Journal of Zhejiang University SCIENCE A
ISSN 1009-3095 (Print); ISSN 1862-1775 (Online)
www.zju.edu.cn/jzus; www.springerlink.com
E-mail: jzus@zju.edu.cn

Wavelet network solution for the inverse kinematics problem in robotic manipulator

CHEN Hua†, CHEN Wei-shan, XIE Tao
(*School of Mechatronics Engineering, Harbin Institute of Technology, Harbin 150001, China*)
†E-mail: chenhua202@126.com
Received June 20, 2005; revision accepted Dec. 21, 2005

Abstract: Wavelet network, a class of neural network consisting of wavelets, is proposed to solve the inverse kinematics problem in robotic manipulator. A wavelet network suitable for dealing with multi-input and multi-output system is constructed. The network is optimized by reducing the number of wavelets handling large dimension problem according to the sample data. The algorithms for sparseness analysis of input data and fitting wavelets to the output data with orthogonal method are introduced. Then Levenberg-Marquardt algorithm is used to train the network. Simulation results showed that this method is capable of solving the inverse kinematics problem for PUMA560.

Key words: Inverse kinematics problem, Robotic manipulator, Wavelet network
doi:10.1631/jzus.2006.A0525 **Document code:** A **CLC number:** TP242.2

INTRODUCTION

The inverse kinematics problem (IKP) for a robotic manipulator involves obtaining the required manipulator joint values for a given desired end-point position and orientation. It is usually complex due to lack of a unique solution and closed-form direct expression for the inverse kinematics mapping. The forward kinematics problem (FKP) is to find the position and orientation of the end-effector based on given joint values which can be easily obtained by analyzing the transformation matrix of the robotic manipulator. Numerous solution strategies have been proposed for the IKP, such as characteristic polynomial (Mavroidis *et al.*, 1994), dyalitic elimination (Raghavan and Roth, 1995), genetic programming (Chapelle and Bidaud, 2001), intelligent algorithm (Wu *et al.*, 2003). Neural networks have significant features such as learning and nonlinear approximation. These properties make neural network a promising approach for solving the IKP studied by many researchers (Guez and Ahmad, 1988; Guo and Cherkassky, 1989; Zhang and Ding, 1997; Chen *et al.*, 2002). We present a new approach, wavelet network, to solve the IKP of manipulators. This work is aimed at finding a method for solving the IKP that could be extended to any robotic manipulator.

This work is organized as follows. A wavelet network qualifying to approach the multi-input and multi-output system is first constructed. Levenberg-Marquardt algorithm is introduced to train the network. Then the approach proposed here is tested to see if it can solve the IKP of a popular manipulator, the PUMA560. Finally, some results and conclusions are drawn.

CONSTRUCTING WAVELET NETWORK

Neural networks have become an attractive tool for modelling complex non-linear systems due to its inherent ability to approximate arbitrary continuous functions. Neural networks are especially powerful tools for handling large scale problems. However, the implementation of neural networks suffers from lack of efficient constructive approaches, both for choosing

network structures and for determining the neuron parameters. Wavelet decomposition has emerged as a powerful tool for function approximation in a manner that readily reveals the properties of the function in localized regions of the joint time-frequency space. But constructing and storing wavelet basis of large dimension is very difficult and entails prohibitive cost. To overcome this shortcoming, it is necessary to develop algorithms to predigest the wavelet basis, reduce the number of wavelets and facilitate construction of the wavelet basis. Due to the similarity between wavelet decomposition and one-hidden-layer neural networks, it is very easy to think that combining wavelets and neural networks can result in a wavelet network with efficient constructive approach.

In this work, a wavelet network with d inputs, T output and L nodes is first built based on wavelet theory. Then an algorithm is used to optimize the wavelet network by reducing the number of wavelets in handling large dimension problem according to the sample data. After constructing the wavelet network, Levenberg-Marquardt algorithm is used to train it and achieve good accuracy.

