Pergamon

0952-1976(95)00007-0

Engng Applic. Artif. Intell. Vol. 8, No. 3, pp. 261–269, 1995

0952-1976/95 $9.50 + 0.00

Contributed Paper

Genetic Adaptive Observers

LA MOYNE L. PORTER II
The Ohio State University, U.S.A.

KEVIN M. PASSINO
The Ohio State University, U.S.A.

(Received September 1994)

A genetic algorithm (GA) uses the principles of evolution, natural selection, and genetics to offer a method for the parallel search of complex spaces. This paper shows how to utilize GAs to perform on-line adaptive state estimation for nonlinear systems. First, it shows how to construct a genetic adaptive observer (GAO) where a GA evolves the gains in a state observer in real time so that the state estimation error is driven to zero. Next, several examples are used to illustrate the operation and performance of the GAO. The paper starts by showing how the GAO can pick the observer gains for a linear state estimation problem. Following this it demonstrates how the GAO performs in estimating the state of a nonlinear, chaotic system for various inputs, noise, and model mismatches.

Keywords: State estimators, observers, adaptation, genetic algorithms.

1. INTRODUCTION

Since the inception of the genetic algorithm concept by Holland[1] in 1975 it has been useful in solving a wide variety of problems. Economics, game theory, and the traveling salesman problem are just a few instances of situations where the GA has been used to prize an optimal solution from a complex, nonlinear search space.[2,3] The GA has also found application in the area of design automation for conventional and intelligent controllers. Typically, for this, a controller is decomposed into a set of parameters which the GA attempts to optimize by using a simulation-based fitness evaluation of candidate controllers in the closed-loop system. For instance, Lee and Takagi[4] designed a fuzzy system using a genetic algorithm, for the inverted pendulum. Controller fitness evaluation is based upon simulation of the system over a variety of initial conditions to obtain a fuzzy controller capable of handling a variety of operating conditions. Their GA manipulates strings which represent input and output membership functions. Their fitness evaluation incorporates a strategy to minimize the number of rules, and "scores" the ability of the fuzzy controller to balance the pendulum. Porter and Borairi[5] use the GA in an eigenstructure assignment technique. Their GA chooses values of a linear feedback matrix to minimize the error between actual closed-loop eigenvalues and desired eigenvalues, as well as actual and desired eigenvectors. Michalewicz *et al.*[6] use the GA to solve certain optimal control problems. Ishibuchi *et al.*[7] design fuzzy controllers for pattern classification, with the GA attempting to minimize the number of rules while maximizing the number of correct classifications. Katai *et al.*[8] present a technique which utilizes a GA and a fuzzy controller to reduce the error between a model of the system and the actual system over a time window. Karr and Gentry[9] use a GA to design a fuzzy controller to control the pH of an acid–base system. Park, Kandel and Langholz[10] optimize a fuzzy reasoning model via a genetic algorithm to control a direct current series motor. Varšek *et al.*[11] use a GA to derive and subsequently optimize rules for the control of an inverted pendulum. Nomura *et al.*[12] present a method for GA tuning of a fuzzy controller that fits input-output data. They utilize a gradient descent method coupled with rule minimization to obtain optimal input membership functions.

It is important to note that approaches similar to these GA-based computer-aided control-system design techniques can be used for the off-line design of state estimators, since the design of observers closely parallels that of controllers. This, however, is not the focus of this paper, which investigates whether GAs can be used for the on-line synthesis/tuning of observers. In work which is closely related, Das and Goldberg,[13] Maclay and Dorey,[14] Kristinsson and Dumont,[15] and Etter *et al.*[16] use the GA for system identification (i.e. to

Correspondence should be sent to: Dr K. M. Passino, Department of Electrical Engineering, The Ohio State University, 2015 Neil Avenue, Columbus, OH 43210, U.S.A.

identify a process model of a plant using input–output data). Yao and Sethares[17] use the GA for nonlinear parameter estimation. Here they use a GA to find the parameters of a partially known nonlinear system by matching input–output data. Using an elitism-type operator they are able to prove convergence of the algorithm.

In this paper a GA is applied to estimate the state vector of a possibly nonlinear system. The estimation scheme closely mimics that of the technique described in Ref. 18, with the objective function similar to those of Refs 15 and 17. Receiving process input and output information, the GA manipulates an observer structure (i.e. the observer gains) in order to reduce the state estimation error to zero. The problem differs from the one studied for conventional adaptive observers[19] in that it is assumed here that there exists a known mathematical model of the process, and hence that parameter estimation for the model used in the observer is not necessary. The adaptive nature of this technique is expressed in the adaptation of the observer gains by the GA, and the main focus is on the development of adaptive observers for nonlinear systems, rather than for linear systems as in Ref. 19.

Section 2 provides some background information on the base-10 GA that is used. Section 3 describes the "genetic adaptive observer" (GAO) that was used for state estimation, and Section 4 illustrates the operation and performance of the GAO for a linear and nonlinear state-estimation problem. Section 5 gives some concluding remarks, and a critique of the technique.

