A Hybrid Intelligent Active Force Controller for Robot Arms Using Evolutionary Neural Networks

S.B. Hussein, H. Jamaluddin, M. Mailah
Universiti Teknologi Malaysia
Skudai, Johor Bahru 81310, Malayisa
E-mail: shamsul@fkm.utm.my

A.M.S. Zalzala
Heriot-Watt University
Edinburgh EH14 4AS, United Kingdom
E-mail: a.zalzala@hw.ac.uk

ABSTRACT

In this paper, we propose a hybrid intelligent parameter estimator for the active force control (AFC) scheme which utilizes evolutionary computation (EC) and artificial neural networks (ANN) to control a rigid robot arm. The EC part of the algorithm composes of a hybrid genetic algorithm (GA) and an evolutionary program (EP). The development of the controller is divided into two stages. In the first stage, which is performed off-line, the proposed EC algorithm is employed to evolve a pool of ANN structures until they converge to an optimum structure. The population is divided into different groups according to their fitness. The elitist group will not undergo any operation, while the second group, i.e. stronger group, undergoes the EP operation. Hence, the behavioral link between the parent and their offspring can be maintained. The weaker group undergoes a GA operation since their behaviors need to be changed more effectively in order to produce better offspring. In the second stage, the evolved ANN obtained from the first stage, which represent the optimum ANN structural design, is used to design the on-line intelligent parameter estimator to estimate the inertia matrix of the robot arm for the AFC controller. In this on-line stage, the ANN parameters, i.e. the weights and biases, are further trained using live data and back-propagation until a satisfactory result is obtained. The effectiveness of the proposed scheme is demonstrated through a simulation study performed on a two link planar manipulator operating in a horizontal plane. An external load is introduced to the manipulator to study the effectiveness of the proposed scheme.

Keywords: Evolutionary computation, genetic algorithms, evolutionary programming, neural networks, robotics, active force control.

1. INTRODUCTION

In general, robotic control can be divided into two main categories, i.e. position control and force control. Some of the popular classical methods in robot position control are on-off control, independent joint control and computed torque control. The two former methods suffer from overshoot problem and only applicable to relatively very slow operations [1], whereas the later method requires the estimated mathematical inverse dynamic model of the robot to operate successfully, which is almost impossible to obtain accurately [2].

In robot force control, the two most used methods are impedance force control (IFC) [3] and hybrid position-force control (HPFC) [4]. However, there are a few drawbacks in applying these two schemes. The performance of the IFC is dependent on how accurate the model of the robot is estimated. In the HPFC, the desired position and force have to be specified by the user and need to be updated when the robot works with a different environment (i.e. different external force) or when the robot internal frictional forces change.

Another robot force control method is active force control (AFC) [5]. AFC operates mainly on measured and estimated parameters hence lessening the computational burden. Some advantages of the AFC strategy is that it has a fast de-coupling property and it can be applied to variable loading conditions. The performance of AFC depends mainly on how accurate the inertia matrix of the robot arm is estimated. The inertia matrix can be estimated via crude approximation, look-up table, iterative learning technique or using ANN algorithms [6].

Since late 1980's, researchers have attempted to implement computational intelligence methods, i.e. ANN, EC, and fuzzy logic in robot control to either function as a robot controller itself, or as part of a controller system. Some of the works on the implementation of ANN in robot control can be found in [7], [8] and [9]. More recently, some researchers have incorporated GAs with classical controller such as PID or with ANN controllers [10].

In this paper, we utilize both EC and ANN to estimate the inertia matrix (IN) of the robot arm in an AFC approach. The EC comprises a GA and EP and is used during off-line stage to evolve a pool of ANN structures until they converge to an optimum one. The selected ANN structure is then used in the on-line stage. In this stage, the ANN is further trained using on-line data from the system. The on-line training is terminated when a satisfactory performance is obtained.

This paper is organized as follows. Section 2 describes the AFC strategy for robot control. In Section 3, a brief literature reviews on the evolutionary artificial neural network (EANN) is presented. The proposed EANN algorithm for the off-line stage is explained in section 4 and the proposed on-line intelligent AFC method is described in section 5. Simulation results of both off-line and on-line

0-7803-6375-2/00/$10.00 ©2000 IEEE.

training stages are shown in section 6. Finally, discussion and conclusion are given in section 7.

