

## ADAPTIVE CONTROL OPTIMIZATION IN END MILLING USING NEURAL NETWORKS

SHIUH-TARNG CHIANG,<sup>†</sup> DING-I LIU,<sup>†</sup> AN-CHEN LEE<sup>†</sup> and WEI-HUA CHIENG<sup>†</sup>

(Received 6 August 1993)

**Abstract**—In this paper, we propose an architecture with two different kinds of neural networks for on-line determination of optimal cutting conditions. A back-propagation network with three inputs and four outputs is used to model the cutting process. A second network, which parallelizes the augmented Lagrange multiplier algorithm, determines the corresponding optimal cutting parameters by maximizing the material removal rate according to appropriate operating constraints. Due to its parallelism, this architecture can greatly reduce processing time and make real-time control possible. Numerical simulations and a series of experiments are conducted on end milling to confirm the feasibility of this architecture.

### 1. INTRODUCTION

THE USE OF COMPUTER numerical control (CNC) systems has grown tremendously in recent decades. However, a remaining drawback of these systems is that the operating parameters, such as feedrate, speed, and depth of cut, are programmed off-line and the selection of these parameters is based on the part programmer's experience and knowledge. To prevent damage to the cutting tool, the operating conditions are usually set extremely conservatively. As a result, many CNC systems are inefficient and run under the operating conditions that are far from the optimal criteria.

For this reason, adaptive control, which provides on-line adjustment of the operating parameters, is being studied with interest. Adaptive control systems can be classified into two types [1]: (1) adaptive control with optimization (ACO) and (2) adaptive control with constraints (ACC). In ACO systems, the controller adjusts the operating parameters to maximize a given performance index under various constraints. In ACC systems, on the other hand, the operating parameters are adjusted to regulate one or more output parameters (typically cutting force or cutting power) to their limit values. In fact, the objective of most ACC systems is also to increase a given performance index by assuming that the optimal solution occurs on a constraint boundary.

Since machining is a time-varying process, most adaptive control methods [2] apply the recursive least squares method to in-process estimate the parameters of the special empirical formula or the linearized model. However, in many cases, no reliable model is available or the reduced linearized model is not accurate enough to depict the input/output (I/O) relationship of the cutting process. Furthermore, as more cutting constraints are taken into account, the computing time for these methods increases, because more parameters must be estimated. Therefore, neural networks, which can map the I/O relationship and possesses massive parallel computing capability, have attracted much attention in research on machining processes.

Chryssolouris and Guillot [3] modeled the machining process by a multiple regression method and a neural network and concluded that neural networks are superior to conventional multiple regression methods. Madey *et al.* [4] had a neural network learn the I/O relationship of a human operator's actions. The neural network could then work like the operator. However, this method is limited to the trained cutting conditions. Rangwala and Dornfeld [5] presented a scheme that used a multilayered

---

<sup>†</sup> Department of Mechanical Engineering, National Chiao Tung University, 1001 Ta Hsueh Road, Hsinchu 300, Taiwan, R.O.C.

perceptron neural network to model the turning process and an augmented Lagrange multiplier (ALM) method to optimize the material removal rate. They presented only a computer simulation. Later, Choi *et al.* [6] experimented with this scheme on a turning process, but employed a different optimum strategy, the barrier function (one of the sequential unconstrained minimization techniques). Since the calculation for optimization takes a great deal of time, their scheme was unable to reduce the error level between the neural network and actual lathe immediately. Thus this method may result in a false optimal result.

Although neural networks can represent more I/O relationships without increasing computing time, the networks employed in all of the previous work in this area require a great deal of time to find the optimal cutting conditions. Thus the calculated optimal conditions are far from the real optimal conditions. More recently, a special type of neural network that parallelizes the optimal algorithm has been used to solve on-line optimization problems [7].

Tank and Hopfield [8] first found that a neural network can seek to minimize the energy function and designed a neural network for finding a function minimum. Chua and Lin [9] used integrator cells to model neurons and mapped the cost function and constraints into a canonical nonlinear circuit based on the Kuhn–Tucker conditions. Kennedy and Chua [10] showed that the linear programming circuit of Tank and Hopfield [8], after some modification, can be reduced to the circuit of Chua and Lin [9]. Rodriguez-Vazquez *et al.* [11] replaced the RC-active technique by an SC-reactive technique, which is more suitable for VLSI implementation. Most of the above networks use the penalty method to solve optimization problems. However, Cichocki and Unbehauen [12] proposed a structure similar to Rodriguez-Vazquez *et al.*'s [11], that, unlike his, employed an ALM method. Their structure parallelizes existing optimal algorithms and constitutes a parallel network. Because of their parallelism, these networks make on-line optimization possible.

In this paper, a neural network based adaptive control with optimization (NNBACO) system that includes two different kinds of neural networks is proposed for on-line selection of optimal cutting parameters and control of the machining process. A back-propagation network with three inputs and four outputs is used as a general-purpose model to learn the end milling. A second network, which parallelizes the ALM method, finds the corresponding optimal cutting conditions based on the cost function and maximum material removal rate (MRR), subject to certain constraints. Owing to the parallel processing ability of this architecture, the processing time will not increase when more constraints are added. Numerical simulations and a set of experiments that apply this NNBACO system on end milling are presented to demonstrate its capabilities. These results show that this system is valid within the cutting conditions examined.

This paper is organized as follows. Section 2 gives an overview of neural networks. Section 3 first presents the circuit used for solving on-line optimization of a cutting process and then describes the architecture of the proposed NNBACO system. Section 4 presents a simulation that illustrates the application of the proposed structure in end milling. Section 5 describes the experimental procedure and results. Conclusions are given in the last section.

## 2. NEURAL NETWORKS

In an artificial neural network, the unit analogous to the biological neuron is referred to as a “processing element” (PE) (see Fig. 1). Like a biological nervous system, an artificial neural network consists of a large number of interconnected PEs. A PE has many inputs  $o_{j,k-1}$ , but only a single output  $o_{i,k}$ , which can fan out to many other PEs in the network. The basic function of a neuron is to sum its inputs and to produce an output. Let  $w_{i,j,k}$  be the weight between the  $i$ th input branch in the  $k$ th layer connected to the  $j$ th neuron in the  $(k - 1)$  layer. Then the output of the  $i$ th neuron  $o_{i,k}$  is given by

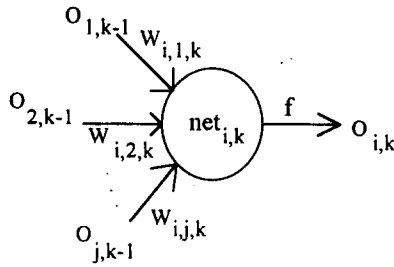


FIG. 1. Schematic diagram of the basic model.

$$o_{i,k} = f\left(\sum_j o_{j,k-1} \cdot w_{i,j,k}\right) \tag{1}$$

where  $f(\cdot)$  is called the activation function.

The operation of neural networks can be divided into two main phases: learning and recall. Learning is the process of adapting the weights in response to stimuli at the inputs. Once the weights have been adapted, the neural network has learned the I/O mapping. Recall refers to how the network processes a stimulus presented at the input and creates a response at the output.

Many kinds of neural network models have been proposed in recent years. A supervised processing neural network, back-propagation, is introduced in the following. The back-propagation network, shown in Fig. 2, was introduced by Rumelhart and McClelland [13] in 1986. A typical back-propagation network is a supervised multilayer network that includes an input layer, an output layer, and at least one hidden layer and in which each layer is fully connected to the succeeding layer.

As illustrated by the solid lines in Fig. 2, the stimuli are fed into the input layer and propagated forward to the output layer. The output is compared with the desired one and then the error signal is propagated backward through the network, as shown by the dashed lines, to upgrade the weights of each layer. The name back-propagation is derived from the backward propagation of the error signal.

The main steps in the back-propagation algorithm are summarized as follows:

- STEP 1 Initialize all weights  $w_{i,j,k}$  with small random values.
- STEP 2 Present input patterns and specify the corresponding desired outputs.
- STEP 3 Calculate the actual outputs of all the nodes, using the present value of the weights, by

$$o_{i,k} = f(\text{net}_{i,k}) \tag{2}$$

where

$$\text{net}_{i,k} = \sum_j [w_{i,j,k} \cdot o_{j,k-1}] \tag{3}$$

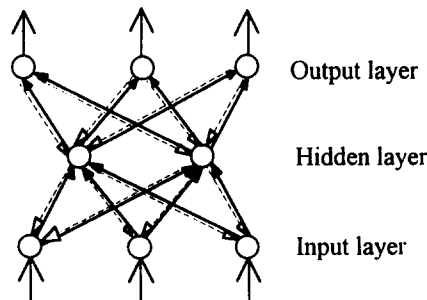


FIG. 2. Three-layered back-propagation neural network.

STEP 4 Find an error term  $\delta_j$  for an output node using the equation

$$\delta_{i,k} = (d_i - o_{i,k}) \cdot f'(\text{net}_{i,k}) . \quad (4)$$

For a hidden layer node, find the error term using the equation

$$\delta_{j,k} = f'(\text{net}_{j,k-1}) \cdot \sum_l [\delta_{l,k+1} \cdot w_{l,i,k+1}] \quad (5)$$

where  $l$  is the number of neurons in the layer above node  $j$ .

STEP 5 Adjust the weights by

$$w(k+1) = w(k) + \eta \cdot \delta_{i,k} \cdot o_{i,k} + \beta \cdot (w(k) - w(k-1)) , \quad (6)$$

where  $\eta$  is the learning rate and  $\beta$  is a constant, between 0 and 1, which determines the effect of past weight change on the current direction of movement in the weight space.

STEP 6 Present another input and go back to STEP 2 cyclically until all the weights converge.

