Sensors & Transducers, Vol. 164, Issue 2, February 2014, pp. 295-301

www.sensorsportal.com

Sensors & Transducers

http://www.sensorsportal.com

Research on Inverse Kinematics Program Optimization of 6R Decoupled Robot

Daode ZHANG, Yuanzhong LI
School of Mechanical Engineering, Hubei University of Technology, 430068, China
E-mail: hgzdd@126.com, yuanzhongsf@163.com

Received: 20 November 2013 /Accepted: 28 January 2014 /Published: 28 February 2014

Abstract: According to complex analytic formula for the six degrees of freedom decoupled robot, a detailed analysis of the six degrees of freedom decoupled robot analytic formula of export process, as well the causes of multiple solutions. The method of increasing the local variables to avoid processor running the same statement repeatedly is proposed. The method to find the most frequency formula appeared in analytic solution replaced with local variables facilitate the use of loop to reduce the amount of code. It effectively reduces the computation time, optimize the computing process. Finally, taking PUMA560-like robot as an example, the calculation result is verified and simulated in Robotics Toolbox of MATLAB. *Copyright © 2014 IFSA Publishing, S. L.*

Keywords: Six degrees of freedom robot, Link parameters, Analytic solution, Program optimization, Verification.

1. Introduction

With the popularity of automation equipment in automated production, automated production continue to increase the degree of automation requirements, industrial robots, especially the 6 DOF revolute joint robots with its greater freedom and lower redundancy design have been applied widely. For robotic research, a basic problem to be solved is: to plan the robot motion process according to the tasks specified by users. This part of work will be done by the robot trajectory planning. Given the position and posture of the end of the robot arm actuator, the processor must calculate the joint variables of the various joints of the robot arm. The calculation of inverse kinematics is a specific problem which needs to be solved by inverse manipulator kinematics. Specific method of transformation from Cartesian space to the robot joint space for robot is achieved by the inverse kinematics solution.

Inverse kinematics is the mapping of the position and orientation of the robot from cartesian space to the joint space of the robot. For the six degrees of freedom mechanical arm, the majority may not be solved analytically. The robot could have analytic solutions only when it has some special structure form. These robots for which an analytic or closed form solution exists are characterized by several intersecting joint axes, and/or many α_i equal to 0 or ± 90 degrees [1]. Many general-purpose robots have six revolute joints currently. A sufficient condition that a manipulator with six revolute joints will have a closed form solution is that three neighboring joint axes intersect at a point [1]. Inverse kinematics solutions of the robots with this configuration are often still exist in a variety of possibilities normally. They should be carried out step by step in the process of solving solution under the circumstances. The final result can be obtained by optimization of the inverse solution through some predetermined rules. For example, the most typical case is that the calculated

theoretical joint displacement exceeds a range set by the actual joint variables.

2. Kinematics Model of the Robot

Link parameters of the manipulator are established with D-H method according to the method brought forward by John J. Craig. The method is that frame {i} is attached to link i and has its origin lying on joint axis i. Link parameters are defined according to the following Fig. 1.

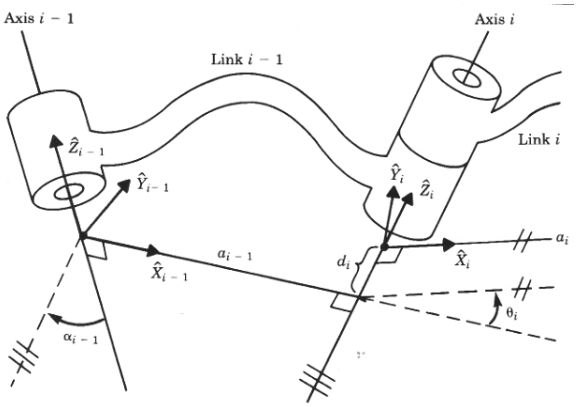

Fig. 1. Sketch Map of Coordinate System [1].

where a_i is the distance from $\hat{Z}_i$ to $\hat{Z}_{i+1}$ measured along $\hat{X}_i$. α_i is the angle between $\hat{Z}_i$ and $\hat{Z}_{i+1}$ measured about $\hat{X}_i$. d_i is the distance from $\hat{X}_{i-1}$ to $\hat{X}_i$ measured along $\hat{Z}_i$. θ_i is the angle between $\hat{X}_{i-1}$ and $\hat{X}_i$ measured about $\hat{Z}_i$.

