J Ind Eng Int (2016) 12:389–400
DOI 10.1007/s40092-016-0146-x

TECHNICAL ARTICLE

On the use of back propagation and radial basis function neural networks in surface roughness prediction

Angelos P. Markopoulos[1] · Sotirios Georgiopoulos[1] · Dimitrios E. Manolakos[1]

Received: 30 January 2015 / Accepted: 23 February 2016 / Published online: 9 March 2016

Abstract Various artificial neural networks types are examined and compared for the prediction of surface roughness in manufacturing technology. The aim of the study is to evaluate different kinds of neural networks and observe their performance and applicability on the same problem. More specifically, feed-forward artificial neural networks are trained with three different back propagation algorithms, namely the adaptive back propagation algorithm of the steepest descent with the use of momentum term, the back propagation Levenberg–Marquardt algorithm and the back propagation Bayesian algorithm. Moreover, radial basis function neural networks are examined. All the aforementioned algorithms are used for the prediction of surface roughness in milling, trained with the same input parameters and output data so that they can be compared. The advantages and disadvantages, in terms of the quality of the results, computational cost and time are identified. An algorithm for the selection of the spread constant is applied and tests are performed for the determination of the neural network with the best performance. The finally selected neural networks can satisfactorily predict the quality of the manufacturing process performed, through simulation and input–output surfaces for combinations of the input data, which correspond to milling cutting conditions.

Keywords Artificial neural networks · Training algorithms · Radial basis function · Surface roughness · Milling

Introduction

Computational methods for modeling and simulation are significant in contemporary manufacturing sector, where quality plays a key role. The industrial and academic interest in the role of computers in manufacturing technology is increasing since modeling and simulation are able to lead to optimization of processes and at the same time reduce expensive and time consuming experimental work. This is especially applicable for material removal processes that are involved in numerous final products that demand high quality, which usually is quantified through surface roughness of the final product; surface roughness influences several attributes of a part such as fatigue behaviour, wear, corrosion, lubrication and surface friction.

Modelling and simulation techniques that are more popular for the analysis of manufacturing processes are the Finite Element Method (FEM), soft computing techniques including Artificial Neural Networks (ANNs) (Galanis and Manolakos 2014; Szabó and Kundrák 2014; Niesłony et al. 2015; Markopoulos et al. 2006) and statistical methods (Raissi et al. 2004; Farsani et al. 2007; Shokuhfar et al. 2008; Saeidi et al. 2013; Ghasemi et al. 2013). In the past years ANNs have emerged as a highly flexible modeling tool applicable in numerous areas of manufacturing discipline (Ezugwu et al. 2005; Al-Hazza et al. 2013). An artificial neural network is defined as "a data processing system consisting a large number of simple, highly interconnected processing elements (artificial neurons) in an architecture inspired by the structure of the cerebral cortex

✉ Angelos P. Markopoulos
amark@mail.ntua.gr

[1] Section of Manufacturing Technology, School of Mechanical Engineering, National Technical University of Athens, Heroon Polytechniou 9, 15780 Athens, Greece

of the brain" (Tsoukalas and Uhrig 1997). Actually, ANNs are models intended to imitate functions of the human brain using its certain basic structures. ANNs have shown to be effective as computational processors for various associative recall, classification, data compression, combinational problem solving, adaptive control, modeling and forecasting, multisensor data fusion and noise filtering. ANNs have been used in connection to milling in various papers in an effort to predict cutting forces, tool wear and cutting temperatures (Zuperl et al. 2006; Ghosh et al. 2007; Adesta et al. 2012). More specifically, many researchers have made several efforts to predict the surface roughness in milling. Statistical and empirical models have been proposed (Zhang et al. 2007). Furthermore, soft computing techniques are quite common, including ANNs (Kohli and Dixit 2005; Özel and Karpat 2005), genetic algorithms (Oktem et al. 2006) and fuzzy logic (Chen and Savage, 2001).

Back Propagation (BP) ANNs are more common than any other kind of network (Al Hazza and Adesta 2013) to be found in the relevant literature. Although this kind of models may be approached in many ways pertaining to the characteristics of the training procedure used, the problem of selecting the most suitable scheme is not addressed satisfactorily by the researchers. In this paper, an attempt is made to test several training BP algorithms to test their characteristics and suitability for the at hand problem. Feed-forward perceptrons are trained with three different back propagation algorithms, namely the adaptive back propagation algorithm of the steepest descent, the back propagation Levenberg–Marquardt algorithm and the back propagation Bayesian algorithm. This way, several models with different characteristics are built and tested and the optimum is selected. The analysis results indicate that the proposed model can be used to predict surface roughness in end milling with a less that 10 % error, even for tests with cutting conditions that were not used in the training of the system. Furthermore, Radial Basis Function (RBF) neural networks are tested with the same problem and the results are compared to the previous ones. A comparative study indicates the advantages and disadvantages of each approach. Furthermore, the models are used for the prediction of surface roughness in milling. The best models in terms of their performance are stored and can be used for the prediction of surface roughness of random input data, confined of course in the extremes of the input data used in the training of the models, but also 3D surfaces are produced for the a priori determination and selection of optimum cutting conditions, based on experimental results and simulation output. It can be concluded that the proposed novel models prove to be successful, resulting in reliable predictions, therefore providing a possible way to avoid time- and money-consuming experiments.

Artificial neural networks

Artificial neural networks are parallel systems which consist of many special, non-linear processors, known as neurons. Like human cerebrum, they are able to learn from examples, they possess generalization capabilities and fault tolerance and they can respond intelligently to new triggers. Each neuron is a primary processing unit, which receives one or more external inputs and uses them to produce an output. It consists of three basic elements: a number of synapses, an adding node and an activation or transfer function. Each synapse is characterized by a specific weight w_i with which the respective input signal x_i is multiplied. The node adds the resulting numbers and finally the activation function "squashes" the neuron output in the normalized (0,1) or (−1,1) range. Neuron model also includes a threshold θ that practically demotes input to the activation function. The activation function input can be increased if a bias term b is used, which is equal to the negative of the threshold value, i.e. $b = -\theta$.

The whole system is perceived as parallel because many neurons can implement calculations simultaneously. The most important feature of a neural network is the structure of the connected neurons because it determines the way the calculations are performed. Apart from the source layer which receives the inputs and the output layer on which the input layer is mapped, a neural network can have one or more intermediate hidden layers. Furthermore, the types of inter-neuron connections determine the characteristics of a network and consequently the tasks for which it is designed to be used. According to this, in feed forward networks, data run exclusively from the input units to the output units while in recurrent networks, feedback connections act beneficially for the training process and the behaviour of the network. A typical feed forward ANN with one hidden layer consisting of four units, six source units and two output units is shown in Fig. 1.