2. A STRATEGY FOR ACTIVE FORCE CONTROL

AFC is a control method first introduced by Hewit and Burdess [11], and is derived from the Newton's second law of motion for a rotating mass, i.e. $\sum T = I\alpha$, where T is the sum of all torques acting on the body, I is the moment of inertia, and α is the angular acceleration. For a robot system which has a serial configuration, the equation of motion becomes $T + Q = I(\theta)\alpha$, where T is the applied torque, Q is the disturbance torques, $I(\theta)$ is the mass moment of inertia of the robot arm and θ is the robot joint angle, and α is the angular acceleration of the robot arm.

The idea of the AFC approach is that if the value of disturbance torques Q' can be computed or measured within an acceptable accuracy, this value could be used to de-couple the actual disturbance torques Q from the applied torque. Hence, the system stays stable even under variable external forces. The estimated value of the disturbance torque can be calculated as $Q' = I'\alpha' - T'$, where the superscript ' denotes a measured or computed quantity. The applied torque T' can be measured by using a current sensor and α' can be measured by using e.g. an accelerometer. On the other hand, I' can be obtained by several means such as a simple estimation or by just assuming a perfect model. The more recent literature adopted iterative learning techniques and ANN to estimate the value of I' [12].

Figure 1 shows the schematic diagram of the AFC method applied to a robot arm together with the resolved motion acceleration controller [13]. There are two types of controllers employed, i.e. the PD resolved-motion acceleration control (PD-RMAC) and the AFC. The PD-RMAC is employed to calculate the reference acceleration command θ_{ddref}. This is followed by a decoupling transfer function $(W')^{-1} = IN/K_m$ where IN is the estimated inertia matrix K_m is the motor constant. This will produce the control signal I_c. The calculation involves inverse kinematics of the robot arm. This is fed into the AFC loop where the actual disturbance compensation is taking place assuming that both the acceleration and the torque vectors are suitably measured.

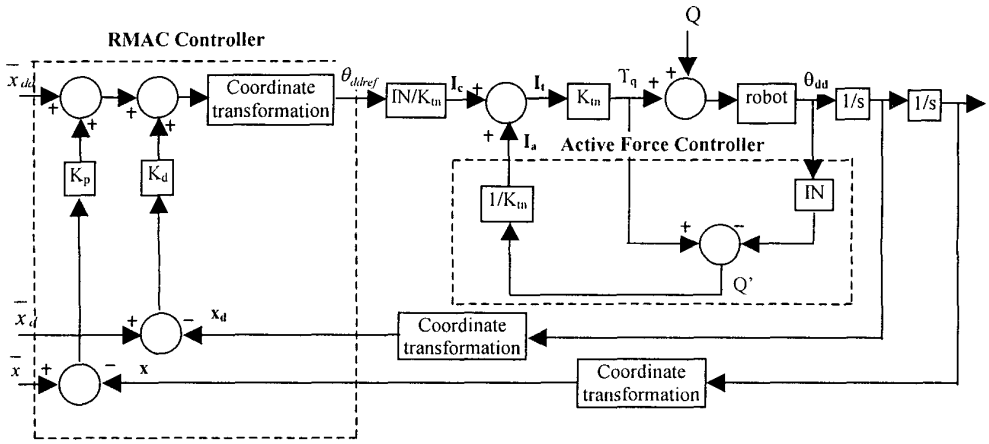


Fig. 1: The block diagram of the active force controller applied to the robot force control

The estimated disturbance torques Q' can be computed by the equations $Q' = IN\ \theta_{dd} - T_q$ and $T_q = K_m I_t$, where θ_{dd} is the acceleration signal, IN is the estimated inertia matrix and T_q is the applied control torque, K_m is the motor constant and I_t is the controlled current. I_t is the sum of I_c and I_a where I_a is the estimated current signal representing the disturbance. The estimation of inertia matrix of the robot arm has been done previously by using traditional techniques such as crude estimation technique, as well as intelligent techniques such as employing ANN in off-line training environment [13]. The main problem in AFC is to obtain a good estimation of IN. In this paper, we propose an evolutionary ANN algorithm to assist the AFC in estimating the value of IN.

