J. Paulo Davim
Editor

Machining

Fundamentals and Recent Advances

Springer

Machining

J. Paulo Davim
Editor

Machining

Fundamentals and Recent Advances

Springer

J. Paulo Davim
Department of Mechanical Engineering
University of Aveiro
Campus Universitário de Santiago
3810-193 Aveiro
Portugal

ISBN 978-1-84800-212-8 e-ISBN 978-1-84800-213-5

DOI 10.1007/978-1-84800-213-5

British Library Cataloguing in Publication Data
Machining : fundamentals and recent advances
1. Machining
I. Davim, J. Paulo
671.3'5
ISBN-13: 9781848002128

Library of Congress Control Number: 2008931573

Cover design: eStudio Calamar S.L., Girona, Spain

Printed on acid-free paper

9 8 7 6 5 4 3 2 1

springer.com

Preface

Machining is the broad term used to describe the removal of material from a workpiece and is one of the most important manufacturing processes. Parts manufactured by other processes often require further operations before the product is ready for application. Machining operations can be applied to work metallic and non-metallic materials such as polymers, wood, ceramics, composites and exotic materials.

M. E. Merchant has written *"Today in industrialized countries, the cost of machining amounts to more than 15% of the value of all manufactured products in those countries."* For this reason and others, machining as part of manufacturing science and technology is very important for modern manufacturing industries.

This book aims to provide the fundamentals and the recent advances in machining for modern manufacturing engineering.

The first three chapters of the book provide the fundamentals of machining with special emphasis on three important aspects: metal cutting mechanics (finite element modelling), tools (geometry and material) and workpiece surface integrity.

The remaining chapters of the book are dedicated to recent advances in machining, namely, machining of hard material, machining of particulate-reinforced metal matrix composites, drilling polymeric matrix composites, ecological machining (near-dry machining), sculptured surface machining, grinding technology and new grinding wheels, micro- and nanomachining, advanced (non-traditional) machining processes and intelligent machining (computational methods and optimization).

The present book can be used as a textbook for a final undergraduate engineering course or for a course on machining at the postgraduate level. However, in general, this textbook can be used for teaching modern manufacturing engineering. It can also serve as a useful reference for academics, manufacturing and materials researchers, manufacturing and mechanical engineers, and professionals in machining and related industries. The scientific interest of this book is evident from the many important centres of research, laboratories and universities in the

world working in this area. Therefore, it is hoped that this book will inspire and enthuse further research in this field of science and technology.

I would like to thank Springer for the opportunity to publish this book and for their competent and professional support. Finally, I would like to thank all the chapter authors for making their work availability for this book.

University of Aveiro, Portugal, *J. Paulo Davim*
December 2007

Contents

12

Intelligent Machining: Computational Methods and Optimization

Sankha Deb and U.S. Dixit

Department of Mechanical Engineering, Indian Institute of Technology Guwahati, Guwahati-781 039, Assam, India.
E-mail: sankha.deb@iitg.ernet.in, uday@iitg.ernet.in

This chapter provides an introduction to intelligent machining. The various computational techniques to achieve the goal of intelligent machining are described. First, a description of neural networks and fuzzy set theory is presented. These are soft computing techniques. Afterwards the application of the finite element method to the machining processes is briefly mentioned. Finally, the optimization of machining processes is described.

12.1 Intelligent Machining

Machining processes are inherently complex, nonlinear, multivariate and often subjected to various unknown external disturbances. A machining process is usually performed by a skilled operator, who uses his decision-making capabilities based on the intuition and rules of thumb gained from experience. To develop a fully automated machining system with intelligent functions like those of expert operators is a very difficult problem. A fully automated system requires the capabilities for automatic control, monitoring and diagnostics of the machining processes. Such a system can be compared with a human operator. A number of sensors provide feedback to the system, in a way the sensory organs provide feedback to a human. Like the brain of a human, an automated system is equipped with a computer for processing the feedback information from sensors in real time and taking the appropriate decision to ensure optimal operating conditions. For the decision making, an automated system must have a model of the machining process. The model can be based on the physics of the process. The physics of the process is understood by a researcher and is converted in mathematical form, usually in the form of differential equations. The differential equations of the process are solved by a suitable technique such as the finite element method (FEM). In many instances, the physics of the process is not known properly. However, if there is a sufficient amount of

data describing the behaviour of the process, a model can be developed based on the data, which is called modelling based on the data.

Modelling based on the physics is accomplished by the computational tools that may be called hard computing methods. On the other hand, modelling based on the data is accomplished by soft computing tools such as neural networks and fuzzy sets. Soft computing tools try to generate approximate solutions of the problem in the presence of uncertain or imprecise physics and/or the process variables. Soft computing differs from conventional (hard) computing in that, unlike hard computing, it is tolerant of imprecision, uncertainty, partial truth and approximation. As for modelling, the process optimization can also be carried out by using either conventional (hard) optimization algorithm or heuristic soft algorithm such as genetic algorithms (GAs). Similarly, the control of the process may be based on either conventional (hard) control techniques or soft control techniques such as fuzzy logic. A hybrid of hard and soft techniques may also be used.

Human beings themselves adapt to changes in the environment. In the same way, an adaptive control mechanism is an integral part of an intelligent machine. An adaptively controlled machine is able to adapt to the dynamic changes of the system caused by the variability of machining process due to changes in the cutting conditions such as the hardness of the work material, tool wear, deflection of the tool and the workpiece, and so on. The following are the main objectives of an adaptive control system:

- to adjust the machining parameters such as cutting speeds and feed rates and/or the motion of the cutter to optimize the machining process by maximizing some performance criteria based on the cost or the production;
- to satisfy various constraints against variations due to external factors and respond to such variations in the process in real time;
- to automatically improve the performance of the machining process through its learning capability.

An adaptive control system that can fulfil the third objective may be said to possess intelligence as one of the meanings of the word ‘intelligence’ is the capacity to acquire and apply knowledge.

For monitoring of machining processes and fault diagnostics, the intelligent machining system must be equipped with the knowledge of how to recognise failures, how to localize them and how to relate the faults and their effects to the operating state. Moreover, state classification and process intervention have to be completed in real time to avoid additional damage when a deviant state has been detected. There are various signals (force, torque, temperature, mechanical vibration, acoustic emission, *etc.*) which correlate with the condition of the machining process. The control and monitoring algorithms should be based on the simultaneous measurement and processing of different signals.

Figure 12.1 shows a scheme of an intelligent machining centre. The intelligent machining centre functions as follows. The design model of the component to be machined is provided to the computer. Using suitable software, the tool path is generated and the spindle speed and feed rate are decided. This information is fed to the machine controller, which provides signals to the drive motors. The adap-

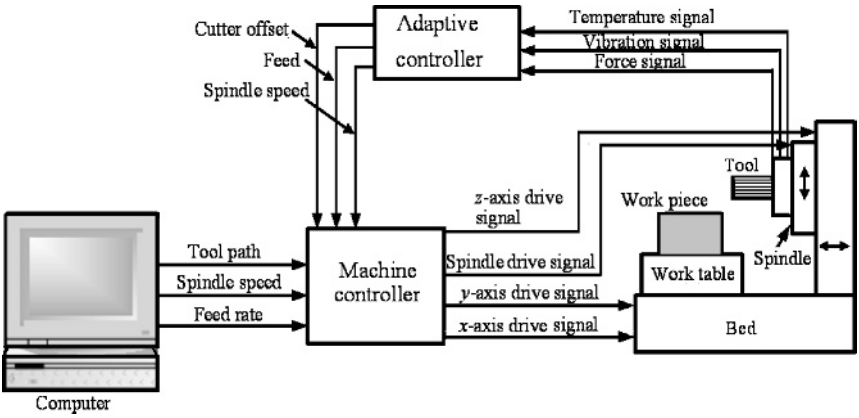

Figure 12.1. An intelligent horizontal machining centre

tive controller adjusts the feed rate, spindle speed and tool path, according to changes in the cutting conditions. The four different types of adaptive control system are: adaptive control optimization (ACO), adaptive control constraint (ACC), geometric adaptive control (GAC) and vibration adaptive control (VAC). The purpose of ACO is to search for the optimum values of feed rate and spindle speed that maximize some performance criteria such as minimum cost of machining or maximum production rate. ACC selects the feasible solution of the feed rate and spindle speed to satisfy the constraints in manufacturing by an algorithm that is rather simpler than ACO. GAC tries to obtain a highly accurate surface finish by adjusting the tool offset against the deflection of tools and work material caused by temperature rise and/or cutting force. The function of VAC is to avoid the vibration or chatter of tools, mainly due to regenerative oscillation, by adjusting the spindle speed or the resonance frequency of machine tools. Feedback about the process can be obtained by means of a number of different kinds of sensors, although the figure shows feedback only in the form of temperature, vibration and force signals.

An intelligent machining system should also possess a memory, like an expert operator. The knowledge available in machining data handbooks and that acquired from experience can be stored in the memory of the intelligent machining system. The system should have the capability to acquire data efficiently, store useful data by filtering out the noise and retrieve it efficiently. At the same time, it is desirable that the system is able to communicate with humans in some way, such that its decisions and actions become transparent to the users concerned. This implies that the acquired or available knowledge can be stored in the form of an expert system.

In this chapter, the computational techniques and optimization methods that can be used to develop intelligent machining systems are briefly described. Section 12.2 describes the application of neural networks in modelling the machining processes. Section 12.3 describes fuzzy-set-based modelling. Section 12.4 describes the hybrid system composed of neural networks and fuzzy set theory, either in the form of the mixture or the compound. Section 12.5 briefly discusses the

application of FEM to the study of machining processes. Section 12.6 describes the optimization techniques useful for optimizing the machining processes. Both conventional (hard) and non-conventional (soft) techniques are touched upon. Finally, Section 12.7 discusses the challenges to be met for developing a truly intelligent machine tool.

