01 Jun 1994

Identification of Cutting Force in End Milling Operations Using Recurrent Neural Networks

Q. Xu

K. Krishnamurthy
Missouri University of Science and Technology, kkrishna@mst.edu

Bruce M. McMillin
Missouri University of Science and Technology, ff@mst.edu

Wen Feng Lu

Recommended Citation

Q. Xu et al., "Identification of Cutting Force in End Milling Operations Using Recurrent Neural Networks," *Proceedings of the IEEE International Conference on Neural Networks, 1994. IEEE World Congress on Computational Intelligence (1994, Orlando, FL)*, vol. 6, pp. 3828-3833, Institute of Electrical and Electronics Engineers (IEEE), Jun 1994.
The definitive version is available at https://doi.org/10.1109/ICNN.1994.374821

IDENTIFICATION OF CUTTING FORCE IN END MILLING OPERATIONS USING RECURRENT NEURAL NETWORKS

Q. Xu,† K. Krishnamurthy,† B. McMillin‡ and W. Lu†
† Department of Mechanical and Aerospace Engineering and Engineering Mechanics
‡ Department of Computer Science
University of Missouri-Rolla
Rolla, MO 65401-0249

ABSTRACT

The problem of identifying the cutting force in end milling operations is considered in this study. Recurrent neural networks are used here and are trained using a recursive least squares training algorithm. Training results for data obtained from a SAJO 3-axis vertical milling machine for steady slot cuts are presented. The results show that a recurrent neural network can learn the functional relationship between the feed rate and steady-state average resultant cutting force very well. Furthermore, results for the Mackey-Glass time series prediction problem are presented to illustrate the faster learning capability of the neural network scheme presented here.

1. INTRODUCTION

End milling operations are widely used to machine complex-contoured parts. During these operations, it is important to control the cutting force. This will prevent premature tool failure or damage the work piece, and minimise tool deflection and tool vibration. But designing the control system to control the cutting force is a difficult task because it is difficult to obtain the process parameters. The metal cutting process is intermittent, and the mechanism of chip formation and cutting mechanics are not fully understood. Hence, the process parameters cannot be identified with any certainty. Often, the cutting force can only be predicted by utilizing empirical relationships and even these are highly nonlinear.

Several models for the milling process have been presented in the past. Based on the chip forming mechanics, Kline *et al.* [1] have presented a mechanistic model to obtain both instantaneous and average force values as a function of the cut geometry and feed rate. This model has been extended to handle variable cutting conditions (changes in axial and radial depths of cut, feed rate and spindle speed) by Fussell and Srinivasan [2], and Kolarits and DeVries [3]. An empirical second-order model for the force response to feed rate changes was used by Lauderbaugh and Ulsoy [4]. They showed that the model parameters vary considerable with the cutting conditions. In a later study, Lauderbaugh and Ulsoy [5] used a least squares algorithm with exponential data weighting to estimate the process parameters for designing a model reference adaptive controller. Olgac and Guttermuth [6] and Fassois *et al.* [7] used autoregressive moving average models to describe the machining process. Although these two studies considered turning operation, the same can be applied to end milling.

Considerable attention is now being focused on using neural networks for identification and control of dynamical systems [8]. Neural networks are attractive because they can be trained off-line with very high accuracy over a large input space without *a priori* knowledge of the system equations, and they can continue to learn (training and learning are used interchangeably here) during on-line application. In particular, recurrent neural networks are being considered because they have the potential for better approximation ability, shorter training period, and wider range of dynamic behavior due to their dynamical nature.

Various approaches to train recurrent neural networks have been presented. Pineda [9] has generalized the backpropagation technique to recurrent neural networks. This method requires a second dynamical system of the same size as the original system to implement the backward propagation equation in the weight update process. Pearlmutter [10] has extended Pineda's work to include time-dependent trajectories.

Karakaşoğlu *et al.* [11] have presented simplified training rules which do not require the solution of a second dynamical system. The three-layer architecture (one input layer, one hidden layer and one output layer) resembles that of feedforward neural networks. Puskorius and Feldkamp [12] and Ku and Lee [13] have used similar feedforward type architectures. The difference is that the neural network evolves according to a set of difference equations rather than differential equations.

