General Robot Inverse Kinematics Solution Based on MPGA-RBFNN

Yi Zhang[a], Fangjun Liu[b]

Chongqing University of Posts and Telecommunications, Chongqing 400065, China

[a]380424865@qq.com, [b]lfjyx163@163.com

Abstract. In order to solve the problem of the inverse kinematics of general robot, such as slow speed in problem-solving and lower accuracy of the solution, a high precision MPGA-RBFNN algorithm is proposed, which introduce the Multiple Population Genetic Algorithm (MPGA) into Radial Basis Functions Neural Network (RBFNN). Combining with the positive kinematics model of general robots, the RBFNN with three-layers is used to solve the inverse kinematics of general robots and the MPGA is adopted to optimize the network structure and connection weights of RBFNN. By using the way of hybrid coding and simultaneous evolutionary, the non-linear mapping from the posture of the robot in the working space to the angle of the joint is realized, avoiding the complicated formula derivation and improving the speed of solving. Finally, the experimental are tested on the 6R robot, the results show that the MPGA-RBFNN algorithm is not only improves the speed of the solving, but also enhances the training success rate and the calculation accuracy.

1. Introduction

The approach to the inverse kinematics solution of robots is the process of calculating the angle value of each joint by knowing the position and orientation of the end effector [1]. Many traditional inverse kinematics solutions, such as the closed-form methods and numerical methods, but they are inadequate for general robots and the solution accuracy can not be guaranteed [2].

Genetic algorithm has the advantages of better global searching ability and stability [3]. Structures [4] used the GA improvement strategies to construct the optimization objective function in order to alleviate the computational difficulty. Particle swarm optimization is used to introduce into GA to solve the inverse kinematics problem efficiently on arbitrary joint chains by Starke [5]. Although the solution can be obtained by GA, the GA is not ideal for solving nonlinear problems and the accuracy of the solution cannot be guaranteed.

Radial Basis Functions Neural Network has the better ability of nonlinear fitting. Zubizarreta [6] have suggested to use the structured RBFNN, which can be trained quickly to calculate the real-time inverse kinematics problems of 3PRS robots. Raşit [7] presented a study based on neural network and GA for the inverse kinematics solution of a six-joint Stanford robotic manipulator to minimize the error at the end effector. Although RBFNN has achieved certain results in solving the inverse kinematics solution of robots, it still has the defects that the network structure is not perfect. Moreover, it is easy to trap in local optimum and increase the output error when use the GA to optimize the RBFNN [8].

In this paper, RBFNN is used to solve the inverse kinematics of general robots and the Multiple Population Genetic Algorithm (MPGA) is adopted to optimize the network structure and connection weights of RBFNN. By using the way of hybrid coding and simultaneous evolutionary, the non-linear mapping from the posture of the robot in the working space to the angle of the joint is realized, avoiding the complicated formula derivation and improving the speed of solving.

2. Kinematics analysis of general robot

To analyze the inverse kinematics problem, the modified Denavit-Hartenberg frame is also given in Fig.1 for robot models, D-H parameters of the Comau NJ-220 robot is given in Table1.The robot has an axial offset in the direction of Z_5 at the joint 5, it causes the three axis can not intersect in the robot end, which is described as the general robot.

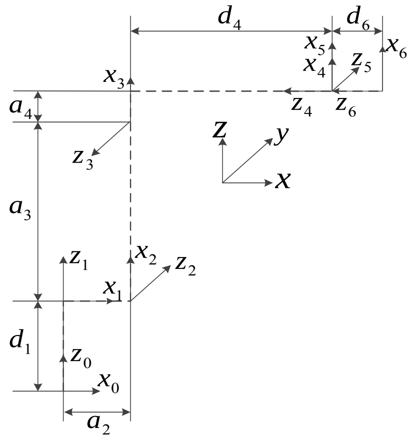


Fig.1 The D-H Model of the Comau NJ-220 Robot

Table 1. Joint parameters of Comau NJ-220 robot

Link i	Twist angle α_{i-1} / (°)	Length a_{i-1} / (mm)	Offset d_i / (mm)	Joint angle θ_i / (rad)
1	0	0	830	[-2.9,2.9]
2	-90	400	0	[-1.57,1.57]
3	180	1175	0	[-1.57,1.57]
4	-90	250	-1125	[-3.14,3.14]
5	-90	0	10	[-3.14,3.14]
6	90	0	-230	[-3.14,3.14]

