GENETIC ALGORITHMS IN SYSTEM IDENTIFICATION

K. Kristinsson
Dept. of Electrical Eng., University of British Columbia, Vancouver, Canada

G.A. Dumont
Pulp and Paper Research Institute of Canada and Dept. of Electrical Engineering
University of British Columbia, 2385 East Mall, Vancouver, B.C., Canada V6T1W5

Abstract

Current online identification techniques are recursive and local search techniques. Here we show how genetic algorithms, a parallel, global search technique emulating natural genetic operators can be used to estimate the poles and zeros of a dynamical system. We also design an adaptive controller based on the estimates. Simulations and an experiment show the technique to be satisfactory and to provide unbiased estimates in presence of colored noise.

INTRODUCTION

On-line system identification methods used to date are based on recursive implementation of off-line methods such as least-squares, maximum-likelihood or instrumental variable. Those recursive schemes are in essence local search techniques. They go from one point in the search space to another at every sampling instant, as a new input-output pair becomes available. Genetic algorithms (GA) [1] on the other hand search many points simultaneously and thus have the potential to converge more rapidly. Genetic algorithms have also been shown to excel in multimodal optimization [2], and thus have the potential to give unbiased estimates in presence of coloured noise. This property can be used to sucessfully identify parameters of an ARMAX model.

GENETIC ALGORITHMS

A genetic algorithm (GA) differs from other search techniques by the use of concepts from natural genetics and selection. It uses simplified genetic operators and Darwinist principles such as that of survival of the fittest.

A GA works with a population of binary strings just like nature works with chromosomes. The binary strings are made from a coding of the parameters which the algorithm should find or identify. Each parameter corresponds to a fixed length binary substring of j bits $[0, \ldots, 2^j - 1]$. The value of the substring is mapped to an interval of the real numbers $[l, u]$ so the precision of the coding is $(u-l)/(2^j-1)$. With n parameters, the final string consists of n concatenated substrings.

Because the algorithm works with a population of strings, is is given more chance to locate the global optimum in a multimodal search space. The initial population is generated randomly and the population size is kept constant throughout the process. The algorithm only requires payoff information (fitness) for each of the string, without the need for assumption such as differentiability. Which makes it very useful for a discontinuous surfaces. The fitness function is in most cases the objective function that should be maximized (or minimized).

A genetic algorithm in its simplest form consists of 3 steps: reproduction, crossover and mutation. For reproduction, strings are chosen according to their normalized fitness. The fitness is normalized with the average value , so the strings with above average fitness will have more offsprings than those with below average fitness. This step directs the search towards the best. Next, new individuals have to be generated by crossover, the main searching operator. This operator takes valuable informations already in the population and combines them to find a highly fit individual. To apply this operator 2 strings are mated at random and a point on the interval $1 \leq k \leq j-1$ is chosen randomly. Two new strings are then created by changing all characters between position 1 and k inclusively. This can best be explained by example. Suppose there are two strings

00000000

11111111

and assume a random number generator comes up with a 3. Then the new strings will be

11100000

00011111

Those two above operators give genetic algorithms much of their power. The search is emphasized towards the best and new regions are explored by using informations about things that have worked well in the past.

The mutation operator, which simply flips the state of a bit, can be viewed as a background(secondary) operator to insure against loss of information in some bit positions and as a way of getting the algorithm out of a stuck state.

The theoretical properties of genetic algorithms can be studied using the theory of schemata proposed by

0-8186-2012-9/89/0000/0597$01.00 © 1989 IEEE

Holland [1]. Although transition rules are probabilistic, the algorithm is not a random search. By use of operators taken from population genetics the algorithm efficiently explores part of the search space where the probability of finding improved performance is high. GA have been shown to behave well on multimodal functions. There is however, no known necessary and sufficient condition under which a function is genetically optimizable. However, numerous studies have shown that functions on which GA fail are pathological, and generally fail to be optimized by any other known technique except exhaustive search [3]. In a recent study by Goldberg [4] it has been shown that even though the algorithm is misleaded, it will converge for a wide range of starting condition (initial population) and under unfavorable conditions.