2. BACKGROUND: A BASE-10 GENETIC ALGORITHM

The GA performs a parallel search of a parameter space by using *genetic operators* (e.g. *selection*, *crossover* and *mutation*) to manipulate a set of encoded strings which represent system parameters.* These genetic operators combine the strings in different arrangements where the optimal configuration being sought is one which maximizes a user-specified *objective function* (also called a "fitness function"). The parallel nature of this search is realized by the algorithm's repetitive processing of a *population* (set) of strings, beginning with an initial population. This initial population is either a set of guesses at potential solutions to the optimization problem, or a random set of strings generated by the computer. A subsequent population is created via evaluation of the objective function, and the use of genetic operators to form a new *generation* of strings which hopefully comprise the best characteristics of the previous set. Ideally, the strings of the new generation are either as capable or more capable of maximizing the value of the objective function than those of the previous population. Typically, the strings that maximize the objective function at the time of termination of the GA are taken to be solutions to the optimization problem. A string is composed of digits (genes), each of which can take on different values (alleles). In the artificial genetic environment described here, alphabets of any desired cardinality can be used in order to encode these values. In a binary environment, an allele can be represented by a 0 or 1. The reproduction operation merely copies selected strings from the old generation into the new generation. Strings are selected for reproduction, based upon their fitness values; thus strings with higher than average fitnesses are preferentially copied into the new generation. Goldberg (see Ref. 2, p. 11) cites the analogy of spinning a roulette wheel partitioned according to the fitness of each individual string with respect to the average fitness of the entire population. Thus, strings with large fitness values occupy a greater portion of the wheel, and are more likely to be selected. The crossover operation is the primary vehicle for developing new structures. Crossover qualifies as a genetic operator since it allows for the exchange of chromosome building blocks (genes), which readily occurs in natural genetics. Once two strings are selected by the reproduction operation, crossover will occur with a probability, p_c, which is specified by the user during initialization of the routine. If crossover occurs, a "cross site" is randomly determined. This cross site is a number between 1 and $p-1$, where p is the length of the string, which determines how much genetic material will be exchanged between the two selected strings. Once the cross site, k, is determined, crossover dictates that the two strings simply exchange the alleles between the $k+1$ position and the end of the string.

The mutation operator is the secondary method for introducing new structures into the population. The mutation operation is performed on a digit-by-digit basis: each digit (position) of the string has an equal probability of mutation, p_m, against which it is tested. When mutation occurs, the string position is changed to a different allele selected from the set of possible digits. Mutation should be used sparingly (by choosing p_m to be small), as increased use results in a random walk through the search space.

In addition to the three GA operators *selection*, *crossover*, and *mutation*, an operator called *elitism* see (Ref. 2, p. 115) will also be utilized. Elitism ensures that the string with the largest fitness value will propagate to the next generation without manipulation by other GA operators. Elitism is used since it is likely that within a sufficiently small time range (i.e. a small number of generations), one set of observer gains will be better than any other set. To perturb the parameters of the best observer unnecessarily may result in an unsatisfactory performance for which no genetic technique can adapt. Hence, by using the elitism operator, over a certain time range the best set of observer gains will consistently determine the state estimate. The

* While a brief overview is provided, it is assumed that the reader has a familiarity with the conventional base-2 GA for which many excellent tutorial introductions exist (see e.g. Refs 2, 3 and 20).

string which defines the best set of observer gains is given a number of copies in proportion to its fitness relative to the average fitness of the population. The remaining slots, if any, are chosen via *selection*, *crossover* and *mutation*.

The operation of the GA changes slightly, depending on the base of the numbers to which the genetic operators are applied. Traditionally GAs have been designed to operate over binary numbers (referred to as "GA_2") and more recently several base-10 GAs ("GA_{10}") have been developed.[3] To avoid the need for encoding and decoding of strings, a GA_{10} algorithm, will be employed that operates similarly to the GA_2 described in Ref. 2, except that:

(i) its digits vary over the numbers 0, 1, 2, . . . , 9 and there is an extra digit for the "+" or "−" sign;
(ii) the strings are split into a portion to the left and to the right of the decimal point; and
(iii) its genetic mutation operator randomly perturbs the digits to any value in 0, 1, 2, . . . , 9 or toggles between "+" and "−" for the sign digit with probability, p_m.

If strings outside the domain are generated by the mutation and crossover operators then another candidate is generated via these operations. For pathological cases, a limit is placed upon the number of successive mutations allowed and in violation of this limit, the parameter (and string) are reset to the maximum or minimum value (whichever is closest to the actual value of the string). It is important to note that the GAO approach presented in this paper does not depend in any critical way on using GA_{10}. The standard GA_2 method works in a similar manner (some simulation-based investigations to verify this for the maximization problem are reported by the authors in Ref. 3, p. 18). GA_{10} is used simply to avoid issues in coding and decoding strings and to make an algorithm for which it is easy to gain insight into its search procedure. For instance, it seems easier to gain intuition about the operation of the GAO since the underlying GA operates over base-10 numbers rather than long binary strings.

