TRAJECTORIES GENERATION USING ARTIFICIAL NEURAL NETWORKS

Luiz Eduardo Nicolini do Patrocínio Nunes, luiz@unitau.br
Valesca Alves Correa, valesca@unitau.br
José Rui Camargo, rui@unitau.br
Carlos Alberto Chaves, carlosachaves@yahoo.com.br
Universidade de Taubaté – Rua Daniel Danelli, S/N, Taubaté, SP, CEP: 12060-440

Abstract. *The inverse kinematics determines the joint angles that result in the desired position of the manipulator's end-effector with regard to the reference coordinated system. The inverse kinematics solution is difficult a time that the mapping between the Cartesian space and the joint space is nonlinear and involves equations that can have multiple solutions. This work presents the application of a neural network with radial basis function (RBF) for the inverse kinematics solution of a robotic manipulator with three degrees of freedom. For a good learning generalization in the training phase of RBF network, some initial and final points had been initially generated inside of the manipulator's work volume. In order to prevent extreme angular oscillations, the angles for each one of the n generated points, they had been optimized by genetic algorithms. In accordance with the cartesian coordinate (x, y) supplied, the RBF neural network implemented in this work supplied angles that it had presented very next to the desired values.*

Keywords*: neural network, inverse kinematics, genetic algorithms, robotic manipulator*

1. INTRODUCTION

The fundamental problems of robot kinematics are the computation of the forward and inverse mappings between joint space and Cartesian task space. The former, mapping from joint space to Cartesian task space, is called forward kinematics and the latter, i.e. mapping from Cartesian task space to joint space, is called inverse kinematics (Craig, 1989). In robot control applications, the inverse kinematics is more important since, by default, robot tool trajectories are usually specified in terms of Cartesian task space coordinates. However, some actuators may receive commands in the joint space coordinates only. In this situation, the trajectories must be transformed to the joint space before the robot is controlled. The inverse kinematics of a serial robot is more complex than the forward kinematics since the inverse kinematics equations are more complex and nonlinear. In this work, inverse kinematics solution using neural networks is presented, some points in the work volume of manipulator are taken to use in the cubic path planning to generate the (θ_1, θ_2, θ_3) joint angles according to different (x, y) Cartesian coordinates. To prevent extreme angular oscillations from initial to final position, genetic algorithms were used to optimize both angular displacement and positional error.

2. ROBOTIC MANIPULATOR

In this work was considered a planar robotic manipulator with three degrees of freedom (Fig. 1). The joint angles θ_1, θ_2 and θ_3 can vary between -π and π. The lengths of the links l_1, l_2 and l_3 are 10 cm.

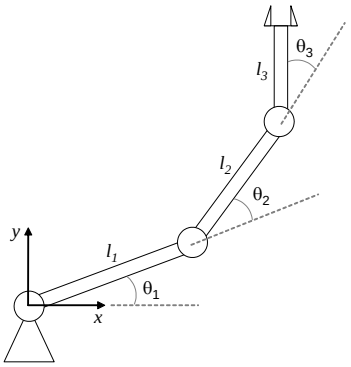

Figure 1. Planar Robotic Manipulator

The Direct Kinematics for this robotic manipulator is given by:

$$\begin{Bmatrix} x \\ y \end{Bmatrix} = \begin{Bmatrix} l_1 \cos(\theta_1) + l_2 \cos(\theta_1 + \theta_2) + l_3 \cos(\theta_1 + \theta_2 + \theta_2) \\ l_1 \sin(\theta_1) + l_2 \sin(\theta_1 + \theta_2) + l_3 \sin(\theta_1 + \theta_2 + \theta_2) \end{Bmatrix} \quad (1)$$

3. RBF NETWORK MODEL

The Radial Basis Function network (RBF) has a feed forward structure consisting of a single hidden layer of J locally tuned units, which are fully interconnected to an output layer of L linear units (Oyama and Tachi, 2000). All hidden units simultaneously receive the n-dimensional real valued input vector X (Fig. 2).