It should be noted that a_i is the length of the link i, and therefore its value can be positive only. The signs of α_i, d_i and θ_i are determined by the actual situation. The choice of the positive direction of the axis is different, the link coordinate system established according to the above method has the very big difference for the different position of the origin of every coordinate system.

The establishment of coordinate system based on the above method is feasible. Take the robot whose configuration is similar to the PUMA560 as an example; construct a kinematics model of the robot [2]. The D-H parameter model of the 6R robot is shown in Table 1.

Let ${}^{i-1}_{i}A$ be a 4*4 homogeneous transform describing the frame {*i*-1} relative to the frame {*i*}, then:

$$ {}^{i-1}_{i}A = Rot(z,\alpha_{i-1})Trans(x,a_{i-1})Rot(z,\theta_i)Trans(z,d_i) \quad (1)$$

Table 1. Parameters of the robot link.

i	$\alpha_{i-1}(^\circ)$	$a_{i-1}(mm)$	$d_i(mm)$	$\theta_i(rad)$
1	0	0	0	θ_1
2	-90	0	0	θ_2
3	0	a_2	d_3	θ_3
4	-90	a_3	d_4	θ_4
5	90	0	0	θ_5
6	-90	0	0	θ_6

The $Rot(z,\alpha_{i-1})$, $Rot(z,\theta_i)$ are Rotation matrix among them, the $Trans(x,a_{i-1})$, $Trans(z,d_i)$ return a homogeneous transformation representing a translation. Rest of the parameters, such as $\alpha_{i-1}, a_{i-1}, \theta_i, d_i$ are determined by the robot configuration. It can be concluded that homogeneous transformation matrix between two adjacent links is:

$$ {}^{i-1}_{i}A = \begin{bmatrix} c\theta_i & -s\theta_i & 0 & a_{i-1} \\ s\theta_i c\alpha_{i-1} & c\theta_i c\alpha_{i-1} & -s\alpha_{i-1} & -s\alpha_{i-1}d_i \\ s\theta_i s\alpha_{i-1} & c\theta_i s\alpha_{i-1} & c\alpha_{i-1} & c\alpha_{i-1}d_i \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad (2)$$

where are the $\sin\theta_i$, $\cos\theta_i$, abbreviated as $s\theta_i$, $c\theta_i$ in order to facilitate writing, i= 1,2, ..., 6. There are 24 D-H parameters in all for the 6 DOF robot. The 6 joints of the robot are all revolute joints, Therefore the six revolute joint displacements are variables, the other D-H parameters are all known. The matrix ${}^{0}_{6}A$ which is obtained by repeated application of (2) is a position and pose matrix representing the last link's coordinate frame {6} with respect to the first link's coordinate frame {1} as follow:

$$ {}^{0}_{6}A = {}^{0}_{1}A(\theta_1){}^{1}_{2}A(\theta_2){}^{2}_{3}A(\theta_3){}^{3}_{4}A(\theta_4){}^{4}_{5}A(\theta_5){}^{5}_{6}A(\theta_6) \quad (3)$$

3. Solving Inverse Kinematics

${}^{0}_{6}A$ is the position and pose matrix which is specified beforehand by every single task in this inverse kinematics solving method, that is, the ${}^{0}_{6}A$ is known given:

$$ {}^{0}_{6}A = \begin{bmatrix} r_{11} & r_{12} & r_{13} & p_x \\ r_{21} & r_{22} & r_{23} & p_y \\ r_{31} & r_{32} & r_{33} & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad (4)$$

