ON-LINE TUNING OF AMB CONTROLLERS USING GENETIC ALGORITHMS

Lichuan Li[1]

ABSTRACT

A preliminary study of controller tuning conducted by genetic algorithms (GAs) is presented. On-line tuning problem is formulated and a specific GA for the purpose is put forward. Experiment results on an active magnetic bearing (AMB) test rig are given to show the effectiveness of this method.

1. INTRODUCTION

In control engineering the final step of the task is usually to implement the controller that has been designed for a given plant. It is almost always the case that the plant is not exactly known, or even with a known plant the controller is usually designed based on a simplified model, where nonlinearity is neglected and high frequency dynamics is not considered. Therefore, no one can be certain that the closed-loop system will work as desired when the controller is correctly constructed, and on-line tuning is almost always a must. Even if the controller is designed based on an accurately identified model, the designer can hardly say the tuning is not necessary. As this manual tuning may need much effort and be time consuming, it is desirable that controller tuning process be conducted automatically by some search algorithm.

GAs are a category of search algorithms imitating the mechanics of natural evolution. It is characterized by being able to locate the global optimal solution, requiring no derivative information of the objective function and its less sensitivity to the dimension of the problem compared with other search algorithms (Michalewicz, Janikow and Krawczyk, 1992; Michalewicz, 1996). Besides, GAs are robust, that is, a satisfactory solution, if not the best, can always be found by GAs. This is indeed an important property in searching a controller. In fact, GAs have been successfully applied to controller design and optimization tasks (Varsek, Urbancic and Filipic, 1993; Michalewicz, Janikow and Krawczyk, 1992). However, to the best knowledge of the author, no on-line work has been reported.

[1]Department of Mechanical Engineering, Xi'an Jiaotong University, Xi'an, Shaanxi 710049, P.R. of China, e-mail: lcli@xjtu.edu.cn.

The main idea of this study is to use GAs as a searching machine to conduct the on-line tuning work automatically. A group of candidate controllers are made to evolve from generation to generation, matching the given but not clearly known plant more and more, and finally meet the prespecified closed-loop performance indices as much as possible. This way, an accurate model of the plant is not needed, that is, accurate identification is not needed, and the tuning task is made easier, especially on the occasion where quite a few interdependent parameters are to be adjusted.

2. DEFINITION OF THE PROBLEM

It is assumed that the controller structure is known. The task is only to determine the values of the controller parameters such that a given performance index (PI) which is a function of the parameters is minimized. A simple case is to search for the parameters of a linear controller so as to achieve the prespecified closed-loop performances. From the view point of optimization, sensitivity function or singular value may be used as an objective function. But in real time implementation this will give rise to a great deal of computation, hence the time requirement may become unreasonable, as for evaluating each candidate solution a frequency response measurement will have to be executed and in GAs many times of evaluation are needed. Thus a time domain PI is preferable, such as I(T)AE or I(T)SE. However, with such a PI the stability of the closed-loop system may become sensitive to parameter drift, because the resultant controller may have parameters lying close to the boundary of the region where the closed-loop system is stable. Here in this study, instead of finding some optimal controller, we take a moderate objective of finding controller parameters that yield given time domain specifications such as rise time and overshoot to a step input.

Consider a linear SISO system shown in Fig. 1, where $G_p(s)$ and $G_c(s)$ are transfer functions

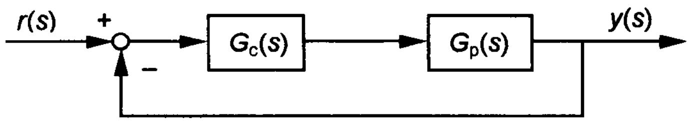

Figure 1. The closed-loop system.

of a given plant and a controller respectively. For each controller parameter $p=(p_1, ..., p_n)^T$ which contains n independent parameters, by applying a step input at $r(t)$ an output $y(t)$ can be measured and a number of time domain performances denoted by $z=(z_1, ..., z_l)^T$ can be computed with the measured data. In order to find the controller parameter that yields specified performances $z_0=(z_{01}, ..., z_{0l})^T$, an objective function

$$f(p) = (z - z_0)^T Q (z - z_0) \qquad (1)$$

is maintained, where Q is a positive definite weighting matrix. Now the problem is defined as to find a p^* that minimizes the objective function (1). When the performances are properly chosen that minimization of the objective function implies closed-loop stability, a constraint on the search region is usually not necessary. However, if *a priori* knowledge enables a constraint on p to be specified, search may be made easier.