In this study, a recurrent neural network will be used to identify the cutting force in end milling operations. Only steady slot cuts will be considered here. First, a recursive least squares (RLS) training algorithm used to train the recurrent neural network will be presented. Second, results for the Mackey-Glass time series prediction problem will be presented to illustrate the faster learning capability of the neural network scheme presented here. Finally, training results for data obtained from a SAJO 3-axis vertical milling machine will be presented. The results show that the recurrent neural network can learn the functional relationship between the feed rate and steady-state average resultant cutting force very well.

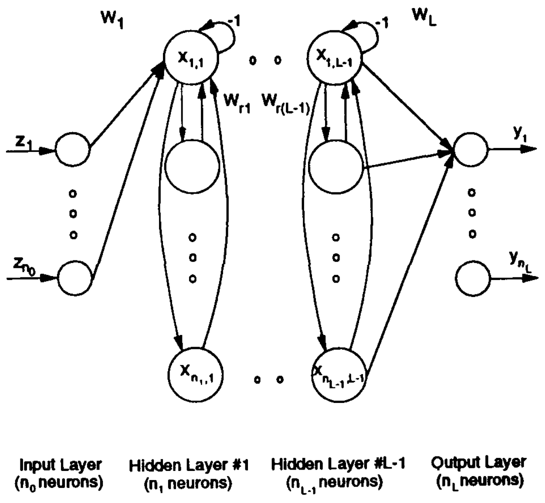
W1
-1
X1,1
-1
WL
X1,L-1
Wr1
Wr(L-1)
z1
y1
zn0
yn_L
Xn1,1
Xn_L-1,L-1
Input Layer (n0 neurons)
Hidden Layer #1 (n1 neurons)
Hidden Layer #L-1 (nL-1 neurons)
Output Layer (nL neurons)

Figure 1: Schematic of a Multilayer Recurrent Neural Network

2. TRAINING ALGORITHM

Figure 1 shows a recurrent neural network with an arbitrary number of hidden layers. The input layer is layer 0 with n_0 neurons, layers 1 - $L-1$ are hidden layers with n_1 - n_{L-1} neurons, respectively, and the output layer is layer L with n_L neurons. The hidden layers form a dynamical neural network with sigmoidal processing elements. The dynamics of the network can be described by

$$\mathbf{T}_1\,\dot{\mathbf{x}}_1 = -\mathbf{x}_1 + \mathbf{W}_{r1}\,\mathbf{g}_1(\mathbf{x}_1) + \mathbf{W}_1\,\mathbf{z}(q-1), \quad (1)$$

$$\mathbf{T}_k\,\dot{\mathbf{x}}_k = -\mathbf{x}_k + \mathbf{W}_{rk}\,\mathbf{g}_k(\mathbf{x}_k) + \mathbf{W}_k\,\mathbf{x}_{k-1}, \quad k = 2,\ldots,L-1, \quad (2)$$

$$\mathbf{y}(q) = \mathbf{W}_L\,\bar{\mathbf{x}}_{L-1}, \quad (3)$$

where $\mathbf{z}(q-1) \in \Re^{n_0}$ is the input vector at the $(q-1)$th time instant, $\mathbf{x}_k = [\,x_{1,k},\ x_{2,k},\ \ldots,\ x_{n_k,k}\,]^T \in \Re^{n_k}$ is a vector describing the state of the neurons in the kth hidden layer, $\mathbf{g}_k(.) : \Re^{n_k} \rightarrow \Re^{n_k}$ is a vector-valued function with sigmoidal elements for the kth hidden layer, $\mathbf{W}_{rk} = [\,\mathbf{w}_{1,rk},\ \mathbf{w}_{2,rk},\ \ldots,\ \mathbf{w}_{n_k,rk}\,]^T$, $\mathbf{w}_{i,rk} = [\,w_{i1,rk},\ w_{i2,rk},\ \ldots,\ w_{in_k,rk}\,]^T \in \Re^{n_k}$ denote the intra-layer connection weights from neurons in the kth hidden layer to the ith neuron within the kth hidden layer, $\mathbf{W}_k = [\,\mathbf{w}_{1,k},\ \mathbf{w}_{2,k},\ \ldots,\ \mathbf{w}_{n_k,k}\,]^T$, $\mathbf{w}_{i,k} = [\,w_{i1,k},\ w_{i2,k},\ \ldots,\ w_{in_{k-1},k}\,]^T \in \Re^{n_{k-1}}$ denote the connection weights from neurons in the $(k-1)$th layer to the ith neuron in the kth layer, $\mathbf{T}_k = \mathrm{diag}[\,T_{1,k},\ T_{2,k}, \ldots, T_{n_k,k}\,] \in \Re^{n_k \times n_k}$ is a diagonal matrix of time constants for the kth hidden layer, $\bar{\mathbf{x}}_{L-1}$ denotes the stable equilibrium state of the neurons in the $(L-1)$th layer for the input $\mathbf{z}(q-1)$ at the qth time instant, $\mathbf{y} = [\,y_1,\ y_2, \ldots, y_{n_L}\,]^T \in \Re^{n_L}$ is the output vector at the qth time instant, and the over dot denotes time derivative. For the input $\mathbf{z}(q-1)$, it is assumed that the neural network given by Eqs. (1) and (2) reaches steady-state before the qth time instant and can be obtained as

