PERGAMON

Engineering Applications of Artificial Intelligence 13 (2000) 611–623

Engineering Applications of
ARTIFICIAL
INTELLIGENCE

www.elsevier.com/locate/engappai

Genetic adaptive state estimation[☆]

James R. Gremling, Kevin M. Passino*

Department of Electical Engineering, The Ohio State University, 2015 Neil Avenue, Columbus, OH 43210, USA

Abstract

A genetic algorithm (GA) uses the principles of evolution, natural selection, and genetics to offer a method for parallel search of complex spaces. This paper describes a GA that can perform on-line adaptive state estimation for linear and nonlinear systems. First, it shows how to construct a genetic adaptive state estimator where a GA evolves the model in a state estimator 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 genetic adaptive state estimator. Its performance is compared to that of the conventional adaptive Luenberger observer for two linear system examples. Next, a genetic adaptive state estimator is used to predict when surge and stall occur in a nonlinear jet engine. Our main conclusion is that the genetic adaptive state estimator has the potential to offer higher performance estimators for nonlinear systems over current methods. © 2000 Elsevier Science Ltd. All rights reserved.

Keywords: Estimation; Genetic algorithms; Jet engine surge/stall control

1. Introduction

A Luenberger observer is used to estimate the state of a linear system (plant), given a model of the linear system (that is, assuming known plant parameters), and it can perform properly given that certain assumptions about the structure of the plant hold (for example, that the state variables are observable). A conventional method for state estimation for a plant whose exact parameters are *not* known is the adaptive Luenberger observer, described in Ioannou and Sun (1996). This observer tunes its estimates of the plant parameters on-line, in addition to its estimates of the states of the plant. A drawback of the adaptive Luenberger observer is that it is primarily useful for linear plants whose model structures are in a specific form. If a linear plant has a model that is not in the required form, a transformation can easily be used to put the model in the proper form. Unfortunately, this may result in transformed states that have an unclear physical meaning (if any physical meaning at all), or that are inadequate for developing a useful control law.

The genetic algorithm (GA) has been used in a large number of applications, in areas ranging from economics and game theory to control system design. It is a stochastic process which attempts to find an optimal solution for a problem by using techniques that are based on Mendel's genetic inheritance concept and Darwin's theory of evolution and *survival of the fittest*. GAs have been used for parameter estimation and system identification (Yao and Sethares, 1994; Kristinsson and Dumont, 1992). The use of the GA for state estimation has been studied in Porter and Passino (1995), where the authors adapt the gains of a linear observer to estimate the state of a nonlinear system.

This paper introduces a new approach to adaptive state estimation where a genetic algorithm is used to estimate the plant parameters and hence to obtain an estimate of the state. The approach is significantly different from the one in Porter and Passino (1995) since there the authors had to assume a given model; here the model is estimated. Moreover, a new set of examples of state estimation, and a comparative analysis with conventional methods, is provided.

The goal of the first two examples is to show that the GA can perform adaptive state estimation in cases that are suitable for the adaptive Luenberger observer. This would support the use of the GA as an alternative method in cases where the Luenberger observer may be difficult to tune. In addition, it is shown that the GA can

[☆]This work was supported in part by National Science Foundation Grant EEC9315257.

*Corresponding author. Tel.: +1-614-292-5716; fax: +1-614-292-7596.

E-mail address: k.passino@osu.edu (K.M. Passino).

be used in cases where conventional methods may fail (nonlinear plants, for example). In particular, the paper shows how to develop a genetic adaptive state estimator for predicting surge and stall in a jet engine (an earlier and shorter version of this portion of this work is contained in Gremling and Passino (1997). Comparisons to the conventional extended Kalman filter (EKF) are made (for an introduction to this standard estimation approach see, Mendel (1995); it is beyond the scope of this paper to introduce such a standard well-known method); the EKF fails for this application while the genetic adaptive state estimator succeeds.

2. Relevant background: a base-10 genetic algorithm

In order for the GA to find the optimal solution to a particular problem, the parameters that comprise a solution must be encoded into a form upon which the GA can operate. To borrow a term from biology and genetics, any set of parameters which may be a solution to the given problem is called a *chromosome*, and the individual parameters in that possible solution are called *traits*. Since the GA will most likely be implemented on a digital computer, each trait must be encoded with a finite number of digits (called *genes*). The more genes in a given trait (or in a chromosome), the longer the GA will take for encoding and decoding purposes and in other operations, so a reasonable length should be chosen. The entire set of chromosomes (that is, the entire set of candidate solutions to the given problem) upon which the GA will operate is called a *population*.

Here, it is important to discuss the assignment of the individual genes. A particular gene can take any one of a given number of values (called *alleles*). The GA described in Goldberg (1989) is a base-2 GA, in which all traits are encoded as binary numbers and all genes may take a value of 0 or 1. For this study, however, a base-10 GA is used, in which each gene can take any value from 0 to 9 (an extra gene representing the sign $\pm$ may be required for each trait). The base-10 approach was chosen for this study because it simplified the encoding/decoding procedure, and it provided for easy and intuitive monitoring of the dynamics of the operation of the GA. Simplification of the encode/decode procedure is especially important here since we are using the GA in a real-time system where encoding and decoding must occur within the sampling interval. Note that none of the techniques presented after this point require the use of a base-10 GA — a base-2 GA could just as easily be used (as can be easily shown in simulation studies). In the discussion that follows, however, any mention of the GA implies a base-10 approach. Overall, we must emphasize that it is not the objective of this paper to determine the best GA to solve the state estimation problem. Our objective is to show how to use a GA to solve the problem. Future work needs to focus on convergence, robustness, and effects of GA mechanics on estimation performance as this is beyond the scope of this work.

To evolve the best solution candidate (or chromosome), the GA employs the *genetic operators* of *selection*, *crossover*, and *mutation* for manipulating the chromosomes in a population. A brief description of these operators follows, and a more detailed description can be found in Goldberg (1989), Michalewicz (1992) and Srinivas and Patnaik (1994). The GA uses these operators to combine the chromosomes of the population in different arrangements, seeking a chromosome that maximizes some user-defined objective function (called the *fitness function*). This combination of the chromosomes results in a new population (that is, the next *generation*). The GA operates repetitively, with the idea that, on average, the members of the population defining the current generation should be as good (or better) at maximizing the fitness function than those of the previous generation. The most fit member of the current generation (that is, the member with the highest fitness function result, or "fitness value") at the time the GA terminates is often taken to be the solution of the GA optimization problem.

The first genetic operator used by the GA for creating a new generation is *selection*. To create two new chromosomes (or *children*) two chromosomes must be selected from the current generation as *parents*. As is seen in nature, those members of the population most likely to have a chance at *reproducing* are the members that are the most fit. The technique used in Goldberg (1989) for selection uses a "roulette wheel" approach. Consider a roulette wheel that is partitioned according to the fitness of each chromosome. The fitter chromosomes occupy a greater portion of the wheel and are more likely to be selected for reproduction. In Yao and Sethares (1994), a selection method is chosen so that a given segment of the population corresponding to the most fit members (that is, the D most fit members) are automatically selected for reproducing. Therefore, the least-fit members have *no* chance of contributing any genetic material to the next generation. Of the D most fit members of the current population, parents are randomly chosen, with equal probability. The latter method of selection is used in this study.

Once two parents have been selected, the *crossover* operation is used. Crossover mimics natural genetics (that is, "inheritance") in that it causes the exchange of genetic material between the parent chromosomes, resulting in two new chromosomes. Given the two parent chromosomes, crossover occurs with a user-defined probability p_c. According to Goldberg (1989), if crossover occurs, a randomly chosen "cross site" is determined. All genes from the cross site to the end of the chromosome are switched between the parent chromosomes, and the children are created. Another approach to crossover, one that is used in Yao and Sethares (1994) and in this study, is that crossover occurs exactly once (that is, $p_c = 1$) for every *trait*, with the cross site within that trait chosen randomly. That is, all genes between the cross site and the end of the trait are exchanged between the parent chromosomes. Crossover helps to seek for other solutions near to solutions that appear to be good.