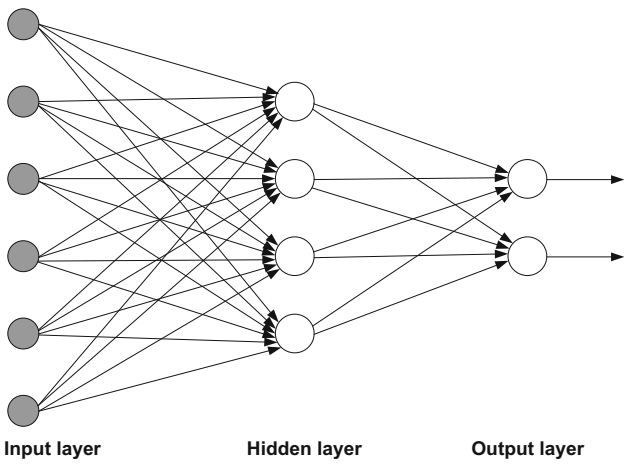

Fig. 1 Single hidden layer feed forward ANN 6-4-2

Finally, the algorithm used for the network training affects its performance and effectiveness. The training of neural networks refers to the procedure adopted to achieve a desired behaviour by modifying the synaptic weights, which allows them to learn from their environment and improve their performance through time. The various methods of adjusting the connections weights constitute different training algorithms. Each algorithm comprises a well-defined set of rules for solving the training problem and has specific advantages and disadvantages. Depending on the environment of the neural network, three different training methods may be distinguished, namely supervised, unsupervised and reinforcement learning. In supervised learning the training is based on examples of desired behaviour. The parameters are adjusted according to the training vector and the error signal between the desired $y_d(t)$ and the calculated values $y(t)$ of the network outputs. The adjustment is implemented with an error correction algorithm like Least Mean Square (LMS). It must be pointed out that typical supervised learning neural networks are the feed forward and the radial basis function networks. On the other hand, in unsupervised learning the weights are modified in response to network inputs only. There are no target outputs available. Most of these algorithms perform clustering operations. They categorize the input patterns into a finite number of classes. This is especially useful in applications such as vector quantization. Finally, reinforcement learning trains the network with the use of a feedback signal called reinforcement signal, which "awards" the right behaviour or "punishes" the wrong behaviour of the network. It can be easily understood that the network receives only a training data set without the respective desired outputs and during the training it tries to find a set of weights which tend to avoid negative reinforcement signals.

In this study, feed forward networks with one or two hidden layers and RBF networks were employed. The output layer had constantly a single neuron corresponding to the predicted value of the surface roughness. The most known neural networks with one or more hidden layers are the multilayer perceptrons (MLP). These networks, unlike simple perceptron, are capable of classifying linearly inseparable patterns and can solve complicated problems. They can handle satisfactorily problems like pattern classification, generalization and functions approximation. They are trained with back propagation algorithms and the transfer function employed is a differentiable sigmoid function like hyperbolic tangent.

Back propagation algorithm

Back propagation algorithm is a supervised learning algorithm which adapts the synaptic weights, aiming to minimize the Mean Squared Error (MSE) between the desired and the actual network outputs after each input vector presentation. Standard back propagation is a gradient descent algorithm in which the network weights are moved along the negative of the gradient of the performance function. The main idea of the algorithm is that the errors of the hidden layers are specified by the back propagation of the output neurons errors. The algorithm includes a forward and a backward phase. During the forward phase the input signals "travel" through the network from input to output, layer by layer, generating eventually a certain response and during the backward phase the error signals are back propagated, from the output layer to the input layer resulting in the adjustment of the network parameters by minimizing the MSE. Therefore, the synaptic weights are adjusted to minimize the criterion:

$$E_p(t) = \frac{1}{2}\sum_k e_k^2(t), e_k(t) = y_{d_k}(t) - y_k(t) \quad (1)$$

and the summation includes all k neurons of the output layer after the presentation of each training pattern p at the input, consequently the minimization of $E_p(t)$ must be done pattern to pattern.

According to the aforementioned, the steps of the back propagation algorithm are the following:

1. Selection of the initial weights using small, positive values.
2. Presentation of the training vector to the network and forward calculation of the input weighted sums and all the neurons output values, neuron by neuron, up to the output layer where the output vector is produced.
3. Calculation of the difference between the actual output vector and the desired output vector as well as of the necessary weights modification.
4. Calculation of the hidden neurons error and the respective change of weights.
5. Adjustment of all the weights, beginning from the output neurons and continuing backwards to the input layer, by adding the corresponding values calculated before and using the equation $w_{ji}(t+1) = w_{ji}(t) + \gamma\delta_j(t)y_i(t)$, where γ is the learning rate parameter. In previous equations, to achieve faster convergence, a momentum term $a\left[w_{ji}(t) - w_{ji}(t-1)\right]$, $0 < \alpha < 1$, is added. The term momentum is used to determine the effect of the previous weight modification to the next one.

It worth noticing that there are two different ways in which the gradient descent algorithm can be implemented: incremental or pattern mode and batch mode. In the incremental mode, the gradient is computed and the weights are updated after each input is applied to the

network. In the batch mode all inputs are applied to the network before the weights are updated; in other words this mode is based on the whole set of examples known as epoch. In the incremental mode, BP algorithm does not always converge to a local minimum because the gradient used for the weights adjustment is computed for a single training example. BP algorithm needs many repetitions to converge; however there is always the possibility to get stuck in a local minimum, often due to false weight dimension choice. So, the weights selection is very important for the successful training and usually randomly small, negative values, uniformly distributed in a narrow range are appointed.

Gradient descent and gradient descent with momentum are two methods often too slow for practical problems. In these cases high performance algorithms that can converge from 10 to 100 times faster than the algorithms discussed previously, like Levenberg–Marquardt algorithm, are used. All these algorithms operate in batch mode.