Genetic algorithms are inherently parallel. Indeed, all strings or individuals in a population evolve simultaneously without central coordination. To realize their full potential, GA must be implemented on parallel computer architectures.

GA IN IDENTIFICATION AND CONTROL

There have been few attempts to use GAs for parameter estimation [6,7,8,9]. None of them have used the estimate to design a controller. We have designed a pole placement controller based on the estimates GA gives.

Identification

Consider the system

$$A(q^{-1})y(t) = B(q^{-1})u(t) + C(q^{-1})e(t) \qquad (1)$$

where the noise $e(t)$ is normally distributed sequence with zero mean and unit variance, q is the forward shift operator, i.e. $y(t+1) = qy(t)$. Our objective is to identify $A(q^{-1})$ and $B(q^{-1})$, when the input $u(t)$ is a PRBS. One can define two sequences $\varepsilon(t)$ and $\eta(t)$ as:

$$\hat{A}(q^{-1})y(t) = \hat{B}(q^{-1})u(t) + \varepsilon(t) \qquad (2)$$

$$\begin{aligned} \hat{A}(q^{-1})\hat{y}(t) &= \hat{B}(q^{-1})u(t) \\ \eta(t) &= y(t) - \hat{y}(t) \end{aligned} \qquad (3)$$

Then we try to minimize $E[\varepsilon^2(t)]$ or $E[\eta^2(t)]$. The first case corresponds to the least-squares case, the second is akin to the instrumental variable case.

As an example, consider the system [5]:

$$\begin{aligned} A(q^{-1}) &= 1.0 - 1.5q^{-1} + 0.7q^{-2} \\ B(q^{-1}) &= 1.0q^{-1} + 0.5q^{-2} \\ C(q^{-1}) &= 1.0 - 1.0q^{-1} + 0.2q^{-2} \end{aligned} \qquad (4)$$

In pole-zero form, the plant can be written as:

$$\begin{aligned} A(q^{-1}) &= 1 - (0.75 \pm j0.37)q^{-1} \\ B(q^{-1}) &= 1.0q^{-1}(1.0 + 0.5q^{-1}) \end{aligned} \qquad (5)$$

Because it does not require linearity in the parameters, genetic algorithm can directly identify the poles and zeros of the system. Poles of a second order system will always be of the form

$$p_{1,2} = \alpha \pm \beta \qquad (6)$$

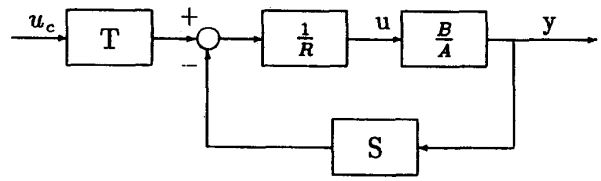


Figure 1: Two-degree of freedom controller

where β is either imaginary or real, i.e. complex conjugate poles or 2 real poles. Because the sign on β is of no importance we can use the sign to deside if the number is imaginary or real. The poles of our system would then be represented by

$$p_{1,2} = [0.75, -0.37] \quad \text{or} \quad p_{1,2} = 0.75 \pm 0.37j \qquad (7)$$

We need then to identify 4 parameters, the gain b_1, the zero z_1 and the pair α, β for the complex poles. Upper bound and lower bound are then defined for the parameters and they coded as 11 bit strings making totally 44 bits string. The population size is set to 100 and probabilities of crossover (p_c) and mutation (p_m) to 0.8 and 0.01 respectively. Depending on the method used, the fitness function is chosen as

$$F(t) = \sum_{i=0}^{w} M - (\varepsilon(t-i))^2 \qquad (8)$$

or as

$$F(t) = \sum_{i=0}^{w} M - (\eta(t-i))^2 \qquad (9)$$

where w is the window size or number of timesteps the fitness is accumulated, and M is a bias term needed to ensure a positive fitness. Rapid convergence is detected by monitoring how many individuals receive no offspring. Whenever the algorithm detects rapid convergence it switches to selection based upon ranking [10].

