RESEARCH ARTICLE OPEN ACCESS

Direct Kinematic modeling of 6R Robot using Robotics Toolbox

Prashant Badoni*
*(Department of Mechanical Engineering, Graphic Era University, Dehradun-248002)

ABSTRACT
The traditional approaches are insufficient to solve the complex kinematics problems of the redundant robotic manipulators. To overcome such intricacy, Peter Corke's Robotics Toolbox [1] is utilized in the present study. This paper aims to model the direct kinematics of a 6 degree of freedom (DOF) Robotic arm. The Toolbox uses the Denavit-Hartenberg (DH) Methodology [2] to compute the kinematic model of the robot.
Keywords **-** Direct Kinematics Solution, MATLAB, Robotic Toolbox, Robot manipulator.

I. INTRODUCTION

Kinematic modeling of a robot is concerned with its motion without consideration of forces. The kinematic modeling of a robot is usually categorized into forward or direct kinematics and inverse kinematics. In the direct kinematics problem, the location of end effector in the work space i.e. position and orientation, is determined based on the joint variables [3] [4]. The inverse kinematics problem refers to finding the values of the joint variables that allows the manipulator to reach the given location. The relationship between direct and inverse kinematics is illustrated in Fig.1.

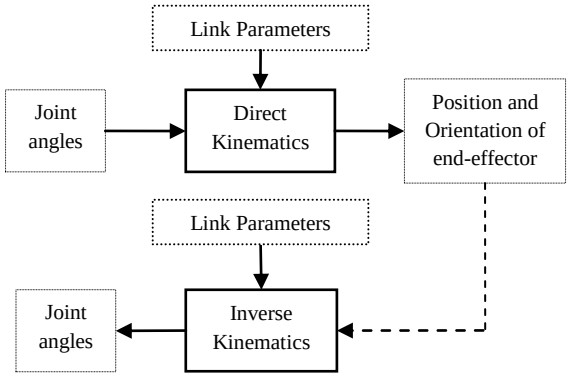

Fig.1. Relationship between direct and inverse kinematics

The present paper describes the modeling and analysis of a 6R robot with the application of Robotics Toolbox [1] in MATLAB. The Toolbox is based on a general method of representing kinematics and dynamics of serial link manipulators by description matrices. In this study, the toolbox is used for direct kinematic solution of a six link robot manipulator. The toolbox is also capable of plotting 3D graphical robot and allows users to drive the robot model. A general serial 6R robot is shown in Fig.2.

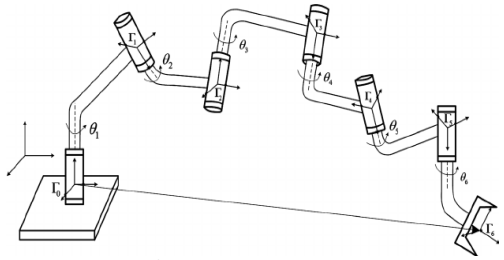

Fig.2. Schematic of a general serial 6R robot manipulator [5]

II. DIRECT KINEMATIC MODEL

Direct kinematic analysis of 6R robot is done by using Robotics Toolbox in MATLAB. Various functions are used to complete the analysis conveniently. First, create the six links and set the D-H parameters as given in Table 1.

Table 1: D-H parameters of the manipulator

Links	a_i	α_i	d_i	θ_i	***Range***
1	0	-90	1	θ_1	0 to 360
2	0	90	4	θ_2	-27 to 189
3	10	0	3	θ_3	-241 to 82
4	10	0	0	θ_4	0 to 360
5	0	-90	3	θ_5	-100 to 100
6	0	0	1	θ_6	0 to 360

Here θ_i is joint angle, d_i is joint offset, a_i is link length and α_i is the twist angle. The limits of each joint angle is given in the above table and utilized in the *m-file*.

