Expert Systems with Applications 39 (2012) 2397–2407

Contents lists available at SciVerse ScienceDirect

Expert Systems with Applications

journal homepage: www.elsevier.com/locate/eswa

Comparing the performance of neural networks developed by using Levenberg–Marquardt and Quasi-Newton with the gradient descent algorithm for modelling a multiple response grinding process

Indrajit Mukherjee [a,*], Srikanta Routroy [b]

[a] Shailesh J. Mehta School of Management, Indian Institute of Technology Bombay, Mumbai 400 076, India
[b] Mechanical Engineering Department, Birla Institute of Technology & Science, Pilani, Rajasthan, Pilani 333 031, India

ARTICLE INFO

Keywords:
Back propagation neural network
Gradient descent algorithm
Levenberg–Marquardt algorithm
Multiple response
Quasi-Newton algorithm

ABSTRACT

Monitoring and control of multiple process quality characteristics (responses) in grinding plays a critical role in precision parts manufacturing industries. Precise and accurate mathematical modelling of multiple response process behaviour holds the key for a better quality product with minimum variability in the process. Artificial neural network (ANN)-based nonlinear grinding process model using backpropagation weight adjustment algorithm (BPNN) is used extensively by researchers and practitioners. However, suitability and systematic approach to implement Levenberg–Marquardt (L–M) and Boyden, Fletcher, Goldfarb and Shanno (BFGS) update Quasi-Newton (Q-N) algorithm for modelling and control of grinding process is seldom explored. This paper provides L–M and BFGS algorithm-based BPNN models for grinding process, and verified their effectiveness by using a real life industrial situation. Based on the real life data, the performance of L–M and BFGS update Q-N are compared with an adaptive learning (A-L) and gradient descent algorithm-based BPNN model. The results clearly indicate that L–M and BFGS-based networks converge faster and can predict the nonlinear behaviour of multiple response grinding process with same level of accuracy as A-L based network.

1. Introduction

In recent years, control of grinding processes by appropriate mathematical models is a key issue in metal cutting industry (Chen & Kumara, 1998; Govindhasamy, MacLoone, Irwin, French, & Doyle, 2005; Krajnik, Kopac, & Sluga, 2005; Luong & Spedding, 2002; Mukherjee & Ray, 2006; Petri, Billo, & Bidanda, 1998; Shin & Vishnupad, 1996). However, development of appropriate mathematical model is considered to be a formidable and challenging task in almost all grinding situations (Feng, Wang, & Yu, 2002; Maksoud & Atia, 2004; Sathyanarayanan, Lin, & Chen, 1992; Shaji & Radhakrisnan, 2003). A significant improvement in process efficiency may be obtained by these model(s) that identifies and determines the critical process variables leading to desired output(s) or response(s). To implement appropriate online grinding process control mechanism, explicit functional model(s) needs to be developed by using mechanistic or data-driven empirical modelling approach (Box & Draper, 1987; Rowe, Yan, Inasaki, & Malkin, 1994). In this context, mechanistic modelling approaches, as proposed in literatures (Choi, Subrahmanya, Li, & Shin, 2008; Farago, 1976; King & Hahn, 1986; Malkin, 1984, 2007; Moulik, Yang, & Chandrasekhar, 2001), cannot consider all intrinsic process complexity manifested due to the presence of a large number of interacting variables (Markos, Viharos, & Monostori, 1998). In addition, due to interdependence between various process variables, these models are found to be inaccurate in many situations (Maksoud & Atia, 2004; Petri et al., 1998; Shin & Vishnupad, 1996). Complexity of the problem is further enhanced if there are nonlinear multiple responses to be modelled simultaneously. There is no adequate and acceptable mechanistic model(s) that can explain varied multiple response grinding situations. Fortunate, empirical models can handle such nonlinear behaviours, considering the internal complexities and dynamics of processes.

