PERGAMON

Engineering Applications of Artificial Intelligence 12 (1999) 185–200

Engineering Applications of
ARTIFICIAL INTELLIGENCE

Genetic adaptive identification and control☆

W.K. Lennon, K.M. Passino *

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

Accepted 1 September 1998

Abstract

Genetic algorithms are computer programs that are developed to crudely emulate the evolution of biological populations according to Darwin's theory of natural selection and the concept of inheritance from genetics put forward by Mendel. Suppose that a controller for a plant is viewed as an individual decision-maker that has been chosen from a population of possible decision-makers to generate a control input to the plant. Suppose that in real-time a population of such decision-makers evolves. The "best" decision-maker from the evolving population is chosen at each step to control the plant, and using the principles of inheritance and survival of the fittest, good decision-makers will be more likely to propagate through the population as it evolves. Generally, as the population evolves and the best decision-maker is chosen at each time step, it adapts to its environment (i.e. the controller adapts to the plant and anything that influences it) and enhanced closed-loop system performance can be obtained. Also, even if there are plant parameter variations or disturbances, the population of decision-makers (controllers) will continually adapt to its environment to try to maintain good performance. This paper discusses a variety of such genetic adaptive control methods, and gives an extensive comparative analysis of their performance relative to conventional adaptive control techniques for an illustrative control application.

Keywords: Genetic algorithm; Genetic adaptive control

1. Introduction

A genetic algorithm (GA) is a parallel search technique that emulates the laws of evolution and genetics to try to find optimal solutions to complex optimization problems (Goldberg, 1989). Some research has been conducted using genetic algorithms to help design control systems, but usually these methods involve off-line design of the control system (Lee and Takagi, 1993, Varšek et al., 1993; Daihee et al., 1994; Porter and Borairi, 1992; Michelewicz et al., 1990). GAs have also been used for off-line system identification (Maclay and Dorey, 1993). The research on the use of GAs for on-line real-time estimation and control include: (Kristinsson and Dumont, 1992), where genetic algorithms are used for system identification of linear systems and coupled with pole-placement-based indirect adaptive control; (Porter and Passino, 1994), where a direct genetic adaptive control method is introduced; and (Porter et al., 1995), where genetic adaptive observers are introduced to estimate plant stakes. The technique in (Porter and Passino, 1994) is applied to the control of a brake system in (Lennon and Passino, 1995). Relevant general ideas on GAs and adaptive systems are in (DeJong, 1980) and other applications and methods are studied in (Renders and Hanus, 1992; Zuo, 1995).

This paper will investigate ways to use genetic algorithms in the on-line control of a nonlinear system, and compare the results obtained with conventional control techniques. A direct genetic adaptive controller and an indirect genetic adaptive controller are developed, and they are combined into a general genetic adaptive controller[1]. Several conventional controllers will be

☆ The work was supported in part by National Science Foundation Grant EEC 9315257.

* Corresponding author. Tel.: 001 614 292 5716; Fax: 001 614 292 7596; e-mail: k.passino@osu.edu.

[1] Indirect adaptive control uses an "identifier" to synthesize a model of the plant dynamics and then information from this model is used to tune a controller (it is said that the controller was tuned "indirectly" by first identifying a model of the plant). For "direct adaptive control", an identifier is not used for the plant; the parameters of the controller are tuned directly (some think of the direct adaptive controller as a "controller identifier"). For more details see, e.g. (Ioannou and Jing Sun, 1996; Åström and Wittenmark, 1989).

PII: S0952-1976(98)00051-7

studied including a proportional-derivative controller, a model reference adaptive controller, and two indirect adaptive controllers. To demonstrate all these control techniques, the problem of cargo ship steering will be studied. In this application, the desired performance is described with a reference model and our control techniques seek to track the output of the reference model. Overall, the goal is not to design the best possible controller for ship steering; this example is simply used to illustrate the ideas.

The direct genetic adaptive controller used here is a type of "genetic model reference adapter controller" (GMRAC) that was originally introduced in (Porter and Passino, 1994). Here, the method in (Porter and Passino, 1994) for fitness evaluation is modified and the idea of initializing the population with some fixed controllers and letting these remain fixed throughout the controller's operation is studied. It is explained how this idea is related to ones in "multiple model adaptive control" (Narendra and Balakrishnan, 1994). The indirect genetic adaptive controller that is studied here is most similar to one in (Kristinsson and Dumont, 1992) where the authors use a GA for model identification and then use the model parameters in a certainty equivalence control law based on a pole-placement method. Here, however, a different method for fitness evaluation is used and a model reference approach is employed. The general genetic adaptive controller studied here is novel in that it combines the direct and indirect approaches. To do this, it uses genetic adaptive identification to estimate the parameters to the model that are used in the fitness function for the direct genetic adaptive controller. Essentially, the general genetic adaptive controller both identifies the plant model and tries to tune the controller at the same time so that if the estimates are inaccurate good control can still be achieved.

While several of the genetic adaptive methods are novel it is emphasized that one of the primary contributions of this paper lies in the comparative analysis with conventional adaptive control techniques. Overall, such a comparative analysis is very important for identifying both the advantages of new intelligent control techniques and their possible disadvantages (Antsaklis and Passino, 1993; Passino, 1996). The remainder of the paper is organized as follows.

In the next section the cargo ship steering problem is defined. Following that, direct conventional and genetic adaptive control methods are developed and their performance is compared. Next, the same is repeated but for indirect adaptive methods and a combined indirect/direct genetic adaptive method. Finally, some concluding remarks are provided, the results of the paper are summarized, and an analysis of the computational complexity of the genetic adaptive controllers is given.

2. The cargo ship steering control problem

The objective in the cargo ship control problem is to control the ship heading, ψ by moving the rudder, δ. A coordinate system is fixed to the ship as shown in Fig. 1. The cargo ship is described by a third-order nonlinear differential equation (Åström and Wittenmark, 1989), that is used in all simulations as the truth model, and is given by

$$\dddot{\psi}(t) + \left(\frac{1}{\tau_1} + \frac{1}{\tau_2}\right)\ddot{\psi}(t) + \left(\frac{1}{\tau_1\tau_2}\right)\left(a\dot{\psi}^3(t) + b\dot{\psi}(t)\right) = \frac{k}{\tau_1\tau_2}\left(\tau_3\dot{\delta}(t) + \delta(t)\right). \quad (1)$$

The input, δ, and the output, ψ, are both measured in radians. The constants a and b are assigned a value of one for all simulations. The constants k, τ_1, τ_2, and τ_3 are defined as

$$k = k_0 \frac{u}{l} \quad (2)$$

$$\tau_i = \tau_{i0}\frac{l}{u}, \; i = 1, 2, 3. \quad (3)$$

where u is the forward velocity of the ship in m/s and l is the length of the ship in m. For the cargo ship, $k_0 = -3.86$, $\tau_{10} = 5.66$, $\tau_{20} = 0.38$, $\tau_{30} = 0.89$, $l = 161$ m and $u = 5$ m/s (nominally). The maximum allowable rudder angle is ± 1.3963 rad ($\pm 80°$).

In all cases, the reference model is

$$W_m(s) = \frac{r^2}{(s+r)(s+r)}$$

with $r = 0.05$. Hence the output of the reference model is

$$\psi_m = W_m(s)\zeta.$$

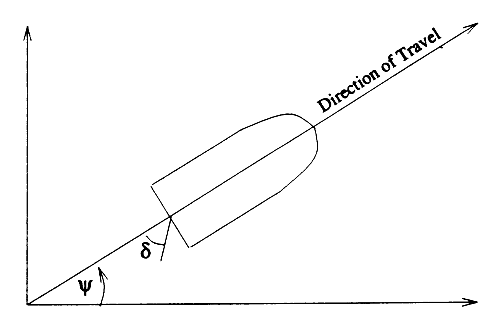

Fig. 1. Cargo ship control problems. The input to the system is the rudder angle, δ. The output of the system is the cargo ship heading, ψ.