A GENETIC ADAPTIVE OBSERVER

3.1. Problem statement

Suppose a given, continuous-time process

$$\dot{x} = f(x, u) \tag{1}$$

$$y = g(x) \tag{2}$$

with a discrete-time representation

$$x(k+1) = \bar{f}(x(k), u(k))$$
$$y(k) = \bar{g}(x(k)). \tag{3}$$

The equations

$$x(k+1) = \Phi x(k) + \Gamma u(k)$$
$$y(k) = Hx(k) \tag{4}$$

may be used to denote (3) if the process is linear. A discrete-time, possibly nonlinear, observer

$$\bar{x}(k+1) = \bar{f}(\bar{x}(k), u(k)) + \bar{f}_1(L, y(k), \bar{x}(k)) \tag{5}$$
$$\bar{y}(k) = \bar{g}(\bar{x}(k))$$

can be defined, where $\bar{x}(k)$ is an estimate of $x(k)$. As is standard, it is assumed in the design of the observer that a model of the process is available [i.e. equation (3)], and it is necessary to show how to pick $\bar{f}_1$ to complete the specification of the observer. If the process is linear with equation (4), the observer in equation (5) becomes

$$\bar{x}(k+1) = \Phi\bar{x}(k) + \Gamma u(k) + L(y(k) - H\bar{x}(k)) \tag{6}$$
$$\bar{y}(k) = H\bar{x}(k)$$

with n states, p inputs and q outputs. $\Phi \in \mathcal{R}^{n\times n}$, $\Gamma \in \mathcal{R}^{n\times p}$, $H \in \mathcal{R}^{q\times n}$, and $L \in \mathcal{R}^{n\times 1}$,[21] so that $\bar{f}_1(L, y(k), \bar{x}(k)) = L(y(k) - H\bar{x}(k))$. In conventional linear observer design one can use, e.g. pole-placement techniques to specify the observer gain L. Normally, such a design process is completed off-line. Here, the GA is used to manipulate the observer L so that the estimation error $x(k) - \bar{x}(k)$ goes to zero. In fact, $\bar{f}_1(L, y(k), \bar{x}(k)) = L(y(k) - H\bar{x}(k))$ will be used to specify the observer structure even if the underlying process is nonlinear. Hence, when the process is nonlinear the problem becomes significantly more challenging, because of the adaptation of an observer structure where the error signal enters linearly so that it performs well for a nonlinear system.

3.2. Genetic adaptive observers

In the GAO a form of the discrete-time observer structure in equation (5) is used to obtain an estimate of the process states. The GAO structure utilizes an objective function which determines the fitness of each set of L gains. Let O_i denote the ith observer (a string of digits parameterizing the L gains) in the population of observers O and assume that the size of the population $|O|$ is m. The objective function utilized in the GAO is of the form, $J_i = \Sigma_{j=1}^{N_p} \alpha_j p_j^2$, where p_j is a parameter used to evaluate the "goodness" of O_i, N_p is the number of those parameters, and the α_j are scaling factors. For example, p_j might represent the amount of error between the estimate of the process output and the process output. J_i should approach zero as the algorithm operates, so $-J_i$ is used, and maximized. For selection purposes, the value of $-J_i$ is mapped to a fitness value, $\bar{J}_i > 0$ as follows:

$$\bar{J}_i = \begin{cases} \dfrac{1}{|J_i|} & J_i \neq 0 \\ 2 * \max(\bar{J}_i) & J_i = 0 \end{cases} \tag{7}$$

where $i = 1,2, \ldots, m$ and m is the population size.

In the rather unlikely event that a $-J_i$ has a fitness of zero, the second equation in equation (7) identifies this

$-J_i$ as the "elite" string by mapping it to the largest fitness value. The quantity 2*max was chosen here more or less arbitrarily. Assigning a fitness value that is too large would result in the next generation completed being dominated by this "zero" fitness valued string. This may not be desirable in all cases, so the 2*max represents an acceptable trade-off. Once a $\bar{J}_i$ has been assigned to all strings, selection proceeds normally, with $\bar{J}_i$ used as the fitness value.

Suppose that the value of $N_p = 2$ so that $-J(i)$ will be derived from a weighting of two quantities. The first term, p_1, is an error term on the current state estimate using

$$\begin{aligned} \bar{x}(k) &= \bar{f}(\bar{x}(k-1), u(k-1)) + L(y(k-1) - H\bar{x}(k-1)) \\ \bar{y}(k) &= \bar{g}(\bar{x}(k)) \\ p_1 &= y(k) - \bar{y}(k) \end{aligned} \tag{8}$$

(where $\bar{x} = 0$) and the second term, p_2) is a error term on the future estimate of the state:

$$\begin{aligned} \bar{x}(k+1) &= \bar{f}(\bar{x}(k), u(k)) + L(y(k) - H\bar{x}(k)) \\ \bar{y}(k+1) &= \bar{g}(\bar{x}(k+1)) \\ p_2 &= \bar{y}(k+1) - y(k). \end{aligned} \tag{9}$$

Using this structure for fitness assignment, the GA chooses L based upon the amount of error between the estimation of the process output and the actual output. As the GAO has no direct information about the n elements of the state vector, and L_i gains, $i = 1,2,\ldots,n$, change merely to suit minimization of this output error. It was found via simulations that this change in L gains can result in unsatisfactory fluctuations in the estimation error of the $n-1$ states. There are many methods of alleviating this penchant for the GA to change the L gains unnecessarily. Three candidates are:

(1) restructuring the objective function to place a penalty on the change of L gains;
(2) using a reference model and controller scheme (similar to that in Ref. 18) to dictate a closed-loop system resonse that mitigates the frequency of L changes; or
(3) employing an averaging of the L_i gains for smoothing.

Of the three proposed schemes, (3) is used, because of its ease in implementation and due to the fact that for the applications in hand, it was found easier to design genetic adaptive observers with this approach.

