PERGAMON

Engineering Applications of Artificial Intelligence 14 (2001) 1–14

Engineering Applications of
ARTIFICIAL
INTELLIGENCE

www.elsevier.com/locate/engappai

Genetic adaptive control for an inverted wedge: experiments and comparative analyses☆

Mathew L. Moore, John T. Musacchio[1], Kevin M. Passino*

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

Abstract

The inverted wedge is a planar robot with two degrees of freedom and a single control input (i.e., it is "underactuated"). The goal is to develop a digital controller that can balance the wedge in the inverted position by shifting a weight on the top of the wedge. Because it is underactuated and has complicated nonlinear dynamics, the inverted wedge is a good testbed for the development of nonconventional advanced control techniques and comparative analysis between control methods. We begin with the development of a nonlinear state feedback controller and direct and adaptive fuzzy controllers, that we will later use as a baseline comparison to show what type of performance is possible for this testbed. Control routines based on the GA have been found to apply to several practical applications in simulation and off-line optimization. Here, we will show that a GA can be used *on-line* in real-time to produce a particularly effective adaptive control method and this is the main contribution of this work. Computational and real-time implementation issues will be discussed and the genetic adaptive strategy will be compared with the state feedback and fuzzy control methods.

Keywords: Genetic algorithms; Adaptive control; Laboratory experiments

1. Introduction

Using conventional techniques, designing controllers for plants with nonlinear dynamics and modeling uncertainties can be quite challenging. By making use of heuristics, intelligent control techniques potentially can greatly simplify the synthesis of a controller for such plants. Although the analysis of intelligent techniques such as fuzzy systems, neural networks, and evolutionary programming, has in many cases been widely studied and show promising results, these techniques are used relatively infrequently in industry. This is possibly because of the lack of experimental implementation and analysis of these techniques.

This paper examines the implementation of an adaptive control method (the genetic model reference adaptive controller (GMRAC)) based on the genetic algorithm (GA) discovered by Holland in 1975, and compares the results to those from direct and adaptive fuzzy controller (a fuzzy model learning reference controller (FMRLC)) and a conventional nonlinear state feedback controller. The GA, an optimization routine based on principles of genetics and evolution, performs a directed random search of a population of controllers to determine which one is the best to implement at a specific sampling time period. The GA has found numerous applications in off-line optimization in both signal processing and control problems. Here, however, we show that a *real-time* implementation of a GA for adaptive control is not only possible, but it is also a worthwhile controller synthesis technique.

The inverted wedge experiment, an underactuated planar robotic system with two degrees of freedom, acts as the testbed for comparing control techniques. Its complexity is simple enough to allow for the development of a crude mathematical model of the system, but, due to the nonlinearities, has enough sophistication to render the standard linear techniques, e.g., proportional-integral-derivative (PID) and linear quadratic regulator (LQR), unsuccessful in physical

☆This work was supported by a National Science Foundation grant.

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

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

[1] J. Musacchio is now with the University of California, Berkeley, USA.

0952-1976/01/$ - see front matter
PII: S0952-1976(00)00047-6

implementation. Coupling between the two degrees of freedom is nonlinear, and due to this and the under-actuation, the system demands the controller to be robust to friction and model uncertainties.

The GMRAC was first introduced in Porter and Passino (1994), with the full version of this work in Porter and Passino (1998). A simulation study of on-line genetic adaptive techniques and the FMRLC as applied to a brake system is discussed in Lennon and Passino (1999a). An indirect genetic adaptive scheme was introduced in Kristinsson and Dumont (1992). In Porter and Passino (1995), the authors show how the GA can be used to adapt observer gains to obtain good state estimation. Another study of the GA for state estimation can be found in Gremling and Passino (1997a), (1997b), and a study of failure estimation using the GA for an automated highway system can be found in Gremling and Passino (1997a, 1997b). In Lennon and Passino (1999b), the authors examine several genetic adaptive strategies and apply them to a ship steering problem. The FMRLC algorithm was first introduced in Layne and Passino (1992, 1993). Implementation studies of the FMRLC can be found in Moudgal (1995). In this paper, we examine direct fuzzy and the FMRLC algorithms applied to a new experiment — the inverted wedge. The main contribution of this paper is, however, the first implementation of the on-line genetic adaptive technique given in the GMRAC algorithm.

This paper is organized in the following manner: first the details of the inverted wedge experiment are introduced and a nonlinear mathematical model is developed. Then a brief overview of the state feedback control law used on the system and the results of this controller are given. The next two sections give a description of the direct fuzzy and adaptive fuzzy methods that were applied, and the results of the implementation of these controllers on the inverted wedge are shown. Then we introduce the GA and the GMRAC, and discuss implementation issues of a real-time GA. Finally, we examine and compare the experimental results of each implementation.

2. The inverted wedge control problem

The inverted wedge, shown in Fig. 1, is similar to the inverted pendulum in that the goal is to regulate the system to zero-state (i.e., the balanced position). The system consists of a cart that rides along the top of the inverted wedge (the "sliding surface") whose position is controlled by a motor via a chain and gear system. Moving the cart along the sliding surface shifts the system's center of gravity, allowing the wedge to balance above its pivot point.

2.1. Dynamical model of the plant

Plant dynamics are described in a fixed world coordinate system (X, Y) whose origin is attached to the base at its pivot point of the wedge (point O). The input to the system, denoted $u(t)$, is a voltage level to the motor which controls the position of the cart (γ, measured from the center of the sliding surface, or $\vec{\rho}_1 + \vec{\rho}_2$ in the world coordinate system). The motor is attached to the center of the wedge framework. The angle of the wedge, α, is measured from the Y-axis, positive counter clockwise. Both γ and α are measured using optical encoders.

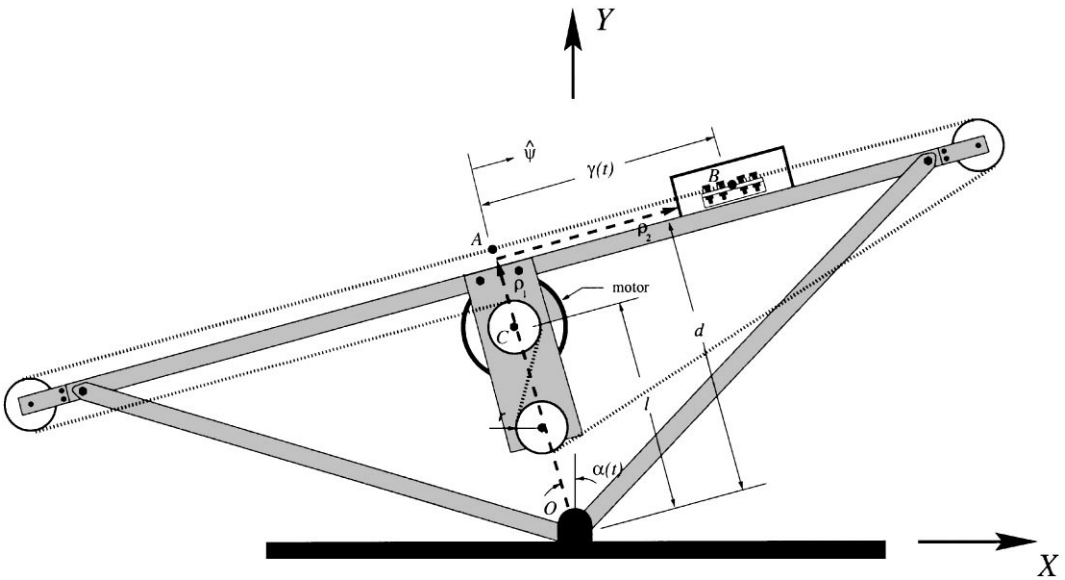

Fig. 1. The inverted wedge experiment.