$$\begin{aligned} \bar{\mathbf{x}}_1 &= \mathbf{W}_{r1}\,\mathbf{g}_1(\bar{\mathbf{x}}_1) + \mathbf{W}_1\,\mathbf{z}, & (4)\\ \bar{\mathbf{x}}_k &= \mathbf{W}_{rk}\,\mathbf{g}_k(\bar{\mathbf{x}}_k) + \mathbf{W}_k\,\bar{\mathbf{x}}_{k-1},\ k = 2,\ldots,L-1, & (5) \end{aligned}$$

Note that a finite amount of time is included for the system to reach steady-state. This is to facilitate, for example, calculation of the steady-state solution in real-time. For convenience, the functional dependence of $\mathbf{z}$ and $\mathbf{y}$ on q will not be written out henceforth.

The problem is to find the connection weights such that the following error function is minimized.

$$^{D}E = \sum_{d=1}^{D} \beta^{D-d} \sum_{n=1}^{n_L} [\,^{d}\hat{y}_n - {}^{d}y_n\,]^2, \quad (6)$$

where β is a weight factor or forgetting factor allowing a higher weight for the last training pair, $\hat{y}_n$ and

y_n are the desired and actual outputs of the nth neuron in the output layer, respectively, and the leading superscript denotes the training pair number. The connection weights can be obtained in a recursive fashion as (see Ref. [14] for details)

$$ {}^{D}\mathbf{w} = {}^{D-1}\mathbf{w} + \eta \, {}^{D}\mathbf{k} \, {}^{D}\delta, \tag{7} $$

where η is a small learning rate, as in gradient descent, the recursive update rules for the Kalman gain ${}^{D}\mathbf{k}$ and the approximate error covariance matrix ${}^{D}\mathbf{P}$ are given by

$$ {}^{D}\mathbf{k} = \frac{{}^{D-1}\mathbf{P} \, {}^{D}\mathbf{a}}{\beta + {}^{D}\gamma^2 \, {}^{D}\mathbf{a}^T \, {}^{D-1}\mathbf{P} \, {}^{D}\mathbf{a}}, \tag{8} $$

$$ {}^{D}\mathbf{P} = \frac{1}{\beta}[\mathbf{I} - {}^{D}\gamma^2 \, {}^{D}\mathbf{k} \, {}^{D}\mathbf{a}^T] \, {}^{D-1}\mathbf{P}, \tag{9} $$

and expressions for ${}^{D}\delta$, ${}^{D}\mathbf{a}$ and ${}^{D}\gamma^2$ for the various layers are as follows:

$\underline{{}^{D}\mathbf{w} = {}^{D}\mathbf{w}_{i,L} \ (i = 1, \ldots, n_L)}$

$$ \begin{aligned} {}^{D}\delta &= {}^{D}\hat{y}_n - {}^{D}y_n \\ {}^{D}\mathbf{a} &= {}^{D}\tilde{\mathbf{x}}_{L-1} \\ {}^{D}\gamma^2 &= 1 \end{aligned} $$

$\underline{{}^{D}\mathbf{w} = {}^{D}\mathbf{w}_{i,rk} \ (i = 1, \ldots, n_k, \ k = 1, \ldots, L-1)}$

$$ \begin{aligned} {}^{D}\delta &= \sum_{n=1}^{n_L} ({}^{D}\hat{y}_n - {}^{D}y_n) \, {}^{D}\tilde{h}_{ni,rk} \\ {}^{D}\mathbf{a} &= \mathbf{g}_k({}^{D}\tilde{\mathbf{x}}_k) \\ {}^{D}\gamma^2 &= \sum_{n=1}^{n_L} {}^{D}\tilde{h}^2_{ni,rk} \end{aligned} $$