Control

Consider the two-degree of freedom pole-placement controller of Figure 1. The control law is

$$R(q)u(k) = T(q)u_c(k) - S(q)y(k) \qquad (10)$$

The closed-loop transfer function is

$$\frac{BT}{AR + BS} \qquad (11)$$

The desired closed-loop transfer function is

$$H_m = \frac{B_m(q)}{a_m(q)} \qquad (12)$$

The controller is designed by solving the Diophantine equation for R_1 and S

$$AR_1 + S = A_0 A_m \tag{13}$$

where A_0 is the observer polynomial [11]. The controller is then given by

$$R = BR_1 \qquad T = B_m A_0 \tag{14}$$

Note that this scheme cancels all zeros. In practice, the Diophantine equation would be slightly different in order not to cancel unstable or ringing zeros. When the true plant is not known, one can use the GA to design an indirect adaptive control scheme as shown of Figure 2.

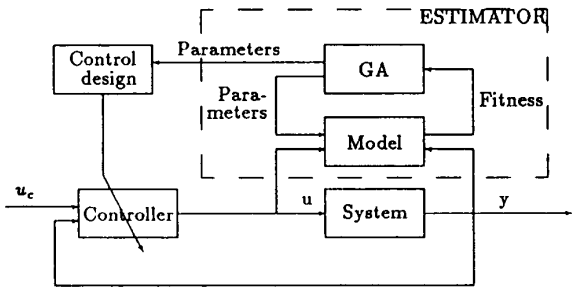


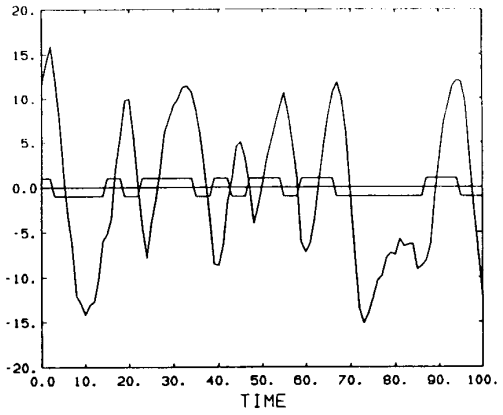
Figure 2: GA adaptive controller

SIMULATIONS

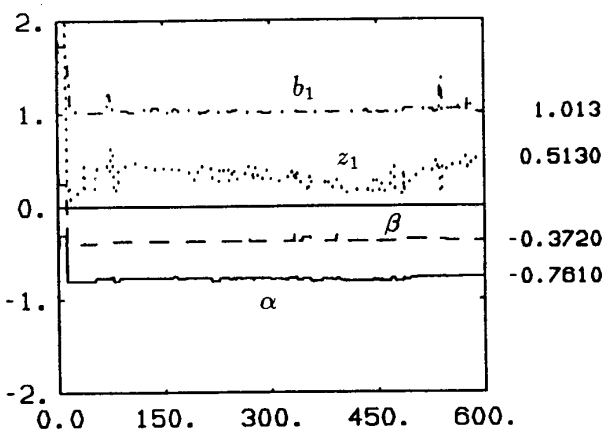
We have used the system (4) from [5] and the fitness has been calculated by (9). For the open-loop identification runs, a PRBS was sent to the plant input. Figure 3 shows results from single run using a fitness given by equation 9. At every sampling interval, the population goes through six generations. We see that the poles and zero are identified withing 150 generations or so, i.e. in 25 sampling intervals, even though colored noise is added to the output. Figure 4 shows the behavior of the RLS scheme under the same conditions. Note that the RLS gives biased estimates as expected and does not seem to converge as fast as the GA. Figures 5 and 6 show the behavior of the GA and the RLS adaptive pole-placement controllers respectively. In both cases, we chose deadbeat control and a deadbeat observer. GA seems to be doing as well as RLS for our pole-placement controller. There are however few spikes in our estimate using GA and subsequently in the output. GA converges faster than RLS in terms of number of input-output data points and it gives unbiased estimates. RLS gives the estimate of a_1 as -1.391 where GA gives -1.522.