After the children have been created, each child is subjected to the *mutation* operator. Mutation occurs on a gene-by-gene basis, each gene mutating with probability p_m. If mutation does occur, the gene that is to mutate will be replaced by a randomly chosen allele (in this case, a randomly chosen value between 0 and 9). The mutation operator helps the GA avoid a local solution to the optimization problem. If all of the members of a population should happen to converge to some local optimum, the mutation operator allows the possibility that a chromosome could be pulled away from that local optimum, improving the chances of finding the global optimum. However, since a high mutation rate results in a random walk through the GA search space, p_m should be chosen to be somewhat small. We have found, however, that in some instances in real-time systems, we need a slightly higher mutation rate. This is the case since the fitness function depends on the dynamically changing state of a system, so the locality of an optimum is time-dependent and we must ensure that the GA is readily capable of exploring new opportunities for maximizing the time-varying fitness functions.

If a chromosome is generated by crossover and mutation, it is possible that one or more of its traits will lie outside the allowable range(s). If this occurs, each trait that is out of range should be replaced with a randomly selected trait that does fall within the allowable range.

In addition to selection, crossover, and mutation, a fourth operator can be used by the GA. This operator, known as *elitism*, causes the single most fit chromosome of a population to survive, undisturbed, in the next generation. The motivation behind elitism is that after some sufficiently small amount of time, a candidate solution may be found to be close to the optimal solution. To allow manipulation of this candidate solution would risk unsatisfactory performance by the GA. Therefore, with elitism, the fitness of a population (seen as the fitness of the best member of the population) *should* be a nondecreasing function from one generation to the next. If elitism is selected, the most fit member of the current generation is automatically chosen to be a member of the next generation. The remaining members are generated by selection, crossover, and mutation. Notice, also, that this allows us to raise the mutation probabilty since we know that we have a good solution available. In Yao and Sethares (1994), as well as in this study, elitism can involve more than just one member. That is, a certain number ρD (possibly more than one) of the most fit members will survive in the next generation without manipulation by crossover or mutation. If the most fit member would point to a local optimum in the GA search space, but a slightly less fit member points to the global optimum, they might both survive in the next generation with this new form of elitism.

To initialize the GA, a chromosome length must be chosen, along with the length and position of each trait on the chromosome. The allowable range for each trait must also be specified. The population size (denoted N) must be specified, along with the method of generating the first population. The individual members may be randomly generated, or they may be initialized to some set of "best guesses". In this study, a randomly generated initial population is always used. In addition, p_c and p_m must be specified. After this initialization, the GA can operate freely to solve its optimization problem.

3. Genetic adaptive state estimation

Consider a general system

$$\dot{x} = M(x, u; \theta),$$
$$y = N(x, u), \tag{1}$$

where x is a vector describing the state of the system, u is the system input, y is the system output, θ is a vector containing the parameters that describe the system, and M and N are functions that relate x, u, y, and θ, and define the operation of the system.

Since the GA is a technique often implemented on a digital computer, it generally operates in discrete time. A more appropriate definition of the general system is then written as

$$x(k+1) = F(x(k), u(k); \theta),$$
$$y(k) = G(x(k), u(k)), \tag{2}$$

if θ is a time-invariant vector, which will be the case in the applications studied in this paper. Note that the functions F and G in Eq. (2) are not the same as M and N in Eq. (1). The GA-based parameter estimator assumes that the structures of F and G in Eq. (2) are known and operates on the equations

$$\hat{x}(k+1) = F(\hat{x}(k), u(k); \hat{\theta}(k)),$$
$$\hat{y}(k) = G(\hat{x}(k), u(k)), \tag{3}$$

where $\hat{\theta}(k)$ is the estimate of the true system parameter vector θ at time k, $\hat{x}(k)$ is the state of this estimated system, and $\hat{y}(k)$ is the output of the estimated system.

The way in which the GA operates on Eq. (3) is as follows. Consider the block diagram shown in Fig. 1. Each parameter estimate in the vector $\hat{\theta}(k)$ is encoded by

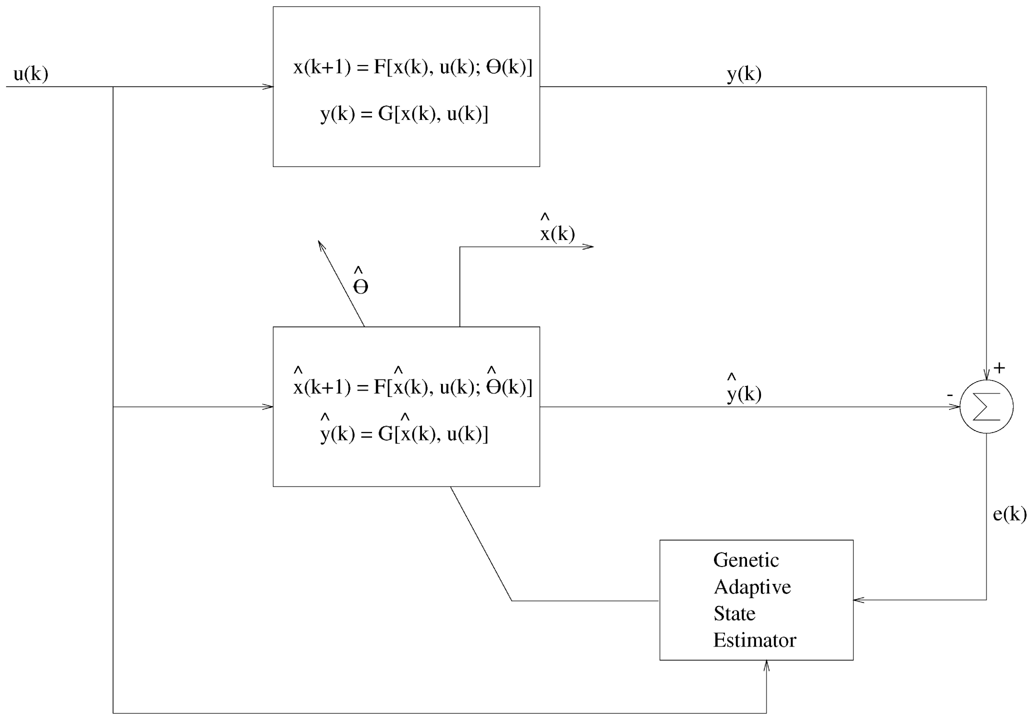

u(k)
x(k+1) = F[x(k), u(k); Θ(k)]
y(k) = G[x(k), u(k)]
y(k)
x̂(k)
Θ̂
x̂(k+1) = F[x̂(k), u(k); Θ̂(k)]
ŷ(k) = G[x̂(k), u(k)]
ŷ(k)
+
-
e(k)
Genetic Adaptive State Estimator

Fig. 1. Block diagram for a general GA-based state estimator.

the GA as a *trait* on a chromosome. Therefore, each chromosome completely represents a set of parameter estimates $\hat{\theta}(k)$, and hence it completely represents an estimated system (since the structures of F and G are assumed to be known). Since the GA works with a *population* of chromosomes, the block in Fig. 1 representing the estimated system can be thought of as a population of candidate systems. The GA employs the selection, crossover, mutation, and elitism operators on this population of candidate systems to evolve a system that best represents the true system. At any time k, the current set of parameter estimates $\hat{\theta}(k)$ will be provided by the *most fit* chromosome in the population at time k (that is, the chromosome with the highest *fitness value* at time k).

In this case, the fitness function for the GA is chosen to *minimize* the squared error between the output $y(k)$ of the actual system and that of the estimated system $\hat{y}(k)$ over a window of the $W+1$ most recent data points. However, since the *maximally* fit member is sought, the fitness function takes the form

$$J = \alpha - \tfrac{1}{2}E^{\mathrm{T}}E, \tag{4}$$

where

$$E = \begin{bmatrix} e_{k-W} \\ e_{k-W+1} \\ \vdots \\ e_k \end{bmatrix} \tag{5}$$

with $e_k = y(k) - \hat{y}(k)$ for $k > 0$ and $e_k = 0$ for $k \le 0$. To guarantee that each member has a positive fitness value, α is selected to be the highest $\frac{1}{2}E^{\mathrm{T}}E$ of any member of the population at time k (that is, the $\frac{1}{2}E^{\mathrm{T}}E$ of the worst member of the population).