Back propagation Levenberg–Marquardt algorithm

Levenberg–Marquardt algorithm (LMA) provides arithmetical solution to a sum of squares of linear functions minimization problem. The functions depend on a common set of parameters and the algorithm is an alternative solution of the Gauss–Newton algorithm (GNA) and the gradient descent method. LMA is more robust than GNA; it uses the second derivatives of the cost function resulting in better and faster convergence and in many cases it is able to give solution even when it starts far away from the final minimum. However, when the functions are "well behaved" and the initial parameters vary between logical values Levenberg–Marquardt is generally slower than Gauss–Newton algorithm. In gradient descent method only the first derivatives are calculated and the information change parameter includes only the direction in which the cost function is minimized. Therefore many solutions leading to convergence may arise, increasing the possibility the time needed for a solution to be found to become prohibitive.

The algorithm can be described by the following equations:

$$e = d - F(\phi.u) \tag{2}$$

$$J = \frac{1}{2}e^2 \tag{3}$$

$$\Delta\phi = -\left(\nabla^2 J(\phi)\right)^{-1}\nabla J(\phi) \tag{4}$$

where e is the observed output error, d is the target output, u is the system output, F is the fuzzy system function, ϕ is a general parameter of the fuzzy system, J is the cost function, $\nabla^2 J(\phi)$ is the Hessian matrix and $\nabla J(\phi)$ is the gradient relative to the cost function (3).

The observed output error is used for the minimization of the cost function by calculating the value of Eq. (4). The target is the minimization of the instant cost of Eq. (3). If the expansion of the Taylor series to the error $e(\phi)$ is applied around the function point then the first derivatives produce the Jacobian matrix:

$$J_s = \begin{bmatrix} \frac{\partial e_1}{\partial \phi_1} & \cdots & \frac{\partial e_1}{\partial \phi_B} \\ \cdots & \cdots & \cdots \\ \cdots & \cdots & \cdots \\ \cdots & \cdots & \cdots \\ \frac{\partial e_L}{\partial \phi_1} & \cdots & \frac{\partial e_L}{\partial \phi_B} \end{bmatrix} \tag{5}$$

where B is the number of the adaptive parameters and L is the outputs number. Finally, the parameters adjustment algorithm is given by:

$$\Delta\phi = N_\phi = -\left(J_s^T J_s + qI\right)^{-1} J_s^T e \tag{6}$$

when q is large, the adjustment method shown above becomes the gradient descent method with step $1/q$ and when q is small it actually becomes Newton's method. Thus, by importing a term of this kind a smooth transition between the methods of gradient descent and Newton–Gauss is achieved and the invertibility problem is eliminated.

The parameters which can be modified, aiming to optimize the training procedure with Levenberg–Marquardt method of each network are the adaptive Marquardt value mu, the mu decrease and mu increase factors, the final target mean squared error, the minimum gradient value and the maximum mu value. The parameter mu is the initial value for q. This value is multiplied by mu decrease value whenever the performance function is reduced by a step. It is multiplied by mu increase whenever a step would increase the performance function. If mu becomes larger than maximum mu, the algorithm is stopped. Generally, training stops when one of the following conditions is fulfilled: the maximum epoch number is reached, the maximum training time is reached, the desired mean squared error is achieved, the gradient value becomes smaller than its minimum value and as mentioned above when the adaptive Marquardt overruns its maximum value.

LMA appears to be the fastest method for training moderate-sized feed forward neural networks. It also has a very efficient Matlab implementation, since the solution of the matrix equation is a built-in function; thus its attributes become even more pronounced in a Matlab setting.

Back propagation Bayesian algorithm

Back propagation Bayesian algorithm updates the weight and bias values according to Levenberg–Marquardt

optimization. Bayesian regularization minimizes initially a linear combination of squared errors and weights and then determines the correct combination so as to produce a network that generalizes well. The parameters which can be modified to optimize each network's training procedure with the Bayesian method are the adaptive Marquardt mu, the mu decrease and mu increase factors, the final target mean squared error, the minimum gradient value and the maximum mu value. In Bayesian regularization back propagation is used to calculate the Jacobian jX of the network performance with respect to the weight and bias values X. Each variable is adjusted according to LMA as shown below:

$$jj = jX \cdot jX \tag{7}$$

$$je = jX \cdot E \tag{8}$$

$$dX = \frac{-(jj + I \cdot mu)}{je} \tag{9}$$

where E is the errors sum and I is the identity matrix. The adaptive value mu is increased by mu increase value until the change of X results in a reduced performance value. The change is then made to the network and mu is decreased by mu decrease value. Training stops whenever one of the conditions mentioned in Levenberg–Marquardt method is fulfilled.

Radial basis function neural networks

Radial basis function neural networks are trained with supervised learning algorithms and can be perceived as improvements of the multilayer feed forward back propagation networks. Many of their characteristic features are similar to those of feed forward neural networks because they perform linear representations and weights summations. However, the transformations performed are local, resulting in their much faster training. Radial basis networks may require more neurons than standard feed forward networks, but often they can be designed to take a fraction of time it takes to train standard feed forward networks. It is proven that RBFs with one hidden layer can approximate any function; as a result they are called universal approximators. Perceptrons also have the capability of universal approximation but only RBFs possess the ability of optimum approximation. Even if structurally they are less complicated than feed forward back propagation networks they can achieve better arbitrary functions approximations with only one hidden layer. Also, it has been observed that they work best when many training vectors are available. When for every input which must be classified there is a basis function $\Phi(\|x - y\|)$ the network will give a function, adaptive to every pattern.

The basic structure of an RBF neural network includes an n dimension input layer, a fairly larger dimension m hidden layer ($m > n$) and the output layer. Typical radial basis functions are Gaussian and logistic function, which are shown below:

$$R_t(\mathbf{x}) = \Phi(\|\mathbf{x} - \mathbf{c_i}\|) = \exp\left[\frac{-\|\mathbf{x} - \mathbf{c_i}\|^2}{2\sigma_i^2}\right] \tag{10}$$

$$R_t(\mathbf{x}) = \Phi(\|\mathbf{x} - \mathbf{c_i}\|) = \frac{1}{1 + \exp\left[\frac{-\|\mathbf{x} - \mathbf{c_i}\|^2}{\sigma_i^2}\right]} \tag{11}$$

where $\boldsymbol{c_i}$ is the i centre of $R_t(\mathbf{x})$ function. The right placement of the centres $\boldsymbol{c_i}$ is very important for the achievement of great learning rates. The σ_i parameter is called amplitude and determines the value of the maximum distance between two inputs that the node has a prominent impact. The radial basis functions selection is not decisive for the network efficiency.

The typical structure of an RBF network is shown in Fig. 2. Input units distribute the values to the hidden layer units uniformly, without multiplying them with weights. Hidden units are known as RBF units because their transfer function is a monotonous radial basis function. Hidden layer outputs are led to the output units and summed with appropriate weights.