3. THE GENETIC ALGORITHM

While in most iterative algorithms only one candidate solution is updated at each iteration, a group of candidate solutions are handled simultaneously in GAs. Each solution is called an individual and the group of the individuals is called a population. Each individual is assigned a positive fitness value evaluated by an objective function, representing its relative goodness. At the beginning, an initial population $P(0)$ is maintained. Then it is updated iteratively by applying genetic operators, yielding a population sequence $P(k)$, k=0, 1, ..., where each one is called a generation. The number of individuals in each population is usually kept constant and called population size. This iterative process proceeds until some termination condition is met and the best individual in the final population is taken as the optimal solution.

Apart from handling a group of solutions instead of a single one, GAs are more characterized by the genetic operators that map $P(k)$ to $P(k+1)$ with randomness. For details the reader is referred to (Michalewicz, 1996; Bäck, 1996). Below a specific GA for minimization which differs slightly from those widely adopted ones is given. It is used later for controller tuning.

In the modified GA an individual p is a real vector $p=(p_1, \ldots, p_n)^T$, corresponding to n controller parameters to be tuned. A population $P(k)=\{p_1, \ldots, p_m\}$ is a set of m such individuals. Before the algorithm is given, let the basic operations be defined. In the following, random numbers denoted by r are uniformly distributed in [0, 1] and those denoted by $s(x)$ are 1 with probability x and 0 with probability $1-x$. All the random numbers are independently generated.

Initiation: Create an initial population $P(0)$.

Evaluation: Evaluate the fitness value of each individual in the current population by the objective function (1). It also gives the least fitness value $f_{\min}$ among the individuals.

Elimination: The individuals in the current population is ranked in such an order that their fitness values increase monotonically with index. Then generate m random numbers $r_1, \ldots, r_m$, and check whether $r_i<(i-1)/(m-1)$ is true for $i=1, \ldots, m$. If in i-th check the result is true, the i-th individual is eliminated from the current population. After elimination the population has only m_a alive individuals left, and $1\leq m_a\leq m$.

Reproduction: This is to fill up the population that has undergone elimination by reproducing $m-m_a$ new individuals from the m_a individuals still alive in $P(k)$. To reproduce one new individual, two individuals, denoted by $p_u=(a_1, \ldots, a_n)$ and $p_v=(b_1, \ldots, b_n)$, are chosen from the m_a alive ones in $P(k)$ randomly. Then generate n random numbers $s_1(p_c), \ldots, s_n(p_c)$, and make a new individual $p_w=(c_1, \ldots, c_n)$, where $c_j=s_ja_j+(1-s_j)b_j$, $j=1, \ldots, n$, and p_c is the cross rate. Finally, generate $2n$ random numbers $s_1(p_m), \ldots, s_n(p_m)$, and $r_1, \ldots, r_n$, and multiply c_j with

$s_j[1+h(2r_j-1)]+(1-s_j)$, where p_m is the mutation rate and h is a positive constant real number for controlling the intensity of mutation, referred to as mutation intensity. This process for reproducing one new individual is executed m-m_a times independently, and then these newly created individuals are placed back in $\boldsymbol{P}(k)$.

Now the algorithm is given in Fig. 2, where f_0 is the acceptable fitness value and N is the allowable maximum generation in case that the GA never terminates. In the elimination, the best individual will have an index of 1 after ranking, so it will not be eliminated, fulfilling naturally the elitist selection (Michalewicz, 1996). To adopt such an elimination procedure instead of the usual selection has another advantage, that is, the probability of the occurrence of two or more identical individuals in a population is very small, which is helpful to maintain the diversity. After the elimination, averagely $m/2$ individuals are missing. This is seen by

```
        give the values of $m$, $p_c$, $p_m$, $f_0$ and $N$
        $k$=0;
        initiation;
        evaluation;
check:  if $f_{min}<f_0$ or $k=N$, then exit;
        $k \leftarrow k+1$;
        elimination;
        reproduction;
        evaluation;
        go to check;
```

Figure 2. The modified genetic algorithm.

$$0/(m-1) + 1/(m-1) + \cdots + (m-1)/(m-1) = m/2 \tag{2}$$

where the i-th item on the left-hand-side is the probability that the i-th individual be eliminated. The reproduction combines the operations of crossover and mutation. The intensity of the crossover is controlled by p_c. Note that when p_c=0.5 the crossover is of most intensity. The mutation, being multiple instead of additive, eliminates intensity differences when the values of the elements of $\boldsymbol{p}$ are of different orders. This multiple mutation also confines the elements within a single sign, which is usually the case for controller parameters. Finally, note that in evaluation the individuals come directly from the previous generation, i.e., the m_a alive ones after elimination, need not be evaluated.