A wavelet network with d inputs, T output and L nodes is built as follows (Zhang and Benveniste, 1992; Zhang, 1997):

$$f_t(\boldsymbol{x})=\sum_{j=1}^{L}\omega_{j,t}\psi_j(\boldsymbol{x}),\quad t=1,2,...,T, \tag{1}$$

where $\psi_j(\boldsymbol{x})$ is called wavelet. It is the dilations and translations form of a mother wavelet $\psi:\mathbb{R}^d\to\mathbb{R}$, and is parameterized as follows

$$\psi_j(\boldsymbol{x})=a_j^{-d/2}\psi\left(\frac{\boldsymbol{x}-\boldsymbol{b}_j}{a_j}\right)=a_0^{-m_j d/2}\psi(a_0^{-m_j}\boldsymbol{x}-\boldsymbol{n}_j b_0),\quad m_j\in\mathbb{Z},\boldsymbol{n}_j\in\mathbb{Z}^d, \tag{2}$$

where the scalar parameters a_0 and b_0 define the step sizes of dilation and translation discretizations (typically a_0=2, b_0=1). Fig.1 shows the wavelet network structure, where $\omega_{i,j}$ is the weight of the network.

As the wavelets in the wavelet network Eq.(1) usually arise from regular dilations and translations of the mother wavelet, their number increases drastically with the dimension. It is costly in resources to construct and store these wavelets. The following algorithm will reduce the number of wavelets in handling large dimension problem according to the sample data.

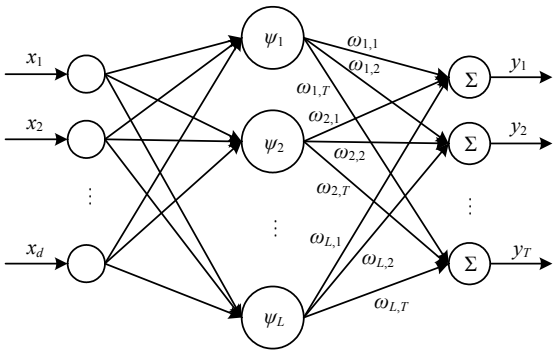

Fig.1 Structure of wavelet network

In the first step, we truncate empty wavelets by analyzing the sparseness of input data. In most practical situations of large dimension, the available sample data are finite and sparse. From the expression of wavelet network Eq.(1), we know that wavelets not overlapping any sample data are useless. Suppose the wavelet functions used in the network are compactly supported or almost compactly supported, the wavelets whose supports do not overlap any sample data should be truncated (Zhang, 1997; Sun *et al.*, 2002). This result is noted in a truncated set of wavelets W.

In the second step, because some terms in W are redundant for estimating the system output, we select the fitting wavelets by orthogonal method according to the output data. We improve on the orthogonal method introduced by Zhang (1997), and make it fit best multi-input and multi-output wavelet network. Using $(\hat{\boldsymbol{X}}, \hat{\boldsymbol{Y}})$ to denote data samples set with N sample pairs. Let $\boldsymbol{\psi}_j=\xi_j[\psi_j(\boldsymbol{x}_1), \ldots, \psi_j(\boldsymbol{x}_N)]^{\mathrm{T}}$, where $\boldsymbol{\psi}_j\in W$, j=1,2,…,L (L is the number of wavelets in W), $\boldsymbol{x}_k\in\hat{\boldsymbol{X}}$, k=1,2,…,N, ξ_j is a scalar so that $\boldsymbol{\psi}_j^{\mathrm{T}}\boldsymbol{\psi}_j$=1, and $\boldsymbol{Y}=[\boldsymbol{y}_1, \ldots, \boldsymbol{y}_T]$, where $\boldsymbol{y}_t=[y_{1t}, \ldots, y_{Nt}]^{\mathrm{T}}$, t=1,2,…,T (T is the dimension of output).

With these notations, we select the "best" subset of W (the size of subset is denoted as s) that spans the space close to the output $\boldsymbol{Y}$. The algorithm first selects the wavelet in W that best fits the output data. Then repeatedly selects the wavelet in the remainder of W that best fits the data while combining with the previously selected wavelets. For computational efficiency, later selected wavelets are orthogonalized to earlier selected ones.

