Numerical Inverse Kinematic Solution for a 7-Dof Robot Arm

Hao Li, Tiehao Wang and Yihsuan Cheng

***Abstract*— This article attempts several ways to obtain numerical inverse kinematic solutions for a 7-Dof Robot Arm. Three numerical solutions, including Cyclic Coordinate Descent, Jacobian Inverse, Jacobian Transpose, are the focus of disscusion. In addition, analytic solution is mentioned as well in comparison with the numerical solutions.**

I. INTRODUCTION

A. Inverse Kinematic and Related Work

From the perspective of forward kinematics, the position and orientation of the end effector are determined by the given joint variables. It is a static geometric computation of the manipulator position and orientation. On the contrary, inverse kinematics computes all possible sets of joint variables that could be used to attain a given position and orientation of the end effector of the manipulator. In general, it is not likely to calculate the solutions by hand for people while using analytic and numerical algorithms are obviously helpful. Inverse kinematics is widely used in many scientific fields. However, the problem is inherently underdetermined to some extent. There are generally many, even infinite, possible poses that satisfy given positions and orientation of the end-effector. Therefore, selecting the optimal or desired solution from these candidates is possible and desirable. For example, it is preferable to have the robot arm work cooperatively to reach desired position rather than letting one of the joints totally reach its limit but keeping other initial positions. In addition, specific poses are better out of the reason for avoiding obstacles or saving energy. Thus taking the advantage of redundancy of the robot arm can dramatically help solving the problem. In order to solve inverse kinematics problem, several ways have been developed. The damped least squares methods for inverse kinematics, jacobian transpose and so on[1] were introduced. Additonally, CCD is another efficient way to have the problem solved[2].

B. Structure Organization of This Report

In chapter 2, the related theories of CCD, jacobian inverse, jacobian transpose and analytical solutions will be discussed. Then in chapter 3, an experiment in 5-Dof robot arm and Franka Emika 7-Dof robot arm adopting four solutions would be processed and different further steps would be executed depending on respective solution, such as taking the advantage of redundancy for jacobian inverse and jacobian transpose while joint limit is introduced for CCD. Then the comparison would be made between them and the conclusion would be drawn in chapter 4.

II. BASIC THEORIES OF THREE NUMERICAL AND ONE ANAYTIC INVERSE KINEMATIC SOLUTIONS

A. Theories of CCD

The purpose of CCD is to minimize the position error and orientation error between desired and current manipulator end-effector poses by adjusting the angle variables of each joint in succession. The process in one iteration is interpreted in Fig.1. Assume that the adjustment starts from joint 4 to joint 2, and the point at the upper left corner represents the desired position. First the position error between the desired position and current position is calculated, then it is mapped to a corresponding angle change of joint 4; subsequently similar manipulation is applied to joint 3 and 2 until all the joints are adjusted in one iteration. Such an iteration is repeated until the desired end effector pose is reached.

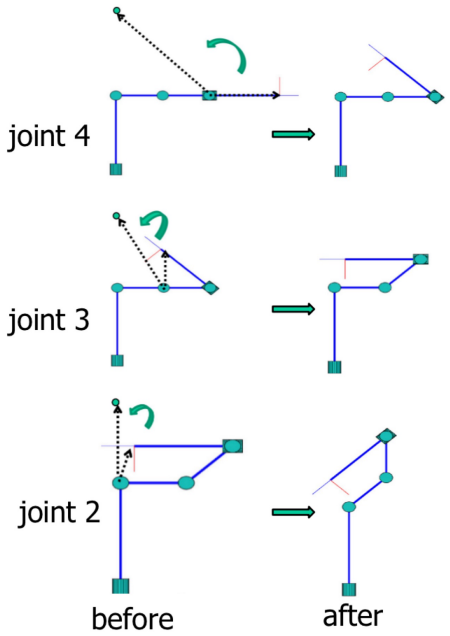


Fig. 1. one iteration of CCD

Mapping from an error to an angle change will be discussed as following, which consists of position error mapping and orientation error mapping. Positon error mapping is interpreted by two equations as in

$$\theta = cos^{-1}\left(\frac{p_e - p_c}{\|p_e - p_c\|} \cdot \frac{p_f - p_c}{\|p_f - p_c\|}\right)$$

$$\vec{d} = \frac{p_e - p_c}{\|p_e - p_c\|} \times \frac{p_f - p_c}{\|p_f - p_c\|}$$

where p_c represents the point of interest now, p_e represents the point of end effector and p_f represents the desired final point and it is shown by Fig.2. Orientation error mapping