$\underline{{}^{D}\mathbf{w} = {}^{D}\mathbf{w}_{i,k} \ (i = 1, \ldots, n_k, \ k = 2, \ldots, L-1)}$

$$ \begin{aligned} {}^{D}\delta &= \sum_{n=1}^{n_L} ({}^{D}\hat{y}_n - {}^{D}y_n) \, {}^{D}\tilde{h}_{ni,rk} \\ {}^{D}\mathbf{a} &= {}^{D}\tilde{\mathbf{x}}_{k-1} \\ {}^{D}\gamma^2 &= \sum_{n=1}^{n_L} {}^{D}\tilde{h}^2_{ni,rk} \end{aligned} $$

$\underline{{}^{D}\mathbf{w} = {}^{D}\mathbf{w}_{i,1} \ (i = 1, \ldots, n_1)}$

$$ \begin{aligned} {}^{D}\delta &= \sum_{n=1}^{n_L} ({}^{D}\hat{y}_n - {}^{D}y_n) \, {}^{D}\tilde{h}_{ni,r1} \\ {}^{D}\mathbf{a} &= {}^{D}\mathbf{z} \\ {}^{D}\gamma^2 &= \sum_{n=1}^{n_L} {}^{D}\tilde{h}^2_{ni,r1} \end{aligned} $$

Here ${}^{d}\tilde{h}_{ni,rk}$ $(k = 1, \ldots, L-1)$ are the steady-state solution of

$$ {}^{d}\dot{h}_{ni,rk} = -\,{}^{d}h_{ni,rk} + g'_{i,k} \sum_{l=1}^{n_k} w_{li,rk} \, {}^{d}h_{nl,rk} + \epsilon_k, \tag{10} $$

where $\epsilon_k = \sum_{l=1}^{n_{k+1}} w_{li,k+1} \, {}^{d}h_{nl,r(k+1)}$, for $k = 1, \ldots, L-2$, $\epsilon_k = w_{ni,L}$, for $k = L-1$, and $g'_{i,k} = (\partial g_{i,k}({}^{d}x_{i,k})/\partial \, {}^{d}x_{i,k})|_{{}^{d}x_{i,k} = {}^{d}\tilde{x}_{i,k}}$. Karakaşoğlu *et al.* [11] exploited the fact it is possible to tailor the sigmoidal function such that $g'_{i,k}$ in Eq. (10) is small. Under this condition, Eq. (10) simplifies to ${}^{d}\tilde{h}_{ni,rk} = \epsilon_k$, and thus precludes the need for integrating Eq. (10), for example, to obtain the steady-state solution.

Training using the RLS algorithm is begun by initialising the **P**-matrices to be equal to the identity matrix multiplied by a large constant. Then, for each training pair, Eqs. (1) and (2) are integrated to obtain the stable equilibrium state of the network. Following this, Eq. (10) is integrated to obtain the steady-state ${}^{d}\tilde{h}_{ni,rk}$ $(k = 1, \ldots, L-1)$ values. Finally, the **k**-vector, **P**-matrix and weights for each neuron are updated. After one pass through the training set, another pass is begun. This is repeated until the error at the output is within desirable bounds. Although training the recurrent neural network is computationally more intensive than a feedforward neural network, one can argue that the recurrent neural network will generally need to be presented with the training set fewer times. Thus the overall computing time will be less, resulting in faster learning. This can be further improved upon by solving the network differential equations and implementing the RLS training algorithm on a parallel computer. In fact, Steck *et al.* [15] have shown that the computation time of the RLS algorithm for a feedforward neural network approaches that of standard backpropagation, the latter not being parallelisable, as more processors are applied to the matrix calculations in a multiple processor machine, such as the Intel iPSC/2 multicomputer.

