Servo parameter tuning for a 5-axis machine center based upon GA rules

Lun-Yu Kuo a, Jia-Yush Yen b, *

a Mechanical Engineering Research Laboratory, Industrial Technology Research Center, Taichung, Taiwan, ROC
b Department of Mechanical Engineering, National Taiwan University, No. 1 Sec. 4 Roosevelt Road, Taipei, Taiwan 10617, ROC

Received 14 August 2000; received in revised form 6 December 2000; accepted 21 February 2001

Abstract

This paper addresses the use of the genetic algorithm (GA) to tune the servo controller in a multi-axis CNC machine center. Because the number of parameters in a CNC controller is usually very large and their effects are highly coupled, tuning these parameters is a task requiring great skill. In this paper, the genetic algorithm is applied to tune the servo parameters in a commercial CNC machine controller. The controller is a PC-based open architecture system, thus allowing the GA to be implemented on a personal computer. In this study, a dynamic mutation amplitude process is also introduced to improve the quality of evolution. Experimental results obtained on a Siemens 840D CNC controller indicate that the proposed method is able to automatically identify the effect of each parameter and to tune the controller so as to achieve expert level performance. © 2001 Elsevier Science Ltd. All rights reserved.

Keywords: Servosystems; Motion control; Machine tool control

1. Introduction

The motion controller in a multi-axis CNC machine center is usually a very complicated system. Unlike the conventional servo controller that is composed of only a single (may be very sophisticated) control law, CNC controllers usually consist of several control blocks that work together to satisfy the various machining requirements.

The servo design philosophy in a CNC controller differs from the traditional control approach because there is almost no prior knowledge of the working plant. The servo engineers must
understand that the loading and the cutting conditions in a machine tool can vary from case to case; therefore a controller tuned with constant plant assumption would not maintain satisfactory performance. To achieve consistent performance, the core motion controller has to sacrifice a great deal of performance to maintain the required system robustness. The various control function blocks then help compensate for the varying path changes, friction, etc. These functions include dynamic response matching, which makes the impedance of all axes identical, speed feedforward control, which controls the time constant of the speed control loop, current feedforward control, which compensates for the motor and load inertia, and the friction compensation, which is useful when the machine encounters errors caused by friction force. Usually, various parameters are available in a CNC machine controller to adjust the various functions. Unfortunately, there is no obvious correlation between the parameters and the servo performance, and their effects are often heavily coupled. Adjusting these parameters requires much experience and patience. In most cases, large controller manufacturers like FANUC or Siemens have to send their engineers out to help customers tune their controllers.

The goal of this research is to make the servo engineer’s task easier by developing an automated tuning method for these parameters. Because a CNC controller is a very complicated system, most previous works have focused on a subset (or a specific function) of a controller [1–7]. The few papers that have addressed the entire system have only described the system at the functional level [8]. Due to developments in machine intelligence, many recent results have started to incorporate machine intelligent into servo systems [9–15]. Basically, these researches have offered new ways to tune controllers with fixed structures or to generate control rules. There are also reports on many successful applications [16–19]. However, they are also restricted to single function controllers. The work by Tarng, Chuang, and Hsu [19] addressed the CNC controller, but they also focused on the federate control function.

This work will address all the most useful control functions in a CNC controller in order to minimize the need for experts. Note that the number of parameters in this kind of controller is very large, and that the effects of these parameters are coupled. There is really no apparent relationship between the parameters and the control outcome. Thus, it is very difficult to establish sensible rule base for tuning these parameters. It is perhaps a very suitable case for applying genetic algorithms (GA). In this research, a PC-based Open CNC controller based on the Siemens online tuning procedure. A GA algorithm is in charge of tuning these parameters. Through the evolution of genes (where each gene represents a parameter), the controller gradually adapts itself so as to achieve better performance. To improve on the quality of convergence, a dynamic mutation amplitude method is also introduced. Experimental results indicate that the GA algorithm is able to achieve expert performance. The control results are superior to results obtained through tuning by a trained expert.