According to the D-H parameters in Table 1, the homogeneous transformation matrix ${}^{i\text{-}1}_{i}T$ can be expressed as:

$$ {}^{i\text{-}1}_{i}T = Screw_X(a_{i\text{-}1},\alpha_{i\text{-}1})Screw_Z(d_i,\theta_i) = \begin{pmatrix} \cos\theta_i & \text{-}\sin\theta_i & 0 & a_{i\text{-}1} \\ \sin\theta_i\cos\alpha_{i\text{-}1} & \cos\theta_i\cos\alpha_{i\text{-}1} & \text{-}\sin\alpha_{i\text{-}1} & \text{-}d_i\sin\alpha_{i\text{-}1} \\ \sin\theta_i\sin\alpha_{i\text{-}1} & \cos\theta_i\sin\alpha_{i\text{-}1} & \cos\alpha_{i\text{-}1} & d_i\cos\alpha_{i\text{-}1} \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{1} $$

By successively multiplying the homogeneous transformation matrix, the transformation relationship between the end effector coordinate system and the robot base coordinate system can be obtained:

$$ {}_6^0T = \prod_{i=1}^{6} {}_i^{i-1}T = \begin{pmatrix} n_x & o_x & a_x & p_x \\ n_y & o_y & a_y & p_y \\ n_z & o_z & a_z & p_z \\ 0 & 0 & 0 & 1 \end{pmatrix} = p(\theta_1, \theta_2, \theta_3, \theta_4, \theta_5, \theta_6) \tag{2} $$

According to the formula (2), the posture $R(n_x, n_y, n_z, o_x, o_y, o_z, a_x, a_y, a_z)$ and position $P(p_x, p_y, p_z)$ of the end effector can be obtained, which is the positive kinematics solution of the robot. At the same time, the data set of the robot operating space can be obtained using the positive kinematics solution. According to the principle of coordinate transformation, the RPY Euler angles β, α, γ of the end effector posture R can be obtained:

$$ \beta = \arctan 2(-n_z, \sqrt{n_x^2 + n_y^2}) \tag{3} $$

$$ \alpha = \arctan 2(n_y, n_x) \tag{4} $$

$$ \gamma = \arctan 2(o_z, a_z) \tag{5} $$

In the process of inverse kinematics solution, the variables $(\beta, \alpha, \gamma, p_x, p_y, p_z)$ are considered as the input variables and the joint variable $(\theta_1, \theta_2, \theta_3, \theta_4, \theta_5, \theta_6)$ is deemed as the output variable.

3. MPGA-RBFNN implementation

There are some problems in the process of the inverse kinematics solution, such as slower speed and lower accuracy. In order to solve the problem, the MPGA-RBFNN algorithm is proposed. The process of the MPGA-RBFNN algorithm is shown in Fig.2.

In Fig.2, in order to meet the searching requirements, MPGA sets different control parameters according to the differences of populations, and the multiple populations are optimized at the same time. In order to achieve co-evolution between multiple populations, different populations are contacted by immigration operator in MPGA.