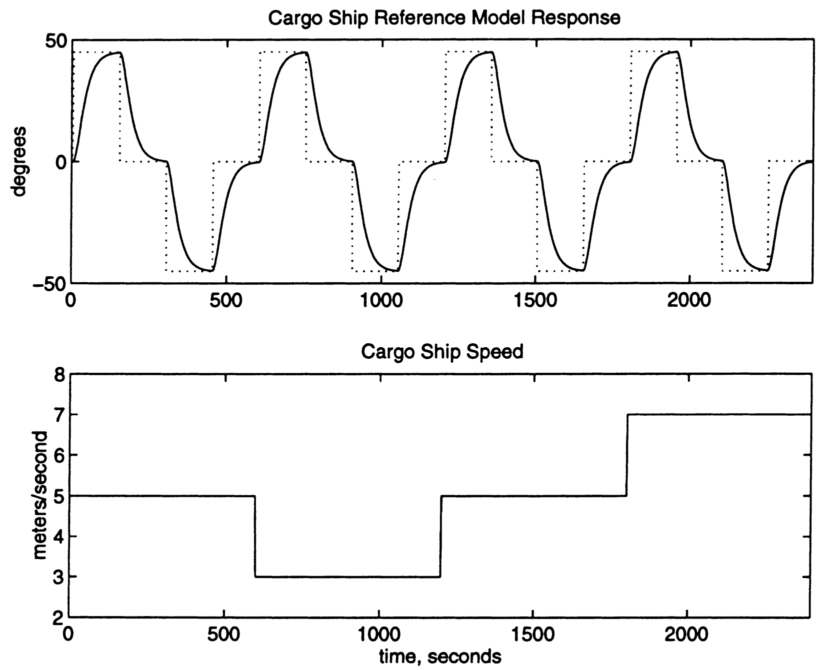

Fig. 2. Reference input and cargo ship speed. The top plot depicts the reference input ζ (the dotted line) and the output of the reference model ψ_m (solid line). The reference input ζ changes by 0.7854 rad (45°) every 150 s. The bottom plot shows the speed of the cargo ship over the course of the simulation. The speed changes by 2 m/s every 600 s.

Here, ζ is the desired cargo ship heading and ψ_m is the "ideal" response, i.e. the response that will be tracked. Note that this response is fairly slow, but the ship is large and hence it is not realistic to request that the rudder change its direction very fast.

In the simulation tests, the speed of the cargo ship, u, is changed every 600 s, beginning at the nominal 5 m/s, dropping to 3 m/s, rising back to 5 m/s and finally rising again to 7 m/s. From Eqs. 2 and 3 it is easy to see that such speed changes significantly affect

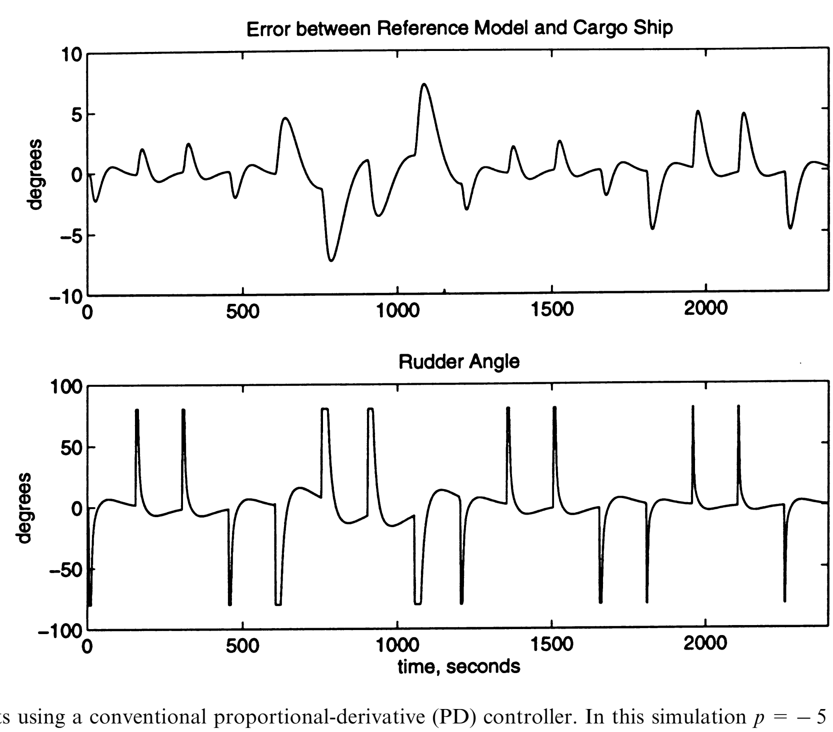

Fig. 3. Results using a conventional proportional-derivative (PD) controller. In this simulation $p = -5$ and $d = -175$.

the dynamics of the ship. Intuitively, as the ship slows down, the rudder becomes less effective as a steering input. Fig. 2 shows the reference input, the reference model output, and the speed of the ship used throughout all simulations to follow.

To provide an idea of how a control system will operate for this system, we show the response for the case where $\delta = -5(\zeta - \psi) - 175\, \mathrm{d}/\mathrm{d}t(\zeta - \psi)$ (i.e. a manually tuned proportional-derivative (PD) controller) in Fig. 3. The PD controller is able to regulate the heading of the ship to within $\pm 25°$ when $u = 5$ m/s, but when $u = 3$ m/s it can only regulate it to within $\pm 7.5°$. Note that the PD gains of $p = -5$ and $d = -175$ were selected because they minimize this tracking error.

3. Direct adaptive control

In this section, the parameters of a controller are directly adjusted to make the error between the cargo ship heading and the reference model output go to zero. The study begins with a conventional model reference adaptive controller from (Ioannou and Jing Sun, 1996) and then a genetic model reference adaptive controller is developed.

3.1. Model reference adaptive control

In this section a model reference adaptive controller (MRAC) (Ioannou and Jing Sun, 1996), with a gradient identification algorithm to identify the parameters of the system, is developed. It is assumed that the cargo ship behaves as a third-order linear system, defined by the transfer function:

$$G(s) = \frac{\psi}{\delta} = \frac{k(1 + \tau_1 s)}{s(1 + \tau_2 s)(1 + \tau_3 s)}. \tag{4}$$

It is assumed that the values of the plant parameters are not known, but it is known that the plant is third-order and that the gain $k < 0$.

Following (Ioannou and Jing Sun, 1996), the control signal is defined as:

$$\delta = \theta_1^T \alpha(s)\omega_1 + \theta_2^T \alpha(s)\omega_2 + \theta_3\psi + c_o\zeta. \tag{5}$$

The adaptive controller will tune the scalars c_0 and θ_3 and the 2×1 vectors θ_1 and θ_2. As in (Ioannou and Jing Sun, 1996), it is assumed an upper bound on the parameter c_0 is known, namely $c_0 < \bar{c}_0 = -0.1$.

The terms in Eq. (5) are given by

$$\omega_1 = \frac{\delta}{\Lambda(s)},$$

$$\omega_2 = \frac{\psi}{\Lambda(s)},$$

$$\epsilon = \frac{W_m(s)\delta - \hat{z}}{m^2},$$

$$\hat{z} = \theta^T \phi_p,$$

$$m^2 = 1 + \phi_p^T \phi_p,$$

$$\phi_p = \left[W_m(s)\omega_1^T W_m(s)\omega_2^T W_m(s)\psi\psi \right]^T,$$

$$\theta = [\theta_1, \theta_2, \theta_3, c_o]^T,$$

$$\alpha(s) = [s1]^T,$$

$$\Lambda(s) = s^2 + s + 1,$$

$$g(\theta) = \bar{c}_o + c_o.$$

If $|c_0(t)| > \bar{c}_0$ or $(\Gamma\epsilon\phi_p)^T \nabla g \le 0$ ($\nabla g = \mathrm{d}/\mathrm{d}\theta g = [000001]^T$) then the controller parameter update law is

$$\dot{\theta} = \Gamma\epsilon\phi_p$$

otherwise we update θ using projection

$$\dot{\theta} = \Gamma\epsilon\phi_p - \Gamma\frac{\nabla_g \nabla_g^T}{\nabla_g^T \Gamma \nabla_g}\Gamma\epsilon\phi_p.$$

Gradient algorithm is used to identify the parameters in the controller and attempt to make the error between the plant and the reference model go to zero.

The vector θ was initialized for this simulation to

$$\theta(0) = [1.3,\ 0.9,\ 9.9 - 0.1,\ 0.4,\ -0.3]^T.$$

These values were selected because the controller will adapt to these parameters over time when the cargo ship maintains a speed of 5 m/s (i.e. we are trying to initialize the parameters as best we can). Performance degrades with other choices for initial parameters (e.g. $\theta(0) = 0$).