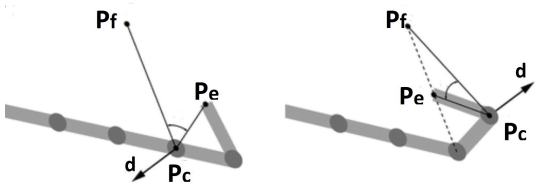


Fig. 2. position mapping

can be implemented by angle and axis representation. Any rotation can be represented as the rotation by a specific angle around a specific axis, as shown in Fig.3. To be precise, as shown in Fig. 3, it is desirable derive the k axis and angle θ from the rotation matrix $R_{k,\theta}$ representing a rotation of θ about k axis which actually defines the orientation error. The angles α and β are used for lengthy calculation deriving the relavant equations, which are not intepreted in detail here.

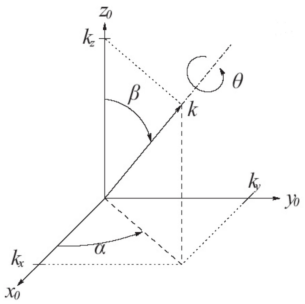


Fig. 3. angle and axis representation

Basically CCD is implemented by adjusting each angle in succession repeatedly until the position error meets specified requirement first and then a similar process is repeated until orientation meets specified requirement. But a trade-off exists between position and orientation requirements because sometimes the descrease of one of them leads to the increase of the other. Therefore, iterations repeat again from beginning until both of them are satisfied. A pseudo code of the algorithm is as follows.

In the implementation of CCD, there is an issue called "joint oscillation" which are supposed to be considered. In the case of fixed direction of iteration, e.g. the angles are always adjusted from bottom to the top or vice versa, the requirement of position and orientation can not be satisfied simutaneously forever. The pose of the manipulator switches repeatedly between two cases shown in Fig. 4. In order to solve this problem, the direction of iteration is alternated, namely, upward and downward adjustments are applied alternatively.

If joint limit is considered, it means three cases as following:

- if a joint variable is adjusted to a new one beyond maximum joint limit, new joint variable is assigned as the maximunm joint limit;

Algorithm 1 Implementation of CCD

Input: p_0: initial position; R_o: intial rotation matrix; p_d: desired position; R_d: desired rotation matrix; p_{th}: position error threshold; θ_{th}: orientation error threshold;

Output: angle variables q_d

```
initial p = p_0 and R = R_0;
repeat
    repeat
        calculate adjustment axis and angles
        adjust current angle
    until (p < p_th)
    repeat
        calculate adjustment axis and angles
        adjust current angle
    until (θ < θ_th)
until (p < p_th&&θ < θ_th)
```

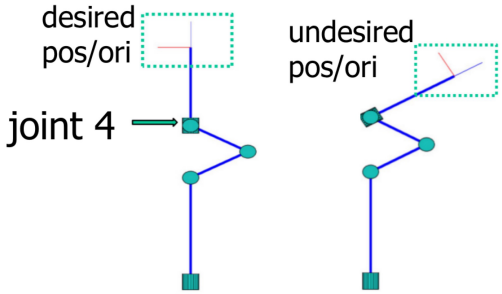


Fig. 4. joint oscillation

- if a joint variable is adjusted to a new one beyond minimum joint limit, new joint variable is assigned as the minimum joint limit;
- if a joint variable is adjusted to a new one within the specified joint limit, new joint variable is kept unchanged.

B. Theories of Jacobian Inverse

Actually, Jacobian Inverse method is a process to model a control system. The purpose of it is to converge the error of position and orientation between current and desired end effector to a preset value. The process of the algorithm is shown in Fig. 5.

