KCI in the 4R case is greater than the value of KCI in RPRP which means 4R is able to move more smoothly and accurately, with less jerky motion and fewer errors in its trajectory according to this index. But initially the value of 4R is 0 and RPRP has a value of 1.7 which means at the start of the trajectory RPRP has better performance.

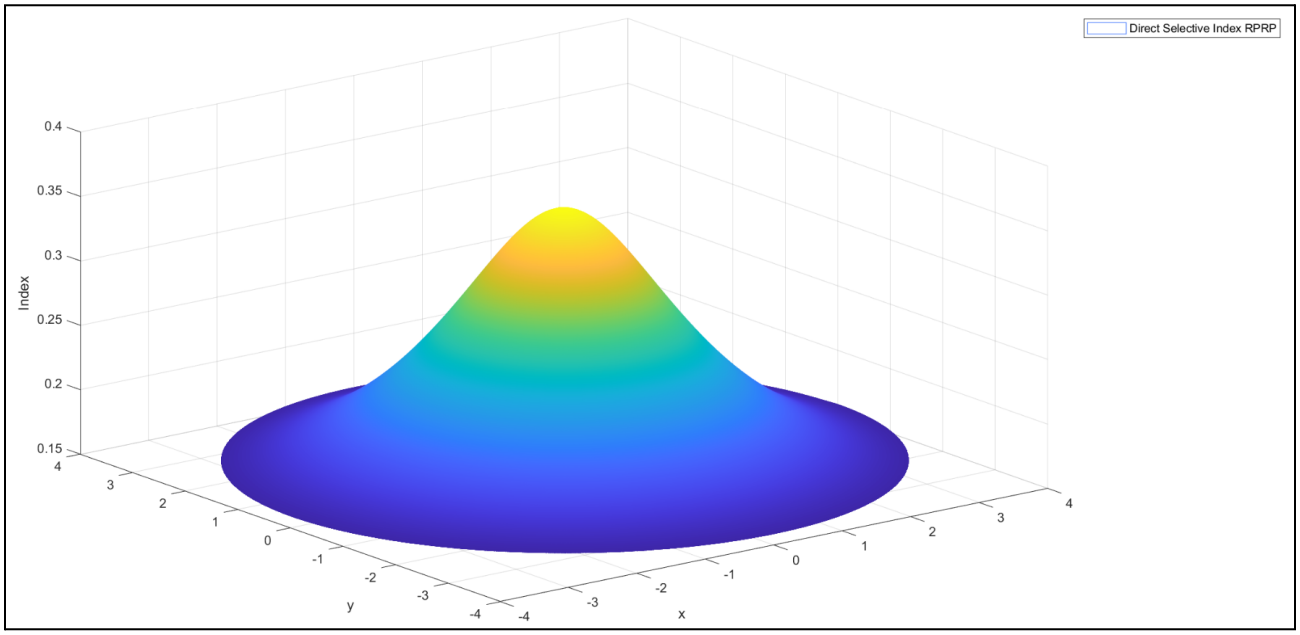
14. Direction Selective Index

The direction-selective index DSI allows independently evaluating the translational capabilities, and hence the performances, of a parallel manipulator along the axes of its world reference frame.

$$\mu = \frac{1}{\sqrt{det(\boldsymbol{J}^{-\mathrm{T}}\boldsymbol{J}^{-1})}}$$

where J is the Jacobian matrix



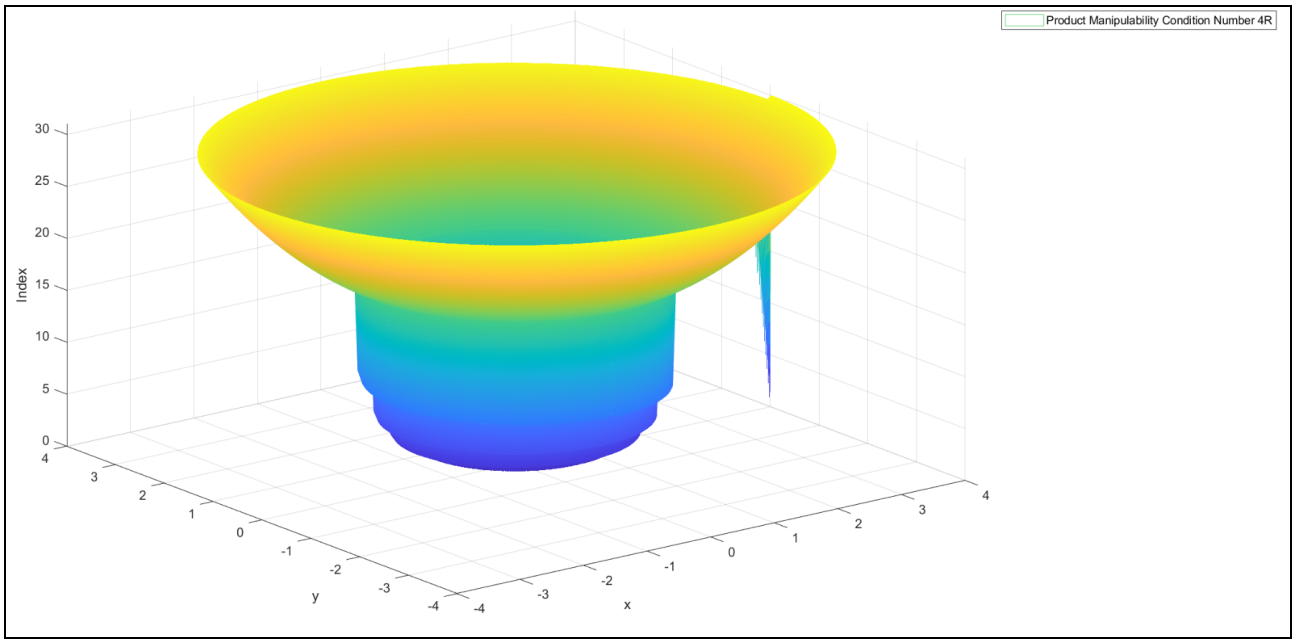


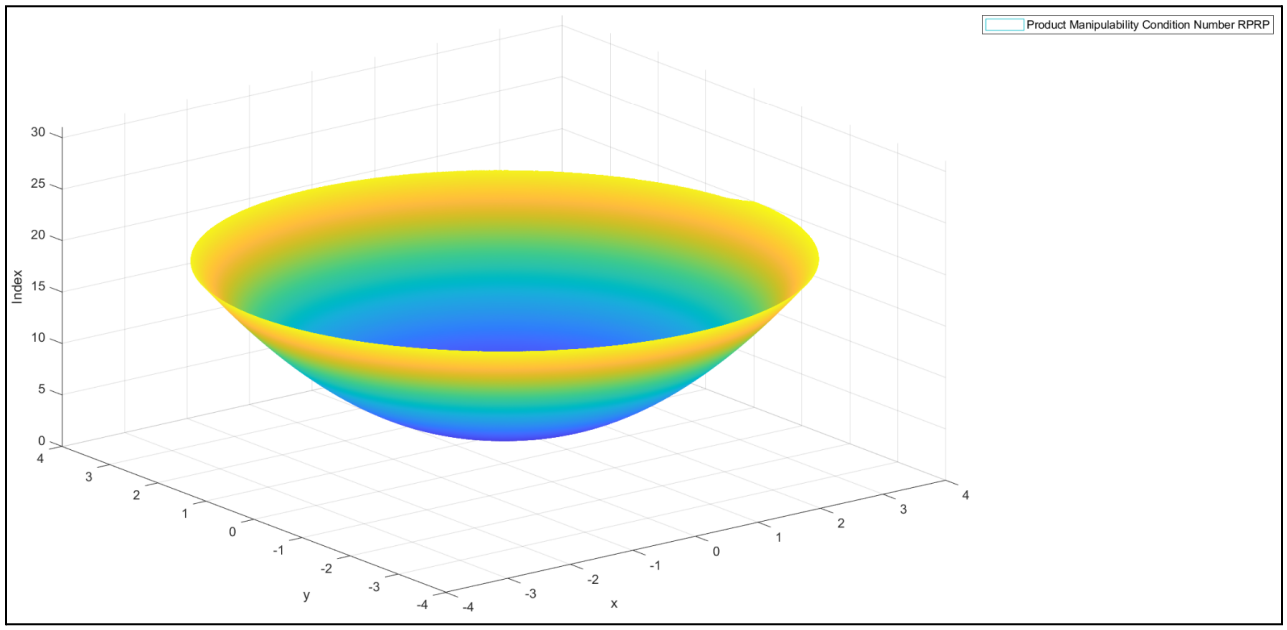
15. Product of the Manipulability Index and Condition Number