This method is implemented merely by taking the average of the last N_t chosen values of the L_i gains, $i = 1,2,\ldots,n$ (chosen by the GA at each time k because L displayed maximum fitness), which are denoted by

$$\text{avg}\{L_i(k)\} = \frac{\sum_{j=k-N_t}^{k-1} L_i}{N_t}. \tag{10}$$

So as to not use a set of average L gains that would produce clearly unacceptable behavior, the set of average values (avg$\{L_i(k)\}$) is used only if its fitness value is within a certain user-defined tolerance of the fitness value of the L gains with maximum fitness. Mathematically, the avg$\{L_i(k)\}$, $i = 1,2,\ldots,n$ is only used if

$$|L^m(k)| - |L^a(k)| \leq \gamma \tag{11}$$

where $|L^m(k)|$ is the absolute value of the fitness of the L gains with maximum fitness at time k, $|L^a(k)|$ is the absolute value of the fitness of avg$\{L_i(k)\}$, $i = 1,2,\ldots,n$, and $\gamma > 0$ is the user-defined tolerance. Once the condition in equation (11) is met, the avg$\{L_i(k)\}$, $i = 1,2,\ldots,n$ is used after that time. Hence, the γ tolerance specifies a termination condition for the adaptation in the GAO (clearly if one expects significant process parameter variations one could keep the adaptation mechanism on so that the system continually adapts). If γ is chosen to be small then the maximally fit members of the population produces a behavior that is similar to the behavior generated by the average L_i. Essentially, the γ tolerance forces a convergence to an algorithm which would not necessarily converge by itself.

The GAO proceeds according to the following pseudo-code:

(1) Initialize the GA. Initialize population. Set initial L gains to acceptable values. This could be either a random choice within specified limits or perhaps the designer has an idea of what L gains to choose (e.g. via conventional observer design guidelines). Set the crossover and mutation probabilities. Set the string length. Select α_is. Select γ. Set N_t. Set $k = 0$.
(2) Collect $y(k)$ and $u(k)$.
(3) Assign a fitness to each O_i, $i = 1,2,\ldots,m$. Calculate p_1 and p_2 from equations (8) and (9).

$$J_i = (\alpha_1 p_1^2 + \alpha_2 p_2^2)$$

(4) The maximally fit O_i is compared with the fitness of the average L_i gains. If the γ tolerance equation (11) is met, then the state estimate is provided by the average L gains from this time onward; otherwise, the maximally fit O_i provides the state estimate at time k.
(5) Produce the next generation using GA operators (i.e. use the GA operators selection, crossover, mutation, and elitism in the

standard manner to generate a new population of observer gains).

(6) Let $k := k+1$. Go to Step 2.

Prior to satisfying the γ tolerance requirement, the GA picks the set of L gains at each time k which will: (1) minimize the error between the current estimate of the process output and the actual process output [equation (8)] and; (2) minimize the difference between the estimate of the process output one time step in the future and the current process output [equation (9)]. Choice of the α_is determines the amount of "future" error to be allowed while attempting to keep the quantity in (1) small. The adaptive nature of the algorithm, which is achieved by the genetic operators in Step 5, ceases when the γ tolerance is satisfied in Step 4.

4. EXAMPLES

This section considers a linear and nonlinear example in order to contrast the differences in difficulty between the two problems, and to show how the GAO can perform for a challenging state-estimation problem. In both examples the average L gains are calculated over the last 200 samples ($N_t = 200$). The L_i averages are not available for consideration until 200 samples of data have been collected, and until that the time maximally fit L gains are used.

4.1. Linear example

Consider the second-order linear continuous-time state equations

$$\dot{x}_1(t) = -2.25x_1(t) - 4.5x_2(t) \tag{12}$$

$$\dot{x}_2(t) = -5.5x_2(t) + u(t) \tag{13}$$

which have poles at -2.25 and -5.5. Using a sampling period of 0.01 s and a forward difference rule, a discrete equivalent of

$$x_1(k+1) = 0.9775x_1(k) - 0.045x_2(k) \tag{14}$$

$$x_2(k+1) = 0.945x_2(k) + 0.01u(k) \tag{15}$$

is obtained.

Given the state x_1 (i.e. $y = x_1$), it is desired to estimate the state x_2 using the GAO. An input $u(k)$ is chosen to be the sum of two sinusoids

$$u(k) = 0.999 + 0.42\cos(1.75k) + 0.32\cos(4.5k). \tag{16}$$

The process state is initialized at $x_1 = x_2 = 1.5$. The observer is initialized with $\hat{x}_1 = \hat{x}_2 = 0$ to represent the possibility that the initial process state is not known. Crossover and mutation probabilities are chosen as 0.6 and 0.4, respectively (on the basis of past experience with GAs). A string length of 16 digits is chosen (eight to represent each L gain), each of which is initialized to a random number between 9.9 and -9.9. The γ value is set at 5×10^{-6}. The values of α_1 and α_2 are 5 and 40, respectively. Figure 1 shows the estimation errors for x_1 and x_2. Figure 2 shows the L gains.

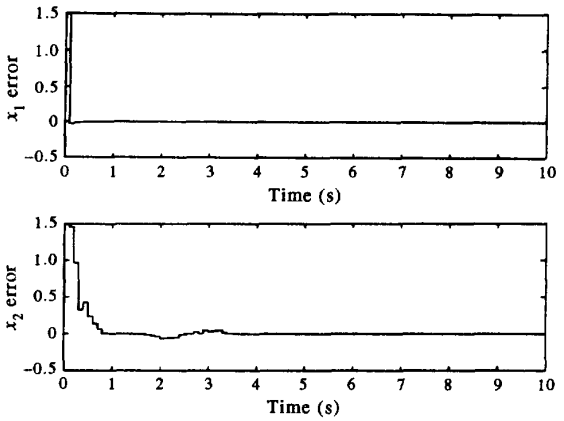

Fig. 1. x_1 and x_2 estimation errors: linear system.