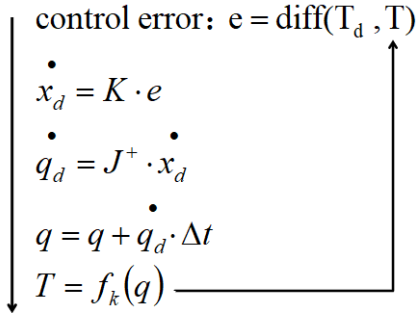


Fig. 5. iteration loop of the Jocobian inverse method

The initial error is got at first by comparing the desired

transformation matrix and the current transformation matrix. The transformation matrices are got by using DH Method[4], [6]. then it is amplified by a constant gain and notified to be a variation rate $\dot{x_d}$ in operational space. Then it is transformed into a value $\dot{q_d}$ in joint space. The slight variation of q is added to the last q. A new Transformation matrix can be got according to the new q. One complete loop is finished. The algorithm will stop when the control error e comes to a desired value.

C. Theories of Jacobian Transpose

The intrinsic target of inverse kinematics algorithm is figuring out a relationship between $\dot{q}$ (angle velocity of every joint) and e (error between the target and current end effector) that makes e converge to zero.[9]

Through the Lyapunov method, it can be verify when

$$\dot{q} = J_A^T(q)Ke$$

the system is asymptotically stable, where $J_A^T(q)$ is Jacobian transpose and K is a positive definite matrix.[7]

And the block diagram is shown in Fig. 6.

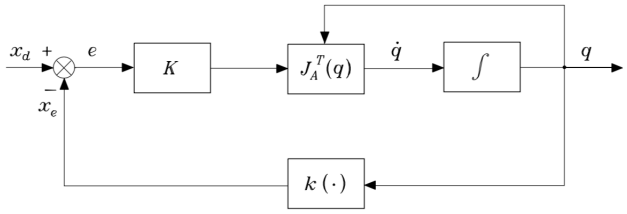


Fig. 6. Block diagram of Jacobian transpose method[7]

The advantages of using Jacobian transpose method are :

a) Simpler computation: The first advantage of taking Jacobian transpose as inverse kinematic algorithm is that the computation is simpler. It only needs to compute forward kinematics and Jacobian transpose on-line, which is simpler than inverse calculation.

b) Singularities: In addition to simpler computation, the other advantage is related to singularities. When a singularity point of the robot is set as the target point, it still works and converge to the zero-error. And the singularity leads to algorithm failure with Jacobian inverse. The reason for the phenomenon is that the a redundant Jacobian martrix leads to no solution for its inverse at singularity point but Jaconian transpose does not.

D. Theories of Analytic Solution

The basic principle of the analytical method is to use the geometrical relationships of robot arms to find the solution of different joints. Pieper[3] showed that the position and the orientation problem of the end-effector of this type of articulated robots (decoupled manipulators) can be independently solved so that the solutions of each angles can be found out one by one. This method is applicable for those robots which have few joints because the complexity of the algorithm arises rapidly with the growth of the number of joints. So it is only used in the 5-DOF robot. The geometric model of the 5-DOF robot is showed in Fig.7.

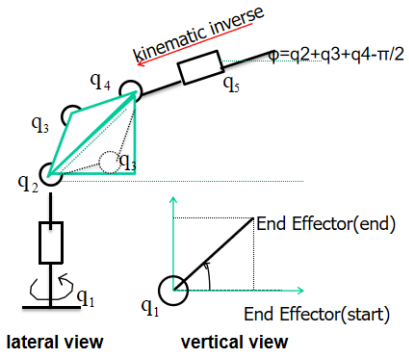


Fig. 7. Geometric model of 5-DOF robot

III. EXPERIMENTS OF FOUR SOLUTIONS

A. Experimental Settings

In order to evaluate the performance of three numerical solutions and one analytical solution, several initial and reference configurations should be predetermined. Then corresponding desired positions and rotation matrics can be calculate from them. Finally the number of iterations from different solutions would be collected and compared, providing the conclusions about convergence speed for 5-Dof and 7-Dof robot arm. For 5-Dof robot arm, initial configuration is arranged as in

$$q_i = \begin{pmatrix} 90^\circ & 45^\circ & 45^\circ & 135^\circ & 90^\circ \end{pmatrix} \quad (1)$$