Both the manipulability and condition number, despite their limitations, are widely accepted and used in local performance measures. To overcome their combined limitations, Kucuk et al. proposed a new performance index (ρ) that is simply a product of the manipulability index (μ) and the condition number (κ).

$$\rho = \mu * \kappa$$

The aim of this new index was to make it independent of the drawbacks of the determinant of the Jacobian.





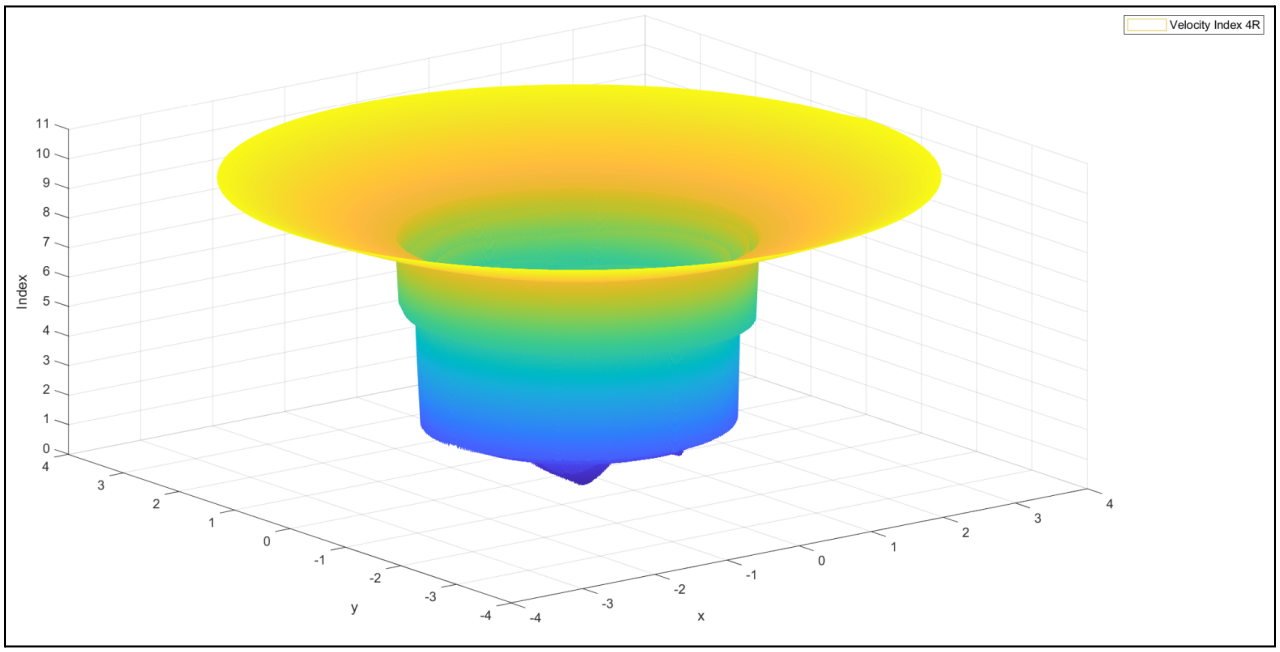
Observations: In the graph of 4R manipulator, we can see that the index value starts from a certain value(approximately 30) which then gradually decreases and after some time it decreases at a much faster rate to some value close to 0.

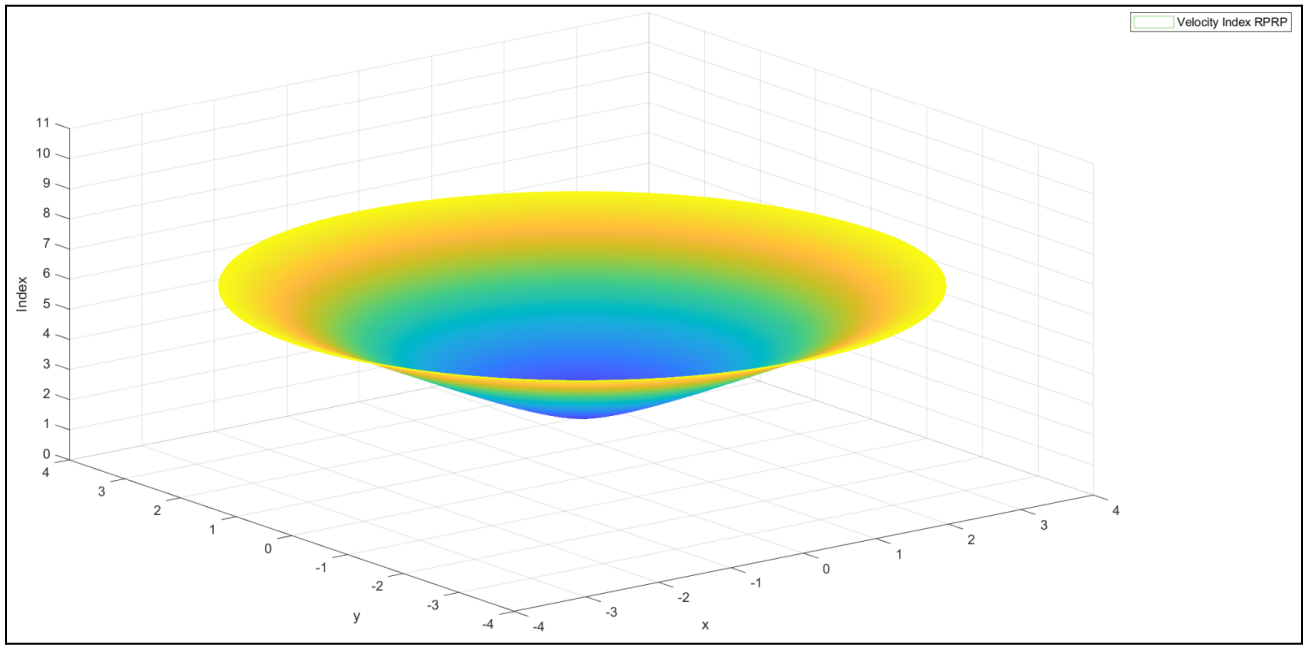
In the graph of RPRP manipulator, we can see that initially, the index value starts from around 20 and gradually decreases to a value near to 5.