EXPERIMENTS

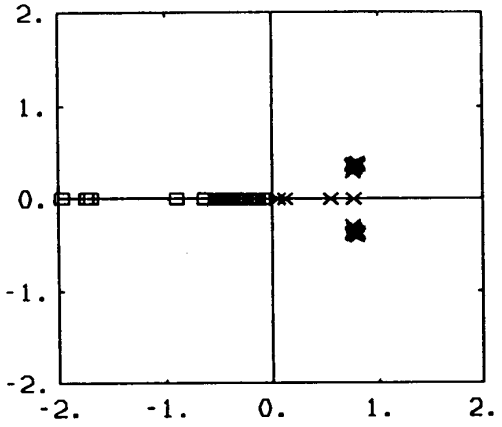
We have tried the algorithm on a real plant, a tank system that has controllable inflow and measureable wa-



(a) Input and output.

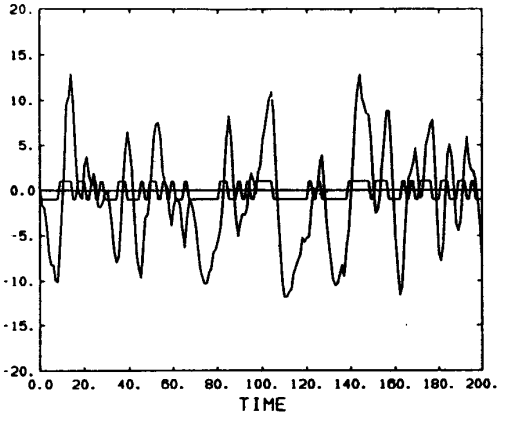


(b) Parameter estimates

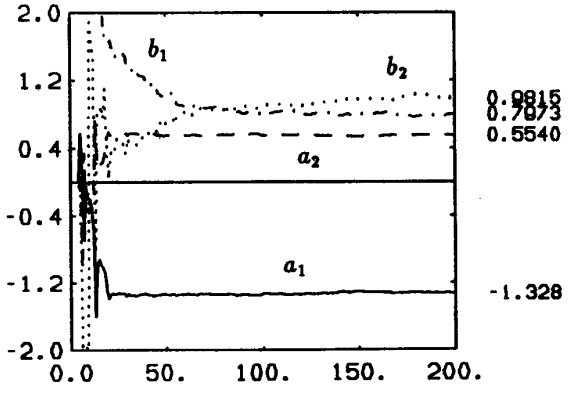


(c) Evolution of pole-zero estimates

Figure 3: GA open-loop identification

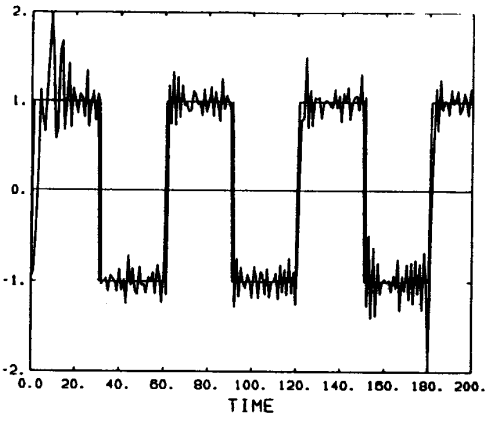


(a) Input and output.