For training, the number of RBF units is primarily selected, which is very important for the success of the procedure, usually by a "trial and error" method. Afterwards, four steps are followed:

1. Input training data are grouped and the centres $\boldsymbol{c_i}$, $i = 1, 2, \ldots, M$ of these groups are defined as centres of the M hidden units. Afterwards, k-centres algorithm is implemented and as a result a certain number of clustering centres is considered and a separation of the whole set of patterns into subsets is performed.
2. The amplitudes σ_i are defined, usually with the p-closest neighbour method which alters the values

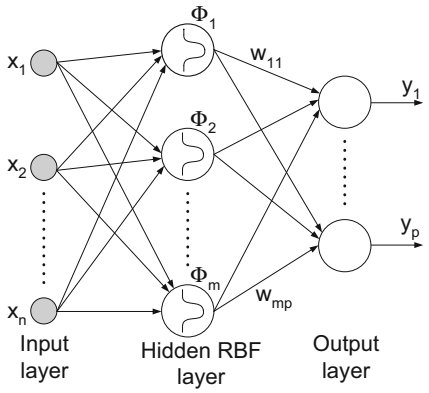

Fig. 2 Basic structure of an RBF neural network

to achieve overlapping of the response of every hidden and its neighbouring unit. The function used is:

$$\sigma_i = \left\{ \frac{1}{p} \sum_{j=1}^{p} \left\| x_i - x_j \right\|^2 \right\}^{1/2} \quad (12)$$

where x_j is the p-closest neighbour of x_i.

3. Transfer functions using Eqs. (10) and (11) are computed for the cases of Gaussian and Logistic function respectively.

4. Weights vector $\boldsymbol{w} = [w_1, w_2, \ldots, w_m]^{\mathrm{T}}$ of the output units is computed, utilizing the minimum squares method:

$$J = \frac{1}{2} \sum_{k=1}^{p} \left\| y_k - a_{ki}^T w \right\|^2 = \frac{1}{2} (y - \mathbf{A}w)^T (y - \mathbf{A}w) \quad (13)$$

where $\boldsymbol{y} = [y_1, y_2, \ldots, y_p]^{\mathrm{T}}$ is the output training vector, $\alpha_{\kappa\iota}$ is the transfer function vector and $\mathbf{A}$ is the hidden units' activation matrix.

Table 1 Training set data

No of Training data	Speed (min^{-1})	Feed (mm/min)	Depth of cut (mm)	Vibrations (μV)	Surface roughness (μm)
1	1500	152.4	1.27	0.10168	1.4224
2	1500	457.2	0.254	0.13581	3.048
3	1500	609.6	0.762	0.19091	2.6162
4	1500	304.8	0.254	0.11231	2.2352
5	1250	304.8	0.254	0.1448	2.54
6	1250	609.6	1.27	0.18291	3.0734
7	1250	152.4	1.27	0.096899	1.8034
8	1000	609.6	1.27	0.18417	3.6068
9	1000	152.4	0.762	0.10976	1.9812
10	1000	304.8	1.27	0.18001	2.3368
11	1000	457.2	0.762	0.16149	3.1496
12	750	457.2	0.762	0.14068	3.7338
13	750	304.8	0.762	0.12654	2.5908
14	750	152.4	1.27	0.089752	1.8288
15	750	609.6	0.762	0.17928	4.3434
16	1500	228.6	0.254	0.08833	1.3462
17	1250	228.6	0.762	0.13814	2.0828
18	1000	533.4	0.254	0.10338	3.7846
19	750	228.6	0.254	0.093096	2.7686
20	750	533.4	0.254	0.11352	4.5212
21	750	533.4	1.27	0.16586	3.81

Table 2 Test set data

No of Test data	Speed (min^{-1})	Feed (mm/min)	Depth of cut (mm)	Vibrations (μV)	Surface roughness (μm)
1	1500	609.6	1.27	0.17874	2.794
2	1250	457.2	0.254	0.14558	2.921
3	1250	381	0.254	0.13378	2.7178
4	1000	533.4	0.762	0.16794	3.683
5	1500	381	0.254	0.14637	2.794
6	1250	533.4	0.254	0.13001	3.2766
7	1000	228.6	0.254	0.091113	2.3368
8	1000	381	0.762	0.14862	2.7432
9	750	533.4	0.762	0.16241	4.1402
10	750	381	1.27	0.15298	2.6416

Networks' modelling

The aim of the study is to test various training algorithms in both BP and RBF networks for the purpose of predicting the surface roughness in milling. Then the results are compared and useful conclusions are drawn. In all the proposed models the same set of input and output data is used. For the application of the method, experimental results from the relevant literature were exploited (Lou 1997). The experiments pertain to the CNC end milling 6061 aluminium blocks. The tool used was a four-flute 3/4 inch diameter milling cutter of HSS. During the machining an accelerometer sensor was used to measure tool vibrations. Spindle speed, feed rate, depth of cut and vibrations were selected as independent variables in this study. Vibrations depend partly on the other three independent variables and thus they could be treated as a dependent variable. However, due to the complex structural system consisting of workpiece, fixture, cutting tool and machine tool the vibrations and consequently the surface roughness cannot be described quite accurately by the limited set of independent variables. Therefore, vibrations are treated as an independent variable, as well.

Two sets of experimental data were obtained: training data set and testing data set. The training data set was obtained on the basis of four levels of spindle speed (750, 1000, 1250, 1500 rpm), six levels of feed rate (152.4, 228.6, 304.8, 457.2, 533.4, 609.6 mm/min) and three levels of depth of cut (0.254, 0.762, 1.27 mm). For each combination of spindle speed, feed rate and depth of cut, the corresponding vibration data (in μV) were recorded. The corresponding value of the roughness average R_α (in μm), the dependent output, was collected for each measurement. The training data used for the analysis are presented in Table 1.