The system can be modeled by a nonlinear function of the form

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

where $\boldsymbol{x}$ is the state of the system. The details of the vector equation can be found by examining the dynamics of the system. The states of the system are the position and velocity of the cart ($\gamma(t)$ and $\dot{\gamma}(t)$, respectively), the angular position and velocity of the wedge ($\alpha(t)$ and $\dot{\alpha}(t)$), and the motor current $I(t)$. Examining the forces along the direction of the sliding surface we obtain the equation

$$\ddot{\gamma} = \frac{1}{m}(F - F_{\mathrm{f}} - mg\sin\alpha),$$

where m is the mass of the cart, F_{f} is the frictional force, and F is the force supplied by the motor. The cart is coupled to the two tracks of the sliding surface using ball bearings, and viscous friction is assumed. Furthermore, we assume that the motor force is proportional to the current supplied to the motor. This results in the equation

$$\ddot{\gamma} = \frac{1}{m}\left(\frac{KI}{r} - K_{\mathrm{f}}\dot{\gamma} - mg\sin\alpha\right).$$

The summation of moments about the pivot point of the wedge yields the second equation, given as

$$\ddot{\alpha} = \frac{1}{J_{\mathrm{w}}}(Mgl\sin\alpha - mg(-d\sin\alpha + \gamma\cos\alpha) - m(d^2\ddot{\alpha} - d\ddot{\gamma} + \gamma^2\ddot{\alpha})). \tag{2}$$

In this equation, J_{w} is the moment of inertia of the wedge, d is the distance from the pivot point to the sliding surface measured perpendicularly. A final equation relates the controller output voltage the armature current of the motor. Using the standard DC motor model, we have

$$\dot{I} = \frac{1}{L}\left(u(t) - RI - K_{\mathrm{b}}\frac{\dot{\gamma}}{r}\right),$$

where R and L are the armature resistance and inductance, K_{b} is the back electromotive force (EMF) constant, and r is the radius of the gear in the drive system.

Using the system dynamic equations and the defined state vector (1) can be written as

$$\dot{\boldsymbol{x}} = \begin{bmatrix} x_2 \\ \frac{1}{m}(\frac{K}{r}x_5 - K_{\mathrm{f}}x_2 - mg\sin x_3) \\ x_4 \\ \frac{1}{J_{\mathrm{w}}}((Mgl - mg)\sin x_3 - mgx_1\cos x_3 \\ \quad + mdx_2 - m(d^2 + x_1)x_4) \\ \frac{1}{L}(u(t) - Rx_5 - \frac{K_{\mathrm{b}}}{r}x_2) \end{bmatrix}, \tag{3}$$

where the state vector $\boldsymbol{x} = [x_1, x_2, x_3, x_4, x_5]^{\mathrm{T}} = [\gamma, \dot{\gamma}, \alpha, \dot{\alpha}, I]^{\mathrm{T}}$, and the plant parameters are given in Table 1.

Table 1
Summary of plant parameters and variables

Plant parameters		
J_{w}	1.4029 kg m^2	inertia of wedge
M	4.86 kg	mass of wedge
m	1.25 kg	mass of cart
l	0.14 m	distance between pivot point and COG of wedge
d	0.17 m	distance between pivot and sliding surface
r	0.019 m	radius of gear
K_{f}	20.0 N/m/s	friction coefficient
R	1.9 Ω	motor resistance
L	0.004 H	motor inductance
K	14.8 oz-in/A	motor torque constant
K_{b}	11.0 V/kRPM	back EMF constant

Note that the wedge system does not fit the parameter strictly or pure feedback forms given in Krstić et al. (1995), making nonlinear controller design using Lyapunov theory difficult. This is coupled with the fact that although the dynamics of the plant may be well known, many of the parameters listed in Table 1 are, at the very least, uncertain. As a result, direct use of feedback linearization techniques is not feasible. Furthermore, certain unmodeled dynamics make it a challenging problem. These include the fact that the drive belt (which is two nylon-covered metal cables with nylon teeth in between) attached to the cart and motor tends to bounce and stretch. We have also not modeled the effects of Coreolus acceleration of the cart, and the centrifugal force component on the cart due to the wedge (the simplifications allow the development of a model that can be analytically linearized). Furthermore, the belt had to remain fairly loose to minimize the gear friction terms (also not modeled in the derivation), and the resulting chain backlash that occurred because of this seemed to have a significant effect on the plant. Other unmodeled nonlinearities include a large deadband in the motor and the changing cart friction that results from not keeping the sliding surface lubricated. Although any one of these may have minimal effect on the system, the combined effect of all unmodeled dynamics seemed to largely influence the wedge behavior.

2.2. Sensing and actuation

All control algorithms were implemented digitally using a computer program with a sampling period of $T_{\mathrm{s}} = 0.005$ s (smaller sampling intervals did not improve performance, and larger ones made it impossible to meet the control objectives). Optical encoders were used to measure both the position of the cart on the sliding surface and the angular position of the wedge (measured from the global Y-axis). From these measurements, the velocities of the cart and wedge are approximated within the computer program using a backward-difference. The

calculation of the velocities utilized a digital filter in order to remove the transients which occur due to the sensing technique. Access to the analog plant from the computer was obtained via a Keithley DAS-20 data acquisition card, which offers both A/D and D/A converters and an 8-bit digital interface, as well as a custom made circuit to convert the output of the optical encoders to a digital count read into the computer.

3. State feedback controller development

In this section we discuss the linear quadratic regulator (LQR) approach that, although unsuccessful for this plant, led to the development of a nonlinear state feedback controller.

3.1. LQR and state feedback control

In order to develop a linear controller for the plant, it is necessary to develop a linear model of the system, of the form

$$\dot{\boldsymbol{x}}(t) = \boldsymbol{A}\boldsymbol{x}(t) + \boldsymbol{B}u(t),$$

where $\boldsymbol{A}$ and $\boldsymbol{B}$ are found by solving the Jacobian around the balanced point $\boldsymbol{x} = \boldsymbol{0}$, $u = 0$. Doing this gives the resulting equation

$$\boldsymbol{A} = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ 0 & -\frac{K_f}{m} & -g & 0 & \frac{K}{mr} \\ 0 & 0 & 0 & 1 & 0 \\ -\frac{mg}{J_w} & \frac{md}{J_w} & \frac{g(md-Ml)}{J_w} & \frac{-md^2}{J_w} & 0 \\ 0 & -\frac{K_b}{Lr} & 0 & 0 & -\frac{R}{L} \end{bmatrix},$$

$$\boldsymbol{B} = [\,0 \quad 0 \quad 0 \quad 0 \quad \tfrac{1}{L}\,]^{\mathrm{T}},$$

where the values of the parameters are given in Table 1.

Since the states of the linear system are controllable, state feedback methods are applicable to the linear system. The feedback gains were originally chosen to solve the infinite horizon linear quadratic regulator (LQR) problem, which consists of minimizing the cost function given by

$$J = \int_0^\infty (\boldsymbol{x}^{\mathrm{T}}\boldsymbol{Q}\boldsymbol{x} + u^{\mathrm{T}}\boldsymbol{R}u)\,\mathrm{d}t, \tag{4}$$

where the matrices $\boldsymbol{Q}$ and $\boldsymbol{R}$ are used to weigh the states and input. Matlab can be used to solve the algebraic Riccati equation that produces the gain vector $\boldsymbol{k}$ that results from the minimization criteria and the control law is implemented by the equation

$$u(t) = -\boldsymbol{k}\boldsymbol{x}.$$

However, although successful in simulation studies, the LQR performed poorly in the implementation for a variety of chosen weighing matrices, forcing the introduction of a nonlinear feedback term and some heuristic tuning of controller gains. This nonlinear feedback term represents the position on the sliding surface, the cart *should* be located to balance the wedge and is calculated from the formula

$$\gamma_{\mathrm{eq}} = \frac{d\sin\alpha + Ml\sin\alpha/m}{\cos\alpha}, \tag{5}$$

where recall, d is the distance measured perpendicular from the sliding surface to the wedge's pivot point, l is the distance from the pivot point to the center of gravity of the wedge and m and M are the masses of the cart and wedge, respectively. Since this nonlinear term also approaches zero as the wedge angle approaches zero, the modified state feedback law will still attempt to regulate the system to zero state. Intuitively, the addition of this term places more emphasis on the effort of balancing of the wedge and relies on the fact that, for the wedge to be perfectly balanced, the cart must (eventually) be in the center of the sliding surface.