12.2 Neural Network Modelling

It is well known that computers can perform a number of jobs at a much faster rate compared to human beings. However, there are many other tasks which a human being can perform in a faster and better way. One such task is recognising a face. A human being can recogonise a face quickly, because the information about various attributes of the face is processed parallely in the brain. The brain contains a neural network consisting of a number of interconnected information-processing units called neurons. A schematic drawing of a neural network showing only two neurons and their connections is shown in Figure 12.2. A neuron consists of a cell body called the soma, a number of fibres called dendrites and a single long fibre called the axon. The dendrites receive the electrical signals from the axons of other neurons. The axon transmits the electrical signal from one neuron to other neurons via the dendrites. The connection between the axon and the dendrite is called a synapse. At the synapses, the electrical signals are modulated by different amounts. The synapses release chemical substances that cause changes in the electrical potential of the soma. When the potential reaches its threshold, an electrical pulse called the action potential is sent down through the axon.

Artificial neural networks are a humble attempt to model biological neural networks. Figure 12.3 shows a schematic diagram of an artificial neuron receiving signals (x_i, $i = 1$ to n) from a number of neurons and emitting signals (o_i, $I = 1$ to k) to be transmitted to a number of neurons. In an artificial neural network, the signals are in the form of numerical values rather than electricity. The action of the synapses is simulated by multiplying each input value by a suitable weight w.

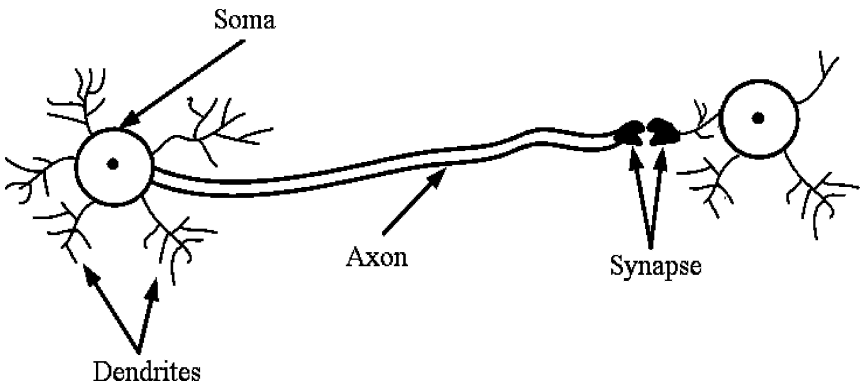

Figure 12.2. A schematic drawing of a biological neural network

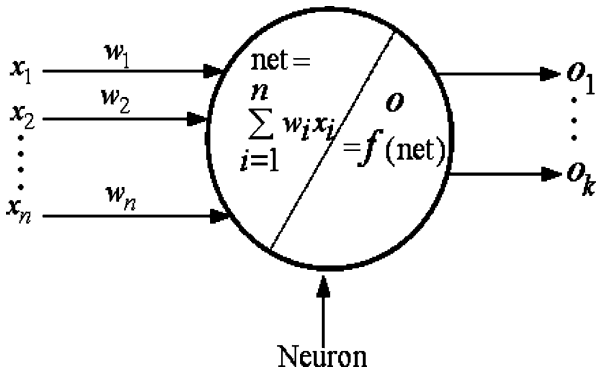

Figure 12.3. A typical artificial neuron

A biological neuron fires an output signal only when the total strength of the input signals exceeds a certain threshold. This phenomenon is modelled in an artificial neuron by calculating the weighted sum of the inputs to represent the total strength of the input signals, and applying a suitable activation (threshold) function to the sum to determine its output.

Neural networks can be classified based on their topology (architecture) and the method of training. The most common neural network architectures are: (1) feedforward neural networks, (2) feedback neural networks, and (3) self-organizing neural networks. Feedforward neural networks are the most popular and widely used ones. There are two types of neural networks in this category: multi-layer perceptron (MLP) neural network and radial basis function (RBF) neural networks. Figure 12.4 shows a feedforward architecture of a typical neural network consisting of three layers. Each layer contains a number of neurons, depicted by circles in the figure. Each layer has full interconnection to the next layer but no connection within a layer. The first layer of the network is known as the input layer whose neurons take on the values corresponding to different variables representing the input pattern. The second layer is known as a hidden layer because its outputs are used internally and not used as the final output of the network. In MLP neural network, there may be more than two hidden layers. In RBF neural network, only one hidden layer is present. In Figure 12.4, only one hidden layer is shown. The final layer of network is known as the output layer. The values of the neurons of the output layer constitute the response of the neural network to an input pattern presented at the input layer. In MLP neural networks, the number of hidden layer in a network is an important design parameter. In both MLP and RBF neural networks the number of neurons in a hidden layer has to be chosen judiciously. The general rule is to design a neural network model which uses fewer parameters (weights and biases).

In recurrent or feedback neural networks, the outputs of some neurons are also fed back to some neurons in layers before them. Thus, signals can flow in both forward and backward directions, as shown in Figure 12.5. A recurrent network is said to have a dynamic memory. The output of such networks at a given instant reflects the current input as well as the previous inputs and outputs. An example of

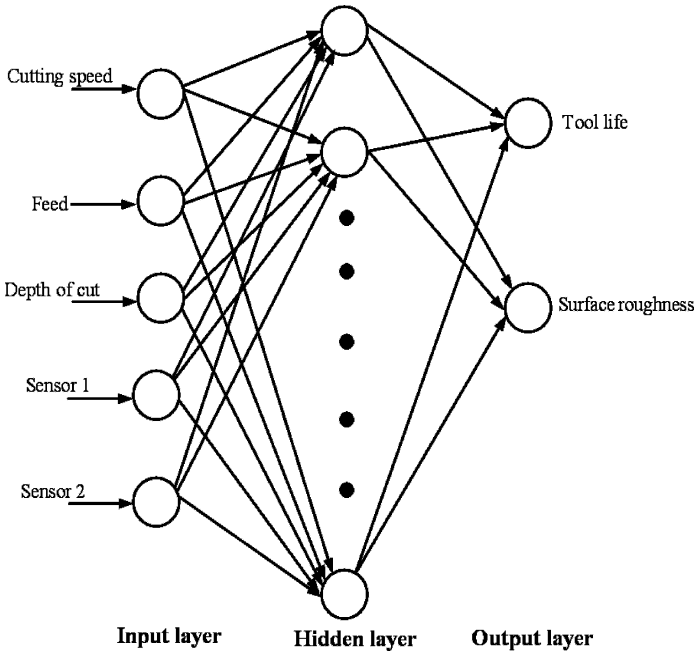

Figure 12.4. A typical feedforward neural network

the recurrent networks is the Hopfield network. Hopfield networks are typically used for classification problems with binary input pattern vectors. Another type of neural network is the self-organizing neural network that consists of neurons arranged in the form of a low-dimensional grid. Each input is connected to all the

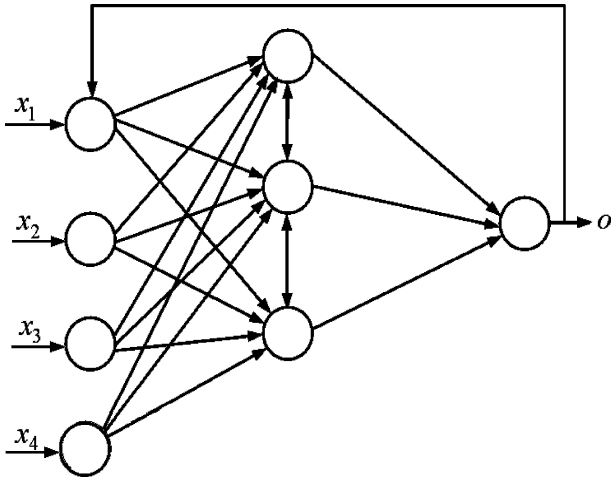

Figure 12.5. A typical recurrent neural netwok

output neurons. This type of network is useful in classifying high-dimensional data by constructing its own topology.

Neural networks need to be trained so that they produce proper response to a given input vector. The training process is an iterative process that adjusts the parameters (weights and biases) of the network until the network is able to produce the desired output from a set of inputs. The process of training the network can be broadly classified into supervised and unsupervised learning. A number of training algorithms based on the supervised learning are available of which the most common is the backpropagation algorithm. The backpropagation algorithm supplies the neural network with a sequence of input patterns and desired output (target) patterns, which together constitute the training exemplars. As an input pattern is presented to the neural network, the output response is calculated on a forward pass through the network. In Figure 12.4, the output of each neuron j in the hidden layer is computed according to the model of Figure 12.3. Given the input signal x_i at the input neurons, the output o_j is given by

$$o_j = f\left(\sum w_{ji} x_i\right), \tag{12.1}$$

where w_{ji} is the weight associated with the jth neuron of the hidden layer and the ith neuron of the input layer. The function f is called the activation function. Some commonly used activation functions are:

$$\textit{Log sigmoid: } o = f(t) = \frac{1}{1+e^{-ct}}, \quad \textit{where } c \textit{ is a constant,} \tag{12.2}$$

$$\textit{Tan hyperbolic: } o = f(t) = \tanh(ct/2), \textit{ where } c \textit{ is a constant,} \tag{12.3}$$

$$\textit{Identity: } f(t) = t. \tag{12.4}$$

The output of each neuron in the output layer is computed in a similar manner. The final output is compared to the desired output, and error terms are calculated for each output neuron. A function of the errors of the neurons in the output layer is then propagated backwards through the network to each layer, and weights of each of the interconnected neurons are adjusted in such a way that the error between the desired output and the actual output is reduced. Any optimization method can be used to find out the weights that minimize the error. Early backpropagation algorithms were based on the steepest-descent algorithm, according to which the maximum decrease in the function is in a direction opposite to the direction of the gradient of the function. Another commonly used backpropagation algorithm is based on the Levenberg–Marquardt method, which is a combination of the steepest-descent method and the quasi-Newton method. At initial iterations, Cauchy's method is followed and the algorithm gradually moves towards the quasi-Newton method.

In back propagation algorithms using the steepest-descent or Levenberg–Marquardt method, the weights may correspond to a local minimum only. This problem can be solved by adding a momentum term to the training rule, which forces the weights to keep moving in the same direction in the error surface without becoming trapped in a local minimum. Momentum simply adds a fraction of

the previous weight update to the current weights. This increases the size of the step taken towards minimum. It is therefore necessary to reduce the learning rate (a factor that determines how much change in the weights is carried out at each step) when using a lot of momentum. If a high learning rate is used, the algorithm may oscillate around the minimum and may become unstable. Too small a learning rate will take a long time to converge. The optimum value of learning rate is often found by trial and error.