Fig. 4 shows the results using the MRAC. It is important to note that when the nonlinear model is used in simulation the theory for the MRAC from (Ioannou and Jing Sun, 1996) does not apply. The MRAC performed very poorly when the cargo ship speed was decreased to 3 m/s. It was unable to adequately adapt to the changing system dynamics and diverged significantly from the reference model (and significant efforts were put into tuning the controller to even get the performance shown in Fig. 4). Clearly

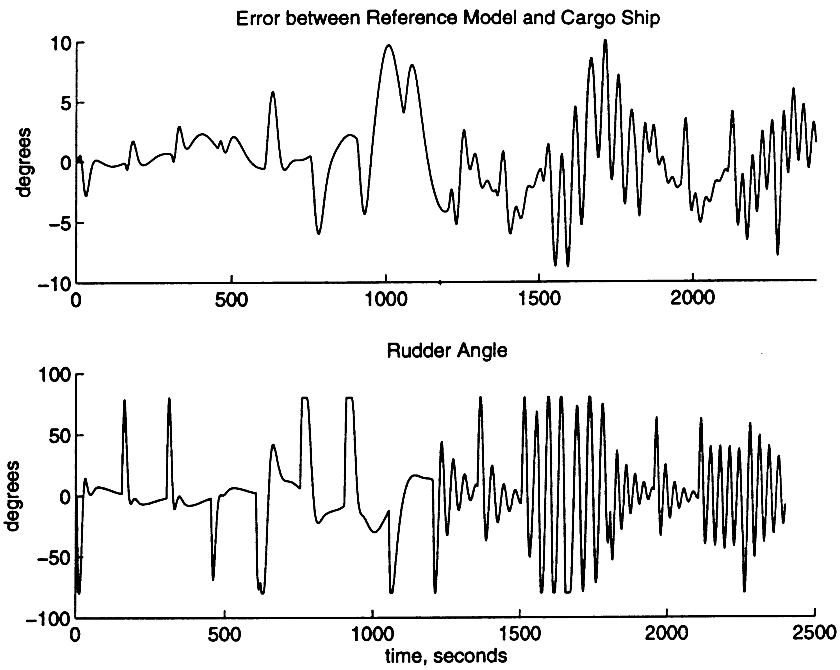

Fig. 4. Results using MRAC with continuous-time gradient identification. The top plot shows the error between the reference model output and the cargo ship response. The bottom plot shows the rudder angle, δ.

the performance is not even as good as that of the fixed PD controller shown in Fig. 3.

3.2. Genetic model reference adaptive control

In this section a type of genetic model reference adaptive controller (GMRAC) (Porter and Passino, 1994) is used to adjust the gains of a PD controller. The GMRAC, shown in Fig. 5 uses a model of the plant and a genetic algorithm to "evolve" controller parameters to minimize the error between the cargo ship heading and the reference model output. The genetic algorithm (GA) uses the principles of evolution and genetics to select and adapt the controller parameters. For our simulations, a set of controllers is used as the GA population, and each controller's potential to control the cargo ship is evaluated.

First, the members of the population are defined, in this case PD controllers. Each individual controller is defined by a 10 digit number, its "chromosome". The

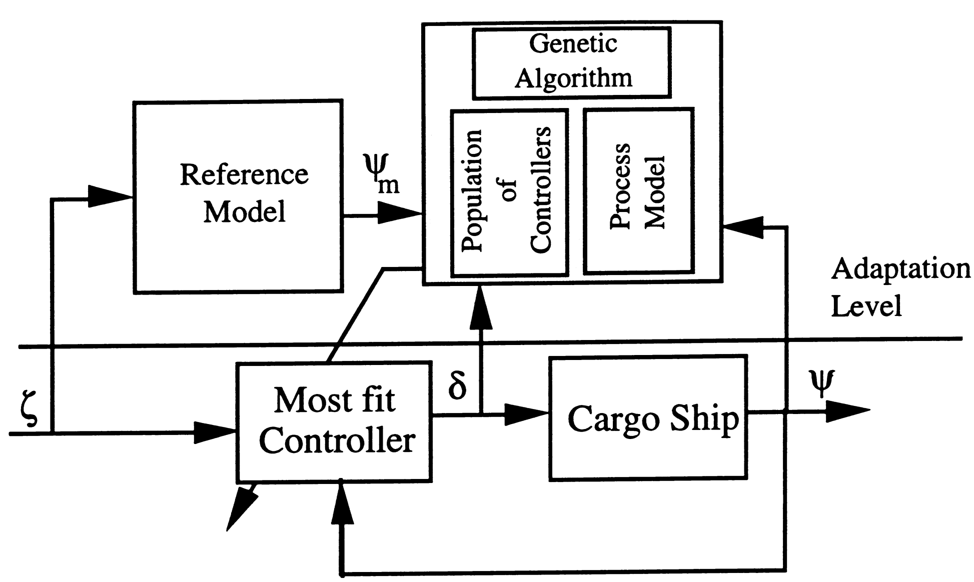

Fig. 5. GMRAC for cargo ship steering.

first four digits describe the proportional gain, and the last six digits describe the derivative gain. The proportional and derivative gains are allowed to lie between

$$-10 < p \leq 0, \qquad -200 < d \leq 0.$$

For example, a possible chromosome is [1234123456]. This would translate into a proportional gain of $p = -1.234$, and a derivative gain of $d = -123.456$. Note that the negative sign is implied, as is the decimal point after one digit for the p gain and after three digits for the d gain. Also, because of restrictions placed on d, the fifth digit in the chromosome must be either a 0 or 1 (for example, if the fifth digit were a 2, then necessarily $p \leq -200$ which is outside the designated region).

During every time step, each member of the population is evaluated on how well it minimizes the error between the predicted closed-loop system response (using a cargo ship model) and the predicted reference model output. The cargo ship model used is the discrete-time approximation (using zero-order hold with sampling time $T = 0.5$) of the third-order continuous time plant shown in Eq. (4), where k, τ_1, τ_2 and τ_3 are as defined in Eqs. 2 and 3 assuming the cargo ship is travelling at the nominal speed of 5 m/s. The discrete-time cargo ship model is

$$G(z) = \frac{\hat{\psi}}{\delta} = \frac{-0.0001909(z+0.9913)(z-0.9827)}{(z-1)(z-0.9972)(z-0.9600)}, \tag{6}$$

which can be written as

$$G(z) = \frac{\hat{\psi}}{\delta} = \frac{a_1 z^2 + a_2 z + a_3}{z_3 + b_1 z^2 + b_2 z + b_3} \tag{7}$$

where the values for a_1, a_2, a_3, b_1, b_2, b_3 can be found from Eq. (6).

To evaluate the controller population, the GA finds the error between the cargo ship heading ψ and the reference input ζ, $e = \zeta - \psi$, and the derivative of that error[2]. For each member of the population, the GA computes the rudder angle using e, $\dot{e}$, and the PD gains on each chromosome. Next it estimates the cargo ship heading and the reference model output $NT = 5$ s into the future. Using this information, the GA can determine which controllers in the population are keeping the cargo ship heading as close as possible to the reference model output.

The following pseudo-code more precisely defines the fitness evaluation of the GA and hence the operation of the GMRAC.

1. Complete the error and derivative of the error between the cargo ship heading and the reference input as

$$e(t) = \zeta(t) - \psi(t), \qquad \dot{e}(t) = \frac{\mathrm{d}e}{\mathrm{d}t}.$$

2. Predict the reference model output, ψ_{m}, NT s into the future using a first-order approximation

$$\hat{\psi}_{\mathrm{m}}(t+NT) = \psi_{\mathrm{m}}(t) + NT\dot{\psi}_{\mathrm{m}}(t).$$

Here $\psi_{\mathrm{m}}(t)$ is not the continuous time derivative, rather it is the discrete time first order approximation of the derivative,

$$\dot{\psi}_{\mathrm{m}}(t) = \frac{\psi_{\mathrm{m}}(t) - \psi_{\mathrm{m}}(t-T)}{T}$$

(notice that the notation for a derivative is being abused).

3. Compute the current error between the cargo ship heading and the reference model output.

$$\epsilon(t) = \psi_{\mathrm{m}}(t) - \psi(t).$$

4. Suppose that the ith candidate controller $C_i = (p_i, d_i)$. For each candidate controller, C_i, do the following:

Determine the rudder angle input using the p_i,d_i gains on the candidate controller chromosome

$$\delta_i = p_i e(t) + d_i \dot{e}(t).$$

Initialize the discrete-time cargo ship model (Eq. (7) with past samples from the cargo ship

$$\hat{\psi}_i(k-j) = \psi(t-jT), \quad j = 0,1,2,$$
$$\delta_i(k-j) = \delta(t-jT), \quad j = 0,1,2.$$

Assuming the rudder angle δ_i stays constant[3] for the next N samples (i.e. $\delta_i(k+1) = \ldots = \delta_i(k+N) = \delta_i$), predict the output of the cargo ship model, $\psi(k+N)$, N steps into the future using Eq. (7):

for $j = 1$ to N

$$\begin{aligned}\hat{\psi}_i(k+j) = {} & a_1\delta_i(k+j-1) + a_2\delta_i(k+j-2) \\ & + a_3\delta_i(k+j-3) - b_1\hat{\psi}_i(k+j-1) \\ & - b_2\hat{\psi}_i(k+j-2) - b_3\hat{\psi}_i(k+j-3).\end{aligned}$$

next j.