3. A BRIEF REVIEW OF EVOLUTIONARY NEURAL NETWORKS

Evolutionary artificial neural network (EANN) is an area of popular interest with many publications aiming to improve the performance of ANN in terms of its generalization capabilities and efficiency, i.e. time and space complexity. Whitley [13] and Fogel [14] have reported some of the early works in this area, while good reviews have been reported by Yao [15, 16].

In general, there are three approaches: the evolution of connection weights, of architectures, and of learning rules.

Some researchers have used a fixed architecture and evolved only the connection weights of the architecture [17], some evolved both the architecture and the connection weights [18][19], and others evolved only the architecture of the network and employed back-propagation or other methods to train the weight [20].

In the evolution of connection weights, two popular representation schemes are binary strings and real numbers. The problem with binary representation scheme is that it needs excessively large number of bits in order to get a precise representation, and the training may take a considerably long time. To overcome this, real numbers representation is introduced which conflicts with applying the classical techniques of crossover. Herrera [21] described and analyzed the real number representation and its operators' capabilities. The connection weights can be evolved via a GA approach which involves two kinds of search operators, crossover and mutation. However, weights can be evolved using EP hence skipping the use of a crossover operator. In evolving architectures, the chromosome representation can be divided into two different schemes, the direct encoding (DE) and indirect encoding (IDE). In DE, the number of neurons along with the connection and the number of hidden layers are directly encoded as a binary string representation. The drawback of this type of encoding is that it is only suitable for handling relatively small number of neurons since large EANN need large representation matrices. On the other hand, the IDE is a compact representation of the EANN architectures, where small chromosomes can grow massive neural network by some development rules during chromosome decoding. Some of the earlier work on this was reported by Kitano [22], while a comparison of both schemes was reported quite recently [23]. The two popular evolutionary algorithms used to evolve the network architecture are GA and EP.

4. THE PROPOSED HYBRID INTELLIGENT AFC CONTROLLER SCHEME

The proposed hybrid intelligent AFC scheme involves two stages of operation: (1) the off-line training stage, which evolves the ANN structure for the AFC loop (see figure 4), and (2) the on-line training, which trains the ANN parameters until a satisfied robot tracking performance is obtained.

4.1 Off-line training stage: evolving the ANN structure

The purpose of executing the off-line training stage is to search for the optimum ANN structure for the AFC problem mentioned earlier. This is done by implementing both GA and EP methods to evolve a pool of ANN structures until it converges. Earlier report [15,16] indicated that in order to maintain the behavioral link between the offspring and their parents, the use of crossover operator should be avoided since it can destroy both parent networks. Therefore, an EP method, which emulates the Lamarckian evolution theory and employs mutation only, is preferred. Examples of such mutation operators are neuron deletion and neuron addition.

However, in the previous work [24], a new EANN algorithm, which combined the GA and EP operators, is presented. It has been shown that the hybrid GA and EP approach can provide fast convergence rate, since GA can explore wider area of search in shorter number of evolution, while EP can maintain the behavioral link of the selected offspring after the competition stage between the GA and EP offspring.

In this paper, a modified version of hybrid GA and EP EANN algorithm is proposed. In this algorithm, we divide the population into three different groups, namely elitist, strong and weak group. Since GA has the capability of distorting the ANN behavioral link between the offspring and their parents, it is a good method to be applied to the weak group of the population. By applying a GA to a weak group, the convergence rate of the evolution can be increased. The crossover operator in GA will break the behavioral link between the offspring and their weak parents and thus producing stronger. In contrast, the EP algorithm is applied to the stronger groups of the population consisting of stronger individuals. Only mutation operators, i.e. neuron deletion and neuron addition, are used thus preserving the behavioral link. The proposed algorithm is shown in figure 2. The strongest individual stays in the elitist group and maintained untouched by GA or EP operator until a stronger offspring is created and win over the elitist. This will force down the loosed elitist to joint in the second group (strong group) and be subjected to EP operation. By this grouping procedure, the time taken for one evolution to finish will be shorter compared to the previous algorithm [24].