The GA uses the current population of parameter estimates, along with the past and present values of the input $u(k)$ and output $y(k)$, to generate past and present estimated system states $\hat{x}(k)$ (that is, if it has data $y(k')$ available it uses it rather than $\hat{y}(k')$ to estimate the state since it is more accurate). A window of $\hat{y}$ values (that is, $\hat{y}(k-W), \hat{y}(k-W+1), \ldots, \hat{y}(k)$) is generated for each candidate set of parameter estimates (that each specify a different "identifier model" (Ioannou and Sun, 1996)), and hence each candidate is assigned a fitness value according to Eqs. (4) and (5). Using these fitness values, the GA is able to select which candidates will be used as parents for generating the next population (or *generation*) of candidate sets of parameter estimates.

3.1. Computational issues

In order to assess the possibility of implementing an algorithm (such as the genetic adaptive state estimator) in real-time, computational complexity must be examined. For any given generation, let n_{a} represent the number of "add" operations that must take place. Also, let n_{m}, r_{i}, and r_{f} represent the numbers of "multiply" operations, random integer generations, and random floating-point number generations that must take place, respectively. The values for n_a and n_m are of primary interest in determining the complexity of the fitness calculation procedure, while r_i and r_f are of primary interest in terms of generating the next population of chromosomes.

First, the focus is on fitness calculation. When looking at the $W+1$ most recent data points, each population member must undergo $W+1$ add operations, which will correspond to a calculation of the error between the actual system output and the estimated system output for each data point. Then, $W+1$ multiply operations will be carried out, as the *squared* error is of sole interest here. All of the squared errors over the window will be added together, resulting in another W add operations. This leads to a total of $2W+1$ add operations and $W+1$ multiply operations for each member (in a population of size N) to generate E^TE. (Therefore, $W+2$ mulitiply operations are necessary for $\frac{1}{2}E^TE$ calculation for each member.)

Each member is examined, and after α is determined, N add operations are implemented to perform the remainder of the $J=\alpha-\frac{1}{2}E^TE$ calculations for the population. In all,

$$n_a = 2N(W+1),$$

$$n_m = N(W+2), \qquad (6)$$

are the numbers of add and multiply operations, respectively, that must take place for a population at any given generation. Depending on the encoding and decoding procedures used for chromosome manipulation, these numbers require some adjustment.

In order to determine how many random-number generations must take place, we must look at the remainder of the operation of the GA (that is, selection, crossover, and mutation). Since every two children are generated by two randomly chosen parents (from the pool of the best D population members), each child can be thought of as the result of one random selection from the pool of D possible parents, on average. Recalling that ρD members survive in the next population by elitism, $N-\rho D$ random integer generations then take place due to parent selection. For every two children generated, n_t crossover operations will occur, where n_t is the number of traits on a chromosome. This means that $(n_t/2)(N-\rho D)$ random integer generations will occur due to crossover operations. Finally, the number of mutation operations taking place in a given population can range from zero to $n_g(N-\rho D)$, with an average value of $p_m n_g(N-\rho D)$, where n_g is the number of genes per chromosome. Therefore, the number of random integer generations r_i needed to produce a new population lies in a range

$$\left(\frac{n_t}{2}+1\right)(N-\rho D) \le r_i \le \left(\frac{n_t}{2}+n_g+1\right)(N-\rho D) \qquad (7)$$

with an average value

$$\bar{r}_i = \left(\frac{n_t}{2}+p_m n_g+1\right)(N-\rho D). \qquad (8)$$

Since it is possible that mutation or crossover will lead to traits that lie outside the allowable ranges, the GA may be required to replace faulty traits with new, randomly generated traits. This will result in the generation of random floating-point numbers. The number of random floating-point number generations r_f will lie in a range

$$0 \le r_f \le n_t(N-\rho D). \qquad (9)$$

To examine the operation of a genetic adaptive state estimator, consider the following examples. First, we consider two linear examples and compare the performance of the genetic adaptive observer and the conventional adaptive Luenberger observer. Following this, we study a physically motivated jet engine estimation problem where we compare the performance of the genetic adaptive state estimator to an extended Kalman filter.

4. A linear example

Consider the third-order linear system

$$\dot{x} = Ax + Bu,$$

$$y = Cx,$$

$$x = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}, \qquad (10)$$

where

$$A = \begin{bmatrix} -2.1 & 1 & 0 \\ -2.1 & 0 & 1 \\ -1 & 0 & 0 \end{bmatrix},$$

$$B = \begin{bmatrix} 1 \\ 0.9 \\ 0.8 \end{bmatrix},$$

$$C = [1 \quad 0 \quad 0] \qquad (11)$$

are constant but unknown system matrices. The poles of this system are -1 and -0.55 ± 0.8352j, and the zeros are -0.45 ± 0.773j, corresponding to a minimum phase system. Note that the system output y is also the state x_1. For this example x_1 is the only measurable state. In some instances (for example, state feedback control) we would like to have an estimate of x_2 and x_3. Next, we outline a conventional approach to adaptive state estimation. Following this, we will introduce a method for genetic adaptive state estimation.

4.1. Adaptive Luenberger observer

As mentioned earlier, a conventional technique that is useful for state estimation is the adaptive Luenberger observer, which can also simultaneously estimate the parameters of the plant model (Ioannou and Sun, 1996). However, the plant model must satisfy a specific structural constraint. This constraint is that the model must be in *observer form*

$$\dot{x} = \begin{bmatrix} & \vdots & I_{n-1} \\ -a_{\mathrm{p}} & \vdots & \cdots \\ & \vdots & 0 \end{bmatrix} x + b_{\mathrm{p}} u,$$

$$y = [1 \quad 0 \quad \ldots \quad 0]x, \tag{12}$$

where

$$a_{\mathrm{p}} = \begin{bmatrix} a_{n-1} \\ a_{n-2} \\ \vdots \\ a_0 \end{bmatrix},$$

$$b_{\mathrm{p}} = \begin{bmatrix} b_{n-1} \\ b_{n-2} \\ \vdots \\ b_0 \end{bmatrix} \tag{13}$$

and n is the order of the system. The system described by Eqs. (10) and (11) satisfies this constraint, with $n = 3$, $a_{\mathrm{p}} = [2.1, 2.1, 1]^{\mathrm{T}}$, and $b_{\mathrm{p}} = [1, 0.9, 0.8]^{\mathrm{T}}$.

The adaptive Luenberger observer takes the form

$$\dot{\hat{x}} = \begin{bmatrix} & \vdots & I_{n-1} \\ -\hat{a}_{\mathrm{p}}(t) & \vdots & \cdots \\ & \vdots & 0 \end{bmatrix} \hat{x} + \hat{b}_{\mathrm{p}}(t)u + (a^* - \hat{a}_{\mathrm{p}}(t))(y - \hat{y}),$$

$$\hat{y} = [1 \quad 0 \quad \ldots \quad 0]\hat{x}, \tag{14}$$

where $\hat{x}$ is the estimate of the state x, $\hat{a}_{\mathrm{p}}(t)$ and $\hat{b}_{\mathrm{p}}(t)$ are the estimates of the vectors a_{p} and b_{p}, respectively, at time t, and a^* is chosen so that

$$A^* = \begin{bmatrix} & \vdots & I_{n-1} \\ -a^* & \vdots & \cdots \\ & \vdots & 0 \end{bmatrix} \tag{15}$$

is a stable matrix (that is its eigen values are all in the left half-plane).