A radial basis function neural network has weights only between the hidden layer and the output layer. There are no weights between the input layer and the hidden layer. With each node in the hidden layer, a centre $\boldsymbol{x_k}$ is associated, which is a vector with the same dimension as the input vector. Usually the centres are chosen in a random manner from the input dataset. The output o then becomes

$$o = \sum_{k=1}^{m} w_k g_k \left(|| \boldsymbol{x} - \boldsymbol{x_k} ||, s_k \right), \tag{12.5}$$

where $\boldsymbol{x}$ is the input vector, $|| \boldsymbol{x} - \boldsymbol{x_k} ||$ is the Euclidian norm, s_k is the spread parameter, w_k are weights, $g_k \left(|| \boldsymbol{x} - \boldsymbol{c_k} ||, s_k \right)$ is the radial basis function (RBF) and m is the number of centres, which is the same as the number of neurons in the hidden layer. Some of the most common RBFs are:

$$\textit{Multiquadrics: } g_k(x) = \left(\left\| \boldsymbol{x} - \boldsymbol{x_k} \right\|^2 + s_k^2 \right)^{1/2}; \tag{12.6}$$

$$\textit{Inverse multiquadrics: } g_k(x) = \left(\left\| \boldsymbol{x} - \boldsymbol{x_k} \right\|^2 + s_k^2 \right)^{-1/2}; \tag{12.7}$$

$$\textit{Gaussians: } g_k(\boldsymbol{x}) = e^{-s_k^2 \left\| \boldsymbol{x} - \boldsymbol{x_k} \right\|^2}; \tag{12.8}$$

$$\textit{Thin plate splines: } g_k(x) = \left\| \boldsymbol{x} - \boldsymbol{x_k} \right\|^2 log \left\| \boldsymbol{x} - \boldsymbol{x_k} \right\|. \tag{12.9}$$

Once the type of RBF function is decided and the centres are fixed, the output o given by Equation (12.5) becomes a linear combination of the weights. With the input and output dataset provided, the weights can be calculated using a multiple linear regression procedure so that the sum squared error between the predicted and the target output is minimized.

The process of training the neural network using supervised learning is applicable to problems where representative exemplars of both input pattern and output (target) patterns are known. In many problems where the target patterns are unknown, unsupervised learning is used. In the unsupervised learning process, the network is provided with a dataset containing input patterns but not with desired output patterns. The unsupervised learning algorithm then performs clustering of the data into similar groups based on the measured attributes or features of the given input patterns serving as inputs to the algorithms. After the training of the neural network by a suitable method, the trained network needs to be tested with unseen data (not included in the training dataset).

In the area of machining, neural networks have been used for the prediction of cutting forces [1, 2], surface roughness [3–6], dimensional deviation [3, 4], tool wear [6, 7] and tool life [8, 9]. Risbood *et al.* [4] fitted a neural network for the prediction of surface roughness and dimensional deviation in a turning process. For the wet machining of steel with a high-speed steel (HSS) tool with 8% cobalt, they used only 18 training data and 8 testing data for the prediction of surface roughness. The weight adjustment is carried out by training data and the trained neural network is tested by some data to ensure that there is no overfitting of the network. The multilayer perceptron neural network architecture used by the authors is shown in Figure 12.6. The input neurons correspond to feed f, cutting speed v, depth of cut d and acceleration of radial vibrations a of the turning tool, and the output neuron corresponds to the surface roughness R_a. The accelerations of the radial vibration were measured using an accelerometer. The training and testing data are shown in Table 12.1. While designing the network topology and training, it was ensured that the testing error was below 20% for each data. Later the fitted neural network was validated with 32 validation data. These data were not used in the training and testing. The results are shown in Table 12.2, where R_a is the experimentally obtained surface roughness and $\hat{R}_a$ is the neural network predicted surface roughness. It is seen that, in 24 out of 32 cases, the error in prediction of surface roughness is less than 20%. Only in four cases is the error more than 25%, with a maximum error of 31.9%. Figure 12.7 shows the performance of the neural network pictorially. When a graph is plotted between actual and predicted values of surface roughness, most of the points are found to fall in the neighbourhood of the line passing through the origin and equally inclined from the two axes. Sonar *et al.* [10] fitted a RBF neural network to the data and obtained almost same prediction accuracy in a shorter computational time.

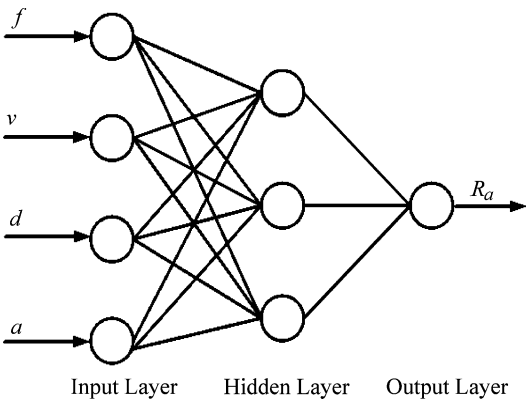

Figure 12.6. The neural network architecture for the prediction of surface roughness From Risbood *et al.* [4] copyright 2003, Elsevier

Table 12.1. Training and testing data for a wet turning operation, from Risbood *et al.* [4] copyright 2003, Elsevier

S. No.	v (m/min)	d (mm)	f (mm/rev)	a (m/s^2)	R_a (μm)
1	107.82	0.30	0.04	0.55	1.74
2	106.47	0.30	0.08	0.65	2.26
3	105.12	0.30	0.16	0.97	3.23
4	104.80	0.60	0.04	2.92	2.74
5	103.55	0.60	0.08	3.66	3.59
6	106.02	0.60	0.16	2.66	2.91
7	47.17	0.30	0.04	0.90	2.31
8	46.55	0.30	0.08	0.73	3.21
9	45.95	0.30	0.16	2.41	4.64
10	48.75	0.60	0.04	0.65	3.18
11	48.14	0.60	0.08	0.58	4.52
12	47.52	0.60	0.16	1.73	5.43
13	27.71	0.30	0.04	0.95	2.06
14	27.35	0.30	0.08	0.83	2.90
15	26.99	0.30	0.16	0.88	5.20
16	27.71	0.60	0.04	0.59	2.87
17	27.35	0.60	0.08	0.72	4.00
18	26.99	0.60	0.16	1.42	6.20
1[a]	74.13	0.40	0.10	2.48	4.80
2[a]	42.98	0.40	0.10	0.67	4.24
3[a]	36.87	0.50	0.12	1.07	4.55
4[a]	35.96	0.30	0.12	0.95	5.21
5[a]	78.10	0.60	0.12	2.10	4.57
6[a]	76.43	0.60	0.05	1.23	2.91
7[a]	73.95	0.30	0.05	0.76	2.52
8[a]	72.92	0.60	0.04	0.79	2.81

[a] test datasets

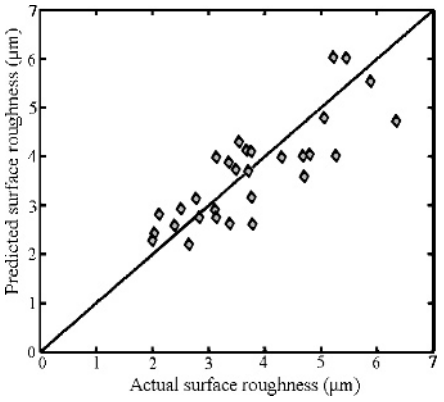

Figure 12.7. Predicted versus actual surface roughness in turning by HSS tool in the presence of coolant. From Risbood *et al.* [4] copyright 2003, Elsevier

Table 12.2. Results of experiments carried out to test the performance of fitted neural network model for the wet turning operation. from Risbood *et al.* [4] copyright 2003, Elsevier

S. no.	F (mm/rev)	v (m/min)	d (mm)	a (m/s^2)	R_a (μm)	$\hat{R}_a$ (μm)	% error
1	0.08	34.71	0.60	0.63	3.67	4.15	–13.10
2	0.04	23.45	0.30	0.42	1.99	2.28	–14.57
3	0.04	64.56	0.40	0.66	2.13	2.81	–31.90
4	0.12	32.87	0.30	0.48	4.29	3.98	7.23
5	0.04	54.28	0.60	0.55	2.78	3.12	–12.23
6	0.16	29.39	0.60	1.12	5.22	6.02	–15.32
7	0.12	20.88	0.40	0.67	4.79	4.04	15.65
8	0.08	54.91	0.40	2.80	3.53	4.33	–22.66
9	0.06	38.50	0.30	0.47	3.37	2.64	21.66
10	0.08	34.71	0.60	0.57	3.73	4.13	–10.72
11	0.04	23.45	0.30	0.52	1.99	2.26	–13.56
12	0.04	64.56	0.40	0.54	2.84	2.75	3.16
13	0.12	32.87	0.30	0.55	5.26	4.03	23.38
14	0.04	54.28	0.60	0.71	3.76	3.17	15.61
15	0.16	29.39	0.60	0.98	5.44	6.02	–10.66
16	0.12	20.88	0.40	0.73	3.76	4.09	–8.77
17	0.08	54.91	0.40	0.73	3.48	3.76	–8.05
18	0.06	38.50	0.30	0.56	3.77	2.63	30.23
19	0.04	48.70	0.40	0.54	2.39	2.56	–7.11
20	0.08	61.88	0.50	0.98	4.68	3.99	14.74
21	0.08	60.56	0.50	0.77	3.14	4.00	–27.38
22	0.16	100.85	0.50	1.49	3.11	2.90	6.75
23	0.04	26.48	0.40	0.37	2.03	2.41	–18.72
24	0.06	49.25	0.30	0.86	3.13	2.78	11.18
25	0.04	106.34	0.30	0.94	2.65	2.20	16.98
26	0.06	45.95	0.60	1.16	3.70	3.70	0.00
27	0.06	101.80	0.60	2.70	2.51	2.93	–16.73
28	0.16	45.99	0.30	1.15	5.87	5.54	5.62
29	0.16	104.20	0.30	2.23	3.36	3.89	–15.77
30	0.16	45.95	0.60	2.28	5.06	4.80	5.14
31	0.16	46.97	0.60	2.43	6.35	4.74	25.35
32	0.16	103.10	0.60	3.32	4.72	3.59	23.94

12.3 Fuzzy Set Theory

In the conventional crisp set theory, an element is either a member of the set or a non-member of the set. With each member, we can associate a number 1 or 0,

depending on whether it is a member or non-member of the particular set. We may call this number the membership grade of the element in the set. In crisp set theory, the membership grades of the elements contained in the set are 1 and the membership grades of the elements not contained in the set are 0. In fuzzy set theory, the value of the membership grade of an element of the universe may have any value in the closed interval [0, 1]. A membership grade 1 indicates full membership and 0 indicates full non-membership in the set. Any other membership grade between 0 and 1 indicates partial membership of the element in the set.