The joint variable is set to be zero at the starting. A few workspace variables are created to define the useful joint angles: *qz* for all starting position, *qr* for ready position or reach position. The trajectory between any of these two joint angle vectors can be generated with respect to time. After that, the forward kinematic can be computed easily by using *fkine* function, which returns a homogeneous transformation for the last link of the robot. Here is the source code in MATLAB.

```
% create links using D-H parameters
% L = Link([theta, d, a, alpha]
L(1) = Link([0  1   0  –pi/2]);
L(2) = Link([0  4   0   pi/2]);
L(3) = Link([0  3  10  0]);
L(4) = Link([0  0  10  0]);
L(5) = Link([0  3   0  –pi/2]);
L(6) = Link([0  1   0   0]);

% Joint limits
L(1).qlim = pi/180*[0 360];
L(2).qlim = pi/180*[-27 189];
L(3).qlim = pi/180*[-241 82];
L(4).qlim = pi/180*[0 360];
L(5).qlim = pi/180*[-100 100];
L(6).qlim = pi/180*[0 360];

% create the robot model
Robot = SerialLink(L);
Robot.name = 'Robot'

% starting position
qz = [0 0 0 0 0 0];

% ready position
qr = [pi/4  0  pi/4  0  –pi/2  pi];

% generate a time vector
t = [0:0.056:2];

% computes the joint coordinate trajectory
q = jtraj(qz, qr, t);

% direct kinematics for each joint co-ordinate
T = Robot.fkine(q)
```

Joint space trajectory can be animated by using the following command:
>>Robot.plot(q)

Starting and ready position of the robot manipulator from this simulation are given below in Fig.3 and Fig.4:

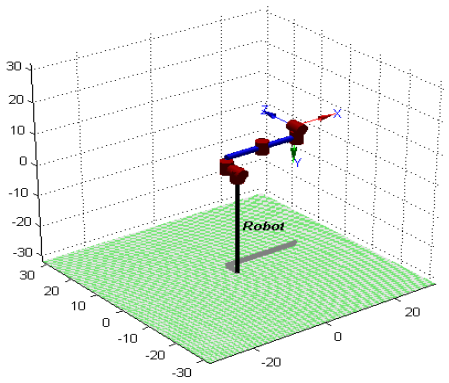

Fig.3. Zero Pose of Robot

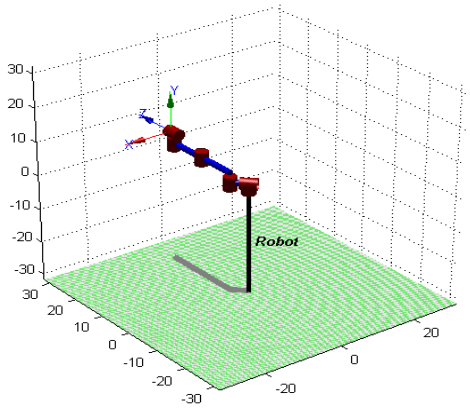

Fig.4. Ready Pose of Robot

3D Trajectory of the end-effector of the robot can be plotted by:
>>figure,plot3(squeeze(T(1,4,:)),squeeze(T(2,4,:)), squeeze(T(3,4,:)));
xlabel('X(inch)'),ylabel('Y(inch)'),zlabel('Z(inch)'), title('End-effector 3D Trajectory'),grid;

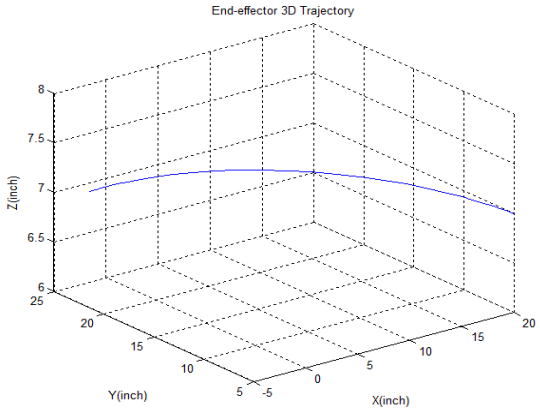

Fig.5. Robot direct kinematics end-effector trajectory

Cartesian coordinates x, y, z of end-effector can be plotted against time by:
>>figure,plot(t,squeeze(T(1,4,:)), '-',t,squeeze(T(2,4,:)),'--',t,squeeze(T(3,4,:)),'-.');
xlabel('Time(s)'), ylabel('Coordinate(inch)'),grid, legend('X','Y','Z'), title('End-effector Position');

