Journal of Mechatronics and Robotics

Original Research Paper

Optimization of Inverse Kinematic Solution of a T4R Robotic Manipulator

Shravan Anand Komakula

Department of Mechanical Engineering, Kakatiya Institute of Technology and Science, Warangal, India

Article history
Received: 24-04-2019
Revised: 12-05-2019
Accepted: 14-05-2019

Corresponding Author:
Shravan Anand Komakula
Kakatiya Institute of Technology and Science, Warangal, India
Email: shravan.komakula@gmail.com

Abstract: Inverse kinematics has always been a major subject of extensive research filled with redundancy. A need for meticulous functioning and fast computation is always high in any mechatronic system and is growing with new advances and human requirements. In this paper, a T4R (T-Twisting, R-Rotary) robot with constrained motion and joint precision of 1 degree is analyzed for its workspace and kinematics. While the redundant inverse kinematic solution is being optimized for precision and limiting errors, further can be used for programming and motion planning a Robotic Manipulator or Arm. The strategies developed in this paper could also be useful for solving the inverse kinematics problems of other similar types of robotic arms.

Keywords: Kinematics, Manipulator, T4R, Redundant, End-Effector, Constrained Motion, Motion Planning

Introduction

Robotic arm actions are executed in the joint angles while robot motions are specified in the Cartesian coordinates. Conversion of the position and orientation of a robot manipulator end-effector from Cartesian space to joint angles called as inverse kinematics problem, which is of fundamental importance in calculating desired joint configurations for robot manipulator design and control. For robotic manipulators that are redundant or with high degrees of freedom (*dof*), an analytical solution to the inverse kinematics is very difficult or impossible as there may be more than one solutions being possible. There is no effective solution to its inverse kinematics to date (Gan *et al*., 2005). For a manipulator with n degree of freedom, at any instant of time joint variables is denoted by $\theta_i = \theta(t)$, i = 1, 2, 3,..., n and position variables $x_j = x(t)$, j = 1, 2, 3,...,m. The relations between the end-effector position x(t) and joint angle θ(t) can be represented by a forward kinematic equation:

$$x(t) = f(\theta(t)) \quad (1)$$

where, *f* is a nonlinear, continuous and differentiable function.

On the other hand, with the given desired end-effector position, the problem of finding the values of the joint variables is inverse kinematics. This can be solved by:

$$\theta(t) = f'(x(t)) \quad (2)$$

The different techniques used for solving inverse kinematics can be classified as algebraic (Craig, 1989), geometric (Lee, 1982) and iterative (Korein *et al*., 1982). The algebraic methods do not guarantee closed form solutions. In the case of geometric methods, closed form solutions for the first three joints of the manipulator must exist geometrically. The iterative methods converge to only a single solution or multiple solutions depending on the end-effector criterions (Alavandar and Nigam, 2008). In this paper, an iterative method is used in the computation of inverse kinematics and the solution is optimized for errors in result, this type of strategies can be used for motion planning and programming robots efficiently. This method is also developed for faster computation and further optimized for compatibility.

Kinematic Description of the Arm

Model

Consider the robot configuration as shown in Figs. 1, 2.

Where:

$\theta_1,\theta_2,\theta_3,\theta_4,\theta_5$ Are the Angles of links relative to horizontal (Degree)
L_1,L_2,L_3,L_4,L_5 Are Link lengths (cm)
(x_1,y_1,z_1) is the point of origin of the chain
(x_5,y_5,z_5) is the end-effector point