Defining a vector $\hat{\theta}$ to consist of the estimated parameters (that is, $\hat{\theta} = [\hat{b}_{\mathrm{p}}^{\mathrm{T}}(t), \hat{a}_{\mathrm{p}}^{\mathrm{T}}(t)]^{\mathrm{T}}$) and using a gradient method following (Ioannou and Sun, 1996), an adaptive law for determining the $\hat{\theta}$ parameters is given by

$$\dot{\hat{\theta}} = \Gamma \varepsilon \phi,$$

$$\varepsilon = \frac{z - \hat{z}}{m^2},$$

$$\hat{z} = \hat{\theta}^{\mathrm{T}} \phi,$$

$$\phi = \left[\frac{\alpha_{n-1}^{\mathrm{T}}(s)}{\Lambda(s)} u, -\frac{\alpha_{n-1}^{\mathrm{T}}(s)}{\Lambda(s)} y \right]^{\mathrm{T}},$$

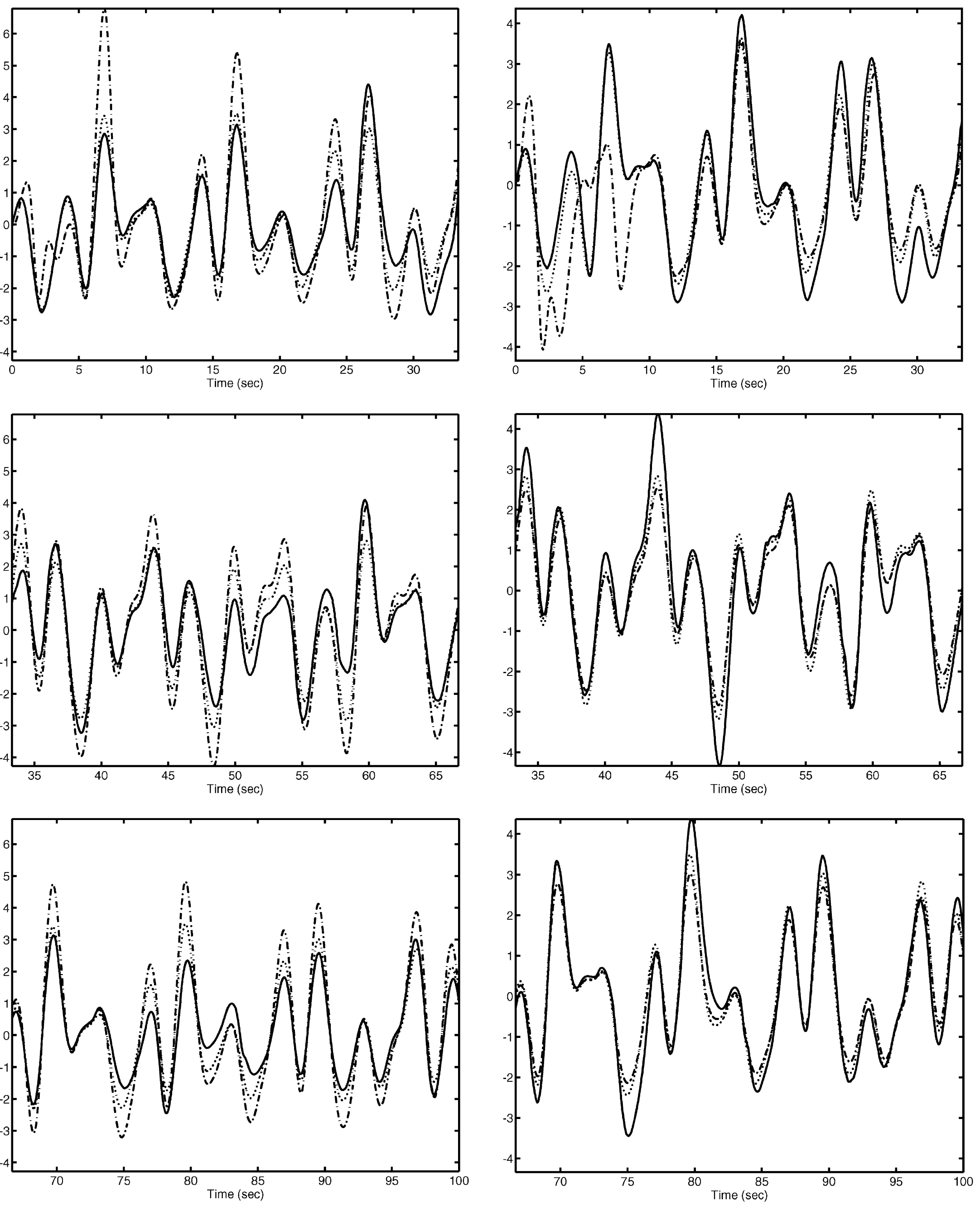
$$z = \frac{s^n}{\Lambda(s)} y, \tag{16}$$

where Γ is a symmetric, positive-definite matrix, $\Lambda(s)$ is a monic Hurwitz polynomial of degree n, and the vector $\alpha_{n-1}(s)$ is defined by $\alpha_{n-1}(s) \triangleq [s^{n-1}, \ldots, s, 1]^{\mathrm{T}}$. The parameter m^2 of the adaptive law can be chosen as $m^2 = 1$ or $1 + \phi^{\mathrm{T}}\phi$. Note that the adaptive observer is implemented in continuous time.

For the system described by Eqs. (10) and (11), the adaptive Luenberger observer was simulated for 100 s, with all states initially at zero. For the simulation, the following choices were made: $m^2 = 1$, $\hat{a}_{\mathrm{p}}(0) = [3, 3, 1]^{\mathrm{T}}$, $\hat{b}_{\mathrm{p}}(0) = [1, 0, 0]^{\mathrm{T}}$, $a^* = [12, 47, 60]^{\mathrm{T}}$, and $\Lambda(s) = s^3 + 3s^2 + 3s + 1$. The choices for a^* and $\Lambda(s)$ were made simply to satisfy the constraints placed on them, while $\hat{a}_{\mathrm{p}}(0)$ was chosen so that the initial estimates would indicate a system with stable poles, and $\hat{b}_{\mathrm{p}}(0)$ was chosen in an attempt to avoid making an initial guess about the location of the system zeros — this choice places all possible zeros of the estimated system at the origin. To meet the constraints on Γ, the following matrix was used: $\Gamma = \tilde{\alpha} I$ ($\tilde{\alpha} > 0$), and after some tuning, the value $\tilde{\alpha} = 1.5$ was chosen.

To guarantee that the input was a sufficiently rich signal (Ioannou and Sun, 1996), the following was used: $u = A_{s_1}\sin(\omega_1 t) + A_{s_2}\cos(\omega_2 t) - A_{s_3}\sin(\omega_3 t) - A_{s_4}\cos(\omega_4 t)$, with $A_{s_1} = 1.5$, $A_{s_2} = 3.9$, $A_{s_3} = 2.6$, $A_{s_4} = 2.1$, $\omega_1 = 3.2$, $\omega_2 = 1.9$, $\omega_3 = 0.7$, and $\omega_4 = 2.5$. Note that while for the linear case the theory of persistency of excitation is well developed (see, for example, Ioannou and Sun, 1996) so we know how to pick the input in this case to ensure convergence, for nonlinear estimation some type of sufficiency of excitation is required but no theory is presently developed so we cannot know if our signal is sufficiently rich. Hence, for the jet engine problem considered later in this paper we cannot be guaranteed convergence (but we also cannot be guaranteed convergence for any other method).

In this example, only the state estimates $\hat{x}_2$ and $\hat{x}_3$ were of interest. The performance of the adaptive Luenberger observer for estimating these states is shown in Fig. 2, where the adaptive Luenberger observer is compared to the GA. Note that in Fig. 2 we perform the simulation for 100 s and split the plots for $\hat{x}_2$ and $\hat{x}_3$ into



(a) Estimation of x_2. (b) Estimation of x_3.

Fig. 2. GA vs. adaptive Luenberger observer for minimum phase linear system state estimation. The dotted lines indicate the actual state, while the solid and dash−dot lines indicate the state estimates from the GA and adaptive Luenberger observer, respectively.

three different plots so that the results are easier to inspect.

