International Journal of Advance Research, IJOAR .org
Volume 4, Issue 1, January 2016, Online: ISSN 2320-9135

WORKSPACE AGILITY FOR ROBOTIC ARM

Karna Patel

Karna Patel is currently pursuing bachelor's degree program in mechanical engineering in Nirma University, India.
PH: +91-8511115136.
E-mail: karnapatel001.kp@gmail.com

ABSTRACT

With advance in field of robotics, more appplications are being explored which can be undertaken by robotic arms. Robotic Arms are extensively used in manufacturing industries for enhancing productivity. To further improve the productivity in manufacturing industries, selection of proper workspace parameters and implementation of an optimal path for Robot Arms is necessary. For this purpose, agility function is used for computing the agility of the Robotic Arm at various points in the Workspace. Agility at a point in the workspace can be different for different direction and thus exhibits anisotropic characteristics. On basis of this analysis, different agility zones, iso-agility lines and iso-agility points can be computed in the workspace of the Robot Arm. These parameters are used for obtaining an optimal path from one point to another in the workspace thereby enhancing productivity of the process. This analysis can also be used to find out optimum parameters of the Robot Arm with respect to the workspace requirements.

KeyWords
Agility, GUI, Kinematics, Optimum Path, Robotic Arm, Trajectory Generation, Workspace Envelope.

INTRODUCTION

The discussion in this paper is regarding the development of a method for enhancement of the productivity of manufacturing processes employing Robot Arms. To enhance productivity of a Robot Arm, selection of proper workspace parameters is important. The response of the Robot Arm would not be uniform throughout the workspace. While moving in some region of the workspace, the Robot Arm could have better response or could consume less time as compared to some other region for moving the same distance. So the equipment setup, arrangement and workspace parameters should be selected in a way so that maximum of the more responsive region of the workspace is utilized. A 3 DOF Robot Arm is taken into account for analyzing agility in the workspace. A mathematical function is defined for Agility. Based on it, agility is computed at various grid points in the workspace. From this information, iso-agility lines and points can be computed. These help in identifying regions with similar response or agility in the workspace. Then using resultant agility at each point, a vector field is generated in the Robot workspace. Matlab is utilized for calculating agility at the grid points and plotting the vector field. Given the start point and end point, the objective now is to choose a path in the vector field to maximize the agility. This will minimize the process time and in turn maximize productivity. Matlab is utilized for calculating the optimum path with the help of optimization toolbox.

ROBOT ARM

A simple 4 DOF Robot Arm is considered for the analysis. All the joints are revolute. Thus it is 4-R type of manipulator. The different degrees of freedom and joint motion are reffered in Figure 1 and Figure 2. R_1, R_2, R_3 and R_4 are the four revolute joints. The axis of R_1 is along Y-axis and is at the centre of the workspace. The axes of R_2, R_3 and R_4 are parallel to each other.