Conclusion: From the above observations we can conclude that RPRP manipulator is more precise to a great extent whereas the 4R manipulator is better as the trajectory is about to end.

16. Velocity index

- $\dot{X} = J * \dot{q}$
- $\dot{X} = J * [1; 1; 1; 1]$
- Velocity index = $\sqrt{\dot{X}^T * \dot{X}}$

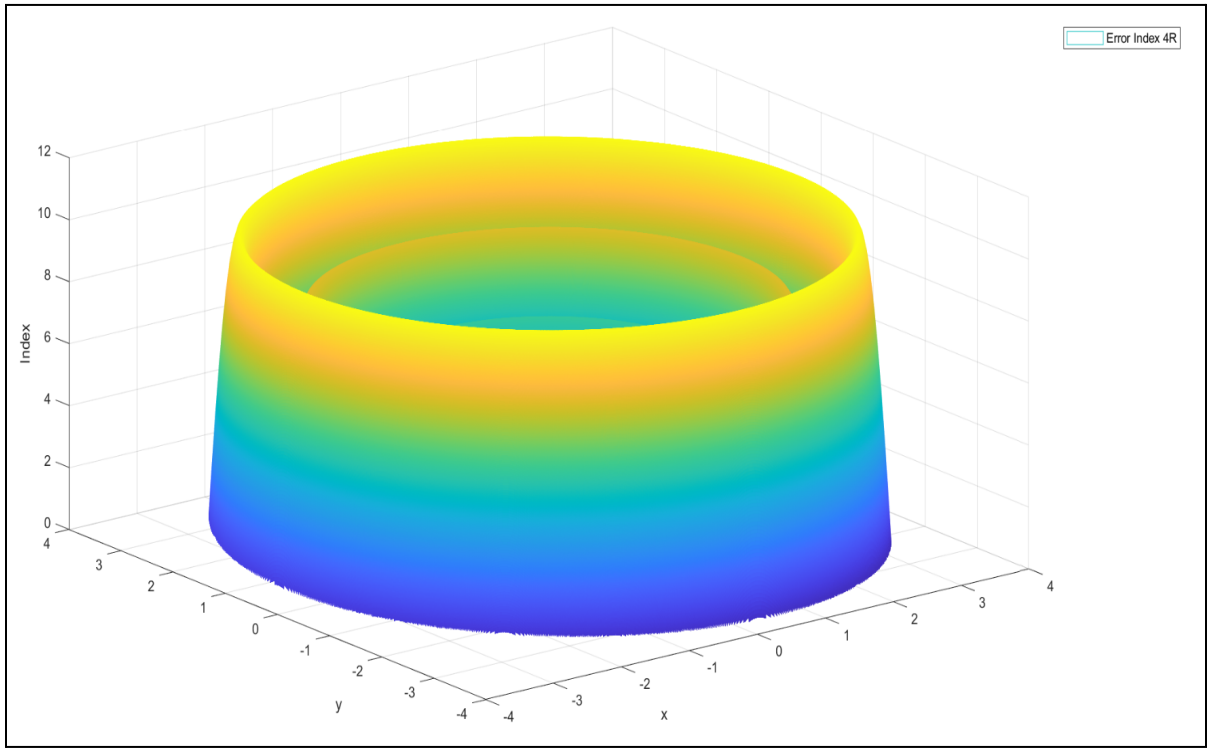


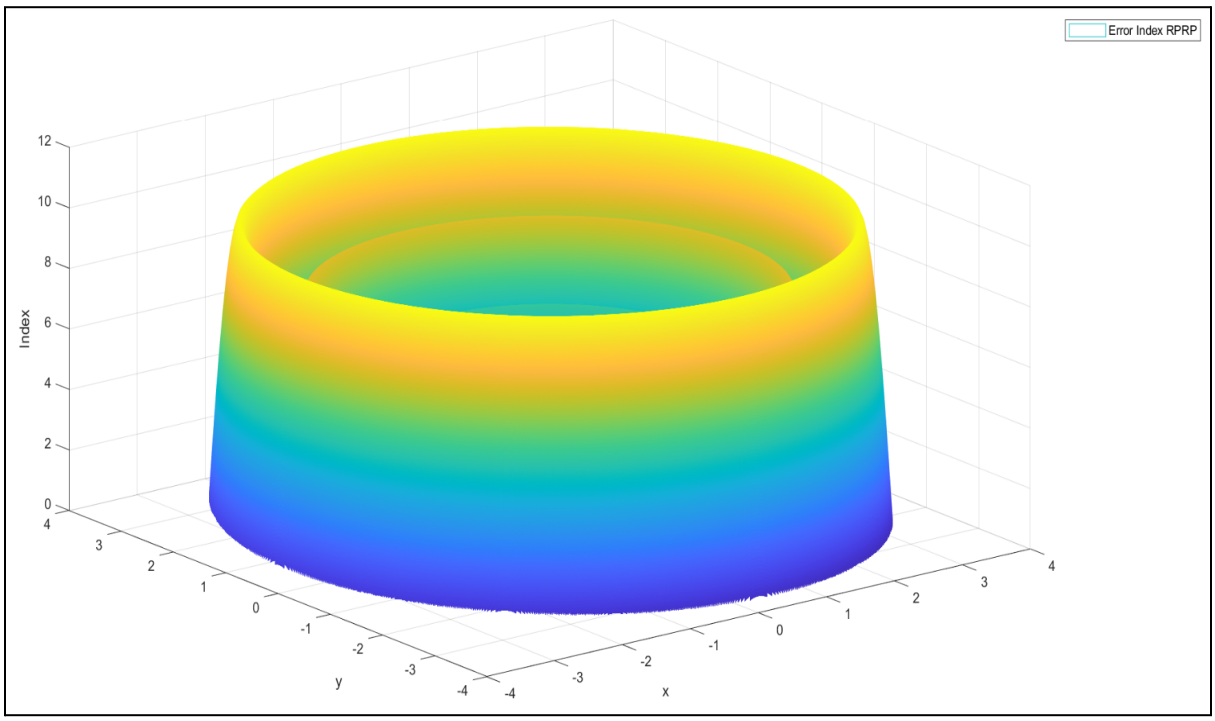


17. Error index

- $[xo; yo] = forward\ kinematics\ (qi)$
- $[xp; yp] = forward\ kinematics\ (qi + 0.05 * qi)$
- $[xm; ym] = forward\ kinematics\ (qi - 0.05 * qi)$
- err = $[xo; yo; xo; yo] - [xp; yp; xm; ym]$

- Error index = $\sqrt{err^{T} * err}$





18. Mobility Index

The mobility index of a robotic manipulator is a measure of its dexterity or freedom of movement, and it is calculated as:

$$MI = 6n - \sum(b_i - 1)$$

where n is the number of degrees of freedom of the manipulator, b_i is the number of intermediate links between the ith joint and the end effector, and the sum is taken over all joints from 1 to n.

For a 4R manipulator, the number of degrees of freedom is 2, since the end effector can move in two directions, X and Y. The manipulator has four revolute joints (R) arranged in a serial configuration, with no intermediate links between the joints and the end effector.

Therefore, we can calculate the mobility index as follows:
MI = 6(2) - [(1-1) + (2-1) + (3-1) + (4-1)] = 6 - 4 = 2

Thus, the mobility index of a 4R manipulator is 2, indicating that it has a relatively low degree of dexterity and limited freedom of movement.

