

Evolutionary Structured RBF Neural Network Based Control of a Seven-Link Redundant Manipulator

Thrishantha Nanayakkara¹, Keigo Watanabe², Kazuo Kiguchi²,
and Kiyotaka Izumi³

¹ Faculty of Engineering Systems and Technology,
Graduate School of Science and Engineering,

² Department of Advanced Systems Control Engineering,
Graduate School of Science and Engineering,

³ Department of Mechanical Engineering,
Faculty of Science and Engineering,

Saga University, 1-Honjomachi, Saga 840-8502, Japan
{watanabe, kiguchi, izumi}@mc.saga-u.ac.jp

Abstract: A method for the identification of complex non-linear dynamics of a multi-link robot manipulator using Runge-Kutta-Gill Neural Networks (RKGNNs) in the absence of input torque information is proposed. The RKGNNs constructed using shape adaptive radial basis functions (RBF) are trained using an evolutionary algorithm. Due to the fact that the main function network is divided into sub-networks to represent detailed properties of the dynamics of a manipulator, the neural networks have greater information processing capacity and they can be tested for properties such as positive definiteness of the inertia matrix. Dynamics of an industrial seven-link manipulator are identified using only input-output position and their velocity data. Promising experimental control results are obtained to prove the ability of the proposed method in capturing highly nonlinear dynamics of a multi-link manipulator in an effective manner.

Keywords: Runge-Kutta-Gill neural networks, Radial basis functions, Multi-link robot manipulators, Evolutionary optimization

1. Introduction

The ability to grasp the full dynamics of manipulators accurately is of much importance in judging the robustness of model-based control strategies such as computed torque controllers. The problem often encountered in the identification of dynamics of multi-link industrial manipulators is the lack of knowledge of necessary data such as input joint torques or the gains of the servo controllers. Identification of inertia parameters using the least-squares algorithm has been investigated in ¹. Yet these methods need the input joint torque information which needs special hardware changes in an industrial manipulator. Identification of dynamics of manipulators using artificial neural networks (ANNs) has been studied in ², ³. These methods suffer from several drawbacks: one is the tendency of backpropagation algorithms to converge to local minima, another is that in some of these methods the trained NN depends on the sampling time width of the training data and the other is that these methods do not make use of the inherent structure of the dynamics of manipulators. Moreover the simulation results are shown only for two link manipulators, where the nonlinearities of the dynamics are relatively simple compared with the actual industrial manipulators. Therefore the ability of these methods to accurately grasp the dy-

namics of a manipulator with high degrees-of-freedom is questionable.

The proposed method synthesizes a structured function NN consisting of sub-networks, so that the sub-networks represent the components of the dynamics of the manipulator⁴), to catch full dynamics of a manipulator. Shape adaptive RBFNNs have been used for each sub-network⁵). The proposed method uses RKGNNs, which is an extension version of Runge-Kutta Neural Networks (RKNNs), for the identification of the dynamics, because RKNNs are capable of identifying the changing rates of the states accurately due to the state space interpolation between one sampling interval⁶). Furthermore, this method only needs one state information to predict the next state and the trained NN of the function is independent of the sampling time width unlike in direct mapping NNs. Weights of the NNs are optimized using a new evolutionary algorithm⁷).

Experimental studies are carried out for a seven-link robot manipulator called PA-10, manufactured by the Mitsubishi Heavy Industries Ltd. Promising results are obtained to prove the ability of the proposed method in capturing highly nonlinear dynamics of a multi-link manipulator in an effective manner.