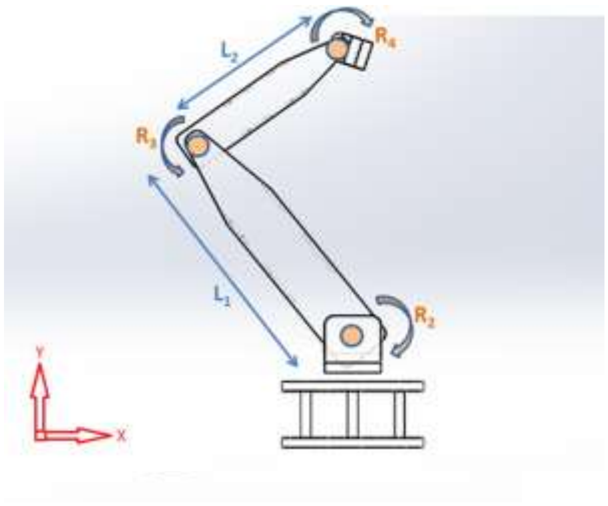

Figure 1: Joint Motion Orthogonal View

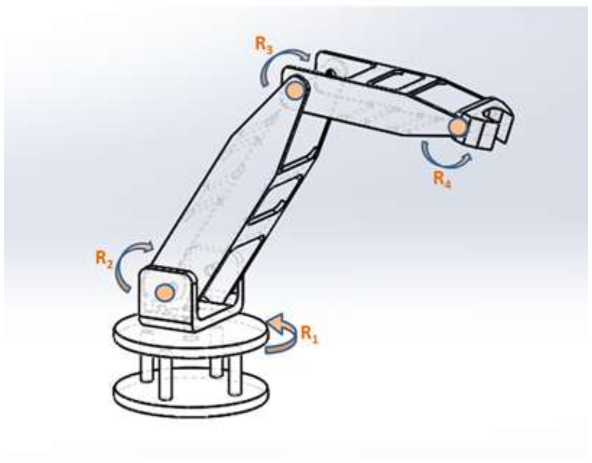

Figure 2: Joint Motion 3D View

CAD Model

The 3d CAD Model of Robot Arm can be generated in any CAD modelling software. Solidworks is utilized to generate 3d CAD Model of the Robot Arm reffered in Figure 3.

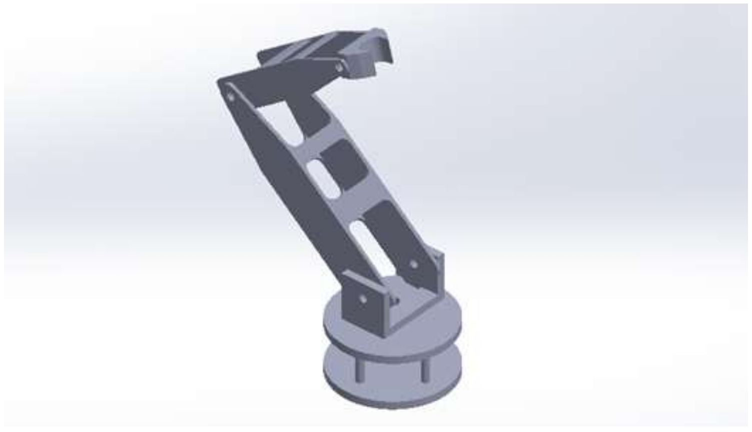

Figure 3: 3D CAD Model

Forward Kinematics

Forward Kinematics is a type of Kinematics. It deals with the problems for finding position and orientation of the end-effector of the Robotic Arm if the joint configurations are given. Usually servo motors are used to actuate the joints of robotic arms. Thus the joint configurations are known at all the times. So forward kinematics is used to find the position and orientation of the end-effector from the position of the servo motors used at joints of the kinematic chain. There are two basic methods for solving forward kinematics problem. One of them uses graphical approach while other uses D-H parameters.
The graphical method is simplest for calculating the position of end-effector. Here the calculation is done only for the position of the end-effector and not for its orientation. In XY plane consider a generalized state of the kinematic chain. Figure 4 shows the angles made by the links. Using these angles, with simple vector algebra, we can present the following equations;

$$x = l_1 cos\vartheta_1 + l_2 cos(\vartheta_1 + \vartheta_2) \qquad (1)$$

$$y = l_1 sin\vartheta_1 + l_2 sin(\vartheta_1 + \vartheta_2) \qquad (2)$$

Here l_1 and l_2 are the link lengths and The θ_1 and θ_2 are the angles made by these links with reference to previous link as shown in Figure 4. These equations can be used to calculate the end-effector positionby using a numerical computing environment like Matlab.

Inverse Kinematics

Inverse Kinematics is used for finding the joint angles for a specific position or orientation of the end-effector. In Figure 4, θ_1 and θ_2 are the joint angles for all mechanical configurations possible. Inverse kinematics is required for computation of agility at various points in the workspace. There may exist more than one, possible solutions for a given problem when solved using inverse kinematics. In our problem we are only interested in the position of end-effector and not its orientation.