Let us consider an example to understand the concept of crisp and fuzzy set. Assume that there are five machined shafts designated by *a*, *b*, *c*, *d* and *e*. The CLA surface roughness values of the machined shafts are 0.6, 0.8, 1.2, 3.0 and 4.1 μm, respectively. If we construct a set *A* as the set of shafts having surface roughness less than 1.5 μm, then clearly the shafts *a*, *b* and *c* are the members of the set and set *A* may be represented as

$$A = \{a, b, c\}. \tag{12.10}$$

Now, let us construct another set *B* of shafts having low surface roughness. There is a degree of subjectivity in deciding the definition of low surface roughness in a particular context. One possible fuzzy set *B* may be

$$B = \{1/a, 0.8/b, 0.5/c, 0/d, 0/e\}, \tag{12.11}$$

In the above set, the shafts have membership grades of 1, 0.8, 0.5, 0 and 0 respectively, indicated before an oblique slash with each shaft. Here, shaft *a* is a full member of the set *B*, shafts *b* and *c* are partial members, and shafts *d* and *e* are non-members. Someone else may wish to form the set *B* as

$$B = \{1/a, 0.9/b, 0.6/c, 0/d, 0/e\}, \tag{12.12}$$

Here, the membership grades of *b* and *c* have changed; nevertheless the membership grade of *b* is more than that of *c*. Thus, although the membership grades are subjective, they are not arbitrary. However, some skill is needed in the formation of a fuzzy set that properly represents the linguistic name assigned to the fuzzy set.

Fuzzy set theory may be called a generalization of conventional crisp set theory. Various set-theoretic operations commonly used in crisp set theory have been defined for fuzzy set theory as well. These operations reduce to their conventional forms for crisp sets. For example, the intersection of two fuzzy sets *A* and *B*, *i.e.,* $A \cap B$ is defined as a set in which each element has a membership grade equal to the minimum of its membership grades in *A* and *B*. Similarly, the union of two fuzzy sets *A* and *B*, *i.e.*, $A \cup B$ is defined as the set in which each element has a membership grade equal to the maximum of its membership grades in *A* and *B*.

The application of fuzzy sets extends to logic. In the classical binary logic, a statement is either true or false. Quantitatively, we can say that the truth value of a statement is either 1 or 0. In fuzzy logic, it is possible for a statement to have any truth value in the closed interval [0, 1]. For example, the statement "The CLA surface roughness value of 1.2 μm is a low surface roughness." may be assigned a truth value of 0.7 by some expert.

Another offshoot of fuzzy set theory is fuzzy arithmetic. Fuzzy arithmetic deals with operations on fuzzy numbers. The fuzzy numbers are generalization of interval numbers. An interval number is specified by an upper and a lower bound. For example, (3, 5) is an interval number with a lower bound of 3 and upper bound of 5. The interval numbers are used when the value of a variable is expected to lie in a range. A fuzzy number consists of the different interval numbers at different membership grades. A requirement of the fuzzy number is that the membership grade function of fuzzy number should be convex. Figure 12.8 shows examples of valid and invalid fuzzy numbers.

The fuzzy arithmetic operations are defined over each α-cut. An α-cut of a membership grade function $\mu(x)$ is the set of all x such that $\mu(x)$ is greater or

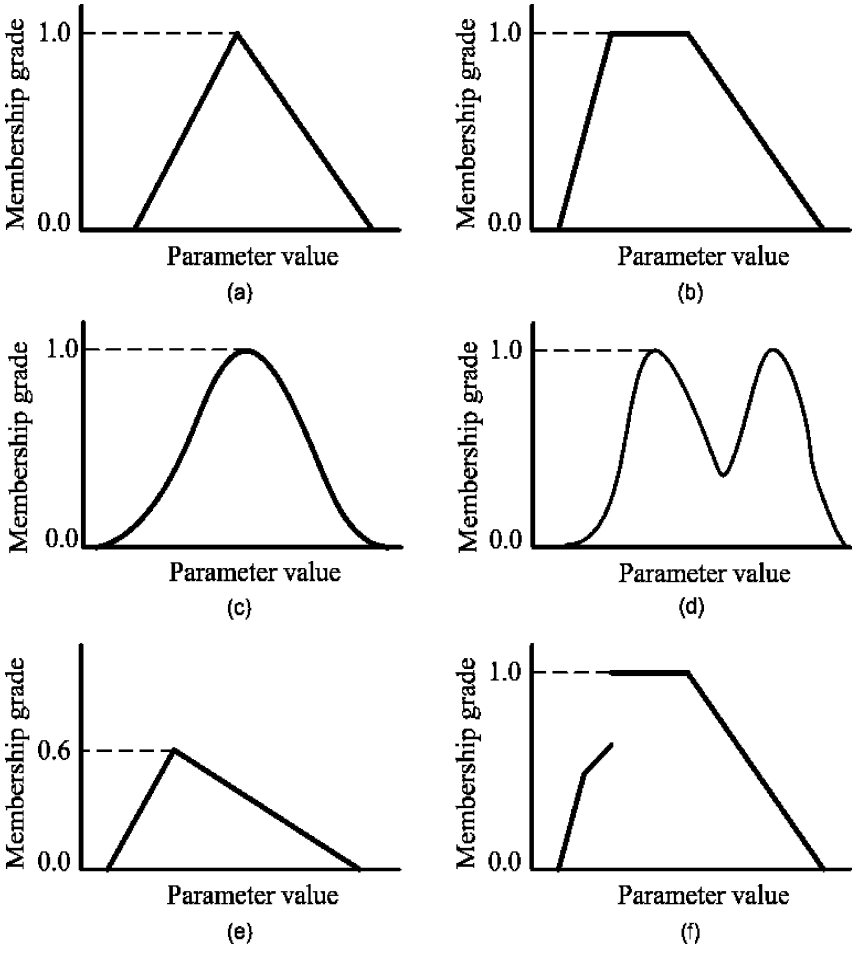

Figure 12.8. The valid and invalid fuzzy numbers: (a) a valid triangular fuzzy number, (b) a valid trapezoidal fuzzy number, (c) a valid bell-shaped fuzzy number, (d) a non-convex invalid fuzzy number (e), a non-normal invalid fuzzy number and (f) a discontinuous invalid fuzzy number

equal to α. Thus, at a particular α-cut, an interval number is obtained corresponding to the interval number at the membership grade of α. If two fuzzy numbers are represented by (a_1^α, a_2^α) and (b_1^α, b_2^α) at an α-cut, then four basic arithmetic operations are defined as follows:

$$\textit{Addition: } \left(a_1^\alpha, a_2^\alpha\right) + \left(b_1^\alpha, b_2^\alpha\right) = \left(a_1^\alpha + b_1^\alpha, a_2^\alpha + b_2^\alpha\right), \tag{12.13a}$$

$$\textit{Subtraction: } \left(a_1^\alpha, a_2^\alpha\right) - \left(b_1^\alpha, b_2^\alpha\right) = \left(a_1^\alpha - b_2^\alpha, a_2^\alpha - b_1^\alpha\right), \tag{12.13b}$$

$$\textit{Multiplication: } \left(a_1^\alpha, a_2^\alpha\right) \times \left(b_1^\alpha, b_2^\alpha\right) = \left(a_1^\alpha \times b_1^\alpha, a_2^\alpha \times b_2^\alpha\right), \tag{12.13c}$$

$$\textit{Division: } \left(a_1^\alpha, a_2^\alpha\right) \div \left(b_1^\alpha, b_2^\alpha\right) = \left(a_1^\alpha \div b_2^\alpha, a_2^\alpha \div b_1^\alpha\right). \tag{12.13d}$$

Other arithmetic operations can be derived from these four basic operations. The fuzzy arithmetic operations can be used to obtain the fuzzy value of the expression. Care must be taken when a variable occurs more than once in an expression. In this case, the blind application of fuzzy arithmetic will provide wider than realistic interval of the value of expression at an α-cut. One way to avoid this problem is to carry out one computation by taking the lower bound, and the other computation by taking the upper bound of the variable at each α-cut.

In many situations, a fuzzy parameter is estimated by a computer code rather than by a closed-form expression. In this case, at each α-cut, the upper bound of the parameter may be obtained by running the code for those combinations of lower and upper bounds of the independent variables that provide the lower and upper bounds of the parameter. In machining process, a number of imprecise variables are involved that can be represented by fuzzy numbers, and computation can be carried out to get an estimate of the dependent variables in the form of fuzzy numbers. The material parameters and the friction, for example, can be treated as fuzzy numbers in finite element code to predict the cutting forces in the form of the fuzzy numbers. This will provide more insight to a higher level decision-maker than the prediction in the form of the crisp numbers.

The fuzzy set theory can also be used to make prediction from the data. As an example, suppose that, in a finish turning process, the surface roughness of the turned job is to be predicted for a particular feed and cutting speed. The estimation of surface roughness consists of four steps. The first step is fuzzification, in which the crisp values of feed and cutting speed are fuzzified, *i.e.*, they are assigned to linguistic fuzzy sets. A variable may be associated with more than one fuzzy set with different membership grades for different sets. For example, the given feed value may be called "high feed" with a membership grade μ_1 and "low feed" with a membership grade μ_2. Similarly, the cutting speed may be called "low cutting speed" with a membership grade μ_3 and "high cutting speed" with a membership grade μ_4.

Once the independent variables are fuzzified, the next step is carried out. In this step, called the rule evaluation step, the strength of the various rules is evaluated.

Various rules are kept in a rule bank, which may be prepared by experts based on their experience or may be generated from data following systematic procedures [11]. For the present example, the four rules could be:

Rule 1: If the feed is low and the cutting speed is low, then the surface roughness is medium.

Rule 2: If the feed is low and the cutting speed is high, then the surface roughness is low.

Rule 3: If the feed is high and the cutting speed is low, then the surface roughness is high.

