IJCCCE Vol.16, No.1, 2016

An Improved Micro Artificial Immune Algorithm Utilizing Employed Honey Bees for the Identification of Nonlinear Systems

Omar Farouq Lutfy

Control and Systems Engineering Department,
University of Technology, Baghdad, Iraq.

e-mail: 60157@uotechnology.edu.iq

Received: 10/9 /2015

Accepted: 29/2 /2016

Abstract – This paper presents an improved micro artificial immune (IMAI) algorithm utilizing basic concepts from swarm intelligence. In particular, to enhance the searching capability of the recently developed micro artificial immune system (Micro-AIS) algorithm, employed honey bees are recruited to provide high-quality antibodies for the working population of the IMAI algorithm. The proposed algorithm is used to find the optimal kernel values for the Volterra series model to identify nonlinear systems. To demonstrate the efficiency of the proposed method, three different types of nonlinear systems are considered, including a highly nonlinear rational system, a heat exchanger, and a continuous stirred tank reactor (CSTR). For all these systems, the IMAI algorithm has achieved accurate modelling results and fast convergence rates. Moreover, a comparative study was conducted with other optimization methods, namely the original Micro-AIS algorithm, the improved particle swarm optimization (IPSO), the real-coded genetic algorithm (GA), the least mean squares (LMS), the least mean p-norm (LMP), and the least mean absolute deviation (LMAD). From this comparative study, the proposed IMAI algorithm has achieved the best modelling performance compared to the other methods.

Keywords: – Improved micro artificial immune algorithm, Artificial immune system, Volterra series model, System identification.

1. Introduction

The artificial immune system (AIS) is one of the recent biologically-inspired algorithms, which was proposed by borrowing basic ideas from the biological immune system. As an evolutionary computation method, the AIS algorithm is based on emulating the main functions of the biological immune system whose task is to distinguish foreign attacking molecules (called antigens) and then destroy them. Compared to the GA, which is the most common and widely used global optimization method, the AIS algorithm can achieve better maintenance of diversity and faster convergence rate [1], [2]. Moreover, compared to other optimization methods, the AIS algorithm is characterized by the simplicity in implementation and the employment of fewer number of control parameters. As a result, the AIS algorithm has been successfully utilized to solve various types of optimization problems [3]-[8].

For instance, Abdollahpour and Rezaeian [6] utilized the AIS algorithm to minimize the maximum completion time (makespan) for the permutation flow shop scheduling problem. On the other hand, Wu et al. [7] proposed a self-adaptive attribute weighting method for Naive Bayes classification using AIS algorithm. In another study [8], a hybrid artificial immune network and particle swarm optimization (PSO) was presented to allocate the order quantity for key suppliers at minimum cost in a manufacturing environment.

However, a drawback of the conventional AIS algorithm employed in the studies mentioned above is that, the population size increases significantly during the cloning process. This increase dictates a long processing time and a considerable memory usage [9], [10]. To handle this difficulty, Herrera-Lozada et al. [9] proposed a modified version of the AIS algorithm which uses a reduced population size with two simple and fast mutation operators. This method was called the Micro-AIS algorithm. Owing to its powerful search capability, the Micro-AIS algorithm has been efficiently employed in different applications [9]-[11].

Classified as a nonlinear modelling technique, the Volterra series model has been successfully exploited in solving various nonlinear system identification problems [12]-[16]. In particular, the Volterra model belongs to the category of polynomial filters. With the ability to approximate a wide range of real nonlinear systems, the Volterra model is characterized by its unique structure in which the model output is generated by a linear combination of model coefficients and nonlinear functions of the input signal. The model coefficients are called kernels of the Volterra model. These kernels must be adjusted by a suitable optimization method in order to attain the best result from the Volterra model. In general, the most commonly used method to optimize the Volterra kernels is the gradient descent method [13]. For example, Jafarian et al. [17] utilized a gradient descent learning method to adjust the Volterra kernels for approximating unknown functions. In another study, Ji and Gan [18] employed the normalized least-mean-square algorithm, which is based on the gradient descent approach, to update the Volterra kernels to predict the sound pressure level and the harmonic distortion in loudspeakers. To compare the performance of different adaptive algorithms, Singh and Chatterjee [19] applied several well-known gradient descent algorithms to find the optimal

Volterra kernels for the problem of nonlinear system identification. However, since gradient descent methods belong to the single direction search methods, they are more likely to trap at local minima in the search space. In order to solve this problem, a more suitable choice to adjust the Volterra kernels is to use evolutionary computation algorithms. These algorithms are based on multiple direction search concept, and hence they possess the ability to find the global optimal solution for a given problem.

In this paper, an improved micro artificial immune (IMAI) algorithm is proposed by borrowing some concepts from swarm intelligence. More specifically, aiming at enhancing the searching capability of the original Micro-AIS algorithm, employed honey bees are used to supply high-quality individuals for the working population of the IMAI algorithm. The proposed evolutionary method is employed to find the optimal values for the Volterra kernels in order to identify nonlinear systems.

2. Volterra Series Model

The Volterra series model has a powerful ability to identify nonlinear systems. Since the output of the Volterra model is computed using only present and past input signals, the stability of the Volterra model is guaranteed [12]. Utilizing a linear combination of nonlinear functions of the input signal, the Volterra model is represented by an infinite series in the form of convolution integrals. The continuous form of the Volterra series model of the q^{th} order is given by the following equation [13]:

$$\begin{aligned} y(t) = h_0 &+ \int_{-\infty}^{\infty} h_1(\tau)x(t-\tau)d\tau \\ &+ \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} h_2(\tau_1,\tau_2)x(t-\tau_1)x(t-\tau_2)d\tau_1\, d\tau_2 \\ &+ \cdots + \int_{-\infty}^{\infty}\int_{-\infty}^{\infty}\cdots\int_{-\infty}^{\infty} h_q(\tau_1,\tau_2,\ldots,\tau_q)\prod_{i=1}^{q}x(t-\tau_i)d\tau_i, \end{aligned} \tag{1}$$

