Inverse Kinematics (IK) Solution of a Robotic Manipulator using PYTHON

Abstract: Present global engineering professionals feels that, robotics is a somewhat young field with extremely ruthless target, the crucial one being the making of machinery/equipment that can perform and feel like human beings. Robot kinematics deals with the study of motion of linkages which includes displacement, velocities and accelerations of a robot manipulator analytically. Deriving the proper kinematic models for an open chain mechanism of a robot is essential for analyzing the performance of industrial robotic manipulators. In this study, first scrutinize a popular class of two and three degrees of freedom open chain mechanism whose inverse kinematics admits a closed-form analytic solution. A simple coding was developed in python in an easy way. In this connection a two link planer manipulator was considered to get inverse kinematic solution developed in python environment. For this task, we present a solution for obtaining the joint variables of linkages to reach the position in a work space with the corresponding input values such as link lengths and position of end effector.

Keywords: Robotic Manipulator, Inverse Kinematics, Joint Variables, PYTHON

Introduction

Robot kinematics deals with the study of motion of linkages which includes displacement, velocities and accelerations of a robot manipulator analytically. Deriving the proper kinematic models for an open chain mechanism of a robot is essential for analyzing the performance of industrial robotic manipulators. The task related to robot trajectory path planning control can be split into two types, one is the coordination of the links of kinematics chain to produce desired motions of the robot and the other is dynamic control i.e., linkage driving mechanism using actuators technology by providing position and velocity sensors. In the elementary level, the robot manipulator design concentrate on physical arrangement of linkages and mechanisms, includes development of forward and inverse kinematic equations with standard existing methods.

Literature Review

In the field of robotic engineering, researchers such as Kircanski and Vukobratovic (1985; 1986), Morris (1987), Hussain and Nobie (1985) and Tsai and Chiou (1989) were used MACSYMA, REDUCE, SMP and SGM methods, these methods requires typical mathematical concepts to achieve forward and inverse solutions. Khatib (1987) worked in this area and gave solution to prevent singular positions obtained during the control of path planning within the work volume. Lloyd and Hayward (1993) developed a new design of multi-RCCL motion generator which is operated by a programming with 'C' language in a UNIX environment. Mandava and Vandavilli (2016) proposed a closed form solution for 18 DOF biped robot based on inverse kinematics and concept of Zero Moment Point (ZMP) method. Sreenivasulu (2012) attempted an inverse solution for a two degree freedom robotic manipulator using geometric approach. Nugroho et al. (2014) designed and implemented to develop a NAO humanoid robot using kinematic approach. Sadiq et al. (2017) applied to obtain exact solutions to found optimal path using particle swarm optimization (PSO) algorithm in Cartesain space map for two degrees of freedom robotic manipulator. Chaitanyaa and Reddy (2016) developed a model for optimization of path planning of two degree of freedom robotic manipulator.
using genetic algorithm approach. Kanayama et al. (1990) proposed a stable tracking control rule for nonholonomic vehicles to found reasonable target adapted to autonomous mobile robots. Mohamed and Duffy (1985) studied the instantaneous kinematics of end effector platform of fully parallel robot type devices using screw theory concept. Jones and Walker (2006) introduced a modular approach method to get solution for inverse kinematics for multisection continuum robots. Radavelli et al. (2012) presented a comparative study of kinematics of robot manipulators between DH convention and Dual Quaternion approach.

Chen et al. (2015) developed an improved algorithm from screw theory to estimate inverse kinematic solution for a robotic manipulator. Chirikjian (1994) also studied on kinematics of a robotic system by considering metamorphic levels. Robot kinematics and inverse solution methods for different robotic linkages described by various authors of text books like Craig (1989), Malley (2011) and Murray (2017). Raheem et al. (2019) presented in their work, how to enhance the work space followed by robot end effector in the space using metaheuristic methods. Hudgens and Tesar (1988) concentrate their study on kinematic studies on micromanipulators especially for parallel linkages. Sun et al. (2017) proposed analytical inverse kinematic solution using DH notations. Tsai and Morgan (1985), Zhuang et al. (1992) developed a solution for general six and five degrees of freedom manipulators by continuation methods. Veitschegger and Wu (1986) studied on measurement methods in the analysis of robot kinematics to achieve accurate end effector positions. Webster and Jones (2010) designed a kinematic model for continuum robots with constant curvature. Yang et al. (2016) discussed in their book on utilization of automation techniques in the field of applied robotics.