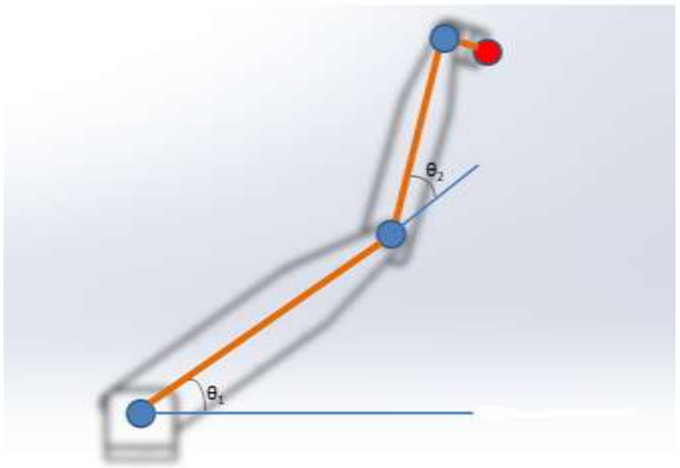

Figure 4: Generic Orientation of Links

Let x and y be the co-ordinates of the end-effector in XY plane,

$$x = l_1 c_1 + l_2 c_{1+2} \qquad (3)$$

$$y = l_1 s_1 + l_2 s_{1+2} \qquad (4)$$

On squaring and simplifying these equations, we get;

$$\Theta_2 = arccos\left(\frac{x^2 + y^2 - l_1^2 - l_2^2}{2 l_1 l_2}\right) \qquad (5)$$

$$\Theta 1 = arcsin\left(\frac{y(l_1 + l_2 c_2) - x l_2 s_2}{x^2 + y^2}\right) \qquad (6)$$

As seen in adove equation, Θ_1 is dependent on Θ_2 and thus we need to calculate the value of Θ_2 first. Thus from equation 5 and 6 we can calculate the the joint angles for a particular position of the end-effector.

Work Envelope

The workspace of a robot manipulator can be described as the set of points that can be reached by its end affector considering a physically possible mechanical configuration. The workspace of Robot Arm is reffered in Figure 5. It shows a plot in the vertical plane XY. This Figure indicates the reach of the Robotic Arm given the constraints of each of its joints. The constraints for each joint are given below in the Table 1.

Joint	Type	Range
R_1	Revolute	-Π to Π
R_2	Revolute	-Π/2 to Π/2
R_3	Revolute	-2Π/3 to 2Π/3
R_4	Revolute	-Π to Π

Table 1: Joint Constraints

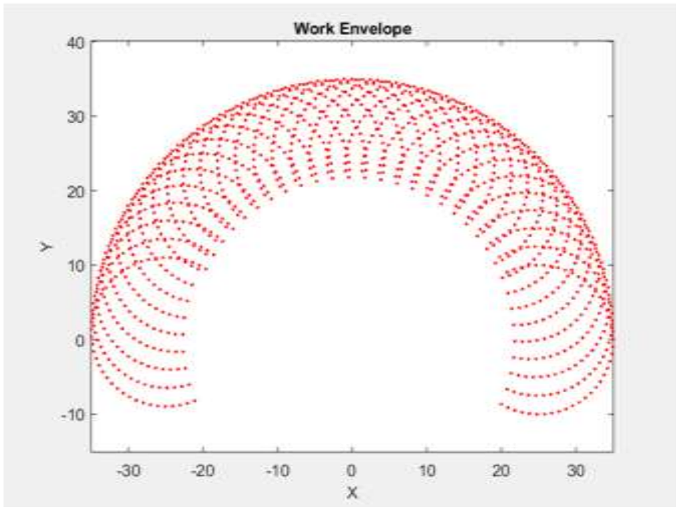

Figure 5: Workspace in XY plane

Agility Function

Agility can be put forward as the ability of the system to adapt in a quick manner to situations. It is a measure of quickness of the system. Thus faster the response, more will be the agility of the system. The agility in terms of a Robot Arm would be the measure of quickness of physical movement. While the quickness of movement of the arm is dependent on the mechanical configuration and parameters of the actuator. Thus agility changes with change in mechanical configuration and actuator parameters at a given point in the workspace of the Robot Arm. For a given Robot Arm the agility varies in the workspace. As the agility depends on the mechanical configuration, it will depend on the direction of motion exhibiting anisotropic characteristics. Here the analysis is done in Cartesian system by defeining agility for 3 axes – X, Y and Z. Agility at a point in the workspace can be different along different axes. The agility at point P along X-axis is denoted by $(A)^p{}_X$. Similarly $(A)^p{}_Y$ and $(A)^p{}_Z$ denote the agility at a point P along Y-axis and Z-axis respectively. Here (A) is used to denote agility.