Rule 4: If the feed is high and the cutting speed is high, then the surface roughness is medium.

In each rule, the 'if' part is called the antecedent and the 'then' part is called the consequent. The strength of a rule is equal to the truth value of the antecedent. If the antecedent consists of the statements separated by 'and', which is equivalent to intersection operation, the truth value of the antecedent is equal to the minimum of the truth values of each of the statements. For example, consider the first rule. The membership of the given feed value in the fuzzy set 'low feed' is μ_2, as described before. Hence, the truth value of the statement 'the feed is low' is μ_2. Similarly, the membership of the given cutting speed in the fuzzy set 'low cutting speed' is μ_3. As both statements in the antecedent are separated by 'and', the truth value of rule 1 is min(μ_2, μ_3). This is the strength of the rule. If the antecedent consists of the statements separated by 'or', which is equivalent to the union operation, the truth value of the antecedent is equal to the maximum of the truth values of each of the statements. For a given feed and cutting speed, a number of rules are applicable with different strengths. Now, we pay attention to the consequent part of the rules. Let us assume that the strengths of the four rules in the above example are 0.06, 0.14, 0.24 and 0.56. Membership functions for the fuzzy sets "low surface roughness", "medium surface roughness" and "high surface roughness" are shown in Figure 12.9. These fuzzy sets are clipped at the membership grades corresponding to the rule strengths, as shown in Figure 12.10. The clipping is done by slicing off the portion of the membership function having membership grade more than the strength of the rule.

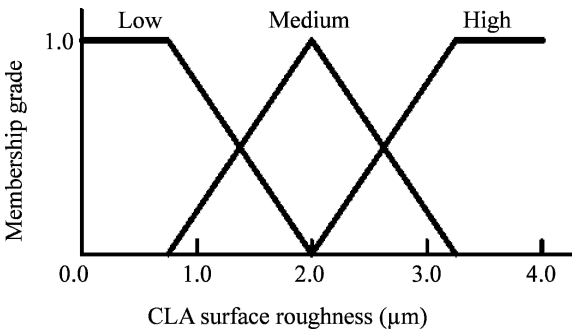

Figure 12.9. Fuzzification of CLA surface roughness

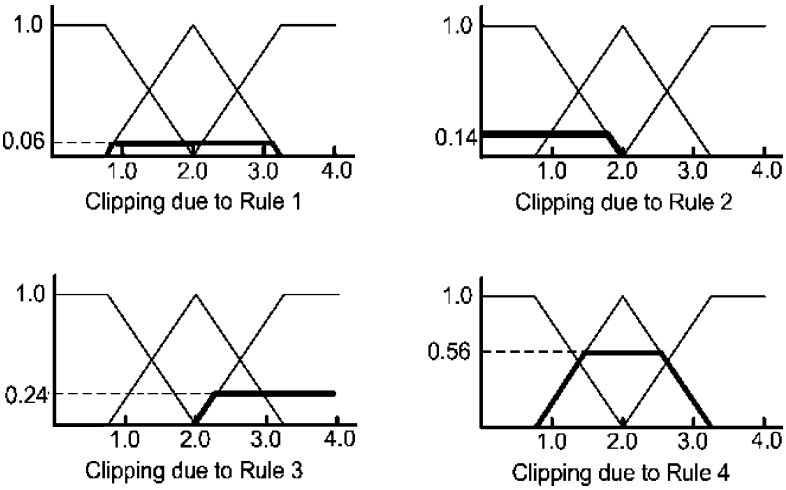

Figure 12.10. Clipping of the fuzzy sets based on the strengths of the rules

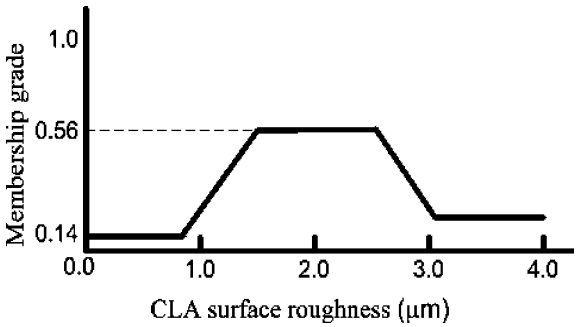

Figure 12.11. Aggregated fuzzy output for prescribed input parameters

The third step is rule aggregation. The clipped rules are aggregated by applying union operation as shown in Figure 12.11. This provides the output *viz.* surface roughness in this case, in the form of a fuzzy variable. This needs to be defuzzified in the fourth step. There are various methods of defuzzification. One method is finding out the centroid of the area covered by the membership function of the aggregated output. The defuzzified output corresponds to the horizontal coordinate of the centroid. Another simpler method is to take the output as the mean of the outputs at the maximum membership grade. In this way, the surface roughness can be predicted for a given set of input variables.

12.4 Neuro-fuzzy Modelling

Neural networks have a good learning capability. However, they act as black boxes. Fuzzy-set-based systems can incorporate expert knowledge and are gener-

ally transparent. Often, it is convenient to apply both these techniques together. There are two ways in which both these techniques can join hands. One way is to have two separate modules for the neural network and fuzzy set, and use these modules in the main task appropriately. For example, for predicting the surface roughness in a turning process, Abburi and Dixit [12] trained a neural network by using shop floor data. The trained network was used to generate a large number of predicted datasets. These datasets were used to feed a fuzzy-set-based rule-generation module. The generated rules were in the form of IF–THEN rules, providing transparency to a user. The developed rule base was used for predicting the surface roughness by using a fuzzy inference system. The second way is to have an entirely different technique which uses the features of both the fuzzy set and the neural network, but can be classified into neither category. One such technique is the adaptive neuro-fuzzy inference system (ANFIS) [13], which can be considered a child of neural network and fuzzy logic. In this section, we briefly describe the ANFIS.

A typical architecture of ANFIS having two inputs as feed f and cutting speed v is shown in Figure 12.12. The input data is fuzzified in the first layer. Each neuron in this layer represents a linguistic fuzzy set such as "small feed", "large depth", *etc.* Each neuron takes the value of an input and emits a membership grade corresponding to that input. Thus, each neuron houses a membership function. The membership function may be fixed based on the expert knowledge, or may contain some adjustable parameters called premise parameters.

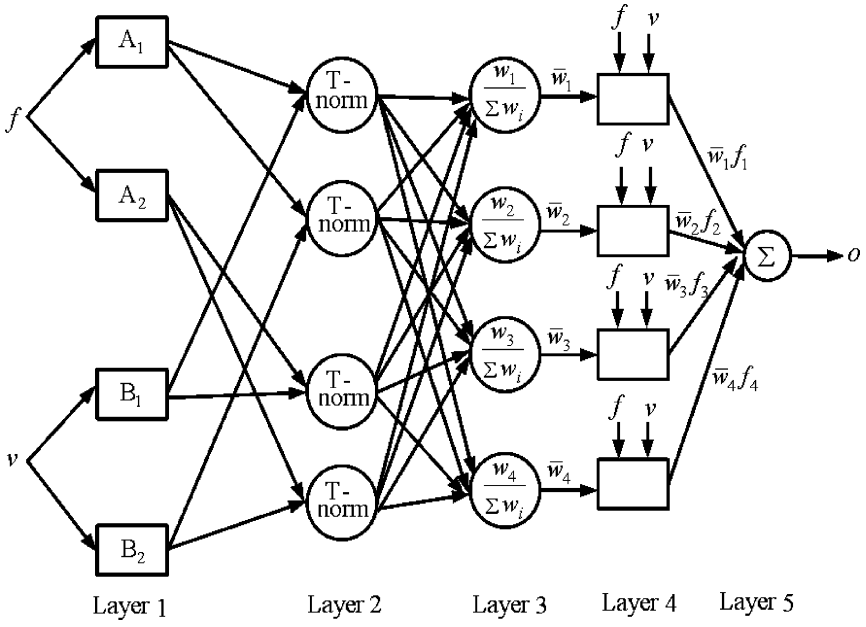

Figure 12.12. A typical ANFIS architecture

Layer 2 of the ANFIS contains neurons which basically emit the strength of various rules. The number of neurons in this layer will be equal to the number of fuzzy rules. The input to each neuron is the membership grades of feed and depth of cut. In the previous section, we mentioned that the strength of a rule may be evaluated by the "minimum" operator. Thus, if the feed value has a membership grade of 0.7 in the fuzzy set "large feed" and the depth of cut has a membership grade of 0.6 in the fuzzy set "large depth of cut", then the strength of the rule having the IF part of "If feed is large and depth of cut is large" can be considered as min(0.7, 0.6), *i.e.*, 0.6. Note that it is not the only way to calculate the strength of a rule containing two statements connected by an "and". In general any T-norm operator can be used to represent fuzzy intersection or "and" in the English language. Some of the most frequently used T-norm operators are:

$$\textit{Minimum: } T_{min}(\mu_1, \mu_2) = min(\mu_1, \mu_2), \tag{12.14a}$$

$$\textit{Algebraic product: } T_{ap}(\mu_1, \mu_2) = \mu_1 \mu_2, \tag{12.14b}$$

$$\textit{Bounded sum: } T_{bs}(\mu_1, \mu_2) = max(0, \mu_1 + \mu_2 - 1), \tag{12.14c}$$

$$\textit{Drastic product: } T_{dp}(\mu_1, \mu_2) = \begin{cases} \mu_1, \textit{ if } \mu_2 = 1, \\ \mu_2, \textit{ if } \mu_1 = 1, \\ 0 \quad \textit{otherwise.} \end{cases} \tag{12.14d}$$

Layer 3 normalizes the strength of the rules. Each neuron in layer 4 emits the weighted output corresponding to a rule. The output of the *i*th fuzzy rule is taken as

$$f_i = p_i f + q_i v + r_i, \tag{12.15}$$

where p_i, q_i and r_i are the adjustable parameters called the consequent parameters. Note that the output of the fuzzy rule given by Equation (12.15) is in the form of a function. This type of rule is used in a Sugeno model. The types of fuzzy rules described in the previous section are used in a Mamdani model. Layer 5 consists of a single neuron, which just sums the inputs.