where $x(t)$ and $y(t)$ are the input and the output signals, respectively, h_0 is a constant kernel, $\tau_1, \tau_2, \ldots, \tau_q$ are variables of integration, $h_1(\tau)$ is the first-order kernel associated with the input signal $x(t)$ at time lag τ, $h_2(\tau_1, \tau_2)$ is the second-order kernel which represents the quadratic properties of the model, and $h_q(\tau_1, \tau_2, \ldots, \tau_q)$ is the qth-order kernel representing the qth-order nonlinear information of the model. The discrete version of (1) is represented by the following equation:

$$\begin{aligned} y[n] = h_0 &+ \sum_{k_1=0}^{N-1} h_1[k_1]x[n-k_1] \\ &+ \sum_{k_1=0}^{N-1}\sum_{k_2=0}^{N-1} h_2[k_1,k_2]x[n-k_1]x[n-k_2] \\ &+ \cdots + \sum_{k_1=0}^{N-1}\sum_{k_2=0}^{N-1}\cdots\sum_{k_q=0}^{N-1} h_q[k_1,k_2,\ldots,k_q]\prod_{i=1}^{q}x[n-k_i], \end{aligned} \tag{2}$$

where N is the memory size of the Volterra model. Despite the great flexibility of (2) to approximate various nonlinear systems of arbitrary order, the practical implementation of such an ideal infinite series is almost impossible due to its enormous computational complexity. Therefore, a truncated version of (2), called the second-order Volterra (SOV) model, is more suitable for practical applications. As the name implies, the SOV model takes only the constant kernel, the first-order kernels, and the second-order kernels of (2). This

truncated SOV model has an acceptable computational burden and at the same time it can provide an attractive approximation performance [19]. For these reasons, the truncated SOV model is adopted in this study. The mathematical representation of the SOV model is given by the following expression [12], [13], [19]:

$$y[n] = h_0 + \sum_{k=1}^{N} h[k]x[n-k+1] + \sum_{k_1=0}^{N-1}\sum_{k_2=k_1}^{N-1} h[k_1, k_2]x[n-k_1]x[n-k_2]. \quad (3)$$

where h_0, $h[k]$, and $h[k_1, k_2]$ represent the constant, the first-order, and the second-order kernels, respectively. For convenience, equation (3) above can be re-written in the following vector form:

$$y[n] = H\,X^T, \quad (4)$$

where T represents the transpose of a vector, H is the Volterra kernel vector, and X is the Volterra input vector. In more details, the two vectors H and X can be represented as follows:

$$H = [h_0, h[1], \ldots, h[N], h[0,0], h[0,1], \ldots, h[0,N-1], h[1,1], \ldots, h[N-1,N-1]], \quad (5)$$

$$X = [1, x[n], \ldots, x[n-N+1], x^2[n], x[n]x[n-1], \ldots, x[n]x[n-(N-1)], x^2[n-1], \ldots, x^2[n-(N-1)]]. \quad (6)$$

The length of each of the two vectors, H and X, can be calculated by the following equation:

$$L = \frac{(N+1)(N+2)}{2} \quad (7)$$

As can be concluded from (3), the adjustable parameters of the SOV model are given by the kernels contained in vector H. Hence, in order to exploit the SOV model for solving a particular problem, the values of these kernels must be determined by a suitable optimization method. In this study, the proposed IMAI algorithm is utilized to optimize the SOV kernel values for identifying nonlinear systems.

3. Artificial Immune System

Classified as a population-based meta-heuristic method, the artificial immune system (AIS) has recently shown a remarkable efficiency in solving different optimization problems [3]-[8]. The AIS algorithm can achieve better maintenance of diversity and faster convergence rate compared to other evolutionary algorithms, such as the GA [1], [2], [10]. The main operators of the AIS algorithm were inspired by the potential functionalities of the biological immune system. This system acts as a powerful biological defense network whose task is to detect and destroy external substances, called antigens, which might attack the human body. For this purpose, the immune system uses a huge variety of antibodies to counteract the attacking antigens. These antibodies are proteins which are produced by certain cells, called the B cells. The B cells can be stimulated by specific antigens when the affinity value is high. In particular, the affinity value denotes the relationship degree between the antibody and the antigen. Once an antigen is detected, B cells with high affinity value to that antigen will start to produce many antibodies, called clones, to fight the external antigen. After destroying the antigen, one of the clones, with the

highest affinity value to the antigen, is selected to be a memory cell and lives longer in the human body. Then, if the same or similar antigen is encountered again, it will be detected and destroyed in a faster and more effective process, since the immune system has all the information on the antigen from the memory cells of the so-called immunological memory [7], [10], [20].

3.1. The Micro-AIS Algorithm

Based on the biochemical operators of the immune system described above, several AIS algorithms have been proposed for various applications. One of these effective AIS algorithms is the Micro-AIS algorithm [9] which was proposed to handle the increase in population size during the cloning process. Due to its remarkable search capability, the Micro-AIS algorithm has been successfully utilized to solve various optimization problems [9]-[11].

The Micro-AIS algorithm utilizes a small randomly generated population of five antibodies. These antibodies enter a nominal convergence loop which iterates for 10 generations. During this nominal convergence loop, several immune operators are applied on the working population, namely, selection, cloning, maturation, re-selection, and self-regulation. Figure 1 illustrates a flowchart of the Micro-AIS algorithm.

As indicated in Figure 1, after the nominal convergence loop is completed using 10 iterations, new five antibodies are generated from the working population. These antibodies are constituted from the best two antibodies in the working population and three other randomly generated antibodies. Subsequently, a new nominal convergence loop is commenced again using these five antibodies as the working population. This procedure is repeated until a stopping condition is satisfied and the result is the final optimized solution found by the Micro-AIS algorithm.