Since neural networks have a highly parallel structure, they are well suited to parallel implementation. Such an implementation can result in very fast processing and can achieve a very high degree of fault tolerance. In addition, since neural networks can naturally process many inputs and have many outputs, they are readily applicable to multivariable systems. Because of these promising features, recently neural networks have been widely applied in fields such as image compression, character recognition, and automatic control [14]. Moreover, specially designed chips for neural networks have also been developed.

### 3. NEURAL NETWORK BASED ADAPTIVE CONTROL WITH OPTIMIZATION IN END MILLING

In what follows, the mathematical formulation for optimization of cutting conditions is described and the parallel structure used for solving the on-line optimization problem of a cutting process is developed. The neural network based adaptive control with optimization (NNBACO) system applied to the end milling process is proposed in the last subsection.

#### 3.1. Mathematical formulation for optimization of cutting conditions

In a cutting process, in order to prevent damage to the cutting tool and maintain the minimum acceptable workpiece surface finish, there are upper bounds on the cutting forces in the  $X$  and  $Y$  directions ( $F_x, F_y$ ), the power ( $P$ ), and the surface finish ( $R_a$ ). Similarly, because of the variety of cutting tools used and variations in machine capacity, the input variables (feedrate per tooth ( $f_t$ ), axial depth of cut ( $A_d$ ) and radial depth of cut ( $R_d$ )) also have their own operating ranges. Although there are many factors that restrict the operating conditions, a high material removal rate is also required. Hence, the above description can be transformed into an optimization problem: under the limitations on the inputs and outputs, find a set of optimal inputs that will maximize the MRR, i.e.

maximize the performance index

$$F = f_t \cdot A_d \cdot R_d \quad (7)$$

subject to the following constraints on the input variables:

$$\begin{aligned} \text{Min. of } f_t &\leq f_t \leq \text{Max. of } f_t \\ \text{Min. of } R_d &\leq R_d \leq \text{Max. of } R_d \\ \text{Min. of } A_d &\leq A_d \leq \text{Max. of } A_d \end{aligned} \quad (8)$$

and the following constraints on the output variables:

$$\begin{aligned}
 F_x - \text{Allowable } F_x &\leq 0 \\
 F_y - \text{Allowable } F_y &\leq 0 \\
 P - \text{Allowable } P &\leq 0 \\
 R_a - \text{Allowable } R_a &\leq 0 .
 \end{aligned} \tag{9}$$

Since the above optimization problem must be solved on-line during control of a machining process, a special type of neural network is introduced to solve this problem in the next subsection.

### 3.2. Optimization using neural networks

Nonlinear constrained programming is a basic tool in systems where a set of design parameters are optimized subject to inequality constraints. Many numerical algorithms have been developed for solving such problems [15]. The main disadvantage in applying these conventional optimal algorithms to many industrial applications is that they generally converge slowly. However, in many engineering and scientific problems, e.g. automatic control, on-line optimization is required.

The goal of this section is to propose a parallel structure, one that parallels the conventional optimal algorithm, to solve this problem. The adapted optimization theory is described first.

Consider the following nonlinear constrained optimization problem: minimize a scalar cost function

$$F(\mathbf{X}) = F(x_1, x_2, \dots, x_n) \tag{10}$$

subject to the inequality constraints

$$G_j(\mathbf{X}) \leq 0 \quad j = 1 \dots m \tag{11}$$

and the equality constraints

$$H_k(\mathbf{X}) = 0 \quad k = 1 \dots q \tag{12}$$

where the vector  $\mathbf{X}$  is referred to as the vector of design variables. The sequential unconstrained minimization technique (SUMT) is one method used to solve constrained optimization problems. It turns a constrained problem into an unconstrained one. After that, an unconstrained optimization method can be applied.

The SUMT creates a pseudo-objective function of the form

$$A(\mathbf{X}, r_p) = F(\mathbf{X}) + r_p \cdot \Gamma(\mathbf{X}) , \tag{13}$$

where  $F(\mathbf{X})$  is the original cost function and  $\Gamma(\mathbf{X})$  is an imposed penalty function, the form of which depends on the SUMT employed. The scalar  $r_p$  is a multiplier that determines the magnitude of the penalty. Since the augmented Lagrange multiplier (ALM) method is an efficient and reliable SUMT, it will be adapted in the following.

In the ALM method, the pseudo-objective function is

$$\begin{aligned}
 A(\mathbf{X}, \lambda, r_p) = F(\mathbf{X}) + \sum_{j=1}^m \{ \lambda_j \cdot (G_j(\mathbf{X}) + z_j^2) + r_p \cdot (G_j(\mathbf{X}) + z_j^2)^2 \} \\
 + \sum_{k=1}^q \{ \lambda_{k+m} \cdot H_k(\mathbf{X}) + r_p \cdot (H_k(\mathbf{X}))^2 \} ,
 \end{aligned} \tag{14}$$

where  $\lambda_j$  is the Lagrange multiplier and  $z_j$  is a slack variable. Because the new variable  $z_j$  has been added in equation (14), it greatly increases the number of design variables. According to Rockafellar [16], equation (14) is equivalent to

$$A(\mathbf{X}, \lambda, r_p) = F(\mathbf{X}) + \sum_{j=1}^m \{ \lambda_j \cdot \phi_j + r_p \cdot \phi_j^2 \} + \sum_{k=1}^q \{ \lambda_{k+m} \cdot H_k(\mathbf{X}) + r_p \cdot (H_k(\mathbf{X}))^2 \}, \tag{15}$$

where

$$\phi_j = \text{Max} \left[ G_j(\mathbf{X}), \frac{-\lambda_j}{2 \cdot r_p} \right] \tag{16}$$

and the upgrade formulas for  $\lambda_j$  are

$$\lambda_j^{p+1} = \lambda_j^p + 2 \cdot r_p \cdot \phi_j \quad j=1 \dots m \tag{17}$$

$$\lambda_{k+m}^{p+1} = \lambda_{k+m}^p + 2 \cdot r_p \cdot H_k(\mathbf{X}) \quad k=1 \dots q. \tag{18}$$

A detailed flow chart of this algorithm is shown in Fig. 3.

Now, applying a general gradient strategy in the unconstrained part, we develop the recursive discrete-time algorithm

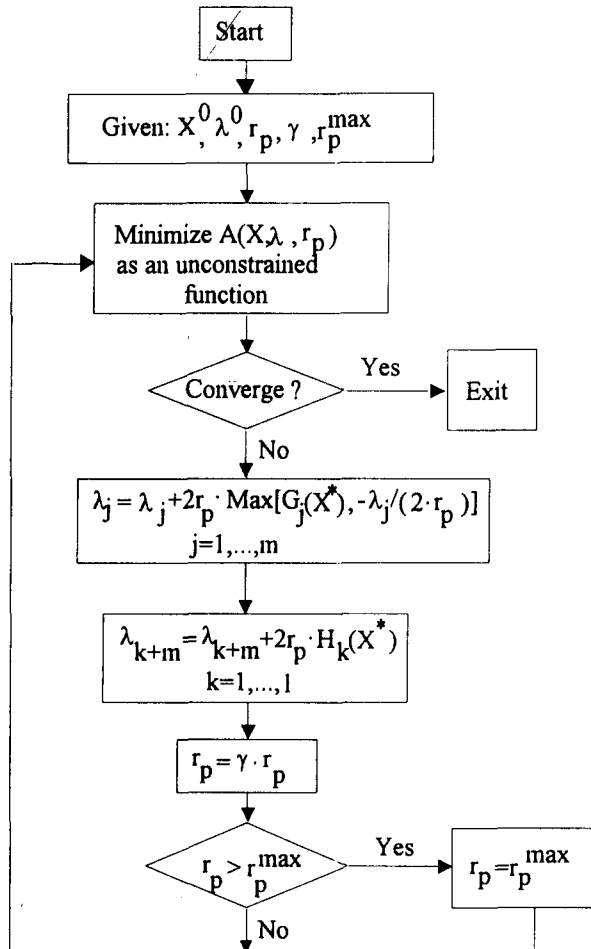


FIG. 3. Algorithm for the ALM method [15].

$$\mathbf{X}(t+1) = \mathbf{X}(t) - \mu_t \cdot \nabla A(\mathbf{X}, \lambda, r_p) \quad (19)$$

where  $\mu_t$  is the step size in the  $t$ th iteration and  $\nabla A(\mathbf{X}, \lambda, r_p)$  is the gradient of  $A(\mathbf{X}, \lambda, r_p)$ , which is defined as

$$\nabla A(\mathbf{X}, \lambda, r_p) = \left[ \frac{\partial A}{\partial x_1}, \frac{\partial A}{\partial x_2}, \dots, \frac{\partial A}{\partial x_n} \right]^T \quad (20)$$

in which

$$\begin{aligned} \frac{\partial A}{\partial x_i} &= \frac{\partial F(\mathbf{X})}{\partial x_i} + \sum_{j=1}^m s_j \cdot (\lambda_j + 2 \cdot r_p \cdot \phi_j) \cdot \frac{\partial G_j}{\partial x_i} \\ &+ \sum_{k=1}^q (\lambda_{k+m} + 2 \cdot r_p \cdot H_k) \cdot \frac{\partial H_k}{\partial x_i} \end{aligned} \quad (21)$$

and

$$s_j = \begin{cases} 1 & \text{if } G_j(\mathbf{X}) \geq \frac{-\lambda_j}{2 \cdot r_p} \\ 0 & \text{otherwise.} \end{cases}$$