2. Identifying the Dynamics of a Manipulator

2.1 Manipulator dynamic model and its properties

Dynamics of a manipulator is given by

$$\tau = M(\theta)\ddot{\theta} + V(\theta, \dot{\theta}) + G(\theta) + F(\theta) \quad (1)$$

where $\tau \in \mathfrak{R}^n$ is the joint torque vector, $\theta, \dot{\theta}, \ddot{\theta} \in \mathfrak{R}^n$ are the joint angle, angular velocity, and angular acceleration vectors, respectively, $M(\theta) \in \mathfrak{R}^{n \times n}$ is the inertia matrix, $V(\theta, \dot{\theta}) \in \mathfrak{R}^n$ is the centrifugal and Coriolis force vector, $G(\theta) \in \mathfrak{R}^n$ is the gravity force vector, and $F(\theta) \in \mathfrak{R}^n$ is the friction vector, where n is the number of joints of the manipulator⁴.

It is well known that the rigid dynamics of equation (1) has the following properties².

Property 1—Boundedness of the Inertia Matrix: The inertia matrix $M(\theta)$ is symmetric and positive definite, and satisfies the following inequalities:

$$m_1\|y\|^2 \leq y^T M(\theta)y \leq m_2\|y\|^2, \quad \forall y \in \mathfrak{R}^n, \quad (2)$$

where m_1 and m_2 are known positive constants and $\|\cdot\|$ denotes the standard Euclidean norm.

Property 2—Skew symmetry: The inertia and centrifugal-Coriolis matrices have the following property:

$$y^T \left(\frac{1}{2} \dot{M}(\theta) - C(\theta, \dot{\theta}) \right) y = 0, \quad \forall y \in \mathfrak{R}^n, \quad (3)$$

where $\dot{M}(\theta)$ is the time derivative of the inertia matrix and $V(\theta, \dot{\theta}) = C(\theta, \dot{\theta})\dot{\theta}$.

2.2 The new concept for effective dynamics identification

It can be seen in⁴ that the derivation of the dynamic equation for a manipulator with higher degrees-of-freedom such as PA-10, is very complex and time consuming. Even if the dynamic structure is known for such a manipulator, many crucial parameters such as link inertia is unknown or partially known. The identification of these unknown parameters involves data collection including the input torque sensor information if conventional least-squares method¹ or direct mapping NNs (DMNN) approach is employed. Yet in most practical situations, the torque sensors for all joints are not built-in with the manipulator. Therefore if torque sensors are to be used, some hardware changes are required that makes additional trouble and expense.

The proposed method attempts to identify the dynamics involving only the reference input of the joint angle, its velocity, and their output information, using the RKGNNs trained by an evolutionary algorithm. This needs rearranging of the dynamic equation in (1) to form

an ordinary differential equation (ODE) of the form $\dot{x} = f(x)$. Given the dynamics of the manipulator as in equation (1), the state vector x can be defined by $x = [\theta \ \dot{\theta}]^T$. Then the ODE given by (4) can be formed using x and the dynamics defined by equation (1) as

$$\begin{bmatrix} \dot{\theta} \\ \ddot{\theta} \end{bmatrix} = \begin{bmatrix} 0 & I \\ 0 & 0 \end{bmatrix} \begin{bmatrix} \theta \\ \dot{\theta} \end{bmatrix} + \begin{bmatrix} 0 \\ M^{-1}(\theta)\{\tau - V(\theta, \dot{\theta})\} \end{bmatrix}. \quad (4)$$

3. RKGNNs and the Structure of the Function NNs

3.1 RKGNNs for the dynamics identification of manipulators

RKGNNs are an extended version of RKNNS⁶. An r th-order RKNNS consists of r identical function networks each with identical network structure and weights connected in such a way to realize r th-order Runge-Kutta algorithm. Each function network models the right hand side of the ODE directly. The RKGNN differs from RKNNS in the way the function NNs are connected. When defining two time steps as $t_i = ih$ and $t_{i+1} = (i+1)h$, the input-output relationship of a fourth-order RKGNN is given by