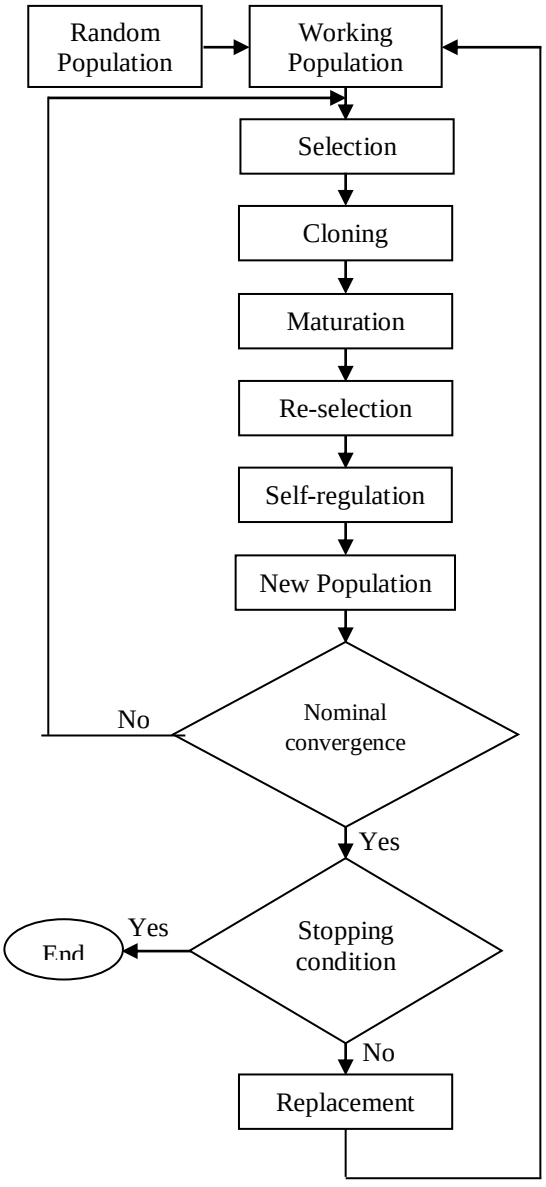


Figure 1. A flow chart of the Micro-AIS algorithm

3.2. The Proposed Improved Micro Artificial Immune (IMAI) Algorithm

In order to enhance the searching efficiency of the Micro-AIS algorithm described above, some principles from

artificial bee colony are used in this study to provide more efficient variations in the decision variables of the antibodies. More specifically, the concept of employed honey bees are utilized to provide high-quality antibodies for the working population instead of selecting these antibodies directly from the existing solutions in the original Micro-AIS described in Figure 1. This improvement in forming the working population has resulted in an obvious enhancement in the searching ability of the original Micro-AIS algorithm, as will be demonstrated by the simulation results in the next sections.

The artificial bee colony (ABC), which was first proposed by Karaboga [21], is based on simulating the foraging behavior performed by a swarm of honey bees. In general, the ABC algorithm consists of three groups of bees, namely, the employed, the onlookers, and the scout bees. The task of the employed bees is to intensively search for high quality food sources around those related to other bees. The food sources represent possible solutions for the problem being optimized. In particular, the number of food sources is equal to the number of bees in the hive, i.e. there is one employed bee for every food source. On the other hand, onlooker bees perform further search on the good food sources found by the employed bees. Finally, when a food source is fully exploited, the employed bees associated with that source become scout bees and start a new search for a possible food source [22]-[24].

In this study, only the first group of bees, in particular the employed bees, are used to improve the Micro-AIS searching performance. In more details, in the global best ABC algorithm proposed by Gao et al. [25], each bee searches around the best solution obtained in the previous iteration in order to find a better food source. For example, in order to generate a candidate food position $V_i = [v_{i,1}, v_{i,2}, \ldots, v_{i,D}]$ from the old one $X_i = [x_{i,1}, x_{i,2}, \ldots, x_{i,D}]$ in the memory, the global best ABC uses the following expression [25]:

$$v_{i,j} = x_{best,j} + \Phi_{i,j}(x_{r1,j} - x_{r2,j}), \quad (8)$$

where $i \in \{1, 2, \ldots, SN\}$, SN is the number of food sources, j is a randomly chosen index from the range $\{1, 2, \ldots, D\}$, D is the number of decision variables (problem dimension), $r1$ and $r2$ are two indices represented by mutually different random integers selected from $\{1, 2, \ldots, SN\}$ and different from the base index i. $x_{best,j}$ is the jth variable of the best individual vector which has the best fitness value in the current population, and $\Phi_{i,j}$ is a random number generated from the range [-1, 1].

As can be seen from (8), the best solution in the current population plays an important role in improving the convergence speed by directing the search movement towards the best solution found so far. In this way, the ABC populations can converge to the global optimum solution quickly.

3.3. Procedure of Applying the Proposed Algorithm for Optimizing the Volterra Kernels

As mentioned before, the original Micro-AIS algorithm uses the best two antibodies resulted from the nominal convergence loop, together with three other randomly generated ones, to constitute the new working population [9]-[11], as illustrated in Figure 1. However, in order to utilize more efficient

antibodies in the working population, the employed honey bees are used here to derive the search method towards finding better solutions. The following procedure clarifies the steps of applying the proposed method to find the optimal values for the Volterra kernels.

Step 1: Initialize the mutation probability (P_m) to set the clones' mutation rate. In addition, initialize the maximum number of generations.

Step 2: Generate randomly a population of five antibodies within a certain range according to the optimized problem. In this study, each of these antibodies contains all the adjustable kernels of a single Volterra model, as defined in (5). These five antibodies are then copied into the working population which enters a nominal convergence loop for 10 iterations, see Figure 1.

Step 3: Evaluate the objective function for each antibody in the working population using the mean square of error (MSE) criterion which has the following definition:

$$\mathrm{MSE} = \frac{1}{N_p}\sum_{k=1}^{N_p}(y(k) - y_m(k))^2 \qquad (9)$$

where $y(k)$ and $y_m(k)$ are the system output and the Volterra model output at time sample k, respectively, and N_p is the number of training patterns. After that, determine the affinity value for each antibody using the following expression:

$$affinity = \frac{1}{Objective \quad function + \varepsilon} \qquad (10)$$

where ε is a small constant which is used to avoid division by zero.

Step 4: Sort the antibodies according to their affinity values in a descending order. Therefore, the antibody with the highest affinity value will be the best antibody, and it is called *BestAb* in the Micro-AIS algorithm.

Step 5: Perform the cloning process on the five sorted antibodies. This number of antibodies, which is only five, is adopted to avoid the problem of increasing the population size during the cloning process of the conventional AIS algorithm. The cloning process is done according to the following formula:

$$N_c = \sum_{i=1}^{n}\big(n - (i-1)\big), \qquad (11)$$

where N_c is the number of clones to be generated for each antibody, n is the total number of antibodies in the population, and i is the index of the current antibody starting from *BestAb* which is the antibody with the highest affinity value. In this way, by using the cloning operator in (11), a population of five antibodies will generate a population of 15 clones. More precisely, *BestAb* will produce five clones; the second ranking antibody will produce four clones and so on until the worst antibody is reached which will produce a single clone.