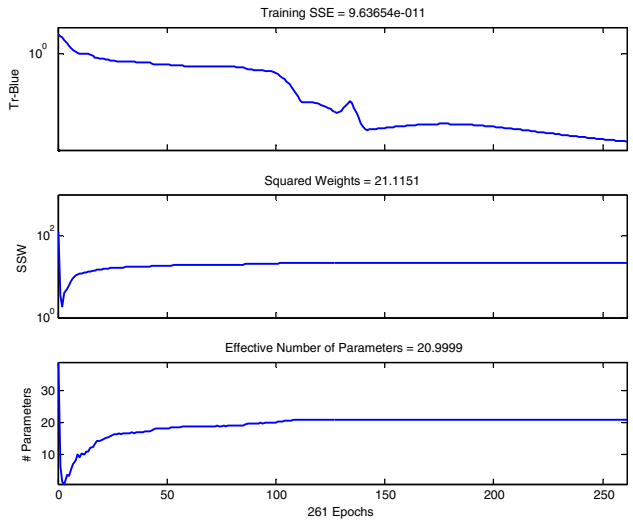

Fig. 3 MSE during training

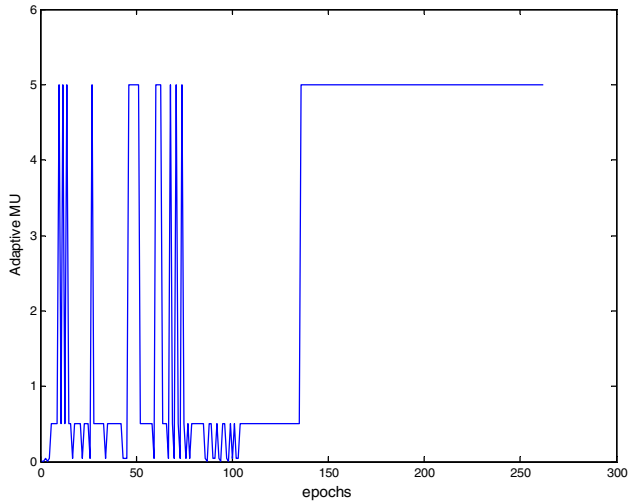

Fig. 4 Adaptive mu during training

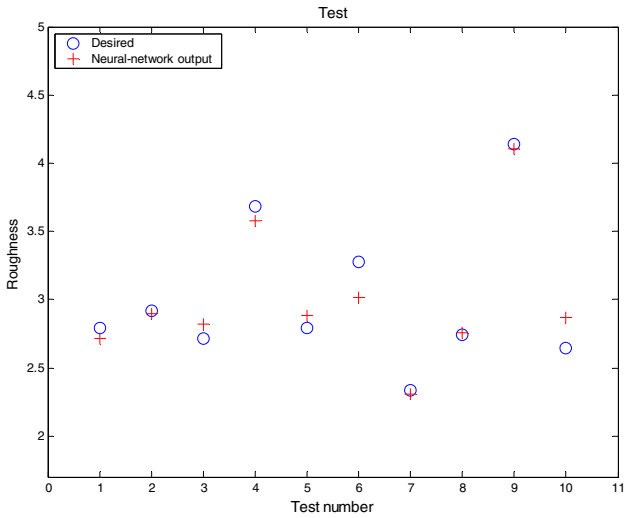

Fig. 5 Experimental values of surface roughness versus the NN predicted ones

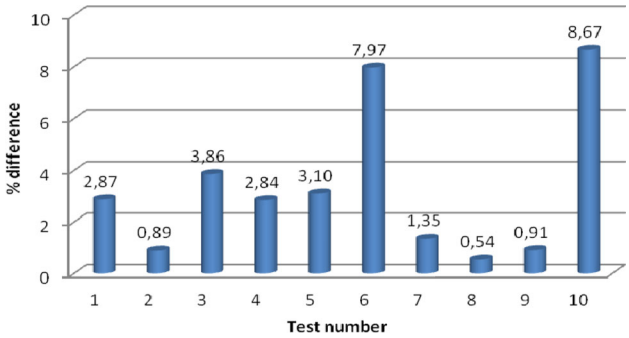

Fig. 6 The discrepancy of the predicted value from the experimental in percentage, for test data