Mathematical Function

In general agility can be mathematically given by the summation of the agile properties squared. The mathematical function for agility at a point in workspace can be written as;

$$(A) = \sum_{k=1}^{n} c_k a_k^2 \qquad (7)$$

Here a_k is the kth agile property and c_k is the respective weight for the kth agile property.
As agility is measure of quickness of physical movement of the Robot Arm, the mathematical function should contain the term of velocity or time. Velocity can prove to be a better choice being a vector. As velocity is a vector, agility will also be a vector. Thus here the only agile property is velocity. Thus the mathematical function for agility at point P can be rewritten as;

$$(A) = c_p\, v(t)_p^{\,2} \qquad (8)$$

Here $v(t)_p$ is the velocity at point P as a function of time and c_p is the weight at point P.

Assumptions

Few assumptions are taken in order to carry out the analysis successfully. The hypothesis is can be verified after making the following assumptions.

1. The response of the system is considered continuous to reduce the complexity of the problem. The angular velocity delivered by the motors is assumed to be a constant value. The change in the angular velocity(ω) and angular acceleration(α) are neglected at the start and end of the motion.
2. The total agility at a point in workspace is assumed to be vector resultant of the agilities at that point.
3. The grid points in the workspace are assumed to be located very nearby to each other in a dense fashion. Thus linear interpolation would be a good measure to compute the values between two grid points.
4. The torque delivered by motors is constant with time and there is no lag in motion due to change in loading conditions.
5. The value of agility at a point for the motion of Robot Arm radially inward with respect to axis of revolute joint (R1) is same to that of the value of agility for radially outward motion. Similarly the value of agility for a point for clockwise motion with respect to the base or axis of revolute joint (R_1) is same to that of the value of agility for anti clockwise motion.

COMPUTING AGILITY

Agility is computed at all necessary grid points by using the mathematical function of agility. It has to be computed in all three directions. It is not required to compute the agility at all grid points in the workspace. For the Robot Arm, it can be understood that the plot will be symmetrical about the base or axis of revolute joint (R_1). Thus computing for the grid points on one side can be skipped. This can save some computation time. The computation of agility along X-axis and Y-axis is complex as compared to computation of agility along Z-axis. This is because here two revolute joints are responsible for the considered mechanical configuration while computing along X-axis and Y-axis. While only one revolute joint is responsible for considered mechanical configuration while computing along Z-axis.

Agility along X-axis

The computation of the agility along X-axis is initiated from the bottom of the workspace. The computation is initiated at the minimum gridstep in direction of Y-axis. At this step the agility for all the points excluding symmetrical points on X-axis in XY plane is computed. The procedure is repeated after adding a gridstep in direction of Y-axis. Thus all points are computed at a particular value of Y co-ordinate on XY plane. Number of points for computation change with addition of a gridstep each time depending on the nature of workspace. In the case of the Robot Arm, the workspace exhibits a circular symmetry. Thus the value of agility at all points for radially inward or outward motion will be same as that of the value of the agility along X-axis computed on XY plane for respective gridsteps or value of Y co-ordinate. This is referred in Figure 6 and 7.