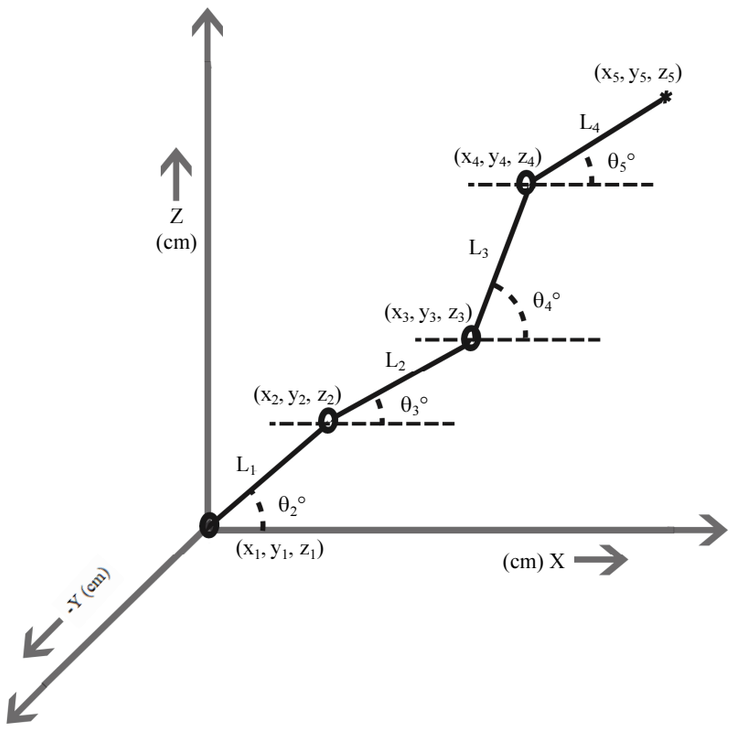

Fig. 1: Side view of the planar kinematic chain

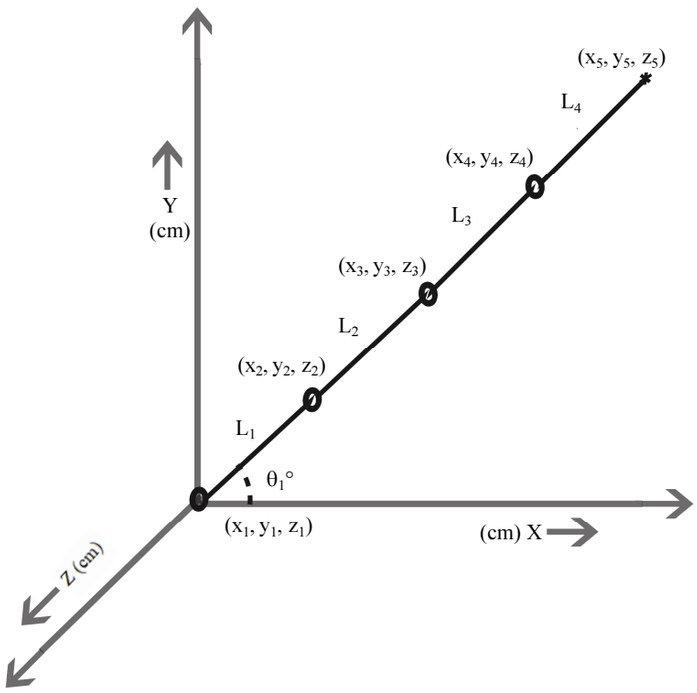

Fig. 2: Top view of the planar kinematic chain

The side view of the planar kinematic chain is shown above in Fig. 1 with vertical Z-axis and horizontal X-axis where we can observe angles $\theta_2,\theta_3,\theta_4,\theta_5$ which are taken with respect to the X-Y plane.

The top view of the planar kinematic chain is shown above in Fig. 2 with vertical Y-axis and horizontal X-axis where we can observe angle θ_1 which are taken with respect to X-Z plane. Angles taken in an anti-clockwise direction are taken as positive and clockwise directions are taken as negative.

Degrees of Freedom

AT4R robot has 5 degrees of freedom i.e., it requires five parameters to specify a particular position of the chain. To control a robot the input parameters required are in the form of angles to drive servo motors and the output being the positioning of end-effector in 3D space.

Constraints and Limits

Considering that joints actions are performed using servo motors which have a precision of 1 degree, 5 such kinds of joints are present in T4R robot and each servo motor has 180 degrees of constrained motion only, this limits can be mathematically represented in the range of angles as:

θ_1^o [0 180]
θ_2^o [0 180]
θ_3^o [θ_2^o-90 θ_2^o+90]
θ_4^o [θ_3^o-90 θ_3^o+90]
θ_5^o [θ_4^o-90 θ_4^o+90]

Methodology

The methodology process is as follows:

a. Defining a virtual model of the robot.
b. Computing inverse kinematics using iterations in Matlab for different end-effector positions.
c. Plotting the results and observing the errors for optimizing.

Computation Process

The inputs for the program are three coordinates of end-effector along with the angle made by fourth link $\theta_5^o$with horizontal and the limit of error in cm. The process begins with calculating the coordinates of joint 4 in the three-dimensional system with θ_1^o using X5, Y5 coordinates and tangent rule. Then the joint 4 is rotated through the Z-axis through an angle θ_1^o to place the kinematic chain in the plane of X-Z axis. In the next step an array locus of points tracing the path of the circle with L_1 as radius and X_1,Z_1 as the center, then another array locus of points tracing the path of joint 3 i.e., the circle with L_3 as radius and rotated point of joint 4 as the center are computed. Then the distance between points on the circles with each other is iterated and when the distance is equal to L_2 with a precision of 0.1 cm there forms a feasible position of the chain. This precision is required as the mathematical results are not always accurate to higher decimal points and rounded off to 6 decimal places in Matlab. Thus joint angles made by links in this position are calculated using the points on circles and the values are floored to the nearest integer as our consideration constraint, the servos accept only integer angles. The floored angles undergo forward kinematics using trigonometric equations on sine and cosine rules and the error in 3D point obtained to the required end-effector point is calculated if the distance is less than or equal to the error limit defined at the starting of the program then it is plotted and the link angles are printed in the command window. This data can be further used for motion planning.

This type of process is adapted to decrease the time for computation so that the algorithm can be used for real-time programming of a robot control unit.

Computation Program

The program is written using trigonometric relations in Matlab software and additional user-defined functions are also given below, they are:

a. compute_circle
b. compute_circle_360
c. F_KIN_Angles

For computation, taking link lengths to be L_1=9 cm, L_2=9 cm, L_3=9 cm, L_4=13.5 cm and (x_1,y_1,z_1) being the origin (0, 0, 0). Input parameters for the program are end-effector coordinates and angle of the fourth link along with the limiting amount of error. The output consists of remaining four angles of the all possible positions of the chain which satisfy the error limit and end-effector criterion. The code is given below in Matlab programming language.

```
%% Input: End effector coordinates and Q5°
X5=15; Y5=15; Z5s=20; Q5=45;
Precision=0.1;    Error=0.2;%%    Error Limit
L1=9;L2=9;L3=9;L4=13.5;
figure(1); drawnow; hold on; grid on; grid minor; axis([X5-0.21 X5+0.21 Y5-0.21 Y5+0.21]);
viscircles([X5 Y5],0.01,'lineWidth',1);
viscircles([X5 Y5],0.05,'lineWidth',1);
viscircles([X5 Y5],0.1,'lineWidth',1);
viscircles([X5 Y5],0.15,'lineWidth',1);
viscircles([X5 Y5],0.2,'lineWidth',1);
```

```
plot(X5,Y5,'r*');            xlabel('X
Coordinate'); ylabel('Y Coordinate');
legend('(X5,Y5)');
figure(2); drawnow; hold on; grid on;
grid minor;  axis([X5-0.21 X5+0.21 Z5-
0.21 Z5+0.21]);
viscircles([X5
Z5],0.01,'lineWidth',1);
viscircles([X5
Z5],0.05,'lineWidth',1);
viscircles([X5 Z5],0.1,'lineWidth',1);
viscircles([X5
Z5],0.15,'lineWidth',1);
viscircles([X5 Z5],0.2,'lineWidth',1);
plot(X5,Z5,'r*');            xlabel('X
Coordinate'); ylabel('Z Coordinate');
legend('(X5,Z5)');
figure(3); drawnow; hold on; grid on;
grid minor;  axis([Z5-0.21 Z5+0.21 Y5-
0.21 Y5+0.21]);
viscircles([Z5
Y5],0.01,'lineWidth',1);
viscircles([Z5
Y5],0.05,'lineWidth',1);
viscircles([Z5 Y5],0.1,'lineWidth',1);
viscircles([Z5
Y5],0.15,'lineWidth',1);
viscircles([Z5 Y5],0.2,'lineWidth',1);
plot(Z5,Y5,'r*');            xlabel('Z
Coordinate'); ylabel('Y Coordinate');
legend('(Z5,Y5)');
%% Rotation of chain to X-Z Axis
Dis5=sqrt(X5^2+Y5^2+Z5^2);
if(Dis5>40.5)
fprintf('Cannot reach point 5\n');
end
Q1=atand(Y5/X5);
Q1=floor(Q1);
Q1=Angle_Convert( Q1,Y5,X5 );
X5p=X5*cosd(-Q1)-Y5*sind(-Q1);
Y5p=X5*sind(-Q1)+Y5*cosd(-Q1);
Z5p=Z5;
X4p=X5p-L4*cosd(Q5);
Z4p=Z5p-L4*sind(Q5);
%%Converting to 2D analysis
x4=X4p; z4=Z4p; a=1;
if (sqrt(x4^2+z4^2)>27)
    fprintf('Cannot reach point 4\n');
end
X2=0;Z2=0;X3=0;Z3=0;
[X2,Z2]=compute_circle_360(L1,0,0,X2,Z
2);
[X3,Z3]=compute_circle(L2,x4,z4,X3,Z3)
;
 Xo=0; Yo=0; Zo=0;
for i=1:1:180
for j=1:1:2000
Length=sqrt((X2(i)-X3(j))^2+((Z2(i)-
Z3(j))^2));
if((Length>=(L2-Precision))&&
(Length<=(L2+Precision)))
              x2(a)=X2(i);
z2(a)=Z2(i);
              x3(a)=X3(j);
z3(a)=Z3(j);
Q2(a)=atand(z2(a) / x2(a));
Q2(a)=Angle_Convert(Q2(a),z2(a),x2(a)
);
Q3(a)=atand( (z3(a) - z2(a))/(x3(a) -
x2(a)) );
Q3(a)=Angle_Convert( Q3(a), (z3(a) -
z2(a)),(x3(a) - x2(a)) );
Q4(a)=atand( (z4-z3(a))/(x4-x3(a)) );
Q4(a)=Angle_Convert(Q4(a),(z4-
z3(a)),(x4-x3(a)) );
q2=floor(Q2(a));      q3=floor(Q3(a));
q4=floor(Q4(a));
if               (q3<=q2+90)&&(q3>=q2-
90)&&(q4<=q3+90)&&(q4>=q3-
90)&&(Q5<=q4+90)          &&(Q5>=q4-90)
Xp=L1*cosd(q2)+L2*cosd(q3)+L3*cosd(q4)
;
Zp=L1*sind(q2)+L2*sind(q3)+L3*sind(q4)
;
[X,Y,Z]=F_KIN_Angles(Q1,q2,q3,q4,Q5);
if (Xo~=X)&&(Yo~=Y)&&(Zo~=Z)
       Xe=abs(X-X5);
       Ye=abs(Y-Y5);
       Ze=abs(Z-Z5);
if(sqrt(Xe^2+Ye^2+Ze^2)<=Error)
       a=a+1;
fprintf(['Angles
',num2str(Q1),',',num2str(q2),
',',num2str(q3),',',num2str(q4),',',nu
m2str(Q5),' | '])
fprintf(['Point
(',num2str(X),',',num2str(Y
,',',num2str(Z),') ']);
fprintf(['Error in X=',num2str(Xe),' ,
Y=',num2str(Ye),',
Z=',num2str(Ze),'\n']);
%%Plotting points
figure(1);          hold          on;
plot(X,Y,'b.','markersize',8);
figure(2);          hold          on;
plot(X,Z,'b.','markersize',8);
figure(3);          hold          on;
plot(Z,Y,'b.','markersize',8);
end
Xo=X; Yo=Y; Zo=Z;
end
end
```

```
end
end
end
figure(1);
title(['(',num2str(X5),',',num2str(Y5)
,',',num2str(Z5),')']);
saveas(gcf,['(',num2str(X5),',',num2st
r(Y5),',',num2str(Z5),') XY.jpg']);
figure(2);
title(['(',num2str(X5),',',num2str(Y5)
,',',num2str(Z5),')']);
saveas(gcf,['(',num2str(X5),',',num2st
r(Y5),',',num2str(Z5),') XZ.jpg']);
figure(3);
title(['(',num2str(X5),',',num2str(Y5)
,',',num2str(Z5),')']);
saveas(gcf,['(',num2str(X5),',',num2st
r(Y5),',',num2str(Z5),') ZY.jpg']);

function            [X2,Y2]            =
compute_circle(l,x,y,X2,Y2)
th = 0:pi/1000:2*pi;
X2 = l * cos(th)+x;
Y2 = l * sin(th)+y;
end

function            [X2,Y2]            =
compute_circle_360(l,x,y,X2,Y2)
th = 1:1:180;
X2 = l * cosd(th)+x;
Y2 = l * sind(th)+y;
end

function[X,Y,Z]=F_KIN_Angles(Q1,Q2,Q3,
Q4,Q5)
L1=9;L2=9;L3=9;L4=13.5;
x=L1*(cosd(Q2))   +   L2*(cosd(Q3))   +
L3*(cosd(Q4)) + L4*(cosd(Q5));
    z=L1*(sind(Q2))  +  L2*(sind(Q3))  +
L3*(sind(Q4)) + L4*(sind(Q5));
    X=x*(cosd(Q1));
    Y=x*(sind(Q1));
    Z=z;
End
```