Step 6: In this step, the maturation of clones is achieved by using a specific mutation operator. At the beginning of the nominal convergence loop, the mutation probability for each group of clones obtained from the same antibody is determined by the following expression:

$$prob\_mutation(i) = \frac{Aff(i)}{\sum_{i=1}^{n} Aff(i)}, \qquad (12)$$

where *i* is the index of the antibody which will set the mutation probability for the group of clones generated from it and *n* is the total number of antibodies in the population. This mutation probability is calculated according to the affinity value of each antibody and decreases uniformly in each iteration. In this way, the group of clones obtained from *BestAb* mutates less than other groups of clones that were produced from the remaining antibodies. In order to uniformly decrease the mutation probability at each iteration within the nominal convergence loop, the following expression is utilized:

$$\begin{aligned} &if\ P_m \le \frac{prob\_mutation(i)}{iteration}, \\ &then\ apply\ \ the\ mutation\ operator \end{aligned} \qquad (13)$$

where P_m is the predefined mutation probability and the variable iteration in the denominator represents the current iteration within the nominal convergence loop.

In order to perform the mutation of clones, the following operator is used:

$$x' = x + \frac{(rand \cdot range)}{(iteration \cdot group\_N_c)}, \qquad (14)$$

where *x'* is the mutated decision variable, *x* is the decision variable to be mutated, *rand* is a uniform random number in the range [0, 1], *iteration* is the current iteration within the nominal convergence loop, and *group_Nc* is the number of clones in each group of antibodies. For the five clones derived from *BestAB*, $range \in [LB,UB]$ in (14) is a random number between the lower bound (*LB*) and the upper bound (*UB*) of the decision variables. However, for the remaining clones, *range* is the corresponding value (decision variable) from *BestAb*.

Step 7: Sort the 15 clones in a descending order according to their affinity value. The number of clones, which is 15, is selected as suggested by the original Micro-AIS algorithm in order to limit the population size to a relatively small number of clones and to avoid the problem of producing a large population size as was usually done in the conventional AIS algorithm. This limited population size, which is only 15, contributes in reducing the processing time and the memory requirements for the Micro-AIS algorithm. Subsequently, constitute a new population from the best two clones and three other clones which are randomly selected from the 15 mature clones. This selected population of five clones enters a new iteration of the nominal convergence loop.

Step 8: After the nominal convergence loop, which comprises 10 iterations, is completed, mature clones are considered as food sources and (8) described above is used to further search for high quality food sources (clones) from the existing ones. Once each candidate source position is produced using (8), its performance is evaluated and compared with that of the old food source. If the new food source has a better quality than the old one, it replaces the old food source. Otherwise, the old food source is kept for the next generation. Finally, a new working population is made from the best two solutions found by the employed bees along with three other randomly generated solutions.

Step 9: If the stopping condition is satisfied, the algorithm is terminated. Otherwise, go to Step 3 to start a new nominal convergence loop using the population formed in Step 8.

4. Simulation results

To show the effectiveness of the proposed method, three different types of highly nonlinear systems are considered in this section. The objective is to model these systems by the Volterra model using the proposed IMAI algorithm to optimize the Volterra kernels. To achieve this objective, a specific code was written in an m-file under the environment of MATLAB software. Figure 2 illustrates the identification structure used to model the nonlinear systems considered in this study.

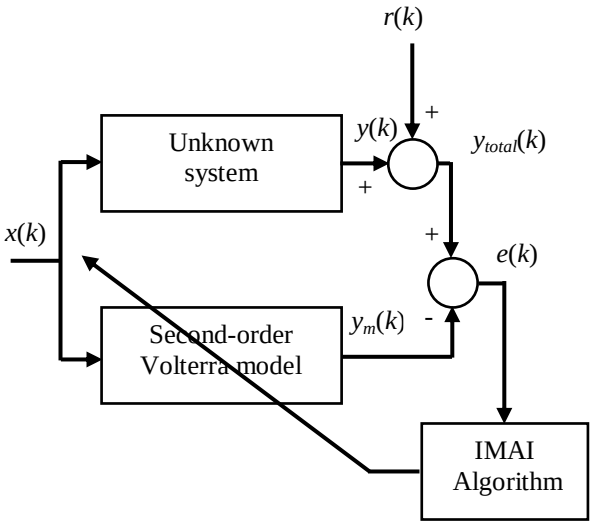


Figure 2. The identification structure of Volterra series modelling using the proposed IMAI algorithm

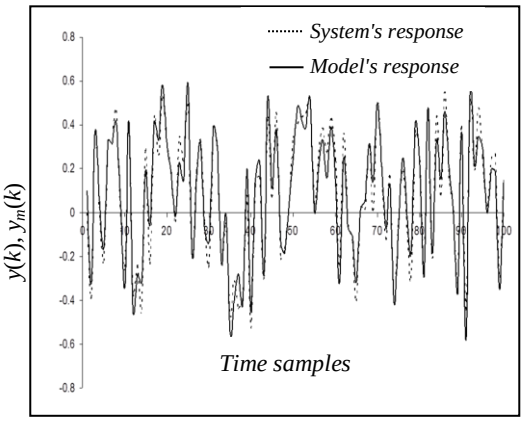
In Figure 2, $x(k)$ and $y(k)$ are the input and the output signals of the unknown system at time sample k, respectively, $y_m(k)$ is the Volterra model output, $r(k)$ represents the measurement noise, $y_{total}(k)$ is the summation of $y(k)$ and $r(k)$, and $e(k)$ is the modeling error signal.