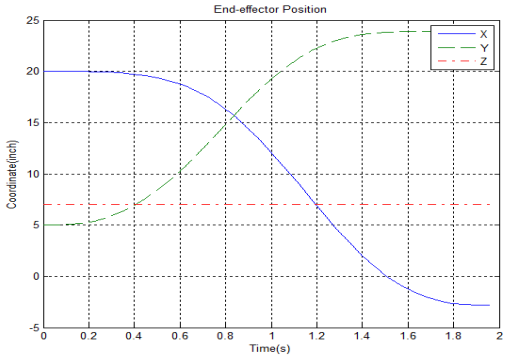

Fig.6. Robot direct kinematics end-effector X, Y, Z coordinates

Generalized coordinates for each joint of the robot can be plotted against time by:

```
>>subplot(6,1,1); plot(t,q(:,1));
xlabel('Time(s)'); ylabel('Joint1(rad)');
subplot(6,1,2); plot(t,q(:,2));
xlabel('Time(s)'); ylabel('Joint2(rad)');
subplot(6,1,3); plot(t,q(:,3));
xlabel('Time(s)'); ylabel('Joint3(rad)');
subplot(6,1,4); plot(t,q(:,4));
xlabel('Time(s)'); ylabel('Joint4(rad)');
subplot(6,1,5); plot(t,q(:,5));
xlabel('Time(s)'); ylabel('Joint5(rad)');
subplot(6,1,6);plot(t,q(:,6));
xlabel('Time(s)'); ylabel('Joint6(rad)');
```

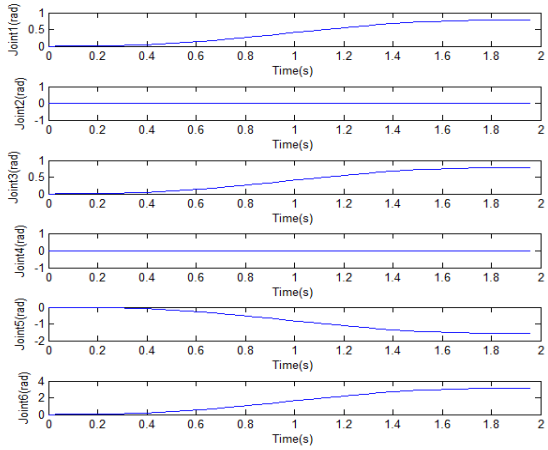

Fig.7. Robot direct kinematics joint angles against time

Velocity and acceleration profile can be obtained by:

```
>> [q,qd,qdd] = jtraj(qz, qr, t)
```

Velocity profile of joints of the robot manipulator is given by:

```
>>subplot(6,1,1); plot(t,q(:,1));title('Velocity');
xlabel('Time(s)'); ylabel('Joint1(rad/s)');
subplot(6,1,2); plot(t,q(:,2));
xlabel('Time(s)'); ylabel('Joint2(rad/s)');
subplot(6,1,3); plot(t,q(:,3));
xlabel('Time(s)'); ylabel('Joint3(rad/s)');
subplot(6,1,4); plot(t,q(:,4));
xlabel('Time(s)'); ylabel('Joint4(rad/s)');
subplot(6,1,5); plot(t,q(:,5));
xlabel('Time(s)'); ylabel('Joint5(rad/s)');
subplot(6,1,6);plot(t,q(:,6));
xlabel('Time(s)'); ylabel('Joint6(rad/s)');
```