The optimization problem to be solved as given in equations (7)–(9) can be presented in a mathematical form as follows:

maximize the performance index (MRR)

$$F(\mathbf{X}) = x_1 \cdot x_2 \cdot x_3 \quad (22)$$

subject to the inequality constraints

$$\begin{aligned} G_j(\mathbf{X}) &= x_j - x'_j & j=1,2,3 \\ G_j(\mathbf{X}) &= x''_{j-3} - x_{j-3} & j=4,5,6 \\ G_j(\mathbf{X}) &= y_{j-6}(x_1, x_2, x_3) - y'_{j-6} & j=7,8,9,10, \end{aligned} \quad (23)$$

where  $x'_j$  and  $x''_j$  are the upper and lower limits on the input variables  $x_j$  and  $y'_j$  is the allowable value of the output  $y_j$ . Since there are no equality constraints, the partial differential term of  $H_k(\mathbf{X})$  with respect to  $x_i$  in equation (21) can be removed. Substituting equations (22) and (23) into (21), we obtain

$$\begin{aligned} \frac{\partial A}{\partial x_i} &= x_j \cdot x_k + s_i \cdot (\lambda_i + 2 \cdot r_p \cdot \phi_i) - s_{i+3} \cdot (\lambda_{i+3} + 2 \cdot r_p \cdot \phi_{i+3}) \\ &+ \sum_{j=1}^4 \left[ s_{j+6} \cdot (\lambda_{j+6} + 2 \cdot r_p \cdot \phi_{j+6}) \cdot \frac{\partial G_{j+6}}{\partial x_i} \right], \end{aligned} \quad (24)$$

where  $\phi_i$ ,  $\phi_{i+3}$ , and  $\phi_{i+6}$  depend on the subscript and they are as follows:

$$\begin{aligned} \phi_j &= \text{Max} \left[ G_j(\mathbf{X}), \frac{-\lambda_j}{2 \cdot r_p} \right] = \text{Max} \left[ (x_j - x'_j), \frac{-\lambda_j}{2 \cdot r_p} \right], & \text{if } j=1,2,3 \\ \phi_j &= \text{Max} \left[ G_j(\mathbf{X}), \frac{-\lambda_j}{2 \cdot r_p} \right] = \text{Max} \left[ (x''_{j-3} - x_{j-3}), \frac{-\lambda_j}{2 \cdot r_p} \right], & \text{if } j=4,5,6 \\ \phi_j &= \text{Max} \left[ G_j(\mathbf{X}), \frac{-\lambda_j}{2 \cdot r_p} \right] = \text{Max} \left[ (y_{j-6} - y'_{j-6}), \frac{-\lambda_j}{2 \cdot r_p} \right], & \text{if } j=7,8,9,10. \end{aligned} \quad (25)$$

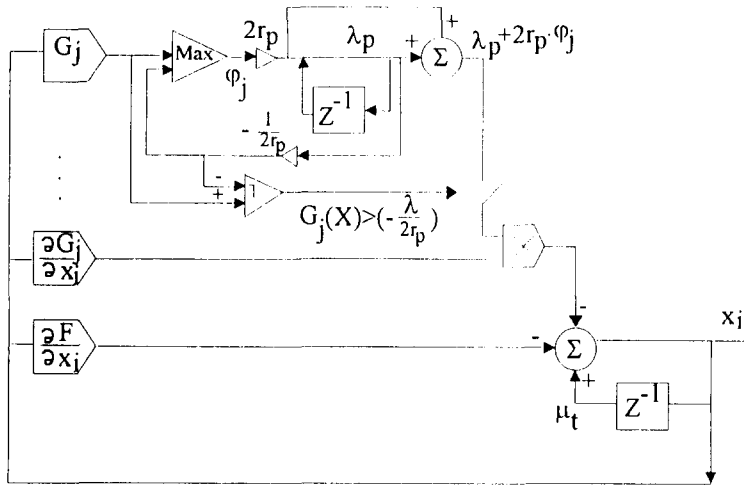


FIG. 4. Circuit of a constrained nonlinear optimization solver based on the ALM method.

The parallel structure corresponding to equation (24) is proposed and shown in Fig. 4, where the step size  $\mu_t$  is adopted as a constant. The main feature of this architecture is that it uses the ALM method, which converges more quickly than other penalty methods. This optimization circuit is not like the neural networks described in Section 2, but it has many of the features of neural networks. For example, this circuit performs massive parallel processing of analog signals and is capable of adapting the connection weights to accelerate convergence during the optimization process.

### 3.3. Procedure for NNBAO system in end milling

The proposed NNBAO system is shown in Fig. 5, in which there are two different kinds of neural networks employed. Neural network (I) is used to learn the appropriate mappings between the input and output variables of the machining process, so we shall refer to it as the neural network for modeling. This neural network is a three-layered back-propagation network with three input nodes, each representing feedrate per tooth, axial depth of cut, and radial depth of cut, and four output nodes, i.e. the forces in the X and Y directions, cutting power, and the surface finish of the workpiece.

Neural network (II), which is described in Section 3.2, is used to determine the optimal inputs, so we shall refer to this network as the neural network for optimization. Since neural network (I) is used to describe the cutting process, the derivatives in equation (24) can be calculated by a forward pass through the back-propagation network. The detailed derivation is shown in the Appendix. This can greatly reduce computing time in solving the optimization problem.

The procedure of this NNBAO system can be summarized as follows:

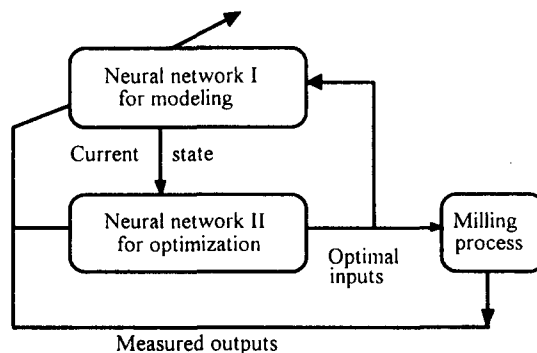


FIG. 5. The neural network based ACO (NNBAO) system.



- STEP 1 Under the initial constraints, neural network (II) determines a set of optimal inputs and then sends them into the milling machines and neural network (I).
- STEP 2 The measured outputs of the milling machine, corresponding to the optimal input, are used as the desired output to train neural network (I).
- STEP 3 Neural network (II) uses the newly upgraded neural network (I) as the model of the end milling process to find the optimal inputs and sends them into the milling machine and neural network (I).
- STEP 4 STEP 2 and STEP 3 are repeated until the termination of the cutting process.

#### 4. SIMULATION

In the following, the theoretical model of the end milling process is described. A simulation employing this theoretical model and the results are presented to confirm the feasibility of the NNBACO system.

##### 4.1. Theoretical models of end milling

Since there are four cutting constraints taken into account in this simulation (the average cutting force in the  $X$ ,  $Y$  directions, cutting power and surface finish) we use four corresponding theoretical models to describe the end milling. These models are presented below.

The theoretical models for the average cutting forces in the  $X$ ,  $Y$  directions presented by Lee *et al.* [17] are expressed as

$$F_x = K_t \cdot N_f \cdot f_t \cdot A_d \cdot \{K_r \cdot [\sin(2 \cdot \beta_{en}) - 2 \cdot \beta_{en}] + [1 - \cos(2 \cdot \beta_{en})]\} / (8 \cdot \pi) \quad (\text{N})$$

$$F_y = K_t \cdot N_f \cdot f_t \cdot A_d \cdot \{K_r \cdot [1 - \cos(2 \cdot \beta_{en})] + [2 \cdot \beta_{en} - \sin(2 \cdot \beta_{en})]\} / (8 \cdot \pi) \quad (\text{N}), \quad (26)$$

where  $\beta_{en}$  is the tooth entry angle,  $N_f$  is the tooth number of the end mill,  $K_t$  indicates the ratio of tangential cutting force to the chip load, and  $K_r$  indicates the ratio of radial to tangential cutting force.  $K_t$  and  $K_r$  for the aluminum with hardness 55 HRB are

$$K_t = 757.7045 \cdot f_t^{-0.0558} \quad (\text{N/mm}^2)$$

$$K_r = 0.2627 \cdot f_t^{-0.2279} . \quad (27)$$

The theoretical model for the cutting power can be expressed as

$$P = 1.6 + 4.26 \cdot [(N_f \cdot A_d \cdot R_d \cdot f_t)^{0.66}] \cdot \text{rpm} / 97422 \quad (\text{kW}), \quad (28)$$

where rpm is the spindle speed, expressed in revolutions per minute.

According to [18], the surface finish of the workpiece can be modeled by

$$R_a = f_t^2 / (8 \cdot R) \quad (\text{mm}) \quad (29)$$

in which  $R_a$  is the average peak to valley height on the workpiece surface and  $R$  is the radius of the end mill. The deformation of the cutting tool is ignored here.

##### 4.2. Computer simulation and results

In order to investigate the stability and adaptation of the NNBACO system, we simulated a procedure similar to that described in Section 3.3 and the real end milling process was replaced by the theoretical model, i.e. equations (26)–(29).

Since an untrained neural network has no knowledge about the system, unpredictable results may be obtained. Therefore, initial training was applied to neural network (I).

In initial training, the first step is to generate the I/O samples from the theoretical model. Different values spanning the allowable range of each input variable yield a total of 420 input combinations. These input variables were used to determine the

output quantities, and they composed the set of I/O samples which were used for initial training and testing.

The three-layered back-propagation network with three inputs and four outputs described in Section 3.1 was employed for neural network (I). In the initial training, we start with three hidden nodes and the number of nodes was increased until a low error rate was achieved. The error rate is the ratio of incorrect samples, i.e. samples in which the absolute difference between the theoretical output and the output of the neural network exceeds 10%, to all 420 samples. The simulation results for the initial training, with ten hidden-layer nodes, are shown in Fig. 6; the results indicate that after 150 iterations the error rate approached 10%.

If the number of iterations used in training the neural network is increased, the error rate may be reduced slightly. But, from a macroscopic point of view, the error rate remains in the vicinity of 10%. Increasing the number of nodes in the hidden layer may also decrease the error rate, but it may make it more difficult to realize the proposed structure physically and may cause more time to be consumed in calculating the optimal inputs. Hence ten hidden-layer nodes were chosen in this stage.