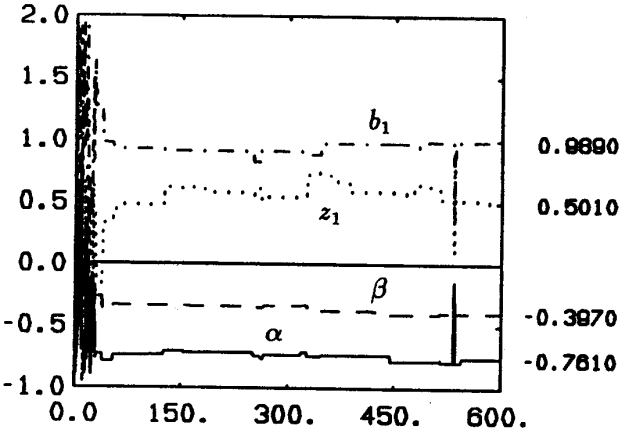


(b) Parameter estimates

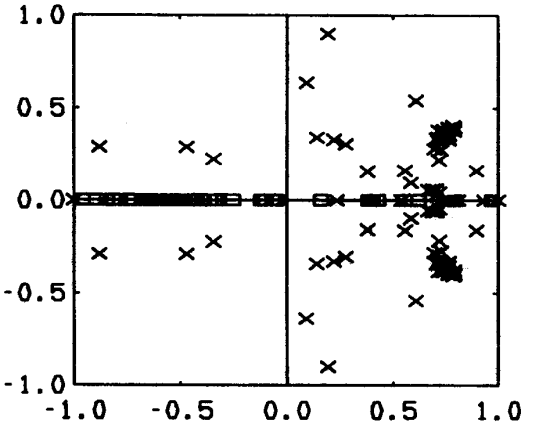
Figure 4: RLS open-loop identification



(a) Input and output.

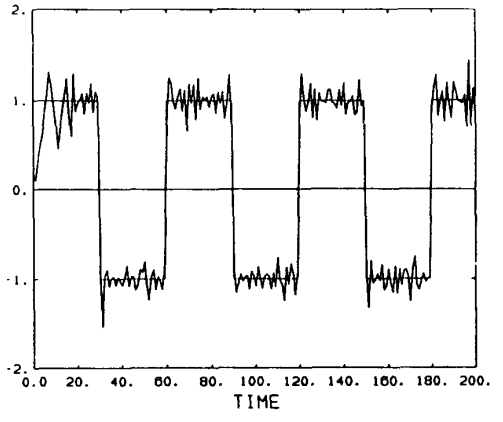


(b) Parameter estimates

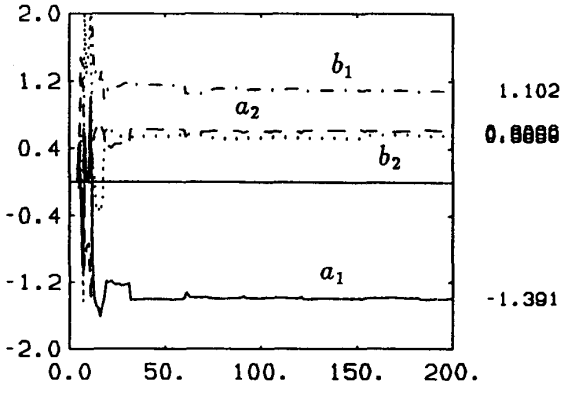


(c) Evolution of pole-zero estimates

Figure 5: GA adaptive pole-placement control

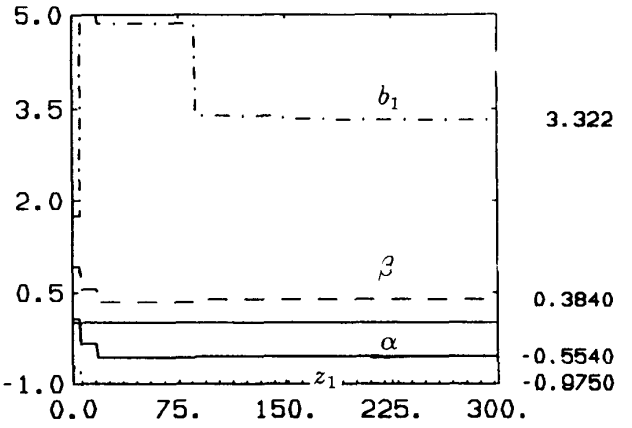


(a) Input and output.

