Quest Journals
Journal of Software Engineering and Simulation
Volume 9 ~ Issue 10 (2023) pp: 05-10
ISSN(Online) :2321-3795 ISSN (Print):2321-3809
www.questjournals.org

Research Paper

Simulation Implementation of Robot Kinematics Inverse Solution Algorithm Based on Newton Raphson Iteration in CoppeliaSim

Yan Zhang[1], Guofen Zhang[2]
[1](School of Information Science and Technology, Taishan University)
[2](School of Information Science and Technology, Taishan University)
Corresponding Author: Yan Zhang

ABSTRACT: *This study is based on the Newton Raphson iterative algorithm to study the problem of solving inverse kinematics of robots. By establishing a robot model in CoppeliaSim and conducting simulation experiments on the algorithm using Python code, evaluations were conducted from different iterations, target poses, and error analysis. The results show that the algorithm has good performance and accuracy, and can provide effective solutions for solving robot inverse kinematics problems. In the future, further experimental verification can be conducted on actual robots, and other advanced algorithms and technologies can be studied to obtain more efficient and accurate solution methods.*

KEYWORDS: *Inverse kinematics, Newton Raphson algorithm, Target pose, CoppeliaSim*

R*eceived 15 Oct., 2023;* ***R****evised 28 Oct., 2023;* ***A****ccepted 31 Oct., 2023 © The author(s) 2023.*
P*ublished with open access at* ***www.questjournals.org***

I. INTRODUCTION

Robot inverse kinematics solution refers to solving the joint angles of each joint of the robot based on a given target position and posture. It is a very important research field with extensive application value in robot motion control, path planning, visual navigation, and other fields. By solving inverse kinematics, robots can autonomously move in complex environments, achieving precise position control and posture adjustment.

The Newton Raphson iterative algorithm is a numerical method for solving the roots of nonlinear equations. This algorithm is based on the Newton method and gradually approximates the roots of the equation through an iterative process. It utilizes the idea of Taylor series to expand the roots of nonlinear equations into power series form, and gradually approximates the true roots of the equation by iteratively updating the approximate solution. The Newton Raphson iterative algorithm has a high convergence speed and usually performs well in dealing with nonlinear problems.

CoppeliaSim is a powerful robot simulation software that provides a rich library of robot models and physical engine simulation functions, supporting 3D visualization environments and script programming. By using CoppeliaSim, researchers can simulate the motion process of robots on computers, evaluate and optimize their performance. In addition, CoppeliaSim can also be used for the verification and testing of robot control algorithms, providing strong support for the development and application of robots.

The aim of this study is to use the Newton Raphson iterative algorithm to solve the inverse kinematics of robots through CoppeliaSim simulation software, in order to improve the motion performance and control accuracy of robots. This study used CoppeliaSim simulation software to establish a robot model and improved the Newton Raphson iterative algorithm to improve the convergence speed and stability of the algorithm. Experimental verification was conducted on the inverse kinematics solution algorithm; Evaluate the effectiveness and feasibility of the algorithm through experimental results analysis, and further optimize the algorithm; Provide theoretical support and experimental basis for the intelligent and autonomous development of robots.

This study has important theoretical significance and practical value. Firstly, by studying the optimization methods of the Newton Raphson iterative algorithm, we can further deepen our understanding of numerical methods for solving nonlinear equations and enrich the theory of numerical computation. Secondly, using CoppeliaSim simulation software for experimental verification of robot inverse kinematics solution can avoid risks and uncertainties in actual operation, and provide effective experimental means for the intelligent and autonomous development of robots. Finally, the results of this study can be applied to the fields of path planning, motion control, visual navigation, etc. of robots, improving their motion performance and control accuracy, and providing important technical support for practical applications.

The existing algorithms for solving robot inverse kinematics based on Newton Raphson iteration still have problems in terms of convergence speed, stability, and adaptability to diverse environments. With the complexity of robot application scenarios, the difficulty of solving inverse kinematics is gradually increasing. Therefore, further research and improvement of algorithms are needed to meet the needs of practical applications such as real-time and adaptability.

