

# Evolving Runge-Kutta-Gill RBF Networks to Estimate the Dynamics of a Multi-Link Manipulator

Thrishanta Nanayakkara\*, Keigo Watanabe\*\*<sup>†</sup> and Kiyotaka Izumi\*\*\*

\*Graduate School of Science and Engineering,  
Faculty of Engineering Systems and Technology,

\*\*Department of Advanced Systems Control Engineering,  
Graduate School of Science and Engineering,

\*\*\*Department of Mechanical Engineering,  
Faculty of Science and Engineering,

Saga University, 1-Honjomachi, Saga 840-8502, Japan

<sup>†</sup>E-mail : watanabe@me.saga-u.ac.jp

## ABSTRACT

This paper proposes a method for identification of dynamics of a multi-link robot arm using Runge-Kutta-Gill Neural networks (RKGNN). Shape adaptive radial basis function (RBF) neural networks have been employed with an evolutionary algorithm to optimize the shape parameters and the weights of the RKGNN. Due to the fact that the RKGNN can accurately grasp the changing rates of the states, this method can effectively be used for long term prediction of the states of the robot arm dynamics. Unlike in conventional methods, the proposed method can even be used without input torque information because a torque network is part of the functional network. This method can be proposed as an effective option for dynamics identification for manipulators with high degrees of freedom, as opposed to the derivation of dynamic equations and making additional hardware changes in the case of statistical parameter identification such as linear least-squares method.

## 1. INTRODUCTION

Identification of dynamics of robotic manipulators is vital in designing suitable controllers such as computed torque algorithms [1]. In most of the industrial robot arms, the parameters of the dynamics are vaguely known or incompletely known to design proper control algorithms. Identification of inertia parameters using the least squares algorithm has been investigated in [2], [3]. Identification of dynamics using neural networks has been studied in [4]. These methods have the problems of finding a globally optimum solution and the difficulty of identifying a complex system with high degrees of freedom due to the backpropagation algorithm to find weights. Moreover these methods use input torque information for the calculations that can be expensive on one hand and on the other hand, needs special hardware changes. Therefore the motivation of this research has been to develop a method that is capable of identifying the dynamics of a complex multi-link robot arm with basic reference input joint angle and its velocity, and their output data.

The proposed method uses Runge-Kutta-Gill Neural Networks (RKGNNs), which is an extension version of Runge-Kutta Neural Networks (RKNN), 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 interval [5]. Furthermore, this method only needs one state information to predict the next state and the trained neural network of the function is independent of the sampling time interval unlike in direct mapping neural networks [6]. The function neural network has sub networks to represent the components of the dynamics of the manipulator [1]. Radial Basis Function (RBF) neural networks have been used for each subnetwork [7]. Due to the complex structure of the dynamics of a manipulator, the number of weights of the total neural network system demands for efficient optimization techniques to train the networks. Therefore optimization of weights using a new evolutionary optimization algorithm has been proposed in this paper [8], [9], [10], [11]. Due to the property of monotonically reducing standard deviation of mutation, the convergence rate is fast in the evolutionary algorithm [11]. In the proposed algorithm, sustaining adequate standard deviation of mutation throughout the optimization process guarantees the convergence to a global optimality [12].

Simulation studies have been carried out for a three-link robot arm for simplicity. Proportional Derivative (P-D) controller was designed to control the three link manipulator and then the system identification studies using the RKGNN have been carried out for three cases: the input torque information is known; the gains of the P-D controller are known; no information of the controller is known whatsoever.

Rest of this paper is organized as follows: In section 2, a brief introduction to the basic components of the dynamics of a multi-link robot arm and the problems in a practical dynamics identification study is explained. In section 3, the RKGNNs and the structure of the functional networks are explained. In section 4, the procedure of training the neural networks for a three-link manipulator is explained. In section 5, the results are discussed.