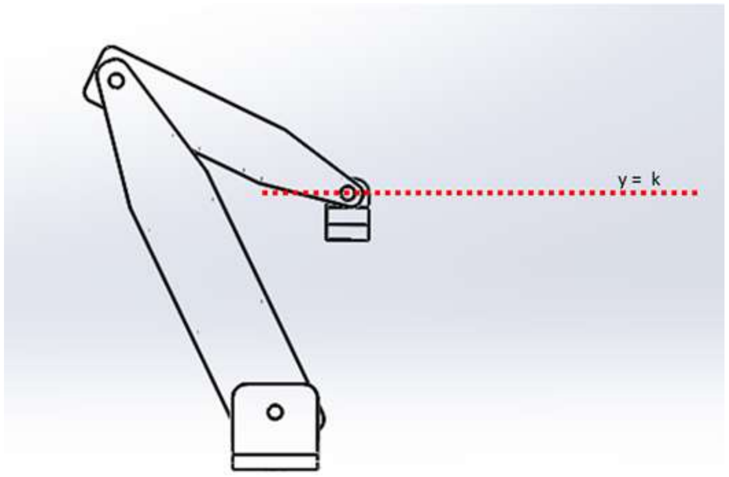

Figure 6: Computing Agility along X-axis

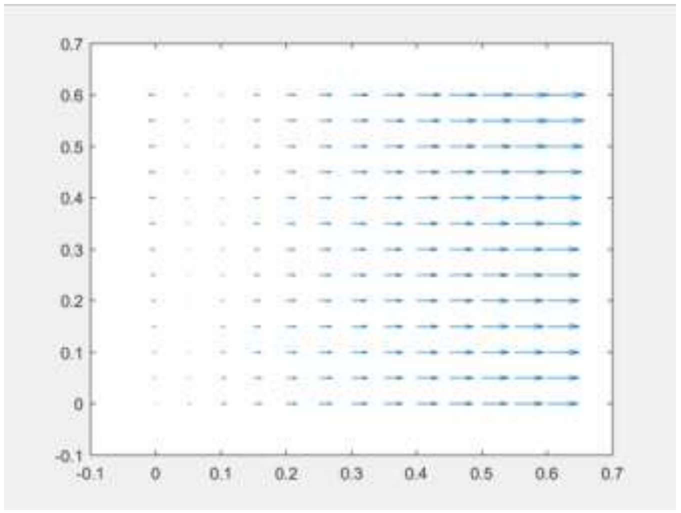

Figure 7: Agility along X axis (XY plane)

Agility along Y-axis

The procedure is similar to that for computation of agility along X-axis. The addition of gridstep here will in direction of X-axis each time. Thus all the points at a particular value of X co-ordinate will be computed in the XY plane and then after addition of the gridstep, the same procedure will be repeated for a different value of X co-ordinate. Here the value of agility at all points for motion along the axis will be same as that of the value of the agility along Y-axis computed on XY plane for respective gridsteps or value of X co-ordinate. This is referred in Figure 8 and 9.

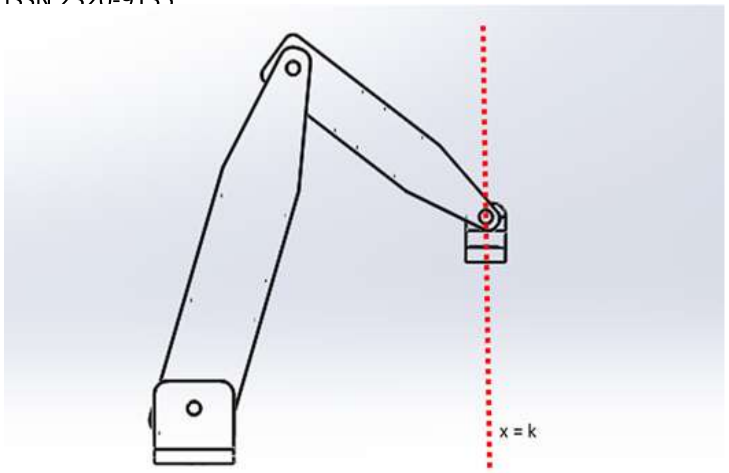

Figure 8: Computing Agility along Y-axis

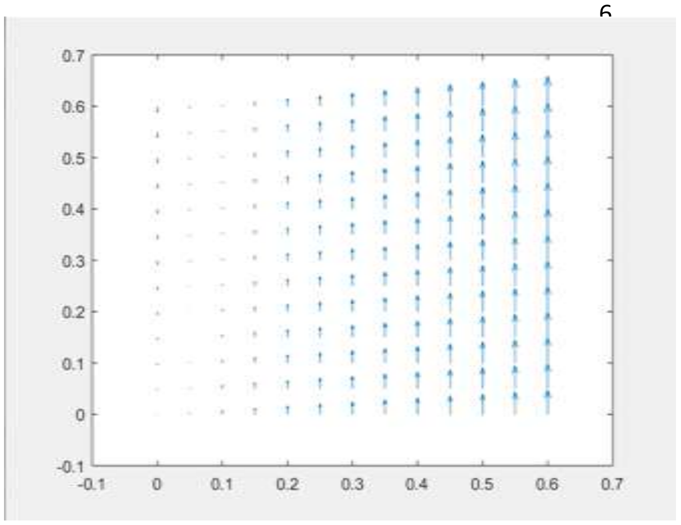

Figure 9: Agility along Y axis (XY plane)