Literature depicts that previous investigators focused on various aspects of methods involved in design and development of inverse kinematics solutions for a different configurations of robotic manipulators. Compared to geometric approaches, programmable studies on inverse kinematic solutions have been found to be a limited extent. Also found that nobody applied PYTHON software to get inverse solution of robotic linkages with multi degrees of freedom problems especially in robotic field.

**Inverse Kinematics**

The robot inverse kinematics task is concerned with the recognition of the whole feasible and proper sets of joint variables that would understand the solution to find out the positions and orientations of the end effector. In the inverse kinematics problem would not specify constantly a unique solution compared with forward solution i.e., number of solutions to be obtained for one end effector position to reach the specified position and orientation.

**Case I. Two Link Planar Manipulator Manipulator**

Consider a two link planer manipulator having link lengths $l_1$, $l_2$ and joint angles $\theta_1$ and $\theta_2$ to reach a desired position $(P_x, P_y)$ as shown in Fig. 1. For this, inverse solution is derived from geometric approach as per the diagram shown in Fig. 2 is as follows:

$$p_x = l_1 \cos \theta_1 + l_2 \cos (\theta_1 + \theta_2)$$

(1)

![Fig. 1: Schematic diagram of 2 link planar manipulator](image)
Fig. 2: Geometric model for a 2 link manipulator

\[ p_x = l_1 \sin \theta_1 + l_2 \sin(\theta_1 + \theta_2) \]  
\[ p_y = l_1 \cos \theta_1 + (l_1^2 + l_2^2 \cos^2 \theta_1) + 2l_2 \cos \theta_1 \cos(\theta_1 + \theta_2) \]  
\[ p_x^2 + p_y^2 = l_1^2 + l_2^2 + 2l_1l_2 \cos \theta_1 \cos(\theta_1 + \theta_2) \]  
\[ \cos \theta_1 = \frac{p_x^2 + p_y^2 - l_1^2 - l_2^2}{2l_1l_2} \]  
\[ \sin \theta_1 = \pm \sqrt{1 - \left(\frac{p_x^2 + p_y^2 - l_1^2 - l_2^2}{2l_1l_2}\right)^2} \]  

Finally, two possible solutions for \( \theta_2 \) can be written as:

\[ \theta_2 = \arctan 2\left(\pm \sin \theta_2, \cos \theta_2\right) \]  

Then, multiply each side of Equation 1 by \( \cos \theta_1 \) and Equation 2 by \( \sin \theta_2 \) and add the resulting equations in order to find the solution of \( \theta_1 \) in terms of link parameters and the known variable \( \theta_2 \):

\[ \cos \theta_1 \theta_2 = l_1 \cos \theta_1 + l_1 \cos \theta_1 \cos \theta_2 - l_2 \cos \theta_1 \sin \theta_1 \sin \theta_2 \]
\[ \sin \theta_1 \theta_2 = l_1 \sin \theta_1 \cos \theta_2 + l_1 \sin \theta_1 \cos \theta \sin \theta_2 + l_2 \sin \theta_1 \cos \theta_1 \sin \theta_2 \]
\[ \cos \theta_1 \theta_2 + \sin \theta_1 \theta_2 = l_1 \left(\cos^2 \theta_1 + \sin^2 \theta_2\right) \]
\[ +l_1 \cos \theta_1 \left(\cos^2 \theta_1 + \sin^2 \theta_2\right) \]
\[ -\sin \theta_1 \theta_2 = l_1 \sin \theta_1 \cos \theta_2 - l_1 \sin \theta_1 \cos \theta \sin \theta_2 \]
\[ +l_2 \cos \theta_2 \sin \theta_1 \cos \theta_1 \sin \theta_2 + l_2 \cos \theta_1 \sin \theta_2 \]  
\[ +\cos \theta_1 \theta_2 = l_1 \sin \theta_1 \left(\cos^2 \theta_1 + \sin^2 \theta_2\right) \]
\[ +l_2 \sin \theta_1 \cos \theta_1 \sin \theta_2 + l_2 \cos \theta_1 \sin \theta_2 \]
\[ +\cos \theta_1 \theta_2 = l_1 \sin \theta_1 \left(\cos^2 \theta_1 + \sin^2 \theta_2\right) \]