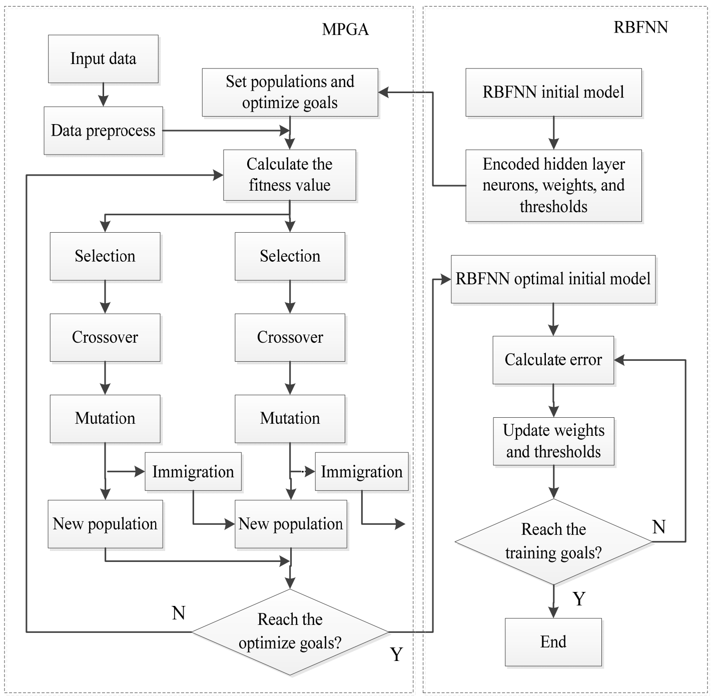

MPGA
RBFNN
Input data
Data preprocess
Set populations and optimize goals
Calculate the fitness value
Selection
Crossover
Mutation
Immigration
New population
Selection
Crossover
Mutation
Immigration
New population
Reach the optimize goals?
N
Y
RBFNN initial model
Encoded hidden layer neurons, weights, and thresholds
RBFNN optimal initial model
Calculate error
Update weights and thresholds
Reach the training goals?
N
Y
End

Fig.2 Algorithm flow chart of MPGA-RBFNN

4. RBFNN algorithm

In the network structure of this paper, the space posture $P=\{p_x,p_y,p_z,\beta,\alpha,\gamma\}$ of the end effector is taken as the six inputs of RBFNN, the output is the angle of each joint $(\theta_1,\theta_2,\theta_3,\theta_4,\theta_5,\theta_6)$, which is the solution of the inverse kinematics.

In the selection of hidden layers, the Gaussian function is selected as the basis function, the expression is given:

$$\phi(x,x_i)=G(x,x_i)=G(\|x-x_i\|)=\exp(-\frac{1}{2b_i^2}\|x-x_i\|) \tag{6}$$

In above equation, $x\in R$ is the N-dimensional input vector, $x_i\in R$ is the center vector of the i th hidden layer unit in RBFNN, b_i is the width of the i th hidden layer unit in RBFNN. The output of the i th hidden layer unit in RBFNN can be expressed as the linear weighted sum of n basis functions:

$$y_n(k)=\sum_{i=1}^{n}\omega_i\times\phi(x,x_i)=\sum_{i=1}^{n}\omega_i\times\exp(-\frac{1}{2b_i^2}\|x-x_i\|) \tag{7}$$

The network error function is defined at the time k :

$$e(k)=\sum_{i=1}^{n}\left[t(k)-y_n(k)\right] \tag{8}$$

Where $t(k)$ is the expected output of the network at the moment k .

4.1. Encoding scheme

Assume that the maximum number of hidden layer neurons is l, and the number of output neurons is n, the MPGA chromosome encoding is listed as followed:

$$c_1c_2\cdots c_l\omega_{11}\omega_{21}\cdots\omega_{l1}\omega_{12}\omega_{22}\cdots\omega_{l2}\omega_{1m}\omega_{2m}\cdots\omega_{lm}b_1b_2\cdots b_m \tag{9}$$

Where $c_1c_2\cdots c_l$ is the encoding scheme of the hidden layers, $b_1b_2\cdots b_m$ is the encoding scheme of the thresholds, $\omega_{11}\omega_{21}\cdots\omega_{l1}\omega_{12}\omega_{22}\cdots\omega_{l2}\omega_{1m}\omega_{2m}\cdots\omega_{lm}$ is the encoding scheme of the connection weights. These all adopt real number encoding, b_j is the threshold of the j th output layer neuron.