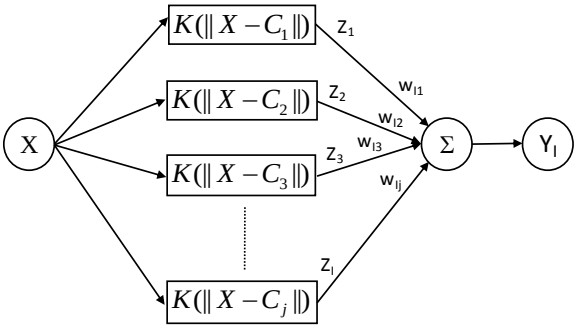

Figure 2. Radial basis function neural network

The main difference from that of Multi Layer Perceptron (MLP) is the absence of hidden-layer weights. The hidden-unit outputs are not calculated using the weighted-sum mechanism/sigmoid activation; rather each hidden-unit output Z_j is obtained by closeness of the input X to an n-dimensional parameter vector μ_j associated with the jth hidden unit. The response characteristics of the jth hidden unit (j = 1, 2, …, J) is assumed as,

$$Z_j = K\left(\frac{\|X-\mu_j\|}{\sigma_j^2}\right) \tag{2}$$

where K is a strictly positive radially symmetric function (kernel) with a unique maximum at its 'centre' μ_j and which drops off rapidly to zero away from the centre. The parameter σ_j is the width of the receptive field in the input space from unit j. This implies that Z_j has an appreciable value only when the distance $\| X-\mu_j \|$is smaller than the width σ_j. Given an input vector X, the output of the RBF network is the L-dimensional activity vector Y, whose lth component (l = 1, 2 … L) is given by:

$$Y_l(X) = \sum_{j=1}^{J} w_{l_j} Z_j(X) \tag{3}$$

For l = 1, mapping of eq. (1) is similar to a polynomial threshold gate. However, in the RBF network, a choice is made to use radially symmetric kernels as 'hidden units'. RBF networks are best suited for approximating continuous or piecewise continuous real-valued mapping $f : R^n \rightarrow R^L$, where n is sufficiently small. These approximation problems include classification problems as a special case. From equations (2) and (3), the RBF network can be viewed as approximating a desired function $f(X)$ by superposition of non-orthogonal, bell-shaped basis functions. The degree of accuracy of these RBF networks can be controlled by three parameters: the number of basis functions used, their location and their width. In the present work we have assumed a Gaussian basis function for the hidden units given as Z_j for j = 1, 2, … J, where:

$$Z_j = exp\left(-\frac{\|X-\mu_j\|^2}{2\sigma_j^2}\right) \tag{4}$$

and μ_j and σ_j are mean and the standard deviation respectively, of the jth unit receptive field and the norm is the Euclidean.

3.1. Training of RBF neural networks

A training set is an m labeled pair $\{X_i, d_i\}$ that represents associations of a given mapping or samples of a continuous multivariate function. The sum of squared error criterion function can be considered as an error function E to be minimized over the given training set. That is, to develop a training method that minimizes E by adaptively updating the free parameters of the RBF network. These parameters are the receptive field centres μ_j of the hidden layer Gaussian units, the receptive field widths σ_j, and the output layer weights (w_{ij}). Because of the differentiable nature of the RBF network transfer characteristics, one of the training methods considered here was a fully supervised gradient-descent method over E. In particular, μ_j, σ_j and w_{ij} are updated as follows:

$$\Delta\mu_j = -\rho_\mu \nabla_{\mu_j} E \tag{5}$$

$$\Delta\sigma_j = -\rho_\sigma \frac{\partial E}{\partial \sigma_j} \tag{6}$$

$$\Delta w_{ij} = -\rho_w \frac{\partial E}{\partial w_{ij}} \tag{7}$$

where ρ_μ, ρ_σ, and ρ_w are small positive constants. This method is capable of matching or exceeding the performance of neural networks with back-propagation algorithm, but gives training comparable with those of sigmoidal type of Feed Forward Neural Network (FFNN).