Agility along Z-axis

The velocity along Z-axis is proportional to the distance of end affector from the axis of revolute joint (R_1). Further there is only one revolute joint responsible for the mechanical configuration. Thus the computation is easier. Agility along Z-axis is computed at all the points excluding the symmetrical points on XY plane starting from inner centre and going radially outward. So the agility will be same for all points at a fixed radius from the central axis. But it will increase with increase in radius. This is refered in Figure 10.

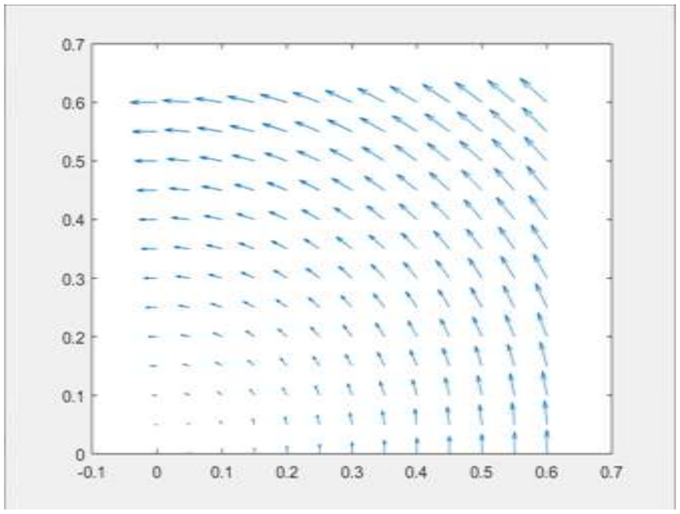

Figure 10: Agility along Z axis (XZ plane)

Mathematics Used

As from the equation 3 and equation 4;

$x = l_1 c_1 + l2 c_{1+2}$ (9)

$y = l_1 s_1 + l_2 s_{1+2}$ (10)

On differentiation, we get;

$\dot{x} = l_1 s_1 \dot{\vartheta}_1 + l_2 s_{1+2} \dot{\vartheta}_{1+2}$ (11)

$\dot{y} = - l_1 c_1 \dot{\vartheta}_1 - l_2 c_{1+2} \dot{\vartheta}_{1+2}$ (12)

Here $\dot{x}$ and $\dot{y}$ are the velocities along X-axis and Y-axis respectively. $\dot{\theta}_1$ and $\dot{\theta}_2$ indicate the angular velocities imparted by the motors and thus are known to us. Thus if we know the value of θ_1 and θ_2 for the co-ordinates x and y, we can calculate the velocity at these points and in turn agility also. The value of θ_1 and θ_2 are calculated by inverse kinematics mentioned earlier and substituted in these equations. After calculating velocity, agility is calculated as per equation 8.
In this manner agility at all the points can be calculated. This is done with help of Matlab which is a numerical computing environment.

Agility Vector Field

Agility vector field is a plot of total agility at all points in the workspace. Total agility is given by the resultant of all three vectors at a given point. Mathematically it can be given as;

$$(A)^p_{total} = \sqrt{{(A)_X^p}^2 + {(A)_Y^p}^2 + {(A)_Z^p}^2} \tag{13}$$

This value of the resultant and its direction is plotted for each point in the workspace by using arrows indicating the direction and their length indicating the magnitude. This plot of vector field helps in understanding the regions with better agility and choose an optimal path for motion.