The simplified equation obtained as follows:

\[ \cos \theta_1 \theta_2 + \sin \theta_1 \theta_2 = l_1 + l_2 \cos \theta_2 \]  
\[ -\sin \theta_1 \theta_2 + \cos \theta_1 \theta_2 = l_2 \sin \theta_2 \]
Now, multiply each side of Equation 6 by \( p_x \) and Equation 7 by \( p_y \), and add the resulting equations in order to obtain \( \cos \theta_1 \):

\[
\cos \theta_1 = \frac{p_x (l_1 + l_2 \cos \theta_2) + p_y l_2 \sin \theta_2}{p_x^2 + p_y^2}
\]

Equation (8)

\[
\sin \theta_1 \text{ is obtained as:}
\]

\[
\sin \theta_1 = \pm \sqrt{1 - \left(\frac{p_x (l_1 + l_2 \cos \theta_2) + p_y l_2 \sin \theta_2}{p_x^2 + p_y^2}\right)^2}
\]

Equation (9)

As a result, two possible solutions for \( \theta_1 \) can be written:

\[
\theta_1 = \arctan 2 \sqrt{1 - \left(\frac{p_x (l_1 + l_2 \cos \theta_2) + p_y l_2 \sin \theta_2}{p_x^2 + p_y^2}\right)^2, \frac{p_x (l_1 + l_2 \cos \theta_2) + p_y l_2 \sin \theta_2}{p_x^2 + p_y^2}}}
\]

Equation (10)

Using Equations 5 and 10 joint variables are found by substituting the given input values. In this method laborious calculations involved and it increases by increasing the series of linkages. So it is necessary to develop some programming codes. Till now the researchers utilized Artificial Neural Networks (ANN), MATLAB, C&C++, Fuzzy Logics and some statistical techniques. In this study a programmable code developed for Inverse Kinematics (IK) of a robotic manipulator in simple manner to understand by the young people. Python is an efficient and easy to learn programming language. One of the most popular applications of Python is in numerical analysis. Solving problems related to numerical analysis using programming languages makes the task easier. It takes less effort to code using Python compared to other programming languages like C/C++/Java. Within a shorter period of time, one can do more with relatively less code. Also, unlike C, Python has a large library of built-in functions and the programming syntax is much simpler. It could save a great deal of time since it allows one to focus on actual research rather than tool being used.

### Results and Discussion

In the inverse kinematics, geometric approach contains lot of mathematical expressions. These are increased by increasing number of linkages causes more complexity to solve manually but it is little bit easy in MATLAB. In this study a simplified code generated in PYTHON environment in easy way with fast generation of output results. In the present discussion two and three link planer manipulators were selected and the same procedure can also be applied to complex robotic linkages. For this case selected a linkages of length 10 cm and 5 cm to reach a position (12.99 cm, 2.5 cm) as \( x \), \( y \) positions of end effector or tip of the manipulator. To found the joint variables of manipulator linkages, Equation 5 and 10 are applied and obtained the values. Then these values are compared with PYTHON outputs.