The submatrix $\begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix}$ represents the last link's coordinate frame's pose with respect to the first link's coordinate system. The submatrix $\begin{bmatrix} p_x \\ p_y \\ p_z \end{bmatrix}$ represents the position of the origin of the last link's coordinate system with respect to the first link's coordinate system. Segregation variable method is a routine procedure applied to solve the inverse kinematics [3]. The part which is dependent on θ_1 of the formula (3) is moved to the left side of the equation, then:

$$\left[{}^0_1A(\theta_1)\right]^1_6A={}^1_2A(\theta_2){}^2_3A(\theta_3){}^3_4A(\theta_4){}^4_5A(\theta_5){}^5_6A(\theta_6) \quad (5)$$

Equating the (2,4) elements from both sides of it after simplification of the formula (5), it can be concluded that:

$$\theta_1=a\tan 2(p_y,p_x)-a\tan 2(d_3,\pm\sqrt{p_x^2+p_y^2-d_3^2}) \quad (6)$$

Let

$$K=\frac{p_x^2+p_y^2+p_z^2-a_2^2-a_3^2-d_3^2-d_4^2}{2*a_2} \quad (7)$$

Then

$$\theta_3=a\tan2(a_3,d_4)-a\tan2(K,\pm\sqrt{a_3^2+d_4^2-K^2}) \quad (8)$$

Likewise, putting the part of formula (3) which is dependent on θ_1, θ_2 and θ_3 on the left-hand side of the equation, that is,

$$[{}^0_1A(\theta_1){}^1_2A(\theta_2){}^2_3A(\theta_3)]^{-1}={}^3_4A(\theta_4){}^4_5A(\theta_5){}^5_6A(\theta_6) \quad (9)$$

Equating the (1,4) elements from both sides of it, as well as the (2,4) elements after simplification of the formula (8), it can be concluded that:

$$\theta_2+\theta_3=a\tan 2[(a_2s_3-d_4)(c_1p_x+s_1p_y)-(a_2c_3+a_3)p_z, (a_3+a_2c_3)(c_1p_x+s_1p_y)+(a_2s_3-d_4)p_z] \quad (10)$$

The rest of the angle θ_4, θ_5 and θ_6 can be calculated in the same manner. Detailed calculation process, see references [1]. There is a typical inverse kinematics result which is included to illustrate the optimization of calculating program.

$$\theta_4=a\tan2(r_{23}c_1-r_{13}s_1,r_{33}s_{23}-r_{13}c_1c_{23}-r_{23}s_1c_{23}) \quad (11)$$

$$s_5=r_{33}(s_{23}c_4)+r_{23}(c_1s_4-s_1c_{23}c_4)-r_{13}(c_1c_{23}c_4+s_1s_4) \quad (12)$$

$$c_5=r_{13}(-c_1s_{23})+r_{23}(-s_1s_{23})+r_{33}(-c_{23}) \quad (13)$$

$$\theta_5=a\tan 2(s_5,c_5) \quad (14)$$

$$s_6=r_{11}(s_1c_4-c_1c_{23}s_4)+r_{31}(s_{23}s_4)-r_{21}(s_1c_{23}s_4+c_1c_4) \quad (15)$$

$$c_6=r_{11}[(c_1c_{23}c_4+s_1s_4)c_5-c_1s_{23}s_5]+r_{21}[(s_1c_{23}c_4-c_1s_4)c_5-s_1s_{23}s_5]-r_{31}(s_{23}c_4c_5+c_{23}s_5), \quad (16)$$

$$\theta_6=a\tan 2(s_6,c_6), \quad (17)$$

It should be pointed out that the formula (6), (8) can be respectively calculated from two different angles θ_1, θ'_1, θ_3 and θ'_3. They are combined with each other, there may be four different results [4]. The formula (10) also has four different values. It can be concluded from the formula (10) that joint angle 2 has four different values θ_2, θ'_2, θ''_2, and θ'''_2. The other four sets of solutions can be obtained by "flipping" the wrist of the manipulator after obtained four sets of solutions such as $\theta_4(\theta'_4,\theta''_4,\theta'''_4)$, $\theta_5(\theta'_5,\theta''_5,\theta'''_5)$ and $\theta_6(\theta'_6,\theta''_6,\theta'''_6)$. There are a total of eight sets of solutions. Eight sets of solutions generated as shown in the following Table 2.