According to the experiment set-up and workpiece used in our laboratory, the limits on the input and output variables that were used in the simulation were as follows: The upper limitation on the input variables:

$$f_t: 0.33 \text{ (mm/tooth)}$$

$$R_d: 12.5 \text{ (mm)}$$

$$A_d: 24.5 \text{ (mm)} .$$

The allowable outputs:

$$F_x: 800 \text{ (N)}$$

$$F_y: 1600 \text{ (N)}$$

$$P: 1.8 \text{ (kW)}$$

$$R_a: 0.0015 \text{ (mm)} .$$

In order to examine the performance of the NNBACO system under specific cutting conditions, one of the input variables (radial depth of cut) was specified as varying between three specific values, and it changed from one value to another every one hundred iterations. Through this simulation, not only the static (constant  $R_d$ ) but also the dynamic (varying  $R_d$ ) performance of this system were investigated.

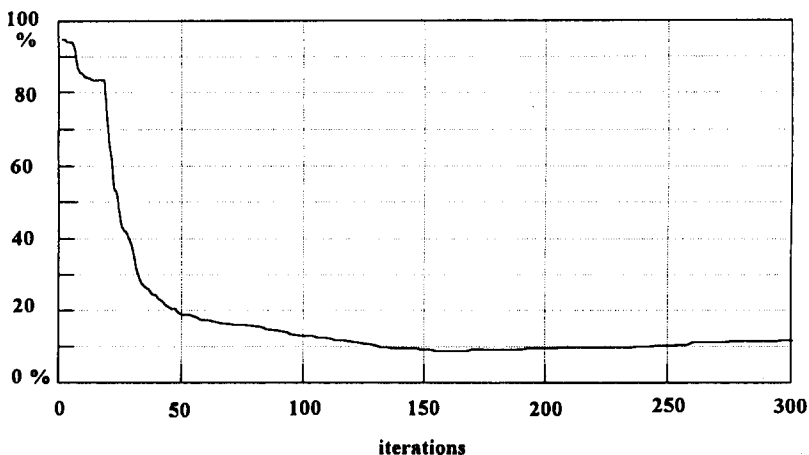


FIG. 6. Learning error in the initial training.

Figure 7(b) shows the three values (2, 6 and 10 mm) for the radial depth of cut, and Figs 7(a) and (c) show the corresponding optimal inputs ( $A_d$  and  $f_t$ ). Because the neural network can not “learn” as fast as the change in the radial depth of cut, there were some transient states at the beginning of each state. When the radial depth of cut is 2 mm, the inequality constraints of the cutting power are satisfied more easily than when the depth is 6 or 10. Hence the product of  $f_t$  and  $R_d$ , shown in Fig. 7(d), has the greatest value when  $R_d = 2$ . In these figures, we also find that as the system varies, the steady optimal inputs do as well.

The neural network’s outputs corresponding to the optimal inputs are shown in

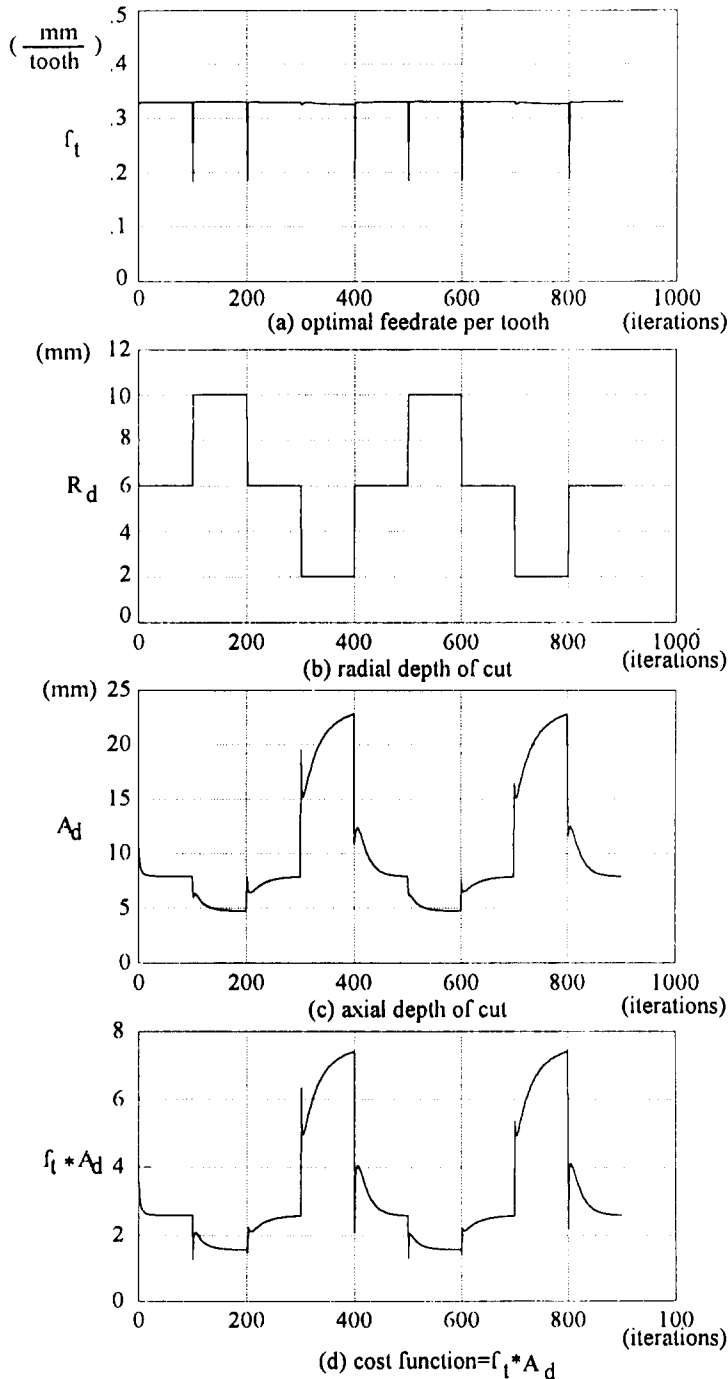


FIG. 7. Optimal inputs in simulation.

Fig. 8. Because the calculation of the optimizer is based on the neural network, all the inequality constraints, i.e. equation (23), are satisfied. However, in order to satisfy the inequality constraints for the cutting power, all the other outputs of the neural networks are far from the upper limits.

Most of the outputs of the theoretical model, shown in Fig. 9, are below the maximum limitations. But some overshoots can be detected in the cutting power during the changes between states. We also find that in these figures the maximum MRR is mainly limited by the cutting power.

The learning errors in the simulation are shown in Fig. 10, in which the error is defined as follows

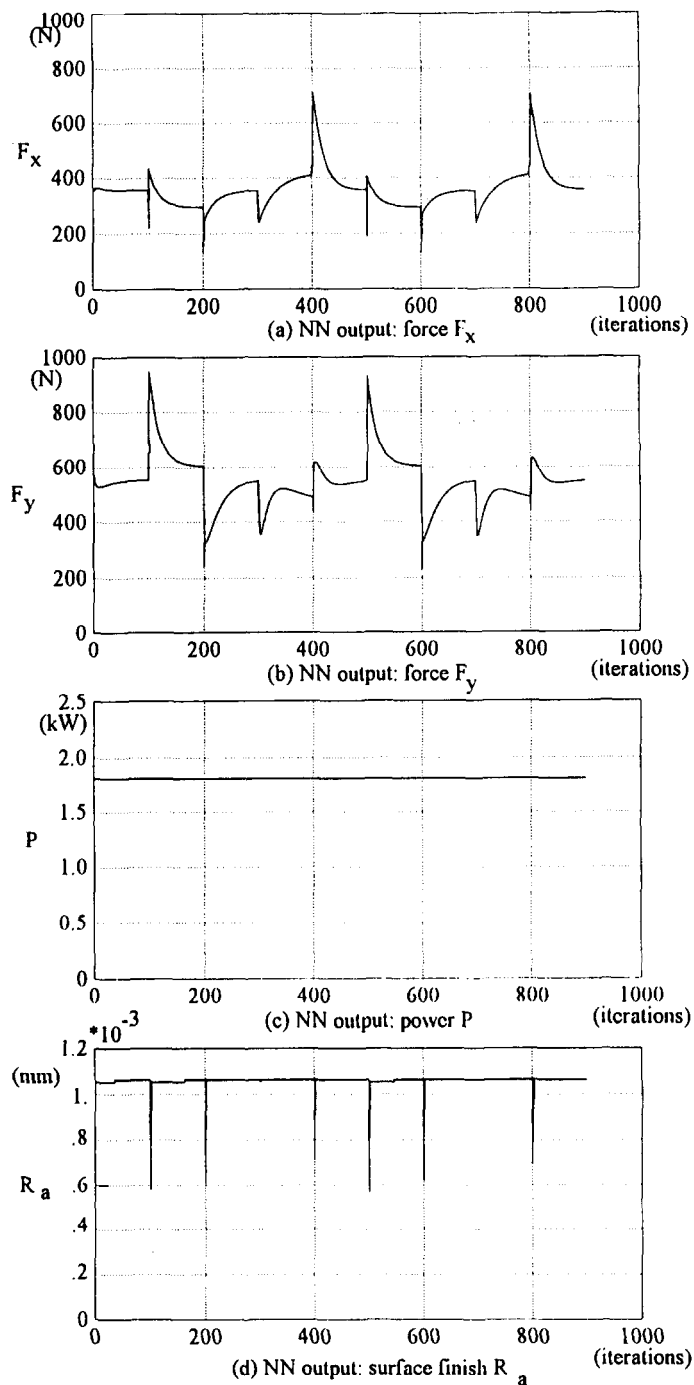


FIG. 8. Neural network outputs in simulation.