At iteration i, we use $\boldsymbol{\psi}_{l_i}$ to denote the wavelet to be selected from W, $\boldsymbol{\psi}_{l_1}, \boldsymbol{\psi}_{l_2}, ..., \boldsymbol{\psi}_{l_{i-1}}$ were selected in previous iterations, and $\boldsymbol{q}_{l_k}$ is a unit orthogonalized version of $\boldsymbol{\psi}_{l_k}$. Now, orthogonalize the remaining vectors $\boldsymbol{\psi}_j$,

$$\boldsymbol{p}_j = \boldsymbol{\psi}_j - [(\boldsymbol{\psi}_j^{\mathrm{T}}\boldsymbol{q}_{l_1})\boldsymbol{q}_{l_1} + \cdots + (\boldsymbol{\psi}_j^{\mathrm{T}}\boldsymbol{q}_{l_{i-1}})\boldsymbol{q}_{l_{i-1}}], \quad (3)$$

and choose

$$l_i = \arg\max_j \{[(\boldsymbol{p}_j^{\mathrm{T}}\boldsymbol{y}_1)^2 + ... + (\boldsymbol{p}_j^{\mathrm{T}}\boldsymbol{y}_T)^2]/\boldsymbol{p}_j^{\mathrm{T}}\boldsymbol{p}_j\}. \quad (4)$$

After s iterations, the wavelet neurons $\boldsymbol{\psi}_{l_1}$, …, $\boldsymbol{\psi}_{l_s}$ in the wavelet network were selected. The network output can be obtained from

$$\hat{f}_t(\boldsymbol{x}) = \sum_{i=1}^{s} \omega_{l_i,t}\boldsymbol{\psi}_{l_i}(\boldsymbol{x}), \quad t = 1,2,...,T, \quad (5)$$

$\boldsymbol{\omega}$ is used to denote the matrix made up of $\omega_{l_i,t}$, and $\boldsymbol{\psi} = [\boldsymbol{\psi}_{l_1}, ..., \boldsymbol{\psi}_{l_s}]$, and is the least square solution of the equation $\boldsymbol{Y}=\boldsymbol{\psi}\boldsymbol{\omega}$. In addition, the previous orthogonalized procedure equivalent to the factorization $\boldsymbol{\psi}=\boldsymbol{Q}\boldsymbol{A}$, where $\boldsymbol{A}$ is an upper triangular matrix, $\boldsymbol{Q} = [\boldsymbol{q}_{l_1}, ..., \boldsymbol{q}_{l_s}]$ has orthogonal columns. Thus we have the equation $\boldsymbol{Q}^{\mathrm{T}}\boldsymbol{Y}=\boldsymbol{A}\boldsymbol{\omega}$ which can be used to compute $\boldsymbol{\omega}$ easily.

After constructing the wavelet network, its training network is necessary for it to approach the system. Levenberg-Marquardt algorithm is used here to train the wavelet network. The Levenberg-Marquardt algorithm is a variation of Newton's method that was designed for minimizing functions that are sums of squares of other nonlinear functions. This is very well suited to neural network training where the performance index is the mean squared error.

The Levenberg-Marquardt algorithm is given below

$$\Delta\boldsymbol{\omega} = (\boldsymbol{J}^{\mathrm{T}}(\boldsymbol{\omega})\boldsymbol{J}(\boldsymbol{\omega}) + \mu\boldsymbol{I})^{-1}\boldsymbol{J}^{\mathrm{T}}(\boldsymbol{\omega})\boldsymbol{e}(\boldsymbol{\omega}) \quad (6)$$

where $\boldsymbol{e}$ is the error vector, $\boldsymbol{J}$ is the Jacobian matrix. This algorithm's very useful feature is that as variable μ is increased it approaches the steepest descent algorithm with small learning rate, while as μ is decreased to zero the algorithm becomes Gauss-Newton. The Levenberg-Marquardt algorithm provides a nice compromise between the speed of Newton's method and the guaranteed convergence of steepest descent.