The network training is carried out by adjusting the premise and consequent parameters so as to minimize the error. In the forward pass, the consequent parameters p_i, q_i and r_i can be obtained by a least-squares procedure for fixed premise parameters of the membership functions. In the backward pass, the consequent parameters are fixed and the premise parameters can be obtained by a suitable optimization procedure to minimize the backpropagated error. Apart from the gradient descent algorithm, a number of optimization algorithms may be used for this purpose. The reader may also note that this is not the only possible topology for an ANFIS. For further details of soft computing techniques, the readers may refer to the textbooks [13–15].

12.5 A Note on FEM Modelling

We have discussed some of the soft computing techniques. These techniques require huge amount of data for proper modelling. It is always better to take advantage of the help of the physics of the process of making a model. The physics of the process needs to be expressed in the form of differential equations. The minite element method (FEM) is a tool to solve these differential equations. In this method, the domain is discretized into a number of small elements. The output variable to be determined is approximated by a combination of functions that are continuous inside the elements and possess at least some order of continuity at the interface of the two elements. Each element contains certain points, which are called nodes. The approximating function is expressed in the form of the nodal values. The nodal values of the output variable are obtained in such a manner that the differential equations are best satisfied with the given approximation. The details of the FEM can be found in a number of textbooks [16–18].

In machining, the FEM has been extensively used for the determination of cutting temperatures [19–22]. For determining the temperature distribution in the tool and the workpiece by a control-volume approach, the following heat conduction equation needs to be solved:

$$k\left[\frac{\partial^2 T}{\partial x^2}+\frac{\partial^2 T}{\partial y^2}+\frac{\partial^2 T}{\partial z^2}\right]+\dot{Q}=\rho c\left[\frac{\partial T}{\partial t}+u\frac{\partial T}{\partial x}+v\frac{\partial T}{\partial y}+z\frac{\partial T}{\partial z}\right], \qquad (12.16)$$

where k is the thermal conductivity, T is the temperature, $\dot{Q}$ is the rate of heat generation per unit volume, ρ is the density, c is the specific heat, t is the time and (u, v, w) are the velocity components of a particle at coordinates (x, y, z). The heat generation is due to plastic deformation of the work material and the friction at the tool surfaces.

For obtaining the heat generation due to plastic deformation and also for obtaining the cutting forces, the machining process may be simulated using the finite element method. There are two approaches for modelling: the updated Lagrangian [23] and Eulerian [24] methods. In the updated Lagrangian approach, the motion of each particle is followed, whereas in the Eulerian approach a control volume is chosen to find various quantities of interest at the spatial coordinates. In general, the finite element simulation of machining processes is computationally expensive due to the non-linear nature of the problem. The finite element simulation of machining process is carried out iteratively. In each iteration, a number of linear equations need to be solved. Among the Eulerian and Lagrangian formulation, the latter takes much more time than the former but is able to predict more detailed information such as the residual stresses in machining. The accuracy of the finite element simulation is dependent on the accuracy of the material parameters and friction characteristics. The machining process occurs at high strain rates and temperature. Therefore, the flow stress of the work material need to obtained from deformation tests at high strain rate and temperature. There is always some uncertainty in the determination of flow stress. The uncertainty is greater for the values of the friction at tool–job and tool–chip interface. It therefore becomes more meaningful to carry out finite element simulations with fuzzy parameters. This,

however, further increases the computational time. One way to reduce the computational time is to train a neural network from the data obtained by FEM and use the neural network for prediction of the machining parameters.

Besides the simulation of machining process, the finite element model has been used for the predictions of tool wear and fracture of the cutting tool [25–28]. It has also been used for predicting the integrity (residual stresses, microhardness and microstructure) of machined surfaces [29, 30].

12.6 Machining Optimization

Optimization of machining processes is one of the most widely investigated problems in machining. The objective functions in the machining problems are: (1) minimization of cost of machining, (2) maximization of production rate and (3) maximization of profit rate. A weighted combination of these objectives may be taken, or the problem can be solved as a multi-objective optimization problem. In the machining optimization problem, there are constraints on tool life, surface finish, cutting force, machining power, *etc*. Usually the machining processes are performed in a number of passes, the last pass being the finishing pass and other passes being the rough passes. In a multipass machining process, the cutting speed, feed and depth of cut in each pass are the primary variables. We shall discuss the issues in machining optimization by taking the example of multipass turning.

12.6.1 Objective Functions and Constraints

Consider the multipass turning of a cylindrical work piece of length L, initial diameter D_0 and final diameter D_f. The production time per component is given by

$$T_p = T_{tR} + \frac{t_c T_{tR}}{T} + T_{tF} + \frac{t_c T_{tF}}{T} + T_l + t_{ts}\,, \tag{12.17}$$

where T_{tR} is the total cutting time for rough machining, t_c is the time required for changing a tool, T is the tool life, T_{tf} is the total cutting time for finish machining, T_l is the job loading and unloading time, and t_{ts} is the total tool setting time for a job. The total cutting time for rough machining is obtained as the summation of the cutting time for m roughing passes. Thus,

$$T_{tR} = \sum_{i=1}^{m} \frac{\pi L D_{i-1}}{v_{R_i} f_{R_i}}\,, \tag{12.18}$$

where v_{R_i} and f_{R_i} are the cutting speed and feed, respectively, at the ith roughing pass and D_{i-1} is the workpiece diameter at the beginning of that pass. Note that the cutting speed is defined as the surface speed of the work piece at the beginning of the pass. The total tool setting time is given as

$$t_{ts} = (m+1)t_s\,, \tag{12.19}$$

where t_s is the setting time for each pass.

The tool life T is a critical parameter for the objective function. The tool life is a function of machining parameters, *viz.* feed (f), cutting velocity (v) and depth of cut (d) as well as the tool–job combination and the machine tool. For a given machine tool and tool–job combination, the tool life can be expressed as a function of f, v and d. A neural network can be used for online prediction of the tool life.

The cost of machining for a workpiece, C_p, can be expressed as

$$C_p = C_0 T_p + C_{t_1} \frac{T_{tR}}{T} + C_{t_2} \frac{T_{tF}}{T}, \tag{12.20}$$

where C_0 is the operating cost per unit time, C_{t_1} is the tool change cost for a roughing tool and C_{t_2} is the tool change time for a finishing tool. If the same tool is used for roughing and finishing passes, then $C_{t_1} = C_{t_2}$. If the price of a work piece is fixed as P_p, then the profit rate P_R is expressed as

$$P_R = \frac{P_p - C_p}{T_p}. \tag{12.21}$$

Equations (12.17), (12.20) and (12.21) can be used as the objective functions.

The machining optimization problem may be subjected to various constraints. Some of the constraints may be as follows:

1. Constraint on the tool life: The tool life should not be too low or too high. If the tool life is too low, then it will fail without machining even one piece. While machining a workpiece, if the tool is changed in between, it will affect the surface finish and dimensional accuracy of the workpiece. As a thumb rule, the tool life should be at least such that 20 components can be produced without the need of replacing the tool. In other words, the deterioration in tool life should be limited to less than 5% in machining a component. At first sight, it appears that there is no need to have a constraint on the maximum permissible tool life. However, a close examination of shop floor practice reveals that there is no advantage in operating at parameters providing too high value of tool life. Most factories follow the policy of replacing tools after a certain interval of time. If the tool life is too high, tools may be underutilized. Also, if certain types of tools have been specifically procured for a particular batch production, often it may not be advantageous to preserve the tools once the entire batch has been produced. In that case also, too high a tool life will mean underutilization of tools. However, in most cases, the constraint on tool life may turn out to be an inactive constraint.
2. Constraint on the surface finish: Most of the time a constraint on the upper limit of surface roughness is prescribed. However, the possibility of having a constraint on the lower limit of the surface roughness cannot be ignored. Surface roughness affects the heat transfer characteristics and a rough surface is beneficial, for example, in order to have greater heat transfer in a pool boiling process. Similarly, a specific tribological characteristic may be obtained by controlling the surface roughness. For the given process pa-

rameters, the surface roughness may be predicted by a trained neural network as described in Section 12.2.

3. Constraints on the machining forces: The neural network can be used for the modelling of cutting forces in a machining process. These forces should be less than the prescribed forces to avoid tool breakage and excessive deflections of the job and the tool.
4. Constraint on the machining power: The machining power is obtained by dividing the product of the main cutting force and cutting speed by the efficiency of the machine tool. The machining power constraint requires that the machining power be less than the prescribed power.
5. Constraints on vibrations: The vibrations during a machining process should be low as they affect the accuracy and precision of the job.
6. Constraint on dimensional deviation: The dimensional deviation of the workpiece should be kept within the prescribed limit. If a model for predicting the dimensional deviation in machining is available, it can be used in this constraint. Otherwise the constraints on machining forces will indirectly impose the constraint on dimensional deviation.
7. Constraints on geometric relations: It is obvious that the number of passes m in a machining operation is an integer quantity. The following geometric relation has to be satisfied for the multi-pass turning process:

$$D_0 = D_f + 2\sum_{i=1}^{m} d_{R_i} + 2d_F \quad , \tag{12.22}$$

where D_f is the diameter of the finished job, d_{R_i} is the depth of cut of the i^{th} roughing pass and d_F is the depth of cut of the finish pass.

Apart from these constraints, there are variable bounds on the cutting speed, feed and depth of cut.

12.6.2 Optimization Techniques

The general form of a single-objective optimization problem is

$$\begin{aligned} &\textit{Minimize } f(\boldsymbol{x}), \\ &\textit{subject to certain constraints.} \end{aligned} \tag{12.23}$$

where $\boldsymbol{x}$ is a vector containing the decision variables. The maximization of a function is equivalent to the minimization of the negative of the function. Thus, even a maximization problem can be expressed in the form of Equation (12.23). There are a plethora of optimization techniques available in the literature [31–33]. We shall provide a glimpse of some techniques.

12.6.2.1 Golden Section Search Method

For one-dimensional minimization, there is an efficient technique for finding the minimum of a unimodal function. A function which has got only one minimum in

a certain interval is called a unimodal function in that interval. Figure 12.13(a) shows a unimodal function. Figure 12.13(b) shows a multi-modal function for comparison. The minimum of a unimodal function can be found by a number of region-elimination methods that require only the function values but not derivative information. The general procedure of a region-elimination method is as follows:

Step 1: Find an interval (a, b) in which the minimum is expected to lie. The interval can be obtained by physical consideration or by a systematic mathematical procedure. The length of the interval should be large enough to ensure the presence of a minimum in the range.