And the reference configuration is chosen as in

$$q_d = \begin{pmatrix} 0^\circ & 45^\circ & 45^\circ & 90^\circ & 0^\circ \end{pmatrix} \quad (2)$$

For 7-Dof robot arm, initial configuration is arranged as in

$$q_i = \begin{pmatrix} 0^\circ & -45^\circ & 0^\circ & -90^\circ & 0^\circ & 90^\circ & 0^\circ \end{pmatrix} \quad (3)$$

And the reference configuration is chosen as in

$$q_d = \begin{pmatrix} 45^\circ & 45^\circ & 45^\circ & 45^\circ & 90^\circ & 45^\circ & 0^\circ \end{pmatrix} \quad (4)$$

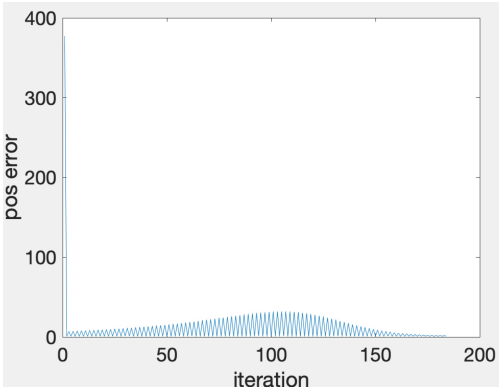
Joint limit of the 7-Dof robot arm in radian, which is shown in Tab.I , can be further taken into consideration for CCD, while the redundancy of the 7-Dof robot arm can be used to make the joints approach specified angles for jacobian inverse and jacobian transpose. In this experiment, joint 3 is set to -45° for this purpose.

TABLE I
JOINT LIMIT SETTING FOR 7-DOF

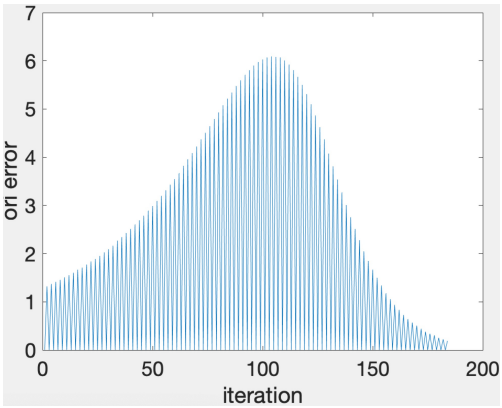
joint limit	joint 1	joint 2	joint 3	joint 4	joint 5	joint 6	joint 7
q_{max}	2.897	1.762	2.897	−0.069	2.897	3.752	2.897
q_{min}	−2.897	−1.762	−2.897	−3.071	−2.897	−0.017	−2.897

B. Experimental Result of CCD for 5-Dof and 7-Dof

As we can see from the results, final position error is smaller than 0.1% while final orientation error is smaller than 0.2 degree. The positon and orientation error diagrams of 5-Dof and 7-Dof robot arms are given in Fig. 8 and Fig.9. From the position error and orientation error diagrams, it is obvious that the error change follows a sawtooth-shaped broken line, which is consistent to the trade-off mentioned before.



(a) error of position



(b) error of orientation

Fig. 8. position and orientation error of 5-Dof robot arm

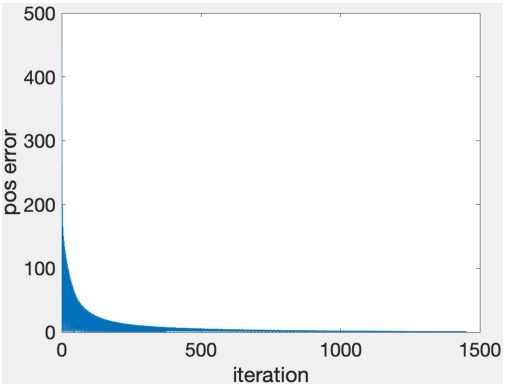
If joint limits are taken into consideration, the number of iterations increases slightly, as shown in Fig.10.