In the following sections, Section 2 presents a brief description of the functions and the associated parameters of a multi-axis CNC controller. This section will also include some expert opinions on how to tune these parameters. Section 3 presents the experimental CNC machine center as well as the actual implementation in detail. Section 4 presents various control experiment situations, experimental results and some discussion of the data will be presented. Section 5 presents conclusions.
2. The multi-axis CNC machine tool controller

The system setup was designed to facilitate research on the motion control performance of a 5-axis machining tool. The 5-axis machining tool was developed in the “High Speed Machining of Complex Surfaces” project sponsored by the Ministry of Economic Affairs of the ROC government. The setup utilizes a home grown PC-based controller. The controller, is based on use of Siemens 840D control kernel for the machine tool controller. The interpolation time for this controller is 2 ms. Thanks to the CNC controller’s open architecture, it is possible to bypass the interface adaptation problem and to integrate in-house control modules into the system. Fig. 1 shows the block diagram of the position loop control.

In the following subsections the effects of some of the most important motion control functions will be explained, and a set of associated parameters will be defined.

2.1. Dynamic response matching

The goal of this parameter is to align the impedance of all the servo axes. Although it is possible to tune separate axis loop gains so as to make the errors from each axis similar, there are times when the performance of some axes might deteriorate. Long rise time or large overshoot may occur. In addition, a high servo gain will lead to better disturbance suppression; one should not sacrifice good servo gain in order to match axes having inferior performance. For these reasons, the system uses dynamic response matching to synchronize axis performance and to reduce the resulting errors. In the course of developing the algorithm and programming the software, we define a dynamic response matching parameter called DYNAMIC.MATCHING.TIME_CONSTANT (DMTC), and this term will be used in the following description. To set this parameter, the axis with the lowest gain is selected as the base axis. If the time constant of some axis is 3 ms faster than the base axis, then the parameter of this axis is set to 0.003 s.

2.2. Speed feedforward control

The speed feedforward control gain, as the name indicates, adjusts the feedforward controller gain. We define a speed feedforward control gain parameter called SPEED.FEEDFORWARD_GAIN (SFG). A parameter called SPEED.CONTROL.TIME_CONSTANT (SCTC) is also introduced to represent the equivalent time constant of the speed control loop. The equivalent time constant of the speed control loop is generally set to 1/10th time constant of the position control loop. The time constant normally ranges from 1 to 5 ms. If the time constant of the speed control loop is set correctly, a feedforward control factor with a value of 1 can produce a speed loop error that is close to zero. Of course, there is always some error in evaluation, and further

---

Fig. 1. CNC controller block diagram.
fine-tuning is needed. Generally, larger values of these parameters lead to stronger feedforward effects.

2.3. Current feedforward control

The parameter that represents the axis inertia is called AXIS_INERTIA (AI), and it influences the effect of the current loop feedforward gain. The current loop mainly works to eliminate the inertia effect. The total axis inertia includes the inertia of the motor and that of the load. Another parameter that controls the time constant of the current loop is called EQUIVELENT_CURRENT_CONTROL_TIME_CONSTANT (EQCCTC). The value of this parameter can be obtained from the step response of the current loop. Setting a large value for AI larger and a small value for EQCCTC will enhance the effect of the current loop feedforward control, however, one usually tries to avoid setting very large values to these parameters.

Notice that the original following error will be cancelled out by the action of the feedforward control if the feedforward control is set precisely. Thus, theoretically, the servo gain factor for the position control gain \( (K_p) \) should not affect the following error during trajectory tracking.

2.4. Friction compensation

When the machine moves in a circle, position errors occur at the boundary of each quadrant. These errors are caused by friction and are commonly called “spikes”. A set of associated parameters is defined so as to compensate for this effect. The parameters are defined as follows: FRICTION_COMPENSATION_ENABLE (FCE), enables friction compensation; FRICTION_COMPENSATIONAMPLITUDE (FCA), the friction compensation value (amplitude); and FRICTION_COMPENSATION_TIME_CONSTANT (FCTC), the friction compensation time constant. If FCA and FCTC were tuned perfectly, there would be no “spikes” on the circle. Fig. 2 shows four possible situations where the parameters are set too large or too small.

In Fig. 2a, the real path is concave on the boundaries of quadrants, and the concave area is

![Fig. 2. Spikes caused by friction.](image_url)
sharp. This means that FCA was set too large. In Fig. 2b, the real path is concave on the boundaries of quadrants, and the concave area is round. This means that FCTC was set too large. In Fig. 2c, the real path is protruding on the boundaries of quadrants, and the protruding area is sharp. This means that FCA was set too small. In Fig. 2d, the real path is protruding on boundaries of quadrants, and the protruding area is round. This means that FCTC was set too small.