4.1.2 The chromosome representation

Binary string representation is used to represent the structure of the multi-layer feed-forward (MLFF) network. The structural parameters evolved are the number of neuron in the hidden layer, number of hidden layer and types of transfer function in the hidden layer. Figure 2 shows an example of the chromosome representation in a binary string and its corresponding ANN structure. In the previous work in [24], integer representation is used to represent the ANN structures. However, integer representation is less efficient for evolving the ANN structure. This is mainly due to the unsuitable real-numbered crossover and mutation methods been adopted to the integer numbered chromosomes. Further more, the binary crossover and mutation methods are proven to be efficient especially for a short chromosome length such as in this application.

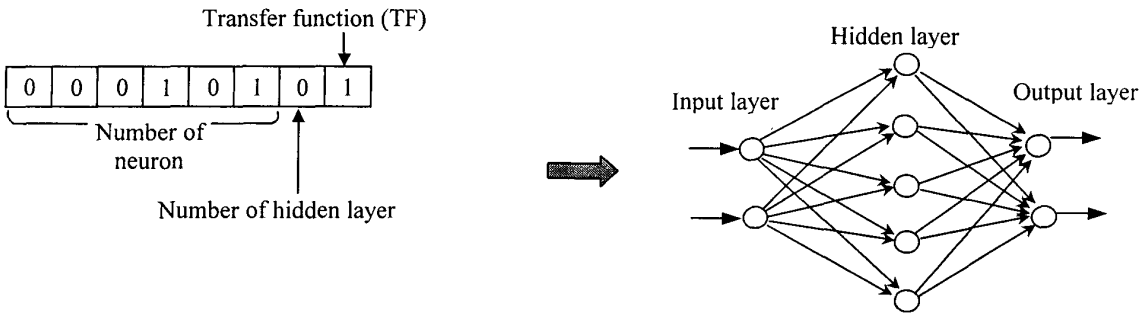


Fig. 2: Binary string representation of ANN structure (top) and its corresponding ANN (bottom)