Note that relatively good tracking performance is achieved in Fig. 1, and Fig. 2 shows that at $t \approx 3.5$ seconds, the γ tolerance in equation (11) is satisfied. Note that for this linear case one could design, *a priori*, observer gains that would result in better performance. This example merely illustrates that the off-line design procedure for linear observers can be automated via an on-line GA for this example. The next example shows how the GAO can evolve the L gains for a nonlinear state-estimation problem that does not have a simple procedure for observer design.

4.2. A nonlinear example

This nonlinear equation is taken from Ref. 22 and for a certain choice of input $u(t)$ describes a chaotic, glycolytic oscillator:

$$\begin{aligned} \dot{x}_1(t) &= -x_1(t)x_2^2(t) + u(t) \\ \dot{x}_2(t) &= x_1(t)x_2^2(t) - x_2(t). \end{aligned} \tag{17}$$

Using a sampling period, $T_s = 0.01$ s and a forward difference rule a discrete equivalent of

$$\begin{aligned} x_1(k+1) &= x_1(k) - T_s(x_1(k)x_2^2(k) + u(k)) \\ x_2(k+1) &= x_2(k) + T_s(x_1(k)x_2^2(k) - x_2(k)) \end{aligned} \tag{18}$$

is obtained.

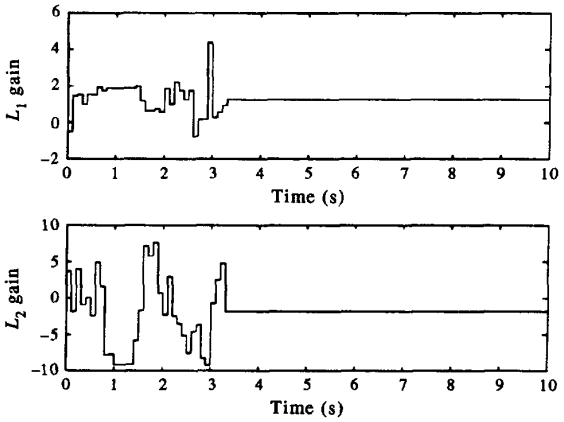

Fig. 2. L gains: linear system.

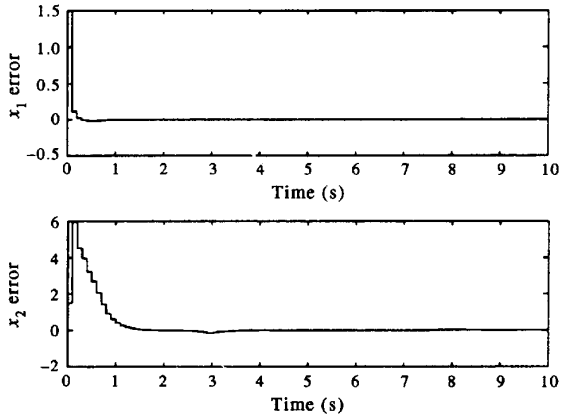

Fig. 3. x_1 and x_2 estimation errors.

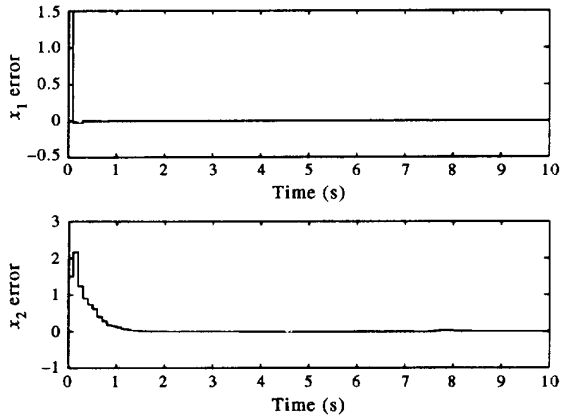

Fig. 5. x_1 and x_2 estimation errors: chaotic input.

The same $u(t)$ is chosen as in the previous example. Receiving the state x_1 as the process output ($y = x_1$), it is necessary to estimate x_2. The observer structure is initialized with $\hat{x}_1 = \hat{x}_2 = 0$. The process state is initialized at $x_1 = x_2 = 1.5$. Crossover and mutation probabilities are chosen as 0.6 and 0.4, respectively. A string length of 16 digits is chosen (eight to represent each L gain). Using the standard observer design procedure,[21] linearizations of the system around the initial conditions ($x_1 = 1.5$, $x_2 = 1.5$) indicate that L_1 should be positive and L_2 should be negative. The GA is "smart" enough to isolate the correct sign choices in the linear case but for the nonlinear case, it must be helped in its choice of L by providing a good initial population and reasonable limits on the maximum and minimum values of L. It is for this reason that L_1 is initialized to a random number between 9.9 and 0.0, and L_2 is initialized to a random number between -0.01 and -9.9. The γ value is set at 5×10^{-5}. After some simulation studies, the values of α_1 and α_2 are chosen to be 5 and 80, respectively. Figure 3 shows the estimation errors for x_1 and x_2. Figure 4 shows the L gains. At $t \approx 3.0$ s, the γ tolerance in equation (11) is satisfied. Overall, the GAO performs quite well.