This new resulting state feedback law implemented is given by

$$u(t) = k_1(\gamma - \gamma_{\mathrm{eq}}) + k_2\dot{\gamma} + k_3\alpha + k_4\dot{\alpha}.$$

The final values for the gains, which were obtained via heuristic tuning, were $k_1 = 181.14$, $k_2 = 23.76$, $k_3 = -41.04$, $k_4 = -47.52$.

3.2. State feedback results

The heuristic nonlinear state feedback controller results are shown in Fig. 2. Initial conditions place the cart in the middle of the sliding surface at rest, with the wedge at a stationary angle of about 15°. Note that the state feedback controller only balances the wedge within a small steady-state value, partially due to a large deadband in the actuation. Attempts to counteract this,

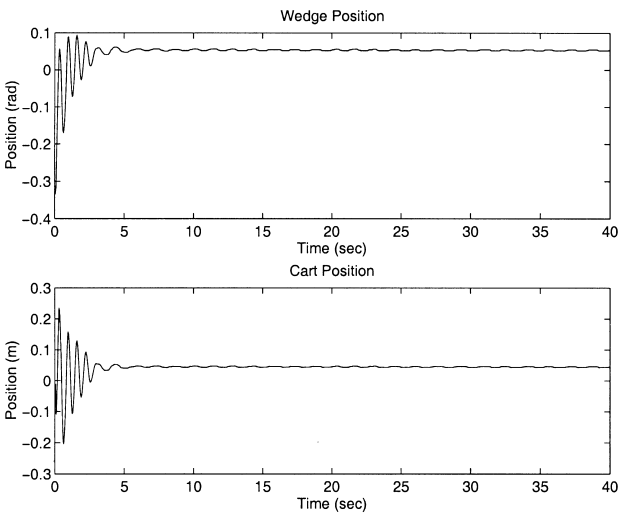

Fig. 2. Result of wedge balancing using the state feedback controller.

by slightly increasing the absolute value of the controller output when close to zero helped, but did not eliminate this problem (and, in fact, tended to cause greater oscillation).

4. Direct and adaptive fuzzy control

Fuzzy control provides a nonlinear mapping between state values and controller output, and also allows the use of heuristics in the design. This is beneficial in this case, since for the most part we can determine what the controller should do in any given case. This section discusses the development and results of both direct and adaptive fuzzy controllers as applied to the inverted wedge.

4.1. Direct fuzzy control

The multiple input−single output (MISO) fuzzy controller used for balancing the inverted wedge utilized four inputs and produced the motor voltage input. The inputs to the fuzzy system consisted of the error, $e(kT_s) = \alpha_r - \alpha$, and the change of that error

$$\delta_e(kT_s) = \frac{(e(kT_s) - e((k-1)T_s)}{T_s},$$

and the cart position and discretized velocity,

$$\gamma(kT_s) \quad \text{and} \quad \delta_\gamma(kT_s) = \frac{(\gamma(kT_s) - \gamma((k-1)T_s)}{T_s},$$

multiplied by their respective gains, g_e, g_{δ_e}, g_γ, g_{δ_γ}. The cart variables were utilized since they were found to be useful to the controller system; the position of the cart is necessary to keep the controller from pushing the cart beyond the limitation of the physical system, and the velocity of the cart was used for similar reasons. In addition, these choices for inputs seem logical, since they could be considered as the state variables of the system without the motor.

A direct fuzzy controller implements a rule base that is generated from a set of IF−THEN rules. Typically, this set of rules is developed heuristically based on the knowledge of the plant. An example IF−THEN rule in the case of the inverted wedge system could be

IF e is positive − large AND δ_e is positive − large

AND ... THEN u is negative − large.

Note that this example rule does not explicitly name the additional two inputs from the cart as we only use it for illustrative purposes. Here we use linguistic variables "positive-large", "positive-medium", and "positive-small" to quantify the error terms that are large, medium or small positive values; likewise we would use terms such as "negative-large" if the error was a large negative value, and so on. To quantify what exactly is "positive-large", etc., we use the membership functions shown in Fig. 3. Their universes of discourse (that is, their domain) is normalized to cover a range of $[-1, 1]$.

The inverted wedge system with four inputs, and seven membership functions each, contains a rule base of 2401 $(= 7^4)$ rules. Also, because we are using symmetric, *overlapping* triangular-shaped membership functions (shown in Fig. 3), it is possible to have 16 $(= 2^4)$ rules on at once (Passino and Yurkovich, 1998). Fortunately, there is a large amount of symmetry

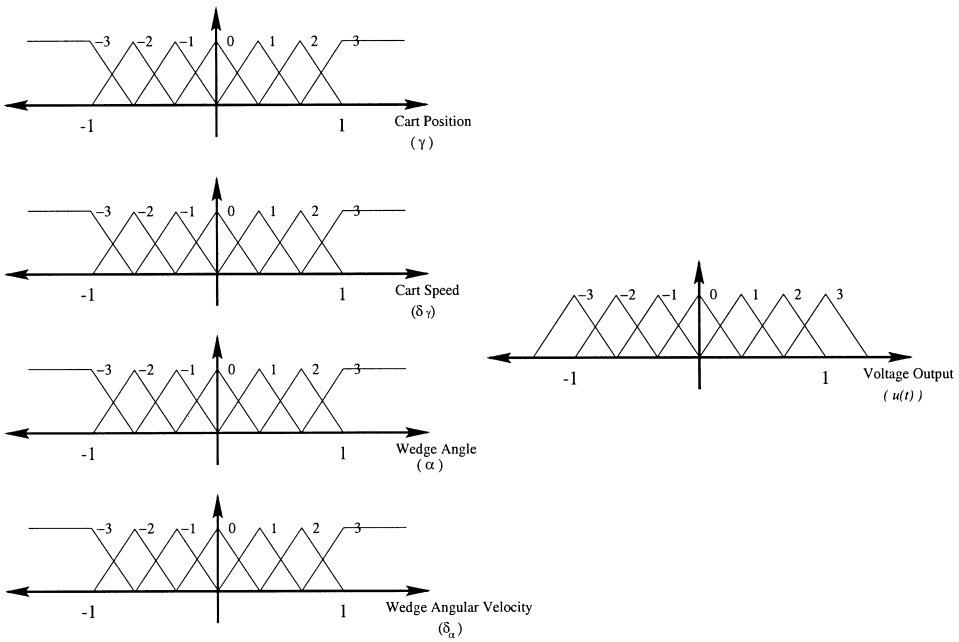

Fig. 3. Membership functions for the fuzzy controller.

between the respective rule tables. For example, consider the case where the position of the cart is zero (in the center of the sliding surface) and its velocity is zero. If the error e is positive large, than the angle of the wedge is too far clockwise (with reference value of zero radians), and a negative voltage must be applied to the motor to move the cart to the left. The amount of voltage applied depends on the present state of the change in error. If the change in error is such that the error is moving toward zero (what we want) then we do not need to apply as large a voltage as if the change in error was such that the error is moving away from zero. The same type of logic can be applied if the error was negative, corresponding to an angular position too far in the counterclockwise direction. The resulting rule table for the case for $\gamma = 0$ and $\delta_\gamma = 0$ is shown in Table 2. In this table the linguistic values of each of the inputs go from negative-large to positive-large, and are assigned the respective numeric values $-3-3$.