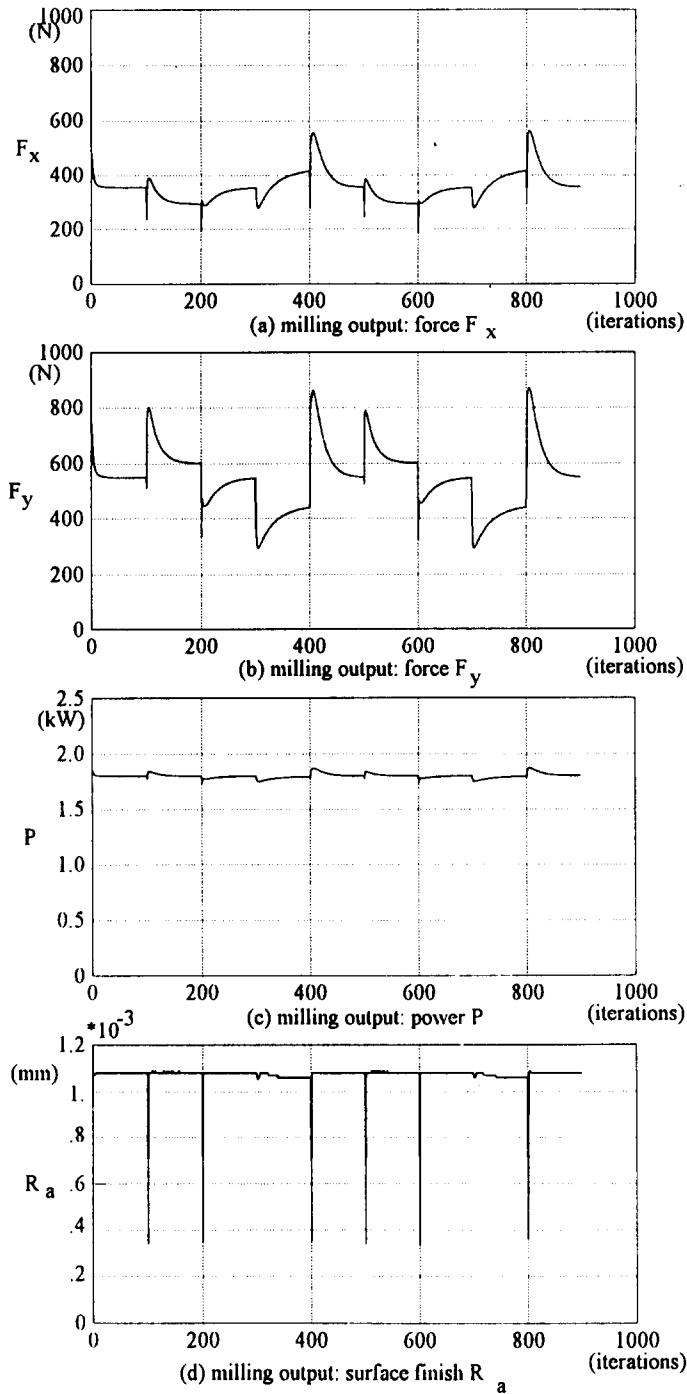


FIG. 9. Outputs from the theoretical model in simulation.

$$\text{error} = \frac{|\text{theoretical output} - \text{neural network output}|}{\text{theoretical output}} \quad (30)$$

Because of large variation of weights at the beginning of step change in  $R_d$ , the optimizer takes more iterations to find a set of optimal inputs. As the neural network approaches the steady state, the optimal inputs can be obtained in fewer iterations. This means that the neural network can describe the system more and more accurately in the static state. Compared with the results of initial training, the error diminished very quickly.

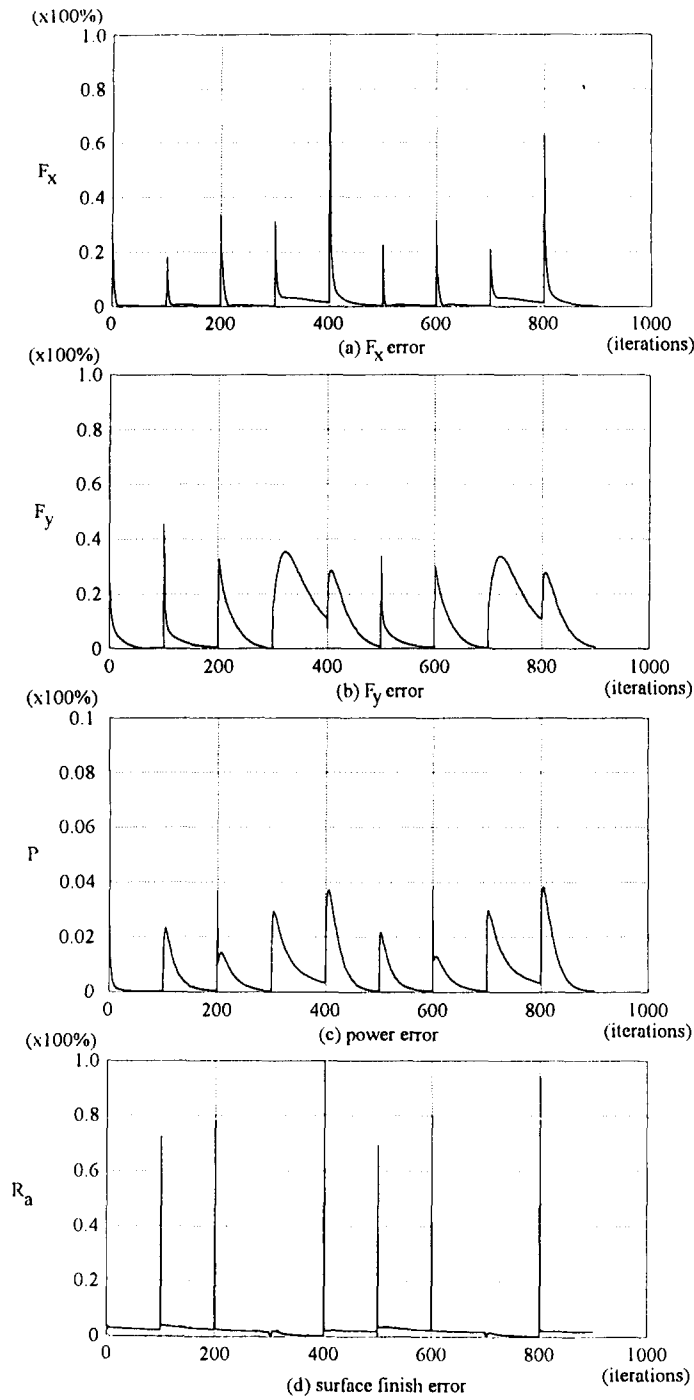


FIG. 10. The learning errors in the simulation.

After the simulations in this section, we adopted some patterns from the initial training set to test the underlying neural network and found that for patterns located near the optimal inputs the neural network generated outputs with a very small error. But for patterns far from the optimal values, the network generated outputs with a large error. This result implies that the neural network has only partial knowledge after training.

## 5. EXPERIMENTS

5.1. *Experimental set-up*

We conducted a series of experiments with end milling to confirm the feasibility of the proposed NNBACO system.

The experimental set-up for the NNBACO system is shown in Fig. 11, which also defines the positive directions of the three axes. The feedrate command to the servo-driver was generated by a hardware interpolator. Each NC servo loop (per axis) included a DC servo motor and a P.I.D. Velocity and position feedback were provided by tachometer and encoders.

A vertical knee-type milling machine (3 horse power spindle motor) was used for the experiments. The cutting forces were measured by a dynamometer (Kistler 9257B) and a charge amplifier (Kistler 5007). The A/D converter used was a 12-bit one; the collected data were saved to the memory of a 486 personal computer by DMA (direct memory access) transfer. The sampling time in the A/D converter was 1 ms and the NNBACO system updated the feedrate parameter in 100 ms.

The data processing and the control algorithm were implemented using Borland C++ V.3.0 on a 32-bit 486 PC. The other machining conditions and parameters of the workpiece are listed below:

Workpiece: aluminum alloy (T6061).

Cutting tool: end mill,  
25 mm diameter,  
four teeth,  
160 mm total length.

Cutting conditions:  
spindle speed: 300 rpm,  
axial depth of cut: 25 mm.

No coolant.

5.2. *Experiments*

We conducted three main series of experiments, in which two differently shaped workpieces were machined. Details of the experimental conditions and the dimensions of the workpiece are shown in Figs 12(a) and (b).

The first experiment is conventional cutting that the feedrate per tooth was set to be constant under the constraints on the maximum allowable cutting forces.

In the second experiment, the proposed NNBACO system was applied in the end milling to demonstrate its performance. Because of the limitations of the laboratory equipment used, the neural network for modeling was set up with only two outputs,  $F_x$  and  $F_y$ , and the axial depth of cut was kept at 25 mm. There were ten nodes in the hidden layer. The radial depth of cut and other cutting conditions were set up as described in Figs 12(a) and (b).

In the last experiment on each workpiece, regardless of the variation in radial depth

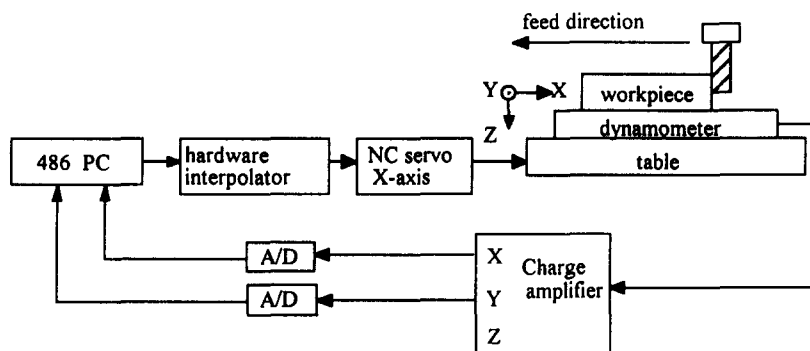
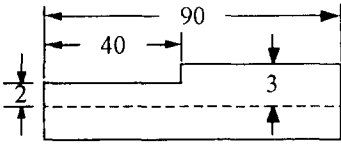


FIG. 11. The experimental set-up.