Statistical regression (Montgomery & Peck, 1992; Rencher, 1995, 1998), fuzzy set theory (Zadeh, 1965) and artificial neural network (ANN) (Fu, 2003) are some of the empirical modelling techniques used for prediction and control of grinding processes. These empirical models can be developed based on (i) designed experimentation data (Kwak, 2005; Montgomery, 2001) or (ii) actual production data, or (iii) by using mixture of production and experimental data. Using both production and experimental data can provide a robust mathematical model (Coit, Jackson, & Smith, 1998). However, due to many constraints in actual multi-stage manufacturing environment (Petri et al., 1998; Mukherjee, 2007),

* Corresponding author. Tel.: +91 22 25767742; fax: +91 22 25722872.
E-mail addresses: indrajitmukherjee@iitb.ac.in (I. Mukherjee), srikantaroutroy@gmail.com (S. Routroy).

0957-4174/$ - see front matter © 2011 Elsevier Ltd. All rights reserved.
doi:10.1016/j.eswa.2011.08.087

gathering experimental data is too difficult, if not impossible. Some of limitations of designed experimentation-based data collection in such environment are worth mentioning:

(i) Data are gathered based on additional cost of experimentations, which may be uneconomical in many line-layout type of mass manufacturing situations (Coit et al., 1998; Mukherjee, 2007).
(ii) Data may not reflect the effect of actual manufacturing environment (Coit et al., 1998).
(iii) They may be biased to certain operators, or certain procedures.

Therefore, using actual production data over a period of time in varied process conditions is a viable alternative for many researchers and practitioners.

Several applications of artificial neural network techniques in nonlinear cutting process problems have been reported in the open literatures. Supervised learning network using back propagation algorithm, proposed by Rumelhart, Hilton, and Williams (1986), have been successfully applied for modelling (a) a typical creep feed super alloy-grinding (Sathyanarayanan et al., 1992), and (b) prediction of material removal rate and surface finish parameter of a typical abrasive flow machining (Jain, Jain, & Kalra, 1999). Shin and Vishnupad (1996) provide an intelligent grinding process control scheme based on neuro-fuzzy optimization approach. Chen and Kumara (1998) use a hybrid approach of fuzzy set (Zadeh, 1965) and ANN-based technique (ANN) for modelling a grinding process. Petri et al. (1998) develop and propose a backpropagation ANN network (BPNN) model for predicting surface finish, and dimensional accuracy in a grinding process. Cus, Zuperl, and Milfelner (2006) shows the effectiveness and superiority of a modified BPNN as compared to analytical model for cutting force prediction of end milling operation. Mukherjee and Ray (2008a) provided a systematic solution methodology for multivariate (linear and nonlinear) model development, and implementation for grinding processes. Kumar and Choudhury (2007) uses a hybrid algorithm, having its root in Levenberg–Marquardt (L–M) algorithm to predict wheel wear and surface roughness of HSS in electro-discharge diamond grinding process. Haber and Alique (2003) proposed an intelligent supervisory system for tool-wear prediction of milling process, created on an ANN network, trained by using L–M algorithm. Sorensen, Norgaard, Ravn, and Poulsen (1999) describes a control method for non-linear, open-loop unstable, non-minimum-phase plants systems based on generalized predictive control using Quasi-Newton algorithm-based ANN model. Vafaeesefat (2009) uses a BPNN network trained by L–M to model a creep feed grinding force and determine its optimal process conditions.

Literature review by the authors on modelling of nonlinear multiple response grinding processes clearly revealed that there is only very few research works (Sedighi & Afshari, 2010; Tawakoli, Rabiey, & Lewis, 2009; Urbaniak, 2004) on successful implementation of Levenberg–Marquardt algorithm-based BPNN. However, no application of Quasi-Newton (Q-N) algorithm based BPNN in grinding process modelling is reported in literature. Therefore, a need is felt to verify the suitability of L–M and Q-N performance as compared to gradient descent algorithm used in BPNN-based grinding process models.