3. RESULTS

3.1 Mackey-Glass Time Series Prediction Problem

The effectiveness of the present neural network scheme will be shown by predicting the values produced by the Mackey-Glass equation [16]

$$ \dot{x}(t) = \frac{0.2 \, x(t-\tau)}{1 + x^{10}(t-\tau)} - 0.1 \, x(t). \tag{11} $$

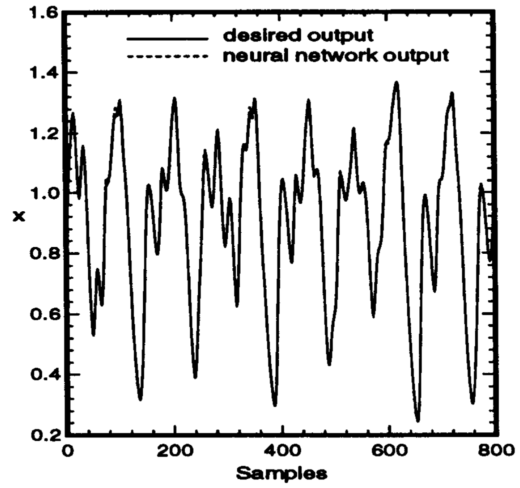

Figure 2: Training Result for the Mackey-Glass Time Series Prediction Problem

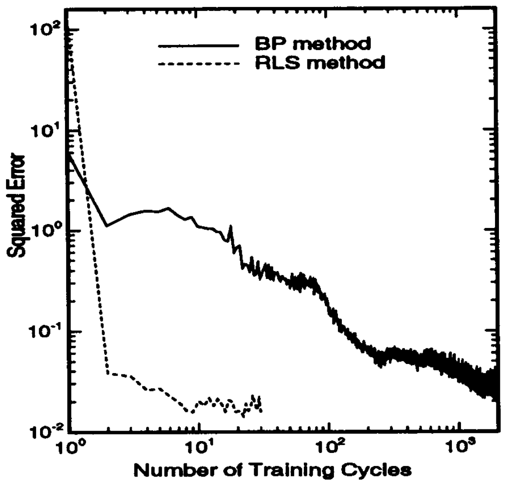

Figure 3: Learning Curves for the Mackey-Glass Time Series Prediction Problem

Specifically, a recurrent neural network will be trained to predict $x(t+6)$ using $x(t-6m)$, $m = 0, \ldots, 5$ as inputs. The Mackey-Glass time series prediction problem has been recognized as a difficult problem because Eq. (11) results in a chaotic time series.

In this study, τ was chosen to be 30 and the time series values were obtained using the 4th-order Runge-Kutta equation with the initial condition x=0.8. Training was started with random values between $\pm$ 1 for the connection weights, η=0.25, β=0.96, and ${}^0\mathbf{P}=10^3$ **I**. The forward and backward propagation equations were numerically integrated using the 4th-order Runge-Kutta equation with zero initial values. The diagonal elements of $\mathbf{T}_1$ and $\mathbf{T}_2$ were chosen to be 0.002. The input layer included one bias neuron with its value set equal to 1 and two hidden layers with 5 neurons each were chosen. The sigmoidal function chosen was $g(x) = -1 + 2/(1 + e^{-x})$. Figure 2 shows the desired and neural network outputs after 5 cycles with 800 data points each. As can be seen, the two curves are almost identical showing that the recurrent neural network has been trained to identify the nonlinear system dynamics. The sum of the squared error (squared error for short) in this case was calculated to be 2.7×10^{-2}.

To evaluate the performance of the recurrent neural network, the prediction problem was solved by training a 4-layer feedforward neural network with the same number of connection weights using the standard backpropagation algorithm. The same parameters (random values between $\pm$ 1 for the initial connection weights, bias value, sigmoidal function, training set and learning rate) as in the recurrent neural network case were chosen. Figure 3 shows the training results. It is clear that a very large number of training cycles (in excess of 2000) are required to reduce the squared error to the level achieved by the recurrent neural network in a small number of training cycles.