Example 1: This example is a highly nonlinear rational system which can be defined by the following discrete-time equation [13]:

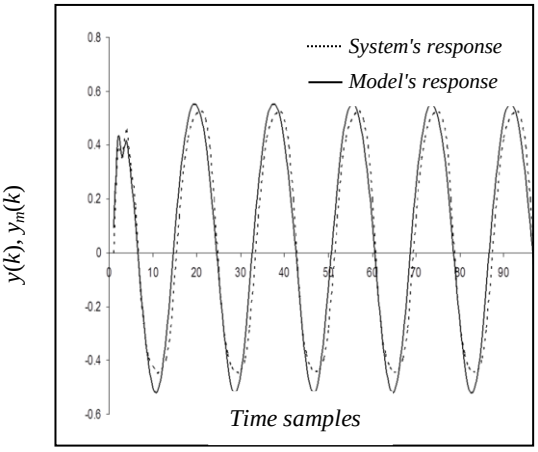
$$y(k)=\frac{0.3y^2(k-1)+0.8x(k-1)+0.6y(k-2)}{1+x^2(k-1)+y^2(k-1)}. \quad (15)$$

To compare the modeling result of the proposed method with that achieved by the IPSO [13], [26] in modelling Example 1, the parameters of the IMAI algorithm were set as follows: number of generations was set to 330 and P_m was set to 0.3. Utilizing the identification structure shown in Figure 2, the input signal $x(k)$ is a random sequence uniformly generated from the interval [-1, 1] and the measurement noise $r(k)$ is applied as a Gaussian noise of $N(0, 0.001)$.

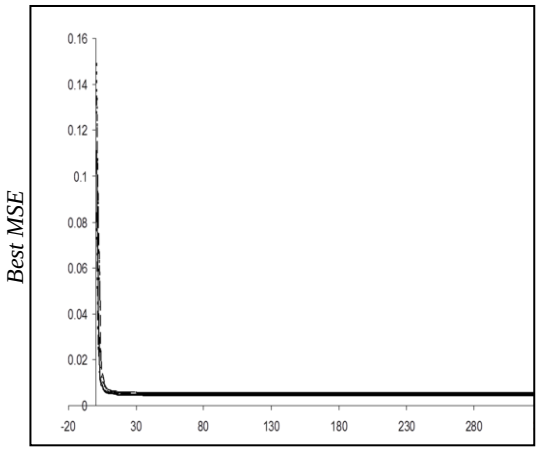
Figure 3(a) shows the Volterra modelling performance for the training signal when the memory size (N) in (3) is set to 5.



(a)



(b)



(c)

Figure 3. Example 1 using N = 5 (a) Volterra modeling result for the training signal (b) Volterra modeling result for the testing signal (c) best MSE against iterations for Run 1 to Run 4.

From Figure 3(a), it can be seen that the Volterra model, optimized by the proposed algorithm, has achieved a good modelling result with a MSE of 4.525×0^{-3}. In order to evaluate the generalization ability of the Volterra model, a testing signal was applied at the input of the unknown system and the Volterra model. This testing signal has the following form [13]:

$$x(k) = 0.8\cos(\frac{\pi}{9}k) \qquad (16)$$

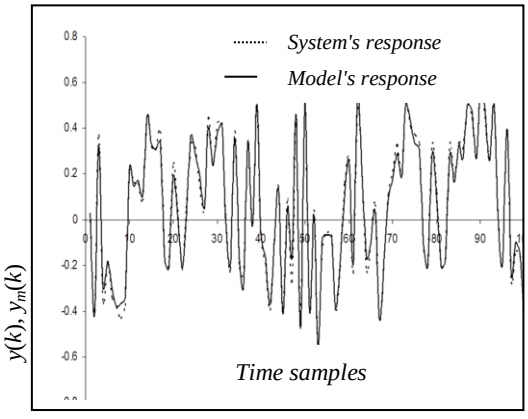
Figure 3(b) shows the simulation result of this test, where the Volterra model has accomplished an acceptable modelling performance with a MSE of 6.8×10^{-3}.

To demonstrate the robustness of the proposed IMAI algorithm, four different runs (Run 1 - Run 4) with four different initial values for the antibodies were conducted. Figure 3(c) depicts the resulting MSE against generations for all the runs. As shown in Figure 3(c), despite the differences in initializing the antibodies and the input signals in each run, the proposed method has shown a remarkable robustness by achieving almost identical performances for the four runs. Moreover, the IMAI algorithm has shown fast convergence rates for all the runs by minimizing the MSE from the first few generations.

It is worth noticing that as the memory size (N) of the second-order Volterra model in (3) increases, more accurate modelling results can be achieved. Therefore, in the following tests, a memory size of 8 is considered for comparison purposes. Figures 4(a), 4(b), and 4(c) illustrate the simulation results for the same tests considered before when N was 5. Compared to the previous results for N = 5, Figures 4(a), 4(b), and 4(c) show a better Volterra modelling performance when the memory size is increased to 8.

In order to compare the performance of the proposed method with that of the IPSO proposed in [13], Table 1 summarizes the Volterra results trained by the proposed method and by the IPSO to model Example 1.

Table 1 evidently shows that the proposed method has outperformed the IPSO in terms of achieving less MSE values for all the cases.



(a)

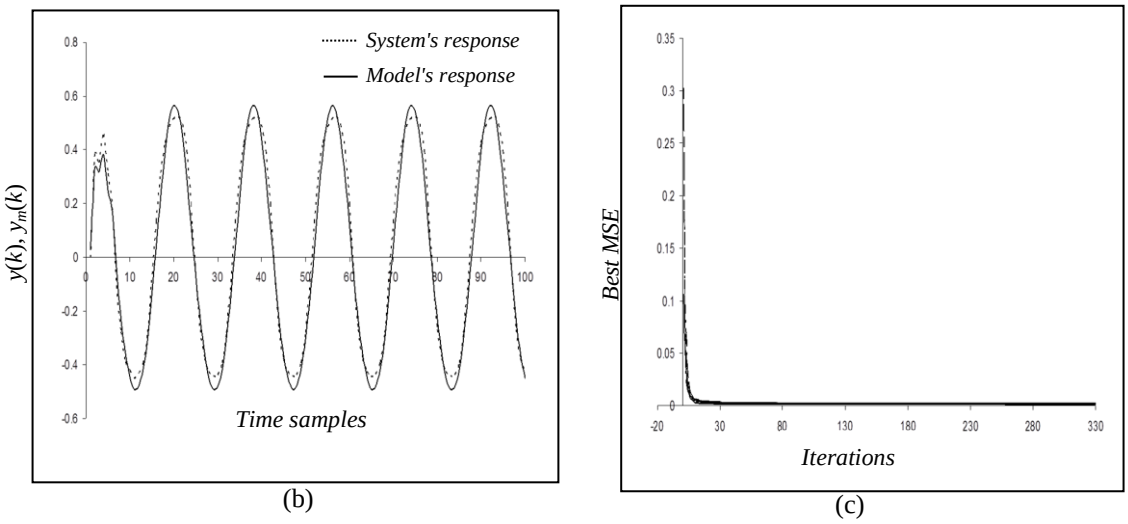


Figure 4. Example 1 using $N = 8$ (a) Volterra modeling result for the training signal (b) Volterra modeling result for the testing signal (c) best MSE against iterations for Run 1 to Run 4.