Inspired by this need, this paper first provides a basic theoretical background to develop BPNN networks using gradient descent, Levenberg–Marquardt (L–M), and BFGS update Quasi-Newton (Q-N) algorithm for modelling nonlinear multiple response process. Subsequently, the performances of the trained networks are compared using real life grinding process data. The process data was collected from an automobile engine manufacturing unit, located in eastern India.

2. Development of backpropagation neural networks

Backpropagation neural network (BPNN) (Rumelhart et al., 1986; Widrow & Lehr, 1990) is a key development and popular choice for researchers in various process modelling applications (Chen, Lin, Yang, & Tsai, 2010; Coit et al., 1998; Feng et al., 2002; Mukherjee & Ray, 2008b; Sathyanarayanan et al., 1992; Wasserman, 1993; Zhang & Huang, 1995). The prediction ability of dynamic nonlinear behaviour of multiple responses makes BPNN an attractive choice for many researchers. In addition, the network can be trained without prior assumptions on the functional form (e.g. linear, quadratic, higher order polynomial and exponential) or any other distribution assumptions. These inherent properties, provide BPNN a clear edge over complex nonlinear regression technique (Basheer & Hajmeer, 2000).

Few of the attractive features of BPNN are:

(i) learning and adoptability allows the process to update or modify its internal structure according to changing environment,
(ii) ability to handle imprecision and fuzzy information and capability of generalization,
(iii) noise-insensitivity provides accurate prediction in the presence of uncertain data and measuremental error (Bates & Watts, 1988).

In case of fully interconnected artificial neural network with single hidden layer neuron processing units (Fig. 1), each neuron has an adjustable weight factor (w_{ij}). w_{ij} is the weights assigned for the connection between node i and j.

The network architecture also has bias terms, which can improve convergence property of the network. The nature of mapping relationship between input and output vectors is defined by these weights within the network. The input layer acts as an input data holder that distributes the input to the first hidden layer. The outputs from the first hidden layer then become the inputs to the second layer and so on. The last layer acts as the network output layer. In feedback network, signals propagate from output of any neuron to the input of any neuron (Zhang & Huang, 1995). Based on the

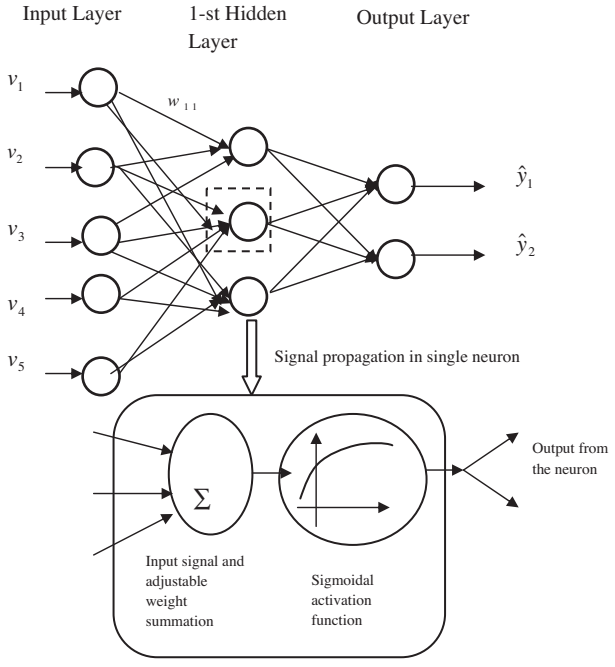


Fig. 1. A multilayered perceptron networks and signal propagation.

amount of guidance that the learning process receives from outside agent, network can be further classified as supervised or unsupervised. The difference between predicted and target response produces an error signal, which is used to modify the network weights. The adjustment of weights is intended to be in a direction that reduces the difference between network output and actual response vectors. Training set vectors are applied to the network repeatedly, until the error measure is at an acceptable value or other convergence criteria of intrinsic algorithm is met. The change in weights may be done at the time each input vector is applied. This is also known as 'sequential mode of training'. The change can also be averaged and weights are changed after all the input vectors are applied. This type of training is called 'batch mode of training'. As this technique work on calculating error starting from output layer down through hidden layer, it is named as 'back propagation of error' with modified delta rule (Bates & Watts, 1988; Rumelhart et al., 1986).