4.2. Fitness function

There exist the actual output error and expected output error in the neural network, in order to evaluate MPGA-RBFNN algorithm, the fitness function is introduced in this paper as:

$$F=\frac{1}{e(k)}=\frac{1}{\sum_{i=1}^{n}\left[t(k)-y_n(k)\right]} \tag{10}$$

Where $e(k)$ is the sum of squared error between the actual output and the expected output of the neural network. Generally speaking, the better the fitness, the smaller the error and vice versa.

4.3. Genetic Operators

In this paper, nonlinear ranking strategy based on roulette method is used to be the selection operator, the crossover operator adopts the way of combining multi-point crossings with uniform crossovers. The crossover probability is expressed as:

$$P_c(i)=P_{c\min}(i)\times(1-rand(N,1))+P_{c\max}(i)\times rand(N,1) \tag{11}$$

Where $P_{c\min}(i)$ and $P_{c\max}(i)$ are the minimum and maximum value of the cross probability of the i th population respectively.

The mutation operator based on dynamic mutation rate is used to be the mutation operator in this paper. The mutation probability commonly used is given as:

$$P_b(i)=P_{b\min}(i)\times(1-rand(N,1))+P_{b\max}(i)\times rand(N,1) \tag{12}$$

$$P_b(i+1)=\frac{P_c(i)}{3}\text{-}\ (\frac{GEN\text{-}gen}{GEN})\times(\frac{P_c(i)}{3}-P_b(i)) \tag{13}$$

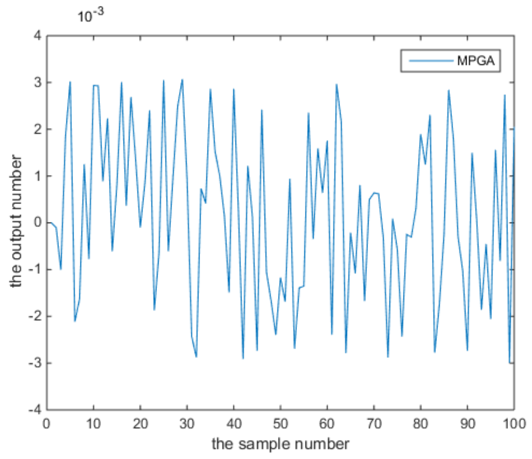
Where $P_{b\min}(i)$ and $P_{b\max}(i)$ are the minimum and maximum value of the mutation probability of the i th population respectively. $P_b(i+1)$ is the mutation probability of the $(i+1)$ th population, $P_c(i)$ is the cross probability of the i th population, gen is the iterations of the current algorithm and GEN is the total number of iterations.

5. Experimental results and analysis

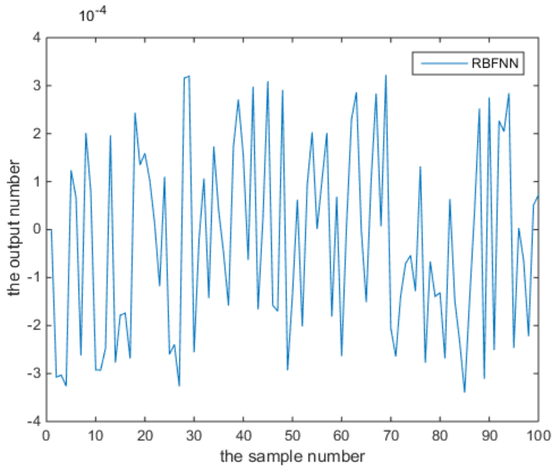
5.1. Accuracy analysis

Non-repetitive random sampling method is used to select 100 space posture and position from the training samples $\{P_x, P_y, P_z, \beta, \alpha, \gamma\}$ as the input samples, it is corresponding 100 groups joint angles. The training sample is normalized to be the input of the RBFNN, the output of the network is the joint angle θ_i, The RBFNN is trained and the error indicator is set as 10^{-5}. Taking the θ_2 as the example the output error graphs of MPGA, RBFNN and MPGA-RBFNN are shown in Fig.3 at the same training time. In order to further compare the differences among the three algorithms, 50 groups data of joint angle is randomly selected from the previous 100 groups data no-repeatly to be the test data for verification. The results of the three groups are shown in Table 2.