4.2. Genetic adaptive state estimation

Although the adaptive Luenberger observer can be implemented in continuous time, the GA must use a discrete-time approximation to the system under investigation. Using a forward-looking difference, the system model given in Eq. (12) can be written as

$$x(k) = x(k-1) + T\begin{bmatrix} & \vdots & I_{n-1} \\ -a_{\mathrm{p}} & \vdots & \dots \\ & \vdots & 0 \end{bmatrix} x(k-1)$$
$$+ Tb_{\mathrm{p}}u(k-1),$$

$$y(k) = [1 \quad 0 \quad \dots \quad 0]x(k),$$

$$x(k) = \begin{bmatrix} x_1(k) \\ x_2(k) \\ x_3(k) \end{bmatrix}, \tag{17}$$

where a_p and b_p have already been defined for this example.

The GA-estimated system model can then be written as

$$\hat{x}(k) = \hat{x}(k-1) + T\begin{bmatrix} & \vdots & I_{n-1} \\ -\hat{a}_p(k-1) & \vdots & \dots \\ & \vdots & 0 \end{bmatrix}\hat{x}(k-1) + T\hat{b}_p(k-1)u(k-1),$$

$$\hat{y}(k) = [1 \quad 0 \quad \dots \quad 0]\hat{x}(k),$$

$$\hat{x}(k) = \begin{bmatrix} \hat{x}_1(k) \\ \hat{x}_2(k) \\ \hat{x}_3(k) \end{bmatrix}. \tag{18}$$

In this case, the GA is actively estimating the parameters in a_p and b_p while using these parameter estimates in Eq. (18) to update the state estimates $\hat{x}(k)$.

Since the fitness function for the GA relies on an error $e(k) = y(k) - \hat{y}(k)$, and since $y(k) = x_1(k)$ and $\hat{y}(k) = \hat{x}_1(k)$, there is no guarantee that the state estimates $\hat{x}_2(k)$ and $\hat{x}_3(k)$ will accurately track the true states $x_2(k)$ and $x_3(k)$, respectively. However, due to the dependence of each state on the other two states (seen in Eqs. (17) and (18)), it still seems possible that acceptable state tracking will occur.

After an ad hoc tuning of the GA (as there is no defined procedure for such tuning), the following parameters were used: $N = 200$, $p_m = 0.2$, $D = 70$, $\rho D = 10$, $W = 50$, and $g_t = 10$, where g_t is the number of generations executed per sampling period. Note the large population size N. This was necessary because six parameters were being estimated (a_2, a_1, a_0, b_2, b_1, and b_0), implying a large parameter search space.

The ranges used for $\hat{a}_2$, $\hat{a}_1$, $\hat{a}_0$, $\hat{b}_2$, $\hat{b}_1$, and $\hat{b}_0$ were $[1.8, 2.3]$, $[1.8, 2.3]$, $[0.8, 1.2]$, $[0.8, 1.2]$, $[0.7, 1.1]$, and $[0.6, 1.1]$, respectively. This would indicate that the knowledge of the system parameters was somewhat accurate, although the parameters were not known exactly. In many cases this is true — a system model and its parameters may be known to a reasonable degree of accuracy, but the possibility of system or environmental variations can limit that accuracy. Consider, for example, a system whose parameters will vary slightly with temperature, pressure, or humidity. Those parameters would be best characterized by the ranges within which they are likely to vary.

The GA-based state estimator was simulated for 100 s, with a sampling period T of 0.01 s. The input was the same as that used for the adaptive Luenberger observer, and the estimated states were initialized to zero. The results are shown in Fig. 2 where the performance of the GA is compared with that of the adaptive Luenberger observer.

Table 1
Squared error sums for GA linear system state estimation

	$E_{s_{x_2}}$	$E_{s_{x_3}}$
Average	4328.79	2008.48
Minimum	1887.24	1327.38
Maximum	8557.97	3091.29
Shown in Fig. 2	4362.25	1946.64

To verify the validity of the results shown in Fig. 2, some method for quantifying a typical or average GA execution should be developed. Consider a "squared error sum", which is a summation, over an entire simulation, of the squared error between a state and its estimate. In other words,

$$E_{s_\kappa} = \sum_{k=1}^{m_g} (\kappa(k) - \hat{\kappa}(k))^2, \tag{19}$$

where κ is the state in question, $\hat{\kappa}(k)$ is the estimate of κ at time sample k, and m_g is the total number of time samples in the simulation.

Here, 50 simulations were executed, with squared error sums generated for $\hat{x}_2$ and $\hat{x}_3$ according to Eq. (19). The average, minimum, and maximum squared error sums over the 50 simulations are given in Table 1, along with the sums for the GA results shown in Fig. 2 and the error for the adaptive Luenberger observer for $\hat{x}_2$ is 4365.23 and for $\hat{x}_3$ is 3427.46. From the table, it is clear that the GA results shown in Fig. 2 could be considered an average or representative case. Examining the figure and the error sums, then, it appears that the GA performs as well as the adaptive Luenberger observer for this example, aside from some transient behavior that Luenberger observer has for $\hat{x}_2$. Indeed, this supports the GA as an alternative method for adaptive state estimation.

5. A linear example with a nonminimum phase zero

Now, consider a third-order linear system with the same model structure as in Eq. (10), but with parameters defined by

$$A = \begin{bmatrix} -2.9 & 1 & 0 \\ -2.91 & 0 & 1 \\ -0.959 & 0 & 0 \end{bmatrix},$$