2.5. Loop gain analysis

Intrinsically, the loop gain can be interpreted as $K_v = \text{velocity/following error (unit: 1000/min )}$. In this experiment our this gain is defined by a the POSITION_LOOP_GAIN parameter (PLG). A small loop gain can result in a large following error. A large loop gain, on the other hand, may induce overshoot and instability. Theoretically, the loop gain of all the axes can be identical. But this is only good when all the speed control loops have the same dynamic behavior. This can be achieved by tuning the PLG$_v$ of each axis or by adjusting the dynamic matching parameters introduced above. Note that there may still be some differences among the axes after this tuning process is performed. Therefore, the best position control gain values for the individual axis may still be slightly different. These are the values that are supposed to be determined by the automatic GA algorithm.

The position loop control starts working right after the loop gain is set. Notice that the AXIS_ACCELERATION (AA) parameter, which is the maximum axis acceleration value, should not be set too large because nonlinear effects, such as saturation, might be invoked and position loop overshoot may become apparent. The recommended value for AA is from 0.5 to 2 mm/s$^2$. The maximum acceleration value should be set slightly smaller than the maximum current allows. If it is set to the same value as the maximum current limit, extra current will be needed to overcome the additional cutting force, and saturation may appear.

3. CNC parameter tuning based upon the GA

The GA is now introduced. It is desirable to give the controller intelligence so that optimal system performance can evolve. The main elements in the GA are the chromosomes. A chromosome is composed of several genes. Different gene values cause the performance of chromosomes to vary. However, one cannot directly give the “best” gene values needed to formulate the best chromosome because each gene may or may not couple with another. That means that one never knows where the “optimal point” is. Therefore, the GA is designed to simulate the result of human evolution. Different chromosomes exchange genes through what is called crossover so as to generate new chromosomes. Some genes in the chromosomes might change without crossover occurring and this is called “mutation”, which is also part of the process in the natural evolution. In the program we have developed, a random value is generated from 0 to 1. If, for example, the mutation rate is set to be 0.2, than a random value less than 0.2 will produce a program mutation command, and a value larger than 0.2 will not. The newly generated chromosomes are tested according to the assigned rule to determine if they should be kept in a “survival pool” or discarded. The GA followed in this research uses chromosome to represent combination of a certain controller parameters. Each chromosome, or each set of motion parameters, gives a performance index
Fig. 3. GA tuning experiment procedure. (A) New parameters generated by the GA and transmitted to the Siemens control kernel using the Windows DDE method and Siemens MPI network. (B) Measured data of the machine and drives and the associated machine status. (C) 5 servo motors used to drive 3 linear axes and 2 rotary axes. (D) Position sensor signals from encoders of the 5-axis machine tool.

that describes how the machine tool works. Crossover in our research means motion control parameter exchanging between different chromosomes. Mutation means that a motion parameter in one chromosome changes its value according to some probabilities. In the “survival” pool, better chromosomes are added and poor chromosomes are pushed out. In this research, the quality of a chromosome is based on the machine circular motion precision. By repeating this process, it is possible to get a good set of machine parameters (a good chromosome). This set of parameters will be a good candidate for high-precision machining in the machine tool system.

In our setup, a personal computer serves as an open architecture controller, which transmits newly computed motion parameters to the Siemens 840D kernel through the MPI network. A reset signal is also transmitted after the parameter is transferred so that new parameters can take effect. After new parameters become active, a “part program” is transmitted to command that a circular motion be performed. The position data of each axis during circling are gathered from the MPI network and saved on a hard disk. The PC then computes the following error. A proposed “fitness function” is then used to evaluate the chromosome, and evolution takes place. The procedure is shown in Fig. 3.

3.1. Experiment on parameters evolving synchronously with the GA

The five parameters in Table 1 dominated the position loop performance. In the first stage of the experiment, the XY plane of the 5-axis machine was selected as the target. According to the