## 2. IDENTIFYING THE DYNAMICS OF A MANIPULATOR

Dynamics of a manipulator is given by

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

where  $\tau$  is the joint torque vector,  $\theta$  is the joint angle vector,  $\dot{\theta}$ ,  $\ddot{\theta}$  are joint angular velocity and angular acceleration

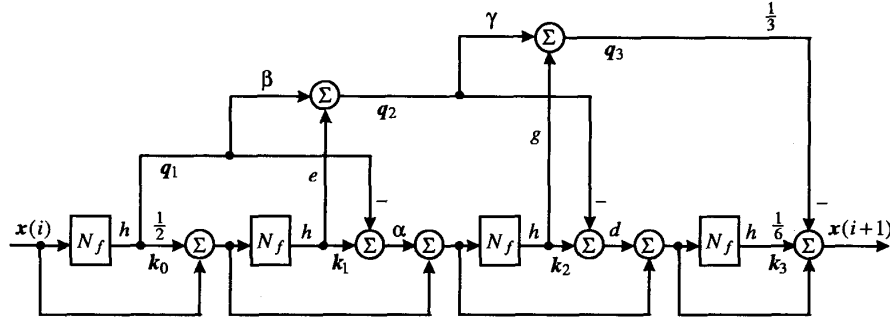


Fig. 1 Structure of the fourth-order RKGNN by the function neural networks  $N_f$ .

vectors respectively,  $M(\theta)$  is the inertia matrix,  $V(\theta, \dot{\theta})$  is the coriolis and centrifugal force vector,  $G(\theta)$  is the gravity force vector and  $F(\theta)$  is the friction vector [1].

It can be seen in [1] that the derivation of this dynamic equation for a manipulator with higher degrees of freedom such as PA-10 arm, manufactured by Mitsubishi Heavy Industries, is very complex and time consuming. Even if the dynamic equation 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 method [2], [3] or direct mapping neural network (DMNN) approach [6] is employed. Yet in most practical situations, the torque sensors 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 joint angle and 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

$$\dot{x} = f(x). \quad (2)$$

Given the dynamics of the manipulator as in equation (1), the state vector  $x$  can be defined by

$$x = [\theta \quad \dot{\theta}]^T. \quad (3)$$

Then the ODE given by (2) can be formed using  $x$  and the dynamics defined by equation (1) as

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

The right-hand side of the above ODE gives the rate of change of the states given by the left-hand side. Therefore for this family of ODEs, the right hand side may be very accurately identified by RKGNNs.

### 3. RKGNNs AND THE STRUCTURE OF THE FUNCTIONAL NETWORKS.

The basic idea of a RKGNN is depicted by Fig. 1. From Fig. 1, it can be seen that the prediction of the next state involves one previous state information and the sampling interval is external to the function neural network model. One other feature is that all four neural network 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.

The RKGNN algorithm contains the following steps.

Step 1:

$$\begin{aligned} k_0 &= hN_f(x(i)) \\ x^{(1)}(i+1) &= x(i) + \frac{1}{2}k_0 \\ q_1 &= k_0 \end{aligned}$$

Step 2:

$$\begin{aligned} k_1 &= hN_f(x^{(1)}(i+1)) \\ x^{(2)}(i+1) &= x^{(1)}(i+1) + \alpha(k_1 - q_1) \\ q_2 &= \beta q_1 + e k_1 \end{aligned}$$

Step 3:

$$\begin{aligned} k_2 &= hN_f(x^{(2)}(i+1)) \\ x^{(3)}(i+1) &= x^{(2)}(i+1) + d(k_2 - q_2) \\ q_3 &= \gamma q_2 + g k_2 \end{aligned}$$

Step 4:

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

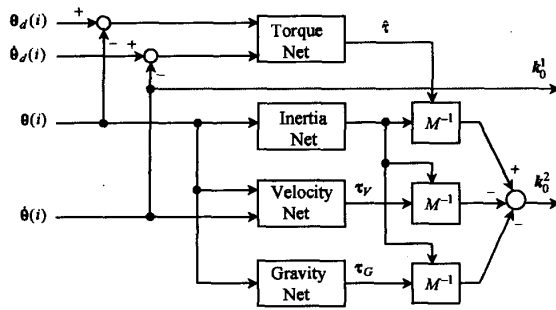


Fig. 2 Basic concept of the functional neural network to identify each component of the dynamic equation.