The training of the RBF network is radically different from the classical training of standard FFNNs. In this case, there is no changing of weights with the use of the gradient method aimed at function minimization. In RBF networks with the chosen type of radial basis function, training resolves itself into selecting the centres and dimensions of the functions and calculating the weights of the output neuron. The centre, distance scale and precise shape of the radial function are parameters of the model, all fixed if it is linear. Selection of the centres can be understood as defining the optimal number of basis functions and choosing the elements of the training set used in the solution. It was done according to the method of forward selection. Heuristic operation on a given defined training set starts from an empty subset of the basis functions. Then the empty subset is filled with succeeding basis functions with their centres marked by the location of elements of the training set; which generally decreases the sum-squared error or the cost function. In this way, a model of the network constructed each time is being completed by the best element. Construction of the network is continued till the criterion demonstrating the quality of the model is fulfilled. The most commonly used method for estimating generalization error is the cross-validation error (Venkatesan and Anitha, 2006).

4. CUBIC TRAJECTORY

This section investigates, in a constant time, the trajectory problem of the manipulator from an initial point until a final point in the Cartesian space. In the search of a function for each joint between the initial position, θ_0, and the final position, θ_f, the set of angles can be calculated, on the basis of the position and final orientation of the manipulator, through the use of its kinematics equations. It is necessary four restrictions in the function $\theta(t)$ that belong to the joints. Here, $\theta(t)$ represents the angular position in the instant of time t (Köker et. Al., 2004).

$$\begin{aligned} \theta(0) &= \theta_0 \\ \theta(t_f) &= \theta_f \end{aligned} \tag{8}$$

Two additional restrictions come from a function that is continuous in the point of the joint speed, i.e. the speed will be zero in the initial and final points of the joint.

$$\begin{aligned} \dot{\theta}(0) &= 0 \\ \dot{\theta}(t_f) &= 0 \end{aligned} \tag{9}$$

These four conditions can be provided for a polynomial of third order. Whereas a cubical polynomial has four coefficients, then, a cubical trajectory can be written as:

$$\theta(t) = a_0 + a_1 t + a_2 t^2 + a_3 t^3 \tag{10}$$

The speed and the acceleration of the joint along the trajectory are given respectively by:

$$\begin{aligned} \dot{\theta}(t) &= a_1 + 2a_2 t + 3a_3 t^2 \\ \ddot{\theta}(t) &= 2a_2 + 6a_3 t \end{aligned} \tag{11}$$

If the Eq. (10) and (11) are combined, four equations with four unknown quantity are gotten:

$$\begin{aligned} \theta_0 &= a_0 \\ \theta_f &= a_0 - a_1 t_f + a_2 t_f^2 + a_3 t_f^3 \\ 0 &= a_0 \\ 0 &= a_1 + 2a_2 t_f + 3a_3 t_f^2 \end{aligned} \tag{12}$$

Finally, with solution of the equations above, the coefficients are found as follow:

$$\begin{aligned} a_0 &= \theta_0 \\ a_1 &= 0 \\ a_2 &= \frac{3}{t_f^2}(\theta_f - \theta_0) \\ a_3 &= \frac{-2}{t_f^3}(\theta_f - \theta_0) \end{aligned} \tag{13}$$

Using the equations above, the Eq. (14) is derived and used to compute the function of reference position and speed:

$$\theta_i(t) = \theta_{i0} + \frac{3}{t_f^2}(\theta_{if} - \theta_{i0})t^2 - \frac{2}{t_f^3}(\theta_{if} - \theta_{i0})t^3 \quad i = 1,..,m \text{ (12)} \tag{14}$$

5. GENETIC ALGORITHMS

The Genetic Algorithms (GAs) constitute one technique of search and optimization, highly parallel, inspired in Darwin's principle of evolution. The natural principles, on which the GA's were inspired, are simple. The selection principle privileges the most apt individuals and, therefore, with more probability of reproduction. The individuals with more descendants have more chances of transmitting their genetic codes to the next generations (Michalewicz, 1994). Such genetic codes constitute the identity of each individual and are represented in the chromosomes. These principles are emulated in the construction of computing algorithms that search the best solution for one determined problem, through the evolution of populations of codified solutions through artificial chromosomes. The components of a GA include: inicialization, selection, crossing and mutation as illustrates the Fig. 3 (Kalra et. al., 2003):