To calculate the mobility index for RPRP manipulator, we need to determine the number of degrees of freedom and the number of intermediate links for each joint. The first and third joints are revolute, and they have one intermediate link each, which is the link connected to the previous joint and the link connected to the next joint. The second and fourth joints are prismatic, and they have no intermediate links.

Therefore, we can calculate the mobility index as follows:

MI = 6n - [(1-1) + (0-1) + (1-1) + (0-1)] = 6 - 2 = 4

Thus, the mobility index of RPRP manipulator is 4, indicating that it has a relatively high degree of dexterity and freedom of movement.

19. Redundancy Index

The redundancy index in robotics is a measure of how many degrees of freedom a robotic manipulator has beyond the minimum number required to perform a given task. A robotic manipulator with a redundancy index of zero has just enough degrees of freedom to perform the task, while a manipulator with a redundancy index greater than zero has additional degrees of freedom that can be used to optimize the task or accommodate changing conditions.

Redundancy index = (Number of joints) - (Number of dimensions in the workspace)

For a 4R manipulator, the minimum number of degrees of freedom required to reach any point in the workspace is 2. The manipulator has 4 joints, which would provide 4 degrees of freedom. However, the end effector of the manipulator can only move in 2 dimensions,

which means that the manipulator has a redundancy index of 2.

Redundancy index = 4 - 2 = 2

Therefore, the redundancy index for a 4R manipulator is 2.

For a RPRP manipulator, the minimum number of degrees of freedom required to reach any point in the workspace is 2. The manipulator has 4 joints, which would provide 4 degrees of freedom. However, not all of these degrees of freedom are independent due to the constraint that all links are in the same plane.

Redundancy index = 4 - 2 = 2

Therefore, the redundancy index for a RPRP manipulator is 2.