The nonlinear example is continued with four more test cases, with the same GA parameters as the previous case:

(1) Input $u(t) = 0.999 + 0.42\cos(1.75t)$ which makes the system chaotic (Figs 5 and 6)
(2) Input from (1) with noise $0.1 \sin(2\pi 3t)$ added to the sensed process output (Figs 7 and 8)
(3) White, Gaussian noise (mean = 1.25, SD = 0.5) as an input to the system (Figs 9 and 10),
(4) Model mismatch where the actual process is described by

$$\dot{x}_1(t) = -0.8x_1(t)x_2^2(t) + u(t)$$

$$\dot{x}_2(t) = x_1(t)x_2^2(t) - x_2(t)$$

but the model of the process used by the GAO remains as in equation (18) (Figs 11 and 12). The input to the system is the same as in (3). This case represents the

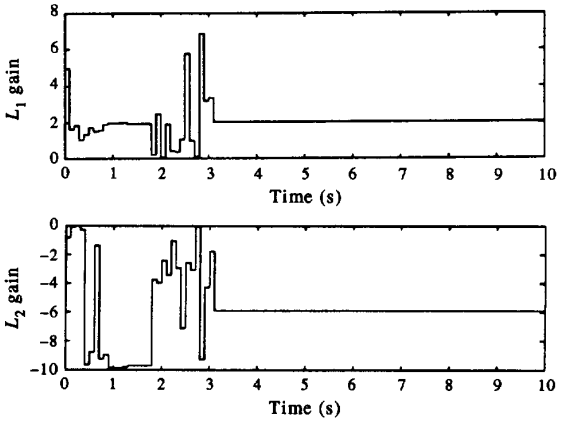

Fig. 4. L gains.

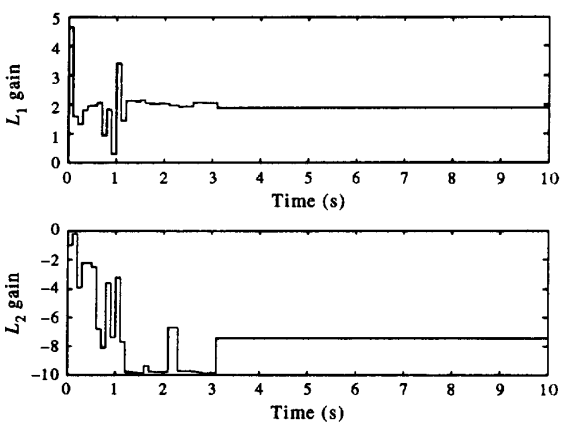

Fig. 6. L gains: chaotic input.

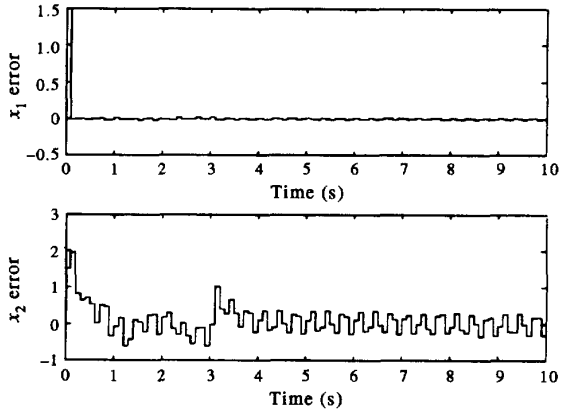

Fig. 7. x_1 and x_2 estimation errors: chaotic input with noise.

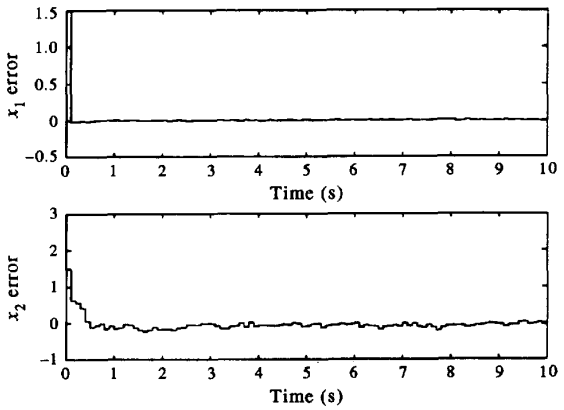

Fig. 9. x_1 and x_2 estimation errors: white, Gaussian noise.

more realistic situation, where the model of the process used in the GAO is not an accurate representation of the true process.

In all of these figures, reasonable estimates of x_1 and x_2 are obtained. The added noise to the sensed output and the model mismatch test cases show the largest amount of estimation error (see Figs 7 and 11). The technique seems to be highly dependent upon the accuracy of the process model, but for these examples did not seem to be as dependent upon the process input. For the input which makes the system chaotic (Figs 5 and 6), good estimation accuracy is still obtained. In fact, it was found that merely exciting the process with a constant input $[u(t) = 0.999]$ still resulted in the reduction of the error to zero for the linear and nonlinear systems. Finally, note that the final L_i gains found in each case were different, since the behavior of the nonlinear system changes significantly for the different cases studied. In addition the GAO is stochastic in operation and different L_i gains could be found for the same input to the process. As an interesting aside it was found that fixing the L_1 gain using the γ tolerance, but allowing L_2 to be picked by the GAO, still resulted in good performance, and fixing L_2, but allowing L_1 to change resulted in poor performance.