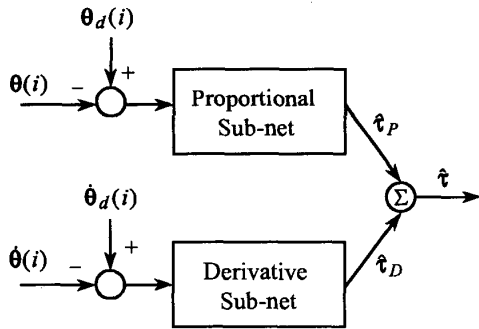


Fig. 3 Torque network for identifying the proportional and derivative gains in a P-D servo controller.

$$\begin{aligned} \mathbf{x}^{(4)}(i+1) &= \mathbf{x}^{(3)}(i+1) + \frac{1}{6}k_3 - \frac{1}{3}a_3 \\ \mathbf{x}(i+1) &\equiv \mathbf{x}^{(4)}(i+1) \end{aligned}$$

In this case,  $h$  is the time step width,  $d = (\sqrt{2}+1)/\sqrt{2}$ ,  $e = (2-\sqrt{2})$ ,  $g = (2+\sqrt{2})$ ,  $\alpha = (\sqrt{2}-1)/\sqrt{2}$ ,  $\beta = -2+3/\sqrt{2}$ , and  $\gamma = -(2+3/\sqrt{2})$ .

The functional neural network  $N_f(\cdot)$  as shown in Fig. 2 contains components for identification of each component of the dynamic equation as shown in equation (1). Since information about the controller is unknown, a torque network is constructed that will grasp the dynamics of the controller.

The structure of the torque network is depicted by Fig. 3, where two sub networks for 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 arm, where  $\theta_d$  is the desired angular trajectory and  $\dot{\theta}_d$  is the desired angular velocity. If detailed information of the controller is known, the structure of the network can be altered accordingly.

The structure of the sub-networks of the inertia network is designed such that they identify the diagonal and upper

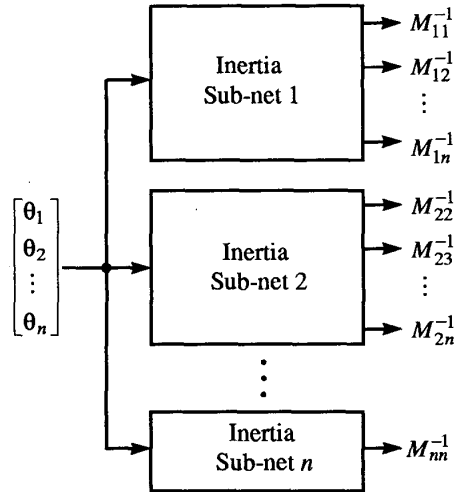


Fig. 4 Inertia network considering the symmetry of the inertia matrix.

triangular components of the inertia matrix of the dynamic equation, because the inertia matrix must be symmetric and positive definite. The block diagram of the inertia subnetworks can be seen in Fig. 4.