MATLAB CODE

Code for RPRP manipulator

```
%%
clear;close all;clc
%%
dt=1/1000;
p = 200;
t=0:dt:p+dt;
theta = p*pi - t*pi;
r = 4*theta/(p*pi);
x = r.*cos(theta);
y = r.*sin(theta);
dx=diff(x)./dt;
dy=diff(y)./dt;
v=[dx;dy];
%%
q=[0;1;0;1];
manipulability_index = [];
Global_Manipulability_index = 0;
condition_no_index = [];
Local_conditioning_index = [];
weighted_Frobenius_norm = [];
harmonic_mean = [];
stochastic_index = [];
task_space_control_index = [];
dynamic_manipulability= [];
minimum_singular_value_index = [];
global_Isotropy_Index = [];
structural_length_index = [];
```

```
velocity_index = [];
error_index = [];
Product_manipulability_condition_no = [];
Direct_selective_index = [];
kinematic_performance_index = [];
for k=1:1:length(x)-1

   th1 = q(1,k); r1 = q(2,k); th2 = q(3,k); r2 = q(4,k);

   J = jaco_RPRP(th1,r1,th2,r2);
   [xo,yo] = fwd_kin_RPRP(q(:,k));
   [xp,yp] = fwd_kin_RPRP(q(:,k) + 0.05*q(:,k));
   [xm,ym] = fwd_kin_RPRP(q(:,k) - 0.05*q(:,k));
   manipulability_index(1,k) = manipul_index(J);
   Global_Manipulability_index = Global_Manipulability_index +
Glo_manipul_index(manipulability_index(1,k),r(1,k),dt,p);
   condition_no_index(1,k) = Cond_no_index(J);
   Local_conditioning_index(1,k) = 1/(condition_no_index(1,k));
   weighted_Frobenius_norm(1,k) = Frob_norm(J);
   harmonic_mean(1,k) = h_mean(J);
   stochastic_index(1,k) = stochastic(J);
   task_space_control_index(1,k) = task_control(J);
   dynamic_manipulability(1,k) = Dynamic_manipul(th1,r1,th1+th2,r2,J);
   minimum_singular_value_index(1,k) = MSV(J);
   global_Isotropy_Index(1,k) = Global_Iso(J);
   structural_length_index(1,k) = Struc_len_index(p,r1,r2);
   velocity_index(1,k) = vel_index(J);
   Product_manipulability_condition_no(1,k) =
pro_manipul_cond(manipulability_index(1,k),condition_no_index(1,k));
   Direct_selective_index(1,k) = direct_selec_index(manipulability_index(1,k));
   kinematic_performance_index(1,k) = kin_perform_index(J);
   error_index(1,k) = err_index(xo,yo,xp,yp,xm,ym);
   Err=[x(k);y(k)]-[xo;yo];

   if (q(2,k)<=1 && q(2,k)>=0 && q(4,k)<=1 && q(4,k)>=0)
       q(:,k+1)=q(:,k) + 1*pinv(J)*(v(:,k) + 1000*Err)*dt;
   elseif ((q(2,k)>1 || q(2,k)<0) && q(4,k)<=1 && q(4,k)>=0)
       q(:,k+1)=q(:,k) + [1;0;1;1].*(pinv(J)*(v(:,k) + 1000*Err)*dt);
   elseif ((q(4,k)>1 || q(4,k)<0) && q(2,k)<=1 && q(2,k)>=0)
       q(:,k+1)=q(:,k) + [1;1;1;0].*(pinv(J)*(v(:,k) + 1000*Err)*dt);
   else
       q(:,k+1)=q(:,k) + [1;0;1;0].*(pinv(J)*(v(:,k) + 1000*Err)*dt);
   end
end
animate_4r(q,x,y)
%%
x(end)=[];
y(end)=[];
ti = -4:0.003:4;
```

```
[x1,y1] = meshgrid(ti,ti);
%%
F = TriScatteredInterp(x',y',manipulability_index');
z1 = F(x1,y1);
figure('WindowState','maximized')
mesh(x1,y1,z1);
zlim([0 6.5]);
legend('Manipulability index RPRP');
xlabel("x");ylabel("y"),zlabel('Index');
saveas(gcf,'Manipulability_index_RPRP.png')
%%
F = TriScatteredInterp(x',y',condition_no_index');
z1 = F(x1,y1);
figure('WindowState','maximized')
mesh(x1,y1,z1);
legend('condition no index RPRP');
xlabel("x");ylabel("y"),zlabel('Index');
saveas(gcf,'Condition_Number_RPRP.png')
%%
F = TriScatteredInterp(x',y',Local_conditioning_index');
z1 = F(x1,y1);
figure('WindowState','maximized')
mesh(x1,y1,z1);
zlim([0 1]);
legend('Local conditioning index RPRP');
xlabel("x");ylabel("y"),zlabel('Index');
saveas(gcf,'Local_Conditioning_index_RPRP.png')
%%
F = TriScatteredInterp(x',y',weighted_Frobenius_norm');
z1 = F(x1,y1);
figure('WindowState','maximized');
mesh(x1,y1,z1);
zlim([0 3]);
legend('Weighted Frobenis norm RPRP');
xlabel("x");ylabel("y"),zlabel('Index');
saveas(gcf,'Weighted_Frobenius_Norm_RPRP.png')
%%
F = TriScatteredInterp(x',y',harmonic_mean');
z1 = F(x1,y1);
figure('WindowState','maximized');
mesh(x1,y1,z1);
zlim([0 1.45]);
legend('Harmonic mean manipulability index RPRP');
xlabel("x");ylabel("y"),zlabel('Index');
saveas(gcf,'Harmonic_Mean_Manipulability_Index_RPRP.png')
%%
F = TriScatteredInterp(x',y',stochastic_index');
z1 = F(x1,y1);
figure('WindowState','maximized');
```

```
mesh(x1,y1,z1);
zlim([0 4.2]);
legend('Stochastic manipulability index RPRP');
xlabel("x");ylabel("y"),zlabel('Index');
saveas(gcf,'Stochastic_Manipulability_Index_RPRP.png')
%%
F = TriScatteredInterp(x',y',task_space_control_index');
z1 = F(x1,y1);
figure('WindowState','maximized');
mesh(x1,y1,z1);
zlim([0 12]);
legend('Task Space Control index RPRP');
xlabel("x");ylabel("y"),zlabel('Index');
saveas(gcf,'Task_Space_Control_Index_RPRP.png')
%%
F = TriScatteredInterp(x',y',dynamic_manipulability');
z1 = F(x1,y1);
figure('WindowState','maximized');
mesh(x1,y1,z1);
zlim([0 25]);
legend('Dynamic manipulability index RPRP');
xlabel("x");ylabel("y"),zlabel('Index');
saveas(gcf,'Dynamic_Manipulability_Index_RPRP.png')
%%
F = TriScatteredInterp(x',y',minimum_singular_value_index');
z1 = F(x1,y1);
figure('WindowState','maximized');
mesh(x1,y1,z1);
zlim([0 1.8]);
legend('Minimum Singular Value Index RPRP');
xlabel("x");ylabel("y"),zlabel('Index');
saveas(gcf,'Minimum_Singular_Value_Index_RPRP.png')
%%
F = TriScatteredInterp(x',y',global_Isotropy_Index');
z1 = F(x1,y1);
figure('WindowState','maximized');
mesh(x1,y1,z1);
zlim([0 1]);
legend('Global Isotropy Index RPRP');
xlabel("x");ylabel("y"),zlabel('Index');
saveas(gcf,'Global_Isotropy_Index_RPRP.png')
%%
F = TriScatteredInterp(x',y',structural_length_index');
z1 = F(x1,y1);
figure('WindowState','maximized');
mesh(x1,y1,z1);
zlim([0.1 0.2]);
legend('Structural Length_Index RPRP');
xlabel("x");ylabel("y"),zlabel('Index');
```

```
saveas(gcf,'Structural_Length_Index_RPRP.png')
%%