Table 1. Comparison results of the proposed method and the IPSO in training the Volterra model

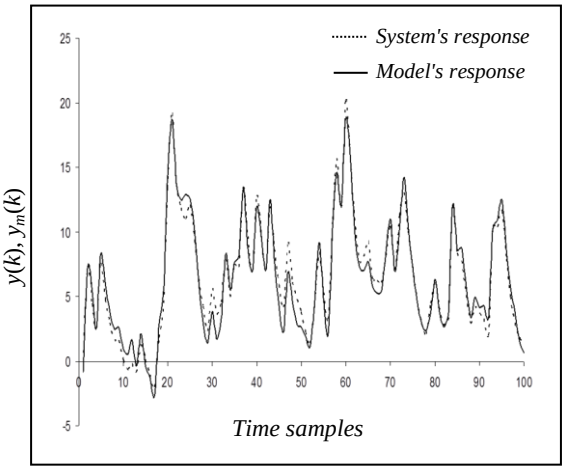
Optimization Method	Criterion	Example 1	
		$N = 5$	$N = 8$
IPSO method	Training MSE (Average of 4 runs)	6.213×10^{-3}	2.434×10^{-3}
	Testing MSE	9.29×10^{-3}	4.913×10^{-3}
Proposed method	Training MSE (Average of 4 runs)	4.875×10^{-3}	1.608×10^{-3}
	Testing MSE	6.8×10^{-3}	2.8×10^{-3}

Example 2: The second example is the nonlinear heat exchanger process represented by the following Hammerstein model [11], [27]:

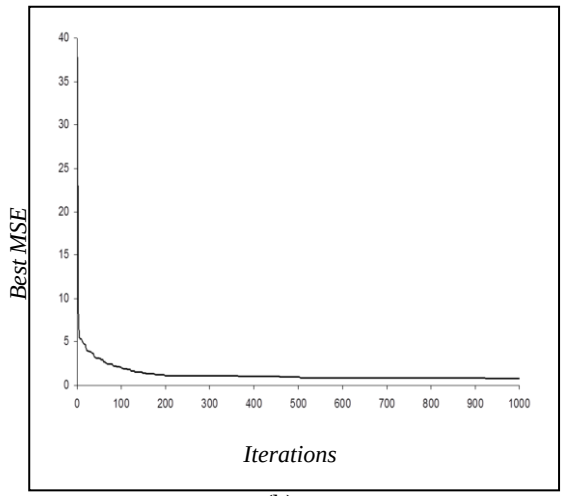
$$w(k) = -31.549x(k) + 41.732x^2(k) - 24.201x^3(k) + 68.634x^4(k) \quad (17)$$

$$y(k) = \frac{0.207q^{-1} - 0.1764q^{-2}}{1 - 1.608q^{-1} + 0.6385q^{-2}} w(k)$$

where $w(k)$ is the static nonlinearity, $x(k)$ is the process flow rate, and $y(k)$ is the process exit temperature. The input to the process $x(k)$ is applied as a random signal uniformly generated from the interval [0, 1]. The IMAI parameters were assigned the following values: number of generations = 1000 and $P_m = 0.3$. In the Volterra series equation, the memory size was set to 8. The modeling result of this process is illustrated in Figure 5(a), where it can be seen that the Volterra model has done well in tracking the input signal. The best objective functions against generations are depicted in Figure 5(b), which shows the fast convergence rate achieved by the proposed IMAI algorithm.



(a)



(b)

Figure 5. Example 2 (a) Volterra modeling result (b) best MSE against iterations.

Example 3: In this example, the Volterra series is used to model the CSTR process, which exhibits a highly nonlinear dynamical behavior.

In this process, a single irreversible exothermic reaction A⟶B is assumed to occur in the reactor. The CSTR process model is represented by the following two nonlinear ordinary differential equations [28]:

$$\dot{C}_A = \frac{q}{V}(C_{AF} - C_A) - k_0 C_A \exp\left(\frac{-E}{RT}\right),$$
$$\dot{T} = \frac{q}{V}(T_f - T) + \frac{(-\Delta H)k_0 C_A}{\rho C_\rho}\exp\left(\frac{-E}{RT}\right) + \frac{\rho_c C_{\rho c}}{\rho C_\rho V} q_c \left[1 - \exp\left(\frac{-hA}{q_c \rho_c C_{\rho c}}\right)\right] \times (T_{cf} - T) \quad (18)$$

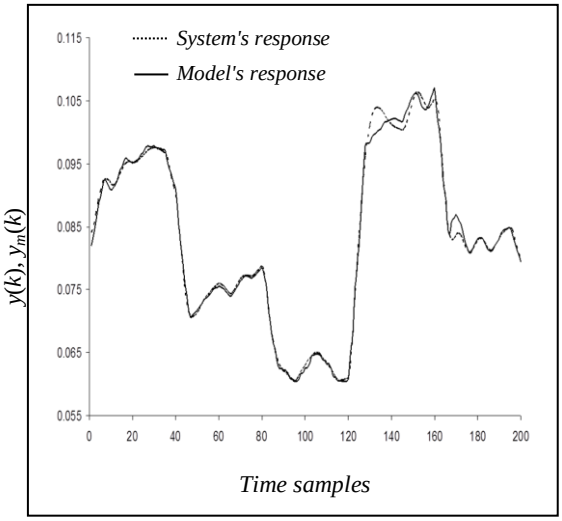
where C_A represents the product concentration of component A, T is the reactor temperature, q is the feed flowrate, and q_c is the coolant flowrate. In this example, q_c is the input signal to the CSTR process, while C_A is the corresponding output signal. The remaining parameters of the CSTR model are given in Table 2.