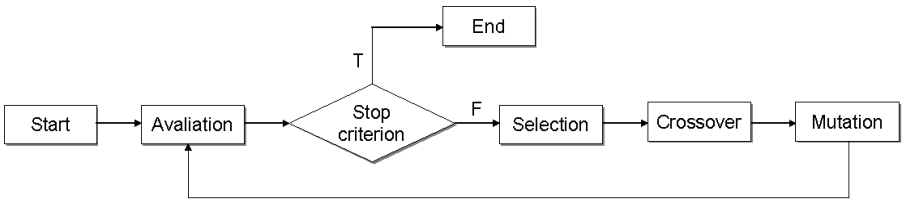

Figure 3. Component of a GA

5.1. Individual representation

The success of a genetic algorithm for a specific problem of optimization depends on the representation of an individual in the population (Kalra et. al., 2003). Each possible solution in the search space is represented by a sequence of symbols *s* generated from an alphabet (binary or real). Each *s* sequence corresponds to a chromosome and each element of *s* is equivalent to a gene. For a robotic manipulator, the individuals in a population can be represented, with real codification, through the joint angles: $\{\theta_1\ \theta_2\ \theta_3\}$. The real codification was chosen to avoid succeeding conversions of the binary code or gray for real values, saving, thus, computing time.

5.2. Inicialization

In the inicialization process, a population of chromosomes is generated randomly. The size of the population affects the efficiency and the performance of the GA (Goldberg, 1989). A population of small dimension can take the GA to converge quickly to a maximum local, while a very big population, damages the computing performance of the algorithm. The initial population for a robot with three degrees of freedom is generated randomly respecting the inferior (*L*) and superior (*U*) limits of each joint variable:

$$\theta_i^L \leq \theta_i \leq \theta_i^U \qquad i = 1,2,3 \tag{15}$$

5.3. Evaluation

To each structure (solution) is associated a numerical value (fitness) that represents the quality of this structure and indicates its aptitude degree. The value of fitness is gotten through the objective function. The objective function of this study aims at the minimization of the position error of the end-effector manipulator and the smallest joint angular displacement. The positioning error is calculated through the Euclidean Distance between the current and final coordinates of the manipulator:

$$E_p = \sqrt{(x_i - x_f)^2 + (y_i - y_f)^2} \tag{16}$$

Being (x_f, y_f) the desired position and (x_i, y_i) the current position, gotten through the Direct Kinematics calculation of the manipulator (Eq. 1) with the use of the current angles generated by the GA. The angular error is also gotten through the Euclidean Distance between the initial and final configurations of the joint angles of the manipulator:

$$E_a = \sum_{i=1}^{3} \left\| \{\theta_{i,f}\} - \{\theta_{i,in}\} \right\| \tag{17}$$

Where $\{\theta_{i,in}\}$ are the initial configuration angles of the manipulator, $\{\theta_{i,f}\}$ are the current angles generated by the GA and ||.|| denotes the Euclidean distance. In this work, the positioning error and the angular displacement are approached together through a multi-objective function (Nunes *et al*., 2006). Using the weighting factors method, that satisfies the restriction $\omega_1 + \omega_2 = 1$, the optimization problem is defined as the inverse of the error:

$$fitness = \frac{1}{\omega_1 E_p + \omega_2 E_a} \tag{18}$$

The weighting factors ω_1 and ω_2 are experimentally chosen as 0.12 and 0.88 respectively.

5.4. Seletion

The selection process in GAs chooses individuals for the reproduction. The selection is based on the individuals aptitude: best individuals have more probability of being chosen for the reproduction. The selection method chosen for this work was Stochastic Universal Sampling (SUS), in which the individuals are mapped out in adjacent segments whose length is the same as the value given for the evaluation function to each individual. In this method it is used N hands equally spaced between them (N = number of parents to be selected) and the roulette spins only once. The chosen parents are the individuals marked for the N hands. The distance between the hands will be $1/N$ and the position of the first hand is given for a number generated randomly between 0 and $1/N$. The method SUS is considered fast enough for the serial processing and more efficient than the Roulette selection methods, Stochastic Rest and Ranking (Baker, 1987).