II. NEWTON RAPHSON ITERATIVE ALGORITHM

The Newton Raphson algorithm is an iterative algorithm used to solve the roots of nonlinear equations. This algorithm approximates the true roots of the equation through continuous iterations, and each iteration calculates the next approximate solution based on the current approximate solution until the preset accuracy requirements or iteration limit are met.

The Newton Raphson formula is as follows

$$x_1 = x_0 - \frac{f(x_0)}{f'(x_0)}$$

The mathematical description of the Newton Raphson iteration method is shown in Figure 1. Let the nonlinear equation that requires a solution be f(x)=0, where f(x) is a continuous differentiable function. The Newton Raphson iteration method approximates the roots of the equation by:

(1) Choose an initial approximate solution x_0;

(2) According to the formula of Newton's method, calculate the next approximate solution x_1;

(3) Using the current approximate solution x_1 and the derivative $f'(x_1)$ of the Newton method, calculate the next approximate solution x_2;

(4) If the preset stopping conditions are met (such as reaching the preset number of iterations or error tolerance), stop the iteration and output the current approximate solution as the root of the equation; Otherwise, return to step 2 to continue iteration.

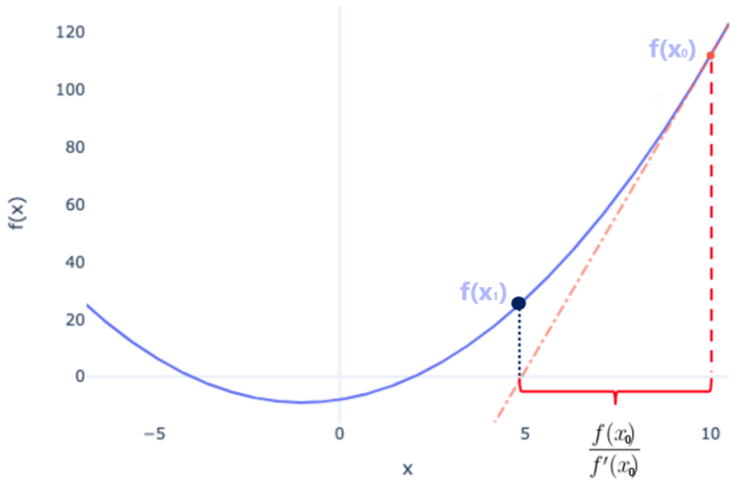

Figure 1. Newton Raphson Method

The selection of initial approximate solutions has a significant impact on the convergence and speed of the algorithm. In general, one can choose an approximate value of the equation as the initial solution, or try using multiple different initial solutions to obtain better results. If some prior information or initial estimates can be obtained, this information can also be used to select better initial approximate solutions.

Robot inverse kinematics refers to solving the joint angles of each joint of a robot based on a given target position and posture. The specific steps to solve the inverse kinematics problem of a robot using the Newton Raphson iterative algorithm are as follows:

(1) Transforming the inverse kinematics problem of robots into nonlinear equation form:

$$\theta_{n+1} = \theta_n - (J^T J)^{-1} J^T(\theta_n)\Delta f(\theta_n)$$

Among them θ is the joint angle. f(θ) is the forward kinematics solution function. J is the Jacobian matrix, which collects all first-order partial derivatives of a multivariate function f(θ).

(2) Select appropriate initial approximate solutions;

(3) Using the Newton Raphson iteration method for iterative solution;

(4) Determine whether convergence occurs based on the preset stop conditions. If convergence occurs, output the current approximate solution as the inverse kinematics solution. Otherwise, return to step 3 to continue the iteration. In the implementation process, it is also necessary to consider factors such as motion constraints, joint range, and physical limitations of the robot to ensure the feasibility and effectiveness of the solution.

III. INTRODUCTION TO COPPELIASIM

CoppeliaSim is a powerful robot simulation software that provides a rich library of robot models and physical engine simulation functions, supporting 3D visualization environments and script programming.