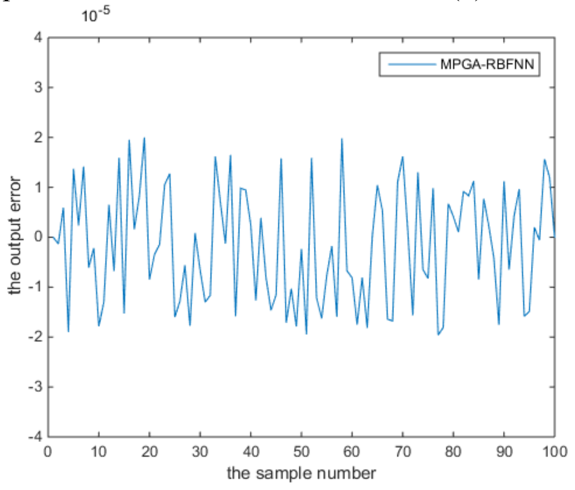
As can be seen from Fig.3, the changing process of the output error is shown by three algorithms in the inverse kinematics resolution of the robot. Although 3 algorithms get the output error, the order of magnitude of the output error maintain at the 10^{-5} level when MPGA-RBFNN is used to solve the inverse kinematics of the robot, comparing with MPGA and RBFNN, the performance of MPGA-RBFNN is better.



(a)MPGA output error



(b)RBFNN output error



(c)MPGA-RBFNN output error

Fig.3 The output error of three kinds of algorithm

Table 2 The results in solving inverse kinematics

number		$\theta_1(^\circ)$	$\theta_2(^\circ)$	$\theta_3(^\circ)$	$\theta_4(^\circ)$	$\theta_5(^\circ)$	$\theta_6(^\circ)$
1	Target	-90.000000	-45.000000	90.000000	90.000000	90.000000	45.000000
	MPGA	-90.002193	-45.003086	90.003115	90.002009	90.001961	45.001583
	RBFNN	-89.999823	-45.000273	90.000419	90.000173	89.999798	45.000215
	MPGA-RBFNN	-90.000014	-45.000009	89.999986	90.000010	90.000004	45.000008
2	Target	45.000000	-45.000000	45.000000	60.000000	-90.000000	0.000000
	MPGA	44.998493	-45.001625	45.000118	60.001894	-89.998255	0.001706
	RBFNN	45.000159	-44.999092	45.000510	60.000918	-90.001094	-0.001006
	MPGA-RBFNN	44.999972	-45.000019	45.000017	60.000010	-89.999708	0.000021
3	Target	15.000000	-35.000000	45.000000	-90.000000	-45.000000	90.000000
	MPGA	15.002159	-35.001357	45.001268	90.001158	-45.004031	90.002420
	RBFNN	15.000208	-35.000237	44.999959	90.000108	-44.999391	90.000216
	MPGA-RBFNN	15.000010	-34.999990	44.999869	90.000003	-45.000021	89.999903

According to the results of the joint angle θ_2 in the first group data, it can be seen that the error value is 0.003086 when the MPGA is used to solve the inverse kinematics, the error value is 0.000273 when the RBFNN is used to solve the inverse kinematics and the error value is 0.000009 when the MPGA-RBFNN is used to solve the inverse kinematics. It improves 99.7% and 96.7% compare with the MPGA and the RBFNN. In conclusion, although the solution of the inverse kinematics can approximately obtained through both MPGA and RBFNN, the computational accuracy of the MPGA-RBFNN algorithm is better.

5.2. Speed analysis

After training samples are obtained, the newrb function in the MATLAB neural network toolbox is used to create and train for RBFNN, the network training is shown in Fig.4.