(a)

<p>Down milling</p> <p>Workpiece A:</p>  <p>(unit: mm)</p>	
Experiment 1 (Constant feedrate)	Case: case a-1 Cutting condition $f_t$ : 0.12 mm/tooth Result: Figure 13
Experiment 2 (NNBACO system is apply)	Case: case a-2 Cutting condition $f_t$ : 0.075-0.25 mm/tooth Result: Figure 14
Experiment 3 (Rd to NN is kept constant = 2 mm)	Case: case a-3 Cutting condition $f_t$ : 0.075-0.25 mm/tooth Result: Figure 15
Maximum allowable force — $F_x$ : 400 N, $F_y$ : 450 N	

(b)

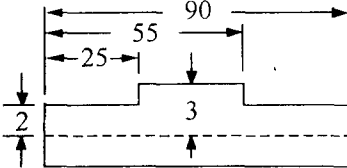
<p>Down milling</p> <p>Workpiece B:</p>  <p>(unit: mm)</p>	
Experiment 1 (Constant feedrate)	Case: case b-1 Cutting condition $f_t$ : 0.12 mm/tooth Result: Figure 16
Experiment 2 (NNBACO system is apply)	Case: case b-2 Cutting condition $f_t$ : 0.075-0.25 mm/tooth Result: Figure 17
Experiment 3 (Rd to NN is kept constant = 2 mm)	Case: case b-3 Cutting condition $f_t$ : 0.075-0.25 mm/tooth Result: Figure 18
Maximum allowable force — $F_x$ : 400 N, $F_y$ : 450 N	

FIG. 12. (a) Cutting conditions for workpiece A. (b) Cutting conditions for workpiece B.

of cut, the input (radial depth of cut) to the neural network was fixed at specific “false” values as defined in Figs 12(a) and (b). The difference between the “real” and “false” input is like a noise value. Thus, the error tolerance of the NNBACO system can be investigated from this experiment.



### 5.3. Results and discussion

The geometry of the first workpiece, shown in Fig. 12(a), consisted of two steps. In order to satisfy the force limitation (the maximum allowable forces in the  $X$ ,  $Y$  directions were 400 and 450 N, respectively), the feedrate was kept at 0.12 (mm/tooth) in case a-1. The measured force is shown in Fig. 13.

The NNBACO system was applied in case a-2, in which the feedrate decreased to a lower level as the cutting tool stepped up to a higher stair. The results are shown in Fig. 14. The error defined is similar to equation (30), but the theoretical outputs are replaced by the measured signals. Since the neural network cannot learn very accurately when the radial depth of cut changes, there are two peaks in the measured force. After a while, however, the measured force in the  $X$  direction stabilized around the preset value, which means that the optimal cutting conditions are constrained by the force in the  $X$  direction in this case. Due to the sudden change in the input to the neural network, there is a peak in the neural network's output shown in Fig. 14(c).

Although the radial depth of cut varied from 2 to 3 mm, the input to the neural network was kept at 2 mm throughout the tool path in case a-3. The results, as depicted in Fig. 15, show that the peak in the neural network's output vanished, because there was no change in the input to the network. However, the two peaks are still present in the measured force and the error diagram, i.e. Figs 15(b) and (d).

For the other cases of experiment, there were three steps in the second workpiece, and the distance between each step was shorter. This workpiece, as shown in Fig. 12(b), includes ascending and descending parts. The maximum allowable forces in the  $X$ ,  $Y$  directions were again set as 400 and 450 N, respectively.

In case b-1, feedrate was kept at 0.12 (mm/tooth). The measured forces are shown in Fig. 16.

The results of case b-2 obtained using the NNBACO system are shown in Fig. 17. For the same reasons described in case a-2, there are four peaks in the neural network's output. In Fig. 17(c), we see that the optimal cutting conditions are limited sometimes by the force in the  $X$  direction and sometimes by the force in the  $Y$  direction. From the Fig. 17(d), we see that the error in the third step is not less than that in the first step. It means that the system only possesses knowledge near about optimum region. As the state ( $R_d$ ) returns to the previous state, the system also needs to reoptimize the cutting conditions as before.

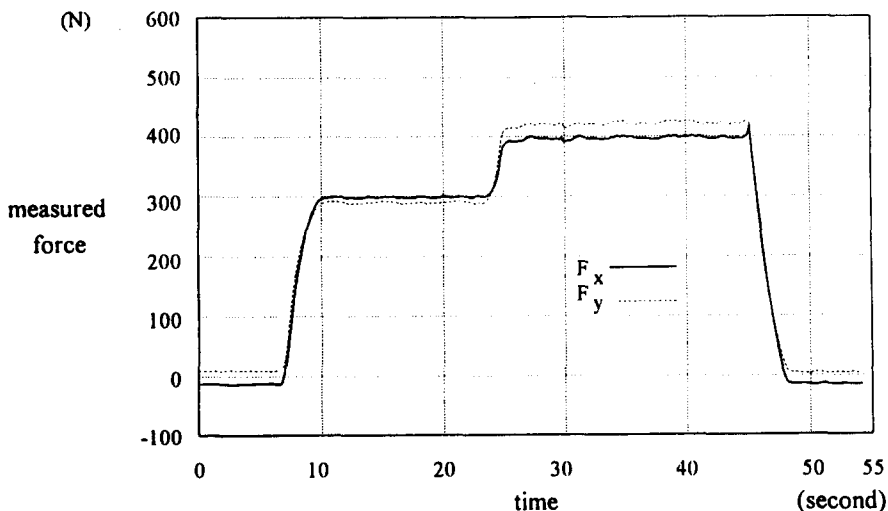


FIG. 13. The results of case a-1.

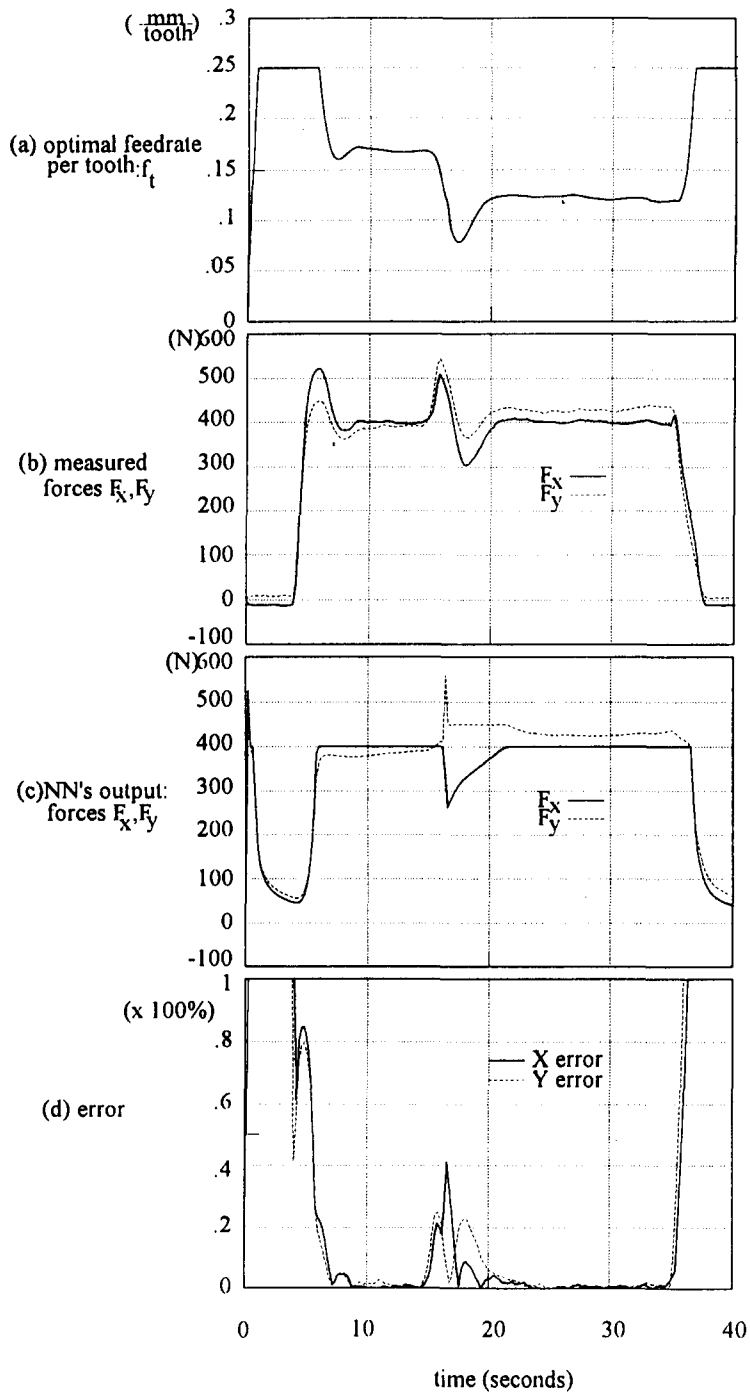


FIG. 14. The results of case a-2.