The following is the code developed for Inverse Kinematics (IK) of a two link RR planar robotic manipulator in the PYTHON environment.

```python
import math
class Kine2:
    def calcAngle(self, reqcos):
        res1 = math.atan2(math.sqrt(1 - math.pow(reqcos, 2)), reqcos)
        res2 = math.atan2(math.sqrt(1 - math.pow(reqcos, 2)), reqcos)
        return (res1, res2)
    def inputCalc(self):
        px = float(input("px:"))
        py = float(input("py:"))
        l1 = float(input("l1:"))
        l2 = float(input("l2:"))
        ctheta2 = (px * (l1 + l2 * ctheta2) + py * l2 *
                      math.pow(ctheta2, 2)) / (2 * (1 * l2) * l2)
        stheta2 = math.sqrt(1 - math.pow(ctheta2, 2))
        ctheta1 = (px * (1 + l1 * ctheta2) + py) / (px**2 + py**2)
        stheta1 = math.sqrt(1 - math.pow(ctheta1, 2))
        print("theta1: {} \ and \ {} ").format(math.degrees(theta1),
                      math.degrees(ctheta1))
        print("theta2: {} \ and \ {} ").format(math.degrees(theta2),
                      math.degrees(ctheta2))
        kine2 = Kine2()
        kine2.inputCalc()
```

### Inputs and Outputs:
px: 12.99 cm
py: 2.5 cm
l1: 10 cm
l2: 5 cm
theta1: 8.21477006006 and -8.21477006006
theta2: 54.5073278888 and -54.5073278888

---

R. Venkata Neeraj Kumar and R. Sreenivasulu / Journal of Mechatronics and Robotics 2019, Volume 3: 542-551
DOI: 10.3844/jmnrsp.2019.542.551
Fig. 3: Elbow up position and orientation of 2 link manipulator for first solution

Fig. 4: Elbow down position and orientation of 2 link manipulator for first solution
The following are the different configurations executed by feeding the input values in developed programme and depicted in the following Fig. 3 to 6.

**Case II. Three Link RR Planar Manipulator**

Consider a three link RR planar manipulator along xy plane. For this mechanism also, similar geometric approach attempted for obtaining inverse solution. But the results in this method revealed that when number of links are increased then the solution procedure becomes complex. So for serial planer manipulators geometric approach gives complex solution. In this connection, PYTHON code provides easy steps to achieve solution and it helps to reduce the time and also gives accurate performance while running the programme comparing with MATLAB. The programme code and elbow down and elbow up configurations shown in the following Fig. 7 to 9.
Fig. 7: Geometric model diagram of 3 link manipulator

Fig. 8: Elbow down position and orientation of 3 link manipulator for first solution
The following is the code developed for Inverse Kinematics (IK) of a three link RR planar robotic manipulator in the PYTHON environment.

```python
import math
class Kine3:
    def cosCalc(self, a, b, c):
        angle = math.acos((a*a + b*b - c*c)/(2*a*b))
        return angle

    def inputCalc(self):
        xe = float(input("xe: "))
        ye = float(input("ye: "))
        phie = math.radians(float(input("phie in degrees: ")))
        l1 = float(input("l1: "))
        l2 = float(input("l2: "))
        l3 = float(input("l3: "))

        xw = xe - l3 * math.cos(phie)
        yw = ye - l3 * math.sin(phie)
        r = math.sqrt(xw**2 + yw**2)
        gamma = self.cosCalc(r, l1, l2)

        theta2 = math.pi - self.cosCalc(l1, l2, r)
        theta1 = math.atan2(yw, xw) - gamma
        theta3 = phie - theta1 - theta2

        print("theta1: {} and {}.format(math.degrees(theta1), math.degrees(theta1 + 2 * gamma))")
        print("theta2: {} and {}.format(math.degrees(theta2), math.degrees(theta2 - 1))")
        print("theta3: {} and {}.format(math.degrees(theta3), math.degrees(theta3 + 2 * (theta2 - gamma)))")

kine3 = Kine3()
kine3.inputCalc()

print("theta1:")
```

**Inputs and Outputs**

- $\text{xe}: 12.99 \text{ cm}$
- $\text{ye}: 2.5 \text{ cm}$
- $\text{phie in degrees}: 130$
- $\text{l1}: 10\text{ cm}$
- $\text{l2}: 5\text{ cm}$
- $\text{l3}: 2\text{ cm}$
- $\text{theta1}: -8.29738622523704 \text{ and } 16.05539094694738$
- $\text{theta2}: 37.12800501988501 \text{ and } -37.12800501988501$
- $\text{theta3}: 101.1697336026387 \text{ and } 151.07261407293765$