As GAs are problem dependent, there are many variations based on the standard frame in the literature. The GA given above is one of many possible ones that can be used for on-line controller tuning. However, comparisons with other ones has not been further addressed.

4. EXPERIMENT

The experiment is done on the axial channel of a five-DOM AMB test rig introduced in (Li, 1997). The actuator has a current control configuration. So its linearized model is

$$G_p(s) = \frac{y(s)}{i(s)} = \frac{b_0}{s^2 - a_0} \tag{3}$$

where $y(s)$ is the displacement, $i(s)$ is the current input and a_0, b_0 are positive constants. For this plant a phase-lead controller

$$G_c(s) = \frac{B_1 s + B_0}{s + A_0} \tag{4}$$

is used, where B_1, B_0 and A_0 are three parameters to be tuned. For the performances of the closed-loop system, desired rise time z_{01} and overshoot z_{02} to a step reference input are specified. So the objective function to be minimized is

$$f(p) = q_{11}(z_1 - z_{01})^2 + q_{22}(z_2 - z_{02})^2 \tag{5}$$

where z_1 and z_2 are measured rise time and overshoot respectively, $\boldsymbol{p}=(B_1, B_0, A_0)$ is the parameter vector, and q_{11}, q_{22} are positive constants. To obtain measured rise time and overshoot resulted from the controller with given parameters, the controller is converted into a discrete one and loaded to the DSP based AMB control system, where a step reference input is applied and the output is sampled at 128 consecutive sampling instants with 2048 corresponding to 0.25 mm of displacement. The sampling interval is 200 μs. In handling the rise time and overshoot the integer version of the sampled data is used and the unit of time is the sampling interval. The rotor is positioned with a reference of -250 and then a step input from -250 to 250 is applied in order to use the region where the plant is more linear. Suppose the initial, stead-state, and first peak output are y_{00}, y_{ss} and y_p respectively, the rise time is defined as

$$z_1 = t_{90} - t_{10} \tag{6}$$

where t_{10} is the instant that $y(t)= y_{00}+0.1(y_{ss}-y_{00})$ for the first time and t_{90} is the instant that $y(t)=y_{00}+0.9(y_{ss}-y_{00})$ for the first time. As the data are sampled at discrete instants linear interpolation is used when necessary. The overshoot is defined as

$$z_2 = (y_p - y_{ss})/(y_{ss} - y_{00}) \tag{7}$$

At the beginning, some initial values of the controller parameters which provide closed-loop stabilization are known. They are obtained with the automatic start-up method (Fritsche and Arnold, 1995; Li, 1997). However, with the method the performances are not easy to be satisfactory while stabilization is usually possible. The initial parameter is $\boldsymbol{p}_0$=(6825, 2573500, 2100), which leads to a rise time of about 12 and an overshoot of about 0.16. The desired values are chosen as z_{01}=8 and z_{02}=0.1, and the weighting factors are chosen as q_{11}=1 and q_{22}=10000 in order to make the two indices comparable.

The GA parameter settings are m=10, p_c=0.5, p_m=0.25, h=0.02, f_0=0.5 and N=100. In the initiation each individual in $\boldsymbol{P}(0)$ is obtained by multiplying the three elements of $\boldsymbol{p}_0$ with $1+0.05(2r_j-1)$, where r_j (j=1,2,3) are random numbers. Run the GA with the given parameters,