$$x(i+1) = x(i) + \frac{h}{6}(k_0 + e k_1 + g k_2 + k_3) \quad (5)$$

where h is the sampling time width, and

$$k_0 = N_f(x(i))$$

$$k_1 = N_f(x(i) + \frac{h}{2}k_0)$$

$$k_2 = N_f(x(i) + ahk_0 + bhk_1)$$

$$k_3 = N_f(x(i) + chk_1 + dhk_2)$$

in which $a = 1/\sqrt{2} - 1/2$, $b = 1 - 1/\sqrt{2}$, $c = -1/\sqrt{2}$, $d = 1 + 1/\sqrt{2}$, $e = 2 - \sqrt{2}$, and $g = 2 + \sqrt{2}$, with the function NN denoted by $N_f(\cdot)$. The fundamental idea of an RKGNN is depicted by Fig. 1. Note that if $a = 0$, $b = 1/2$, $c = 0$, $d = 1$, $e = 2$, and $g = 2$, the RKGNN becomes an RKNNS.

From Fig. 1, it can be seen that the prediction of the next state involves one previous state information and the sampling width is external to the function NN model. One other feature is that all four NN models given by N_f in the RKGNN are the same. Therefore finding the optimum weights for one N_f is all that is required to identify the system.

When using an RKGNN that minimizes the memory storage, we further define the following new vectors:

$$q_1 = k_0$$

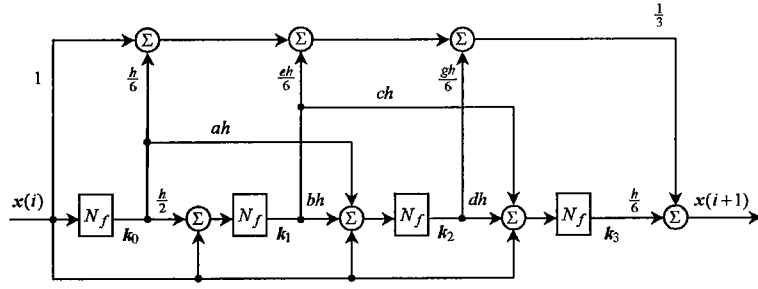


Fig. 1 Overall arrangement of one-step ahead predictive network based on the Runge-Kutta-Gill method with the function neural networks N_f .

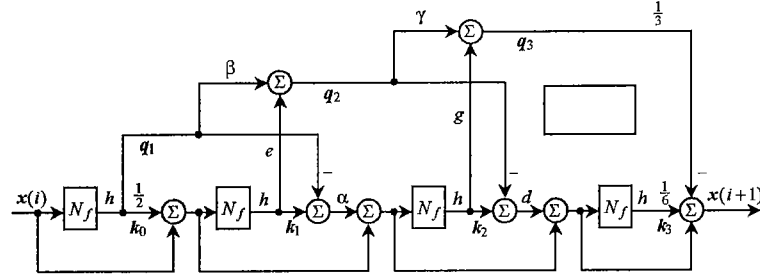


Fig. 2 Overall arrangement of one-step ahead predictive network based on the alternative Runge-Kutta-Gill method with the function neural networks N_f .

$$q_2 = \left(-2 + \frac{3}{\sqrt{2}}\right)k_0 + (2 - \sqrt{2})k_1$$

$$q_3 = -\frac{1}{2}k_0 - (1 + \sqrt{2})k_1 + (2 + \sqrt{2})k_2.$$

Then, the RKGNN algorithm contains the following four steps:

Step 1:

$$k_0 = hN_f(x(i))$$

$$x^{(1)}(i+1) = x(i) + \frac{1}{2}k_0$$

$$q_1 = k_0$$

Step 2:

$$k_1 = hN_f(x^{(1)}(i+1))$$

$$x^{(2)}(i+1) = x^{(1)}(i+1) + \alpha(k_1 - q_1)$$

$$q_2 = \beta q_1 + e k_1$$

Step 3:

$$k_2 = hN_f(x^{(2)}(i+1))$$

$$x^{(3)}(i+1) = x^{(2)}(i+1) + d(k_2 - q_2)$$

$$q_3 = \gamma q_2 + g k_2$$

Step 4:

$$k_3 = hN_f(x^{(3)}(i+1))$$

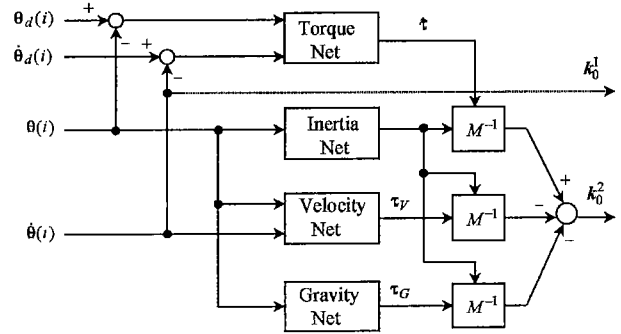


Fig. 3 Basic concept of the function NN to identify each component of the dynamic equation.

$$x^{(4)}(i+1) = x^{(3)}(i+1) + \frac{1}{6}k_3 - \frac{1}{3}q_3$$

$$x(i+1) \equiv x^{(4)}(i+1)$$

where, $\alpha = 1 - \sqrt{2}/2$, $\beta = -2 + 3/\sqrt{2}$, and $\gamma = -(2 + 3/\sqrt{2})$. The alternative illustration is given in **Fig. 2**.

3.2 The anatomy of the function NN

The dynamics equation of a rigid link manipulator can be analyzed into the following basic components: one is the inertia matrix, the other is the velocity dynamics given by centrifugal and Coriolis force vectors, another is the gravity force vector. In the proposed method, separate networks are constructed to identify the dynamics of each of these components. In addition to these, in a practical industrial application, the information about the

servo controller is unknown. Therefore a torque network is constructed that will grasp the dynamics of the servo controller. Thus, it is very important to notice that the function $NN N_f(\cdot)$ as shown in Fig. 3 contains components for the identification of each component of the dynamic equation as shown in equation (1).

3.2.1 Torque network The torque network has two sub-networks for the identification of proportional and derivative gains of the actual controller are employed. The angular error vector given by $(\theta_d - \theta)$ and the angular velocity error vector given by $(\dot{\theta}_d - \dot{\theta})$ are given as inputs to the proportional and derivative networks respectively to produce a suitable torque vector for the robot manipulator, where θ_d is the reference angular trajectory and $\dot{\theta}_d$ is the reference angular velocity. If detailed information of the controller structure is known, then the structure of the network can be altered accordingly. Moreover, the torque data is directly provided to the function NN, if the input torque can be measured for each joint.

3.2.2 Inertia network The structure of the sub-networks for the inertia network is designed such that they identify the diagonal and upper triangular components of the inertia matrix of the dynamic equation. This helps to make sure that the inertia matrix is symmetric and positive definite.

3.2.3 Velocity network The velocity network set includes centrifugal and Coriolis components. Basically the output of the velocity network is given by

$$V(\theta, \dot{\theta}) = B(\theta)[\dot{\theta}\dot{\theta}] + C(\theta)[\dot{\theta}^2] \quad (6)$$

where $B(\theta)$ is a matrix of Coriolis effect, whose size is $n \times n(n-1)/2$ and $[\dot{\theta}\dot{\theta}]$ is a $n \times (n-1)/2$ vector consisting of

$$[\dot{\theta}\dot{\theta}] = [\dot{\theta}_1\dot{\theta}_2 \quad \dot{\theta}_1\dot{\theta}_3 \quad \dots \quad \dot{\theta}_{n-1}\dot{\theta}_n]^T. \quad (7)$$

$C(\theta)$ is a matrix of centrifugal effect, whose size is $n \times n$, and

$$[\dot{\theta}^2] = [\dot{\theta}_1^2 \quad \dot{\theta}_2^2 \quad \dots \quad \dot{\theta}_n^2]^T. \quad (8)$$

3.2.4 Gravity network The gravity network consists of one RBF network with n RBFs and n outputs, if the manipulator will be moved in the gravity direction.

The construction of sub-networks of each of these component networks involves construction of shape adaptive RBFNNs.