The velocity network set which includes centrifugal and Coriolis components are shown in Figs. 5, 6 and 7 respectively. Basically the output of the velocity network is given by

$$V(\theta, \dot{\theta}) = B(\theta)[\dot{\theta}\dot{\theta}] + C(\theta)[\dot{\theta}^2] \quad (4)$$

where,  $B(\theta)$  is a matrix of size  $n \times n(n-1)/2$  and

$$[\dot{\theta}\dot{\theta}] = [\dot{\theta}_1\dot{\theta}_2 \quad \dot{\theta}_1\dot{\theta}_3 \quad \dots \quad \dot{\theta}_{n-1}\dot{\theta}_n]^T. \quad (5)$$

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

$$[\dot{\theta}^2] = [\dot{\theta}_1^2 \quad \dot{\theta}_2^2 \quad \dots \quad \dot{\theta}_n^2]^T. \quad (6)$$

The construction of subnetworks of each of these component networks involve construction of shape adaptive RBF neural networks as can be seen in Fig. 8.

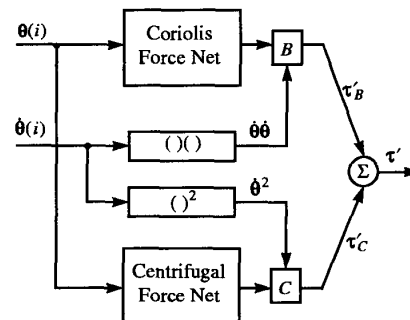


Fig. 5 Velocity network consisting of the centrifugal and Coriolis force networks.

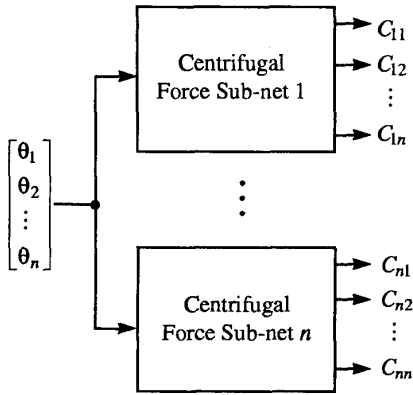


Fig. 6 Block diagram of the centrifugal network.

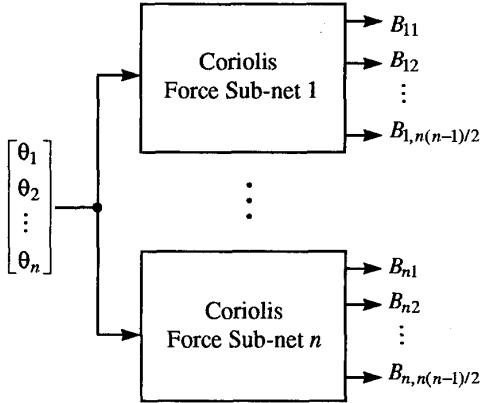


Fig. 7 Block diagram of the Coriolis network.

The input-output mapping relationship of the radial basis function (RBF) in this application is given by

$$\phi_l(\mathbf{y}; \bar{\mathbf{y}}_l, \sigma_l) = \exp\left\{-\frac{(\mathbf{y} - \bar{\mathbf{y}}_l)^T(\mathbf{y} - \bar{\mathbf{y}}_l)}{\sigma_l^2}\right\} \quad (7)$$

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 radial basis function,  $\sigma_l^2$  is the variance of the  $l$ th RBF. Therefore the output at the  $k$ th output node of the subnetwork  $N_{sub}(\cdot)$  is given by

$$[N_{sub}(\mathbf{y})]_k = \sum_{l=1}^N W_{kl} \phi_l(\mathbf{y}; \bar{\mathbf{y}}_l, \sigma_l), k = 1, \dots, m \quad (8)$$

where  $N$  is the number of RBFs that construct the sub-net and  $W_{kl}$  is the weight from  $l$ th RBF to  $k$ th output node.

#### 4. PROCEDURE OF THE SYSTEM IDENTIFICATION AND RESULTS

In this case, a three-link manipulator was considered for simplicity. The relevant parameters are given in Table 1.

Before carrying out the system identification studies, a P-D controller was designed to represent the servo controller of an actual manipulator, and the P-D gains were found for a given desired trajectory using an evolutionary algorithm.