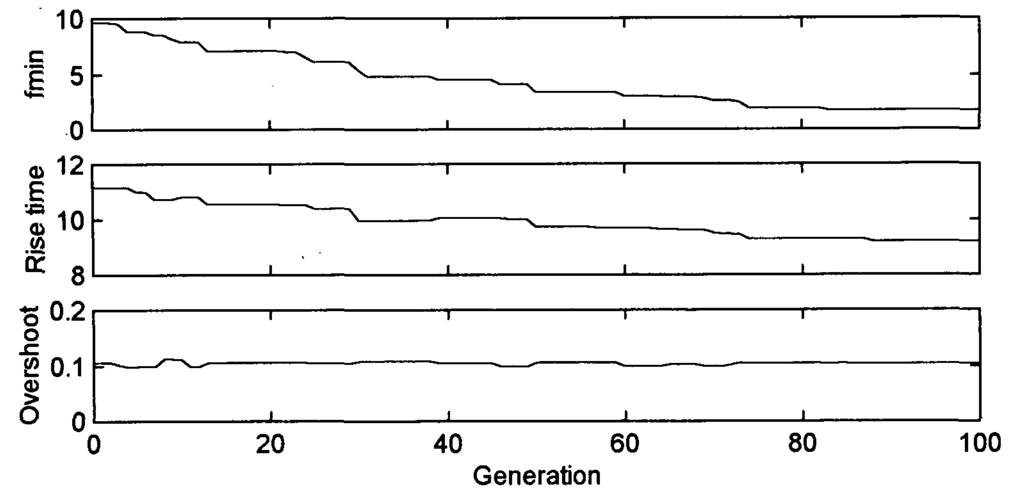

Figure 3. Experiment results of the first run.

it terminates by reaching the maximum generation. Fig. 3 shows the fitness value, rise time and overshoot of the best individuals in each generation. That in the final generation is p^*=(7781, 2474200, 1928) with f_{min}=1.782, and the corresponding rise time and overshoot are 9.310 and 0.103 respectively. It is observed that while the fitness goes down steadily, the overshoot fluctuates about the desired value and the rise time is always greater than desired. This may be explained as the overshoot is sensitive to B_1 only, but more than one parameter must change simultaneously in the correct direction in order to reduce the rise time without affecting the overshoot too much. This problem may be relieved by choosing a smaller weight for the overshoot term in (5) or other forms of controller parameterization.

The experiment is done again with q_{22}=1000, ten times smaller than before. To make the termination condition comparable f_0 is reduced to be 0.25. Fig. 4 shows the results. It is seen

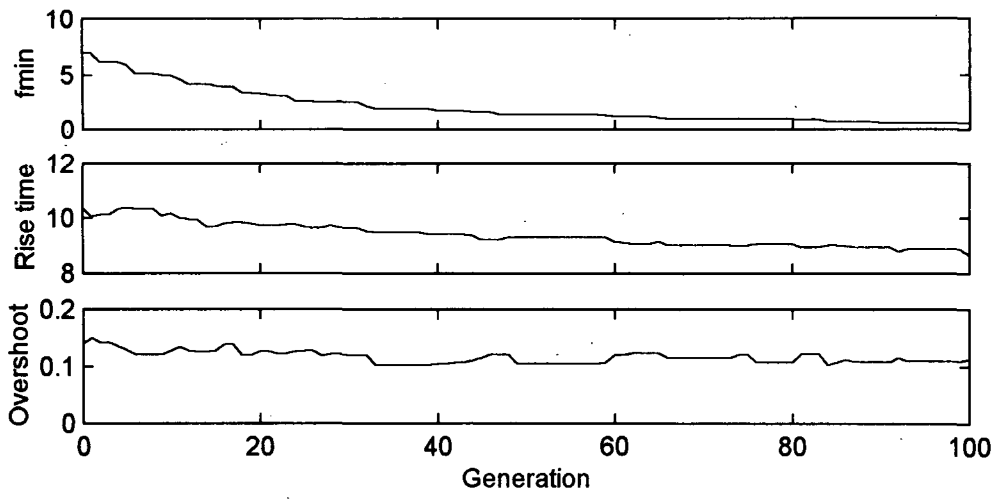

Figure 4. Experiment results of the second run.

that this time the rise time fluctuates more and the overshoot has an obvious decline tendency. The best individual in the final population turns out to be $\boldsymbol{p}^*$=(8373, 2634600, 1924) with a fitness of 0.591, and the associated rise time and overshoot are 8.665 and 0.112 respectively, being a little better than the previous. However, the iteration still terminates with the maximum generation being reached. The experiment is done for a third time with another form of controller

$$G_c(s) = \frac{K(T_1 s + 1)}{T_2 s + 1} \tag{8}$$

where the gain K and time constants T_1 and T_2 are addressed directly. In terms of (4), when K rises both B_1 and B_0 will rise, which gives rise to a shorter rise time while the overshoot is not affected as much as before. Besides, T_1 has strong effect on overshoot but little on rise time. By the decoupling effect of (8) better results can be expected. The results are shown in Fig. 5. The