[2] A continuous-time derivative is used. It is assumed that it is available for the PD controller in the last section and therefore available here. Note, however that the results change very little if a backwards difference approximation is used for $\dot{e}$.

[3] As an alternative, one could let the controller C_i vary the rudder angle δ_i over this time interval. A constant control signal δ_i is assumed because it is more accurate than estimating $\delta_i(k+j)$ based on the gains of the candidate controller and an estimation of the error signal. The inherent accuracy of approximating the derivative of the error signal $\dot{e}(t)$ using the discrete-time cargo ship model are further amplified by the relatively large derivative gain of the PD controller.

Using the output of the cargo ship model, estimate the error between the cargo ship heading and the reference model output NT s into the future.

$$\hat{\epsilon}_i(t+NT) = \hat{\psi}_{\mathrm{m}}(t+NT) - \hat{\psi}_i(k+N).$$

Estimate the derivative of the error between the cargo ship model output and the reference model output, using a first order approximation.

$$\dot{\epsilon}_i(t) = \frac{\hat{\epsilon}_i(t+NT) - \epsilon(t)}{NT}.$$

Of course the $\dot{\epsilon}$ is not a continuous-time derivative, merely a discrete time approximation.

Assign fitness, J_i, to each controller candidate, C_i:

$$\bar{J}_i = \epsilon(t) + \beta\dot{\epsilon}_i(t).$$

Then

$$J_i = \frac{\alpha}{\bar{J}_i^2 + \alpha}.$$

The values $\alpha = 0.001$ and $\beta = 5$ were chosen. The choices for α and β are explained below.

5. Repeat step 4 for each member of the population.

6. The maximally fit controller becomes the controller used for the next time step.

The value chosen for α sets an upper bound on the fitness J_i and its choice should be considered carefully. If α is too small, then a large disparity in fitness values will exist in the population (a small change in $\bar{J}_i$ could result in a large change in J_i), and in general only a few individuals will be selected to reproduce into the next generation (assuming roullette-wheel selection, as described below). The result could be a population of nearly identical individuals and hence eliminate the parallel-search mechanism of the GA. However, if α is chosen too large, then all members of the population will have nearly equal fitness values, and hence nearly equal chances to reproduce, thereby compromising the entire "survival of the fittest" nature of the GA. In general, α is selected to be about an order of magnitude smaller than the average value of $\bar{J}_i$. This appears to be a reasonable compromise value.

The value for β is chosen based on the desired performance of the system and the ability to control the plant. Heuristically speaking, β defines how quickly we would like the error ϵ between the cargo ship heading and the reference model to reach zero. To see this, notice that the fitness function is maximized when $\bar{J}_i^2$ is minimized, which corresponds to when $\bar{J}_i = 0$. Looking at the equation for $\bar{J}_i = \epsilon(t) + \beta\dot{\epsilon}_{\mathrm{i}}(t)$, the fitness function is maximized when $\dot{\epsilon}_{\mathrm{i}}(t) = -epsi\backslash rm\ (t)/\beta$. For example, if $\epsilon(t) = 1.0$, then we would like $\dot{\epsilon}_{\mathrm{i}}(t)$ to be –0.2 so that the error is driven to zero in approximately $\beta = 5$ s. In this example, the best controller is the one that produces $\dot{\epsilon}_{\mathrm{i}}(t)$ closest to –0.2. When choosing β, it is desired to make it as small as possible, but it is not desirable to induce oscillations, nor can physical constraints such as rudder input saturation and slow system dynamics be ignored.

The "look ahead time window", N, was chosen to be 10 samples ($NT = 5$ s) as a compromise of conflicting interests. In general it is good to make N large, because often the effects of the current input signal are not readily apparent at the output of the system. The longer the time window, the better we are able to assess the effects the input has on the system. However, the error between what the cargo ship model predicts and how the actual cargo ship behaves increases as N increases, thereby degrading the accuracy of the fitness function. Also, in this application, it is assumed that the rudder angle stays constant for N s, an assumption that becomes less valid as N increases. In general it has been found that time windows of 10–20 discrete-time samples work well.

Once each controller in the population has been assigned a fitness J_i, the GA uses the roullette-wheel selection process, as described by Goldberg (Goldberg, 1989) to pick which controllers will "produce" into the next generation. The individuals selected to reproduce are said to be "parents" of the next generation. In the roullette-wheel selection process, the probability of an individual reproducing into the next generation is proportional to the fitness of that individual. More specifically, the probability $p_{\mathrm{p}i}$ that the ith member of the population will be selected as the jth member of the parent pool is

$$p_{\mathrm{p}i} = \frac{J_i}{\sum_{k=1}^{N} J_k}.$$

Note that some individuals will likely be selected to become a parent more than once (indicating they will have more than one offspring) while others will not be selected at all. In this manner, unfit individuals are generally removed from the population while fit individuals multiply.

Once the parents of the next generation have been selected, they are randomly paired together. Each pair of parents then has a probability, p_{c} of undergoing "crossover", in which some digits in the parent chromosome are exchanged with some digits of the other parent chromosome. This is most commonly done by selecting a location on the chromosome (the crossover site) and exchanging all digits past that point on the chromosome with the digits in the same locations on the mating chromosome. However, crossover is dome differently in all our genetic algorithms in this paper. Here, once two parents have been selected for crossover, each digit on the chromosome has a 0.5 probability of being exchanged for the digit in the same location on the mating chromosome. The digits that crossover maintain their original position in the chromosome, and are only exchanged with the digit in the same position on the other chromosome. For example, if two "parent" chromosomes, [1111111111] and [3333333333], undergo crossover, the resulting "children" chromosomes could be [1133313111] and [3311131333]. Note that this method of crossover is not standard, but we have found it to be more effective than the traditional method when more than two traits are encoded onto the chromosome because it allows for the possibility of two non-adjacent traits on a chromosome to remain together after crossover. Crossover is a form of local search in the p, d parameter search space. The probability of crossover, p_c, was set at 0.9.

After crossover, each digit in each child chromosome has the probability, p_m, of mutating. If a digit is selected for mutation, then that digit is replaced by a new randomly selected digit. For example, a chromosome [1111111111] may be mutated to [1115111111]. Note that a digit may be "mutated" to its original value, in effect not being mutated at all. The probability of mutation, p_m, was set at 0.1 and should take into account the possibility of these false-mutations. Mutation is a form of global search in the p, d parameter search space.

After mutation, the "children" chromosomes become the next generation of controllers and the process is repeated at the next time step. One exception to this process is the elitism operator (Porter and Passino, 1994), which sidesteps selection, crossover, and mutation and simply places the most fit controller from the previous generation into the next generation without modification. Elitism is used in all GAs in this paper.

The results of the GMRAC are shown in Fig. 6 (recall that non-adaptive PD controller results were shown earlier). The controller performs very well when the cargo ship has a speed of 5 or 7 m/s. It has difficulty when the ship slows to 3 m/s, as do all of the controllers investigated in this paper. Note that since the GMRAC is a stochastic adaptive controller, when it is run again, results may differ from those shown in Fig. 6. Fig. 6 shows the typical behavior of the GMRAC. In the concluding remarks, it is shown how an average controller will behave over a set of 100 simulations.