<table>
<thead>
<tr>
<th>No</th>
<th>Full parameter names</th>
<th>Abbreviations of parameter names</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>(FRICTION_CONTROL_TIME_CONSTANT)</td>
<td>FCTC</td>
</tr>
<tr>
<td>2</td>
<td>FRICTION_COMPENSATION_AMPLITUDE</td>
<td>FCA</td>
</tr>
<tr>
<td>3</td>
<td>SPEED_FEEDWARD_GAIN</td>
<td>SFG</td>
</tr>
<tr>
<td>4</td>
<td>DYNAMIC_MATCHING_TIME_CONSTANT</td>
<td>DMTC</td>
</tr>
<tr>
<td>5</td>
<td>POSITION_LOOP_GAIN</td>
<td>PLG</td>
</tr>
</tbody>
</table>
above table, there were five different motion control parameters on each axis, so there were totally 10 variables for the XY plane in the experiment. For the experiment, a “chromosome” was defined as containing 10 “genes”; each gene corresponded to a motion control parameter. Each chromosome could be regarded as an individual representing a specific parameter combination. Then, a “chromosome pool” was created which had a certain number of chromosomes. Some chromosomes were “selected” as parent chromosomes, and “crossover” and “mutation” processes generated child chromosomes. Therefore, each child chromosome had different genes compared with the parent chromosomes. However, these genes had the “features” of the parent chromosomes. Then the performance of the child chromosomes was determined. After all the fitness values of the child chromosomes were determined, the child chromosomes and parent chromosomes competed in generating new set of parent chromosomes called Parent Generation 2. The process was then repeated to generate Parent Generations 3, 4, 5 and so on. This process continued until all the fitness of the parent chromosomes fell into the range of convergence. This ended the evolution process. The resulting genes served as parameter settings in the CNC controller for the 5-axis machine. In this experiment, the fitness function was the average of the errors between the target circle (command) and the actual position data, written as

\[
F = \frac{1}{N} \sum \text{abs}(\sqrt{(X_i - X_c)^2 + (Y_i - Y_c)^2} - R)
\]

where \( F \) is the fitness (the smaller the better); \( N \) the number of data acquired; \( X_i \) the X-axis position data; \( Y_i \) the Y-axis position data; \( X_c \) the X-axis circle command data; \( Y_c \) the Y-axis circle command data; \( R \) the circle radius.

4. Experimental results

4.1. Experiment A

The experiment data obtained were stored in the PC. It took 24 generations to achieve convergence. A VB program is designed to implement the GA and transmission functions. Fig. 4 shows the operation display of the GA software.

The steps in the experiment were as follows:

1. Assign the feed rate (speed) of circular motion on the XY plane of the 5-axis machine.
2. Assign the center position of circular motion.
3. Assign the number of the parent chromosomes and that of child chromosomes.
4. Initialize randomly the gene values of the parent chromosomes of the first generation.
5. Begin evolution (crossover, mutation and selection).
6. Repeat step 5 until all the fitness values of the parent chromosomes fall below the convergence value.
7. End evolution.

The Experiment A conditioning data were as follows.
feed rate: 2000 mm/min;
circle radius: \(20\sqrt{2}=28.284\) mm (where the distance from the initial point to the center point was 20 mm along the X-axis and 20 mm along the Y-axis);
center positions: \(X = -400, Y = -200\);
the number of parent chromosomes: 45;
the number of child chromosomes: 30;
gene number: 10;
selection method for crossover: Roulette Circle method;
crossover method: Uniform Crossover;
mutation rate: 0.2;
mutation amplitude: 0.8;
convergence range: 5 \(\mu\)m.

The *Roulette Circle* method (Fig. 5) was used as the selection method in the GA algorithm. Because the small fitness was to be determined, the degree of performance for the chromosome was \(RF=1/(Fitness+0.001)\).

Assume that the parent number is 6. Then, the fitness values of 6 chromosomes in a given generation are calculated. The RF values are then computed. These 6 RF values then constitute the circumference of a circle called the Roulette Circle. With a random selection, it is obvious that the chromosome with larger share has a better chance of being selected. This process determines the parent chromosome for each child, and better chromosomes will have a better chance of getting selected. Because the best chromosome is not always selected, one can avoid the monotonous effect, and this helps avoid immature evolution. The proposed approach also uses the popular uniform crossover and mutation techniques. Convergence means after \(n\) generations of evolution, all the fitness values of the parent chromosomes are within the convergence range.

We choose uniform crossover as the crossover method. Uniform crossover is used to randomly generate child chromosome from the two selected parent chromosomes. That is, the child’s gene is either the father chromosome’s gene or the mother chromosome’s gene. The chance for selecting
the farther of the mother is equal and the method is applied to each gene. For example, if a chromosome contain 5 genes, then we randomly generate the binary array as follows:

$$[01011]$$

Because the first binary value is 0, the first gene is chosen to be the gene from the father chromosome. The second binary value is 1, so the second gene come from the mother chromosome, and so on. This is called uniform crossover, and it was the method adopted.