Note that the effect of changing γ and δ_γ simply *shifts* the rule base up or down in the main diagonal direction depending on their values. If the position of the cart is such that it is all the way to the left, we will not want to apply a voltage that will further push the cart to the left in the case that e is positive (for if e is negative, the controller will send out a voltage to move the cart to the right in order to balance the cart) as this would risk damage to the equipment. So when e and γ are opposite in signs and near their respective extremes, we tend to apply less of control voltage to the system than we would with e or γ zero or the same sign. The opposite is true if γ is a large positive value (to the right). Physical understanding of a plant of this type is necessary in order to successfully design the rule base for a fuzzy controller. We omit the details of the remaining 2352 rules ($2401 - 49$) in the interest of brevity and since once you think about the physics of the plant, the rule base is easy to construct.

To develop a crisp output from the fuzzified inputs, center-of-gravity defuzzification is employed and utilizes the output membership functions shown in Fig. 3. The crisp value is the resulting controller output. Finally, note that the fuzzy controller is not linear even with the above choices; since there is a low number of linguistic values and hence membership functions there is an "oscillation" on the shape of the nonlinear map that the controller implements. For more details on fuzzy control, consult Passino and Yurkovich (1998).

Table 2
Rule table for the fuzzy controller with $\gamma = 0$ and $\delta_\gamma = 0$

"Force" u		"Change in error" δ_e						
		-3	-2	-1	0	1	2	3
"Error"	-3	3	3	3	2	2	1	0
e	-2	3	3	2	2	1	0	-1
	-1	3	2	2	1	0	-1	-2
	0	2	2	1	0	-1	-2	-2
	1	2	1	0	-1	-2	-2	-3
	2	1	0	-1	-2	-2	-3	-3
	3	0	-1	-2	-2	-3	-3	-3

4.2. Direct fuzzy controller results

The above design of the fuzzy controller was successful at balancing the wedge only when α was initially within a range of $8-10^\circ$ from the vertical. The control energy utilized is quite significant and the cart tends to oscillate about the zero-position point. The results are shown in Fig. 4. The input and output gains were heuristically tuned to $g_u = 5.0$, $g_e = 2.0$, $g_{\delta_e} = 4.25$, $g_\gamma = 2.25$, and $g_{\delta_\gamma} = 1.25$, which seemed to perform the best.

The results here seem to show that modeling uncertainties tend to have a large effect on the performance of the system. The LQR could not balance the wedge at all when developed using dynamic models and linear approximations. Although the primary factors were taken into account with the development of model, there was no attempt to model the effects of the chain and gear system, which seem to have somewhat of a dominate effect on the system.

Since the state feedback controller could be significantly improved by adding the nonlinear feedback term, γ_{eq}, we attempted to introduce this same term to our direct fuzzy controller as well. The cart position input, γ, was substituted with the nonlinear term γ_{eq}. The results were significantly improved, as shown in Fig. 5, and, in fact, performs as well as the nonlinear state feedback method.

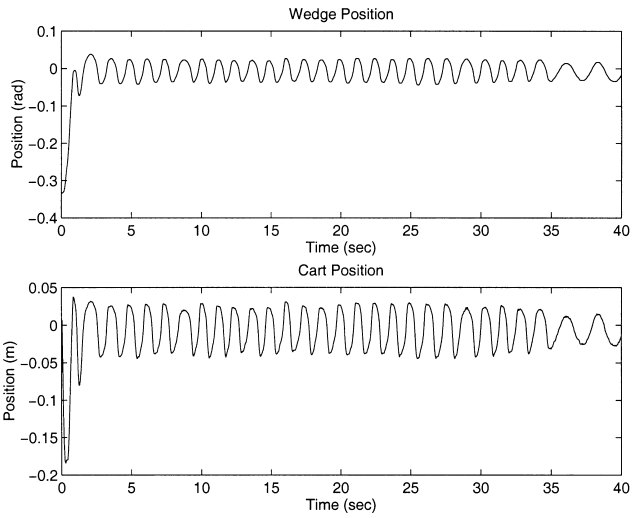

Fig. 4. Results of the direct fuzzy controller.

4.3. Adaptive fuzzy control

The direct fuzzy controller results discussed in the previous section showed that the cart and wedge still tended to remain offset from the zero point (a small steady-state error). From an intuitive sense, an adaptive control technique applied to the wedge may help to improve the response of the balancing, and eliminate the steady-state error that occurs. Here we design and implement a fuzzy model reference adaptive controller (FMRLC) which adaptively tunes the centers of the output membership functions on-line.

The success with the direct fuzzy controller from the previous section is utilized in the design of the FMRLC.

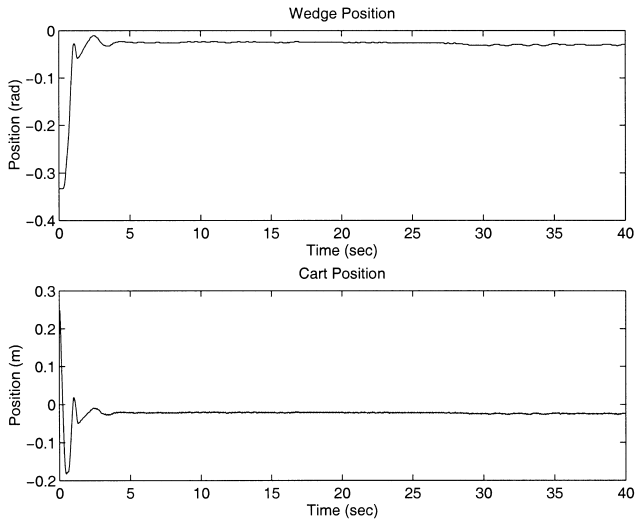

Fig. 5. Results of the direct fuzzy controller with nonlinear input term.

Specifically, the original fuzzy controller of the form just described still controls the system. However, an additional fuzzy system, called the "fuzzy inverse model", will be developed to change the output membership function centers based on the response of the physical system (Layne and Passino, 1992, 1993; Passino and Yurkovich, 1998). In other words, the inverse system will adapt to changes and perform self-tuning of the actual fuzzy controller.

Fig. 6 shows the FMRLC system as applied to the inverted wedge problem. It can be thought of as simply two fuzzy controllers. One of these acts as the true controller for the physical system (in this case, the wedge). The second fuzzy controller updates the rule base of the actual controller by adjusting the centers of the membership functions. In most cases the update law is based on the error between the actual output and the output of a reference model (denoted by $\dot{\boldsymbol{x}}_m = f_m(\boldsymbol{x}_m, u)$, $y_m = h_m(\boldsymbol{x}_m)$) that characterizes the *desired* performance of the system. This reference model represents how we would like the system to behave. In general, we could make the reference model a linear representation of the nonlinear dynamic equations of the plant (since we would like most systems to behave linearly). In the case of the inverted wedge, however, we utilize a reference model that is identically zero (that is, $y_m = 0$). This is beneficial in that it represents the desired state of the system (we wish to balance the wedge) and it is easy to implement.