The equations given below convert joint angles which are with reference to the horizontal axis to with reference to links, these angles can be used as angle inputs to servo motors directly for a specific function:

$$\varnothing_5^o = \theta_5^o + 90 - \theta_4^o \quad (3)$$

$$\varnothing_4^o = \theta_4^o + 90 - \theta_3^o \quad (4)$$

$$\varnothing_3^o = \theta_3^o + 90 - \theta_2^o \quad (5)$$

$$\varnothing_2^o = \theta_2^o \quad (6)$$

$$\varnothing_1^o = \theta_1^o + 90 \quad (7)$$

where, $\varnothing_1$, $\varnothing_2$, $\varnothing_3$, $\varnothing_4$, $\varnothing_5$ are angles relative to Links(Degree).

Results

The circles are plotted in red color with radii of 0.01 cm, 0.05 cm, 0.1 cm, 0.15 cm, 0.2 cm and center as the required end-effector point for reference. The study has been divided into 3 cases with different end-effector positions and computation is done with an error limit of 0.2 cm.

Case Study - I

Input: X5=15; Y5=15; Z5=20; θ_5^o=45;

The Figs. 3, 4 and 5 shows the projections of obtained end-effector points in three planes of axes obtained for this case, which can be used as a reference for observing the outcome of the robot being programmed.

From the figures, we can observe that there are six feasible configurations for the robot possible within the error limits of 0 to 0.08 cm in this case, with θ_1 obtained as 45° and θ_5 also being 45°. The feasible link configuration with least error is highlighted red color in Table 1.

Case Study - II

Input: X_5=15; Y_5=20; Z_5=25; θ_5^o=60;

The Figs. 6, 7, 8 shows the projections of obtained end-effector points in three planes of axes obtained for this case.

Table 1: Summary of case study - i

Angles Obtained (Degrees)			Obtained end effect or point coordinates			
θ_2	θ_3	θ_4	X_5	Y_5	Z_5	Error (cm)
104	46	-32	15.0281	15.0281	19.9834	0.043127
107	44	-29	15.0333	15.0333	20.0413	0.062604
110	42	45	14.9731	14.9731	19.9394	0.071541
112	40	45	15.0549	15.0549	20.0151	0.079035
16	37	45	15.0229	15.0229	19.9732	0.041983
119	35	45	14.9636	14.9636	19.9484	0.072886