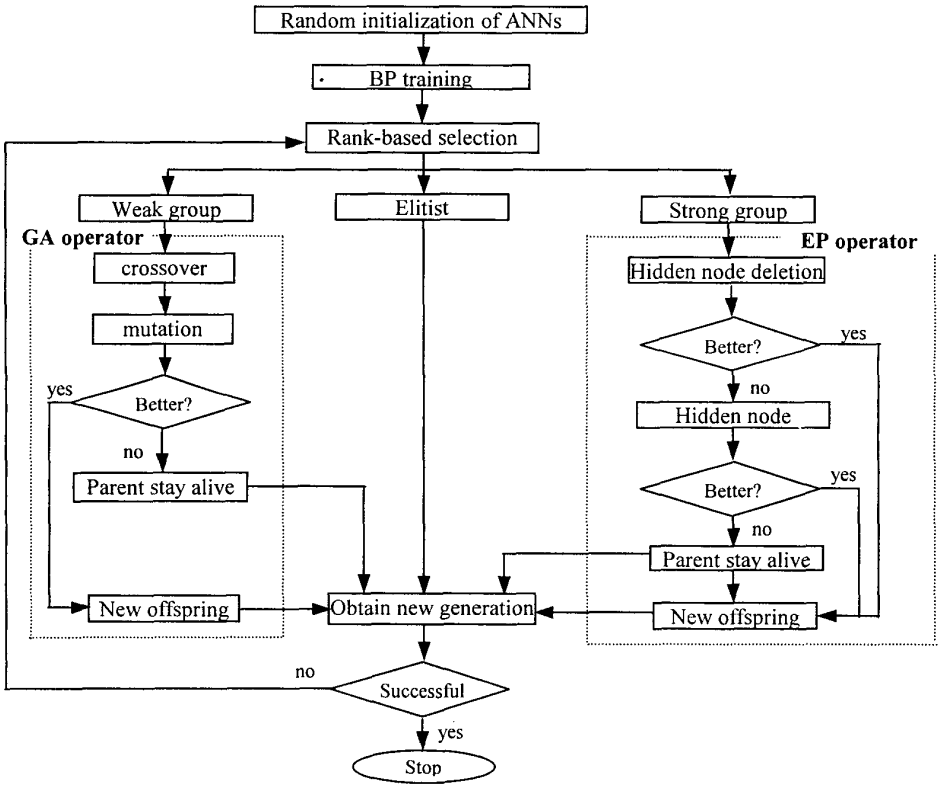


Fig. 3: Flow chart of the major steps in the proposed EANN algorithm

4.1.2 Major steps in the proposed EANN algorithm

The major steps in the proposed EANN algorithm is shown if figure 2. The explanation of the steps is as follows:

1) Randomly generate M number of networks in binary representation within the specified range, which represents the number of neuron, number of hidden layer and type of transfer function (TF) at the hidden layer, as depicted in figure 3.
2) Train each networks using backpropagation (BP) algorithm, for a reasonable number of epoch and evaluate the sum-squared error of each networks and calculate their cost function accordingly.
3) Divide the networks into three different groups, i.e. elitist, strong and weak group, according to their cost function. This is done using the rank-based selection method.
4) Apply GA operator (crossover and mutation) onto the individuals in the weak group. The crossover method employed is the multi-point crossover, and the mutation method used is the simple flip-over method. Then, compare the distorted offspring with their parents. If the offspring is better, replace the parent with the offspring. If not, the parent stays alive for the next generation.
5) Apply the EP operator (mutation only) onto each individuals in the strong group. The mutation steps are as follows:
   a) Delete two neurons from each network and evaluate the offspring's cost function. If the offspring is better than the parent, the offspring replaces the parent.
   b) If not, add two neurons onto the network, and evaluate again. If the offspring is better than the parent, the offspring replaces the parent. If not, the parent stays alive for the next generation.
6) Combine the network obtained from step 5 and 6 and the elitist network to get new generation ANN.
7) Repeat step 4 to 6 for a specified number of generations.

4.2 On-line stage: on-line ANN parameter training

In the previous work [24], the algorithm presented acquires only off-line ANN parameter training by backpropagation and parameter tuning by fast evolutionary programming (FEP) algorithm. This off-line training method has certain limitation issues such as the learning data set has to be updated whenever the robot dynamic properties change. This can be due to internal factors such as friction property, or external factors such as unexpected external disturbances. The other limitation of the off-line training is the accuracy of the training data itself, which usually consist noises. Therefore, an on-line parameter training scheme is proposed to tackle these problems.

The ANN structure selected from the off-line stage is used to design the on-line ANN-AFC controller for the robot force control. The schematic diagram of the controller is shown in figure 4. As shown in the figure, the thick line indicates the implementation of on-line ANN subsystem to estimate the IN of the robot arm for the AFC loop. In this stage, the ANN is further trained on-line by using real input data measured from the robot arm. As the training session progresses, the value of ANN parameters, i.e. weight and bias, are adjusted to reduce the position error of the robot arm. The ANN is trained using BP algorithm.

An external disturbance force (Q) is introduced to the robot arm end-effector. The disturbance force is a sinusoidal variable load with certain range of magnitude and frequency. This is to study the effectiveness of the ANN-AFC controller scheme. The training of the ANN is stopped when appropriate result is obtained.

5 SIMULATION RESULTS

The EANN algorithm in the off-line stage is programmed using MATLAB [25]. Some limitations have been imposed on the EANN in the simulation. The limitations are (a) type of ANN is MLFF with full connectivity between neurons, (b) largest number of neuron in the hidden layer is 32 and one hidden layer, and (c) types of TF at hidden layer is either tan-sigmoid or log-sigmoid.

The on-line ANN-AFC scheme is programmed using SIMULINK. The robot for the ANN-AFC simulation is a rigid two-arm robot which movement is restricted to horizontal axis. The robot parameters and the controller gains (K_p and K_d) are obtained from the previous work [12]. The task given to the robot is to move in a circular trajectory with specified radius while subjected to an external disturbance force (Q) at the end-effector. The value of the disturbance force is made constant with the magnitude of 15N through out the simulation period. The sum-squared of the tracking error (SSE) is monitored to study the performance of the proposed scheme.