The rule base for the fuzzy controller was developed heuristically using the same type of logic as for the direct fuzzy method. Once again, symmetrically spaced triangular membership functions were used. Note that since there are four inputs to the system up to 16 rules can be

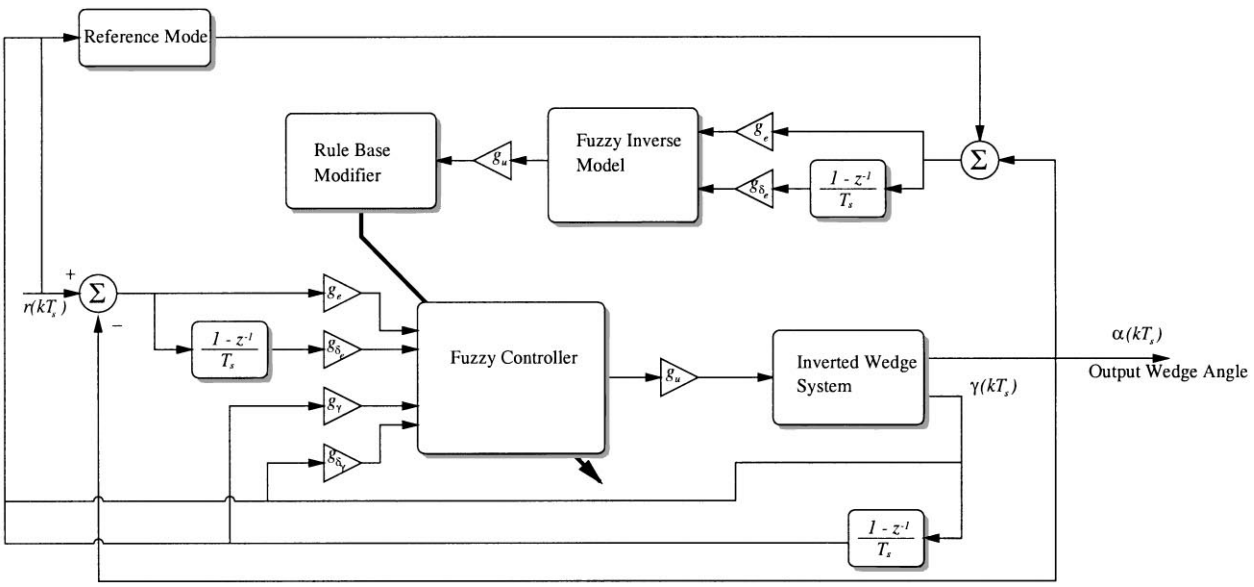

Fig. 6. The FMRLC applied to the inverted wedge problem.

on at one time (2^4). It is important to note that for true learning control, we must have a separate output membership function for each possible input combination. This way, when the output centers are updated by the inverse model, it will only be updating those rules which apply at that time instant (i.e., under the specific conditions the system is in), and will not change the outcome for other conditions. That is, the system will "remember" how to control the plant under a specific condition since it will only update the rules that apply to that condition when the system is in that state. This is a key advantage to this control method, since time consuming re-learning is avoided. Nine membership functions were used for each input, and 6561 ($= 9^4$) membership functions were used for the output. The initial output membership function centers were setup to give the same results as the direct fuzzy controller. That way, the controller initially has a good idea how to control the system.

The inverse fuzzy model is nearly an exact copy of the fuzzy controller. Note that since the reference model for this system is identically zero, the inputs to the updating inverse system will be the same as the inputs to the system controller, except in this case we only use the error and change in the error (e and δ_e). This seems to be a good way to design an update law, since we would like to adjust the system so that we have the best response as possible (i.e., we always want to balance the system as fast as possible). The inverse model also utilizes symmetric triangular-shaped membership functions. Five membership functions were used for the inputs and the output.

In actual implementation, the system introduces a certain delay into the system. That is, an input at a certain time step is not "seen" at the output until several time steps later. As a result, the active regions of the rule base are stored in an array, and the present condition of the system results in an update of active regions T_d time steps in the past, where T_d refers to the delay time of the system. In this case the delay was set to be two time steps. In addition, the inverse fuzzy model was "turned off" (i.e., the output centers of the actual controller are not changed) when the state was close to the balance point, to make the system more robust. Otherwise, the adaptation forces the controller to overactuate and exceed the limitations of the system. For more details on the FMRLC, including how to tune it and design the fuzzy inverse model, see Passino and Yurkovich (1998).

4.4. Results of the FMRLC

In order to obtain a stable controller for the system, adaptation gains in the inverse system were kept small. The final gains chosen were $g_e = 1.25$, $g_{\delta_e} = 1.5$, $g_{\gamma-\gamma_{eq}} = 5.0$, and $g_{\delta_\gamma} = 0.75$ for the controller fuzzy system and $g_e^{inv} = 0.025$ and $g_{\delta_e}^{inv} = 0.4$ for the inverse model. Furthermore, the output gain for the inverse system was $g_{out}^{inv} = 0.001$. The small adaptation gain results in the slower settling time for the system, but remains consistently stable.

Fig. 7 shows the results for the FMRLC. Note that the adaptive technique is much better than the direct fuzzy (without the nonlinear input term) at eliminating the oscillation. More importantly, the figure shows that, due to the adaptation, the error between the output and the desired value is much less. The inverse system output shows the rate of adaptation over time is plotted in Fig. 8 This shows that while early on periodic updates are made to the rule base, as time goes on the system learns and these updates become less frequent.

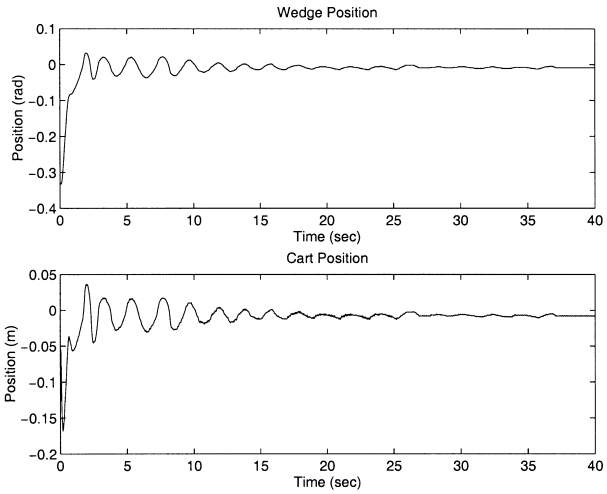

Fig. 7. Results of wedge balancing with the FMRLC control algorithm.

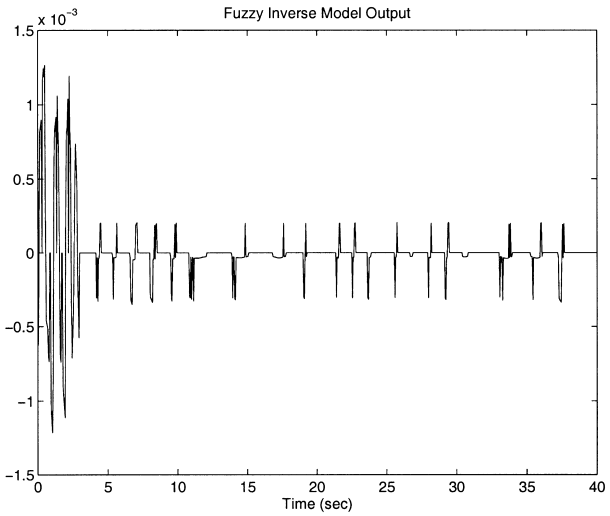

Fig. 8. Adaptation (inverse model) output and controller output over time.

5. Genetic-algorithm-based control

The GA is an optimization routine based on the principles of Darwinian theory and natural genetics. It has primarily been utilized as an off-line technique for performing a directed search for the optimal solution to a problem. Here, we show that the GA can be used on-line in real-time controller implementation to adaptively search through a population of controllers and determine the member most fit to be implemented over a given sampling period. This section starts with a brief introduction to the GA. Then we discuss how to implement the GA-based control algorithm, and the results of the implementation. These results will be compared to those obtained from the direct and adaptive fuzzy control techniques as well as the state feedback implementation.