For the results of the last case b-3, Fig. 18, as in the previous case, we see that the optimal conditions are always constrained by the maximum allowable force in the  $X$  direction. When the results of Figs 17(d) and 18(d) are compared, they seem to be different. In the case of Fig. 17(d), as the radial depth of cut changes, the input of neural network suddenly changes which causes the weights to update tremendously and generates larger force errors. In the case of Fig. 18(d), this situation is avoided

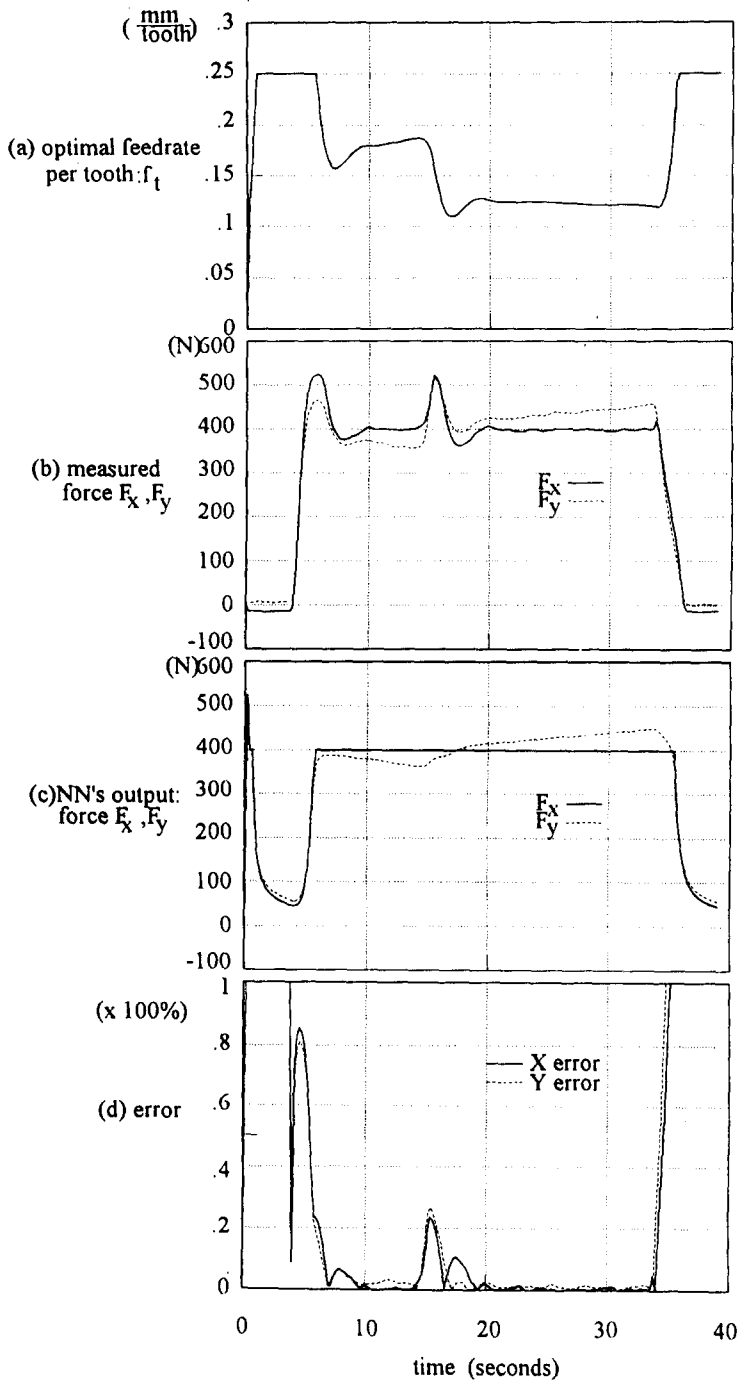


FIG. 15. The results of case a-3.

and smooth transitions of the errors occur as true radial depth of cut changes. However, for large variation of depth of cut, the NNBACO still need to give correct input to reduce the convergence time.

The results of the real-time control experiments presented above show that when applied in end milling the NNBACO system is stable within the range of the cutting conditions examined. Since the cost function is MRR and the radial and axial depth

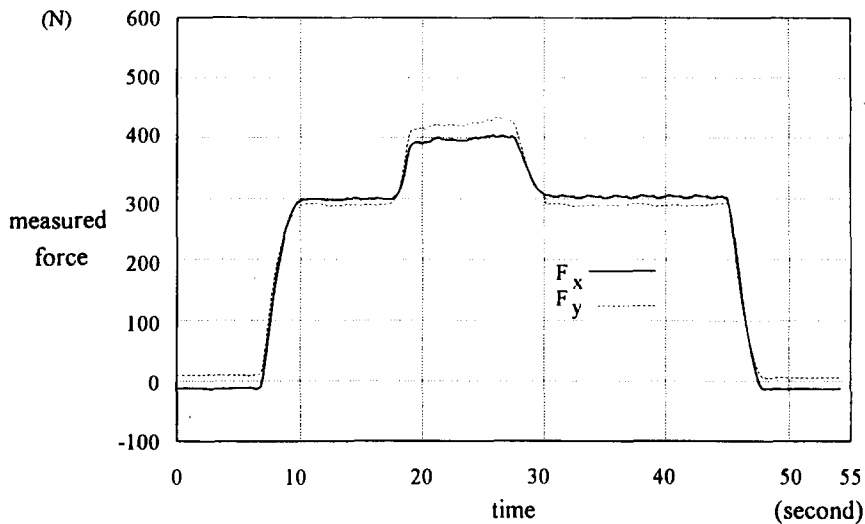


FIG. 16. The results of case b-1.

of cut are kept at specific values in each case studied, we can use the time needed to cut through the workpiece as an index of efficiency. That is, the less time spent, the higher MRR is. The time spent in cutting through each workpiece is given in Table 1. From the table, it is clear that when the NNBACO system is applied MRR is increased greatly.

Although a "false" input was applied in the last experiment on each workpiece, the MRR for these experiments is still high. Moreover, in some cases, because of less variation in the neural network's input ( $R_d$ ), these cases perform better than the "true" input cases. This shows that the NNBACO system is an architecture with high fault tolerance.

## 6. CONCLUSIONS

In this paper, an architecture is presented for on-line determining optimal cutting conditions in an end milling process. The proposed NNBACO system, which includes two different neural networks, differs from conventional adaptive control with optimization (ACO) systems in that (1) multi-constraints are handled simultaneously without increasing the processing time; (2) no specific model exists, but rather a back-propagation network is employed and (3) a special optimal mechanism is adopted to determine optimal cutting conditions.

Although the two neural networks in the NNBACO system are not realized by chip technology, the simulations and experiments presented here show that this architecture can effectively describe the behavior of end milling and increase the cutting efficiency.

Because the neural network (II) performs the optimization in the NNBACO system, neural network (I), the modeling network, only possesses knowledge in the vicinity of a local optimum. In the simulations and the experiments described here, although

TABLE 1. TIME SPENT IN END MILLING WITH RESPECT TO DIFFERENT WORKPIECE SHAPES

		Workpiece A	Workpiece B
Experiment 1 (Feedrate keeps constant)	Case	case a-1	case b-1
	Time	54.2 s	54.2 s
Experiment 2 (Apply NNBACO)	Case	case a-2	case b-2
	Time	40.0 s	34.9 s
Experiment 3 (Apply NNBACO but $R_d$ neglected)	Case	case a-3	case b-3
	Time	39.1 s	34.9 s

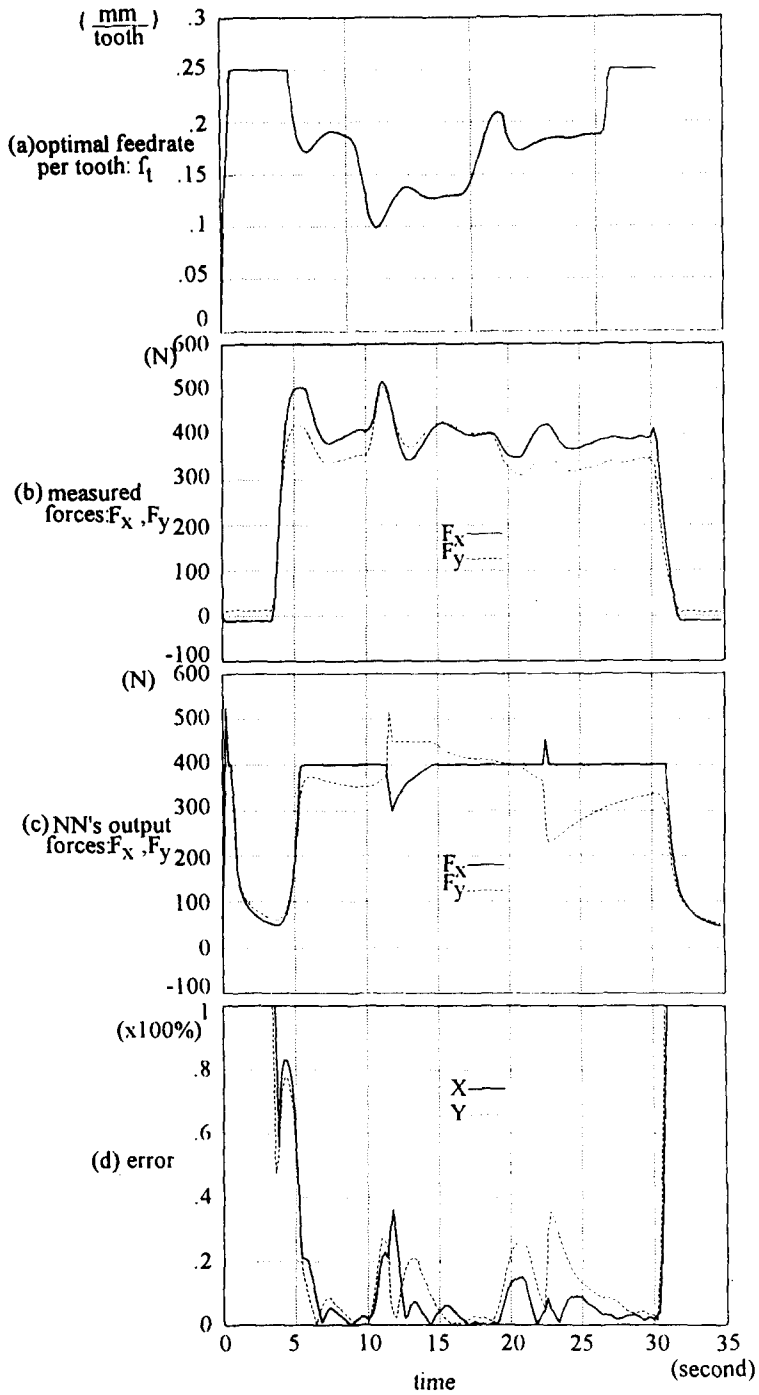


FIG. 17. The results of case b-2.

there were some errors in the beginning, the NNBACO system gradually achieved optimal cutting conditions. And since the NNBACO system itself determines the knowledge to be acquired, i.e. by the neural network for optimization, it behaves like a self-organizing system.