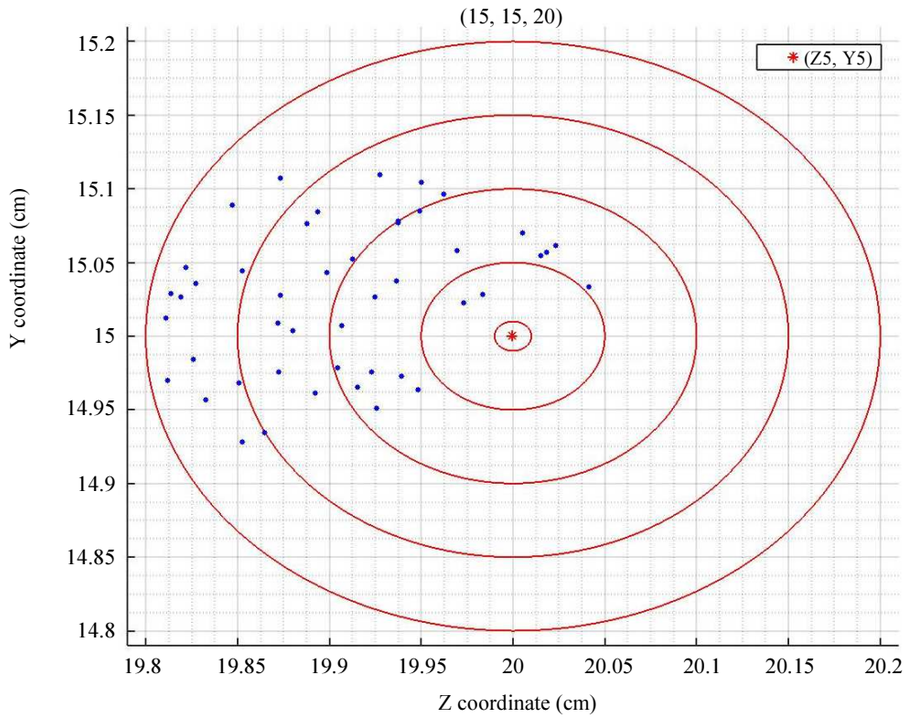

Fig. 3: Projection of case study-i in the Z-Y plane

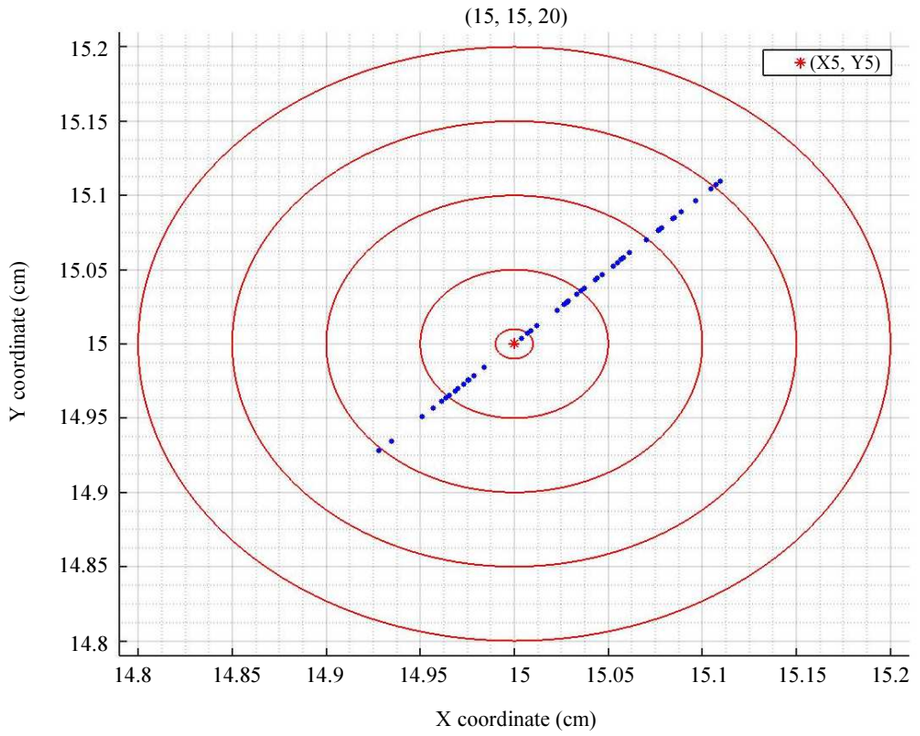

Fig. 4: Projection of case study-i in the X-Y plane

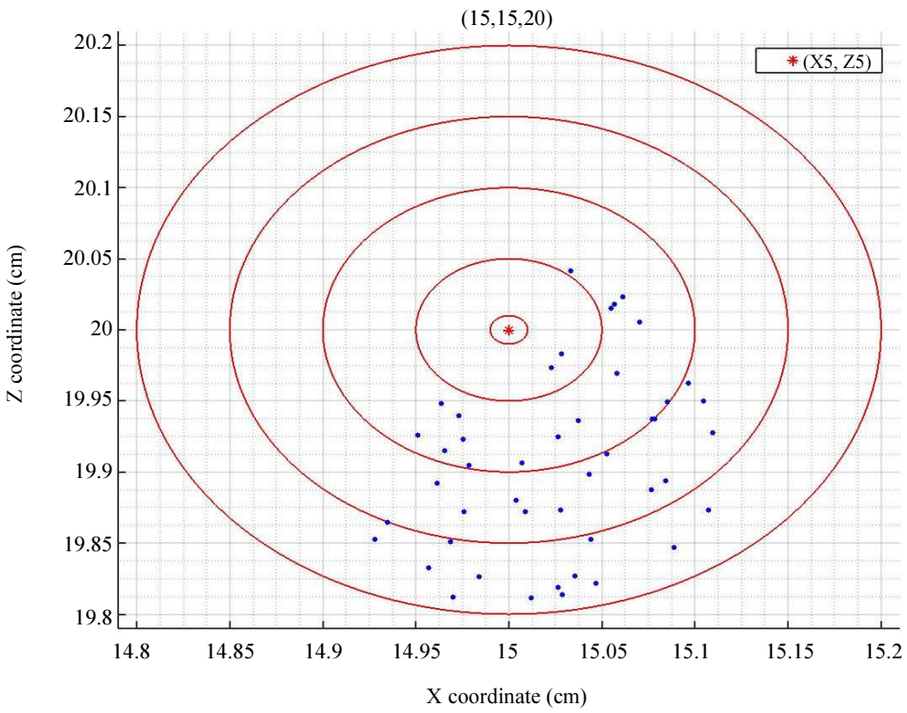

Fig. 5: Projection of case study-i in the X-Z plane

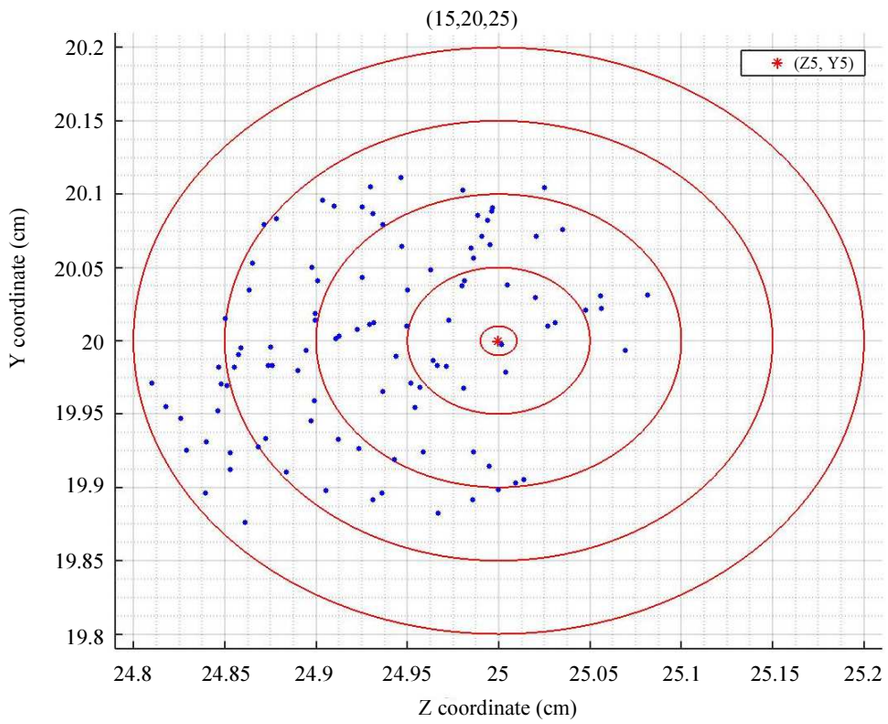

Fig. 6: Projection of case study-ii in the Z-Y plane

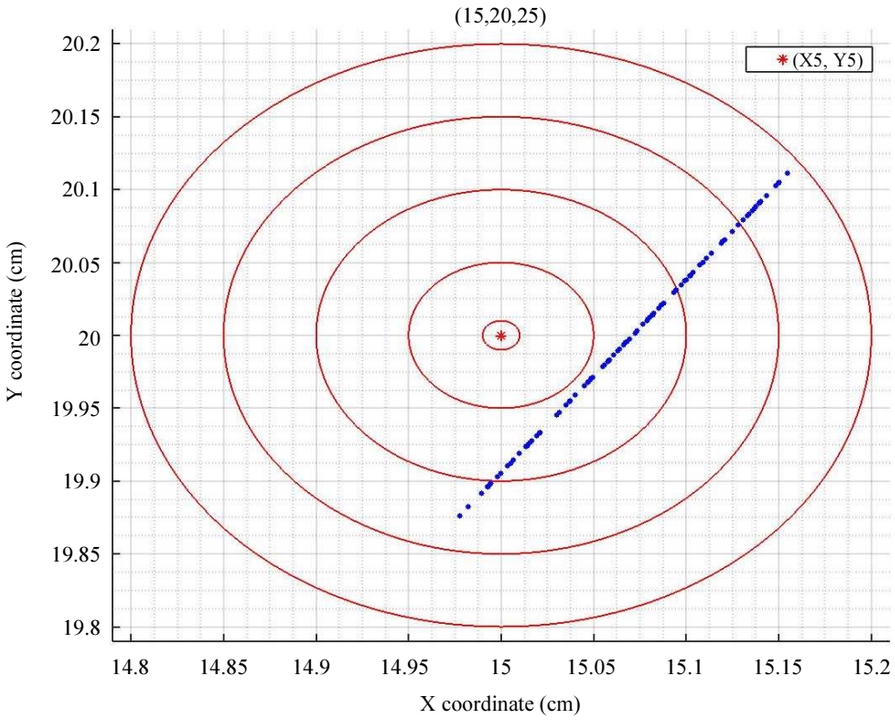