5.1 Result of the EANN (off-line stage)

Figure 5 shows the results of the EANN evolution over twenty generations. The graph shows the total cost function of the population decreasing sharply in the first three generations, and converging after 10 generations. The number of neuron in the hidden layer is also converged to between five to seven. Six neurons is the optimum number of neuron since half of the population converges to this solution. The transfer function of the hidden layer converges into a log-sigmoid.

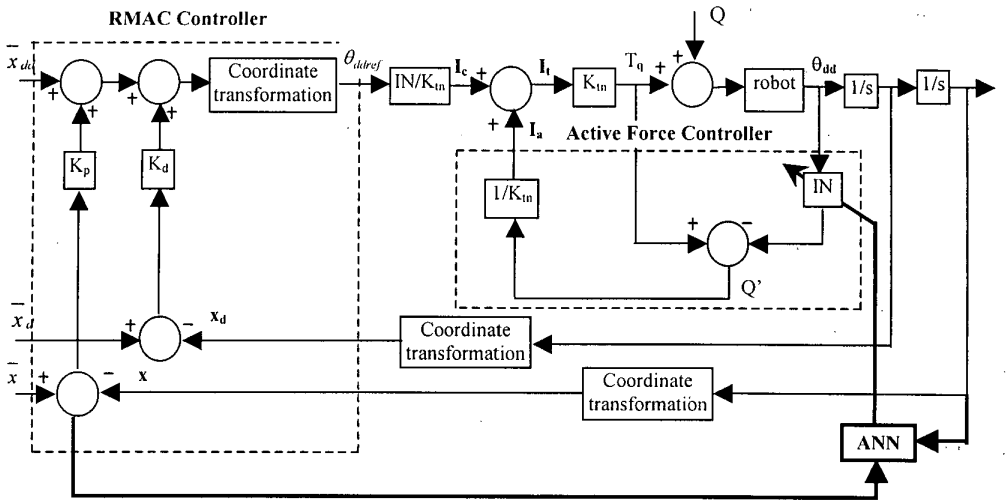
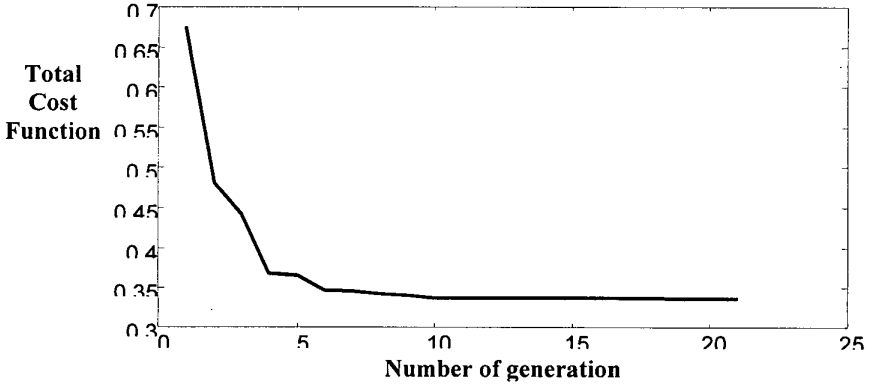


Fig. 4: The schematic diagram of the on-line ANN-AFC controller for robot force control

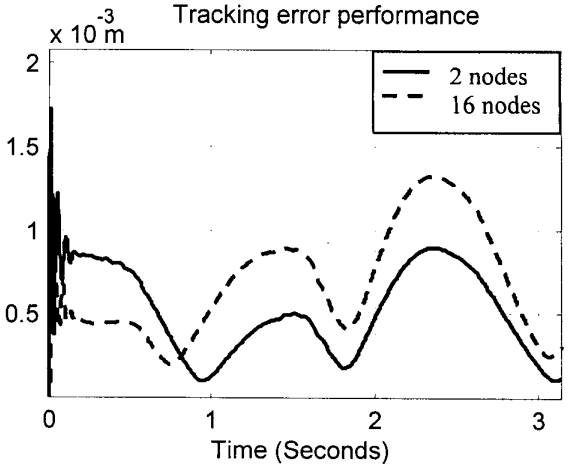


	No. of neuron	Transfer function
Ind. 1	5	logs
Ind. 2	7	tans
Ind. 3	6	logs
Ind. 4	6	logs
Ind. 5	6	logs
Ind. 6	5	logs