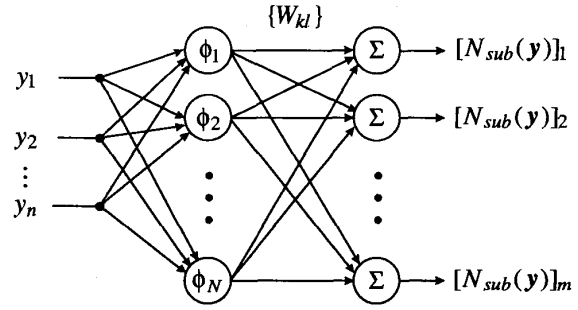


Fig. 8 Basic structure of the Radial Basis Function (RBF) neural network.

Table 1 Physical parameters for the manipulator.

Link	Mass [kg]	Length [m]
Link 1	6	0.2
Link 2	4	0.3
Link 3	3	0.1

The P-D controller can be given by

$$\tau = \mathbf{K}_p(\boldsymbol{\theta}_d - \boldsymbol{\theta}) + \mathbf{K}_d(\dot{\boldsymbol{\theta}}_d - \dot{\boldsymbol{\theta}}) \quad (9)$$

where  $\mathbf{K}_p$  and  $\mathbf{K}_d$  are proportional and derivative gain vectors respectively. An evolutionary algorithm was used to find the optimum gains of the P-D controller by minimizing the tracking error given by

$$\frac{1}{t_N} \sum_{i=1}^{t_N} \|\boldsymbol{\theta}_d(i) - \boldsymbol{\theta}(i)\| + \|\dot{\boldsymbol{\theta}}_d(i) - \dot{\boldsymbol{\theta}}(i)\| \quad (10)$$

where  $t_N$  is the total time span of the trajectory. The optimum gains found are given by  $\mathbf{K}_p = \text{diag}\{32.84, 42.05, 42.31\}$  and  $\mathbf{K}_d = \text{diag}\{42.44, 0.19, 4.86\}$ . The conditions of the evolutionary algorithm in this case were as follows: 100% mutation and 20% crossover was performed. A tournament size of 6 and 200 generations were adopted. The number of individuals was 60.

Then system identification simulation studies were carried out for three cases.

Case 1: Input torque information is known.

Case 2: PD gains are known.

Case 3: No information is available regarding the controller.

Then the weights of the RKGNN and the RBF parameters,  $\bar{\mathbf{y}}_l$  and  $\sigma_l$ , were found using an evolutionary algorithm so as to minimize the cost function given by

$$J = \frac{1}{t_N} \sum_{i=1}^{t_N} \|\boldsymbol{\theta}_d(i) - \boldsymbol{\theta}(i)\| + \|\dot{\boldsymbol{\theta}}_d(i) - \dot{\boldsymbol{\theta}}(i)\| + P \quad (11)$$

where  $t_N$  is the total time span of the trajectory,  $P$  is a penalty. If the minimum of the principle minors of the inertia matrix  $M(\boldsymbol{\theta})$  estimated by the network is negative,  $P$

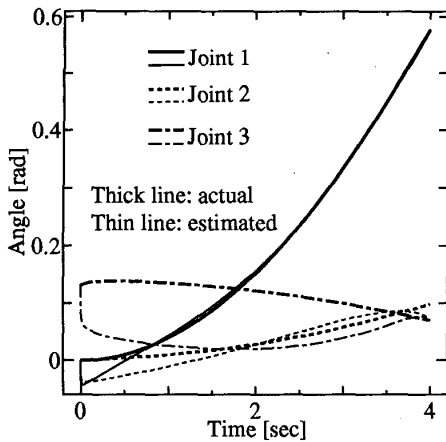


Fig. 9 Estimated angular trajectories of the manipulator in the case where the input torque is known.