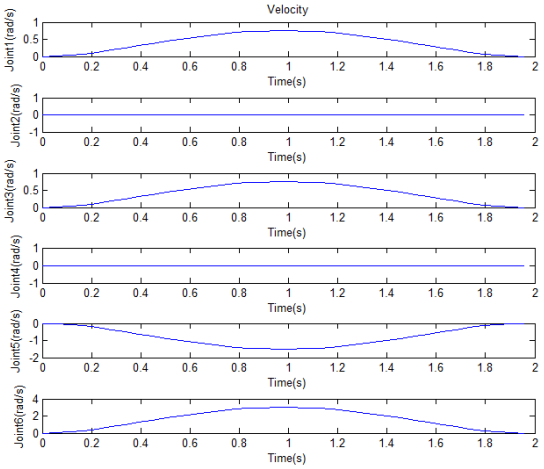

Fig.8. Joint velocity against time

Acceleration profile is given by:

```
>>subplot(6,1,1); plot(t,q(:,1));title('Acceleration');
xlabel('Time(s)'); ylabel('Joint1(rad/s2)');
subplot(6,1,2); plot(t,q(:,2));
xlabel('Time(s)'); ylabel('Joint2(rad/s2)');
subplot(6,1,3); plot(t,q(:,3));
xlabel('Time(s)'); ylabel('Joint3(rad/s2)');
subplot(6,1,4); plot(t,q(:,4));
xlabel('Time(s)'); ylabel('Joint4(rad/s2)');
subplot(6,1,5); plot(t,q(:,5));
xlabel('Time(s)'); ylabel('Joint5(rad/s2)');
subplot(6,1,6);plot(t,q(:,6));
xlabel('Time(s)'); ylabel('Joint6(rad/s2)');
```

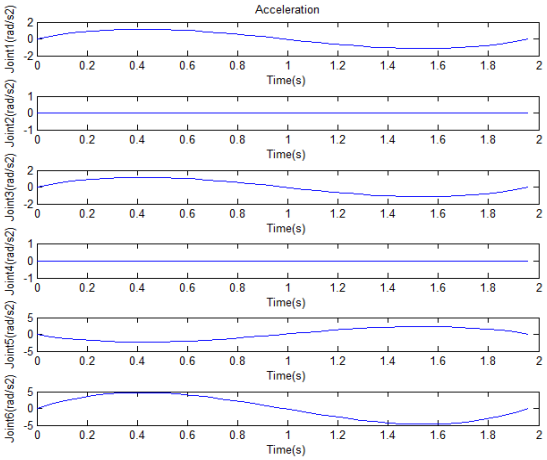

Fig.9. Joint acceleration against time

III. CONCLUSION

In this paper, the direct kinematic model of a 6R robot is presented by utilizing the principal features of the Robotic Toolbox in MATLAB. The Toolbox provides several essential tools for the modeling and analysis of the robotic manipulator, as well as analyzing the experimental results.

REFERENCES

[1] P.I. Corke, A Robotics Toolbox for MATLAB, *IEEE Robotics and Automation Magazine,* Vol.3, No.1, pp.24-32, March 1996.

[2] J. Denavit, R.S. Hartenberg, *A Kinematic Notation for Lower-Pair Mechanisms Based on Matricies*, Trans. of ASME J.App. Mech. vol. 77, pp.215-221, June 1955.

[3] J.J. Crage, *Introduction to Robotics Mechanics and Control*, 3rd Edition, Prentice Hall, 2005.

[4] M.W. Spong, S. Hutchinson and M. Vidyasagar, *Robot Modeling and Control*, 1st Edition, Jon Wiley & Sons Inc, 2005.

[5] Zhongtao Fu, Wenyu Yang and Zhen Yang, Solution of Inverse Kinematics for 6R Robot Manipulators With Offset Wrist Based on Geometric Algebra, *Journal of Mechanism and Robotics,* August 2013, Vol. 5.