Agility in Particular Direction

Let x_1, y_1 and z_1 be the co-ordinates for point 1 and x_2, y_2 and z_2 be the co-ordinates for point 2. If the robot end-effector has to move from point 1 to point 2, the direction of the vector can be given by

$$\cos(\alpha) = \frac{x_1 - x_2}{\sqrt{(x_1-x_2)^2 + (y_1-y_2)^2 + (z_1-z_2)^2}} \tag{14}$$

$$\cos(\beta) = \frac{y_1 - y_2}{\sqrt{(x_1-x_2)^2 + (y_1-y_2)^2 + (z_1-z_2)^2}} \tag{15}$$

$$\cos(\gamma) = \frac{z_1 - z_2}{\sqrt{(x_1-x_2)^2 + (y_1-y_2)^2 + (z_1-z_2)^2}} \tag{16}$$

$$\cos^2(\alpha) + \cos^2(\beta) + \cos^2(\gamma) = 1 \tag{17}$$

With the help of above equations, the agility in the direction from point 1 towards point 2 can be calculated. After getting the required direction cosines, it can be expressed as a unit vector for further analysis.

$$u = \cos(\alpha)\hat{\imath} + \cos(\beta)\hat{\jmath} + \cos(\gamma)k \tag{18}$$

Interpretation Of Agility

It is difficult to obtain the physical interpretation of agility from raw data. Thus plotting it in right manner is important. The following plots will help in understanding agility and getting a physical intuition. These plots also help in solving the final optimization problem.

Iso-Agility Points

The points in the workspace of Robot Arm at which the magnitude of agility is same are Iso-agility points. The plot helps in understanding the regions having similar agility in the workspace.

Iso-Agility Lines

Iso-agility lines are the lines that connect the Iso-agility points. These help in understanding the distribution of agility in the workspace of Robot Arm. The magnitude of agility is same on all the points that lie on Iso-agility line.

Optimum Path

After calculating the total agility at each point, this information is used to find the optimimum path to travel between two points in the workspace of the Robotic Arm. The optimum path refers to the path along which the Robotic Arm takes minimum time to travel along. Thus we can choose the points with maximum agility in the particular direction we intend the arm to move. From the points, the direction of the travel can be determined. Now the agility in that particular direction is calculated at the nearby points of the path. The points with maximum agility are selected. These points are via points for trajectory planning. A cubic polynomial is chosen as to obtain a variable acceleration in the movement of Robotic Arm.

$$\Theta(t) = a_0 + a_1 t + a_2 t^2 + a_3 t^3 \quad (19)$$
$$\dot{\theta}(t) = a_1 + 2a_2 t + 3a_3 t^2 \quad (20)$$
$$\ddot{\theta}(t) = 2a_2 + 6a_3 t \quad (21)$$

Boundary constraints at the initial time t0 and final time tf can be taken as

$$\theta(t_0) = \theta_0 \quad (22)$$
$$\theta(t_f) = \theta_f \quad (23)$$

$$\dot{\theta}(t_0)= 0 \quad (24)$$
$$\dot{\theta}(t_f) = 0 \quad (25)$$

The velocity at the via points should not change. Thus boundary constraint for via points can be taken as

$$\dot{\theta}(t_f)_1 = \dot{\theta}(t_0)_2 \quad (26)$$

Graphical Interface

A graphical user interface is created in Matlab. This interface enables the user to simulate the Robotic Arm and execute calculations. The user can use mouse to click the buttons in the interface to carry out required operations. The interface uses the same algorithms to calculate forward kinematics and inverse kinematics as mentioned in earlier topics.