4. Optimization of C Program Design in Analytical Solution

Some production have very detailed requirements for the path of the robot end-effector and the instantaneous speed of post in industrial automation production, for example, continuous welding operation, a laser cutting operation etc. Most of the trajectory of the robot's end-effector is linear, circular, and special curve specified by users on these conditions. It need the robot's end-effector to move in accordance to the continuous trajectory which is specified by users beforehand, that is, the tasks need to be completed in the Cartesian space.

An algorithm of trajectory for Cartesian space is that: the continuous trajectory in Cartesian space is divided into a number of discrete path points. There is very big difference between the actual motion trajectory of robot and the trajectory which is specified by users if the distance between two adjacent path points is large in these discrete points. A continuous path is separated into many dense path points because of the interval of the adjacent path points should not be too large [5]. It is necessary for controller to solve inverse kinematics for every path points. It is also very time-consuming for controller to do a lot of multiplication and division, square root, calculate transcendental functions because there are many dense path points.

Many intelligent robots are using online programming. It requires that cup should update trajectory in real time. It demands that the time which is spent in calculating the solution of inverse Kinematics should be as short as possible. How to write correct analytical solution program of inverse kinematics and reduce computation time become a very important work. The C program can reduce the number of calculations and effectively reduce the amount of computation through the method of increasing the intermediate variables according to the analysis of the formula above. It is necessary for intelligent robot to achieve the inverse solution on the embedded processor sometimes; using C program [6]. The method is now described in detail as follows:

4.1. Repeat Formula

The program needs to do the same operation on the same formula of analytic solution repeatedly. For example, $p_x^2+p_y^2-d_3^2$, $a_3^2+d_4^2$ appear twice in the formula (6), (7) and (8). Formula (7) can be decomposed into the formula (18):

$$K=\frac{(p_x^2+p_y^2-d_3^2)-(a_3^2+d_4^2)+p_z^2-a_2^2}{2*a_2} \quad (18)$$

Both formula (6) and formula (18) have $p_x^2+p_y^2-d_3^2$, Both formula(8) and formula (18) has $a_3^2+d_4^2$. The program calculates the value of $p_x^2+p_y^2-d_3^2$, and set it equal to temp0, that is , $temp0=p_x^2+p_y^2-d_3^2$. The detailed calculation process is as follows:

$$temp\ 0=p_x^2+p_y^2-d_3^2 \quad (19)$$

$$temp\ 1=a_3^2+d_4^2 \quad (20)$$

$$temp2=\sqrt{temp0} \quad (21)$$

$$temp\ 3=a\tan 2(p_y,p_x) \quad (22)$$

$$\theta_1=temp3-a\tan 2(d_3,temp2) \quad (23)$$

$$\theta'_1=temp3-a\tan 2(d_3,-temp2) \quad (24)$$

The formula (6) can be rewritten like this:

$$K=\frac{temp\ 0-temp\ 1+p_z^2-a_2^2}{2*a_2}, \quad (25)$$

The joint angle 3 also can be calculated in this manner.

Table 2. Eight groups of inverse solution diagram.

1	θ_1	θ_2	θ_3	θ_4	θ_5	θ_6
2	θ_1	θ_2	θ_3	θ_4+180	$-\theta_5$	θ_6+180°
3	θ_1	θ'_2	θ'_3	θ'_4	θ'_5	θ'_6
4	θ_1	θ'_2	θ'_3	θ'_4+18	$-\theta'_5$	θ'_6+180°
5	θ'_1	θ''_2	θ_3	θ''_4	θ''_5	θ''_6
6	θ'_1	θ''_2	θ_3	θ''_4+180	$-\theta''_5$	θ''_6+180°
7	θ'_1	θ'''_2	θ'_3	θ'''_4	θ'''_5	θ'''_6
8	θ'_1	θ'''_2	θ'_3	θ'''_4+180	$-\theta'''_5$	θ'''_6+180°