5.1. The genetic algorithm

The GA performs a parallel, directed, random search for the fittest element of a population within a search space. The population simply consists of strings of numbers, called chromosomes, that hold possible solutions of a problem. The members of a population are manipulated cyclically through three primary "genetic operators" called selection, genetic operation (also referred to as crossover and mutation), and replacement, to produce a new generation (a new population) that tends to have higher overall fitness evaluation. By creating successive generations which continue to evolve, the GA will tend to search for a global optimal solution.

The key to the search is the fitness evaluation. The fitness of each of the members of the population is calculated using a fitness function that characterizes how well each particular member solves the given problem. Parents for the next generation are selected based on the fitness value of the strings. That is, strings that have a higher fitness value are more likely to be selected as parents, and, thus, are more likely to survive to the next generation. This first stage is the selection stage. Although there are many techniques for the selection of parents, a commonly used method is the Roulette Wheel Selection (Goldberg, 1989), a proportionate selection scheme which bases the number of offspring on the average fitness of the population. Strings with greater than the average fitness are allocated more than one offspring. The actual method of selection is relatively inconsequential — the key being that strings with greater fitness tend to produce more offspring.

For the evolutionary process, the selected members of a population are randomly paired together to "mate" and share genetic information. The GA accomplishes this sharing by the swapping of portions of strings between parents. The simplest model is the single-point crossover, where the selection of a random position between 1 and $l-1$, where l is the length of the chromosome, indicates which portions are interchanged between parents. Multiple crossover points can also be selected, where strings swap several portions of their strings. The resulting interchange produces two new strings. The actual interchange is often based on a crossover probability p_c. The sharing of genetic material occurs only if a randomly generated normalized number is greater than p_c; otherwise, the strings are not affected. Once this crossover stage is complete, the new strings are subject to mutation based on the mutation probability p_m. This genetic operation randomly selects a string and position (between 1 and $l-1$) and changes the value at that position to a random number. Since the variations due to the mutation operation occur randomly, this tends to keep the GA from focusing on a local minimum.

The final step in the GA is the replacement operation, which determines how the new subpopulation produced from the genetic operations will be introduced as a new generation. Two primary methods exist — complete generation replacement, where each population produces an equal number of strings, which then completely replace the parent population, and partial replacement where only a small portion of new strings are developed and introduced into the population. In addition, the former procedure may have a tendency to throw away a best-fit solution since the entire generation is replaced; for this reason, often the complete generation replacement method is combined with an "elitist" strategy, where a one or more of the most fit members of a previous population are passed, untouched, to the next generation. This tends to ensure that there always is a string within the population that is a good solution to the problem.

The genetic algorithm procedure, illustrated in Fig. 9, is fairly simple — start with a randomly (or specifically chosen) initial population of members and perform the three above operations based on the user-specified probabilities of occurrence. At the conclusion of this we have automatically produced the next generation. Then repeat the operators on each consecutive generation until the user is satisfied with the results (until a good maxima is found).

A key part of the success of the GA is the encoding of the strings. Obviously, we need to choose parameters that tend to have reasonable information about the problem. The actual number and type of variables that will be encoded as strings is application specific. Common encoding of multiple real-value continuous variables (as opposed to binary or digital variables) consists of placing them in integer form and concatenating them, keeping track of decimal places and variable separation points for decoding. Thus, two real numbers 12.345 and 9.8765 could be represented in a 10-digit long

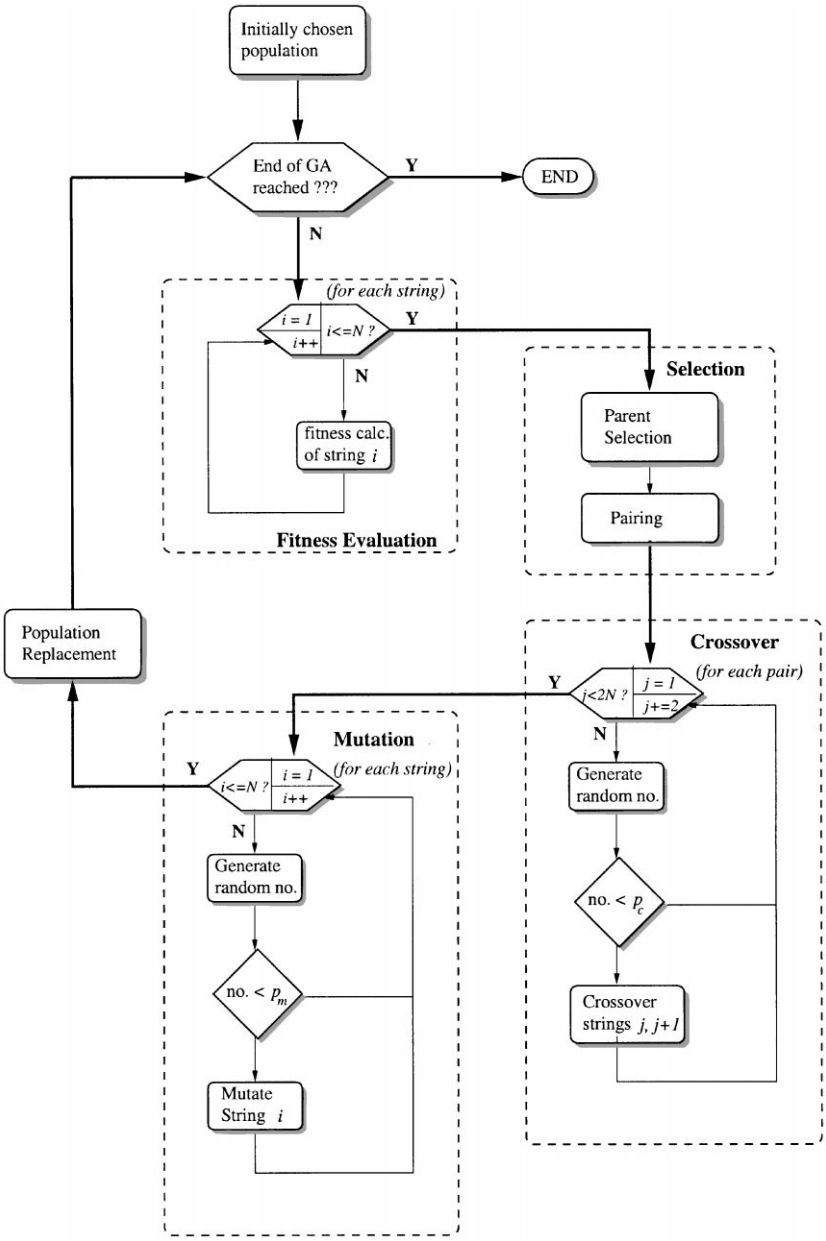

Fig. 9. Flow chart for a discrete implementation of the GA.

string 1234598765. Decoding can occur by first splitting the string at the appropriate point (in this case, between the fifth and sixth digit) and keeping track of the appropriate decimal places. In this case, the actual values are represented by multiplying the integer given by the first portion of the string by 10^{-3} and the integer in the second portion of the string by 10^{-4}. This is representative of the base-10 encoding of strings; a base-2 encoding is also popular, where strings are represented in the binary form as a series of 1s and 0s. Regardless of the base of the encoding, the procedure is the same. The strings are decoded for fitness evaluation, but remain in encoded form for genetic manipulation, producing successive generation until a user-specified termination criterion is reached.

Note that if we encode controller gains as the strings of the population and base the fitness value of each member on the performance of that particular controller the string represents on a model of the system, then we can use the GA to find the optimum controller. Furthermore, if we can implement the GA on-line, and use it to find the best controller at *every* sampling time (or every few sampling times) then we can use the GA to produce an adaptive control routine that will try to find the optimum controller to use under different conditions. This is the basis of the genetic model reference

adaptive control routine that was implemented on the inverted wedge system.