The input-output mapping relationship of the RBF in this application is given by

$$\phi_l(\mathbf{y}; \bar{\mathbf{y}}_l, \sigma_l) = \exp\{-[(\mathbf{y} - \bar{\mathbf{y}}_l)^T(\mathbf{y} - \bar{\mathbf{y}}_l)/\sigma_l^2]\} \quad (9)$$

where $\mathbf{y} \in \mathbb{R}^n$ and $\bar{\mathbf{y}}_l \in \mathbb{R}^n$ is the input vector and the vector of centers of the RBF, σ_l^2 is the variance of the l th RBF. Therefore the output at the k th output node of the sub-network $N_{sub}(\cdot)$ is given by

$$[N_{sub}(\mathbf{y})]_k = \sum_{l=1}^N W_{kl} \phi_l(\mathbf{y}; \bar{\mathbf{y}}_l, \sigma_l), k = 1, \dots, m \quad (10)$$

where N is the number of RBFs that construct the sub-network and W_{kl} is the connection weight from l th RBF to k th output node. Here we make the standard assumption made in NN literature²⁾ given next.

Assumption: When defining the weight matrix $W = \{W_{kl}\}$, $k = 1, \dots, n$, $l = 1, \dots, N$, the ideal weights are bounded by known positive values so that

$$\|W\|_F \leq w_M \quad (11)$$

where $\|\cdot\|_F$ is the Frobenius norm, i.e., $\sqrt{\text{tr}(WW^T)}$, and w_M is known.

This assumption is made in the initialization of the weights of the NN in the evolutionary algorithm employed to train the NNs, in this application.

4. Procedure of the System Identification

The PA-10 manipulator consists of seven links with seven degrees-of-freedom (three rotation axes and four pivot axes) as illustrated in Fig. 4. The main specifications of the PA-10 manipulator are as follows: the length of the manipulator is 1345 mm, the weight of the manipulator is 35 kgf, the maximum combined speed with all axes is 1.55 m/s, the payload weight is 10 kgf, and output torque is 9.8 Nm.

For a given trajectory, the input reference angle, its velocity and their output data of all seven joints were experimentally obtained. Then these experimental data were used to train the RKGNNs to identify the dynamics of the manipulator. The weights of the RKGNN and the RBF parameters, $\bar{\mathbf{y}}_l$ and σ_l , were found using the evolutionary algorithm described in references^{7), 8)}, with the objective function subjected to constraints⁹⁾. The constrained objective function is given by

$$J = \left(\frac{1}{t_N} \sum_{i=1}^{t_N} \|\theta_d(i) - \theta(i)\| + \|\dot{\theta}_d(i) - \dot{\theta}(i)\| \right) + \frac{s_t}{2} \left(\sum_{j=1}^{c_N} [g_j^+(\theta)]^2 \right) \quad (12)$$

where t_N is the total time span of the trajectory, $g_j^+(\theta) = \max\{0, g_j(\theta)\}$ is the additional penalty function for the j th constraint satisfaction and s_t is a log penalty parameter such that $s_t = 10 + \log(i+1)$, and c_N is the number of constraints.

One constraint was considered: which is given by

$$g_1(\theta) : -\Gamma(\theta) \leq 0 \quad (13)$$

where $\Gamma(\theta)$ is the minimum principle minor of the inertia matrix.

Then the parameters of the function NN to be optimized were encoded in the elements of individuals of the evolutionary algorithm as real valued vectors in the following ranges. All the weights were initialized in the range $[-2, 2]$, the variances in $[0, 5]$, and centers of