3.3. GMRAC with fixed population members

Because genetic algorithms are stochastic processes, there is always a small possibility that good controllers will not be found and, hence, degrade performance. While this possibility diminishes with population size, it nevertheless exists. One method to combat this possibility is to seed the population of the GA with individuals that remain unchanged in every generation. These fixed controllers can be spaced throughout the control parameter space to ensure that a reasonably

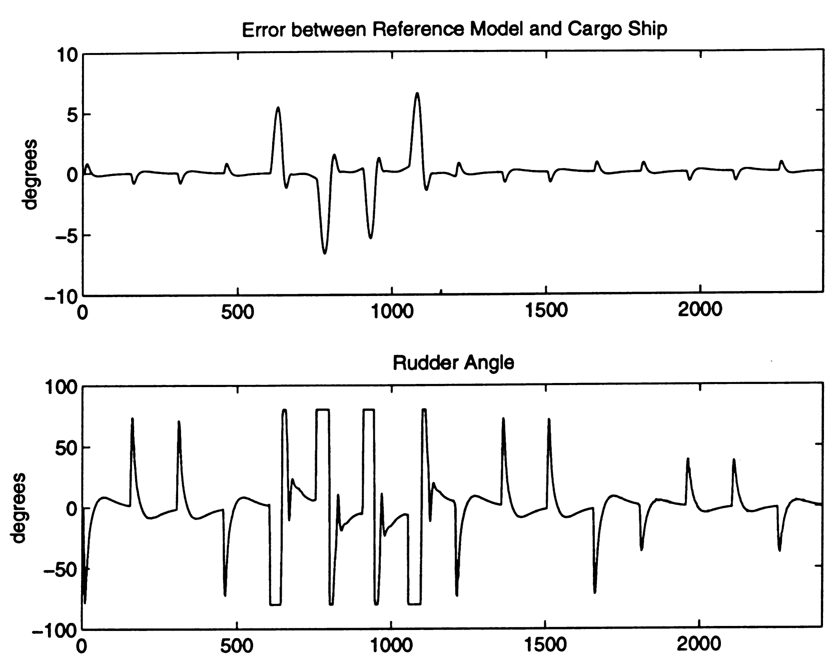

Fig. 6. Results using GMRAC. The top plot shows the error between the reference model output and the cargo ship response. The bottom plot shows the rudder angle, δ.

good controller is always present in the population. Simulations were run for the GMRAC with 25 fixed controllers in the GA population (leaving the remaining 75 controllers to be adapted by the GA as usual). Because the controller gains were restricted to $-10 < p \leq 0$ and $-200 < d \leq 0$, the population was seeded with 25 fixed PD controllers, defined by all possible combinations of $p \in \{-10, -7.5, -5, -2.5, 0\}$ and $d \in \{-200, -150, -100, -50, 0\}$.

Over the course of 100 simulations, the GMRAC with fixed population members had a smaller difference between minimum and maximum errors than did the GMRAC with no fixed population members (see Table 1). This was expected because the fixed models add a deterministic element to the inherently stochastic genetic algorithm.

Using fixed controllers is a novel control technique that appears to decrease the variations in the performance results. The technique is similar to (Narendra and Balakrishnan, 1994) where Narendra and Balakrishnan use fixed plant models to identify a plant and improve transient responses. Likewise, having fixed controllers the population enables the GA to find reasonably good controllers quickly and then search nearby to find better ones.

4. Indirect adaptive control

In this section, there is an attempt to model the non-linear cargo ship dynamics with a simple second-order linear model provided in (Åström and Wittenmark, 1989) and given by

$$G(s) = \frac{k_c}{s(s + p_c).} \tag{8}$$

The parameters k_c and p_c are used to create proportional and derivative gains for a PD controller using the certainty equivalence principle (Ioannou and Jing Sun, 1996). Given the plant model $G_p(s)$ and the controller model $G_c(s) = p + ds$, the closed loop transfer function is

$$T(s) = \frac{G_c(s)G_p(s)}{1 + G_c(s)G_p(s)} = \frac{k_c p + k_c ds}{s^2 + s(p_c + k_c d) + k_c p}.$$

Neglecting the closed-loop system zero and setting the closed-loop system poles equal to the reference model $W_m(s)$ poles, r, we determine the values of the proportional and derivative gains to be

Table 1
Results

Control technique	Sum of the errors squared $e(kT)^2$	Sum of the inputs squared $\delta(kT)^2$
Conventional PD	29.75	3991
MRAC	79.77	8142
Genetic model reference		
Adaptive control		
Minimum	9.66	5956
Average	10.06	5980
Maximum	10.25	5993
Genetic model reference		
Adaptive control with fixed controllers		
Minimum	10.02	5991
Average	10.18	6002
Maximum	10.40	6014
Indirect continuous-time gradient identification	14.83	3570
Indirect discrete-time least-squares identification	27.58	3392
Indirect genetic identification		
Minimum	27.47	3776
Average	29.73	3802
Maximum	36.52	3836
General genetic adaptive control		
Minimum	8.32	5756
Average	8.73	5935
Maximum	9.21	6217
General genetic adaptive control with fixed controllers		
Minimum	8.12	5769
Average	8.66	5980
Maximum	9.09	6467

$$p = \frac{r^2}{k_c}, \tag{9a}$$

$$d = \frac{2r - p_c}{k_c}. \tag{9b}$$

Below there is an attempt to identify the plant model parameters, k_c and p_c, using a continuous-time gradient algorithm, a discrete-time least-squares algorithm and finally with a discrete-time genetic algorithm. Then, Eq. (9) will be used to specify the PD controller gains.

4.1. Continuous-time gradient identification algorithm

Here, a continuous-time gradient algorithm is used to identify the parameters k_c and p_c of the plant model in Eq. (8). Again, the speed of the cargo ship is varied and we observe how well the identification algorithm can track the response of the cargo ship.

Following (Ioannou and Jing Sun, 1996) define

$$z = \theta^{*T}\phi + \eta = \frac{\psi s^2}{\Lambda(s)},$$

$$\theta = [k_c\ p_c]^T,$$

$$\phi = \left[\frac{\delta}{\Lambda(s)} \frac{-\psi s}{\Lambda(s)}\right]^T.$$

The update law is

$$\dot{\theta} = \Gamma \epsilon \phi$$

where

$$\Gamma = \begin{bmatrix} 200 & 0 \\ 0 & 20000 \end{bmatrix},$$

$$\epsilon = \frac{z - \theta^T\phi}{m^2},$$

$$m^2 = 1 + m_s,$$

$$\dot{m}_s = -\delta_0 m_s + \delta^2 + \psi^2,$$

$$\Lambda(s) = s^2 + 2s + 1.$$

The constant δ_0 was set to 0.001. The vector θ was initialized to [–0.0005 .02], which are the values the gradient algorithm settles on when the cargo ship is simulated at a constant speed of 5 m/s (i.e. again the algorithm is given an advantage by starting the values of θ close to their optimum values). The controller p and d gains were continuously adjusted using Eq. (9) and the current values of k_c and p_c.

The results are shown in Fig. 7. As expected, the performance degraded when the cargo ship slowed to 3 m/s. Overall, the indirect adaptive controller performed quite well. The performance is aided by using a continuous-time adaptation mechanism as opposed to the discrete-time mechanism used by the least-squares and GA based methods studied next.

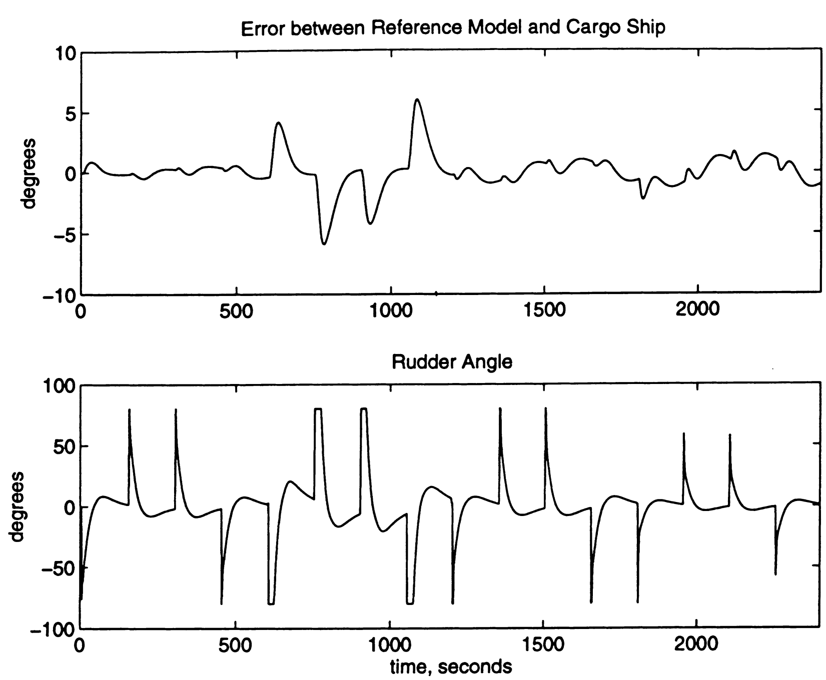

Fig. 7. Results using indirect adaptive control with continuous-time gradient identification. The top plot shows the error between the reference model output and the cargo ship response. The bottom plot shows the rudder angle, δ.