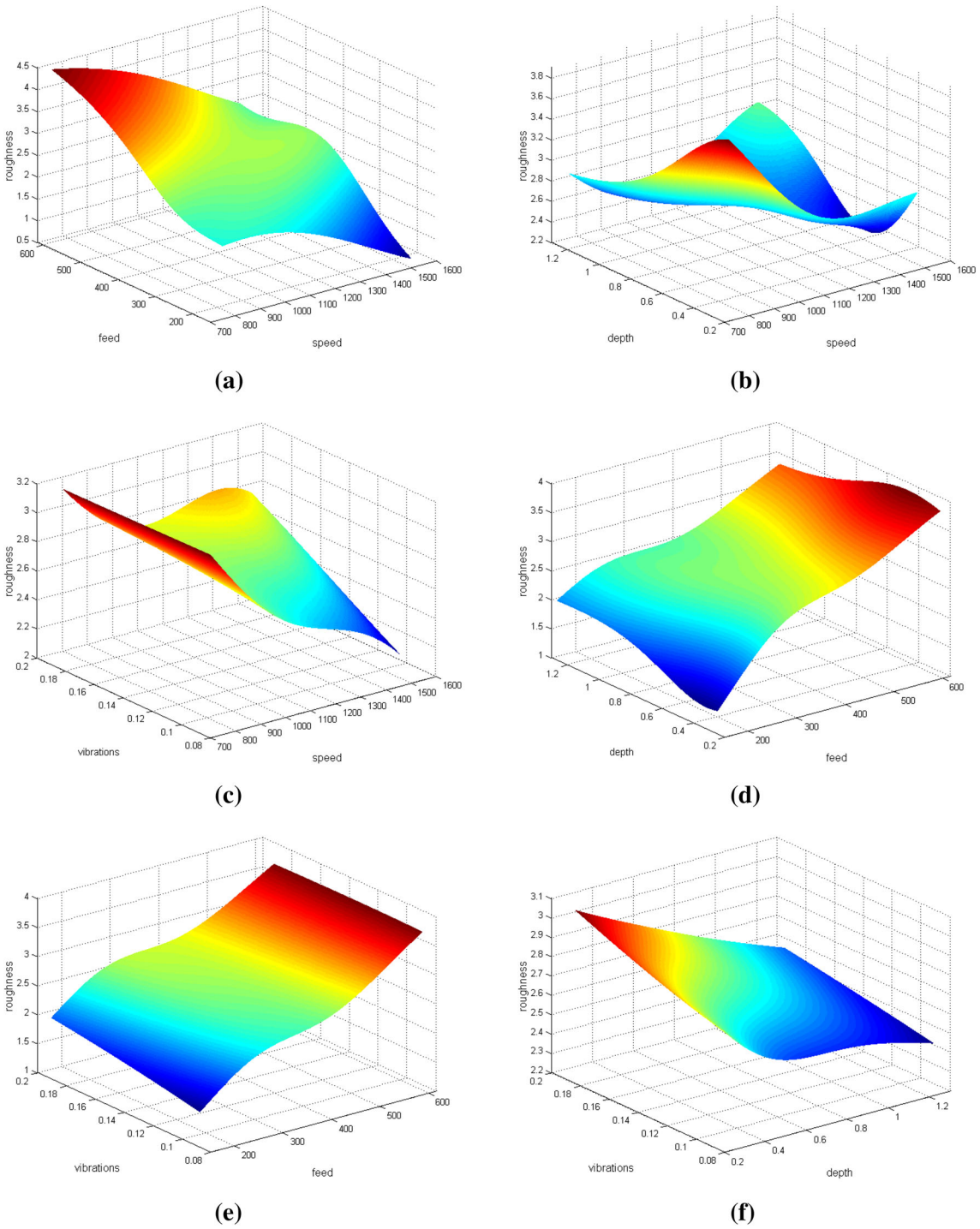

Fig. 7 Input-output surfaces of the optimum neural network

In this work, training data comprised 21 measurements selected randomly out of the 400 measurements originally presented by Lou (1997). The test data set was obtained on the basis of four levels of spindle speed (750, 1000, 1250, 1500 rpm), seven levels of feed rate (152.4, 228.6, 304.8, 381, 457.2, 533.4, 609.6 mm/min) and three levels of depth of cut (0.254, 0.762, 1.27 mm). Also for the test data set the data on vibrations and surface were recorded. The test data set comprised 10 measurements that are shown in Table 2. Note that in the test data set a value for the feed rate, namely 381 mm/min that has not been used in the training data set was also considered. This was chosen to

check whether the constructed system could predict correctly the value of the surface roughness when it has as input, values that it has not been trained for. The aim of this work was to create a system that could predict the surface roughness quite accurately; it is quantified as a small value of Mean Squared Error of training and test data, respectively.

Results and discussion

BP networks

For BP networks three different network training methods are employed, namely the adaptive back propagation algorithm of the steepest descent with the use of momentum term, the back propagation Levenberg–Marquardt algorithm and the back propagation Bayesian algorithm. In the proposed models, regardless of the training algorithm used, some modelling parameters remain the same. More specifically, it is set that the final training error to be smaller than 1×10^{-10} and the maximum number of epochs during which the network could be trained to be equal to 15000. However, the training process stops whenever the desired MSE is achieved or the designated epoch number is reached or finally when one of the parameters of each training algorithm reaches the maximum or minimum value.

It must be noticed that at the input layer all the values need to be of the same size so as to achieve faster and better convergence. Thus, modified values of the experimental data were used; depth of cut and vibration data are multiplied by 5 and 10, respectively and the spindle speed and feed rate data are divided by 500 and 100, respectively. All neural networks used have a four neurons input layer, because there are four types of input data, i.e. spindle speed, feed rate, depth of cut and vibrations, one or two hidden layers with variable number of neurons and a single neuron output layer for the surface roughness which is the system output. The hidden layers use the hyperbolic tangent sigmoid transfer function, which is expressed as:

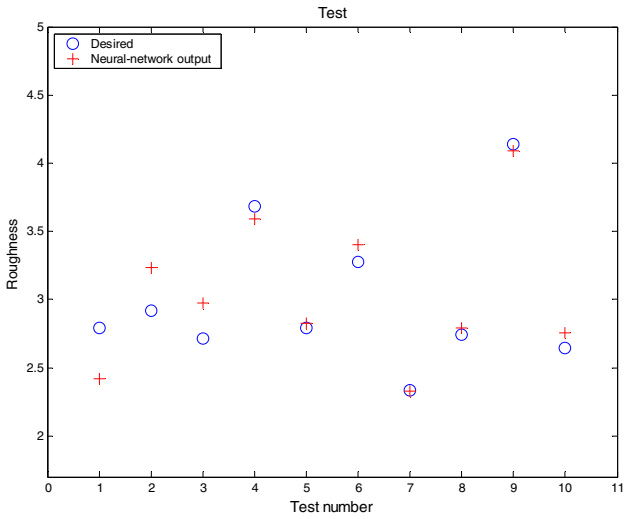

Fig. 8 Surface roughness for test data

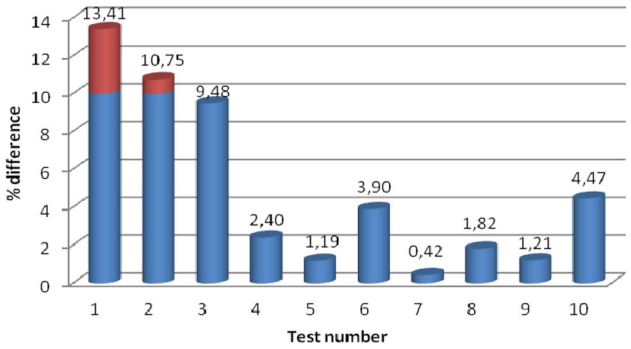

Fig. 9 The discrepancy of the predicted value from the experimental in percentage, for test data

$$f(u) = \tanh\left(\frac{u}{2}\right) = \frac{1 - e^{-u}}{1 + e^{-u}} \tag{14}$$

After training, the best neural network that managed to be fully trained and produced the smallest test mean squared error was the 4-6-1-1 network, meaning with 2 hidden layers with 6 and 1 neurons in the first and second hidden layer, respectively, trained with the back propagation Bayesian algorithm. For this specific network, it took 262 epochs to terminate the training procedure, achieving MSE of training data equal to 4.59×10^{-12}, while the respective MSE of test data was 0.01599 and all test data produced error smaller than 10 %. Obviously, the values of both the mean squared errors of training and test data are significantly small. Also, this network produced the smaller value of mean squared error of all the networks that were trained.

In Fig. 3 the alteration of the value of mean squared error of training data versus the epochs is depicted and in Fig. 4 the alteration of the value of adaptive training factor *mu* versus the epochs is plotted. In Fig. 5 the experimental values of surface roughness and the corresponding calculated values of surface roughness by the neural network for the test data are shown. Finally, Fig. 6 shows the difference of the computed to the experimental surface roughness values for the test data, indicating in most cases a very good prediction.

In Fig. 7a, b, c, d, f, the total surfaces which describe the input–output space, produced when only two of the input variables are altered each time, are shown. The input vector used was (speed = 1125 rpm, feed = 381 mm/min, depth of cut = 0.762 mm and vibrations = 0.1392 μV), therefore, in each figure the variables that are not mentioned take their pre-assigned values from the vector above. The two input variables that are altered each time take all the possible values between their widths of rate.