The NNBACO system is applied to end milling in this paper, but it is obvious that this system is a general-purpose architecture, i.e. it can be extended to other machines to improve cutting efficiency.

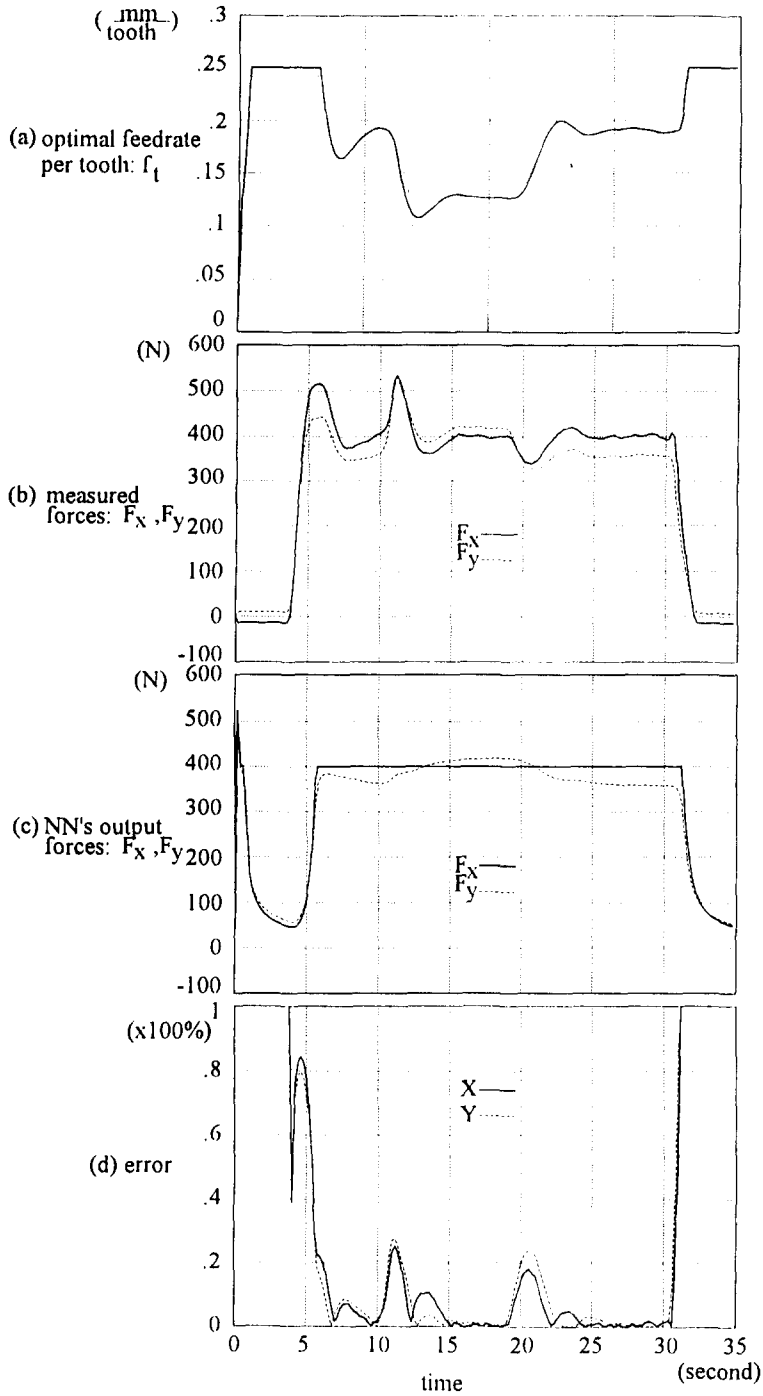


FIG. 18. The results of case b-3.

## REFERENCES

- [1] Y. KOREN, *Computer Control of Manufacturing System*. McGraw-Hill, New York (1983).
- [2] M. A. ELBESTAWI, *ASME J. Dyn. Syst. Meas. Control* **112**, 611 (1990).
- [3] G. CHRYSOLOURIS and M. GUILLLOT, *ASME J. Engng Ind.* **112**, 122 (1990).
- [4] G. R. MADEY, J. WEINROTH and V. SHAH, *J. Intell. Manufact.* **3**, 193 (1992).
- [5] S. R. RANGWALA and D. A. DORNFELD, *IEEE Trans. Syst. Man Cyber.* **19**, 299 (1989).
- [6] G. S. CHOI, Z. WANG and D. A. DORNFELD, *Proc. IEEE, Int. Conf. Robotics and Automation* 1567 (1991).
- [7] J. CHEN, M. A. SHANBLATT and C. MAA, *Int. J. Neural Syst.* **2**, 331 (1992).
- [8] D. W. TANK and J. J. HOPFIELD, *IEEE Trans. Circuits Syst.* **CAS-33**, 533 (1986).
- [9] L. O. CHUA and G. N. LIN, *IEEE Trans. Circuits Syst.* **CAS-31**, 182 (1984).
- [10] M. P. KENNEDY and L. O. CHUA, *IEEE Trans. Circuits Syst.* **CAS-34**, 210 (1987).

- [11] A. RODRIGUEZ-VAZQUEZ, R. DOMINGUEZ, A. RUEDA, J. L. HUERTAS and E. SANCHEZ, *IEEE Trans. Circuits Syst.* **37**, 384 (1990).
- [12] A. CICHOCKI and R. UNBEHANEN, *Int. J. Circuit Theory Appl.* **19**, 161 (1991).
- [13] D. RUMELHART and J. L. MCCLELLAND, *Parallel Distributed Processing: Explorations in the Microstructure of Recognition*, Vols 1 and 2. MIT Press, Cambridge, MA (1986).
- [14] K. S. NARENDRA and K. PARTHASARATHY, *IEEE Trans. Neural Networks* **1**(4), (1990).
- [15] G. N. VANDERPLAATS, *Numerical Optimization Techniques for Engineering Design: with Application*. McGraw-Hill, New York (1984).
- [16] R. T. ROCKAFELLER, *J. Opt. Theory Appl.* **12**, 555 (1973).
- [17] A. C. LEE, S. T. CHIANG and C. S. LIU, *J. Chinese Soc. mech. Engrs* **12**, 412 (1991).
- [18] M. C. SHAW, *Metal Cutting Principles*. Oxford University Press, New York, (1984).

## APPENDIX

In this Appendix, the method used to calculate the derivatives of equation (24) by a forward pass through the network is described. With regard to Fig. 2, we define the following nomenclature:

$net_{i,k}$  the sum of the  $i$ th neuron's inputs in the  $k$ th layer, i.e.

$$net_{i,k} = \sum_j [w_{i,j,k} \cdot o_{j,k-1}] \quad (A1)$$

$o_{i,k}$  the  $i$ th neuron's output in the  $k$ th layer, i.e.

$$o_{i,k} = f(net_{i,k}) \quad (A2)$$

if  $f(\cdot)$  is a sigmoid function then equation (A2) becomes

$$f(net_{i,k}) = \frac{1}{1 + e^{-net_{i,k}}} \quad (A3)$$

$y_i$  the output of the  $i$ th neuron in the output layer. If the neural network has three layers then  $y_i = o_{i,3}$ .

$d_i$  the desired output of the  $i$ th output node.

$w_{i,j,k}$  the weight between the  $i$ th neuron in the  $k$ th layer and the  $j$ th neuron in the  $(k-1)$ th layer.

$E$  defined as the global error

$$E = 0.5 \cdot \sum_i (d_i - o_{i,k})^2 \quad (A4)$$

Since the neural network is used for learning the cutting process,  $y_j$  is the  $j$ th output and  $x_i$  is the  $i$ th input. Then in equation (24)

$$\frac{\partial G_{j+6}}{\partial x_i} = \frac{\partial y_j}{\partial x_i} = \frac{\partial y_j}{\partial net_{i,l}} = r_{j,i,l} \quad (A5)$$

With respect to distinct layers, (A5) is different. The following derivation is divided into "the output layer" and "the hidden and input layer"

for the output layer

$$r_{i,j,n} = \frac{\partial y_j}{\partial net_{i,l}} = y_j \cdot (1 - y_j) \quad (A6)$$

for the  $(n-1)$ th layer

$$\begin{aligned} r_{i,j,n-1} &= \frac{\partial y_j}{\partial net_{i,n-1}} = \frac{\partial y_j}{\partial o_{i,n-1}} \cdot \frac{\partial o_{i,n-1}}{\partial net_{i,n-1}} \\ &= \frac{\partial y_j}{\partial net_{j,n}} \cdot \frac{\partial net_{j,n}}{\partial o_{i,n-1}} \cdot \frac{\partial o_{i,n-1}}{\partial net_{i,n-1}} \end{aligned} \quad (A7)$$

for the  $(n-2)$ th layer

$$\begin{aligned} r_{i,j,n-2} &= \frac{\partial y_j}{\partial net_{i,n-2}} = \frac{\partial y_j}{\partial o_{i,n-2}} \cdot \frac{\partial o_{i,n-2}}{\partial net_{i,n-2}} \\ &= \left\{ \sum_r \left[ \frac{\partial y_j}{\partial net_{r,n-1}} \cdot \frac{\partial net_{r,n-1}}{\partial o_{i,n-2}} \right] \right\} \cdot \frac{\partial o_{i,n-2}}{\partial net_{i,n-2}} \\ &= \left\{ \sum_r \left[ r_{j,i,n-1} \cdot w_{r,i,n-1} \right] \right\} \cdot o_{i,n-2} \cdot (1 - o_{i,n-2}) \end{aligned} \quad (A8)$$

Substituting (A8) into (A5), we can calculate the derivatives of equation (24) by a forward pass through the neural network.