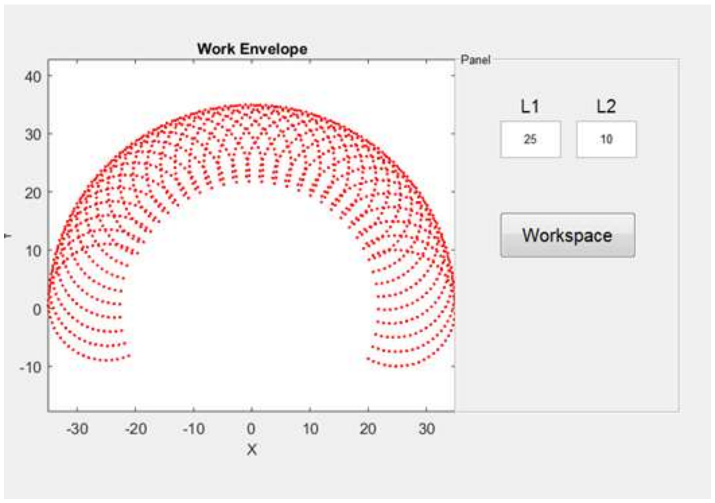

Figure 11: GUI for Plotting Work Envelope

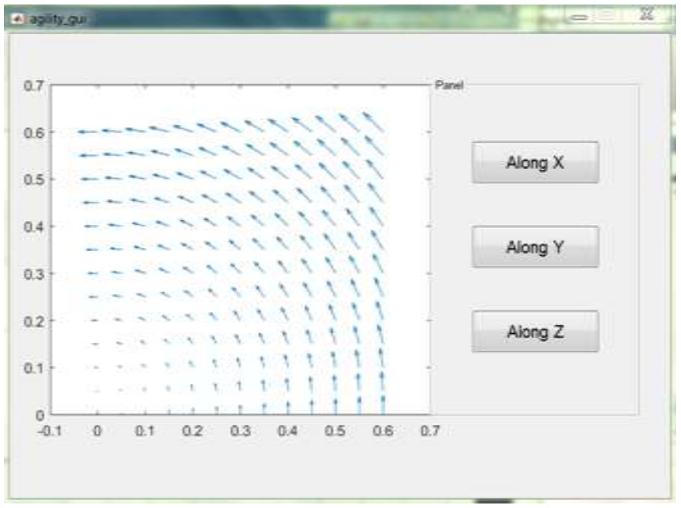

Figure 12: GUI for plotting Agility

The GUI for work envelope takes in the link lengths as inputs and plots a work envelope on the graph. The GUI for agility allows the user to choose agility along different axes and plot on the graph. Moreover user is allowed to change some parameters according to the requirement.

Softwares Used

Software is required for generating a 3d CAD model of Robot Arm and for computation and plotting agility in the workspace.

Solidworks

Solidworks is a 3d CAD modelling software used here for generating the CAD model of Robot Arm. It is used to define the geometric properties, material properties and for assembly of the different components of the robot.

Matlab

Matlab is a numerical-computing environment. Here it has been used for calculating the agility at grid points and plotting. It is also used for generating agility vector field and obtaining the optimal path of motion.

CONCLUSION

By using the Kinematics of a Robotic Arm the Agility can be calculated at different points in the workspace envelope. Total Agility at a point can be expressed as vector sum of agility along different axes. Total Agility for a particular direction is calculated for moving the end-effector of a Robotic Arm from one point to another. For a particular direction, points with maximum agility can be found out and then these points are used as via points for trajectory generation. This generated trajectory would be optimum in reference to time taken to move the end-effector from one point to another.

References

[1] Cristina Castejon, "A Multi-Objective Optimization of a Robotic Arm for Service Tasks" Strojniški vestnik - Journal of Mechanical Engineering 56(2010)5, 316-329 UDC 007.52

[2] K Shivaprakash Reddy, "Optimizing the Performance Evaluation of Robotic Arms with the Aid of Particle Swarm Optimization" (IJACSA) Interna-tional Journal of Advanced Computer Science and Applications, Vol. 3, No. 12, 2012

[3] Kavita Krishnaswamy, Jennifer Sleeman, Tim Oates "Real-Time Path Planning for a Robotic Arm" University of Maryland Baltimore, MD 21250

[4] Wojciech Szynkiewicz , Jacek Błaszczyk "Optimization–based approach to path planning for closed chain robot systems" INT. J. APPL. MATH. COMPUT. SCI., 2011, VOL. 21, NO. 4, 659–670 DOI: 10.2478/V10006-011-0052-8

[5] L.S. Utpat, "Design Optimization of Robotic Arms" International Journal of Engineering Research & Technology (IJERT) ISSN: 2278-0181 Vol. 1 Issue 3, May - 2012

[6] Bahaa Ibraheem Kazem , Ali Ibrahim Mahdi, Ali Talib Oudah, "Motion Planning for a Robot Arm by Using Genetic Algorithm" Jordan Journal of Mechanical and Industrial Engineering Volume 2, Number 3,Sep. 2008 ISSN 1995-6665, Pages 131 - 136

[7] Joseph A. Cascio, "Optimal Path Planning for Multi-arm, Multi-link Robotic Manipulators" Naval Postgraduate School, December 2008