4.2. Combinations of Different Angles

The same trigonometric is being repeated called in analytic solution. For example, $a_3+a_2c_3$, $c_1p_x+s_1p_y$ and $a_2s_3-d_4$ are repeatedly used. Moreover, the same formula will produce a plurality of different calculation results, because there are four different combinations of the joint angle 1 and 3. Aimed at the complex relationship among formulas, an effective method is presented [7]. It is convenient for CPU to calculate the formulas and call the trigonometric functions. The program can define two one-dimensional array such as k[2], t[2] for the formula which contains only one joint angle, for example , $a_3+a_2c_3$, $c_1p_x+s_1p_y$. The array is convenient for program to be calculated and called as follows:

$$k[0]=p_x\sin\theta_1+p_y\cos\theta_1 \quad (26)$$

$$k[1]=p_x\sin\theta'_1+p_y\cos\theta'_1 \quad (27)$$

$$t[0]=a_3+a_2\sin\theta_3 \quad (28)$$

$$t[1]=a_3+a_2\sin\theta'_3 \quad (29)$$

There is a formula (10). It consists of two different joint angles such as joint angle 1 and joint angle 3. The value of formula (10) is dependent on joint angle 1 and joint angle 3. There are four different values of formula (9) because of four different combinations of joint angle 1 θ_1, θ'_1 and joint angle 3 θ_3, θ'_3. The program can define two two-dimensional array which facilitates the use of iteration statement such as $s_{23}[2][2]$ and $c_{23}[2][2]$. The method is described in detail taking the $s_{23}[2][2]$ as an example in the following Table 3.

Table 3. Representation of the array.

$S_{23}[0][0]$	It is represented as $\sin(\theta_2+\theta_3)$ which is calculated by θ_1 and θ_3
$S_{23}[0][1]$	It is represented as $\sin(\theta_2+\theta'_3)$ which is calculated by θ_1 and θ'_3
$S_{23}[1][0]$	It is represented as $\sin(\theta'_2+\theta_3)$ which is calculated by θ'_1 and θ_3
$S_{23}[1][1]$	It is represented as $\sin(\theta'_2+\theta'_3)$ which is calculated by θ'_1 and θ'_3

There are some formulas consisting of several different trigonometric functions sometimes. The formulas are so complex that designing inverse kinematics programs is not easy. It is not very difficult to write the inverse kinematics program after careful analysis of analytical solution which is provided by [1], known the produced sources of multiple values during the solving process of joint angle. θ_2 is derived from θ_1 and θ_3, θ_4 is derived from θ_1, θ_2 and θ_3 so It can be conclude that θ_4 is derived from θ_1 and θ_3 namely. It can be known from (12), (13), (14), (15), (16) and (17) that the joint angle 5 is only concerned with the θ_1 and θ_3, so is joint angle 6. A detailed description of the calculation process of the four different values is introduced taking the c_1s_{23} as an example. The program define a one-dimensional array $c_1[2]$, a two-dimensional array w[2][2] and two variables i, j. Let $c_1[0]$ equal to $\cos\theta_1$, and let $c_1[1]$ equal to $\cos\theta'_1$ w[2][2] is used to store four different values, then,

```
for(i=0;i<2;i++)
  for(j=0;j<2;j++)
    {
     W[i][j]=c1[i]*S23[i][j];
    };
```

There are many formulas similar to the c_1s_{23} in the inverse solution which is provided in [1], they are listed as follows: $c_1c_{23}, s_1c_{23}, s_1s_{23}, s_1s_4, c_1s_4, c_4s_{23}$.

5. Verification and Simulation

Given a set of link parameters of the Puma560 like robots in order to test the program about 6 DOF robot is correct or not. Let $a_2=431.8\,mm$, $a_3=20.32\,mm$, $d_3=124.46\,mm$, $d_4=431.8\,mm$. Assuming that the range of each joint variables varies from -2pi to 2pi, given a group of angles in the rotation range of robot's joints:

$\theta_{fkine}=[1.7841,-1.6290,1.3575,-1.2023,1.3575,-2.8676]$

the angle parameter is in radians. build the robot's kinematics model using the 9.7 version of the robotics toolbox of Matlab [8]. Calculated the position and posture matrix of robot end-effector by the function named ikine which is provided by the toolbox, the matrix T as follows:

$$T=\begin{bmatrix} 0.21096 & -0.22668 & 0.95084 & -14497947 \\ -0.53165 & -0.84228 & -0.08298 & 81.41058 \\ 0.82026 & -0.48801 & -0.29834 & 20.53511 \\ 0 & 0 & 0 & 0 \end{bmatrix} \quad (30)$$

Using VS2008 to achieve the previously proposed algorithm about analytical solution, optimizing the processing program according to the method above, using the position and posture matrix T to initialize the C program, there are eight groups of inverse solution after this program finish running. The results are rounded off to five significant figures according to the following Table 5.