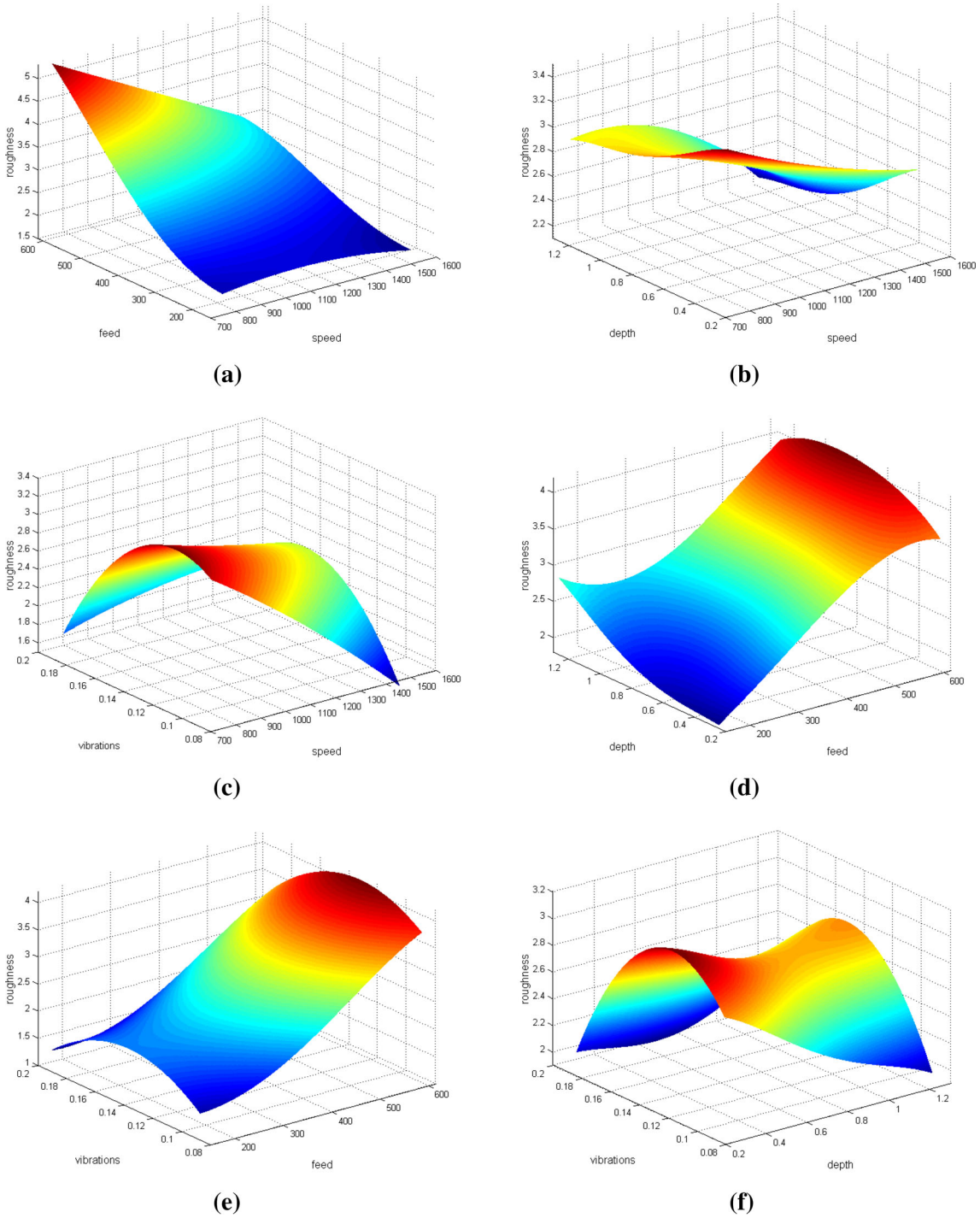

Fig. 10 Input-output surfaces of the optimum RBF network

It is worth noticing that generally neural networks are not stable; each time a network is trained, the initial weights as well as the initial bias values are chosen randomly from the programme. This random selection strongly affects the training procedure and the final error of the network. The same network can achieve complete training for a specific set of initial weights and biases and afterwards can fail to be trained for another set of weight and bias values. For this reason, for each network examined, the network ran 15 times and then the most possible value one could take with only one "running" of the network was chosen.

Table 3 Comparison of optimum BP and RBF networks

Network type	Structure	MSE training	MSE Test	Time (s)	Test data > 10 %
Feed forward	4-6-1-1	4.59×10^{-12}	0.01599	8.041	0
Radial	4-21-1	8.85×10^{-20}	0.034973	0.31	2

RBF networks

In the second part of this work radial basis function networks are examined, which could be trained and then predict surface roughness. For RBFs, the target training mean squared error was chosen equal to 10^{-30}. Once again, all values at the input layer need to be of the same size, trying to achieve faster and better convergence. Thus, the modified values of the experimental data, as used in BP networks, were employed. Furthermore, in the proposed RBFs, the value of spread constant needs to be determined; an inappropriate spread constant could cause RBF overfitting or underfitting. For this purpose an algorithm is developed that can test all the resultant networks for spread values from 0.1 to 100, with changing step equal to 0.1. All RBF networks produced had only one hidden layer. The number of hidden layer neurons was modified from the program with respect to the spread value, so as the target mean squared error to be achieved. Similarly to BP networks, the networks presented in this paragraph have four input neurons, one for each input variable and one output neuron, corresponding to surface roughness. Hidden layer neurons used the radial basis activation function and the output neuron used the linear transfer function.

For every network, the results obtained were the training MSE, the test MSE and the training time. By comparing all the results it was concluded that the optimum network was the one with spread value equal to 15. This network produces the smallest test data MSE and only two of its test data produce difference between experimental and predicted data greater than 10 %; the one was hardly greater than the target value, thus substantially only for one data point there was high percentage error.

Figure 8 depicts the experimental values of surface roughness and the corresponding calculated values of surface roughness by the RBF network for the test data used. Figure 9 shows the discrepancies of the predicted values from the experimental ones in percentage, for the test data.

In Fig. 10a, b, c, d, f, the total input–output surfaces which are produced when only two of the input variables are altered each time, are shown. The input vector used was (speed = 1125 rpm, feed = 381 mm/min, depth = 0.762 mm, vibrations = 0.139 μV) therefore, in each figure the variables that are not mentioned take their preassigned values from the vector above.