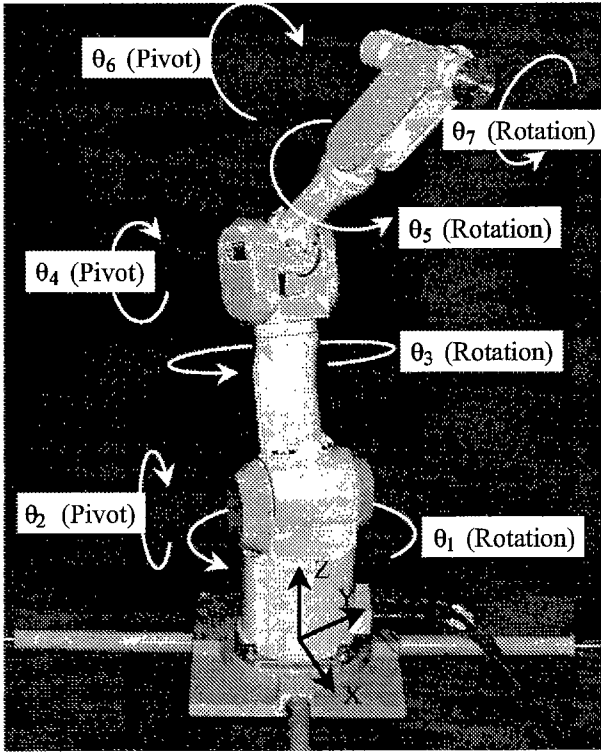


Fig. 4 Mechanical structure of the PA-10 manipulator.

RBFs in $[-2, 2]$ except for the following: weights, variances, and centers of RBFs in the torque network were in $[-10, 10]$, $[0, 100]$, and $[-1, 1]$, respectively; the weights of the gravity network were in $[-5, 0]$; the weights of the velocity net were in $[-0.1, 0.1]$. The conditions of the usage of evolutionary algorithm were as follows: 100% mutation and 20% crossover were employed. The tournament size was 15 and the number of individuals was 100. One individual consists of 3080 elements. The algorithm was run for 200 generations. The size of the regression vector n_p was 2, and the strategic move percentage μ_p was 10%.

5. Controller

A model based controller is used to generate joint velocity commands as given by

$$\ddot{\theta}_c = \hat{M}^{-1}(\theta) \left[\tau_d + \{ K_p e + K_v \dot{e} + K_I \int e \} + \hat{V}(\theta, \dot{\theta}) + \hat{G}(\theta) \right] \quad (14)$$

where $\ddot{\theta}_c$ is the manipulated joint acceleration; \hat{M}^{-1} is the estimated inertia matrix; τ_d is the torque command estimated by the torque network; K_p , K_v ; and K_I are the proportional; derivative and integral gains respectively; e is the vector of joint angle errors; $\hat{V}(\theta, \dot{\theta})$ is the estimated velocity force matrix; and $\hat{G}(\theta)$ is the estimated gravitational force matrix.

6. Experimental Results

The following Figs. 5-10 show the experimental results. The experimental results encourage the use of this

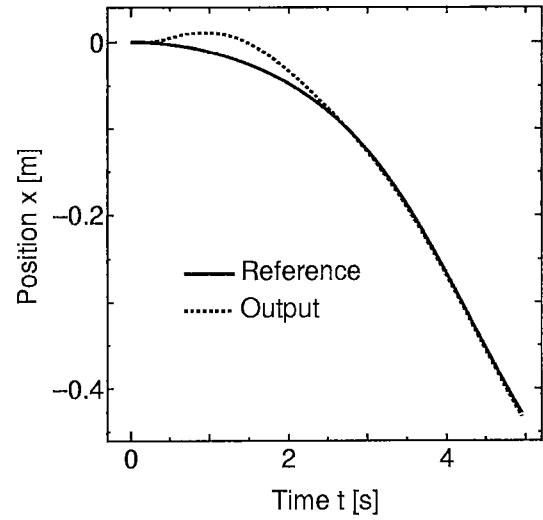


Fig. 5 The reference and the output in x-direction.