$$B = \begin{bmatrix} 0 \\ 1 \\ -0.05 \end{bmatrix},$$

$$C = [1 \quad 0 \quad 0]. \tag{20}$$

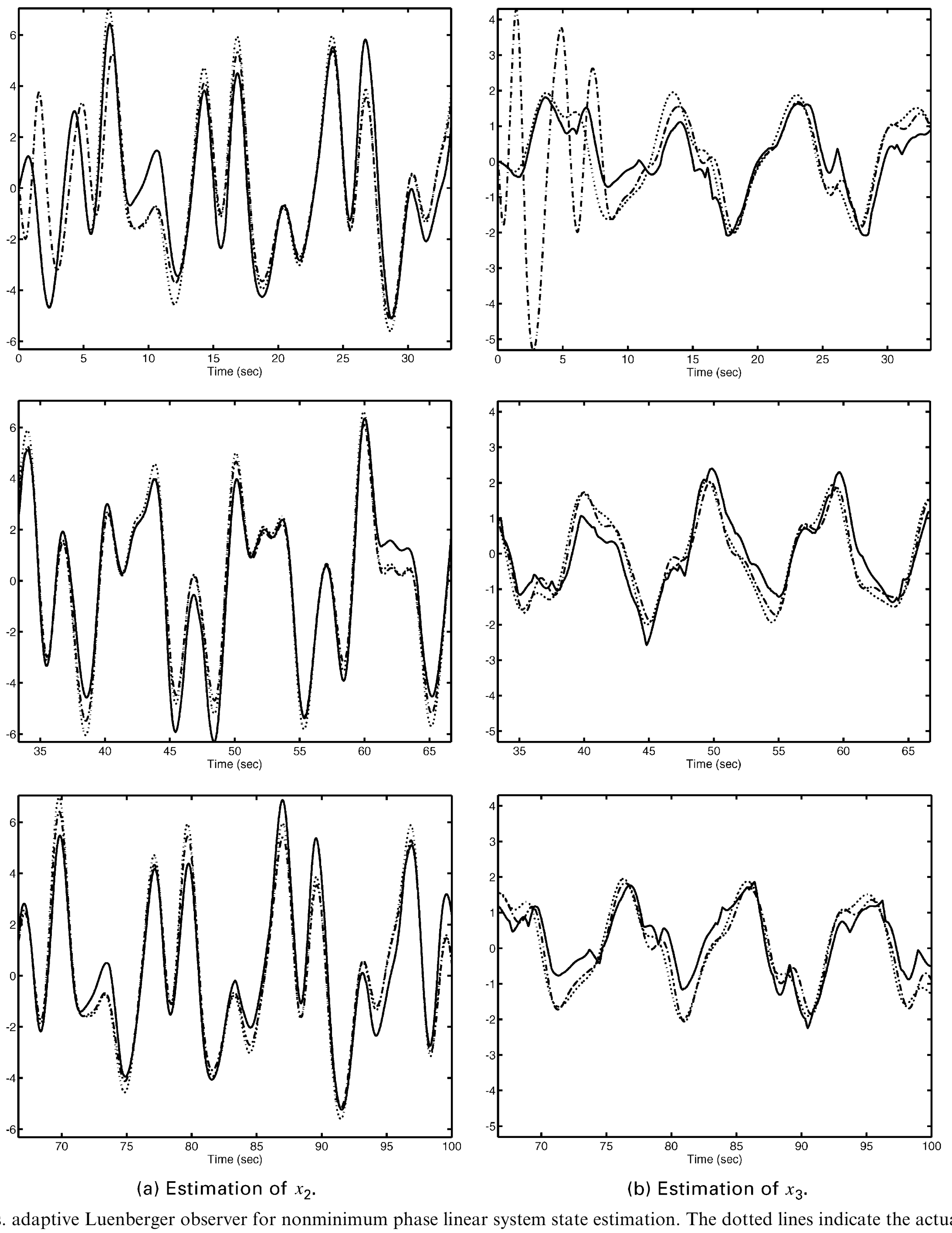


Fig. 3. GA vs. adaptive Luenberger observer for nonminimum phase linear system state estimation. The dotted lines indicate the actual state, while the solid and dash−dot lines indicate the state estimates from the GA and adaptive Luenberger observer, respectively.

This system has poles at -0.7 and $-1.1 \pm 0.4j$, and a zero at 0.05. This, of course, corresponds to a system that is not minimum phase. As in the previous example, the state x_1 is the only measurable state of the system, so it is desired that the states x_2 and x_3 can be estimated.

5.1. Adaptive Luenberger observer

Since this system is also in observer form, the adaptive Luenberger observer can be used for estimation of the states x_2 and x_3. Again, Eqs. (14) and (16) define the observer.

Using the same input as in the previous example, the adaptive Luenberger observer was simulated for 100 s. The same choices for $\hat{a}_p(0), \hat{b}_p(0), a^*$, and $\Lambda(s)$ used in the previous example were used in this case. Using $\Gamma = \tilde{\alpha}I(\tilde{\alpha} > 0)$, it was difficult to tune $\tilde{\alpha}$ so that the estimated parameters in $\hat{a}_p$ and $\hat{b}_p$ remained stable. Using $m^2 = 1 + \phi^T\phi$ instead of $m^2 = 1$, parameter stability was obtained, and $\tilde{\alpha} = 1.5$ was chosen, after

Table 2
Squared error sums for GA nonminimum phase linear system state estimation

	$E_{s_{x_2}}$	$E_{s_{x_3}}$
Average	7244.04	2563.16
Minimum	4596.32	1872.88
Maximum	10931.07	3706.12
Shown in Fig. 3	6602.56	2610.47

some tuning. The results of the simulation are shown in Fig. 3, where the performance of the adaptive Luenberger observer is compared to the GA. From the figure, it can be seen that the GA had some success with the state estimation, though slightly outperformed by the Luenberger observer.

5.2. Genetic adaptive state estimation

For estimating the states x_2 and x_3 of the nonminimum-phase linear system described by Eqs. (10) and (20), the GA used was nearly identical to that used in Section 4. The only difference is that the ranges used for $\hat{a}_2, \hat{a}_1, \hat{a}_0, \hat{b}_2, \hat{b}_1$, and $\hat{b}_0$ were [2.7,3.2], [2.7,3.2], [0.8,1.2], [−0.2,0.2], [0.7,1.2], and [−0.2,0.2], respectively. All other GA parameters were identical. The input used was identical to that in Section 4 (the same as that used by the adaptive Luenberger observer), and the estimated states were initialized to zero, as a 100-s simulation (with $T = 0.01$) was executed.

For validation purposes, 50 simulations were executed, and squared error sums were generated for $\hat{x}_2$ and $\hat{x}_3$. The results are given in Table 2. The results shown in Fig. 3 are characterized by squared error sums that are close to the average values, implying that those results represent a typical or average execution of the GA for this state-estimation problem (the squared error sums for the adaptive Luenberger observer were 6320.00 for $\hat{x}_2$ and 8348.10 for $\hat{x}_3$). From the figures, it can be seen that the GA, though still successful, was somewhat outperformed by the Luenberger observer (note the transient behavior that the Luenberger observer has for $\hat{x}_3$), providing some evidence to support the GA as a possible alternative tool for state estimation. Next, the use of the GA in a state estimator for a nonlinear system is studied.

6. A jet engine compressor

In this section we develop a genetic adaptive state estimator for a jet engine compressor model and compare its performance to that of the extended Kalman filter.

6.1. Nonlinear jet engine estimation problem

To achieve maximum performance from the engines used on current aircraft, those engines must often be operated near the boundaries of instability (that is, with reduced stall and surge margins, and hence reduced safety margins). As a result, problems such as *surge* and *rotating stall* may develop when a heavy demand is placed on an aircraft engine. Surge can be described as a large-amplitude oscillation in the mass flow through the engine, sometimes resulting in a momentary reverse in flow. When surge occurs, an aircraft may experience a phenomenon known as "flameout". Rotating stall consists of one or more localized regions or *cells* of reduced mass flow that will rotate around the circumference of the engine compressor. A rotating stall cell can grow in magnitude, and if it does so, it will grow in a circumferential manner. Clearly, problems such as surge and rotating stall are hazardous, and avoidance of these problems is always desired.

Moore and Greitzer developed a detailed, nonlinear model for surge and rotating stall in Moore and Greitzer (1986a). This model was simplified (through approximation) in McCaughan (1990) and expressed in a three-state form in Krstic and Kokotovic (1995). This three-state model is written as

$$\dot{\Phi}_J = -\Psi_J + \Psi_{C0} + 1 + \tfrac{3}{2}\Phi_J - \tfrac{1}{2}\Phi_J^3 - 3\Phi_J R_J,$$

$$\dot{\Psi}_J = \frac{1}{\beta^2}(\Phi_J - \Phi_T),$$

$$\dot{R}_J = \sigma R_J(1 - \Phi_J^2 - R_J), \qquad (21)$$

where Φ_J is the mass flow, Ψ_J is the pressure rise, $R_J \geq 0$ is the normalized stall cell squared amplitude, Φ_T is the mass flow through the throttle, and σ, β and Ψ_{C0} are constant parameters (with $\sigma > 0$ and $\beta > 0$). For the purpose of simulation, values for Ψ_{C0}, σ, and $1/\beta^2$ had to be determined. Since a realistic simulation was the goal, the values $\Psi_{C0} = 0.5, \sigma = 7$, and $1/\beta^2 = 0.4823$ were chosen, after consulting the experimental results in McCaughan (1990), Moore and Greitzer (1986b) and Greitzer (1976). To generate the control input $\Phi_T(k)$, a proportional-integral (PI) controller was used, taking as its input the error between $\Psi_J(k)$ and a reference signal, which is represented by the dash−dot lines in Figs. 4 and 5. The PI controller parameters used were $K_\mathrm{p} = 5$ and $K_\mathrm{i} = 2$. Note that the controller was implemented in discrete time.