The following configurations drawn for every elbow up and elbow down positions of manipulator links to reach a desired position.
Conclusion
The proposed approach has been demonstrated for a two and three link planar manipulators, it should be examined for a few more intricate arrangements of robotic linkages such as greater than two degrees of freedom manipulators and parallel linkages. This can be possibly implemented for different model cases in which the solution for the mathematical expression of the inverse kinematics is using PYTHON software code.

Acknowledgement
The authors are thankful to management, principal and department of mechanical engineering and computer science engineering staff for providing advices while preparing this paper.

Funding Information
There is no funding agency to support for preparation of this manuscript.

Author’s Contributions
R. Sreenivasulu: Developed the theoretical frame work and supervised the research.
R. Venkata Neeraj Kumar: Developed a programme in PYTHON and obtained the results.

Ethics
This article is an original research paper. There are no ethical issues that may arise after the publication of this manuscript.

References
Chaitanyaa, G. and S. Reddy, 2016. Genetic algorithm based optimization of a two Link planar robot manipulator. Int. J. Lean Think., 7: 1-3.
Chen, Q., S. Zhu and X. Zhang, 2015. Improved inverse kinematics algorithm using screw theory for a six-DOF robot manipulator. Int. J. Adv. Robotic Syst., 12: 140-140. DOI: 10.5772/60834
Chirikjian, G.S., 1994. Kinematics of a metamorphic robotic system. Proceedings of the IEEE International Conference on Robotics and Automation, May 8-13, IEEE Xplore Press, San Diego, CA, USA, pp: 449-455. DOI: 10.1109/ROBOT.1994.351256
Craig, J.J., 1989. Introduction to robotics. Addison-Wesley.
Raheem, F.A., A.T. Sadiq and N.A.F. Abbas, 2019. Robot arm free Cartesian space analysis for heuristic path planning enhancement. Int. J. Mech. Mechatron. Eng., 19: 29-42.

Hudgens, J.C. and D. Tusor, 1988. Fully-parallel six degree-of-freedom micromanipulator: Kinematic analysis and dynamic model. Proceedings of the Trends and Developments in Mechanisms, Machines and Robotics, Sept. 25-28, Kissimmee, FL, USA, pp: 29-37.
Hussain, M.A. and B. Nobie, 1985. Applications of MACSYMA to Kinematics and Mechanical System. In: Application of Computer Algebra, Pavell, R. (Ed.), Kluwer Academic Publishers, Dordrecht, pp: 262-280.
Jones, B.A. and I.D. Walker, 2006. Kinematics for multisection continuum robots. IEEE Trans. Robot., 22: 43-55. DOI: 10.1109/TRO.2005.861458
Kanayama, Y., Y. Kimura, F. Miyazaki and T. Noguchi, 1990. A stable tracking control method for an autonomous mobile robot. Proceedings of the IEEE International Conference on Robotics and Automation, May 13-18, IEEE Xplore Press, Cincinnati, OH, USA, pp: 384-389. DOI: 10.1109/ROBOT.1990.126006
Kircanski, M. and M. Vukobratovic, 1985. Computer-aided generation of manipulator kinematics models in symbolic form. Proceedings of the 15th ISIR, (SIR’ 85), pp: 1043-1049.
Kircanski, M. and M. Vukobratovic, 1986. A new program package for generating symbolic kinematics models of arbitrary serial-link manipulators. Proceedings of the 16th ISIR, (SIR’ 85), pp: 249-258.
Malley, 2011. Introduction to robotics, inverse manipulator kinematics. Int. J. Soft Comput.
Mandava, R.K. and P.R. Vundavilli, 2016. Forward and inverse kinematic based full body gait generation of biped robot. Proceedings of the International Conference on Electrical, Electronics and Optimization Techniques, Mar 3-5, IEEE Xplore Press, Chennai, India, pp: 3301-3305. DOI: 10.1109/ICEEOT.2016.7755317
Mohamed, M.G. and J. Duffy, 1985. A direct determination of the instantaneous kinematics of fully parallel robot manipulators. J. Mechanisms Transmiss. Automat. Design, 107: 226-229. DOI: 10.1115/1.3258713
Morris, R.D., 1987. A symbolic matrix manipulation package for the kinematics analysis of robot manipulator, CIME, 5: 40-56.
Murray, R.M., 2017. A Mathematical Introduction to Robotic Manipulation. 1st Edn., CRC Press, Boca Raton, ISBN-10: 1351469789, pp: 503.
Nugroho, S.A., A.S. Prihatmanto and A.S. Rohman, 2014. Design and implementation of kinematics model and trajectory planning for NAO humanoid robot in a tic-tac-toe board game. Proceedings of the IEEE 4th International Conference on System Engineering and Technology, Nov 24-25, IEEE Xplore Press, Bandung, Indonesia, pp: 1-7. DOI: 10.1109/ICSEngT.2014.7111783
Radavelli, L., R. Simoni, E. De Pieri and D. Martins, 2012. A comparative study of the kinematics of robots manipulators by Denavit-Hartenberg and dual quaternion. Mecánica Comput. Multi-Body Syst., 31: 2833-48.