Fig. 7: Projection of case study-ii in the X-Y plane

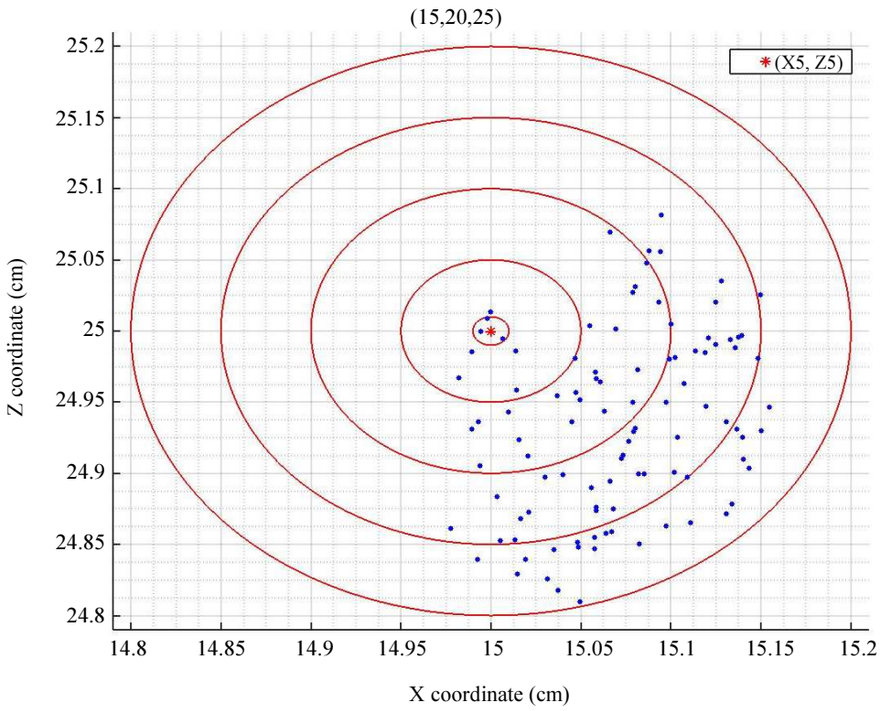

Fig. 8: Projection of case study-ii in the X-Z plane

From the above figures, we can observe that there are five feasible configurations for the robot possible within the error limits of 0 to 0.06 cm in this case, with θ_1 obtained as 53^o and θ_5 being 60^o. The feasible link configurations with least error are highlighted red color in Table 2.