The continuous time model in (18) was numerically solved using the fourth order Runge-Kuta method with a simulation step size of 0.1 minutes.

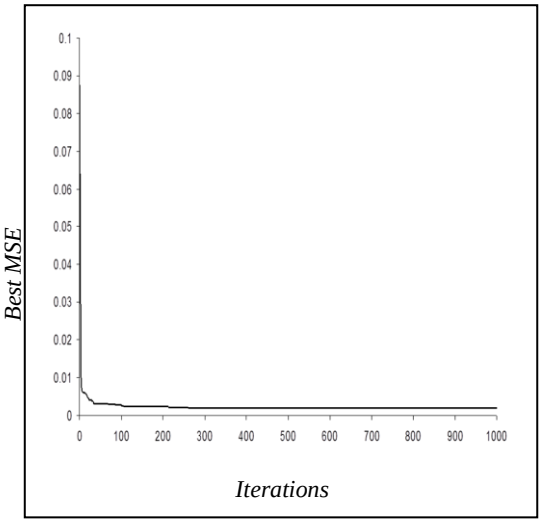
Table 2. CSTR parameters defined in nominal operating conditions

Parameter	Description	*Nominal value*
q	Process flow-rate	100 $l\ min^{-1}$
C_{AF}	Inlet feed concentration	1 $mol\ l^{-1}$
T_f	Feed temperature	350 K
T_{cf}	Inlet coolant temperature	350 K
V	Reactor volume	100 l
hA	Heat transfer coefficient	$7\times10^5\ cal\ min^{-1}\ .\ K^{-1}$
k_0	Reaction rate constant	$7.2\times10^{10}\ min^{-1}$
E/R	Activation energy	$9.95\times10^3\ K$
ΔH	Heat of reaction	$-2\times10^5\ cal\ mol^{-1}$
ρ, ρ_c	Liquid densities	1000 $g\ l^{-1}$
C_ρ, $C_{\rho c}$	Specific heats	1 $cal\ g^{-1}\ .\ K^{-1}$
q_c	Coolant flow-rate	103.41 $l\ min^{-1}$
T	Reactor temperature	440.2 K
C_A	Product concentration	$8.36\times10^{-2}\ mol\ l^{-1}$

The input signal to the CSTR process (q_c) consists of a series of arbitrary step changes within the operating conditions of the process. The IMAI parameters and the Volterra memory size were the same as those used for Example 2. Figure 6(a) depicts the Volterra modeling result for the CSTR process.



(a)



(b)

Figure 6. Example 3 (a) Volterra modeling result (b) best MSE against iterations.

From Figure 6(a), it is obvious that the Volterra model has achieved a good performance in reproducing the actual output of the CSTR process. On the other hand, Figure 6(b) shows the best objective functions against generations. Similar to the optimization results for the previous examples, the IMAI algorithm has achieved a fast convergence by minimizing the objective function from the early stage of the optimization process, as can be seen from Figure 6(b).

Finally, to demonstrate the superiority of the proposed method, a comparative study was conducted with other related optimization techniques. As evolutionary algorithms, this comparison includes the original Micro-AIS [9]-[11] and the real-coded GA [29]. Moreover, as adaptive gradient based techniques, the least mean squares (LMS), the least mean p-norm (LMP), and the least mean absolute deviation (LMAD) algorithms [19] were also considered in this study. Using a Volterra memory size of 8, the same examples considered before are used in this comparative study. In both the original Micro-AIS and the proposed IMAI algorithms, the number of generations was set to 1000 and the mutation probability (P_m) was set to 0.3. Since the two algorithms mentioned above use a nominal convergence loop of 10 iterations with 15 clones, 150,000 performance evaluations are accomplished for 1000 generations. Hence, to achieve a fair comparison, the population size and the number of generations in the GA were set to 50 and 3000 respectively. Moreover, the crossover probability (P_c) and the mutation probability (P_m) in the GA were set to 0.8 and 0.05, respectively. From several tests, the above parameter settings were sufficient to guarantee good optimization results.

To handle the stochastic nature of the evolutionary methods under

consideration, 10 independent runs were done for each of the Micro-AIS, the IMAI, and the GA. Table 3 summarizes the results of this comparative study.

From the results shown in Table 3, it is evident that the proposed IMAI algorithm has outperformed the other optimization methods considered in this study. In particular, the IMAI algorithm has achieved the least values for the MSE for all the systems compared to those achieved by the other methods.

Table 3. Comparison results of the Micro-AIS algorithm, the real-coded GA, the LMS, the LMP, the LMAD, and the proposed method in training the Volterra model

Optimization Method	MSE (Average of 10 runs)		
	Nonlinear system	Heat exchanger	CSTR
Micro-AIS	9.709×10^{-4}	0.957	4.77×10^{-3}
Real-coded GA	42.7×10^{-4}	10.804	33.21×10^{-3}
LMS	204×10^{-4}	6.564	9.4×10^{-3}
LMP	207×10^{-4}	6.582	8.7×10^{-3}
LMAD	316×10^{-4}	9.706	23.5×10^{-3}
Proposed method	8.724×10^{-4}	0.839	4.06×10^{-3}

5. Conclusions

By exploiting basic ideas from swarm intelligence, this paper presented an improved version of the recently developed Micro-AIS algorithm. This improved version, called IMAI algorithm, utilizes employed honey bees to provide high-quality antibodies for the working population of the IMAI algorithm. The proposed algorithm was applied to find the optimal kernel values for the Volterra series model to identify nonlinear systems. From the simulation tests to model three nonlinear systems, the IMAI algorithm has shown its effectiveness in terms of accurate modeling performance and fast convergence rate. Furthermore, from a comparative study, the IMAI algorithm has achieved the best modeling results compared to those of the original Micro-AIS, the IPSO, the real coded GA, the LMS, the LMP, and the LMAD algorithms.

References

[1] A. Prakash and S.G. Deshmukh, "A multi-criteria customer allocation problem in supply chain environment: an artificial immune system with fuzzy logic controller based approach," Expert Syst Appl, vol. 38, pp. 3199-3208, 2011.

[2] R.J. Kuo, W.L. Tseng, F.C. Tien, and T.W. Liao, "Application of an artificial immune system-based fuzzy neural network to a RFID-based positioning system," Comput Ind Eng, vol. 63, pp. 943-956, 2012.