Step 2: Choose two points x_1 and x_2 in the interval (a, b) and evaluate the function $f(x)$ at these points.

Step 3: If $f(x_1) > f(x_2)$, then minimum does not lie in interval (a, x_1), requiring replacement of the interval (a, b) by the interval (x_1, b). Else, if $f(x_2) > f(x_1)$, then minimum does not lie in interval (x_2, b), requiring replacement of the interval (a, b) by the interval (a, x_2). Else, if $f(x_2) = f(x_1)$, the minimum lies in the interval (x_1, x_2), requiring the replacement of the interval (a, b) by the interval (x_1, x_2).

Step 4: If the current interval is not sufficiently small, go to step 2.

It is clear from Figure 12.13(b) that the region-elimination method will not be effective for a multi-modal objective function.

There are a number of region-elimination methods depending on how we choose the points x_1 and x_2. One of the efficient one-dimensional search methods in this category is the golden section search. In this method, for the interval (a, b), two points x_1 and x_2 are chosen as follows:

$$x_1 = \tau a + (1-\tau)b\,, \tag{12.24}$$

$$x_2 = \tau b + (1-\tau)a\,, \tag{12.25}$$

where τ is called a golden number and is equal to 0.618. This procedure ensures that in the subsequent iterations (when the interval gets changed), only one new point has to be chosen. The other point becomes common with a point of the pre-

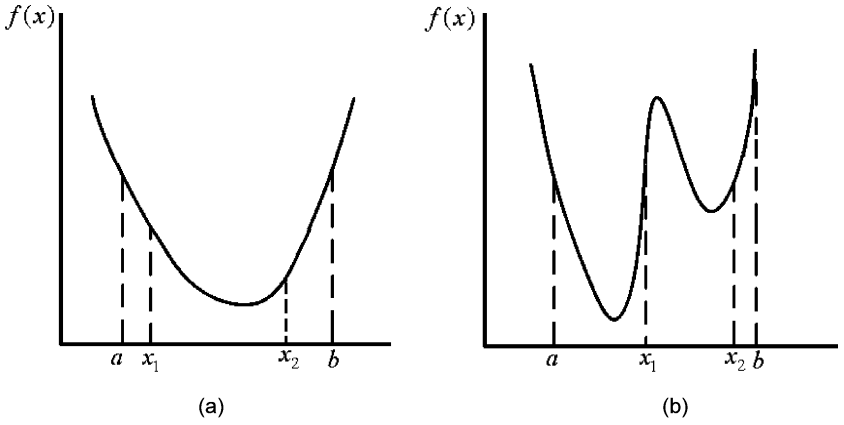

Figure 12.13. (a) A unimodal function. (b) A multimodal function

vious iteration. Thus, at each iteration only one function evaluation is needed. The interval reduces by a factor of $(0.618)^{n-1}$ after n function evaluations. Once the interval has become sufficiently small, the minimum can be taken as the middle point of the interval. For further accuracy, one may take three points in the reduced interval and fit a quadratic function to find the minimum.

12.6.2.2 Sequential Quadratic Programming (SQP) Method

For constrained optimzation of multivariable problems, there are a number of methods, which do not require the gradient information, *viz.* complex search method, random search method, *etc*. These methods are called direct search methods. However, we discuss here a powerful method requiring gradient information. The method is called sequential quadratic programming (SQP) and is quite efficient in handling constrained optimization problems. The principle of this algorithm is explained here.

Consider the following optimization problem with l equality and m inequality constraints:

$$\begin{aligned} &\text{Minimize } f(\boldsymbol{x}), \\ &\text{subject to } h_i(\boldsymbol{x})=0,\ i=1 \text{ to } l;\ g_i(\boldsymbol{x}) \le 0,\ i=1 \text{ to } m \end{aligned} . \tag{12.26}$$

At a particular guess point, Equation (12.26) is converted to the following quadratic form:

$$\begin{aligned} &\text{Minimize } \left(\nabla f(\boldsymbol{x})\right)^T \boldsymbol{d} + \frac{1}{2}\boldsymbol{d}^T \boldsymbol{H}\boldsymbol{d}, \\ &\text{subject to } h_i(\boldsymbol{x}) + \left(\nabla h_i(\boldsymbol{x})\right)^T \boldsymbol{d} = 0,\ i=1 \text{ to } l; \\ &\qquad g_i(\boldsymbol{x}) + \left(\nabla g_i(\boldsymbol{x})\right)^T \boldsymbol{d} \le 0,\ i=1 \text{ to } m. \end{aligned} \tag{12.27}$$

In the above equation, $\boldsymbol{d} = (d_1, d_2, \ldots\ldots, d_n)^T$ is the vector of decision variables of the problem, ∇ is the gradient operator and $\boldsymbol{H}$ is the Hessian matrix. In particular,

$$\left(\nabla f(x)\right)^T = [\partial f/\partial x_1, \partial f/\partial x_2, \ldots\ldots\ldots\ldots, \partial f/\partial x_n] , \tag{12.28}$$

and

$$\boldsymbol{H} = \begin{bmatrix} \dfrac{\partial^2 f}{\partial x_1^2} & \dfrac{\partial^2 f}{\partial x_1 \partial x_2} & \ldots\ldots\ldots & \dfrac{\partial^2 f}{\partial x_1 \partial x_n} \\ \dfrac{\partial^2 f}{\partial x_2 \partial x_1} & \dfrac{\partial^2 f}{\partial x_2^2} & \ldots\ldots\ldots & \dfrac{\partial^2 f}{\partial x_2 \partial x_n} \\ \ldots\ldots & \ldots\ldots & \ldots\ldots & \ldots\ldots \\ \ldots\ldots & \ldots\ldots & \ldots\ldots & \ldots\ldots \\ \ldots\ldots & \ldots\ldots & \ldots\ldots & \ldots\ldots \\ \dfrac{\partial^2 f}{\partial x_n \partial x_1} & \dfrac{\partial^2 f}{\partial x_n^2} & \ldots\ldots\ldots & \dfrac{\partial^2 f}{\partial x_n^2} \end{bmatrix} . \tag{12.29}$$

The gradients and the Hessian matrix are evaluated at the guess point.

The variable $\boldsymbol{d}$ is actually a search direction. It indicates that, moving in this direction, one will get the minimum. The question arises how much we should move. For this, we assume that the minimum point $\boldsymbol{x_m}$ is obtained by the following expression:

$$\boldsymbol{x_m} = \boldsymbol{x} + \lambda \boldsymbol{d} \,, \tag{12.30}$$

where λ is a step length chosen to reduce the value of a suitable merit function. Once the value of $\boldsymbol{x_m}$ is substituted in the merit function, the merit function becomes a function of the scalar λ. The optimum value of λ can be found by a one-dimensional optimization method such as the golden section search method, described in Section 12.6.2.1. The merit function can be a penalty function of the following form:

$$M = f(\boldsymbol{x}) + R\left(\sum_{i=1}^{l} |h_i(x)| + \sum_{i=1}^{m} max(g_i(x), 0)\right), \tag{12.31}$$

where R is a very large value called the penalty parameter.

At the obtained optimum point $\boldsymbol{x_m}$, the objective function is again converted to the form of Equation (12.27), *i.e.*, a quadratic approximation for the function and a linear approximation for the constraints is used. After this, the procedure is repeated. Thus, the optimization process is iterative and the iterations are continued until convergence is obtained.

12.6.2.3 Genetic Algorithms

Conventional optimization methods are not suitable for optimization problems with multiple optimal solutions. There are newer methods, which start the search process with a population of solutions. These methods are inspired by natural processes and a sound mathematical basis for these methods needs to be developed. These methods can find multiple optimum solutions and globally optimum solutions and are suitable for multi-objective optimization problems. One category of such methods is genetic algorithms (GAs). GAs are inspired by the mechanics of natural genetics and natural selection. Although there are a lot of variants of genetic algorithms, we describe here a simple binary-coded genetic algorithm with reference to the optimization of the finish pass turning considering the feed and the cutting speed as decision variables.

In the binary-coded genetic algorithm, the variables are represented in binary form. For example, if feed varies from 0 to 0.15 mm/rev in a turning operation, it can be represented with a 4-bit binary number with 0000 representing zero feed and 1111 representing a feed of 0.15 mm/rev. A finer resolution in feed values may be obtained by choosing a higher bit number. Similarly let us consider that a cutting speed in the range of 0 to 310 m/min is represented by a 5-bit number with 11111 representing the cutting speed of 310 m/min in the usual way. A typical combination of feed and cutting speed can be represented by putting these two binary numbers together to make it a 9-bit string. For example, the reader may observe that 101010000 is a string representing a feed of 0.1 mm/rev and cutting speed of 160 m/min.

The search for the optima in GA is started by taking a number of random strings forming the population. Choosing an appropriate population size is crucial. The population of the strings is successively evolved by three operators: reproduction, crossover and mutation.

In reproduction, the good strings in the population are probabilistically assigned to a large number of copies. The fitness of the string decides how good the string is. In minimization problem, the fitness function can be taken as

$$F(\boldsymbol{x}) = 1/(1 + f(\boldsymbol{x})), \tag{12.32}$$

For infeasible solution, a very low value of fitness may be assigned. There are a number of ways to do reproduction. One popular and easy-to-implement method is the tournament selection. In this, each string plays duel tournaments with two other randomly chosen strings. In a tournament, the fitnesses of the two strings is compared and the winning string is retained. It is clear that the best string will always remain and the worst string will be eliminated from the population. The fate of the other strings is dependent on chance. Finally, it is expected that, as a result of this operation, the number of good strings will be more than the bad ones in the population.

The second operation is crossover. In this, two new strings are created by exchanging the bits between two strings. For example, consider two strings chosen at random from a population, called parent strings. A single-point crossover operation is performed by randomly choosing a crossing site along the string and by exchanging all the bits on the right-hand side of the crossing site as shown;

$$\begin{array}{l} 010001|111 \\ 100110|011 \end{array} \Rightarrow \begin{array}{l} 010001|011 \\ 100110|111 \end{array}$$

The new strings formed are called the children strings. The crossover operation is performed with certain a crossover probability, usually lying between 0.75 to 0.95.