C. Experimental Result of Jacobian Inverse for 5-Dof and 7-Dof

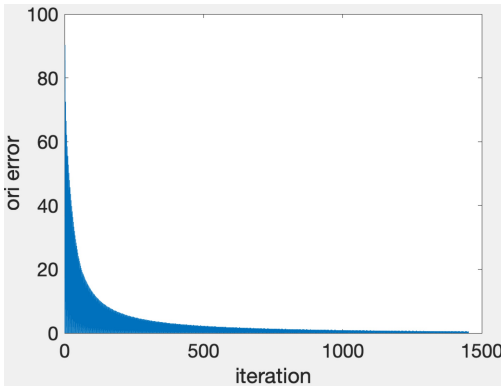
1) Result of Jacobian Inverse for 5-Dof: The Algorithm of Jacobian Inverse is used to get the soluion of all joint angles when the control error converges to zero. The intial and final gesture of the robot are shown in Fig. 11.

The error of orientation and position converges rapidly with the growth of iteration time. It costs around 80 times for the errors to converge to less than the preset value. The result is shown in Fig. 12.

2) Result of Jacobian Inverse for 7-Dof: The Algorithm of Jacobian Inverse is used to get the soluion of all joint angles when the control error converges to zero. The intial and final gesture of the robot are shown in Fig.13.



(a) position error



(b) orientation error

Fig. 9. position and orientation error of 7-Dof robot arm

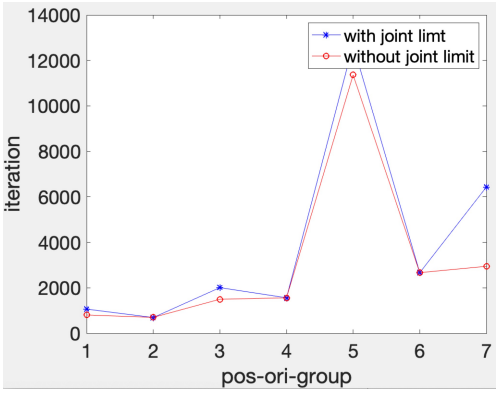


Fig. 10. effect of joint limits

The error of orientation and position converges rapidly with the growth of iteration time. It costs around 70 times for the errors to converge to less than the preset value. The result is shown in Fig. 14.

3) A slight difference between 5-DOF and 7-DOF: Compared with normal 6-DOF robot, 7-DOF robot has redundancy, the null space technique can be used to fix one of the solution of the joint angle to the desired value. Actually, the use of the null space will not influence the final posture of the end effector. The process is shown in Fig. 15. The green block in Fig. 15 is the part of null space. Modifying the parameters in the nullspace will lead to an