The first group of the inverse solution is consistent with the joint variables matrix θ_{fkine} given previously, θ_{fkine} must be one of the possible solutions which are calculated by the program as long as the joint variables of the robot can be selected within a predetermined range. It confirms the accuracy of the analytical solutions and program from the side. More detailed error analysis needs to use the function named "IKINE" which is also provided by the Robotics Toolbox of MATLAB. High precision inverse solutions are calculated iteratively by the function according to the position and orientation matrix T [9]. The program needs further improvement if the error between the high precision inverse solution and the inverse solution listed above is big [10]. The simulation results of (1), (3), (5) and (7) are shown in the Fig. 2-5:

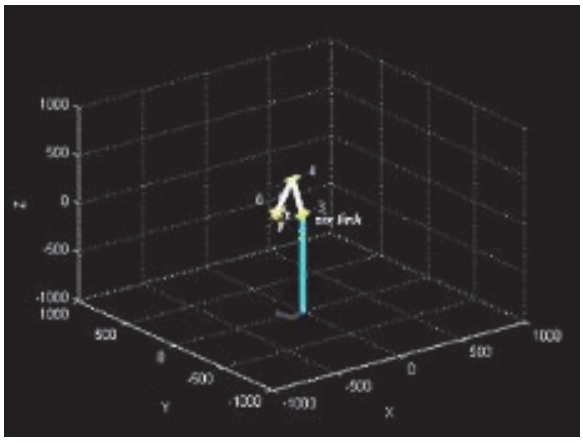

Fig. 2. The simulation result of inverse solution 1.

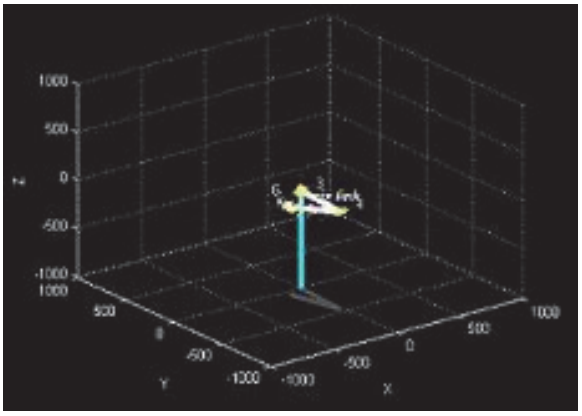

Fig. 3. The simulation result of inverse solution 3.

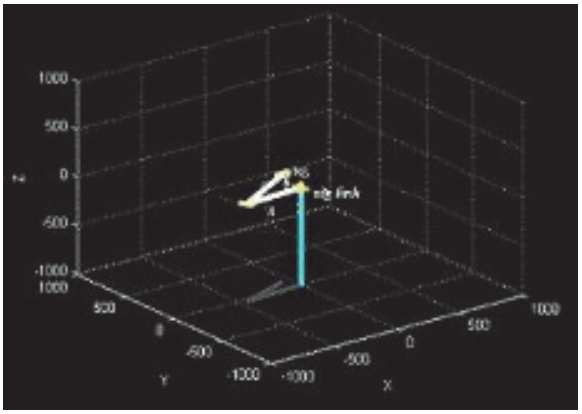

Fig. 4. The simulation result of inverse solution 5.

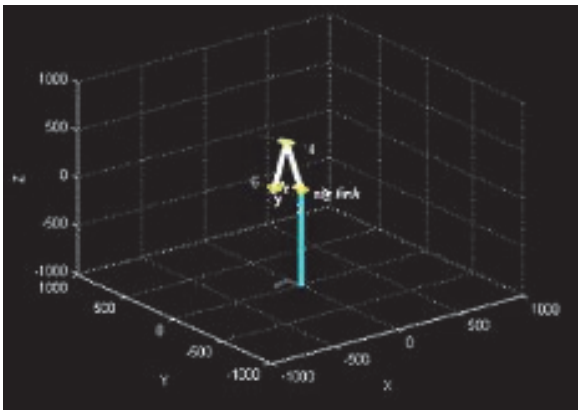

Fig. 5. The simulation result of inverse solution 7.