CoppeliaSim is equipped with a high-performance physics engine that can simulate real-world physical environments, including factors such as gravity, friction, and air resistance. This enables researchers to accurately evaluate and optimize the motion performance of robots in simulation environments. It has an intuitive 3D visualization interface that can display the robot's motion and posture in real-time, making it convenient for researchers to observe and analyze. It provides a large number of robot models, and researchers can choose suitable models for simulation experiments according to their needs.

CoppeliaSim provides an easy-to-use application program interface (API) that supports multiple programming languages such as C++, Python, and more. Researchers can use APIs to call various functions and objects in the simulation environment to achieve customized tasks such as robot control and sensor data collection. Users can achieve complex simulation experiments by writing Python script files.

CoppeliaSim has been widely applied in the field of robotics, including robot motion planning, control strategy research, sensor data fusion, and other aspects.

IV. IMPLEMENTATION OF INVERSE KINEMATICS SOLUTION IN COPPELIASIM

4.1IMPORTING ROBOT MODELS

In CoppeliaSim, various formats of robot model files can be imported, and the robot model can be configured and adjusted. For example, the joint parameters and initial posture of the robot can be set. CoppeliaSim also includes some common robots. This article takes ABB's IRB 140 as an example to discuss the inverse kinematics solution algorithm of this robot. The robot model is shown in Figure 2.

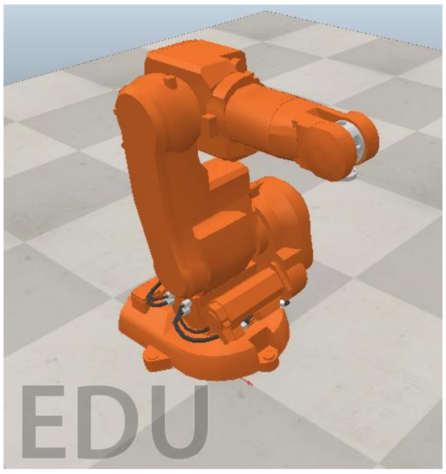

Figure 2. ABB IRB 140 model

The Denavit-Hartenberg parameter is a widely used parameterization method in robotics, which is used to attach a reference coordinate system to the link of a spatial motion chain or robotic arm. It describes the relative position and direction between adjacent links by defining four parameters. The Modified Denavit-Hartenberg parameters connect the link coordinate system to the near end of each link instead of the far end, thereby improving the symbol representation, making it clearer and cleaner, with better universality and scalability. The Modified Denavit-Hartenberg parameters of the ABB robot model are shown in Table 1.

Table 1. Modified Denavit-Hartenberg parameters of ABB IRB

Link	α_{i-1}	a_{i-1}	θ_i	d_i
1	0º	0.06897	0º	0.275
2	90º	0.07	90º	0
3	0º	0.36	0º	0
4	90º	0	180º	0
5	90º	0	-180º	0
6	90º	0	0º	0

4.2IMPLEMENTATION OF SOLUTION

To achieve robot inverse kinematics solution in CoppeliaSim, it is necessary to conduct mathematical modeling for specific robot models and convert the models into executable code.

Firstly, it is necessary to give the target pose of the robot's end tool center point (TCP), which means that the robot's final goal is to align TCP with the target position and pose by changing the joint angle. In CoppeliaSim, the target pose can be set by setting the Dummy pose variable. Specifically, it is necessary to obtain the position and attitude matrices of Dummy, which can be achieved through the *sim.getObjectMatrix()* function.