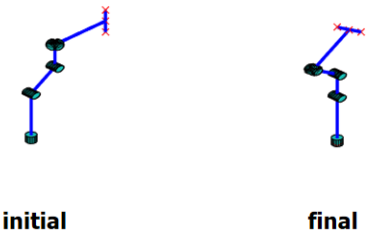


Fig. 11. Initial and final posture of the 5-DOF robot

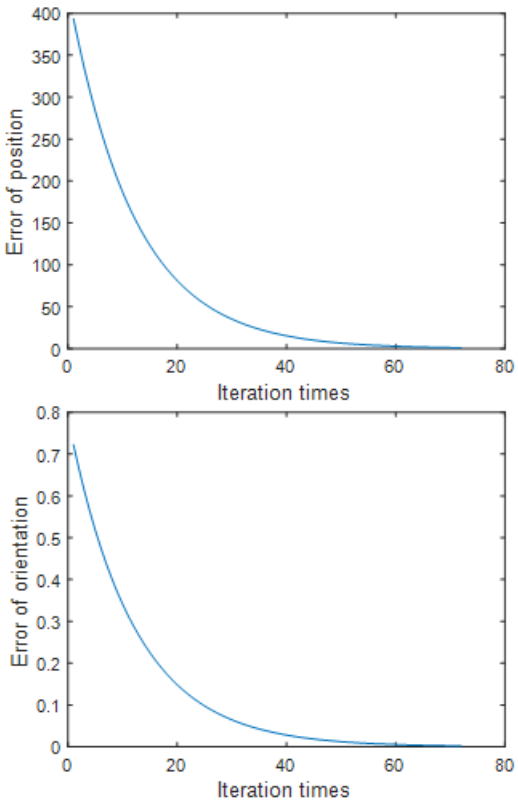


Fig. 12. The error of orientation and position

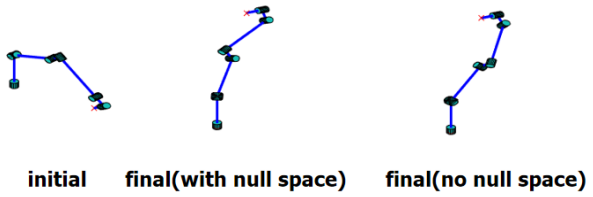


Fig. 13. iteration loop of the Jocobian inverse method

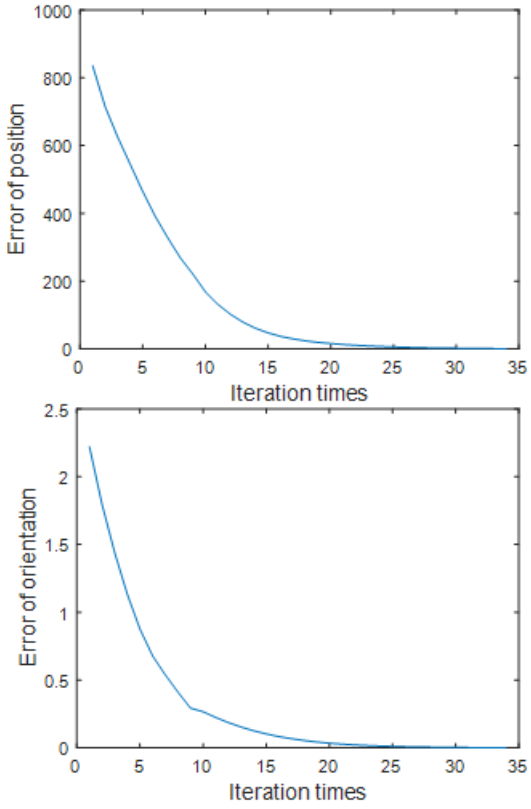


Fig. 14. The error of orientation and position

infinite number of solutions. In fact, the contribution of η is to generate null space motions of the structure that do not alter the task space configurations but allow the manipulator to reach postures which are more dexterous for the execution of the given task.[5] It is actually the secondary task for the robot to finish. The main task is always to put the effector in the correct position and orientation.

two graphs below in Fig.16 show the difference between the situation with null space and without null space. Here gives the functions of η and w.

$$\eta = K_{ns} \times \frac{\partial w}{\partial q}$$

$$w = |q_3 - q_{3d}|$$

The variation of q3 with or without null space are shown in Fig. 16. This two figures show that the q3 will try to converge to a preset value if using the null space.

D. Experimental Result of Jacobic Transpose for 5-Dof and 7-Dof

a) Result of 5-DOF simulation: It costs around 400 times of iteration to converge to the constrain of error with the same initial and desired posture. The diagram in Fig. 17 with red line is the variation of position error during the iteration, and the blue one is the orientation error. It shows that the the error decreases very fast at beginning but it slows down when the error becomes small.

b) Result of 7-DOF simulation: It costs 4000 times of iteration to converge. In the same case, the red line is position error and blue line is for orientation in Fig. 18. And the same phenomenon happened. The error converges faster at beginning and slower when the error is small.

c) Application of redundancy of 7-DOF: As mentioned, when the robot have 7 degree of freedom, the redundancy of freedom can be applied to do some application because there are only 6 DOF in the real world. The formula below