5. CONCLUDING REMARKS

While the results of this paper provide the first approach to using GAs for on-line state estimation and the simulation results indicate good performance for the applications studied, without a mathematical analysis of the GAO, neither convergence of the estimation error nor stability are guaranteed. All of the GAO parameters have an effect on the amount of estimation error in the system, and while the investigation found the design parameters of the GAO easy to choose for the linear system, the nonlinear system example required more careful tuning. The linear system seemed more tolerant to observer gain changes, and the GAO was able to find a "good" set of observer gains with less-stringent assumptions on the initial and subsequent populations than for the nonlinear system. The nonlinear system was very dependent upon the choice of γ and the α_is and large crossover/mutation probabilites were required to obtain good perfor-

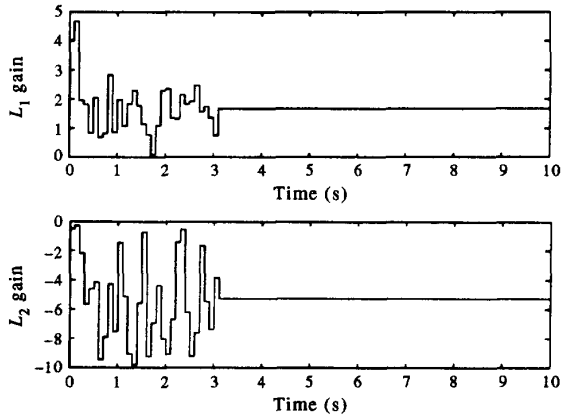

Fig. 8. L gains: chaotic input with noise.

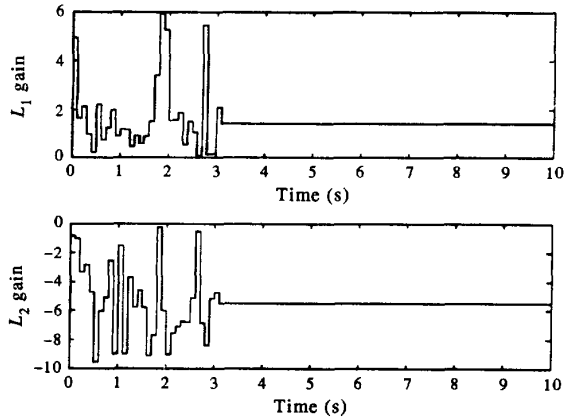

Fig. 10. L gains: white, Gaussian noise.

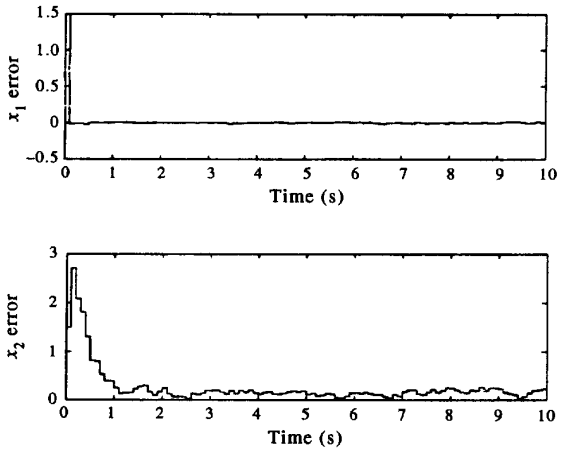

Fig. 11. x_1 and x_2 estimation errors: model mismatch.

mance. Poor choice of these GA parameters sometimes resulted in unstable behavior.

Several directions seem fruitful for future research:

(i) more-general observer structures should be employed rather than the simple linear structure used in this paper;
(ii) other termination conditions besides equation (11) should be investigated;
(iii) stability and convergence analysis must be performed; and
(iv) experimental studies are needed to more fully evaluate the implementation issues associated with the technique.

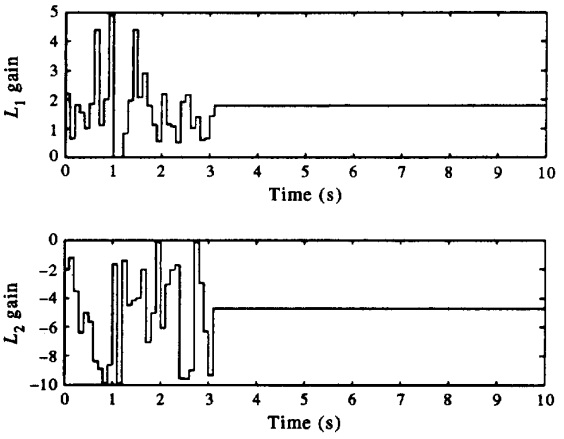

Fig. 12. L gains: model mismatch.

Acknowledgement—This work was supported in part by National Science Foundation Grants IRI 9210332 and EEC 9315257.