5.2. Real-time control using the GA — the GMRAC

Development of a control algorithm based on the GA involves searching for an optimal controller during each (or every several) sampling periods. The GMRAC uses the global adaptive search power of the GA to evolve a set, or population, of possible controllers during the real-time implementation of the control algorithm. The fitness evaluation in this case consists of characterizing the *expected* performance of each controller in the population based on error analysis. This type of evaluation requires both a model of the plant for prediction purposes and the development of a reference model for comparison. In this case, a nonlinear model was developed in Section 2. The resulting nonlinear model is used to predict ahead how well the controller will perform several time steps into the future. As with any model reference controller, the reference model characterizes the *desired* response of the system. That is, the fitness of each of the strings (controllers) in the population is based on how well it makes the model behave like the reference model. In this case, we wish for all the state values to go to zero, so a reference model that clearly characterizes this is $y_m = 0$ (i.e, the reference model output is identically zero). While the GA will evolve the set of controllers to obtain the best (predicted) time-response results, the fitness function evaluation is also used in real-time to select the controller that will be implemented every few sampling periods. That is, the most-fit controller determined by the fitness evaluation each generation cycle becomes the controller that is actually implemented on the plant during the next sampling periods.

The general form for the controller for the inverted wedge GMRAC follows that of the nonlinear state feedback law developed in Section 3.1. This seems to be a good choice given its relatively successful performance. The gains of the nonlinear state feedback controller are encoded as strings. A population of these strings are randomly selected, with the only restriction that the gains represented by the strings do not exceed the following boundaries: $-200 < k_1 < -150$, $-30 < k_2 < -10$, $20 < k_3 < 40$, and $25 < k_4 < 50$. This restriction stems from the examining the wedge system — we do not want to create the controller which will tend to overactuate beyond the limits of the system. We need to tell the GA only to produce strings within the range so that this will not happen. The fitness function which characterizes the fitness of each controller (string) in the population evaluates each string by first decoding them back into controller gains, then, using the plant model, predicts ahead four sampling times the values of the states that would occur given that that controller were actually implemented. Let $e_{i\mathrm{exp}}$ be the expected error four sampling times ahead if controller i ($0 < i \leq N$ where N is the number of members in the population) was implemented on the system. Furthermore, let $\delta_{e_{i\mathrm{exp}}} = (e_{i\mathrm{exp}(kT_s)} - e_{i\mathrm{exp}}((k-1)T_s))/T_s$ be the expected change in error of that controller. We can then define the fitness function as

$$J = \frac{1}{a_1 e_{i\mathrm{exp}}^2 + a_2 \delta_{e_{i\mathrm{exp}}}^2 + a_3 u^2}. \tag{6}$$

This represents the function we want to maximize (so we try and minimize $e_{i\mathrm{exp}}$, $\delta_{e_{i\mathrm{exp}}}$, and u). The fitness is evaluated for each string (controller), and the most fit one (the one with the highest fitness value) becomes the controller that is actually implemented during the next sampling period. The GA then evolves the next generation of controllers using the genetic operators and, thus, tends to search for the controller which optimizes the fitness function. In a heuristic description, the GA evolves a set of controllers to try to find an optimal controller. Note that in implementation, (6) will have to be checked for a divide by zero (or, alternatively, a small positive number can be added to the denominator). The final gains of the fitness function were chosen via trial and error and were found to be $a_1 = 50$, $a_2 = 100$, and $a_3 = 5$.

The size of the population was selected to be $N = 20$ controllers. This number was picked after considering the complexity and time consumption of the calculations necessary for the GA. Despite being a time consuming method, we were able to obtain a generation (and thus an updated controller) every 3 or 4 sampling intervals. Each string in the population has a length of 16 digits, four for each of the controller gains. The crossover probability was set at 1 and the mutation probability was set at 0.1. We found that using such a large mutation rate tends to improve the performance when such a small population size is used.

5.3. Results of the GMRAC

Results of the implementation of the GMRAC on the inverted wedge are shown in Fig. 10. Note that although the GMRAC tends to oscillate the system more at first, it still maintains a low settling time, and actually improves the steady-state error that occurred in the standard nonlinear state feedback case. Furthermore, the cart velocity for the GMRAC implementation tends to remain zero once the zero state is reached, whereas the other control methods tend to oscillate the cart back and forth very quickly around the zero position; as a result, the GMRAC also tends to use less control energy than the other methods. In fact, the response of the GMRAC is comparable to the FMRLC without needing the sudden jumps in control energy that the

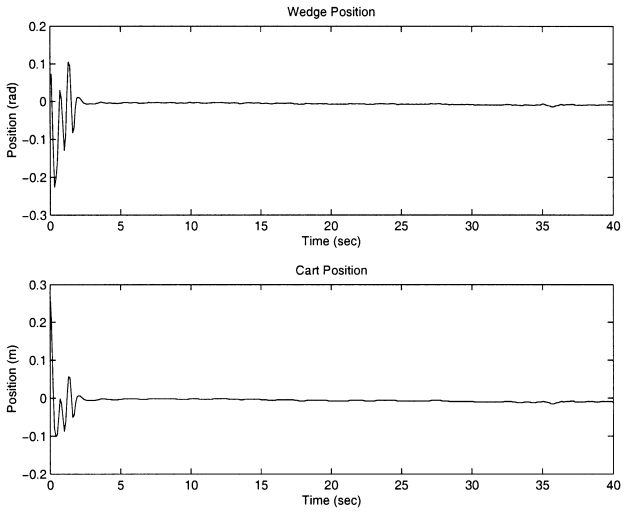

Fig. 10. Results of the GMRAC implementation.

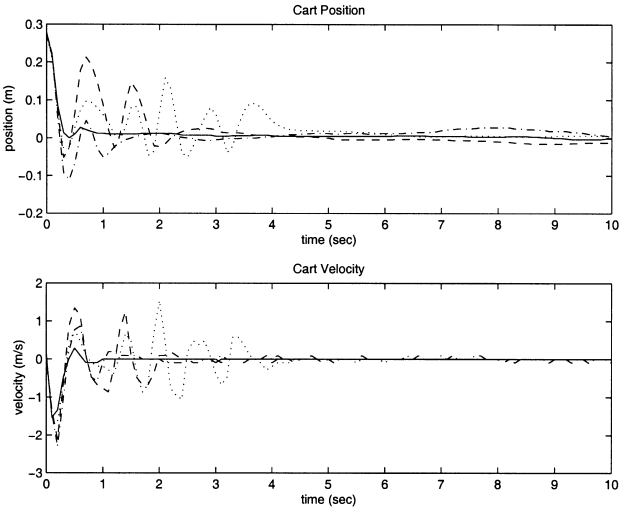

Fig. 11. Results of several trials runs of the GMRAC — cart position and velocity (note that the time scale is different from the earlier plots to illustrate the transient behavior better).

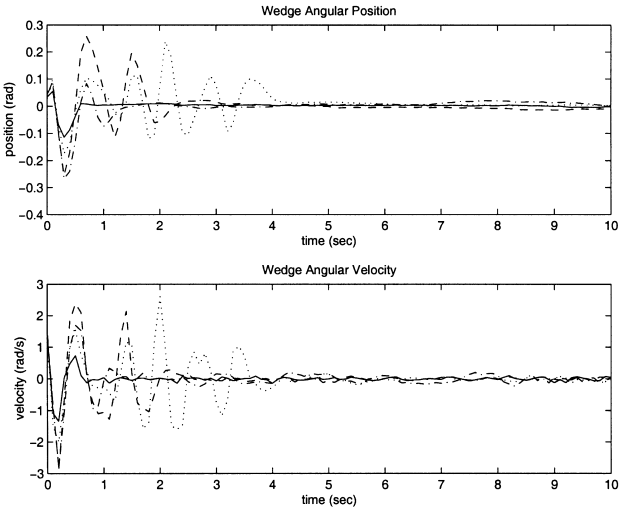

Fig. 12. Results of several trial runs of the GMRAC — wedge position and velocity (note that the time scale is different from the earlier plots).