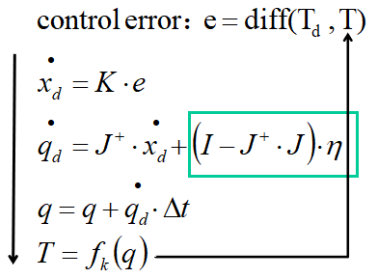


Fig. 15. Initial and final posture of the 7-DOF robot

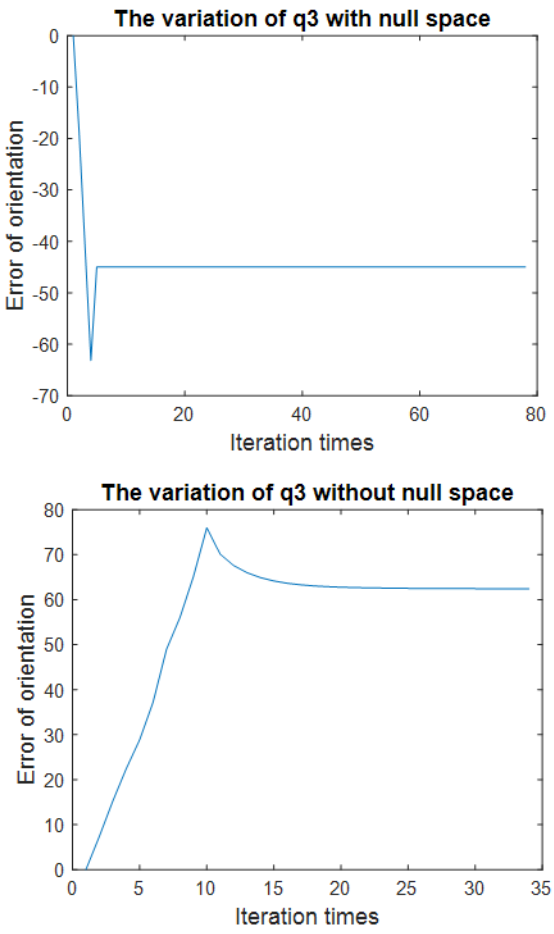


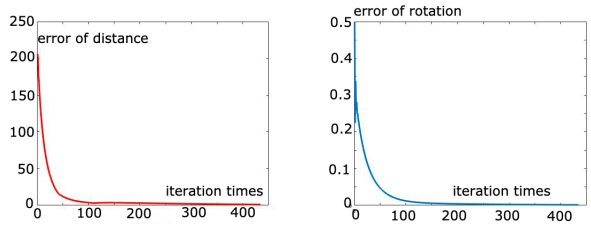
Fig. 16. comparison between using null space and without null space

shows that the method of manipulating the redundancy.[8] [10]

$$\dot{q} = J^{\dagger} e + (I_n - J^{\dagger} J)\dot{q}_0$$

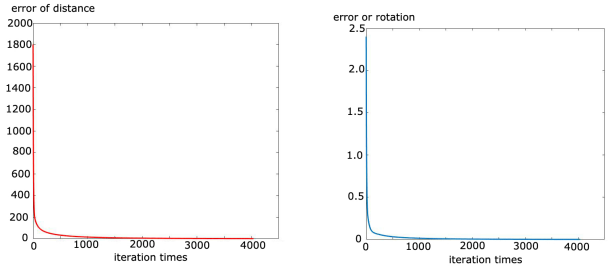
The first term of the right hand side is the normal part of Jacobian transpose method. and the second term projects $\dot{q}_0$ into the null space of the Jacobian matrix. which means that the the robot makes the end effector converges to zero error and manipulates some joints during the converging iteration.

In the example, $\dot{q}_0$ is set as the displacement between current degree of joint 3 and a target degree of joint. and the target of q_3 is set as $-\pi/4$. It means that while the robot



(a) Position error (b) Orientation error

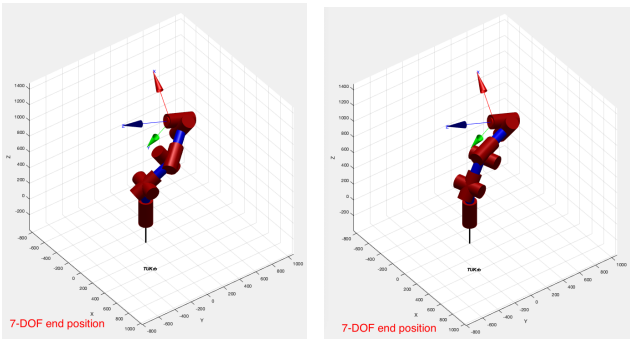
Fig. 17. Variation of error



(a) Position error (b) Orientation error

Fig. 18. Variation of error

arm is approaching the target point, the q_3 will move toward the target value.