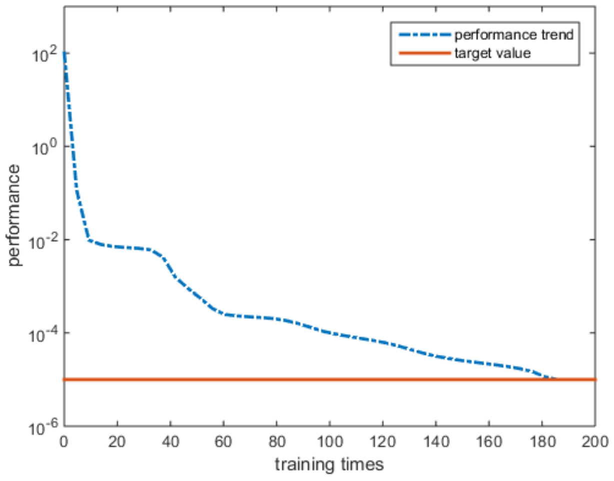


Fig.4 Performance training of MPGA-RBFNN

As we can see from Fig.4, it is clear that the training accuracy of the network is reach to the initial goal after 183 trains. In order to verify the training successful ratio of MPGA-RBFNN, the samples number of 3000 are used to obtain the performance of the three algorithms. The comparison results are shown in Table 3, the training successful ratio of the MPGA-RBFNN is up to 95% at the same sample numbers. the increment is up to 23% and 11% compare with MPGA and RBFNN. Although the convergence steps of MPGA-RBFNN are more compared with MPGA and RBFNN, the running time and training successful ratio of MPGA-RBFNN are more advantageous.

Table 3 The contrast of algorithm performance

Algorithm	Size	The training successful ratio/%	the convergence step	time/s
MPGA	3000	72	131	10.44
RBF	3000	84	117	9.36
MPGA-RBFNN	3000	95	183	9.58

6. Conclusion

In this paper, the MPGA-RBFNN algorithm is proposed based on MPGA and RBFNN, which is used to solve the problems of the inverse kinematics of general robot. Combining with the positive kinematics model of general robots, the MPGA is adopted to optimize the network structure and connection weights of RBFNN. By using the way of hybrid coding and simultaneous evolutionary, the non-linear mapping from the posture of the robot in the working space to the angle of the joint is realized, avoiding the complicated formula derivation and improving the speed of solving. Finally, the experimental is tested on the Comau NJ-220 robot, the experimental results show that the MPGA-RBFNN algorithm improves the stability of the solution, and the training success rate and calculation accuracy are improved. In the future work, we will do further research to reduce the convergence steps.

Acknowledgments

This work was supported by the National Natural Science Foundation of China (Grant No.51775076).

References

[1] Faria C, Ferreira F, Erlhagen W, et al. Position-based kinematics for 7-DoF serial manipulators with global configuration control, joint limit and singularity avoidance [J]. Mechanism & Machine Theory, 2018, 121: 317-334.
[2] Zaplana I, Basanez L. A novel closed-form solution for the inverse kinematics of redundant manipulators through workspace analysis [J]. Mechanism & Machine Theory, 2018, 121:

829-843.
[3] Zang W, Ren L, Zhang W, et al. A cloud model based DNA genetic algorithm for numerical optimization problems [J]. Future Generation Computer Systems, 2017, 81.
[4] Structures F S. Inverse Kinematic Solutions of Dual Redundant Camera Robot Based on Genetic Algorithm [J]. Mathematical Problems in Engineering, 2017, 2017: 1-10.
[5] Starke S, Hendrich N, Magg S, et al. An efficient hybridization of Genetic Algorithms and Particle Swarm Optimization for inverse kinematics [C]// IEEE International Conference on Robotics and Biomimetics. IEEE, 2017.
[6] Zubizarreta A, Larrea M, Irigoyen E, et al. Real Time Direct Kinematic Problem Computation of the 3PRS robot Using Neural Networks [J]. Neurocomputing, 2017, 271.
[7] Raşit Köker. A genetic algorithm approach to a neural-network-based inverse kinematics solution of robotic manipulators based on error minimization [J]. Information Sciences, 2013, 222 (3): 528-543.
[8] Weikuan Jia. An optimized RBF neural network algorithm based on partial least squares and genetic algorithm for classification of small sample [J]. Applied Soft Computing, 2016, 48 (1): 373-384.