In the last operation, mutation, one or more bit of a string may be changed randomly. However, the probability of mutation is kept quite low, between around 0.01 and 0.05. The mutation operation serves the crucial role of preventing the algorithm from becoming stuck in the local optimum.

After the application of three GA operators in succession, a new generation is formed and an iteration is said to be complete. The iterations are carried out until the average fitness in successive generations more or less becomes constant. The entire methodology is illustrated by a flowchart in Figure 12.14. The application of GA to machining process optimization has been carried out by a number of researchers [34–36].

There are other newer population-based techniques that are proving to match the capabilities of GAs and on occasions outperform GAs. Two such techniques are particle swarm optimization (PSO) that was developed by Eberhert and Kennedy [37] and ant colony optimization (ACO) that was first proposed by Dorigo *et al.* [38]. PSO is a population-based algorithm that simulates the social behaviour of a flock of birds for the search of approximate solutions to optimization problems. The ACO algorithm is a population-based algorithm that simulates the social behaviour of a colony of ants for the search of approximate solutions to optimization

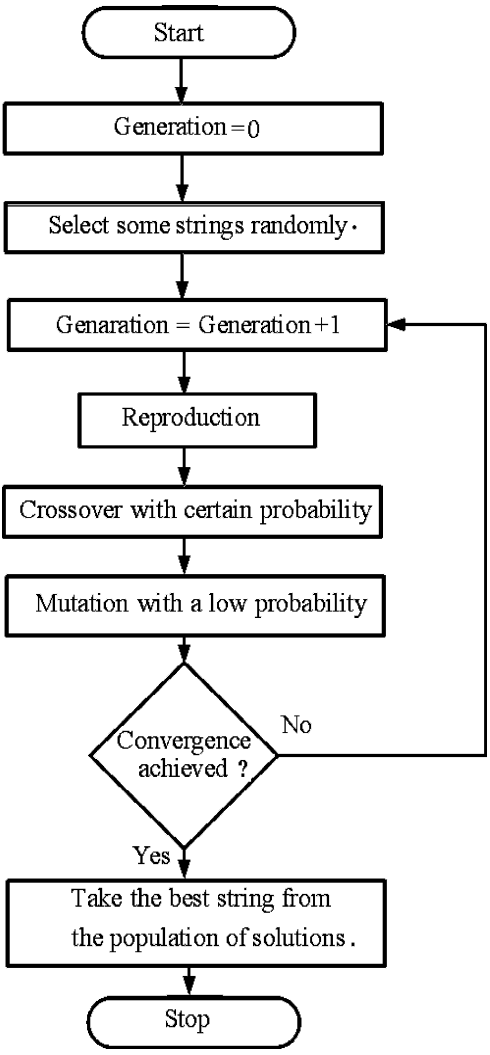

Figure 12.14. A flow chart illustrating the methodology of genetic algorithm

problems. ACO algorithm has been used for the determination of optimal machining parameters such as cutting speed, feed, depth of cut and number of cuts in order to minimize the production cost subject to various machining constraints such as tool life, surface finish, cutting forces and power, and temperatures [39].

12.7 Future Challenges

Earlier in this chapter, we compared the intelligent system of a machine tool with a skilled human operator. The performance of a skilled operator is dependent to

a great extent on the sensory organs and the brain. Therefore, to develop an intelligent system that is able to function like a skilled operator, it is necessary to develop good sensors and efficient computing tools. A lot of research needs to be done to develop effective, compact and low-cost sensors. At the same time, computational methods and optimization algorithms need to be further improved. The compuational methods that model based on the available data need to be made more robust. They should be able to eliminate noise from the data and make the best out of the available information. There should be simultaneous efforts to understand the physics of the process and use that knowledge to complement the data modelling task. The proper formulation of optimization problems for different processes and choosing of an efficient algorithm is another area that needs to be explored further. Lastly, the intelligent system must exploit the developments in Internet technologies, which requires research in the Internet-based machining area.

References

[1] Szecsi T (1999) Cutting force modelling using artificial neural networks. J Mater Process Technol 92:344–349.
[2] Hao, W, Zhu, X, Li, X and Turyagyenda, G (2006), Prediction of cutting force for self-propelled rotary tool using artificial neural networks, Journal of Materials Processing Technology, Vol. 180, pp. 23-29.
[3] Azouzi R, Guillot M (1997) On-line prediction of surface finish and dimensional deviation in turning using neural network based sensor fusion. Int J Mach Tools Manuf 37:1201–1217.
[4] Risbood KA, Dixit US, Sahasrabudhe AD (2003) Prediction of surface roughness and dimensional deviation by measuring cutting forces and vibrations in turning process. J Mater Process Technol 132:203–214.
[5] Kohli A, Dixit US (2004) A neural-network-based methodology for the prediction of surface roughness in a turning process. Int J Adv Manuf Technol 25: 118–129.
[6] Özel T, Karpat Y (2005) Predictive modelling of surface roughness and tool wear in hard turning using regression and neural networks, Int J Mach Tools Manuf 45:467–479.
[7] Ezugwu EO, Arthur SJ, Hines EL (1995) Tool-wear prediction using artificial neural networks. J Mater Process Technol 49: 255–264.
[8] Tosun N, Ozler L (2002) A study of tool life in hot machining using artificial neural networks and regression analysis method. J Mater Process Technol 124:99–104.
[9] Ojha DK, Dixit US (2005) An economic and reliable tool life estimation procedure for turning. Int J Adv Manuf Technol 26: 726–732.
[10] Sonar DK, Dixit US, Ojha DK (2006) Application of radial basis function neural network for predicting the surface roughness in turning process. Int J Adv Manuf Technol 27:661–666.
[11] Chen JC, Black JT (1997), A fuzzy-nets in-process (FNIP) systems for tool-breakage monitoring in end-milling operations. Int J Mach Tools Manuf 37: 783–800.
[12] Abburi NR, US Dixit (2006) A knowledge-based system for the prediction of surface roughness in turning process. Robot Comput-Integr Manuf 22: 363–372.
[13] Jang JSR, Sun CT, Mizutani E (2002) Neuro-Fuzzy and Soft Computing – A computational Approach to Learning and Machine Intelligence. Prentice-Hall of India, New Delhi.

[14] Kosko B (1992) Neural Networks and Fuzzy Systems: a Dynamical Systems Approach to Machine Intelligence. Prentice-Hall, NJ.
[15] Tsoukalas LH, Uhrig RE (1996) Fuzzy and Neural Approaches in Engineering. Wiley, New York, NY.
[16] Reddy JN (1993) An Introduction to the Finite Element Method, second edition. McGraw-Hill, New York, NY.
[17] Cook RD, Malkus DS, Plesha ME (1989) Concepts and Applications of Finite Element Analysis, 3rd edn. Wiley, New York, NY.
[18] Bathe KJ (1996) Finite Element Procedures. Prentice-Hall of India, New Delhi.
[19] Grzesik W (2006), Determination of temperature distribution in the cutting zone using hybrid analytical-FEM technique. Int J Mach Tools Manuf 46: 651–658.
[20] Fang G, Zeng P (2005) Three-dimensional thermo-elastic-plastic coupled FEM simulations for metal oblique cutting processes. J Mater Process Technol 168: 42–48.
[21] Abukhshim NA, Mativenga PT, Sheikh MA (2006) Heat generation and temperature prediction in metal cutting: A review and implications for high speed machining. Int J Mach Tools Manuf 46:782–800.
[22] Tay AOO, Lee KH (1992) Calculation of temperature distribution in machining using a hybrid finite-element-boundary-element method. J Mater Process Technol 29:47–62.
[23] Ozel T (2006) The influence of friction models on finite element simulations of machining. Int J Mach Tools Manuf 46:518–530.
[24] Joshi VS, Dixit PM, Jain VK (1994) Viscoplastic analysis of metal cutting by finite element method. Int J Mach Tools Manuf 34:553–571.
[25] Filice L, Micari F, Settineri L, Umbrello D (2007) Wear modelling in mild steel orthogonal cutting when using uncoated carbide tools. Wear 262: 545–554.
[26] Xie L-J, Schmidt J, Schmidt C, Biesinger F (2005) 2D FEM estimate of tool wear in turning operation. Wear 258: 1479–1490.
[27] Ahmad MM, Draper WA, Derricott RT (1989) An application of the finite element method to the prediction of cutting tool performance. Int J Mach Tools Manuf 29:197–206.
[28] Cakir MC, Sik YI (2005) Finite element analysis of cutting tools prior to fracture in hard turning operations. Mater Des 26:105–112.
[29] Monaghan J, Brazil D (1997) Modelling the sub-surface damage associated with the machining of particle reinforced MMCs. J Comput Mater Sci 9:99–107.
[30] Wen Q, Guo YB, Todd BA (2006) An adaptive FEA method to predict surface quality in hard machining. J Mater Process Technol 173:21–28.
[31] Rao SS (1996) Engineering Optimization – Theory and Practice. Wiley and New Age International (P), New Delhi.
[32] Deb K (2005) Optimization for Engineering Design – Algorithms and Examples. Prentice-Hall of India, New Delhi.
[33] Deb K (2002) Multi-Objective Optimization Using Evolutionary Algorithms. Wiley, Singapore.
[34] Wang X, Da ZJ, Balaji AK, Jawahir IS (2002) Performance-based optimal selection of cutting conditions and cutting tools in multi-pass turning operations using genetic algorithms. Int J Prod Res 40:2053–2065.
[35] Sarvanan R, Asokan P, Vijyakumar K (2003) Machining parameter optimization for turning cylindrical stock into a continuous finished profile using genetic algorithm (GA) and simulated annealing (SA). Int J Adv Manuf Technol 21:1–9.
[36] Abburi NR, Dixit US (2007), Multi-objective optimization of multipass turning processes. Int J Adv Manuf Technol 32:902–910.

[37] Kennedy J, Eberhart R (1995) Particle swarm optimization, Proceedings of the IEEE international conference on neural networks (Perth, Australia), pp. 1942–1948. Piscataway, NJ.
[38] Dorigo M, Maniezzo V, Colorni A (1996) Ant system: optimization by a colony of cooperating agents. IEEE Trans Syst Man Cybern 26:29–41.
[39] Vijayakumar K, Prabhaharan G, Asokan P, Saravanan R (2003) Optimization of multi-pass turning operations using ant colony system. Int J Mach Tools Manuf 43:1633–1639.