Before conducting the Newton Raphson iterative algorithm, it is necessary to determine the initial joint angle of the robot. In CoppeliaSim, the initial joint angle can be determined by reading the joint angle of the robot model using the function *sim.getJointPosition()*. It should be noted that the selection of the initial joint angle has a certain impact on the convergence and speed of the iterative algorithm, so it needs to be selected according to the actual situation. To calculate the inverse kinematics solution of a robot, it is first necessary to calculate its forward kinematics solution based on DH parameters, providing a basis for calculating the derivatives of nonlinear equations using numerical methods in the later stage and avoiding complex formula derivation. The following is the Python code used to implement the forward kinematics problem of a robot.

```
#python
import numpy as np
def forward_kinematics(jointspos,i=0,offset=0):
    theta=jointsangle.copy()
    theta[i]+=offset
    M=np.eye(4,4)
    for i in range(6):
        M=M@ np.array([ [cos(theta[i]), -sin(theta[i]),0,a[i]],
                        [sin(theta[i])*cos(alpha[i]),cos(theta[i])*cos(alpha[i]),-sin(alpha[i]),-d[i]*sin(alpha[i])],
                        [sin(theta[i])*sin(alpha[i]),cos(theta[i])*sin(alpha[i]),cos(alpha[i]),d[i]*cos(alpha[i])],
                        [0, 0, 0, 1]])
    return M[0:3].reshape(-1)
```

The function forward_ kinematics() in the above code calculates the target pose matrix through 6 joint angles. Implementing the Newton Raphson iterative algorithm in CoppeliaSim requires writing corresponding script code. The following is a Python code used to implement the Newton Raphson iterative algorithm to solve the inverse kinematics problem of robots.

```
#python
import numpy as np
def inverse_kinematics():
    theta=[]
    for j in joints:
        theta+=[sim.getJointPosition(j)]
    target_vector=np.array(sim.getObjectMatrix(target,-1))
    for iteration_times in range(10):
        vector= forward_kinematics(theta)
        jacobian_matrix=np.array([[0]*6]*12])
        dx=0.001
        for i in range(6):
            k=((forward_kinematics(theta,i,dx)-vector)/dx+(forward_kinematics(theat,i,dx)-vector)/dx)/2
            for row in range(12):
                jacobian_matrix[row][i]=k[row]
        jacobian_inverse=np.linalg.pinv(jacobian_matrix)
        delta_theta= jacobian_inverse@(vector-target_vector)
        theta=theta- delta_theta
        if (delta_theta **2).sum()<1e-4:
            break
    return theta
```

In the above code, the first step is to import the required libraries and modules, including CoppeliaSim's related functions and numpy libraries. Then, obtain the joint angles of the initial 6 joints and the final target pose; Next, calculate the target pose based on the current joint angle, and use numerical methods to calculate the derivative under the current joint angle, thereby establishing a jacobian matrix of 12x6; During the iterative solution process, calculate the joint angle increment based on the jacobian matrix, target pose, and current pose; Finally, update the joint angle and determine whether the convergence condition has been met. If the convergence condition is reached, i.e. the mean square deviation of joint angle increment tends to 0, the final solution result is output.

4.3 Analysis of simulation results

During the simulation experiment, we collected data on the solution results under different iterations and target postures. The following will analyze these data and provide conclusions.

(1) Comparison of solution results under different iterations:

We tested the solution results of the algorithm under different iterations, and the specific data is shown in the table 2:

Table 2. Modified Denavit-Hartenberg parameters of ABB IRB

Iteration Times	Solution Time (s)	Error(Mean square deviation)
10	0.015	0.0181
20	0.045	0.0113
30	0.076	0.0091
40	0.108	0.0052
50	0.125	0.0047

From the above table, it can be seen that as the number of iterations increases, the solution time and error gradually decrease. Especially when the number of iterations reaches 40 or more, the error has decreased to below 0.005 rad, indicating that the algorithm has good convergence and accuracy. Meanwhile, as the number of iterations increases, the calculation time also gradually increases, so it is necessary to balance between error and calculation time.