From all the results taken it is concluded that training times for all networks were very small. Furthermore, as spread value was rising, test MSE was decreasing up to the moment spread value became equal to 15. For farther increase of spread value the error was increasing, too. Finally, RBF networks are absolutely stable; when the values of their parameters are kept constant they give the same results, no matter how many times they are trained.

In general, test data which produced most frequently error greater than 10 % were these placed in rows 2, 7 and 10 in test data table. Test data 10 contains a feed rate value (381 mm/min) for which it had not been trained. For test data 7, the corresponding feed rate (2228.6 mm/min) was used during training but not combined with the spindle speed value of 1000 rpm. Test data 2 consists of a combination of spindle speed (1250 rpm) and feed rate (457.2 mm/min) for which the network had not been trained. Table 3 tabulates some parameters connected to the performance of the finally selected BP and RBF networks, for comparison.

Conclusions

In this work, various training algorithms for BP networks and RBF networks were put to test for the prediction of surface roughness in end-milling. Four independent variables were used as inputs, namely spindle speed, feed rate, depth of cut and vibrations and the output of the networks was surface roughness. The first system was a feed forward network, which was examined for various numbers of neurons, with one or two hidden layers and for the training process three different methods, steepest descent, Levenberg–Marquardt and Bayesian, were used, with the latter giving the best results. The second system under examination was a radial basis network. Both systems can exhibit advantages and disadvantages when compared to one another and in both cases the results were quite satisfying.

Certain remarks concerning these two approaches are the following:

- RBFs can be trained much faster than perceptrons.
- Test data MSEs of perceptrons trained with the Bayesian method were generally smaller than the errors resulting from RBFs.
- The smallest training error was achieved with radial basis networks.
- Radial basis networks were very stable. Also, the networks trained with the Bayesian algorithm were generally stable contrary to the networks trained with

the Levenberg–Marquardt and steepest descent algorithms.

It can be finally concluded that artificial neural networks can satisfactorily predict surface roughness in milling.

References

Adesta EYT, Al Hazza MHF, Suprianto MY, Riza M (2012) Prediction of cutting temperatures by using back propagation neural network modeling when cutting hardened H-13 steel in CNC end milling. Adv Mater Res 576:91–94

Al Hazza MHF, Adesta EYT (2013). Investigation of the effect of cutting speed on the surface roughness parameters in CNC end milling using artificial neural network. IOP Conf. Series: Materials Science and Engineering 53. doi:10.1088/1757-899X/53/1/012089

Al Hazza MHF, Ndaliman MB, Hasan MH, Ali MY, Khan AA (2013) Modeling the electrical parameters in EDM process of Ti6Al4 V alloy using neural network method. Int Rev Mech Eng 7(7):1464–1470

Chen JC, Savage M (2001) A fuzzy net based multilevel in process surface roughness recognition system in milling operations. Int J Adv Manuf Technol 17:670–676

Ezugwu EO, Fadare DA, Bonney J, Da Silva RB, Sales WF (2005) Modelling the correlation between cutting and process parameters in high-speed machining of Inconel 718 alloy using an artificial neural network. Int J Mach Tools Manuf 45(12–13):1375–1385

Farsani R, Raissi S, Shokuhfar A, Sedghi A (2007) Optimization of carbon fibers made up of commercial polyacrylonitrile fibers using screening design method. Mater Sci Pol 25(1):113–120

Galanis NI, Manolakos DE (2014) Finite element analysis of the cutting forces in turning of femoral heads from AISI 316 l stainless steel. Lect Notes Eng Comp Sci 2:1232–1237

Ghasemi FA, Raissi S, Malekzadehfard K (2013) Analytical and mathematical modeling and optimization of fiber metal laminates (FMLs) subjected to low-velocity impact via combined response surface regression and Zero-One programming. Lat Am J Solids Struct 10(2):391–408

Ghosh N, Ravi YB, Patra A, Mukhopadhyay S, Paul S, Mohanty AR, Chattopadhyay AB (2007) Estimation of tool wear during CNC milling using neural network-based sensor fusion. Mech Syst Signal Process 21(1):466–479

Kohli A, Dixit US (2005) A neural network based methodology for the prediction of surface roughness in a turning process. Int J Adv Manuf Technol 25(1–2):118–129

Lou S (1997) Development of four in-process surface recognition systems to predict surface roughness in end milling. PhD Thesis, Iowa State University, Iowa, USA

Markopoulos A, Vaxevanidis NM, Petropoulos G, Manolakos DE (2006) Artificial neural networks modeling of surface finish in electro-discharge machining of tool steels. In: Proceedings of ESDA 2006, 8th Biennial ASME Conference on Engineering Systems Design and Analysis, 847–854

Niesłony P, Grzesik W, Chudy R, Habrat W (2015) Meshing strategies in FEM simulation of the machining process. Arch Civil Mech Eng 15(1):62–70

Oktem H, Erzurumlu T, Erzincanli F (2006) Prediction of minimum surface roughness in end milling mold parts using neural network and genetic algorithm. Mater Des 27(9):735–744

Özel T, Karpat Y (2005) Predictive modeling of surface roughness and tool wear in hand turning using regression and neural networks. Int J Mach Tools Manuf 45(4–5):467–479

Raissi S, Eslami FR, Shokuhfar A, Sedghi A (2004) Improving carbon fibers quality characteristic by using statistical modeling. Int J Appl Math Stat 2(4):60–72

Saeidi RG, Amin GR, Raissi S, Gattoufi S (2013) An efficient DEA method for ranking woven fabric defects in textile manufacturing. Int J Adv Manuf Technol 68(1–4):349–354

Shokuhfar A, Khalili SMR, Ashenai Ghasemi F, Malekzadeh K, Raissi S (2008) Analysis and optimization of smart hybrid composite plates subjected to low-velocity impact using the response surface methodology (RSM). Thin Walled Struct 46:1204–1212

Szabó G, Kundrák J (2014) Investigation of residual stresses in case of hard turning of case hardened 16MnCr5 Steel. Key Eng Mater 581:501–504

Tsoukalas LH, Uhrig RE (1997) Fuzzy and neural approaches in engineering. Wiley Interscience, New York

Zhang JZ, Chen JC, Kirby ED (2007) Surface roughness optimization in an end-milling operation using the Taguchi design method. J Mater Process Technol 184(1–3):233–239

Zuperl U, Cus F, Mursec B, Ploj T (2006) A generalized neural network model of ball-end milling force system. J Mater Process Technol 175(1–3):98–108