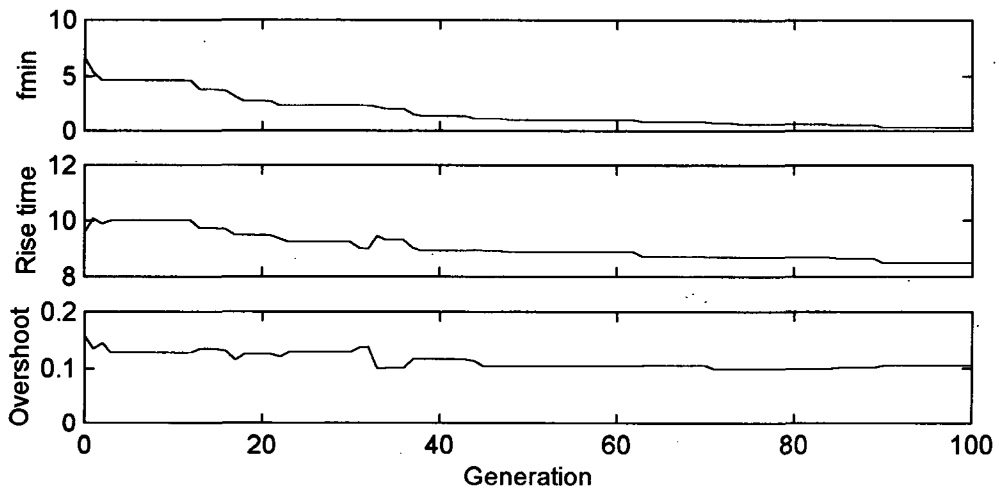

Figure 5. Experiment results of the third run.

GA terminates again with the maximum generation being reached, but the best fitness declines to 0.272, being close enough to f_0=0.25. So this time the result is accepted. The best individual within the final population is $\boldsymbol{p}^*$=(K, T_1, T_2)=(1442, 0.003016, 0.0004705). Translated into the original form it is $\boldsymbol{p}^*$=(B_1, B_0, A_0)=(9375, 3065700, 2126). The associated rise time and overshoot are 8.491 and 0.106 respectively, being close enough to the specified values.

All the three runs have iterated 100 generations. As in the elimination averagely half the individuals are eliminated, the total number of evaluations including those for the individuals in $P(0)$ is about 10+100×(10/2)=510. The time for a run is a little more than the time for evaluation which is about 510×128×200μs≈13s. It is mentioned that GAs are intrinsically parallel (Michalewicz, 1996). But with one physical system the individuals must be evaluated one by one, which means the parallelism no longer exists.

5. CONCLUSIONS

Apart from manual and other tuning methods, it is shown that GA-based tuning may be an additional choice for on-line tuning. The concepts involved are simple, the implementation is easy and the time required is acceptable. Besides, minimization of the objective function provides an 'instant' controller, because the performances come from the physical system.

However, for practical applications some problems must be considered. If the controller is designed based on an accurately identified plant model, every aspect of the performance can be taken into account by the designer. With GA, however, the only access is an objective function, and it is not easy for the objective function to account for all the intentions of the designer, especially when the plant is not well known. So, to make an effective objective function with the available knowledge about the plant is of essential importance.

The minimization problem is in fact of multiple objective. In the experiment desired rise time and overshoot are specified. It is observed that when knowledge about the relationships between the parameters and outcomes are exploited, better results can be achieved. In other words, the effectiveness of the GA relies on the parameterization of the controller. In further research, a learning mechanism that enables GA to learn the effects of the controller parameters on the performances may be considered. With the acquired knowledge the totally random mutation can be biased towards favorable directions so as to reduce the dependence of the effectiveness on controller parameterization.

ACKNOWLEDGEMENT

This topic of research is supported by the Natural Science Foundation of China. The author is also thankful to the International Center of Magnetic Bearings at Swiss Federal Institute of Technology for providing research facilities and the AMB test rig.

REFERENCES

Bäck, T. 1996. *Evolutionary Algorithms in Theory and Practice*, Oxford University Press, New York.

Fritsche, C. and D. Arnold. 1995. "Rotor schweben lassen ohne Modellkenntnis," Semesterarbeit, Institut für Robotik, ETH Zürich.

Li, L. 1997. "Automatic Startup of an AMB System Using Parameter Identification," Internal Report, ICMB, ETH Zurich.

Michalewicz, Z. 1996. *Genetic Algorithms + Data Structures = Evolution Programs*, Springer-Verlag, Berlin.

Michalewicz, Z., C. Z. Janikow and J. B. Krawczyk. 1992. "A Modified Genetic Algorithm for Optimal Control Problems," *Computers Math. Applic.*, 23(12)12:83-94.

Varsek, A., T. Urbancic and B. Filipic. 1993. "Genetic Algorithms in Controller Design and Tuning," *IEEE Trans. Syst., Man and Cybern.*, 23(5):1330-1338.