INVERSE KINEMATICS CALCULATION

We apply wavelet network to solve the IKP of PUMA560 as a numerical example. The architecture of PUMA560 is shown schematically in Fig.2, and the D-H parameters are listed in Table 1. The inverse kinematics problem, mapping the operation space to the joint space, is mathematically represented as f:(x, y, z, α, β, γ)→(θ_1, θ_2, θ_3, θ_4, θ_5, θ_6). On the contrary, the forward kinematics problem, mapping the joint space to the operation space, is mathematically represented as f:(θ_1, θ_2, θ_3, θ_4, θ_5, θ_6)→(x, y, z, α, β, γ). Our idea is that a wavelet network is trained to solve the IKP with the simple forward kinematics solution of PUMA560, so that the nonlinear mapping from the joint space to the operation space is accomplished quite accurately.

In this paper, 4096 training data are used to train

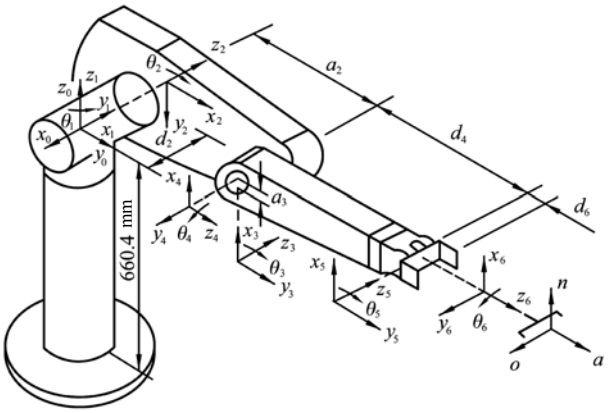

Fig.2 Architecture of PUMA560

Table 1 D-H parameters of the PUMA560

Joint	θ_i (°)	a_{i-1} (mm)	α_{i-1} (°)	d_i (mm)
1	$\theta_1 \in [-160,+160]$	0	0	0
2	$\theta_2 \in [-225,+45]$	0	−90	149.09
3	$\theta_3 \in [-45,+225]$	431.8	0	0
4	$\theta_4 \in [-110,+170]$	20.32	−90	433.07
5	$\theta_5 \in [-100,+100]$	0	90	0
6	$\theta_6 \in [-266,+266]$	0	−90	0

the wavelet network. The joint variables ranging from (−0.5234, −2.0944, −0.5234, −1.9199, 0.6981, −2.0944) to (0.5234, −1.0472, 0.5234, −0.8727, 1.7453, −1.0472), with 4 samplings along each of the six joint variables, are regarded as the outputs. And their forward kinematics solutions are regarded as the network inputs. The wavelet network has six inputs and six outputs corresponding to the joint variables (θ_1, θ_2, θ_3, θ_4, θ_5, θ_6) and the position and orientation variables (x, y, z, α, β, γ). The mother wavelet function we selected is $\psi(\boldsymbol{x})=1.8\|\boldsymbol{x}\|e^{-\|\boldsymbol{x}\|^2/2}$, where $\|\boldsymbol{x}\|=\boldsymbol{x}^{\mathrm{T}}\boldsymbol{x}$.

At first, 3645 wavelets are used to construct the wavelet network by dilations and translations of the mother wavelet. After analyzing the sparseness of input training data, 2335 wavelets are reserved. Then we select 150 fitting wavelets as the activation functions of hidden nodes that span the space close to the output training data, that is the number of hidden nodes of the final optimized wavelet network. At the same time, the initial values of weights $\omega_{j,t}$ are defined. After constructing the wavelet network, Levenberg-Marquardt algorithm is used to train the network. The whole process takes 40 min on a Pentium IV CPU 2.40 GHz computer. Finally, to evaluate the performance of the network, we use 11 testing data to test it. Eleven joint variables uniformly distributing from (0.4363, −1.9199, 0.1396, −1.4835, 1.6057, −1.4835) to (0.5236, −1.3963, −0.2094, −1.1345, 1.2566, −1.3090) are regarded as the desired outputs, and their forward kinematics solutions are regarded as the network inputs. The performance of the network is described by the following mean squared error function.

$$mse=\frac{1}{T}\sum_{t=1}^{T}mse_t=\left(\sum_{t=1}^{T}\sum_{p=1}^{N}(y_{pt}-\hat{y}_{pt})^2\right)/TN\ , \qquad (7)$$

where mse_t is the mean squared error of the tth output variable, T is the network output dimension, N is the number of the sample data, y_{pt} is the desired output, $\hat{y}_{pt}$ is the calculation value resulting from the constructed network, and the unit of orientation angle is radian. Each of the mean squared errors of six outputs and the total mean squared errors of the wavelet network tested by the testing data are shown in Table 2. The maximum squared errors are shown in Table 3.