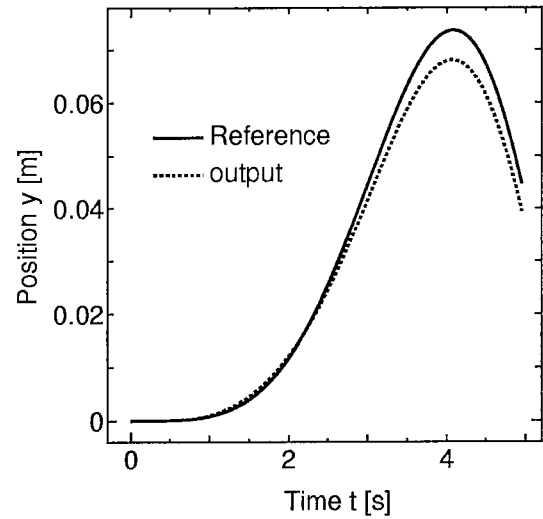


Fig. 6 The reference and the output in y-direction.

method for identification of dynamics of a complex robot manipulator.

7. Conclusions

The Runge-Kutta-Gill neural network (RKGNN) has been developed and it was applied for identifying the complex dynamics of manipulators with higher degrees-of-freedom. It is very useful, because it reduces the burden of deriving the complex dynamic equations for such manipulators. The adoption of RKGNNs makes the long-term prediction of the states very accurate due to its property of interpolation of states within one sampling interval and estimating the rate of change of states accurately. This method is advantageous in the practical situation where input torque information is unknown. The key feature of using a structured function NN gives room to adopt any kind of model based controller due to the separability of the components of the dynamics such as inertia, Coriolis, centrifugal, and gravity matrices.

Experimental studies were carried out to demonstrate

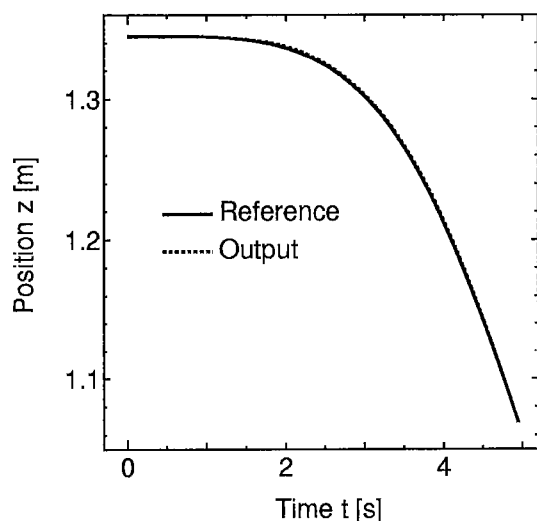


Fig. 7 The reference and the output in z-direction.

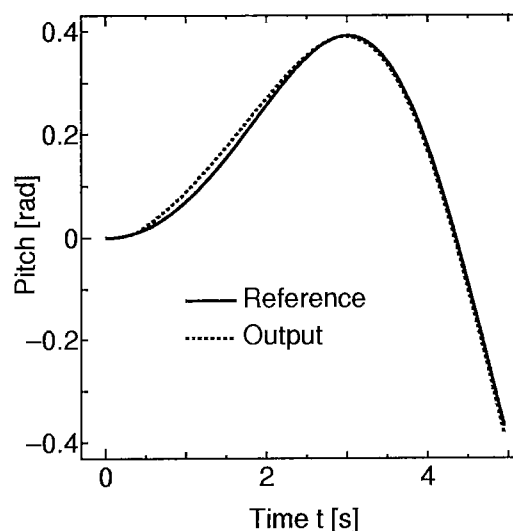


Fig. 9 The reference and the output for the pitch orientation.