Sadiq, A.T., F.A. Raheem and N.A. Abbas, 2017. Optimal trajectory planning of 2-DOF robot arm using the integration of PSO based on D* algorithm and cubic polynomial equation. Proceedings of the 1st International Conference for Engineering Researches, (CER’ 17), Middle Technical University, Baghdad-Iraq, pp: 458-467.

Sreenivasulu, R., 2012. Simulation of desired end point trajectory for a 2-dof planar manipulator. Int. J. Adv. Scientific Technical Res., 5: 688-696.

Sun, J.D., G.Z. Cao, W.B. Li, Y.X. Liang and S.D. Huang, 2017. Analytical inverse kinematic solution using the D-H method for a 6-DOF robot. Proceedings of the 14th International Conference on Ubiquitous Robots and Ambient Intelligence, Jun. 28-Jul. 1, IEEE Xplore Press, Jeju, South Korea, pp: 714-716. DOI: 10.1109/URAI.2017.7992807

Tsai, L.W. and A.P. Morgan, 1985. Solving the kinematics of the most general six-and five-degree-of-freedom manipulators by continuation methods. J. Mechanisms Transmiss. Automat. Design, 107: 189-200. DOI: 10.1115/1.3258708

Tsai, M.J. and Y.H. Chiou, 1989. Symbolic Equation Generation for Manipulators (SEGM). Proceeding of the 4th International Conference on Advanced Robotics, (CAR’ 89), Columbus, OH, pp: 35-61.

Veitschegger, W. and C.H. Wu, 1986. Robot accuracy analysis based on kinematics. IEEE J. Robot. Automat., 2: 171-179. DOI: 10.1109/JRA.1986.1087054

Webster III, R.J. and B.A. Jones, 2010. Design and kinematic modeling of constant curvature continuum robots: A review. Int. J. Robot. Res., 29: 1661-83. DOI: 10.1177/0278364910368147

Yang, C., H. Ma and M. Fu, 2016. Advanced Technologies in Modern Robotic Applications. 1st Edn., Springer, Singapore, ISBN: 9811008299, pp: 419.

Zhuang, H., Z.S. Roth and F. Hamano, 1992. A complete and parametrically continuous kinematic model for robot manipulators. IEEE Trans. Robot. Automat., 8: 451-63. DOI: 10.1109/70.149944

Khatib, O., 1987. A unified approach for motion and force control of robot manipulators: The operational space formulation. IEEE J. Robotics Automation, 3: 43-53.

Lloyd, J. and V. Hayward, 1993. Trajectory generation for sensor-driven and time-varying tasks. Int. J. Robotics Res., 12: 380-393.