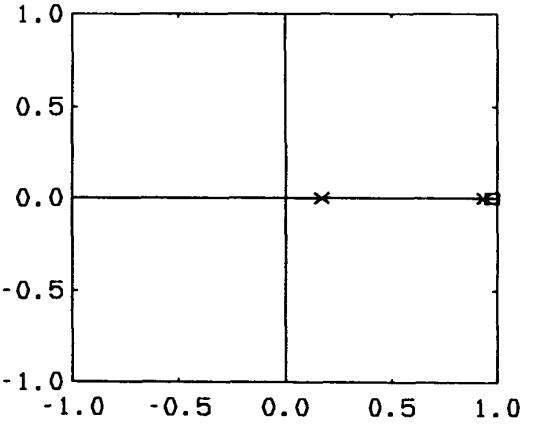


(b) Parameter estimates

Figure 6: RLS adaptive pole-placement control



(a) Parameter estimates



(b) Pole-zero estimates for generations 200 and 300

Figure 7: Tank open-loop identification

ter height. The transfer function of the tank is

$$W_1(s) = \frac{K_1 T_1}{T_1 s + 1} \tag{15}$$

or in discrete time

$$H(z) = \frac{K_1 T_1 (1 - e^{-\frac{T}{T_1}}) z^{-1} (1 - z^{-1})}{(1 - z^{-1})(1 - e^{-\frac{T}{T_1}} z^{-1})} \tag{16}$$

The dynamics are nonlinear because the outflow is a fuction of the tank level and because of the nonlinear characteristic of the pump. The GA assumed that there were 2 poles and one zero, but was able to find the integrator and cancel it with a zero, see Figure 7. The GA was run with the usual mutation and crossover values. Parameter range was $[-1, 1]$, except for b_1, for which it was $[0, 10]$. Population size3 was 100, and 6 generations were generated each sampling interval. Because of the prohibitive time it takes to run the GA on an IBM AT we were not able to do any online control on the tank.

CONCLUSIONS

We have shown that GA can successfully be used to estimate the parameters of a dynamical system. Because GA do not require linearity in the parameters, direct estimation of poles and zeros is feasible. The best estimate in the population can then be used to design an adaptive pole-placement as demonstrated by simulations. Because they are local search algorithms, GA are less likely to provide biased estimates. An area for further research is the exploitation of more than the best string in the population for the design of a robust controller. This is particularly attractive when estimating the continuous time system parameters. We are researching this topic. GAs are parallel algorithm, so every attempt to run the algorithm on non-parallel computer is bound to be slow. For our case the algorithm uses 2 seconds of CPU time for each generation, on a μ-VAX (1 MIPS) for population size of 100 and stringlength of 44. Once parallel computer architecture becomes readily available, GA will become very attractive.

References

[1] Holland, J.H., *Adaptation in Natural and Artificial Systems.* University of Michigan Press, Ann Arbor, 1975

[2] DeJong, K.A., *An analysis of the behavior of a class of genetic adaptive systems.* Doctoral dissertation, University of Michigan, 1975.

[3] Bethke, A.D., *Genetic algorithms as function optimizers.* Doctoral dissertation, University of Michigan, 1980.

[4] Goldberg, D.E., Simple genetic algorithms and the minimal deceptive problem , *Genetic algorithms and simulated annealing*, Lawrence, D. (Ed.), Pitman Publishing, pp.74-88, 1987.

[5] Söderström, T., Ljung, L. and Gustavsson, I., *A comparative study of recursive identification methods*, Report 7427, Lund Institute of Technology, Lund, Sweden, 1974.

[6] Das,R. and Goldberg, D.E., Discrete-Time parameter estimation with Genetic algorithms, *Proceedings of the* 19^{th} *annual Pittsburgh conference on modelling and simulation*, 1988.

[7] Etter,D.M., Hicks,M.J. & Cho,K.H., Recursive adaptive filter design using an adaptive genetic algorithm. *Proceedings of the IEEE International conference on Acoustics, Speech and Signal Processing*, pp.635-638,2,1982.

[8] Smith,T. and DeJong,K.A., Genetic Algorithms applied to the calibration of information driven models of US migration patterns, *Proceedings of the* 12^{th} *Annual Pittsburgh conference on Modelling and Simulation*, pp.955-959, 1981.

[9] Goldberg, D.E., *System identification via genetic algorithm*, Unpublished manuscript, University of Michigan, Ann Arbor, MI, 1981.

[10] Baker,J.E., Adaptive Selection Methods for Genetic Algorithms, *Proceedings of an International conference on Genetic Algorithms and Their Aplications*, pp.101-111,1985.

[11] Åström, K.J. and Wittenmark, B., *Computer controlled systems*, Prentice Hall Inc., Englewood Cliffs N.J., 1984.