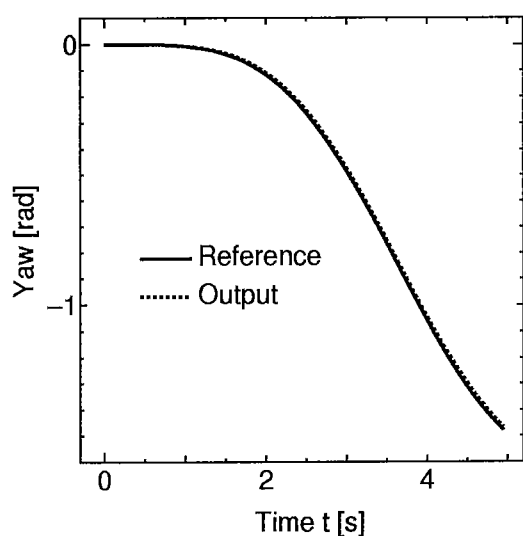


Fig. 8 The reference and the output for the yaw orientation.

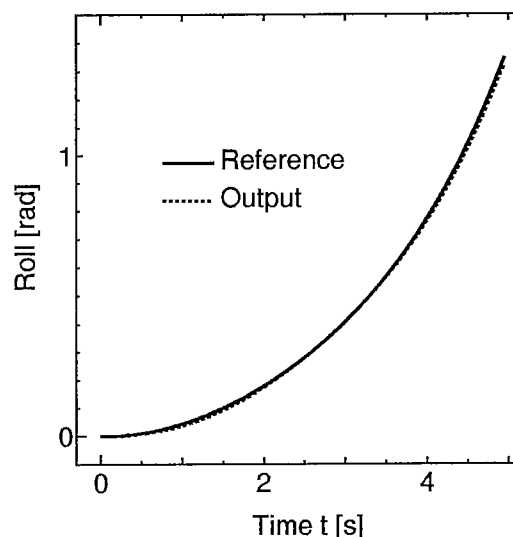


Fig. 10 The reference and the output for the roll orientation.

the ability of the proposed method to capture the complex dynamics of a multi-link manipulator, by using an industrial seven-link robot manipulator called PA-10. Promising experimental results have been obtained.

References

- 1) C. H. An, C. G. Atkeson, J. M. Hollerbach, "Estimation of Internal Parameters of Rigid Body Links of Manipulators," *Artificial Intelligence Memo 887, MIT Artificial Intelligence Laboratory*, 1986.
- 2) C. Kwan, F. L. Lewis, and D. M. Dawson, "Robust Neural-Network Control of Rigid-Link Electrically Driven Robots," *IEEE Trans. on Neural Networks*, vol. 9, no. 4, pp. 481-488, 1998.
- 3) L. Behera, M. Gopal, and S. Chaudhury, "On Adaptive Trajectory Tracking of a Robot Manipulator Using Inversion of Its Neural Emulator," *IEEE Trans. on Neural Networks*, vol. 7, no. 6, pp. 1401-1414, 1996.
- 4) J. J. Craig, *Introduction to Robotics: Mechanics and control*, 2nd edn, Reading, MA: Addison-Wesley, 1989.
- 5) A. R. Web and S. Shannon, "Shape Adaptive Radial Basis Functions," *IEEE Trans. on Neural Networks*, vol. 9, no. 6, pp. 1155-1166, 1998.
- 6) Y.-J. Wang and C.-T. Lin, "Runge-Kutta Neural Network for Identification of Dynamical Systems in High Accuracy," *IEEE Trans. on Neural Networks*, vol. 9, no. 2, pp. 294-307, 1998.
- 7) T. Nanayakkara, K. Watanabe and K. Izumi, "Evolving in Dynamic Environments Through Adaptive Chaotic Mutation," in *Procs. of Fourth International Symposium on Artificial Life and Robotics*, Oita, Japan, Jan 19-22, vol. 2, pp. 520-523, 1999.
- 8) T. Nanayakkara, K. Watanabe, K. Kiguchi, and K. Izumi "Evolutionary Optimization with Strategic Moves Based on Historical Knowledge," in *Procs. of the first SOFT Kyushu Branch Annual Conference*, pp. 21-28, 1999.
- 9) J.-H. Kim, and Hyun Myung "Evolutionary Programming Techniques for Constrained Optimization Problems," *IEEE Trans. on Evolutionary Computation*, vol. 1, no. 2, pp. 129-140, 1997.