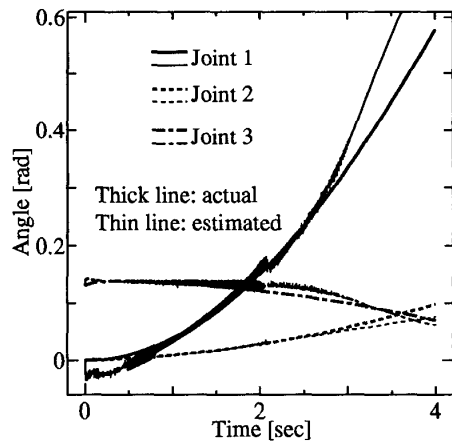


Fig. 11 Estimated angular trajectories of the manipulator in the case where the proportional and derivative gains of the controller are known.

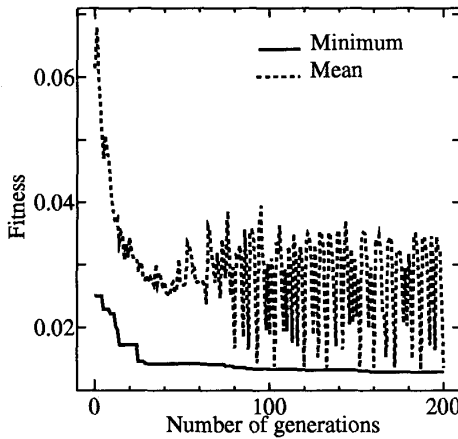


Fig. 10 The evolutionary history of the optimization of network parameters in the case where the input torque is known.

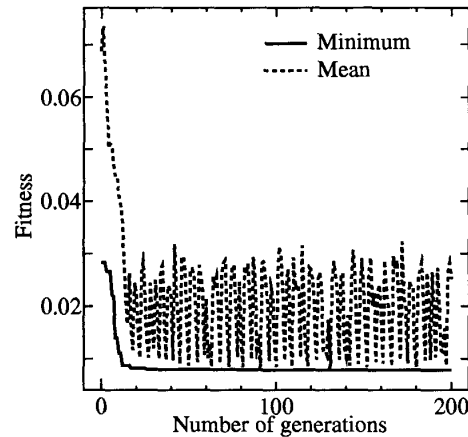


Fig. 12 The evolutionary history of the optimization of network parameters in the case where the proportional and derivative gains of the controller are known.

is given the negative value of the minimum of the principle minors of the inertia matrix  $M(\theta)$  estimated by the network. Otherwise  $P$  is set to zero. The reason to add this penalty is to make sure that the estimated inertia matrix is symmetric and positive definite. The conditions of the usage of evolutionary algorithm were as follows: 100% mutation and 20% crossover was employed. The tournament size was 6 and the number of individuals was 100. The number of elements in one individual varies depending on the case. In the cases 1 and 2, 99 elements and in case 3, 153 elements were in one individual. The algorithm was run for 200 generations for all cases.

#### System Identification Results: Case 1

The following Fig. 9 shows how the identified system is tracking the joint angular trajectories in comparison to the desired trajectories in the case where the input torque information is known. Figure 10 shows the corresponding evolutionary history of the optimization of parameters for the system.

#### System Identification Results: Case 2

The following Fig. 11 shows the results for the case where the gains of the proportional and derivative are known. Figure 12 shows the corresponding evolutionary history of the optimization of parameters for the system.

#### System Identification Results: Case 3

The following Fig. 13 shows the identified results for the case where no information of the controller is known. Figure 14 shows the corresponding evolutionary history of the optimization of parameters for the system.

Note that in Fig. 10, Fig. 12 and Fig. 14 that the mean fitness keeps adequate deviation from the minimum, throughout the evolutionary history. This property helps the evolutionary algorithm to search for better solutions without converging to local minima.