FMRLC tended to exhibit, and it balanced the wedge faster.

Clearly, the GMRAC seems to be an extremely effective control method for this problem. However, there are several drawbacks. First of all, the GMRAC has a tendency to try and force a response too quickly, which is not possible due to the limits of the system. If the experiment is started under certain initial conditions, the cart will tend to jerk too quickly, resulting in ramming against the edge (at which point an emergency shutoff occurs). However, this behavior is even occasionally exhibited by the state feedback controller. Secondly, the GMRAC is a stochastic approach, and, as a result, results will vary from trial run to trial run. This will be examined in the next section.

5.4. Stochastic effects of the GMRAC

Since the GA is a directed *random* search, the results of this control technique will vary each time it is run. One hopes that by tuning parameters, consistent results will be obtained. To show that this is possible, several other trial runs were completed using the GMRAC and are shown in Figs. 11 and 12. Although there are cases where the GMRAC does not perform as well as the results shown in the previous section, for most cases the results are consistently as good as the other methods.

Difficulties in repeating the experiment 100 or even 1000 times make it impossible to provide a full stochastic analysis of the approach where we would provide, e.g., averages of responses. The focus in this first paper on real-time implementations of genetic adaptive control has been on how to get the method to operate in real time, how to tune the method, comparative analyses to show how it can offer advantages over other methods, and an overview of *some* of the stochastic aspects of the behavior of the method. It is beyond the scope of this paper to fully investigate a full stochastic analysis of the method; however, we are presently setting up an experiment in our laboratory that we can automatically run for a week or two; this will provide an experimental framework for a proper analysis of stochastic effects.

6. Conclusions and prospects for further use in industry: adaptive MPC

This paper studied the implementation of several intelligent control techniques as applied to the balancing

of the inverted wedge problem. These included a basic four-input direct fuzzy controller (including the use of the nonlinear input term) and an adaptive fuzzy control technique known as the FMRLC. The direct fuzzy controller provided a nonlinear mapping between inputs and outputs, and allowed the incorporation of heuristics (which model human deduction process) via the use of the rule table. Since we generally know the dynamics necessary to balance the wedge give an initial condition (obtained from studying the dynamics of the system), the use of heuristics in controller design via the fuzzy controllers proved very successful. The FMRLC took the controller design a step further by supplying an inductive update method which produced an adaptive fuzzy controller. The FMRLC incorporates an *inverse* fuzzy system which takes the plant output (α) compares it to the desired response obtained from a reference model, and updates the fuzzy controller's input−output map based on that error. This update occurs via the inverse system.

Then we introduced the GA, the search routine based on Darwinian theory and natural genetics, and showed how it could be applied to control problems to produce an adaptive control method (GMRAC). The GMRAC was then applied to the inverted wedge system, and proved to be rather successful at balancing the wedge. The intelligent controllers were compared to a conventional state feedback method (with nonlinear term). The intelligent methods certainly seemed to perform better than this conventional methods. However, there are certain tradeoffs required to achieve this success that must be kept in mind. These include both greater computational complexity (especially in the genetic-based technique) of the intelligent controllers, and the greater design time required.

It is recognized that the particular experiment we are using is a simple university contraption with little or no relevance to industrial applications. There is, however, relevance in the genetic adaptive control methodology that is proposed. How? To answer this question we will discuss an approach to try to use the genetic adaptive controller for more sophisticated industrial applications. First, the method depends on the availability of a mathematical model (although the indirect genetic adaptive control methods do not) and in practical applications one may not be readily available. There are, however, many times when linear or nonlinear system identification can be used to produce at least a crude model of the plant's behavior. For example, consider the use of such models, even relatively complex ones with many inputs and outputs, in the "model predictive control" (MPC) method that is often used in the process control industry. Second, we need to know how far into the future to simulate the model in the assessment of fitness for each individual in the population. Again, the MPC strategy provides some insights in that it must also cope with this issue. Generally, if your model is good (and there is not too much uncertainty in the underlying process), you want to simulate out into the future further since the simulations will be accurate and hence not a waste of computations (of course, how far to simulate also depends on the complexity of the underlying dynamics). Ultimately, however, the choice of the length of the "moving window" is heuristic and requires experimentation; this should not be surprising as the choice is largely heuristic for practical applications of the MPC method. Third, you must gain insight into how to choose the parameters of the genetic algorithm. To gain this insight note that the GA is a stochastic optimization method for choosing the future control trajectory, and then (just like in the MPC method) the first element of that trajectory is used as the control input (for the individual who we believe will perform the best in the future). The parameters of the GA are simply the parameters to be tuned to perform the best optimization. Elitism makes sure that a reasonably good controller is always present. You choose the mutation and crossover probabilities so that the method is persistent is searching for new controllers (in a quickly changing environment the fitness function will change in time quickly, so you generally will need a higher mutation rate).

From this discussion, the path to use in more sophisticated applications should be clearer, at least for those that regularly use the MPC strategy. The (direct) genetic adaptive method should be viewed as an "adaptive MPC controller". Someone familiar with MPC may just view it as a standard MPC method, with a genetic algorithm used to perform the optimization that is needed to pick the control input (which, in the linear case, is often least squares). This view, is not quite accurate. The genetic adaptive controller maintains (remembers) the population of controllers and keeps these on-line for possible use in the future (e.g., if the plant returns to a similar condition). At the population level, learning occurs via the individuals gaining potential for performance improvement by interations with the plant. Finally, we would note that in an indirect genetic adaptive control method you adapt the model; hence, you obtain a type of indirect adaptive control. If you combine the GMRAC with the indirect strategy you obtain a method that tunes the model that is used to predict into the future. This type of adaptive MPC approach may also be useful for practical applications.

References

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

Gremling, J., Passino, K.M., 1997a. Genetic adaptive state estimation for a jet engine compressor. Proceedings of the IEEE International Symposium on Intelligent Control, Istanbul, Turkey, September, pp. 131–136.

Gremling, J., Passino, K.M., 1997b. Genetic adaptive failure estimation. Proceedings of the American Control Conference, Albuquerque, NM, June, pp. 908–912.

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

Krstić, M., Kanellakopoulos, I., Kototović, P., 1995. Nonlinear and Adaptive Control Design. Wiley, New York.

Layne, J.R., Passino, K.M., 1992. Fuzzy model reference learning control. IEEE Conference on Control Applications, Dayton, OH, September, pp. 686–691.

Layne, J.R., Passino, K.M., 1993. Fuzzy model reference learning control for cargo ship steering. IEEE Control Systems Magazine 13, 23–34.

Lennon, W., Passino, K., 1999a. Intelligent control for brake systems. IEEE Transactions on Control Systems Technology 7 (2), 188–202.

Lennon, W., Passino, K.M., 1999b. Genetic adaptive identification and control. Engineering Applications of Artificial Intelligence 12 (2), 185–200.

Moudgal, V.G., 1995. Fuzzy learning control for a flexible-link robot. IEEE Transactions on Fuzzy Systems 3 (2), 199–210.

Passino, K.M., Yurkovich, S., 1998. Fuzzy Control. Addison-Wesley Longman, Menlo Park, CA.

Porter, L.L., Passino, K.M., 1994. Genetic model reference adaptive control. Proceedings, IEEE International Symposium on Intelligent Control, Columbus, OH, August, pp. 219–224.

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

Porter, L.L., Passino, K., 1998. Genetic adapative and supervisory control. Journal of Intelligent Control and Systems 2 (1), 1–41.