Case Study - III

Input: X_5=10; Y_5=-10; Z_5=10; $\theta_5{}^o$=-45;

The below Figs. 9, 10, 11 shows the projections of obtained end-effector points in three planes of axes obtained for this case.

From the figures, we can observe that there are ten feasible configurations for the robot possible within the error limits of 0 to 0.06 cm in this case, with θ_1 obtained as 45^o and θ_5 being -45^o. The feasible link configuration with least error is highlighted red color in Table 3.

Table 2: Summary of case study - ii

Angles Obtained (Degrees)			Obtained end effect or point coordinates			
θ_2	θ_3	θ_4	X_5	Y_5	Z_5	Error (cm)
2	82	27	15.0551	19.9788	25.0038	0.059154
27	82	2	15.0551	19.9788	25.0038	0.059154
43	72	82	15.0469	19.9679	24.9809	0.059943
43	-9	72	15.0469	19.9679	24.9809	0.059943
72	-9	43	15.0469	19.9679	24.9809	0.059943

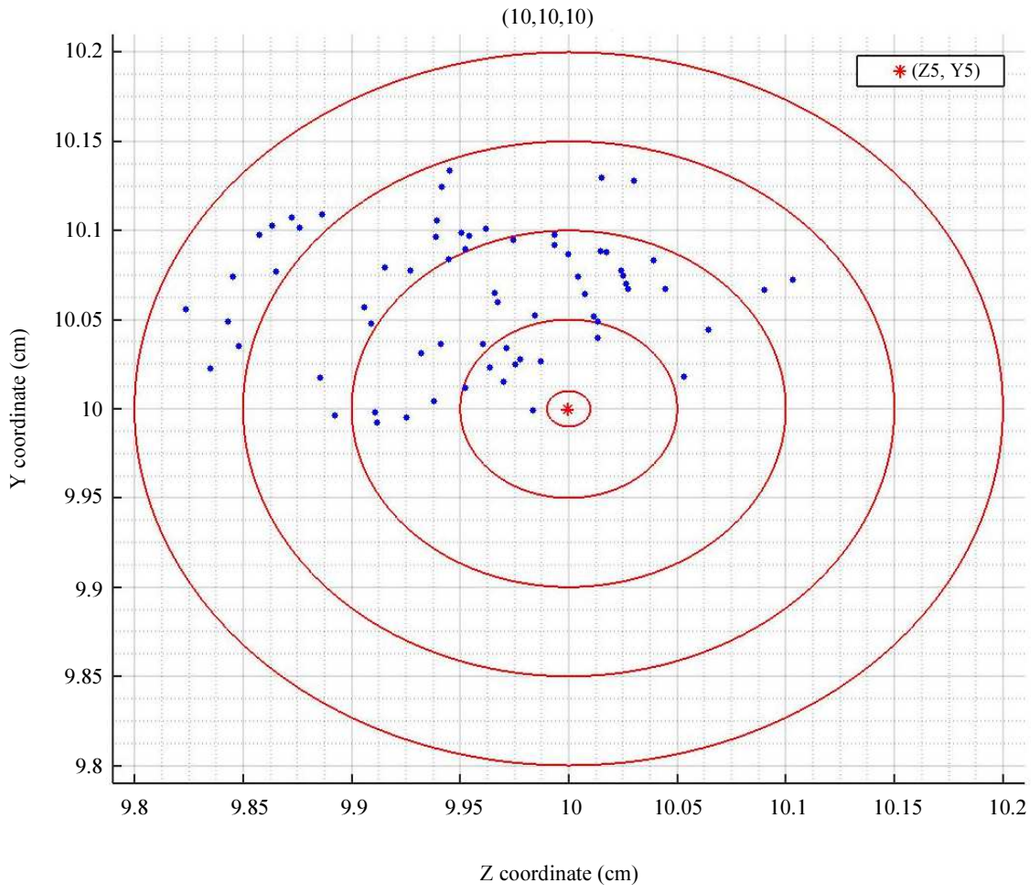

Fig. 9: Projection of case study-iii in the Z-Y plane

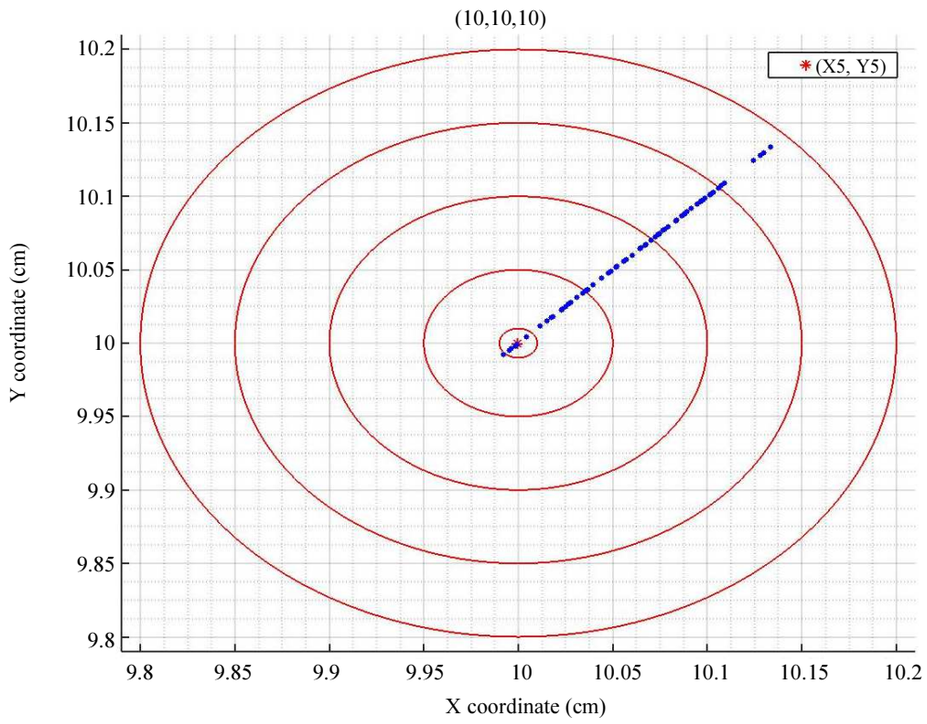

Fig. 10: Projection of case study-iii in the X-Y plane

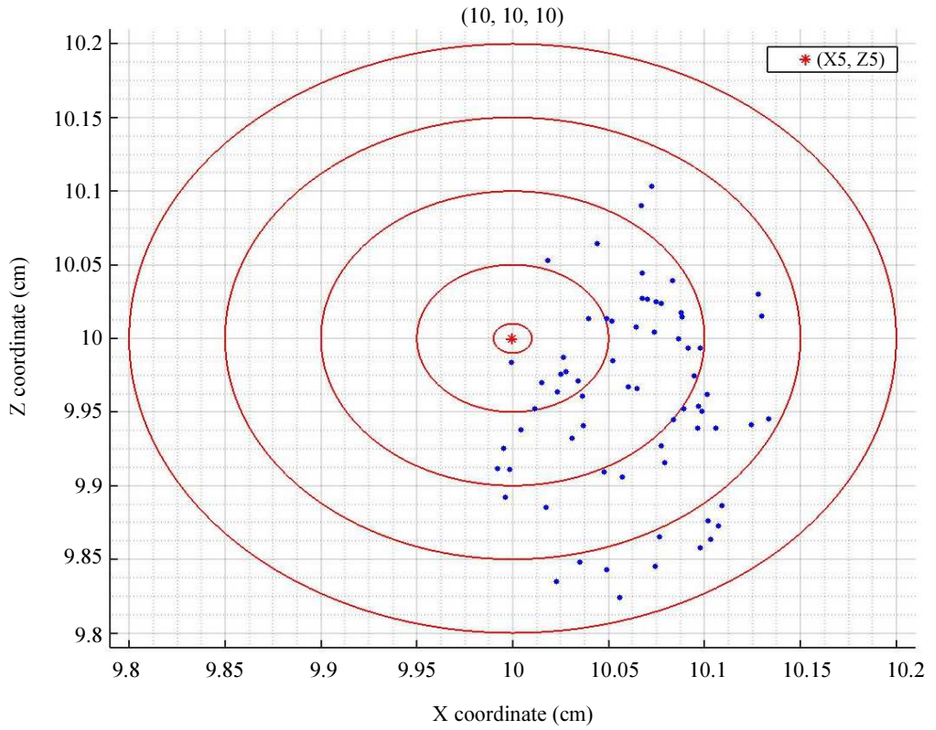

Fig. 11: Projection of case study-iii in the X-Z plane

Table 3: Summary of case study - iii

Angles Obtained (Degrees)			Obtained end effect or point coordinates			
θ_2	θ_3	θ_4	X_5	Y_5	Z_5	Error (cm)
123	83	20	10.0397	10.0397	10.0132	0.057642
124	82	21	10.0183	10.0183	10.0531	0.059077
126	80	22	10.015	10.015	9.9699	0.036799
130	75	26	10.0263	10.0263	9.9871	0.039396
131	74	27	9.9993	9.9993	9.9837	0.016312
137	64	36	10.034	10.034	9.9713	0.056047
138	62	38	10.0232	10.0232	9.9637	0.048941
139	59	41	10.0277	10.0277	9.9776	0.045091
140	56	44	10.0115	10.0115	9.9524	0.050271
140	55	45	10.0251	10.0251	9.9755	0.043192