It would be useful if some method for detecting surge and rotating stall existed. Surge would be evident in an oscillation of the state Φ_J, which represents the mass flow, and rotating stall activity would be evident in the behavior of the state R_J. The ability to detect the onset of either problem would result in the execution of

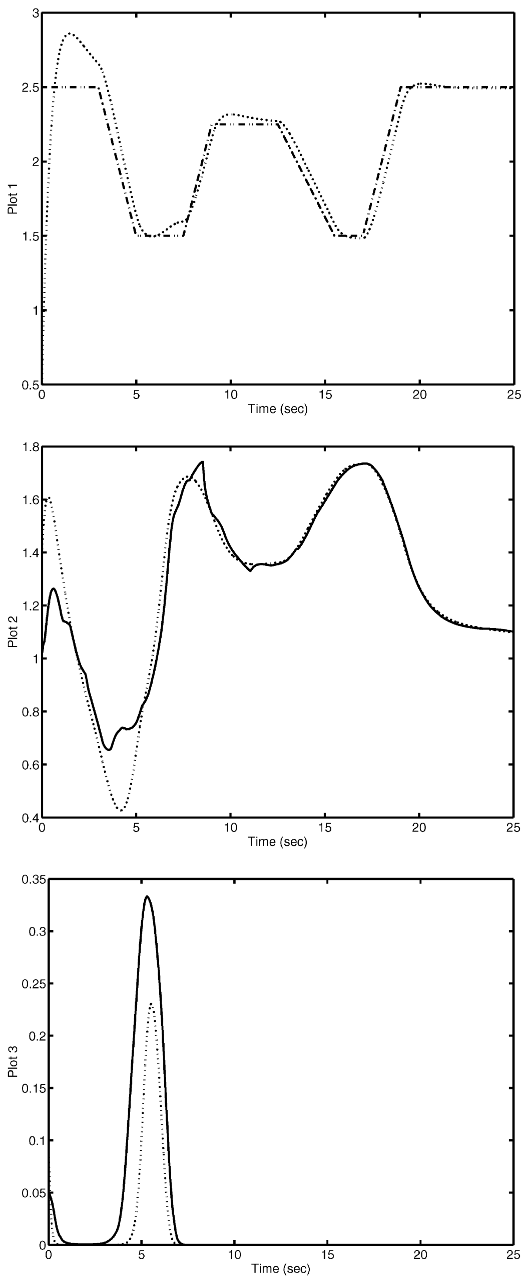


Fig. 4. GA-based state estimation for the jet engine compressor model, Case 1 ($\Psi_J(0) = 0.5$, $R_J(0) = 0.1$, and $\Phi_J(0) = 1.4$). Plot 1 corresponds to Ψ_J, plot 2 corresponds to Φ_J, and plot 3 corresponds to R_J. The dotted lines in the bottom two plots indicate the actual state, while the solid lines indicate the state estimates. The dash−dot line in plot 1 indicates the reference input for Ψ_J.

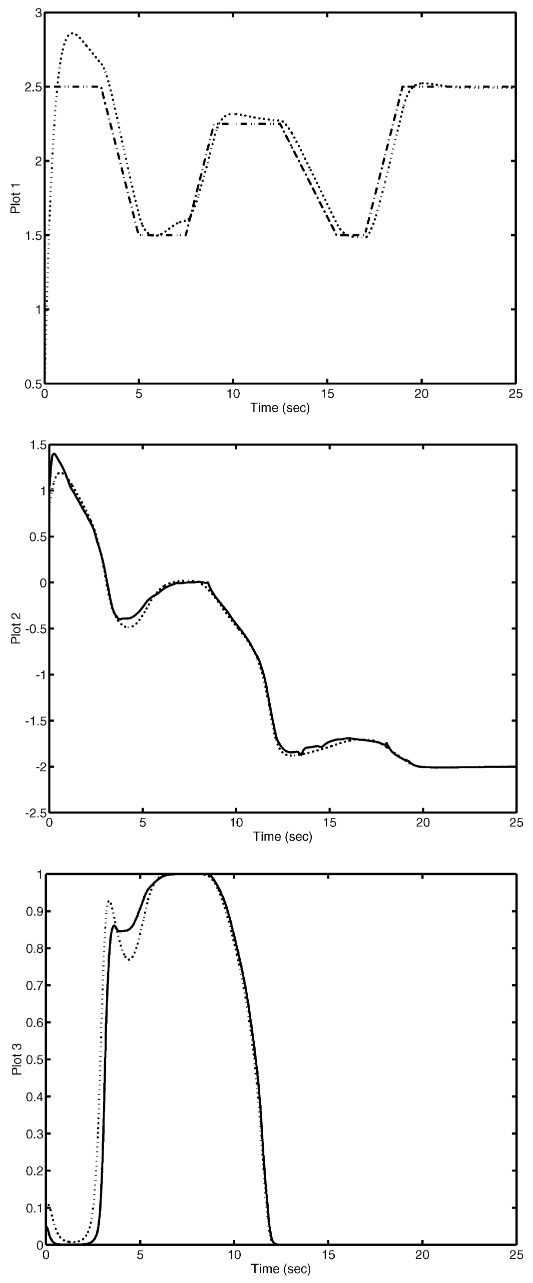


Fig. 5. GA-based state estimation for the jet engine compressor model, Case 2 ($\Psi_J(0) = 0.5$, $R_J(0) = 0.1$, and $\Phi_J(0) = 0.8$). Plot 1 corresponds to Ψ_J, plot 2 corresponds to Φ_J, and plot 3 corresponds to R_J. The dotted lines in the bottom two plots indicate the actual state, while the solid lines indicate the state estimates. The dash−dot line in plot 1 indicates the reference input for Ψ_J.

measures that could help prevent system failure. Unfortunately, neither of the states Φ_J and R_J is measurable during aircraft operation. The only measurable state is Ψ_J, so some method for estimating Φ_J and R_J from the available information (Ψ_J and Φ_T — the control input) is desired.

Since the system is neither linear nor in observer form, the adaptive Luenberger observer is not directly applicable. The goal now is to show that the GA can be used for estimating Φ_J and R_J. The compressor model of Eqs. (21) must first be approximated in discrete time, so using a forward-looking difference, the model can be written as

$$\begin{aligned}\Phi_J(k) = \Phi_J(k-1) + T(&-\Psi_J(k-1) + \Psi_{C0} + 1 \\ &+ \tfrac{3}{2}\Phi_J(k-1) - \tfrac{1}{2}\Phi_J^3(k-1) \\ &- 3\Phi_J(k-1)R_J(k-1)),\end{aligned} \tag{22}$$

$$\Psi_J(k) = \Psi_J(k-1) + \frac{T}{\beta^2}(\Phi_J(k-1) - \Phi_T(k-1)),$$

$$R_J(k) = R_J(k-1) + T\sigma R_J(k-1)(1 - \Phi_J^2(k-1) - R_J(k-1)).$$

As was done in the cases of Sections 4 and 5, the GA actively estimates parameters of the jet engine compressor model, and uses these parameter estimates along with Ψ_J and Φ_T to generate the estimates $\hat{\Phi}_J$ and $\hat{R}_J$. For this purpose, the equations used by the GA are

$$\begin{aligned}\hat{\Phi}_J(k) = \hat{\Phi}_J(k-1) + T(&-\hat{\Psi}_J(k-1) + \hat{c}_1(k-1) \\ &+ \hat{c}_2(k-1)\hat{\Phi}_J(k-1) - \hat{c}_3(k-1)\hat{\Phi}_J^3(k \\ &-1) - \hat{c}_4(k-1)\hat{\Phi}_J(k-1)\hat{R}_J(k-1)),\end{aligned} \tag{23}$$

$$\hat{\Psi}_J(k) = \hat{\Phi}_J(k-1) + T\hat{c}_5(k-1)(\hat{\Phi}_J(k-1) - \Phi_T(k-1)),$$

$$\hat{R}_J(k) = \hat{R}_J(k-1) + T\hat{c}_6(k-1)\hat{R}_J(k-1)(1 - \hat{\Phi}_J^2(k-1) - \hat{R}_J(k-1)),$$

where the estimates for $\Psi_{C0} + 1$ have been combined to form the single-parameter estimate $\hat{c}_1(k)$.

6.2. Genetic adaptive state estimator for a jet engine compressor model

The fitness function for the GA takes the same windowed data approach found in Eq. (4), with $e_k = \Psi_J(k) - \hat{\Psi}_J(k)$ for $k > 0$ and $e_k = 0$ for $k \leq 0$. Again, note that since this fitness function is primarily focused upon driving the error between $\Psi_J(k)$ and $\hat{\Psi}_J(k)$ to zero, there is no guarantee that the state estimates $\hat{\Phi}_J(k)$ and $\hat{R}_J(k)$ will accurately track the true states $\Phi_J(k)$ and $R_J(k)$, respectively. However, due to the dependence of each state on the other two states (seen in Eqs. (22) and (23)), it again seems at least possible that acceptable state tracking will occur.