F = TriScatteredInterp(x',y',velocity_index');
z1 = F(x1,y1);
figure('WindowState','maximized');
mesh(x1,y1,z1);
zlim([0 11]);
legend('Velocity Index RPRP');
xlabel("x");ylabel("y"),zlabel('Index');
saveas(gcf,'Velocity_Index_RPRP.png')
%%
F = TriScatteredInterp(x',y',Product_manipulability_condition_no');
z1 = F(x1,y1);
figure('WindowState','maximized');
mesh(x1,y1,z1);
zlim([0 31]);
legend('Product Manipulability Condition Number RPRP');
xlabel("x");ylabel("y"),zlabel('Index');
saveas(gcf,'Product_Manipulability_Condition_Number_RPRP.png')
%%
F = TriScatteredInterp(x',y',Direct_selective_index');
z1 = F(x1,y1);
figure('WindowState','maximized');
mesh(x1,y1,z1);
legend('Direct Selective Index RPRP');
xlabel("x");ylabel("y"),zlabel('Index');
saveas(gcf,'Direct_Selective_Index_RPRP.png')
%%
F = TriScatteredInterp(x',y',kinematic_performance_index');
z1 = F(x1,y1);
figure('WindowState','maximized');
mesh(x1,y1,z1);
legend('Kinematic Performance Index RPRP');
xlabel("x");ylabel("y"),zlabel('Index');
saveas(gcf,'Kinematic_Performance_Index_RPRP.png')
%%
F = TriScatteredInterp(x',y',error_index');
z1 = F(x1,y1);
figure('WindowState','maximized');
mesh(x1,y1,z1);
zlim([0 12]);
legend('Error Index RPRP');
xlabel("x");ylabel("y"),zlabel('Index');
saveas(gcf,'Error_Index_RPRP.png')
%%
function J = jaco_RPRP(th1,r1,th2,r2)
l1=1;l2=1;
```

```
J = [-(l1+r1)*sin(th1) - (l2+r2)*sin(th1+th2), cos(th1), -
(l2+r2)*sin(th1+th2), cos(th1+th2); (l1+r1)*cos(th1) + (l2+r2)*cos(th1+th2),
sin(th1), (l2+r2)*cos(th1+th2), sin(th1+th2);];
end
%%
function [x,y] = fwd_kin_RPRP(q)
l1=1;l2=1;
x = (l1+q(2,:))*cos(q(1,:)) + (l2+q(4,:))*cos(q(1,:)+q(3,:));
y = (l1+q(2,:))*sin(q(1,:)) + (l2+q(4,:))*sin(q(1,:)+q(3,:));
end
%%
function I = manipul_index(J)
I = sqrt(abs(det(J*J')));
end
%%
function I = Glo_manipul_index(meu,r,dt,p)
A = meu*r*dt*pi; % where dt*pi is d(theta)
B = p*pi;
I = A/B;
end
%%
function I = Cond_no_index(J)
% N = [1/2,0;0,1/2];
% I = (1/2)*sqrt(trace(J*N*J')*trace(inv(J)*N*inv(J')));
I = norm(J)*norm(pinv(J));
end
%%
function I = Frob_norm(J)
n = 4;  % n denotes no. of degree of freedom of manipulator
I = sqrt(trace(J*J')/n);
end
%%
function I = h_mean(J)
I = 1/sqrt(trace(inv(J*J')));
end
%%
function I= stochastic(J)
n =4;  % n denotes no. of degree of freedom of manipulator
t= 2;  % t is no. of degree of freedom for task space
if det(J*J')~=0
   I = sqrt(n*t)*h_mean(J);
else
   I = 0;
end
end
%%
function I = task_control(J)
I = sqrt(det(J*J')*Cond_no_index(J));
end
```

```
%%
function I= Dynamic_manipul(th1,r1,th2,r2,J)
m1=0.5; m2=0.5; m3=0.5; m4=0.5;  l1=1; l2=1;
%M matrix or Inertia Matrix
M11 = (m1+m2+m3+m4)*(l1^2) + (m2+m3+m4)*(r1)^2 +2*(m2+m3+m4)*l1*r1;
M12 = 0;
M13 = m4*cos(th1-th2)*r1*r2 + (m3+m4)*l2*(l1)*cos(th1-th2) +
(m3+m4)*l2*r1*cos(th1-th2) + m4*l1*r2*cos(th1-th2);
M14 = -m4*sin(th1-th2)*(l1+r1);
M21 = 0;
M22 = (m2+m3+m4);
M23 = m3*l2*sin(th1-th2) + m4*(l2)*sin(th1-th2) + m4*r2*sin(th1-th2);
M24 = m4*cos(th1-th2);
M31 = m4*cos(th1-th2)*(l1+r1)*(l2+r2) + m3*l2*(l1+r1)*cos(th1-th2);
M32 = m3*l2*sin(th1-th2) + m4*(l2+r2)*sin(th1-th2);
M33 = m3*(l2^2) + m4*(l2^2) + 2*m4*r2*(l2+ r2/2);
M34 = 0;
M41 = -m4*sin(th1-th2)*(l1+r1);
M42 = m4*cos(th1-th2);
M43 = 0;
M44 = m4;
M = [M11 M12 M13 M14 ; M21 M22 M23 M24 ; M31 M32 M33 M34 ; M41 M42 M43 M44];
I = sqrt(det(J*(inv(M*M'))*J'));
end
%%
function I = MSV(J)
I = min(svd(J));
end
%%
function I = Global_Iso(J)
sigma_min = min(svd(J));
sigma_max = max(svd(J));
I = sigma_min/sigma_max;
end
%%
function I = Struc_len_index(p,r1,r2)
I = (2+r1+r2)/sqrt(p*pi);
end
%%
function I = vel_index(J)
x_dot  = J*[1;1;1;1];
I = sqrt(x_dot'*x_dot);
end
%%
function I = pro_manipul_cond(m,k)
I = m*k;
end
%%
function I = direct_selec_index(m)
```

```
I = 1/m;
end
%%
function I = kin_perform_index(J)
I = (1/2)*sqrt(trace(J*J')*trace(inv(J*J')));
end
%%
function I = err_index(xo,yo,xp,yp,xm,ym)
err = [xo;yo;xo;yo] - [xp;yp;xm;ym];
I = sqrt(err'*err);
end
%%
function [] = animate_4r(q,xt,yt)
xi=q;
figure('WindowState','maximized')
%pause(7)
c=1;
for i=1:10:length(xi)
   theta1 = xi(1,i);
   lr1 = xi(2,i);
   theta2 = xi(3,i);
   lr2 = xi(4,i);
   %Input the l length
   l1 = 1;
   l2 = 1;
   % Homogeneus transformation matrix
   H01 = [cos(theta1) -sin(theta1) 0 l1*cos(theta1);sin(theta1) cos(theta1) 0
l1*sin(theta1);0 0 1 0;0 0 0 1]; %Frame 0 to 1 tranformation
   H12 = [1 0 0 lr1;0 1 0 0;0 0 1 0;0 0 0 1]; %Frame 1 to 2 tranformation
   H23 = [cos(theta2) -sin(theta2) 0 l2*cos(theta2);sin(theta2) cos(theta2) 0
l2*sin(theta2);0 0 1 0;0 0 0 1]; %Frame 2 to 3 tranformation
   H34 = [1 0 0 lr2;0 1 0 0;0 0 1 0;0 0 0 1]; %Frame 3 to 4 tranformation
   H02 = H01*H12;      %Frame 0 to 2 tranformation
   H03 = H02*H23;
   H04 = H03*H34;
   P1 = [H01(1,4) H01(2,4)];
   P2 = [H02(1,4) H02(2,4)];
   P3 = [H03(1,4) H03(2,4)];
   P4 = [H04(1,4) H04(2,4)];
   P4x(c,:) = P4(1,1);
   P4y(c,:) = P4(1,2);
   plot(xt,yt,'--r')
   hold on
   plot(P1(1),P1(2),'ok','LineWidth',1)
   plot(P4x,P4y,'k')
   plot(P2(1),P2(2),'ok','LineWidth',1)
   plot(P3(1),P3(2),'ok','LineWidth',1)
   plot(P4(1),P4(2),'ok','LineWidth',1)
```

```
    plot(0,0,'ok','LineWidth',3)
    xlim([-5 5])
    ylim([-5 5])
    axis square;
    grid minor
    plot([0 P1(1)], [0 P1(2)],'g','LineWidth',2)
    plot([P1(1) P2(1)], [P1(2) P2(2)],'b','LineWidth',2)
    plot([P2(1) P3(1)], [P2(2) P3(2)],'y','LineWidth',2)
    plot([P3(1) P4(1)], [P3(2) P4(2)],'c','LineWidth',2)
    xlabel('X axis (m)','Interpreter','latex')
    ylabel('Y axis (m)','Interpreter','latex')
    set(gca,'FontSize',18)
    drawnow
    hold off
    c=c+1;
end
end
```