(2) Comparison of solution results under different target postures:

In order to test the performance of the algorithm under different target poses, we set different target poses and recorded the data of the solution results. The specific data is shown in the table 3:

Table 3. Modified Denavit-Hartenberg parameters of ABB IRB

Target pose matrix[12x1]	Solution Time (s)	Error (Mean square deviation)
0,-0.866,-0.5,-0.243,-0.342,0.47,-0.814,-0.071,0.94,0.171,-0.296,0.610	0.011	0.0052
0.296,-0.814,-0.5,-0.243,-0.482,0.325,-0.814,-0.171,0.825,0.482,-0.296,0.559	0.029	0.0061
0.296,-0.814,-0.5,-0.043,-0.332,0.403,-0.853,0.029,0.896,0.418,-0.15,0.559	0.013	0.0097
0.433,-0.75,-0.5,-0.243,-0.748,0.01,-0.663,0.029,0.503,0.661,-0.557,0.345	0.017	0.0011

From the above table, it can be seen that as the target pose increases, the solution time and error also gradually increase. However, regardless of how the target pose changes, the algorithm can obtain relatively accurate results. Therefore, the algorithm has good adaptability and robustness.

V. CONCLUSION

This study is based on the Newton Raphson iterative algorithm to study the problem of solving inverse kinematics of robots. By conducting simulation experiments in CoppeliaSim, evaluation and analysis were conducted from various aspects such as different iterations, target poses, algorithm robustness, and error analysis.

Simulation experiments have verified the feasibility and effectiveness of the robot inverse kinematics solution method based on the Newton Raphson iterative algorithm. Under different iterations, as the number of iterations increases, the error of the solution gradually decreases, and the convergence and stability gradually improve. Under different target poses, the algorithm can adapt to different target poses and obtain more accurate results, with a certain degree of adaptability and robustness.

Based on the results of this study, it is possible to further explore and improve the convergence speed and stability of the Newton Raphson iterative algorithm, in order to solve robot inverse kinematics problems more efficiently. The method of this study can be applied to practical robot systems to verify its practicality and reliability. In the future, technologies such as deep learning can be combined to improve the adaptability and generalization ability of algorithms.

REFERENCES

[1]. Xiaoqing Lv and Ming Zhao, Application of Improved BQGA in Robot Kinematics Inverse Solution. Journal of Robotics, 2019. Article ID 1659180.

[2]. Y. LIN, Solution of Inverse Kinematics for General Robot Manipulators Based on Multiple Population Genetic Algorithm. Journal of Mechanical Engineering, 2017. 53(3):p. 1.

[3]. Roland Szabo and Radu-Stefan Ricman, Robotic Arm Position Computing Method in the 2D and 3D Spaces. Actuators, 2023. 12(112): p. 112.

[4]. Liu Yiyang, et al, A General Robot Inverse Kinematics Solution Method Based on Improved PSO Algorithm. IEEE Access, 2021. 9: p. 32341-32350.

[5]. Huizhi Zhu, et al, A Novel Hybrid Algorithm for the Forward Kinematics Problem of 6 DOF Based on Neural Networks. Sensors, 2022. 22: p. 5318.

[6]. Luis Antonio Orbegoso Moreno, Kinematic control for a quadruped robot using the Newton-Raphson method. Revista Elektrón, 2021. 5(1): p. 77-82.

[7]. Mingzhe Liu and Qiuxiang Gu, Kinematics Model Optimization Algorithm for Six Degrees of Freedom Parallel Platform, 2023. 13: p. 3082.

[8]. Yang Wang, Liming Wang, Yonghui Zhao, Research on Door Opening Operation of Mobile Robotic Arm Based on Reinforcement Learning. Applied Sciences, 2022. 12: p.5204.

[9]. https://www.coppeliarobotics.com/coppeliaSim.

[10]. Štefan Ondočko, et al, Inverse Kinematics Data Adaptation to Non-Standard Modular Robotic Arm Consisting of Unique Rotational Modules. Applied Sciences, 2021. 11: p. 1203.

[11]. Jorge Gustavo Pérez-Fuentevilla, et al, Synchronization Control for a Mobile Manipulator Robot (MMR) System: A First Approach Using Trajectory Tracking Master–Slave Configuration. Machines, 2023. 11: p. 962.