(a) q_3 = free (b) $q_3 = -\pi/4$

Fig. 19. End posture compare between redudancy application and without application

The left picture of Fig. 19 is the end position without redundancy resolution, the result of value of q_3 close to 0.45π . The right one is the result with setting $-\pi/4$ as the target of q_3, the result of q_3 close to $-\pi/10$. The effect is obvious. q_3 was moving toward the set value. The reason for the difference of q_3 and $-\pi/4$ is that the error of end effector is settled as the finish condition, then no matter what q_3 is, the iteration is finished once the end effector get the point.

E. Experimental Result of Analytic Solution for 5-Dof and 7-Dof

First, the position and orientation of the end effector are known, using the inverse kinematic the position of joint 4 can be easily found. The lateral view of the robot shows that the position of joint 2 is also fixed. Then the triangle

relationship of joint 2, joint 3 and joint 4 are found. Actually there are two groups of solution. From the vertical view of the robot arm, the solution of joint 1 is clearly known.

Using the analytical method, two different groups of solution are got. they are shown in Fig. 20 and Fig. 21.

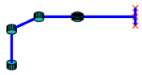


Fig. 20. first solution using analytical method

Fig. 21. Second solution using analytical method

IV. CONCLUSION

This paper compares three different numerical method to solve the inverse kinematic problem. There are advantages and disadvantages in every methods. User can choose an appropriate method according to different situations.

As for singularity, Jacobian Inverse has a problem when the target point is the singularity point. CCD is the only one method that can deal with joint limit requirement. But it is the only one which can not apply redundancy resolution or adjust convergence speed. Jacobian inverse and transpose can adjust speed by changing the gain of the input error. Jacobian inverse gets the highest speed of convergence, which means that it always needs the least iteration times. As using matlab to simulate these methods, CCD is more complicated than the others. The details are shown in table II.

TABLE II

METHODS COMPARISON TABLE

	Cyclic Coordinate Descent	Jacobian Inverse	Jacobian Transpose
Singularity-robustness	OK	NO	OK
Joint limit consideration	OK	NO	NO
Redundancy application	NO	OK	OK
Adjustable convergence speed	NO	OK	OK
Convergence speed(5 DOF)	-	++	+
Convergence speed(7 DOF)	-	++	+
Ease of implementation	-	+	+

REFERENCES

[1] Buss S R. Introduction to inverse kinematics with jacobian transpose, pseudoinverse and damped least squares methods[J]. IEEE Journal of Robotics and Automation, 2004, 17(1-19): 16.

[2] Kenwright B. Inverse kinematicscyclic coordinate descent (CCD)[J]. Journal of Graphics Tools, 2012, 16(4): 177-217.

[3] D. L. Pieper, "The kinematics of manipulators under computer control", Stanford Artificial Intelligence Report, 1968.

[4] J. Denavit, R. S. Hartenberg, "A kinematic notation for lower-pair mechanisms based on matrices", ASME Journal of Applied Mechanics, 23:215-221, 1955.

[5] Jingguo Wang, Yangmin Li and Xinhua Zhao, "Inverse Kinematics and Control of a 7DOF Redundant Manipulator Based on the ClosedLoop Algorithm", International Journal of Advanced Robotic Systems, Vol. 7, No. 4 (2010), pp. 110, ISSN 17298806.

[6] Sun, J., Cao, G., Li, W., Liang, Y. and Huang, S. (2017). Analytical inverse kinematic solution using the D-H method for a 6-DOF robot. In: 2017 14th International Conference on Ubiquitous Robots and Ambient Intelligence (URAI). IEEE, pp.714-716.

[7] Schreiber G., Hirzinger G. (1998) Singularity Consistent Inverse Kinematics by Enhancing the Jacobian Transpose. In: Lenari J., Husty M.L. (eds) Advances in Robot Kinematics: Analysis and Control. Springer, Dordrecht

[8] G. Antonelli, S. Chiaverini and G. Fusco (2000) Kinematic Control of Redundant Manipulators with On-Line End-Effector Path Tracking Capability Under Velocity and Acceleration Constraints

[9] W. A. Wolovich and H. Elliott, "A computational technique for inverse kinematics Las Vegas, Nevada, USA, 1984, pp. 1359-1363.

[10] Siciliano, B. J Intell Robot Syst (1990) 3: 201. https://doi.org/10.1007/BF00126069