This ends the evolution. The flow chart of the GA program is shown in Fig. 6. The program flow chart is explained in detail in the following.

A. Initializing chromosomes means to assign random parameters values to the chromosomes. In random initialization, limitations for maximum and minimum values are given for each motion control parameter.

B. Checking fitness means computing the machine circular positioning errors defined by the formula in the previous section. Getting a new parent generation means to get in good child chromosomes and push out bad parent chromosomes. The pool always keeps the same number of chromosomes.

C. Checking parent convergence means the fitness values of all the chromosomes are within a preset value.

D. If the current chromosomes are not satisfactory, the evolution should continue. Selection uses the Roulette Circle to choose chromosomes for gene crossover and gene mutation.

In the experiment results, $P_{xx}$ represents the $xx$th chromosome. The results are sorted and listed from the best chromosome (P1) to the worst chromosome (P45). The following table shows the physical meaning of each gene (Table 2).

The first generation parent genes were generated randomly. Take parameter 9 (the $X$-axis position gain) and parameter 10 (the $Y$-axis position gain) as an example, Fig. 7 shows the their initial values. Fig. 7 also shows the corresponding fitness values. The 45 chromosomes are sorted.
The GA algorithm.

![Diagram of the GA algorithm]

Table 2
Physical meaning of the genes

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>X-axis friction compensation time constant</td>
</tr>
<tr>
<td>2</td>
<td>Y-axis friction compensation time constant</td>
</tr>
<tr>
<td>3</td>
<td>X-axis friction compensation constant maximum value</td>
</tr>
<tr>
<td>4</td>
<td>Y-axis friction compensation constant maximum value</td>
</tr>
<tr>
<td>5</td>
<td>X-axis velocity feedforward weight</td>
</tr>
<tr>
<td>6</td>
<td>Y-axis velocity feedforward weight</td>
</tr>
<tr>
<td>7</td>
<td>X-axis dynamic match time constant (unit: ms)</td>
</tr>
<tr>
<td>8</td>
<td>Y-axis dynamic match time constant (unit: ms)</td>
</tr>
<tr>
<td>9</td>
<td>X-axis position control loop P gain</td>
</tr>
<tr>
<td>10</td>
<td>Y-axis position control loop P gain</td>
</tr>
</tbody>
</table>

from best, number 1, to worse, number 45. The values vary from 2.3 to 73.9. One observes no significant correlation between the gene values and the fitness values. Experience tells us, however, that the position loop gains should play an important role in the machine servo performance. This is explained by the fact that all the genes are generated randomly, therefore their corresponding fitness values can vary widely.

Evolution includes selection, crossover, mutation, fitness evaluation, and competition to produce the next generation of parent chromosomes. The number of parents is 45 and they produce 30 children. With this choice, it can take some hours for evolution to finish. The numbers of the
parents and the children cannot be further reduced because reduced population also reduced the
chance to generate good children. The convergence range in experiment A is 5 µ, and the goal
is achieved after 24 generations of evolution. Fig. 8 shows the resulting chromosomes in the
final generation.

The GA tuning result from Experiment A is a very good result. Basically, the control result
obtained by a very experienced CNC servo engineer with the help of the CNC controller tuning
tool can have an average following error of 5.6 µ. The result obtained using the proposed GA
tuning approach is 2.3 µ, an obvious improvement. Notice that this is the result obtained using a
commercial CNC machine. The prospects for future implementation are very good.

After evolution, the 9th gene and 10th gene of the final chromosomes (the X and Y position
loop gains) all converge to 3.0621 and 3.1030. The other genes values can vary greatly. This
leads the authors to suspect that these two genes play a dominant role in the fitness function.
This is not saying that these genes are “absolutely dominant” because one might design a different
fitness algorithm. Notice that the convergence range was set to 5 µ, and that parameters 1–8 gave
rise only to very small changes in the fitness value (in the circular test); thus, only parameters 9
and 10 converged to certain values and other parameters did not. The other parameters (1–8) did
not have any opportunity to “evolve” so as to converge to a certain value. If one sets the conver-
gence region to a smaller value, this might allow parameters 1–8 to evolve further to reach convergence.