4.2. Discrete-time least-squares identification algorithm

In this section, the discrete-time least-squares identification algorithm is used to identify the linear plant model parameters. The model used is the linear, zero-order hold, discrete time equivalent ($T = 0.5$ s) of the second-order linear continuous-time plant $G(s)$ as shown in Eq. (8).

$$G(z) = \frac{\hat{\psi}}{\delta} = \frac{k_d z^2}{(z-1)(z-p_d)}. \qquad (10)$$

The discrete-time equivalent cargo ship model must be written in polynomial form for the least-squares algorithm. The cargo ship model $G(z)$ can be written as

$$G(z) = \frac{a_0 z^2}{z^2 + b_1 z + b_2}.$$

Because it is assumed, from the continuous-time plant model $G_p(s)$, that the plant contains an integrator, or $1/s$ term, it can likewise be assumed that the discrete-time model contains an integrator, or $1/z-1$ term. Therefore, when transforming the discrete-time parameters back into continuous time parameters, it can be assumed that one discrete-time pole is set at 1 and, therefore, the second pole is simply $p_d = b_2$. Obviously, the discrete-time gain, $k_d = a_0$.

The update laws for the least squares algorithm are as follows:

$$x(k) = \left[-y(k-1) - y(k-2)u(k)\right]^T,$$

$$\hat{\theta}(k) = [b_1\, b_2\, a_0],$$

$$\hat{y} = x^T(k)\hat{\theta}(k),$$

$$P(k) = P(k-1) - P(k-1)x(k) \left[1 + x^T(k)P(k-1)x(k)\right]x^T(k)P(k-1),$$

$$\hat{\theta}(k) = \hat{\theta}(k-1) + P(k)x(k)\left[y(k) - x^T(k)\hat{\theta}(k-1)\right].$$

If some knowledge of the plant parameters is assumed, namely their relative magnitudes, the covariance matrix, P, and parameter estimate, θ, can be initialized to more quickly identify the parameters of the system and hence improve performance. The covariance matrix was initialized to

$$P(0) = \begin{bmatrix} 5 & 0 & 0 \\ 0 & 5 & 0 \\ 0 & 0 & 20 \end{bmatrix}$$

and the plant parameters estimates were initialized to

$$\hat{\theta}(0) = \left[-1.99 \quad 0.99 \quad -0.001\right].$$

The least-squares algorithm was developed with the assumption that the plant parameters remain constant. Because in our simulations the speed of the cargo ship is varied and hence the nonlinear system dynamics vary, the least-squares algorithm has difficulty continually adjusting to the changing plant parameters. To compensate for this and improve the performance, the covariance matrix, P, was re-initialized to $P(0)$ every 150 s, before every change in the reference input[4].

After each time step, the current estimate of the discrete-time plant parameters k_d and p_d is used, and the following zero-order hold transforms are used to compute the continuous-time plant parameters.

$$k_c = \frac{k_d}{p_d T^2}, \qquad (11a)$$

$$p_c = \frac{1-p_d}{p_d T}. \qquad (11b)$$

The controller p and d gains were adjusted after every sample ($T = 0.5$ s) using Eq. (9) and the current values of k_c and p_c.

The results are shown in Fig. 8. The controller outperformed the non-adaptive PD controller, but it did not perform particularly well. This can mostly be attributed to the difficulty in identifying a continuous-time non-linear system using a simple discrete-time plant with a large sampling interval.

4.3. Indirect genetic adaptive control

In this section, the use of a second-order linear discrete-time model of the cargo ship is used and a genetic algorithm is used to identify the parameters of this model. The cargo ship model parameters are then used in a certainty equivalence based adaptive controller. Using the same second-order continuous-time model as shown in Eq. (8), an approximate discrete-time model can be derived. Using the zero-order hold discrete-time approximation with sampling time $T = 0.5$ s, the following is obtained

$$G(z) = \frac{\hat{\psi}}{\delta} = \frac{k_d z^2}{(z-1)(z-p_d)}. \qquad (12)$$

In this application, the genetic algorithm evolves the parameters k_d and p_d by looking at the past values for δ and ψ and attempting to minimize the error between the cargo ship heading ψ and the output of the cargo ship model, ψ. The overall strategy is shown in Fig. 9.

The individuals of the GA population are defined by 10 digits, the first five digits describe the k_d parameter,

[4] That is, a periodic "covariance reset" is performed, somewhat different than in (Ioannou and Jing Sun, 1996). The reset is performed periodically to avoid the computational complexity of having to test the eigenvalues of P at each step.

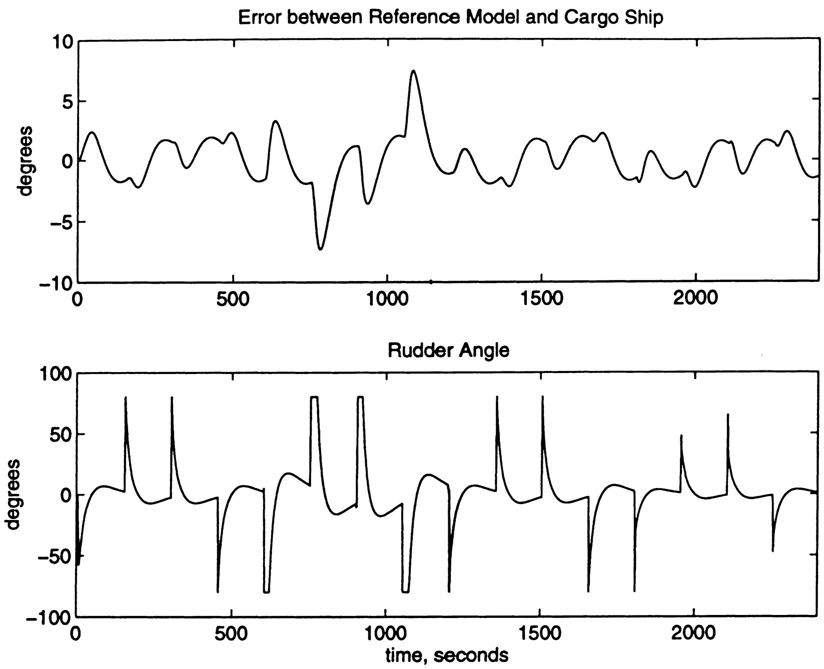

Fig. 8. Results using indirect adaptive control with discrete-time least squares identification. The top plot shows the error between the reference model output and the cargo ship response. The bottom plot shows the rudder angle, δ.

and the last five digits describe the p_{d} parameter. The parameter values are restricted to lie between

$$-0.001 < k_{\mathrm{d}} \leq -0.0001, \quad 0.5 \leq p_{\mathrm{d}} < 1.0.$$

Note that both parameters are fairly restricted. While shrinking the parameter search space improves the identification performance, it also requires some a priori knowledge of the cargo ship dynamics. But for many of the techniques in this paper, the existence of such knowledge is assumed. For instance, in both the gradient and least-squares algorithms, some a priori knowledge was also required to initialize the plant parameters and adaptation gains.

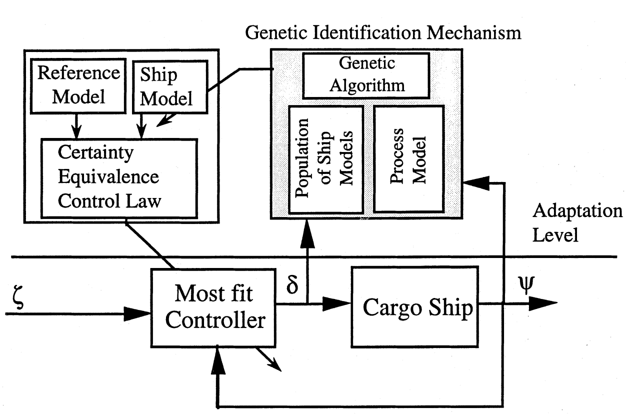

Fig. 9. Indirect genetic adaptive control for cargo ship steering.

The following pseudo-code defines the fitness evaluation used in the genetic identification algorithm. For each plant model candidate in the population, P_i, that is characterized by $P_i = (k_{\mathrm{d}i}, p_{\mathrm{d}i})$, do the following:

1. Initialize the discrete-time model in Eq. (12) with the past discrete-time samples of the cargo ship heading.

$$\hat{\psi}_i(k - N - j) = \psi(t - NT - jT), \quad j = 0, 1.$$

2. Using Eq. (12), plant model candidate P_i, and the past $N = 200$ samples of the rudder angle input, $\delta(k)$, estimate the past N samples of the cargo ship heading, $\psi(k)$.