REFERENCES

1. Holland J. *Adaptation in Natural and Artificial Systems*. The University of Michigan Press, Ann Arbor (1975).
2. Goldberg D. E. *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison-Wesley, New York (1989).
3. Michalewicz Z. *Genetic Algorithms + Data Structures = Evolution Programs*. Springer-Verlag, New York (1992).
4. Lee M. A. and Takagi H. Integratng design stages of fuzzy systems using genetic algorithms. In *Second IEEE International Conference on Fuzzy Systems*, San Francisco, CA, pp. 612–617 (1993).
5. Porter B. and Borairi M. Genetic design of linear multivariable feedback control systems using eigenstructure assignment. *Int. J. Systems Sci.* **23**, 1387–1390 (1992).
6. Michelewicz Z. *et al.* Genetic algorithms and optimal control problems. In *Proceedings of the 29th Conference on Decision and Control*, Honolulu, HI, pp. 1664–1666 (1990).
7. Ishibuchi H., Nozaki K. and Yamamoto N. Selecting fuzzy rules by genetic algorithm for classification problems. In *Second IEEE International Conference on Fuzzy Systems*, San Francisco, CA, pp. 1119–1124 (1993).
8. Katai O., Ida M., Sawaragi T., Iwai S., Kohno S. and Kataoka T. Constraint-oriented fuzzy control schemes for cart-pole systems by goal decoupling and genetic algorithms. In *Fuzzy Control Systems* (Kandel A. and Langholz G. Eds), pp. 181–195. CRC Press, Boca Raton (1994).
9. Karr C. and Gentry E. Fuzzy control of ph using genetic algorithms. *IEEE Transact. Fuzzy Systems* **1**, 46–53 (1993).
10. Park D., Kandel A. and Langholz G. Genetic-based new fuzzy reasoning models with application to fuzzy control. *IEEE Transact. Systems, Man Cybernetics* **24**, 39–47 (1994).
11. Varšek A., Urbaničič T. and Filipič B. Genetic algorithms in controller design and tuning. *IEEE Transact. Systems, Man Cybernetics* **23**, 1330–1339 (1993).
12. Nomura H., Hayashi I. and Wakami N. A self-tuning method of fuzzy reasoning by genetic algorithm. In *Fuzzy Control Systems* (Kandel A. and Langholz G. Eds), pp. 338–354. CRC Press, Boca Raton (1994).
13. Das R. and Goldberg D. Discrete-time parameter estimation with genetic algorithms. In *Proceedings 19th Annual Pittsburgh Conf. Modeling Simulation*, Pittsburgh, PA, pp. 2391–2395 (1988).
14. Maclay D. and Dorey R. Applying genetic search techniques to drivetrain modeling. *IEEE Control Systems*, **13**, 50–55 (1993).
15. Kristinsson K. and Dumont G. System identification and control using genetic algorithms. *IEEE Transact. Systems, Man, Cybernetics* **22**, 1033–1046 (1992).
16. Etter D., Hicks M. and Cho K. Recursive adaptive filter design using adaptive genetic algorithm. In *Proc. IEEE Int. Conf. Acoustics, Speech, Signal Processing* **2**, 635–638 (1982).
17. Yao L. and Sethares W. A. Nonlinear parameter estimation via the genetic algorithm. *IEEE Transact. Signal Processing* **42**, 927–935 (1994).
18. Porter II L. L. and Passino K. M. Genetic model reference adaptive control. In *Proc. IEEE Int. Symp. Intelligent Control*, Columbus, OH, pp. 219–224, 16–18 August (1994).
19. Narendra K. S. and Annaswamy A. M. *Stable Adaptive Systems*, pp. 140–180. Prentice-Hall, New Jersey (1989).
20. Srinivas M. and Patnaik L. M. Genetic algorithms: A survey. *IEEE Comput.* pp. 17–26 (1994).
21. Frankliin G. F., Powell J. D. and Workman M. L. *Digital Control of Dynamic Systems*, p. 252. Addison Wesley, Reading, MA (1990).
22. Holden A. V. *Chaos*. Princeton University Press, New Jersey (1986).

AUTHORS' BIOGRAPHIES

La Moyne L. Porter II received a B.S. in Chemical Engineering and Electrical Engineering at Stanford University in 1991 and an M.S. in Electrical Engineering at The Ohio State University in 1994. He has worked at British Petroleum Research configuring a graphical interface for control of a chemical reactor, and at Intel doing research in a manufacturing environment. He is currently a graduate student at Stanford University pursuing a Ph.D. His interests include English literature, nonlinear systems, adaptive control and intelligent control.

Kevin M. Passino received his M.S. and Ph.D. in Electrical Engineering from the University of Notre Dame in 1989 and his B.S.E.E. from Tri-State University in 1983. He has worked in the control systems group at Magnavox Electronic Systems Co., Ft Wayne, IN, on research in missile control and at McDonnell Aircraft Co., St Louis, MI, on research in flight control. He spent a year at Notre Dame as a Visiting Assistant Professor and is currently an Assistant Professor in the Department of Electrical Engineering at The Ohio State University. He is an Associate Editor for the *IEEE Transactions on Automatic Control*; served as the Guest Editor for the *1993 IEEE Control Systems Magazine* Special Issue on Intelligent Control; is currently a Guest Editor for a special track of papers on Intelligent Control for *IEEE Expert Magazine*; and is on the Editorial Board of the *International Journal for Engineering Applications of Artificial Intelligence*. He was a Program Chairman for the *8th IEEE Int. Symp. on Intelligent Control*, 1993 and he is serving as the General Chair for the *11th IEEE Int. Symp. on Intelliigent Control*. He is co-editor (with P. J. Antsaklis) of the book *An Introduction to Intelligent and Autonomous Control*, Kluwer Academic Press, 1993. He is a member of the IEEE Control Systems Society Board of Governors. His research interests include intelligent and autonomous control techniques, nonlinear analysis of intelligent control systems, failure detection and identification systems, and scheduling and stability analysis of flexible manufacturing systems.