Two different sets of initial conditions were used. In Case 1, $\Psi_J(0) = 0.5, R_J(0) = 0.1$, and $\Phi_J(0) = 1.4$ were used, and in Case 2, $\Psi_J(0) = 0.5, R_J(0) = 0.1$, and $\Phi_J(0) = 0.8$ were used. Although the only difference between the two cases is found in $\Phi_J(0)$, this leads to a significant difference in the behaviors of Φ_J and R_J, as is further explained in McCaughan (1990), Moore and Greitzer (1986b) and Greitzer (1976).

For both cases, the following GA parameters were used, after some tuning: $N = 200$, $p_m = 0.2$, $D = 66, \rho \cdot D = 8, W = 100$, and $g_t = 10$. The ranges for $\hat{c}_1, \hat{c}_2, \hat{c}_3, \hat{c}_4, \hat{c}_5$, and $\hat{c}_6$ were [1.0,2.0], [1.2,1.8], [0.2,0.8], [2.0,4.0], [0.3,0.7], and [5.0,9.0], respectively. The initial state estimates were chosen to be $\hat{R}_J(0) = 0.05$ and $\hat{\Phi}_J(0) = 1.0$ because a stable equilibrium point of the jet engine compressor model occurs when $\Phi_J = 1.0$ and because choosing $\hat{R}_J(0) = 0$ would result in $\hat{R}_J(k) = 0$ for all $k \geq 0$. The results are shown in Figs. 4 and 5 for the two initial conditions.

Table 3
Squared error sums for GA-based state estimation of the jet engine compressor model, Case 1

	$E_{s_{R_J}}$	$E_{s_{\Phi_J}}$
Average	310.91	1116.81
Minimum	21.14	527.02
Maximum	1770.35	2104.74
Shown in Fig. 4	287.06	1051.25

Table 4
Squared error sums for GA based state estimation of the jet engine compressor model, Case 2

	$E_{s_{R_J}}$	$E_{s_{\Phi_J}}$
Average	186.77	226.11
Minimum	45.29	47.53
Maximum	697.34	602.34
Shown in Fig. 5	144.81	275.74

The GA-based state estimator for the jet engine compressor model was simulated for 25 s with $T = 0.01$. To verify the validity of the results, 50 simulations were executed, and squared error sums were generated for $\hat{\Phi}_J$ and $\hat{R}_J$. The results are given in Tables 3 and 4. The squared error sums for the simulations shown in Figs. 4 and 5 are given in the tables to verify that the figures correspond to average or typical GA simulations.

From the figures it can be seen that the GA performed quite well for Case 2. For Case 1, there was some difficulty with the tracking, but this does not constitute a complete failure. Notice that the state estimates still indicated significant changes when such changes occurred in the actual states. For example, when there was a sizable growth in the actual rotating stall cell amplitude, the GA-based estimates indicated such a growth. Even though the estimate of the size of the cell was somewhat inaccurate, the growth was still detected (and hence it will provide advance warning of, for example, rotating stall). This tracking of the *behavior* of a state is still acceptable or useful in many cases, although the tracking of the state itself may be slightly in error. In light of this discussion, there is evidence here that the GA is a useful tool for gaining information about non-measurable states in nonlinear systems.

6.3. Comparison to the conventional extended Kalman filter

To apply the conventional extended Kalman filter (EKF) (Mendel, 1995) to this nonlinear state-estimation problem one needs to specify the noise covariances for the state and measurement equations. Using diagonal covariance matrices for both cases (a 3×3 matrix for the state equation and a 1×1, that is, scalar, for the output equation) with small diagonal values (for example, all with values of 0.0000001) the EKF becomes unstable in all test runs we attempted (that is, after about 12 s the estimate of each value becomes unbounded by departing from the true value very fast; before 5 s the estimate is reasonable accurate but from 5 s to 12 s it is quite bad). Decreasing the values of the noise covariances to even smaller values (even all zero values for the noise covariance matrix for the state equation) still resulted in an EKF that was unstable, and hence completely unable to properly estimate the variables. On the other hand, the genetic adaptive state estimator of the last section *never* failed for the many simulation runs that we tested it for (for example, for the 50 simulations reported in the last section and many other simulation runs). Clearly, the genetic adaptive state estimator deserves serious consideration compared to the standard (and often successful) extended Kalman filter for this nonlinear state estimation problem.

7. Concluding remarks

This study has introduced a novel genetic adaptive state-estimation technique and studied its performance for three cases — two linear cases and a nonlinear jet engine compressor. For the linear cases, the performance of an average GA execution was compared with that of the adaptive Luenberger observer (a nonlinear estimation method), and it was seen that the GA performed nearly as well as the Luenberger observer. For the nonlinear case, the GA alone was studied (since the assumptions needed to apply the adaptive Luenberger observer are not satisfied), and although exact numerical tracking of the states related to surge and rotating stall was not always achieved, adequate behavior tracking did occur. That is, the GA-based estimator was indeed able to detect the onset of surge and the growth of stall cells (see Figs. 4 and 5). In the final subsection of the paper we studied the use of the extended Kalman filter for the jet engine application and explained how it failed for this application. From these three cases, evidence can be gathered that may point to the existence of the GA as an alternative method for state estimation, particularly in nonlinear cases.

Clearly, however, further research of the GA as a state estimation tool is needed. Among this research, possible topics could include

- alternative GAs (for example, different fitness functions, representation schemes such as base-2, genetic operators, and so on),
- determination of the *best* GA for the state estimation problem,
- tuning procedures for the GA,
- mathematical stability, convergence, and robustness analysis,
- actual implementation issues (for example, processor speed, memory resources, possible hardware operations, etc.).

Also, due to the computational complexity of the GA, it seems logical to attempt conventional approaches for state estimation *before* employing the GA, especially for linear systems. From the results in this study, however, there is evidence that the GA does provide a viable alternative when other techniques encounter difficulty, particularly when dealing with nonlinear systems.

References

Goldberg, D.E., 1989. Genetic Algorithms in Search, Optimization, and Machine Learning. Addison-Wesley, Reading, MA.

Greitzer, E.M., 1976. Surge and rotating stall in axial flow compressors. Journal of Engineering for Power 98, 190–198.

Gremling, J.R., Passino, K.M., 1997. Genetic adaptive state estimation for a jet engine compressor. IEEE International Symposium on Intelligent Control, Istanbul, Turkey, July 16–18, pp. 131–136.

Ioannou, P.A., Sun, J., 1996. Robust Adaptive Control. Prentice-Hall, Engelwood Cliffs, NJ.

Kristinsson, K., Dumont, G., 1992. System identification and control using genetic algorithms. IEEE Transactions on Systems, Man, and Cybernetics 22 (5), 1033–1046.

Krstic, M., Kokotovic, P.V., 1995. Lean backstepping design for a jet engine compressor model. IEEE Conference on Control Applications.

McCaughan, F.E., 1990. Bifurcation analysis of axial flow compressor stabilty. SIAM Journal of Applied Mathematics 50 (5), 1232–1253.

Mendel, J.M., 1995. Lessons in Estimation Theory for Signal Processing, Communications, and Control. Prentice-Hall, Englewood Cliffs, NJ.

Michalewicz, Z., 1992. Genetic Algorithms + Data Structures = Evolution Programs. Springer, New York, NY.

Moore, F.K., Greitzer, E.M., 1986a. A theory of post-stall transients in axial compression systems: Part I — development of equations. Journal of Engineering for Gas Turbines and Power 108, 68–76.

Moore, F.K., Greitzer, E.M., 1986b. A theory of post-stall transients in axial compression systems: Part II — application. Journal of Engineering for Gas Turbines and Power 108, 231–239.

Porter, L.L., Passino, K.M., 1995. Genetic adaptive observers. Engineering Applications of Artificial Intelligence 8 (3), 261–269.

Srinivas, M., Patnaik, L.M., 1994. Genetic algorithms: a survey. IEEE Computer 17–26.

Yao, L., Sethares, W.A., 1994. Nonlinear parameter estimation via the genetic algorithm. IEEE Transactions on Signal Processing 42 (4), 927–935.