It is observed that the performance of the network rests with the number of the hidden nodes and the size of the training datasets. The more the hidden nodes of the final optimized wavelet network, the better the performance is. It is the same if we enlarge the size of the training datasets. Of course, more hidden nodes or training data make computing time increase. In each case, the best trade-off has to be found.

Table 2 Mean squared errors (×10^{-3})

mse	mse_1	mse_2	mse_3	mse_4	mse_5	mse_6
0.1765	0.2766	0.1446	0.0292	0.2591	0.1984	0.1512

Table 3 Maximum squared errors (×10^{-3})

mse_1	mse_2	mse_3	mse_4	mse_5	mse_6
0.5952	0.3724	0.0690	0.5311	0.6609	0.3113

CONCLUSION

We proposed a solution to the IKP using wavelet network. Combining wavelet and neural networks can remedy the weakness of each other, resulting in networks with efficient constructive methods and capable of handling large dimension problems. The network is optimized by sparseness analysis of the input data and fitting wavelets selection with orthogonal method to the output data. Simulation results for PUMA560 indicate the feasibility of our approach.

One advantage of the proposed method for solving the IKP is that it can be extended to any robotic manipulator, given a set of operation space and joint space parameter values. In addition, it also can be used to solve the FKP for parallel manipulator with any selected degree of freedom and configuration.

References

Chapelle, F., Bidaud, P., 2001. A Closed Form for Inverse Kinematics Approximation of General 6R Manipulators Using Genetic Programming. Proceedings of the 2001 IEEE Internatioal Conference on Robotics and Automation, Seoul, Korea, **4**:3364-3369.

Chen, X.S., Chen, Z.L., Xie, T., 2002. An accurate solution to the inverse kinematic problem of a robot manipulator based on the neural network. *Chinese Journal of Robot*, **24**(2):130-133 (in Chinese).

Guez, A., Ahmad, Z., 1988. Solution to the Inverse Kinematics Problem in Robotics by Neural Networks. IEEE

International Conference on Neural Network, San Diego, California, **2**:617-621.

Guo, J., Cherkassky, V., 1989. A Solution to the Inverse Kinematic Problem in Robotics Using Neural Network Processing. IEEE International Conference on Neural Network, Washington D.C., **2**:299-304.

Mavroidis, C., Ouezdou, F.B., Bidaud, P., 1994. Inverse kinematics of six degree of freedom 'general' and 'special' manipulators using symbolic computation. *Robotica*, **12**(5):421-430.

Raghavan, M., Roth, B., 1995. Solving polynomial systems for the kinematics analysis and synthesis of mechanisms and robot manipulator. *Journal of Mechanical Design*, **117**:71-79.

Sun, W., Wang, Y., Mao, J., 2002. Using Wavelet Network for Identifying the Model of Robot Manipulator. Proceedings of the 4th World Congress on Intelligent Control and Automation, **2**:1634-1638.

Wu, Y.C., Hsieh, N.H., Chen, Z.R., 2003. The Application of Intelligent Algorithm Principle on Robot Exercise. Proceedings of the 2003 IEEE/ASME International Conference on Advanced Inteligent Mechatronics (AIM 2003), p.371-376.

Zhang, Q.H., 1997. Using wavelet network in nonparametric estimation. *IEEE Trans. Neural Network*, **8**(2):227-236. [doi:10.1109/72.557660]

Zhang, Q., Benveniste, A., 1992. Wavelet networks. *IEEE Trans. Neural Networks*, **3**(6):889-898. [doi:10.1109/72.165591]

Zhang, W., Ding, Q., 1997. Inverse kinematics for a 6 DOF manipulator based on neural network. *Transactions of Nanjing University of Aeronautics and Astronautics*, **14**(1):73-76.