Table 5. Eight groups of inverse solution.

No.	Degree (rad)					
1	1.7841	-1.6290	1.3575	-1.2023	1.3575	-2.8676
2	1.7841	-1.6290	1.3575	1.9393	-1.3575	0.2740
3	1.7841	-4.8383	1.8781	-1.8077	1.9172	0.7167
4	1.7841	-4.83833	1.8781	1.3339	-1.9172	3.8583
5	0.3342	-4.4023	1.3575	-0.4109	1.7852	0.5899
6	0.3342	-4.4023	1.3575	2.7307	-1.7852	3.7315
7	0.3342	-1.5126	1.8781	-2.740	1.6035	-2.4089
8	0.3342	-1.5126	1.8781	0.4011	-1.6035	0.7327

6. Conclusions

This paper analyzes calculation process of analytical solution of the inverse kinematics of a 6 DOF robot. Aiming at the complex inverse solution of the 6 DOF robots. The method is presented to save the calculating time and optimize the process of computing [11]. It analyzes the kinematical modeling of the general Puma560 like robot in accordance with the D-H representation. Using VS2008 to achieve the proposed algorithm about analytical solution, Optimizing the program according to the method above [12]. It provides a reference for calculating the inverse solution of the 6 DOF robots using other programming languages. Finally, the calculation results are verified and the simulation result validates the validity of this method [13].

Acknowledgements

This work is supported by Natural Science Foundation of Hubei Province (No. 2012FFB00604) and National Natural Science Foundation of China (No. 51174084).

References

[1]. J. Craig, Introduction to robotics mechanics and control, *China Machine Press*, 2005.
[2]. P. I. Corke, A robotics toolbox for MATLAB, *IEEE Robotics and Automation Magazine*, Vol. 3, Issue 1, 1996, pp. 24-32.
[3]. Tan Ming, Xu De, Advanced robot control, *Higher Education Press*, 2007.
[4]. Zhenbiao Wu, Zhengjia Wang, Industrial robot, *Huazhong University of Science and Technology press*, 2006.
[5]. B. Niku, Introduction to robotics analysis, systems, applications, *Publishing House of Electronics Industry*, 2004.
[6]. Yannian Rui, Robot technology and its application, *Chemical Industry Press*, 2008.
[7]. F. S. Ren, S. J. Chen, X. Y. Guan, Special-purpose welding robot for intersection welding seam, *Transactions of the China Welding Institution*, Vol. 30, Issue 6, 2009, pp. 59-62.
[8]. Tianqi Wang, Liangyu Li, Jianfeng Yue, Inverse kinematics analysis of tubular joint welding robot, *China Welding*, Vol. 21, Issue 2, 2012, pp. 55-58.
[9]. Wang Wei, Yun Chao, Zhang Ling, Designing and optimization of an off-line programming system for robotic belt grinding process, *Chinese Journal of Mechanical Engineering*, Vol. 24, Issue 6, 2011, pp. 647-656.
[10]. V. I. Utkin, Variable structure systems with sliding modes, *IEEE Transaction on Automatic Control*, Vol. 22, Issue 2, 1977, pp. 212-222.
[11]. O. Kaynak, K. Erbatur, and M. Ertugrul, The fusion of computationally intelligent methodologies and sliding-mode control – A survey, *IEEE Transactions on Industrial Electronics*, Vol. 48, Issue 1, 2001, pp. 4-16.

[12]. P. I. Corke, Symbolic algebra for manipulator dynamics, *Commonwealths Scientific and Industrial Research Organization CSIRO*, Australia, 1996.
[13]. R. Steinvorth, H. Kaufman, Direct model reference adaptive control of robots, in *Proceedings of the Conference on Information Sciences and Systems CISS, 15-17 March 1991*, pp. 667-672.