4.2. Experiment B

Experiment B was performed with the following data:

feed rate: 2000 mm/min;
circle radius: \(20\sqrt{2}=28.24\) mm (where the distance from initial point to the center point is 20 mm along X-axis and 20 mm along Y-axis); center positions: \(X = 400, Y = 200\); the number of parent chromosomes: 45; the number of child chromosomes: 30; gene number: 10; selection method in crossover: the Roulette Circle method; crossover method: Uniform Crossover; mutation rate: 0.3; mutation amplitude: 0.8; convergence range: 5 µ.

In this experiment, all the parameters were the same as in Experiment A except that the mutation rate was increased to 0.3. The evolution of this process took 17 generations. It is interesting to note that the fitness of the chromosomes in the final generation showed a large range of variation (ranging from 2.39 to 4.76 µ instead of 2.34 to 3.2 µ in Experiment A).

4.3. Experiment C

In an effort to resolve the problem of “bad convergence quality” due to the large variation in the fitness value, this research employed a new method in Experiment C, a “dynamic mutation amplitude” method. Experiment C was conducted under the same conditions studied in Experiment B except that the mutation rate was set to 0.3 and the mutation amplitude started at 0.8 and was reduced to 0.4 after 70% of the chromosomes were within convergence range (the convergence range was still set to 5 µ). The idea was to reduce the frequency of mutation when a large portion of the chromosomes had shifted to a good “trend” toward evolution. This could also be regarded as a slow down in evolution when the final goal was close.

Experiment C took 19 generations to converge. This was 5 generations faster than Experiment A but 2 generations slower than Experiment B. However, as expected, this result was obtained at the expense of slightly slower convergence; the convergence values now ranged from 2.19 to 3.595 µ.

4.4. Further experiments

In the fourth experiment, the machine was run at a higher feed rate. The results obtained could be useful for high-speed machine applications. For actual machining implementations, different parameter sets could be combined together and switched on line to match different machining processes. In this experiment, the feed rate was set to 5000 mm/min, and the convergence range was relaxed to 10 µ. Convergence took 27 generations. It is reasonable that evolution was more difficult because the higher feed rate induced stronger nonlinear effects and disturbances in the system. The command frequency was higher, and more unpleasant high frequency characteristics of the plant were excited. Parameters 9 and 10 converged to 3.5143 and 3.7598, respectively. The gains were higher but the tendency of the Y-axis to require higher gain was still observed (the Y-axis had greater inertia).
The first four experiments were carried out on the XY plane. If one can focus on the 3D motion performance of 5-axis machining, errors in the XY, YZ, and ZX plane circles should be observed. The new circular test experiments were carried out on these three planes. All the conditions were kept the same (a 2000 mm/min federate) except that instead of 10 parameters in the XY plane, this experiment used 15 parameters because three axes were involved. After 60 generations of evolution, the fitness value ranged from 24.487 to 26.173 μm, and it took 1.5 h of PC time. Convergence was slow because the total number of parameters had been increased to 15.

To show of the effectiveness of the process, Fig. 9 shows the circling diagram of a “bad” chromosome. The parameters have not yet been tuned by the proposed automatic GA method. The maximum error reaches 144.20 μm, and it is obvious that the trajectory following characteristics are bad. Fig. 10 shows the result for a good chromosome. The trajectory following character-
istics have clearly improved, and the maximum error is now reduced to 2.69 µm. As explained above, an experienced servo engineer can tune the parameters to the 5.6 µm level, and the proposed method can always achieve error less than 3 µm error.

5. Conclusions

This research used the self-designed PC-Based controller based on the Siemens 840D kernel as a test bench to implement an automatic GA based parameter-tuning algorithm. Through the evolution of genes (where each gene represents a parameter), the controller gradually adapted to achieve better control performance. The experiment results serve as a good reference for extracting the most important parameters for the controller. The experiments were carried out under different conditions to examine the convergence properties. To improve on convergence, a dynamic mutation amplitude method was introduced. The results obtained in these experiments indicate that the GA algorithm was able to achieve expert level performance. The control results were superior to the results obtained for a controller tuned by a trained expert.

Acknowledgements

This project is supported in part by the Industrial Technology Research Institute, ROC, under project number 893K51AQ2, which is a subcontract from the Ministry of Economic Affairs, ROC, and in part by the National Science Council under project number NSC 89-TPC-7-002-008.

References