[3] G. Pan, K. Li, A. Ouyang, and K. Li, "Hybrid immune algorithm based on greedy algorithm and delete-cross operator for solving TSP," Soft Comput., DOI: 10.1007/s00500-014-1522-3, 2014.

[4] R. Shang, L. Jiao, Y. Ren, L. Li, and L. Wang, "Quantum immune clonal coevolutionary algorithm for dynamic multiobjective optimization," Soft Comput., vol. 18, pp. 743-756, 2014.

[5] R. Shang, H. Ma, J. Wang, L. Jiao, and R. Stolkin, "Immune clonal selection algorithm for capacitated arc routing problem," Soft Comput., DOI 10.1007/s00500-015-1634-4, 2015

[6] S. Abdollahpour and J. Rezaeian, "Minimizing makespan for flow shop scheduling problem with intermediate buffers by using hybrid approach of artificial immune system," Appl Soft Comput., vol. 28, pp. 44-56, 2015.

[7] J. Wu, S. Pan, X. Zhu, Z. Cai, P. Zhang, and C. Zhang, "Self-adaptive attribute weighting for Naive Bayes classification," Expert Syst Appl., vol. 42, pp. 1487-1502, 2015.

[8] R.J. Kuo, C.M. Pai, R.H. Lin, and H.C. Chu, "The integration of association rule mining and artificial immune network for supplier selection and order quantity allocation," Appl Math Comput., vol. 250, pp. 958-972, 2015.

[9] J.C. Herrera-Lozada, H. Calvo, and H. Taud, "A micro artificial immune system," Polibits, vol. 43, pp. 107-111, 2011.

[10] O.F. Lutfy, "Wavelet neural network model reference adaptive control trained by a modified artificial immune algorithm to control nonlinear systems," Arab J Sci Eng, vol. 39, pp. 4737-4751, 2014.

[11] O.F. Lutfy, "Control of nonlinear systems utilizing a wavelet neural network controller optimized by micro artificial immune systems," Proceedings of the Engineering Conference of control, computers, and mechatronics, Baghdad, Iraq, pp. 113-119, 2014.

[12] Y.-S. Yang, W.-D. Chang, and T.-L. Liao, "Volterra system-based neural network modeling by particle swarm optimization approach," Neurocomputing, vol. 82, pp. 179-185, 2012.

[13] W.-D. Chang, "Volterra filter modeling of nonlinear discrete-time system using improved particle swarm optimization," Digit Signal Process, vol. 22, pp. 1056-1062, 2012.

[14] A. Maachou, R. Malti, P. Melchior, J.-L. Battaglia, A. Oustaloup, and B. Hay, "Nonlinear thermal system identification using fractional Volterra series," Control Eng Pract, vol. 29, pp. 50-60, 2014.

[15] C.A. Schmidt, S.I. Biagiola, J.E. Cousseau, and J.L. Figueroa, "Volterra-type models for nonlinear systems identification," Appl Math Model, vol. 38, pp. 2414-2421, 2014.

[16] R.A. Sandler, S.A. Deadwyler, R.E. Hampson, D. Song, T.W. Berger, and V.Z. Marmarelis, "System identification of point-process neural systems using probability based Volterra kernels," J Neurosci Meth, vol. 240, pp. 179-192, 2015.

[17] A. Jafarian, S. Measoomy, and S. Abbasbandy, "Artificial neural networks based modeling for solving Volterra integral equations system," Appl Soft Comput., vol. 27, pp. 391-398, 2015.

[18] W. Ji, and W.-S. Gan, "Identification of a parametric loudspeaker system using an adaptive Volterra filter," Appl Acoustic, vol. 73, pp. 1251-1262, 2012.

[19] T.S.D. Singh and A. Chatterjee, "A comparative study of adaptation algorithms for nonlinear system identification based on second order Volterra and bilinear polynomial filters," Measurement, vol. 44, pp. 1915-1923, 2011.

[20] N.S. Halvaiee and M.K. Akbari, "A novel model for credit card fraud detection using artificial immune systems," Appl Soft Comput., vol. 24, pp. 40-49, 2014.

[21] D. Karaboga, "An idea based on honey bee swarm for numerical optimization," Erciyes University, Technical Report-TR06, Kayseri, Turkey, 2005.

[22] M. Govardhan and R. Roy, "Generation scheduling in smart grid environment using global best artificial bee colony algorithm," Int J Elec Power, vol. 64, pp. 260-274, 2015.

[23] M. Maeda and S. Tsuda, "Reduction of artificial bee colony algorithm for global optimization," Neurocomputing, vol. 148, pp. 70-74, 2015.

[24] H. Habbi, Y. Boudouaoui, D. Karaboga, and C. Ozturk, "Self-generated fuzzy systems design using artificial bee colony optimization," Inform Sciences, vol. 295, pp. 145-159, 2015.

[25] W. Gao, S. Liu, and L. Huang, "A global best artificial bee colony algorithm for global optimization," J Comput Appl Math, vol. 236, pp. 2741-2753, 2012.

[26] W.-D. Chang and S.-P. Shih, "PID controller design of nonlinear systems using an improved particle swarm optimization approach," Commun Nonlinear Sci Numer Simulat, vol. 15, pp. 3632-3639, 2010.

[27] H. Al-Duwaish and W. Naeem, "Nonlinear model predictive control of Hammerstein and Wiener models using genetic algorithms," Proceedings of the IEEE International Conference on control application, Mexico City, pp. 465-469, 2001.

[28] O.F. Lutfy, S.B. Mohd Noor, M.H. Marhaban, and K.A. Abbas, "A genetically trained adaptive neuro-fuzzy inference system network utilized as a proportional-integral-derivative-like feedback controller for non-linear systems," Proc Inst Mech Eng Part I- J Syst Control Eng, vol. 223, pp. 309-321, 2009.

[29] O.F. Lutfy, S.B. Mohd Noor, M.H. Marhaban, and K.A. Abbas, "Nonlinear modeling and control of a conveyor-belt grain dryer utilizing neuro-fuzzy systems," Proc Inst Mech. Eng. Part I- J Syst. Control Eng., vol. 225, pp. 611-622, 2010.