Code for 4R manipulator

```
%%
clear all;close all;clc
%%
dt=1/1000;
p = 200;
t=0:dt:p+dt;
theta = p*pi - t*pi;
r = 4*theta/(p*pi);
x = r.*cos(theta);
y = r.*sin(theta);
dx=diff(x)./dt;
dy=diff(y)./dt;
v=[dx;dy];
%%
q=[0;0;0;0];
manipulability_index = [];
Global_Manipulability_index = 0;
condition_no_index = [];
Local_conditioning_index = [];
weighted_Frobenius_norm = [];
harmonic_mean = [];
stochastic_index = [];
task_space_control_index = [];
dynamic_manipulability = [];
minimum_singular_value_index = [];
global_Isotropy_Index = [];
structural_length_index = [];
velocity_index = [];
```

```
error_index = [];
Product_manipulability_condition_no = [];
Direct_selective_index = [];
kinematic_performance_index = [];
for k=1:1:length(x)-1
   th1=q(1,k);th2=q(2,k);th3=q(3,k);th4 =q(4,k);
   J = jaco_4(th1,th2,th3,th4);

   [xo,yo] = fwd_kin_4(q(:,k));
   [xp,yp] = fwd_kin_4(q(:,k) + 0.05*q(:,k));
   [xm,ym] = fwd_kin_4(q(:,k) - 0.05*q(:,k));
   manipulability_index(1,k) = manipul_index(J);
   Global_Manipulability_index = Global_Manipulability_index +
Glo_manipul_index(manipulability_index(1,k),r(1,k),dt,p);
   condition_no_index(1,k) = Cond_no_index(J);
   Local_conditioning_index(1,k) = 1/(condition_no_index(1,k));
   weighted_Frobenius_norm(1,k) = Frob_norm(J);
   harmonic_mean(1,k) = h_mean(J);
   stochastic_index(1,k) = stochastic(J);
   task_space_control_index(1,k) = task_control(J);
   dynamic_manipulability(1,k) =
Dynamic_manipul(th1,th1+th2,th1+th2+th3,th1+th2+th3+th4,J);
   minimum_singular_value_index(1,k) = MSV(J);
   global_Isotropy_Index(1,k) = Global_Iso(J);
   structural_length_index(1,k) = Struc_len_index(p);
   velocity_index(1,k) = vel_index(J);
   Product_manipulability_condition_no(1,k) =
pro_manipul_cond(manipulability_index(1,k),condition_no_index(1,k));
   Direct_selective_index(1,k) = direct_selec_index(manipulability_index(1,k));
   kinematic_performance_index(1,k) = kin_perform_index(J);
   error_index(1,k) = err_index(xo,yo,xp,yp,xm,ym);

   Err=[x(k);y(k)]-[xo;yo];
   q(:,k+1)=q(:,k) + 1*pinv(J)*(v(:,k) + 1000*Err)*dt ;
end
animate_4r(q,x,y)
%%
x(end)=[];
y(end)=[];
ti = -4:0.003:4;
[x1,y1] = meshgrid(ti,ti);
%%
F = TriScatteredInterp(x',y',manipulability_index');
z1 = F(x1,y1);
figure('WindowState','maximized')
mesh(x1,y1,z1);
zlim([0 6.5]);
legend('Manipulability index 4R');
xlabel("x");ylabel("y"),zlabel('Index');
```

```
saveas(gcf,'Manipulability_index_4R.png')
%%
F = TriScatteredInterp(x',y',condition_no_index');
z1 = F(x1,y1);
figure('WindowState','maximized')
mesh(x1,y1,z1./500);
zlim([0 100]);
legend('condition no index 4R');
xlabel("x");ylabel("y"),zlabel('Index');
saveas(gcf,'Condition_Number_4R.png')
%%
F = TriScatteredInterp(x',y',Local_conditioning_index');
z1 = F(x1,y1);
figure('WindowState','maximized')
mesh(x1,y1,z1);
zlim([0 1]);
legend('Local conditioning index 4R');
xlabel("x");ylabel("y"),zlabel('Index');
saveas(gcf,'Local_Conditioning_index_4R.png')
%%
F = TriScatteredInterp(x',y',weighted_Frobenius_norm');
z1 = F(x1,y1);
figure('WindowState','maximized');
mesh(x1,y1,z1);
zlim([0 3]);
legend('Weighted Frobenis norm 4R');
xlabel("x");ylabel("y"),zlabel('Index');
saveas(gcf,'Weighted_Frobenius_Norm_4R.png')
%%
F = TriScatteredInterp(x',y',harmonic_mean');
z1 = F(x1,y1);
figure('WindowState','maximized');
mesh(x1,y1,z1);
zlim([0 1.45]);
legend('Harmonic mean manipulability index 4R');
xlabel("x");ylabel("y"),zlabel('Index');
saveas(gcf,'Harmonic_Mean_Manipulability_Index_4R.png')
%%
F = TriScatteredInterp(x',y',stochastic_index');
z1 = F(x1,y1);
figure('WindowState','maximized');
mesh(x1,y1,z1);
zlim([0 4.2]);
legend('Stochastic manipulability index 4R');
xlabel("x");ylabel("y"),zlabel('Index');
saveas(gcf,'Stochastic_Manipulability_Index_4R.png')
%%
F = TriScatteredInterp(x',y',task_space_control_index');
z1 = F(x1,y1);
```

```
figure('WindowState','maximized');
mesh(x1,y1,z1);
zlim([0 12]);
legend('Task Space Control index 4R');
xlabel("x");ylabel("y"),zlabel('Index');
saveas(gcf,'Task_Space_Control_Index_4R.png')
%%
F = TriScatteredInterp(x',y',dynamic_manipulability');
z1 = F(x1,y1);
figure('WindowState','maximized');
mesh(x1,y1,z1);
zlim([0 25]);
legend('Dynamic manipulability index 4R');
xlabel("x");ylabel("y"),zlabel('Index');
saveas(gcf,'Dynamic_Manipulability_Index_4R.png')
%%
F = TriScatteredInterp(x',y',minimum_singular_value_index');
z1 = F(x1,y1);
figure('WindowState','maximized');
mesh(x1,y1,z1);
zlim([0 1.8]);
legend('Minimum Singular Value Index 4R');
xlabel("x");ylabel("y"),zlabel('Index');
saveas(gcf,'Minimum_Singular_Value_Index_4R.png')
%%
F = TriScatteredInterp(x',y',global_Isotropy_Index');
z1 = F(x1,y1);
figure('WindowState','maximized');
mesh(x1,y1,z1);
zlim([0 1]);
legend('Global Isotropy Index 4R');
xlabel("x");ylabel("y"),zlabel('Index');
saveas(gcf,'Global_Isotropy_Index_4R.png')
%%
F = TriScatteredInterp(x',y',structural_length_index');
z1 = F(x1,y1);
figure('WindowState','maximized');
mesh(x1,y1,z1);
zlim([0.1 0.2]);
legend('Structural Length Index 4R');
xlabel("x");ylabel("y"),zlabel('Index');
saveas(gcf,'Structural_Length_Index_4R.png')
%%
F = TriScatteredInterp(x',y',velocity_index');
z1 = F(x1,y1);
figure('WindowState','maximized');
mesh(x1,y1,z1);
zlim([0 11]);
legend('Velocity Index 4R');
```

```
xlabel("x");ylabel("y"),zlabel('Index');
saveas(gcf,'Velocity_Index_4R.png')
%%
F = TriScatteredInterp(x',y',Product_manipulability_condition_no');
z1 = F(x1,y1);
figure('WindowState','maximized');
mesh(x1,y1,z1);
zlim([0 31]);
legend('Product Manipulability Condition Number 4R');
xlabel("x");ylabel("y"),zlabel('Index');
saveas(gcf,'Product_Manipulability_Condition_Number_4R.png')
%%
F = TriScatteredInterp(x',y',Direct_selective_index');
z1 = F(x1,y1);
figure('WindowState','maximized');
mesh(x1,y1,z1);
legend('Direct Selective Index 4R');
xlabel("x");ylabel("y"),zlabel('Index');
saveas(gcf,'Direct_Selective_Index_4R.png')
%%
F = TriScatteredInterp(x',y',kinematic_performance_index');
z1 = F(x1,y1);
figure('WindowState','maximized');
mesh(x1,y1,z1);