Fig. 5: The result of total cost function of the population for 20 generations (left) and final population structures

Figure 6 shows the tracking error performance of the off-line ANN-AFC scheme for different number of nodes. It shows that by selecting different number of nodes, the tracking error performance of the robot arm is changed. Therefore, during offline stage, the network needs to be evolved to obtain the optimum ANN structure.

Fig. 6: The tracking error of the robot arm for different number of nodes



5.2 Results of ANN-AFC scheme (on-line stage)

The ANN structure obtained in the off-line stage is used to design the on-line ANN for the ANN-AFC controller. In this stage, the ANN is trained for 15 seconds and it is shown in figure 7 that the tracking error of the robot arms is considerably reduced.

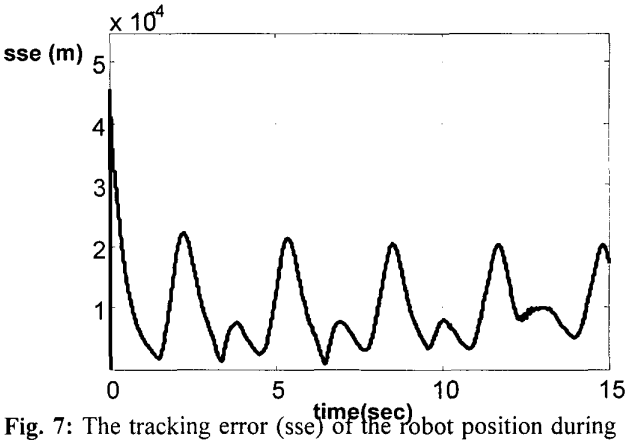


Fig. 7: The tracking error (sse) of the robot position during on-line training session

6 CONCLUSIONS

It is found out from the result of the simulation that the proposed EANN algorithm did converge to the optimum ANN structure. However, the final generation shows that there are three optimum solutions to the number of neuron in the hidden layer i.e. five, six and seven neurons. The variation in the solutions is due to the mutation operator in the EP algorithm, which deletes or adds two neurons at a time. A better convergence can be obtained if the neuron deletion and addition is reduced to one neuron at a time. However, this will need longer generation before it can converge.

The ANN-AFC scheme shows an impressive improvement after 15 seconds of on-line training. This shows that the proposed scheme performs very well under constant external force condition. However, further investigations are required to study whether or not the scheme's performance is maintained if the robot arm is subjected to variable external loading conditions. It is found out that the on-line training can perform better than the off-line training. This is true since the off-line training data contain some unknown noise, which can affects the ability of the network generalization.

In conclusion, it is shown that the combination of EC and ANN can help to produce a better intelligent robot controller scheme. EC is a good tool for optimization the ANN structure and ANN is a good tool for assisting the on-line parameter estimation for AFC controller. This work also showed that the EP and GA can be used together to evolve the ANN structures. Future works should be concentrated on applying variable loading conditions to the robot arm to study the robustness of the scheme. Some improvement should be studied such as using adaptive learning rate and momentum rate for the on-line ANN-AFC scheme.