3.2 End Milling Problem

A recurrent neural network will be trained to learn the functional relationship between the feed rate and steady-state average resultant cutting force. Only steady slot cuts are considered in this study. A SAJO Model VF54 3-axis vertical milling machine with a 7.5 HP spindle drive was used to obtain the training data. The milling machine has been retrofitted such that it can be controlled by a 486-66 computer. A Data Translation DT2839 high channel count, high speed board was used to sample the analog cutting force signals and to generate the pulse train for driving the x-y table stepper motors. The cutting force components were measured by a Kistler Model 9257B 3-component dynamometer together with Kistler Model 5004 charge amplifiers. Slot milling was carried out with a 0.5 in diameter high speed steel 4-flute cutter on 6061 aluminum

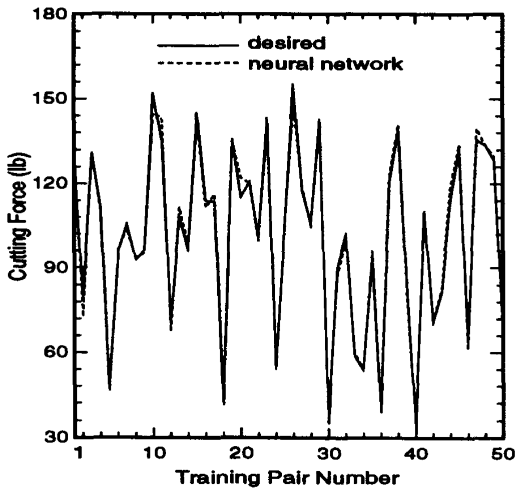

Figure 4: Desired and Neural Network Outputs - Training Result

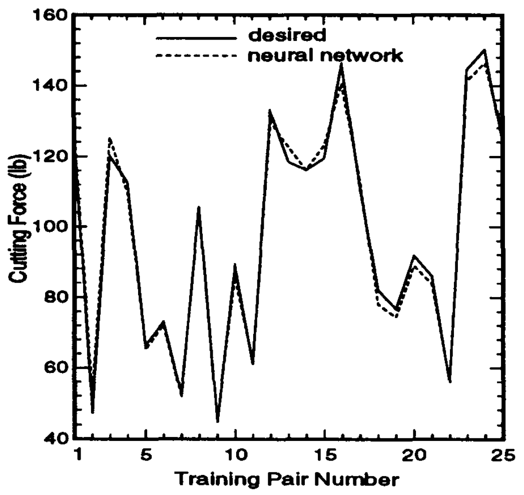

Figure 5: Desired and Neural Network Outputs - Generalization Test

blocks.

Training data was obtained by measuring the cutting force in the x and y directions for various feed rates between 0.006 in/s and 0.08 in/s. The axial depth of cut was set at 0.1 in and the force signals were sampled at 1000 samples/s. As the focus of attention here is on steady slot cuts, the force signals were measured under steady cutting conditions. Because of the mechanics of the cutting process, the sampled force signals vary as the cutter rotates. To solve the fluctuating force problem, past studies have used either the peak forces or forces averaged over one spindle revolution. As the tool deflection is of concern, the average resultant cutting force was used in this study. This was calculated as

$$F_r = \sqrt{F_x^2 + F_y^2}, \qquad (12)$$

where F_x and F_y are the sampled force signals averaged over one spindle revolution in the x and y directions, respectively. Note that the cutting force in the z direction was not considered because it was small and the slots were cut in the x direction. Thus the training set consists of the feed rates and corresponding steady-state average resultant forces. A total of 75 feed rate-resultant force training pairs were obtained.

The recurrent neural network was trained using the leaving-one-out strategy with one exception; twenty five training pairs were left out each time. After 3 times, the cycle was repeated. The inputs to the recurrent neural network were the current and past two samples of the feed rate, and past two samples of the steady-state average resultant cutting force. The output of the recurrent neural network is the current steady-state average resultant cutting force. Training was started with random values between $\pm$ 1 for the connection weights, η=0.35, β=0.96, and $^0\mathbf{P}=10^3$ **I**. The forward and backward propagation equations were numerically integrated using the 4th-order Runge-Kutta equation with zero initial values. The diagonal elements of $\mathbf{T}_1$ and $\mathbf{T}_2$ were chosen to be 0.002. The input layer included one bias neuron with its value set equal to 1 and two hidden layers with 3 neurons each were chosen. The sigmoidal function chosen was $g(x) = -1 + 2/(1 + e^{-2x})$. Figure 4 shows the training result after only three cycles. From the figure it is clear that the recurrent neural network can predict the steady-state average resultant cutting force very accurately. To check the generalization capability, the recurrent neural network was tested with the twenty five training pairs left out in the last training set. From Fig. 5 it is clear that the recurrent neural network can generalize well.

4. CONCLUDING REMARKS

A recurrent neural network was successfully trained to learn the functional relationship between the feed rate and steady-state average resultant cutting force. The neural network scheme presented has faster learning capability and is well suited for on-line applications. System identification and control of the cutting force in three-dimensional end milling operations are currently being pursued.