5.5. Crossing and mutation

The individuals selected for the following population are recombined through the Crossover operator. This operator is considered the main characteristic of the GAs. The pairs of individuals are chosen randomly and new individuals are created from the interchange of the genetic material. The descendants will be different, however, with genetic characteristics of both. This method (single-point crossover) is the most applied one and was used in this work. The chromosomes created from the crossover operator are, later, submitted to the mutation operation. Based on the probability pm of mutation, the content of a chromosome position is modified.

6. NEURAL NETWORK IMPLEMENTATION

A Radial Basis Function neural network (RBF) with two neurons in the input layer, 200 neurons in the hidden layer and three neurons in the output layer were trained to provide the manipulator´s inverse kinematics solution. The RBF network receives as entered the cartesian coordinates (x, y) and supplies, as exit, the angles (θ_1, θ_2, θ_3) corresponding. The schematic diagram of the system is given in Fig. 4.

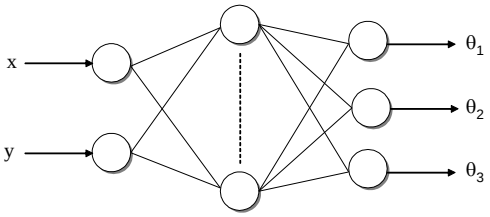

Figure 4. A schematic diagram of the implemented system

Firstly, 1200 points in the manipulator work space was generated. Cubic trajectory planning, explained in Section 4, was used to generate the (θ_1, θ_2, θ_3) joint angles according to different (x, y) Cartesian coordinates, from initial point to target point. These values were recorded in a file to form the learning set of the neural network. To prevent extreme angular oscillations from initial to target point (Fig. 5), the angles for each generated points, had been optimized by genetic algorithms, described in Section 5.

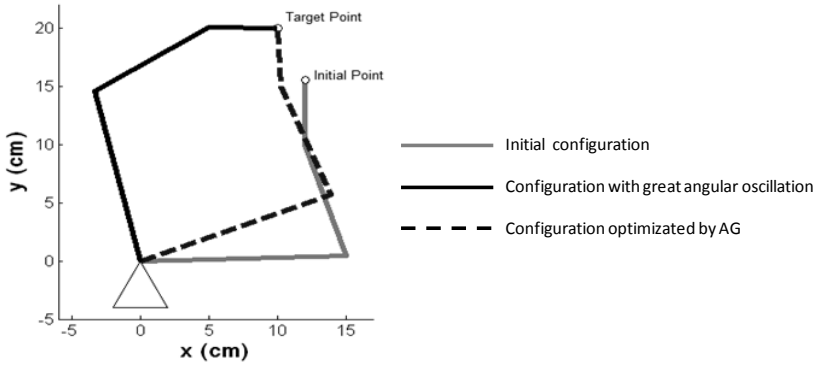

Figure 5. Angular oscillation

Five thousand of these data was used in training of neural network, and the rest was used in testing. The training process is completed until the error is acceptable. To understand whether the error is acceptable or not, Euclidean Distance that refers to difference between the end-effector and the target point is used. As a result, it can be decided that the error is acceptable or not depending on the obtained distance. Here, the sensitivity of the application is also important in the decision of acceptable error. It has been completed approximately in 300 iterations. The number of neurons in hidden layer was found experimentally. Error at the end of the learning is 0.0000114 for training set. The direct kinematics equations given in Eq. (1) were used to prepare the training set of the neural network.

7. RESULTS

The neural network was evaluated and used to get the inverse kinematics solution of the robotic manipulator. The neural network results were compared with the target values and the relative errors of the angles had been evaluated for the trajectory defined for the manipulator. Due the great amount of points used for the test, will be shown the results for some experiments. For the (x, y) coordinated pairs presented to the neural net, were obtained the results presented in Tab. 1.