REFERENCES

1. H. Asada and J.J.E. Slotine, *Robot Analysis and Control*, John Wiley and Son, 1986
2. F.L. Lewis, C.T. Abdallah, D.M. Dawson, *Control of Robot Manipulators*, Ed. John Griffin, Macmillan Pub., New York, 1993
3. N. Hogan, Impedance control: An approach to manipulator, part i, ii, iii, *ASME Journal of Dynamic Systems, Measurement, and Control*, vol.3,pp. 1-24, 1985
4. M. H. Raibert and J. J. Craig, Hybrid Position/Force Control of Manipulators, *Trans. ASME J. Of Dynamic Systems, Measurement, and Control*, 102, June 1981, pp. 126-133
5. J.R. Hewit and J.S. Burdess, Fast Dynamic Decoupled Control for Robotics Using Active Force Control, *Mechanism and Machine Theory*, Vol. 16, No.5, 1981, 535-542
6. M. Mailah, Intelligent Active Force Control Using Neural Network and Iterative Learning Algorithms, Ph.D Thesis, University of Dundee, 1998
7. D. T. Pham and S. J. Oh, Adaptive Control of a Robot Using Neural Networks, *Robotica*, vol. 12, 1994, pp. 553-561
8. S. Jung, *Neural Network Controllers for Robot Manipulators*, Ph.D thesis, University of California Davis, 1996
9. A. Rana and A. M. S. Zalzala, A Neural Network Based Collision Detection Engine for Multi-Arm Robotic Systems, *Int'l Journal of Intelligent Control Systems*, vol. 2, No. 4, 1998, pp. 531-558

10. N. Chaiyaratana and A. M. S. Zalzala, Hybridisation of Neural Networks and Genetic Algorithms for Time-Optimal Control, *Congress on Evolutionary Computation*, vol. 1, July 1999, pp. 389-396
11. J.R. Hewit and J.S Burdess, An Active Method for the Control of Mechanical Systems in The Presence of Unmeasurable Forcing, *Transactions on Mechanism and Machine Theory*, vol. 21, No. 3, 1986, pp. 393-400
12. M. Mailah, A simulation Study on the Intelligent Active Force Control of a Robot Arm using Neural Network, *Jurnal Teknologi UTM*, No.30(D), June 1999, pp. 55-78
13. D. Whitley, T. Starkweather and C. Bogart, Genetic Algorithms and Neural Networks: Optimizing Connection and Connectivity, *Parallel Computing*, vol.14 No.3, 1990, pp. 347-361
14. D. B. Fogel, L.J. Fogel, and V. W. Porto, Evolving Neural Networks, *Biological Cybernetics*, 63, 1990, pp. 487-493
15. X. Yao, A Review of Evolutionary Artificial Neural Networks, *International Journal of Intelligent Systems*, vol.8, No.4, April 1993, pp. 539-567
16. X. Yao, A New Evolutionary System for Evolving Artificial Neural Networks, *IEEE Transactions on Neural Networks*, vol. 8, No. 3, May 1997, pp. 694-713
17. D. Fogel, Using evolutionary programming to create networks that are capable of playing tic-tac-toe, *in Proceedings of IEEE International Conference on Neural Networks*, San Francisco:IEEE, 1993, pp. 875-880
18. D. Dasgupta and D. McGregor, Designing application specific neural networks using the structured genetic algorithm, *in Proceedings of COGA NN-92 – IEEE International Workshop on Combinations of Genetic Algorithms and Neural Networks*, Baltimore: IEEE,1992, pp. 87-96
19. K.F. Man, K.S. Tang and S. Kwong, *Genetic Algorithms*, Springer, London, 1999, pp. 155-172
20. S. Oliker, M. Furst, and O. Maimon, A distributed genetic algorithm for neural network design and training, *Complex System*, vol. 6, no. 5, 1992, pp. 459-477
21. F. Herrera, M. Lozano, and J.L. Verdegay, Tackling Real-Coded Genetic algorithms: Operators and Tools for Behavioural Analysis, *Artificial Intelligence Review*, 12, 1998, pp. 265-319
22. H. Kitano, Designing Neural Networks using Genetic Algorithm with Graph Generation System, *Complex Systems*, vol.4, 1990, pp. 461-476
23. A.A. Siddiqi and S.M. Lucas, A Comparison of Matrix Rewriting versus Direct Encoding for Evolving Neural Networks, *Proceedings of IJCNN, FUZZ-IEEE,ICEC*, 1998, pp. 392-397
24. S.B. Hussein, A.M.S. Zalzala, H. Jamaluddin, and M. Mailah, An Evolutionary Neural Network Controller for Intelligent Active Force Control, *Evolutionary Design and Manufacturing*, Ed. I.C. Parmee, Springer-Verlag, London, 2000
25. Matlab, *The Mathworks Inc.*, ver. 5.3.0.10183 (R11), Jan. 1999