legend('Kinematic Performance Index 4R');
xlabel("x");ylabel("y"),zlabel('Index');
saveas(gcf,'Kinematic_Performance_Index_4R.png')
%%
F = TriScatteredInterp(x',y',error_index');
z1 = F(x1,y1);
figure('WindowState','maximized');
mesh(x1,y1,z1);
zlim([0 12]);
legend('Error Index 4R');
xlabel("x");ylabel("y"),zlabel('Index');
saveas(gcf,'Error_Index_4R.png')
%%
function J = jaco_4(th1,th2,th3,th4)
l1=1;l2=1;l3=1;l4=1;
J=[-l1*sin(th1)-l2*sin(th2+th1)-l3*sin(th3+th2+th1)-l4*sin(th4+th3+th2+th1),-l2
*sin(th2+th1)-l3*sin(th3+th2+th1)-l4*sin(th4+th3+th2+th1),-l3*sin(th1+th2+th3)-
l4*sin(th4+th3+th2+th1),-l4*sin(th4+th3+th2+th1);l1*cos(th1)+l2*cos(th2+th1)+l3
*cos(th3+th2+th1)+l4*cos(th4+th3+th2+th1),l2*cos(th2+th1)+l3*cos(th3+th2+th1)+l
4*cos(th4+th3+th2+th1),l3*cos(th1+th2+th3)+l4*cos(th4+th3+th2+th1),l4*cos(th4+t
h3+th2+th1)];
end
%%
function [x,y] = fwd_kin_4(q)
l1=1;l2=1;l3=1;l4=1;
```

```
x=l1*cos(q(1,:)) + l2*cos(q(1,:)+q(2,:)) +
l3*cos(q(1,:)+q(2,:)+q(3,:))+l4*cos(q(1,:)+q(2,:)+q(3,:)+q(4,:));
y=l1*sin(q(1,:)) + l2*sin(q(1,:)+q(2,:)) +
l3*sin(q(1,:)+q(2,:)+q(3,:))+l4*sin(q(1,:)+q(2,:)+q(3,:)+q(4,:));
end
%%
function I = manipul_index(J)
I = sqrt(abs(det(J*J')));
end
%%
function I = Glo_manipul_index(meu,r,dt,p)
A = meu*r*dt*pi; % where dt*pi is d(theta)
B = p*pi;
I = A/B;
end
%%
function I = Cond_no_index(J)
% N = [1/2,0;0,1/2];
% I = (1/2)*sqrt(trace(J*N*J')*trace(inv(J)*N*inv(J')));
I = norm(J)*norm(pinv(J));
end
%%
function I = Frob_norm(J)
n = 4;     % n denotes no. of degree of freedom of manipulator
I = sqrt(trace(J*J')/n);
end
%%
function I = h_mean(J)
I = 1/sqrt(trace(inv(J*J')));
end
%%
function I = stochastic(J)
n =4;    % n denotes no. of degree of freedom of manipulator
t= 2;     % t is no. of degree of freedom for task space
if det(J*J')~=0
   I = sqrt(n*t)*h_mean(J);
else
   I = 0;
end
end
%%
function I = task_control(J)
I = sqrt(det(J*J')*Cond_no_index(J));
end
%%
function I= Dynamic_manipul(th1,th2,th3,th4,J)
m1=0.5; m2=0.5; m3=0.5; m4=0.5;  l1=1; l2=1; l3=1; l4=1;
%M matrix or Inertia Matrix
M11 = l1^2*m1 + l1^2*m2 + l1^2*m3 + l1^2*m4;
```

```
M12 = l1*l2*m2*cos(th1 - th2) + l1*l2*m3*cos(th1 - th2) + l1*l2*m4*cos(th1 -
th2);
M13 = l1*l3*m3*cos(th1 - th3) + l1*l3*m4*cos(th1 - th3);
M14 = l1*l4*m4*cos(th1 - th4);
M21 = (m2*(2*l1*l2*cos(th1)*cos(th2) + 2*l1*l2*sin(th1)*sin(th2)))/2 +
(m3*(2*l1*l2*cos(th1)*cos(th2) + 2*l1*l2*sin(th1)*sin(th2)))/2 +
(m4*(2*l1*l2*cos(th1)*cos(th2) + 2*l1*l2*sin(th1)*sin(th2)))/2;
M22 = (m2*(2*l2^2*cos(th2)^2 + 2*l2^2*sin(th2)^2))/2 + (m3*(2*l2^2*cos(th2)^2 +
2*l2^2*sin(th2)^2))/2 + (m4*(2*l2^2*cos(th2)^2 + 2*l2^2*sin(th2)^2))/2;
M23 = (m3*(2*l2*l3*cos(th2)*cos(th3) + 2*l2*l3*sin(th2)*sin(th3)))/2 +
(m4*(2*l2*l3*cos(th2)*cos(th3) + 2*l2*l3*sin(th2)*sin(th3)))/2;
M24 = (m4*(2*l2*l4*cos(th2)*cos(th4) + 2*l2*l4*sin(th2)*sin(th4)))/2;
M31 = (m3*(2*l1*l3*cos(th1)*cos(th3) + 2*l1*l3*sin(th1)*sin(th3)))/2 +
(m4*(2*l1*l3*cos(th1)*cos(th3) + 2*l1*l3*sin(th1)*sin(th3)))/2;
M32 = (m3*(2*l2*l3*cos(th2)*cos(th3) + 2*l2*l3*sin(th2)*sin(th3)))/2 +
(m4*(2*l2*l3*cos(th2)*cos(th3) + 2*l2*l3*sin(th2)*sin(th3)))/2;
M33 = (m3*(2*l3^2*cos(th3)^2 + 2*l3^2*sin(th3)^2))/2 + (m4*(2*l3^2*cos(th3)^2 +
2*l3^2*sin(th3)^2))/2;
M34 = (m4*(2*l3*l4*cos(th3)*cos(th4) + 2*l3*l4*sin(th3)*sin(th4)))/2;
M41 = (m4*(2*l1*l4*cos(th1)*cos(th4) + 2*l1*l4*sin(th1)*sin(th4)))/2;
M42 = (m4*(2*l2*l4*cos(th2)*cos(th4) + 2*l2*l4*sin(th2)*sin(th4)))/2;
M43 = (m4*(2*l3*l4*cos(th3)*cos(th4) + 2*l3*l4*sin(th3)*sin(th4)))/2;
M44 = (m4*(2*l4^2*cos(th4)^2 + 2*l4^2*sin(th4)^2))/2;
M = [M11 M12 M13 M14 ; M21 M22 M23 M24 ; M31 M32 M33 M34 ; M41 M42 M43 M44];
I = sqrt(det(J*(inv(M*M'))*J'));
end
%%
function I = MSV(J)
I = min(svd(J));
end
%%
function I = Global_Iso(J)
sigma_min = min(svd(J));
sigma_max = max(svd(J));
I = sigma_min/sigma_max;
end
%%
function I = Struc_len_index(p)
I = 4/sqrt(p*pi);
end
%%
function I = vel_index(J)
x_dot  = J*[1;1;1;1];
I = sqrt(x_dot'*x_dot);
end
%%
function I = pro_manipul_cond(m,k)
I = m*k;
end
```

```
%%
function I = direct_selec_index(m)
I = 1/m;
end
%%
function I = kin_perform_index(J)
I = (1/2)*sqrt(trace(J*J')*trace(inv(J*J')));
end
%%
function I = err_index(xo,yo,xp,yp,xm,ym)
err = [xo;yo;xo;yo] - [xp;yp;xm;ym];
I = sqrt(err'*err);
end
%%
function []=animate_4r(q,xt,yt)
xi=q;
figure('WindowState','maximized')
c=1;
for i=1:30:length(xi)
   theta1=xi(1,i);
   theta2=xi(2,i);
   theta3=xi(3,i);
   theta4=xi(4,i);
   l1=1; %Input the l length
   l2=1; %Input the l length
   l3=1;
   l4 = 1;
   % Homogeneus transformation matrix
   H01 = [cos(theta1) -sin(theta1) 0 l1*cos(theta1);sin(theta1) cos(theta1) 0
l1*sin(theta1);0 0 1 0;0 0 0 1]; %Frame 0 to 1 tranformation
   H12 = [cos(theta2) -sin(theta2) 0 l2*cos(theta2);sin(theta2) cos(theta2) 0
l2*sin(theta2);0 0 1 0;0 0 0 1]; %Frame 1 to 2 tranformation
   H23 = [cos(theta3) -sin(theta3) 0 l3*cos(theta3);sin(theta3) cos(theta3) 0
l3*sin(theta3);0 0 1 0;0 0 0 1]; %Frame 2 to 3 tranformation
   H34 = [cos(theta4) -sin(theta4) 0 l4*cos(theta4);sin(theta4) cos(theta4) 0
l4*sin(theta4);0 0 1 0;0 0 0 1]; %Frame 3 to 4 tranformation
   H02=H01*H12;      %Frame 0 to 2 tranformation
   H03=H01*H12*H23;
   H04 = H03*H34;
   P1=[H01(1,4) H01(2,4)];
   P2=[H02(1,4) H02(2,4)];
   P3=[H03(1,4) H03(2,4)];
   P4 =[H04(1,4) H04(2,4)];
   P4x(c,:)=P4(1,1);
   P4y(c,:)=P4(1,2);
   plot(xt,yt,'--r')
   hold on
   plot(P1(1),P1(2),'ok','LineWidth',1)
   plot(P4x,P4y,'k')
```

```
    plot(P2(1),P2(2),'ok','LineWidth',1)
    plot(P3(1),P3(2),'ok','LineWidth',1)
    plot(P4(1),P4(2),'ok','LineWidth',1)

    plot(0,0,'ok','LineWidth',3)
    xlim([-5 5])
    ylim([-5 5])
    axis square;
    grid minor
    plot([0 P1(1)], [0 P1(2)],'g','LineWidth',2)
    plot([P1(1) P2(1)], [P1(2) P2(2)],'b','LineWidth',2)
    plot([P2(1) P3(1)], [P2(2) P3(2)],'r','LineWidth',2)
    plot([P3(1) P4(1)], [P3(2) P4(2)],'g','LineWidth',2)
    xlabel('X axis (m)','Interpreter','latex')
    ylabel('Y axis (m)','Interpreter','latex')
    set(gca,'FontSize',18)
    drawnow
    hold off
    c=c+1;
end
end
```