ACKNOWLEDGEMENTS

The authors are pleased to acknowledge the finacial support of the National Science Foundation under Grant Number MSS-9216479 and the Missouri Department of Economic Development through the Manufacturing Research and Training Center-UMR. Also, the authors would like to thank Alan Su and Ilyoung Ahn for their assistance with the experimental work.

REFERENCES

1. Kline, W. A., DeVor, R. E. and Lindberg, J. R., "The Prediction of Cutting Forces in End Milling with Application to Cornering Cuts," *Int. J. on Machine Tool Design Research*, Vol. 22, No. 1, pp. 7-22, 1982.

2. Fussell, B. K. and Srinivasan, K., "An Investigation of the End Milling Process Under Varying Machining Conditions," *ASME J. of Engineering for Industry*, Vol. 111, pp. 27-36, Feb. 1989.

3. Kolarits, F. M. and DeVries, W. R., "A Mechanistic Dynamic Model of End Milling for Process Controller Simulation," *ASME J. of Engineering for Industry*, Vol. 113, pp. 176-183, May 1991.

4. Lauderbaugh, L. K. and Ulsoy, A. G., "Dynamic Modeling for Control of the Milling Process," *ASME J. of Engineering for Industry*, Vol. 110, pp. 367-375, Nov. 1988.

5. Lauderbaugh, L. K. and Ulsoy, A. G., "Model Reference Adaptive Force Control in Milling," *ASME J. of Engineering for Industry*, Vol. 111, pp. 13-21, Feb. 1989.

6. Olgac, N. and Guttermuth, J. R., "A Simplified Identification Method for Autoregressive Models for Cutting Force Dynamics," *ASME J. of Engineering for Industry*, Vol. 110, pp. 288-296, Aug. 1988.

7. Fassois, S. D., Eman, K. F. and Wu, S. M., "A Fast Algorithm for On-Line Machining Process Modeling and Adaptive Control," *ASME J. of Engineering for Industry*, Vol. 111, pp. 133-139, May 1989.

8. Narendra, K. S. and Parthasarathy, K., "Identification and Control of Dynamical Systems Using Neural Networks," *IEEE Trans. on Neural Networks*, Vol. 1, No. 1, pp. 4-27, March 1990.

9. Pineda, F. J., "Dynamics and Architecture for Neural Computation," *J. of Complexity*, Vol. 4, pp. 216-245, 1988.

10. Pearlmutter, B. A., "Learning State Space Trajectories in Recurrent Neural Networks," *Neural Computation*, Vol. 1, pp. 263-269, 1989.

11. Karakaşoğlu, A., Sudharsanan, S. I. and Sundareshan, M. K., "Identification and Decentralised Adaptive Control Using Dynamical Neural Networks with Application to Robotic Manipulators," *IEEE Trans. on Neural Networks*, Nov. 1993.

12. Puskorius, G. V. and Feldkamp, L. A., "Model Reference Adaptive Control with Recurrent Networks Trained by the Dynamic DEKF Algorithm," *Proc. Int. Joint Conf. on Neural Networks*, pp. II-106 - II-113, June 1992.

13. Ku, C.-C. and Lee, K. Y., "Nonlinear System Identification Using Diagonal Recurrent Neural Networks," *Proc. Int. Joint Conf. on Neural Networks*, pp. III-839 - III-844, 1992.

14. Xu, Q., Krishnamurthy, K., McMillin, B. and Lu, W., "A Recursive Least Squares Training Algorithm for Multilayer Recurrent Neural Networks," *Dept. of Mech. and Aero. Eng. and Eng. Mech., University of Missouri-Rolla, Tech. Memo. 28*, Aug. 1993.

15. Steck, J. E., McMillin, B., Krishnamurthy, K., and Leininger, G., "Parallel Implementation of a Recursive Least Squares Neural Network Training Method on the Intel iPSC/2," *J. of Parallel and Distributed Computing*, Vol. 18, pp. 89-93, 1993.

16. Govind, G. and Ramamoorthy, P. A., "An Adaptive-Topology Neural Architecture and Algorithm for Nonlinear System Identification," *Proc. IEEE Int. Conf. on Neural Networks*, pp. 1301-1306, 1993.