For $j = 1$ to N

$$\begin{aligned}\hat{\psi}_i(k - N + j) = {} & k_{\mathrm{d}i}\delta(k - N + j) \\ & - (-1 - p_{\mathrm{d}i})\hat{\psi}_i(k - N + j - 1) \\ & - p_{\mathrm{d}i}\hat{\psi}_i(k - N + j - 2),\end{aligned} \quad (13)$$

next j.

3. Compute the error between the estimated output, $\hat{\psi}$, and the actual sampled cargo ship heading, ψ, using

$$\hat{e}_i = \sum_{j=1}^{N}\left[\psi(k - N + j) - \hat{\psi}_i(k - N + j)\right]^2.$$

4. Assign fitness, J_i, to each plant model candidate:

$$J_i = \frac{\alpha}{\hat{e}_i + \alpha}.$$

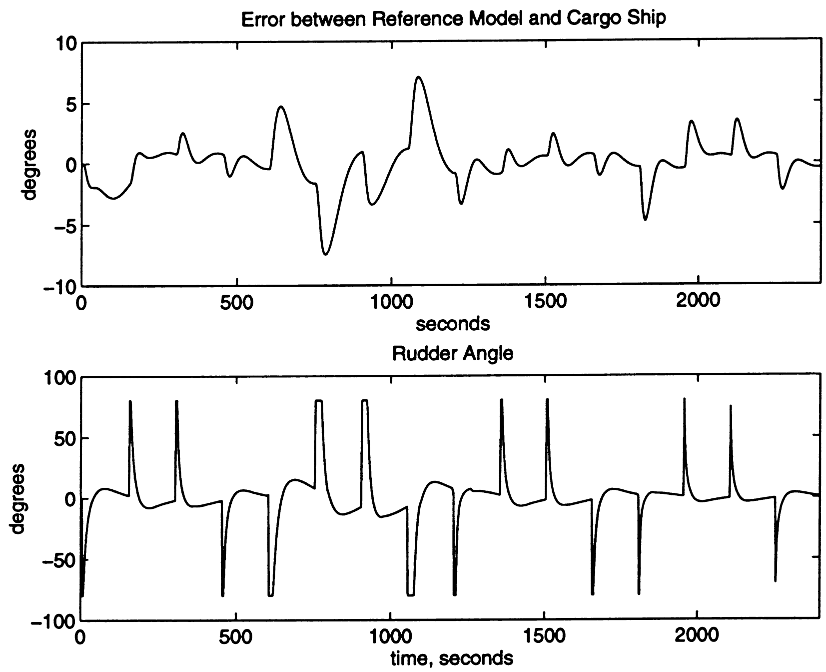

Fig. 10. Results using indirect genetic adaptive control. The top plot shows the error between the reference model output and the cargo ship response. The bottom plot shows the rudder angle, δ.

Here $\alpha = 0.02$ and was selected for the same reasons as discussed for the GMRAC.

5. Repeat steps 1–3 for each member of the population.

6. The maximally fit cargo ship model becomes the model used for the next time step.

The model estimation window, N, was set to 200 samples (100 s). Large values for N obviously increase computation time, but also improve the estimation performance of the identifier. Small values for N increase the likelihood that a "bad" plant model will be selected that does not accurately estimate the long-term behavior of the actual system, thereby causing a "bad" controller to be selected that adversely affects the closed-loop system performance.

One problem with genetic adaptive identification is that the GA attempts to minimize the prediction error of the cargo ship model; it does not necessarily find the best parameters to identify the plant. Therefore, the parameters the GA determines to be the best for one time instant may be far removed from the parameters the GA found just one time instant before. The result is that the GA switches between many plant models and hence the parameters it identifies are somewhat erratic. Having plant parameters that move quickly cause the controller parameters to also move quickly, which results in a noisy control signal. To compensate for this, the plant parameters are passed through a low-pass filter[5], the discrete-time (zero-order hold, $T = 0.5$) equivalent of $F(s) = 0.01/s + 0.01$.

Once the filtered estimates of the discrete-time plant model parameters k_d and p_d are obtained, they can be transformed into continuous-time plant model parameters k_c and p_c using the conversions in Eq. (11). The controller p and d gains are then adjusted using Eq. (9) and the current values of k_c and p_c.

The results are shown in Fig. 10. As was the case for the least-squares adaptive controller, it is difficult to identify the parameters of a non-linear system using a discrete-time plant with large sampling interval. However, another problem with the genetic identification algorithm is that the GA attempts to minimize the prediction error (which it does very well); it does not attempt to find the best parameters for the cargo ship model. While it is often the case that the model with the minimum prediction error will be the model with the best parameters, it does not apply to non-linear systems where there is not necessarily an "ideal" linear system model. The GA does very well at switching between cargo ship models to minimize the prediction error, but pays no attention to finding an "optimal" cargo ship model. Hence, using the cargo ship model parameters to define a controller is a difficult task[6].

[5] An alternative to this approach would be to lower the mutation probability, but it has been found that this method works better.

[6] One approach to solving this problem would be to use a parameterized feedback-linearizable model for the identifier structure and a certainty equivalence control law that is based on differential geometric methods.

5. General genetic adaptive control

In this section, the direct and indirect genetic model reference adaptive control are combined into a general genetic adaptive controller (GGAC). In particular, a genetic algorithm is used to identify a cargo ship model (as used in the indirect genetic adaptive controller), and another genetic algorithm is used to evolve the best controller (as used in the GMRAC). Fig. 11 shows the general genetic adaptive control system.

In this control technique, the discrete-time third-order linear plant model obtained by discretizing Eq. (4) using the bilinear transformation

$$G(z) = \frac{k_d(z+1)(z+1)(z-z_d)}{(z-1)(z-p_{d1})(z-p_{d2})}$$

is used.

The genetic identification algorithm identifies the parameters k_d, z_d, p_{d1}, and p_{d2}. Note that these parameters are different than those found in the indirect adaptive control technique because previously a second-order plant model was assumed and the zero-order hold transformation to approximate the discrete time model was used.

The plant model chromosomes consist of 16 digits, allotting four digits for each of the four parameters. The parameters are restricted as follows:

$$-0.001 < k_d \leq 0, \quad 0.8 \leq p_{d1}, p_{d2}, z_d < 1.0.$$

The fitness function is identical to the one described previously for the indirect genetic adaptive controller, except the model estimation window, N, was decreased from 200 samples to 100 samples to decrease computation time. The probability of crossover was $p_c = 0.8$, the probability of mutation was $p_m = 0.1$.

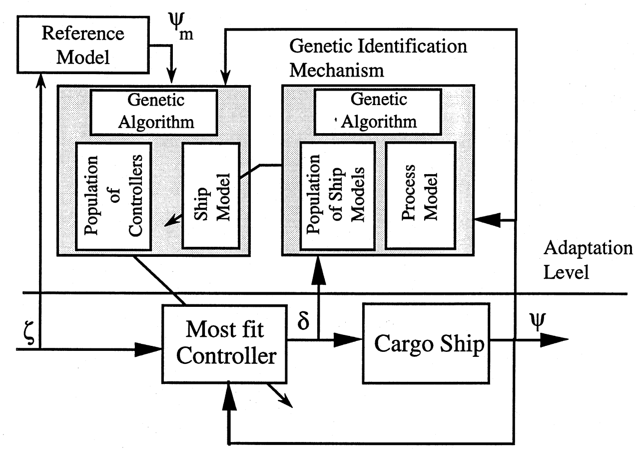

Fig. 11. General genetic adaptive control for cargo ship steering.

The best cargo ship model parameters are passed directly to the genetic adaptive controller which uses them in its fitness evaluation of the population of controllers. The genetic adaptive controller is identical to the genetic model reference adaptive controller described previously, with the obvious exception that the plant model used in the fitness function is continually adapting.