Conclusion

Even if there is no perfect solution to an inverse kinematic problem with redundancy, there are methods which can be used to compute an effective solution and can be optimized for the requirement. These type of developments in computation processes contribute to a reduction in computation time and also can be adopted in high precise robots and program with efficiency.

Acknowledgment

The author would like to acknowledge the faculty of Department of Mechanical Engineering at Kakatiya Institute of Technology and Science, Warangal for the constant motivation and support, without whose guidance this research would not be possible.

Ethics

Authors should address any ethical issues that may arise after the publication of this manuscript.

Author's Contributions

Shravan Anand Komakula: Designed the research plan and organized the study, programmed using Matlab and analyzed the theoretical results.

References

Alavandar, S. and M.J., Nigam, 2008. Inverse kinematics solution of 3dof planar robot using ANFIS. Int. J. Comput. Commun. Control, 3: 150-155.

Craig, J.J., 1989. Introduction to Robotics: Mechanisms and Controls. 1st Edn., Addison-Wesley, Reading, MA.

Gan, J., E. Oyama, E. Rosales and H. Hu, 2005. A complete analytical solution to the inverse kinematics of the Pioneer 2 robotic arm. Robotica, 23: 123-129. DOI: 10.1017/S0263574704000529

Korein, J.U. and N.I. Balder, 1982. Techniques for generating the goal-directed motion of articulated structures. IEEE Comput. Graph. Applic., 2: 71-81. DOI: 10.1109/MCG.1982.1674498

Lee, G.C.S., 1982. Robot arm kinematics, dynamics and control. Computer, 15: 62-79. DOI: 10.1109/MC.1982.1653917