Table 1. Simulation results.

Target point (x, y)	**Neural network results**			**Average Relative Error (%)**
	θ_1	θ_2	θ_3	
(11.0600, 16.6530)	0.15446	1.93340	1.61350	0.004224
(-4.4292, 25.1580)	1.21700	2.43000	1.99510	0.024799
(-19.8880, 20.1140)	2.06200	2.82100	2.29770	0.000000
(12.4790, 25.4990)	0.78677	1.45790	1.41400	0.072212
(15.9000, 18.1020)	0.21384	1.53240	1.39820	0.000000

The average relative error showed in Tab. 1 is based on the current values of the joint angles in the target points, gotten by Genetic Algorithms. In the Tab. 1, the used neural network reached an average relative error of in maximum 0.072212% in the joint angles determination. The angles had been substituted in the Eq. 1 in order to get x and y positions. The results are presented in Tab. 2. The biggest relative positional error in x was 0.14901% and, in y, 0.00497%.

Table 2. Cartesian coordinates determination.

Target point (x, y)	**(x, y) position obtained by Neural network results**	**Relative Error in x (%)**	**Relative Error in y (%)**
(11.0600, 16.6530)	(11.0610, 16.6530)	0.00904	0.00000
(-4.4292, 25.1580)	(-4.4358, 25.1580)	0.14901	0.00000
(-19.8880, 20.1140)	(-19.8880, 20.1150)	0.00000	0.00497
(12.4790, 25.4990)	(12.4960, 25.5000)	0.13623	0.00392
(15.9000, 18.1020)	(15.9010, 18.1020)	0.00629	0.00000

8. CONCLUSION

In this work, the inverse kinematics solution using RBF neural network for a three-joint robotic manipulator is presented. A genetic algorithm was implemented for minimum angular displacement from initial to target point of the trajectory. The training angles for the neural network had been generated by a cubic trajectory. The neural network presented good performance during the tests. It is observed that RBF neural network can be used in inverse kinematics problem.

9. REFERENCES

Baker, J., 1987, "Reducing bias and inefficiency in the selection algorithm, In Genetic Algorithms and their Applications", Proceedings of the Second International Conference on Genetic Algorithms, p.14–21.

Craig, J, J., 1989, "Introduction to Robotics: mechanics and control", Addison-Wesley, 2nd ed.,Mass, 450 p.

Goldberg, D. E., 1989, "Genetic Algorithms in Search, Optimization and Machine Learning", Mass: Addison-Wesley Publishing.

Kalra, P., Prakash, N. R., 2003, "A Neuro-genetic Algorithm Approach for Solving the Inverse Kinematics of Robotics Manipulators", IEEE International Conference on Systems, Man and Cybernetics, vol. 2, pp. 1979-1984.

Köker, R., Öz, C., Çakar, T., Ekiz, H., 2004, "A Study of Neural Network Based Inverse Kinematics Solution for a Three-Joint Robot", Robotics and Autonomous Systems, Elsevier, vol. 49, pp. 227-234.

Michalewicz, Z., 1994, "Genetic Algorithms + Data Structures = Evolution Programs", Springer-Verlag, 3 ed.

Nunes, L. E. N. P., Gamarra Rosado, V. O., Grandinetti, F. J., 2006, "Geração de Trajetória de um Manipulador Robótico Planar de Três Graus de Liberdade Através de Algoritmos Genéticos", Anais do XVI CBA – Congresso Brasileiro de Automática, Salvador, BA, out. 2006.

Oyama, E., Tachi, S., 2000, "Modular Neural Net System for Inverse Kinematics Learning", Proceedings of IEEE International Conference on Robotics & Automation, San Francisco, pp. 3239-3246.

Venkatesan, P., Anitha, S., 2006, "Application of a radial basis function neural network for diagnosis of diabetes mellitus", Current Science, vol. 91, no. 9, 10.

10. RESPONSIBILITY NOTICE

The authors are the only responsible for the printed material included in this paper.