The results using the GGAC are shown in Fig. 12. The GGAC does an excellent job of tracking the refer-

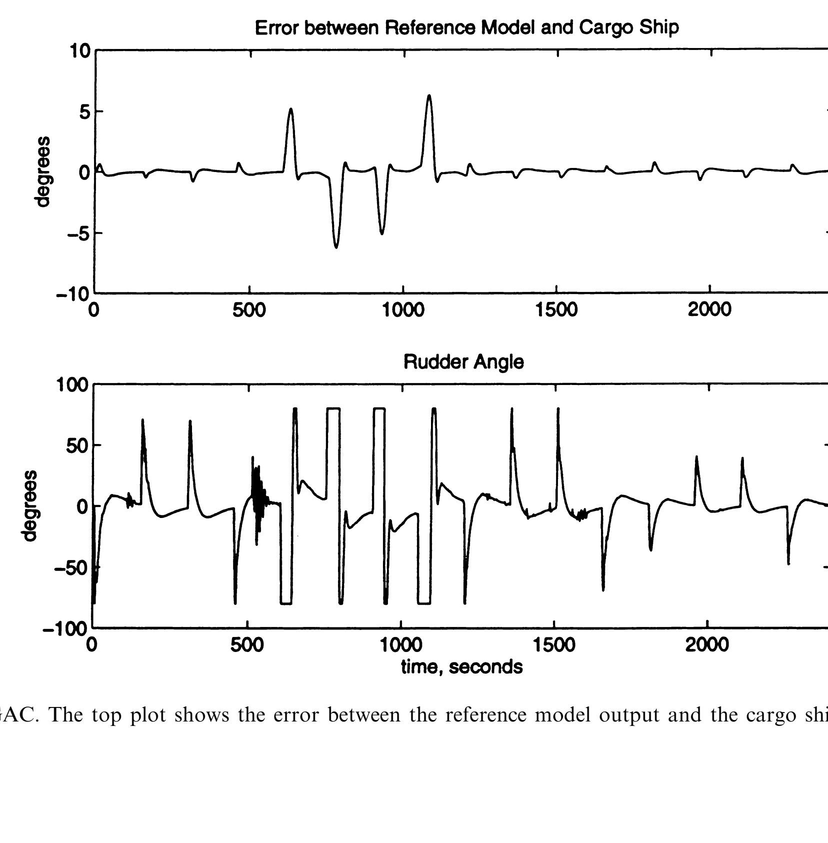

Fig. 12. Results using GGAC. The top plot shows the error between the reference model output and the cargo ship response. The bottom plot shows the rudder angle, δ.

ence model. In fact, the GGAC performs the best of all the control techniques investigated in this paper at tracking. However it should be noted that the GGAC also requires a large amount of input energy and quick movement of the rudder, which may be difficult to achieve in practice.

The GGAC was also run with fixed controllers, identical to the ones used for the GMRAC with fixed controllers. The results were not only more consistent (as measured by the variations in the tracking error), but they were also better in terms of reference model tracking (i.e. the tracking was smaller on average with fixed controller models than without them). However, the GGAC with fixed controllers required on average greater input energy than the GGAC without fixed controllers. The results for this case are summarized in Table 1.

6. Concluding remarks

In this paper several approaches to genetic adaptive control have been introduced and a comparative analysis between several conventional and genetic adaptive controllers for a ship steering application has been provided.

6.1. Summary of results

To see how all the controllers perform relative to each other, consider Table 1 which shows numerical results for the simulations in this paper. The error (in degrees) between the cargo ship heading and the reference model was sampled every 0.1 s, this error was squared and the sum of the squared errors was summed over the entire simulation. The same was done for the rudder angle input, δ, summing the squared error (in degrees) measured every 0.1 s. For the genetic adaptive controllers, 100 simulations were conducted and the minimum, average and maximum values of the sums of the squared errors are provided; the same was done for the rudder inputs.

As mentioned previously, the general genetic adaptive controller performed the best on average when measured by the ability to track the reference model. However, it also required a large amount of input energy, as measured by δ^2 and shown in Table 1. While the general genetic adaptive controller performed very well, there are still many uncertainties about this technique. No proofs of stability, convergence, or robustness have been established. Moreover, there is currently no way to prove that the schemes will achieve a specific desired transient performance. In addition, there is currently no way to verify that the genetic adaptive controllers will ever find an optimal controller in the space of candidate controllers.

The indirect discrete-time least-squares adaptive controller uses the least control energy but achieved poor tracking. The indirect continuous-time gradient adaptive controller performed very well as a compromise between reference model tracking and minimum input energy. The indirect genetic adaptive controller did not perform particularly well. The conventional MRAC performed the worst, both in tracking performance and input energy. Finally, the genetic model reference adaptive controller tracked very well, but it too required a large amount of input energy.

6.2. Computational complexity

Perhaps the biggest concern with genetic adaptive control techniques is the computational complexity of the algorithm. To better understand the computation time of the genetic algorithms, we carefully examined our simulation program (written in the C language) and computed roughly the number of operations required per generation. Using the variables P to represent the population size, N to represent the look-ahead time window steps in the fitness function, and C to represent the length of the chromosome, the following equation was arrived at:

$$O = P(P + 40N + 20C + 60).$$

Here, O represents the number of operations per generation (i.e. per time step) of the genetic algorithm, where an operation is any addition, multiplication, subtraction, division, assignment, increment, comparison, or declaration. This equation represents a rough estimate; we were careful to overestimate calculations when simplifying this equation.

Using this equation, it can be seen that the GMRAC requires roughly 760,000 operations per generation or 1.52 million operations per s (assuming a sampling time of $T = 0.5$ s). This number is less for the case with fixed population members. The indirect genetic adaptive controller requires approximately 836,000 operations per generation or 1.7 million operations per s. The GGAC in Section 5 uses two genetic algorithms and requires approximately 3.2 million operations per s. Of course, this computation time could be reduced with more streamlined code and a smaller population size and chromosome length. Because this research used simulations, there was virtually no attempt to minimize the computation time. It is clear, however, that substantial improvements could be made in terms of processing time. Nevertheless, with the cheap and powerful microprocessors widely available today, a controller that requires 3.2 million operations per s is certainly implementable.

Acknowledgements

The authors would like to thank the anonymous reviewers for their helpful comments. W. Lennon would like to acknowledge the partial support of a Center for Automotive Research Fellowship.

References

Antsaklis PJ, Passino KM (eds) 1993. An Introduction to Intelligent and Autonomous Control. Klumer, Norwell.

Åström, K., Wittenmark, B., 1989. Adaptive Control. Addison–Wesley, New York, p. 356.

Daihee, Park, Kandel, A., Langholtz, G., 1994. Genetic-based new fuzzy reasoning models with application to fuzzy control. IEEE Transactions on Systems, Man and Cybernetics 24 (1), 39–47.

DeJong, K., 1980. Adaptive system design: a genetic approach. IEEE Transactions on Systems, Man and Cybernetics 10 (9), 566–574.

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

Ioannou, A., Jing, Sun, 1996. Robust Adaptive Control. Prentice–Hall, Englewood Cliffs, NJ, pp. 372–396.

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

Lee, M.A., Hideyuki, Takagi, 1993. Integrating design stages of fuzzy systems using genetic algorithms. Second IEEE International Conference on Fuzzy Systems, San Francisco, CA. IEEE Press, Piscataway, NJ, pp. 612–617.

Lennon, W.K., Passino, K.M., 1995. Intelligent control for brake systems. In: Proceedings, IEEE International Symposium on Intelligent Control, Monterey, California. IEEE Press, Piscataway, NJ, pp. 499–504.

Maclay, D., Dorey, R., 1993. Applying genetic search techniques to drivetrain modeling. IEEE Control Systems 13 (2), 50–55.

Michelewicz, et al., 1990. Genetic algorithms and optimal control problems. In: Proceedings of the 29th Conference on Decision and Control, Honolulu, Hawaii. IEEE Press, Piscataway, NJ, pp. 1664–1666.

Narendra, Kumpati, S., Jeyendran, Balakrishnan, 1994. Improving transient response of adaptive control systems using multiple models and switching. IEEE Transactions on Automatic Control 39 (9), 1861–1866.

Passino, K.M., 1996. Towards bridging the perceived gap between conventional and intelligent control. In: M.M. Gupta, N.K. Sinha (Ed.). Intelligent Control: Theory and Applications, Chap. 1. IEEE Press, Piscataway, NJ, pp. 1–27.

Porter, B., Borairi, M., 1992. Genetic design of linear multivariable feedback control systems using eigenstructure assignment. International Journal of Systems Science 23 (8), 1387–1390.

Porter, L., Passino, K., 1994. Genetic model reference adaptive control. In: Proceedings, IEEE International Symposium on Intelligent Control, Columbus, Ohio. IEEE Press, Piscataway, NJ, pp. 219–224.

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

Renders, J., Hanus, R. 1992. Biological learning metaphors for adaptive process control: a general strategy. IEEE International Symposium on Intelligent Control, Glasgow, Scotland. IEEE Press, Piscataway, NJ, pp. 469–474.

Varšek, A., Urbančič, T., Filipič, B., 1993. Genetic algorithms in controller design and tuning. IEEE Transactions on Systems, Man and Cybernetics 23 (5), 1330–1339.

Zuo, W., 1995. Multivariable adaptive control for a space station using genetic algorithms. In: IEE Proceedings on Control Theory and Application, 142, 2, pp. 81–87.