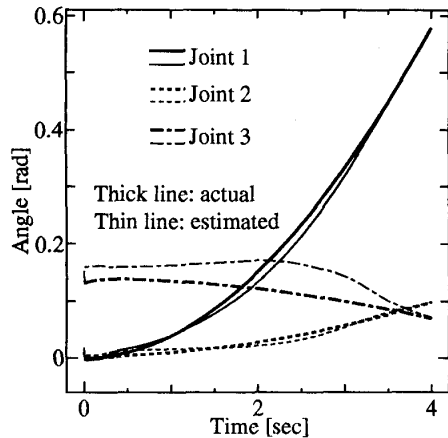


Fig. 13 Estimated angular trajectories of the manipulator in the case where no information of the controller is known.

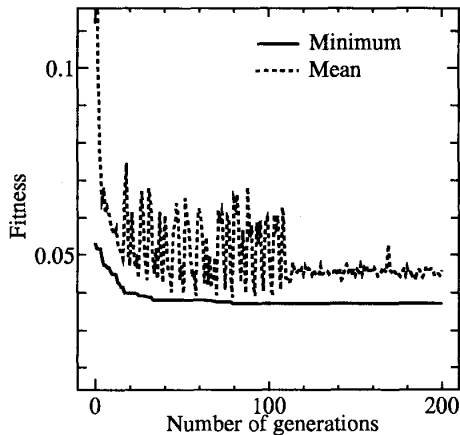


Fig. 14 The evolutionary history of the optimization of network parameters in the case where no information of the controller is known.

## 5. CONCLUSIONS

The Runge-Kutta-Gill Neural network (RKGNN) has been developed and it was applied for identifying the complex dynamics 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.

Simulation studies were carried out to identify the dynamics of the manipulator for three cases: the input torque sensor information is known; the gains of the PD controller are known; no information of the controller is known. The results are promising. The next step would be to design the computed torque controller using the identified the dy-

namics using the RKGNNs.

## 6. REFERENCES

- [1] J. J. Craig, *Introduction to Robotics: Mechanics and control*, 2nd edn, Reading, MA: Addison-Wesley, 1989.
- [2] 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.
- [3] H. Kawasaki and K. Nishimura, "Terminal Link Parameter Estimation of Robotic Manipulators," *IEEE Trans. on Robotics and Automation*, vol. 4, no. 5, 1988, pp. 485-490.
- [4] C.-J. Wu and C.-H. Huang, "Back-Propagation Neural Networks for Identification and Control of a Direct Drive Robot," *J. of Intelligent and Robotic Systems*, vol. 16, no. 1, 1996, pp. 45-64.
- [5] 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, 1998, pp. 294-307.
- [6] J. R. Noriega and H. Wang, "A Direct Adaptive Neural-Network Control for Unknown Nonlinear Systems and Its Application," *IEEE Trans. on Neural Networks*, vol. 9, no. 1, 1998, pp. 27-34.
- [7] A. R. Web and S. Shannon, "Shape Adaptive Radial Basis Functions," *IEEE Trans. on Neural Networks*, vol. 9, no. 6, 1998, pp. 1155-1166.
- [8] T. Nanayakkara, K. Watanabe and K. Izumi, "Evolving in Dynamic Environments Through Adaptive Chaotic Mutation," *Procs. of Fourth International Symposium on Artificial Life and Robotics*, vol. 2, 1999, pp. 520-523.
- [9] D. E. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning*, Reading, MA: Addison-Wesley, 1989.
- [10] D. B. Fogel, "An Introduction to Simulated Evolutionary Optimization," *IEEE Trans. on Neural Networks*, vol. 5, no. 1, 1994, pp. 3-14.
- [11] T. Bäck and H. P. Schwefel, "An Overview of Evolutionary Algorithms for Parameter Optimization," *Evolutionary Computation*, vol. 1, no. 1, 1993, pp. 1-23.
- [12] Y. Leung, "Degree of Population Diversity—A perspective on Premature Convergence in Genetic Algorithms and its Markov Chain Analysis," *IEEE Trans. on Neural Networks*, vol. 8, no. 5, 1997, pp. 1165-1175.