The performance measure generally used to compare different learning algorithm is mean-square of error or *MSE* (Mukherjee, 2007; Tsai & Wang, 2001), and expressed as

$$MSE(t) = \frac{1}{Np}\sum_{i=1}^{N}\sum_{k=1}^{k=p}\left[\left(d_k^i\right)^2 - \left(y_k^i\right)^2\right], \quad (1)$$

where t is the number of training epoch completed. y_k^i is the kth network predicted output activation of ith training pattern, d_k^i is the kth actual or observed output of the of ith training pattern, and N is the total number of training pattern.

Backpropagation calculate the derivative of the performance function with respect to weight and bias variable matrix (W). Each variable in W is adjusted according to the gradient descent algorithm (Rardin, 2003) with momentum and learning rate.

$$dW_t = m_c dW_{t-1} + l_r m_c \frac{d(Performance\ Function)}{dW_t}, \quad (2)$$

where t is the number of epoch. W_{t-1} is the previous change to the weight or bias, l_r is the learning rate, m_c is the momentum coefficient, and performance objective function is assumed to be of quadratic form. A brief discussion on momentum coefficient, learning rate, and number of epoch is given below.

Momentum coefficient

Momentum coefficient (Hagan, Demuth, & Beale, 2002; Howard & Beale, 1998) describes the proportion of the weight change that is added to each subsequent weight change. Low momentum causes weight oscillation and instability, and preventing the network from learning. High momentum can cripple network adaptability. For stable back-propagation, the momentum factor should be kept less than one. Momentum factors close to unity is needed to smooth error oscillations. During the middle of training, when steep slope occurs, a small momentum factor is recommended. However, during the end of training, large momentum factor is desirable.

Learning rate

Error surface for multilayer perceptron may vary in different regions of parameter space. Using simple steepest descent BPNN approach, learning rate (Hagan et al., 2002) is usually kept at a constant throughout the training phase. However, if the learning rate is too high, the algorithm may oscillate and become unstable. In contrary, if the learning rate is too small, the algorithm will take too long to converge (Howard & Beale, 1998). Variable learning rate provides better flexibility to handle such situations.

Number of epochs

On a particular epoch, each training cases are submitted to the network. Then the target and actual outputs are compared to see the error performance. The error measure together with the error surface gradient is used to adjust the weights. This process is repeats. The initial network weight configuration is selected at random. The learning of the network stops, as soon as any of the following conditions is satisfied:

(i) maximum number of epoch (repetitions) predefined is reached,
(ii) maximum amount of iteration time given has been reached,
(iii) minimum level of error performance is reached,
(iv) performance gradient falls below the minimum gradient predefined.

In this paper, for all the three algorithms selected, the convergence criteria or performance goal is set to 0.001 or if number of epoch reaches 2000. Momentum coefficients, m_f is selected as 0.9 (Hagan et al., 2002; Howard & Beale, 1998). The choice of squashing functions or transfer function (Howard & Beale, 1998) does not seem to be critical as long as the function is non-linear and neuron outputs are bounded (Wasserman, 1993). However, hyperbolic tangent function is selected for this study. It is to be mentioned that as back propagation algorithm is a steepest decent type of search, there is always a chance for the algorithm to get trapped in local error minima. In addition, selection of intrinsic parameter values and order of differential functions usually influences the BPNN performance. As and when the learning is completed, the final values of weights are fixed. The final weights (W_{Final}) are used for testing and other network functional 'recalling' sessions. The regression equation can be expressed as,

$$\widehat{Y} = W_{Final}[X], \quad (3)$$

where X is the array of independent input variables, and $\widehat{Y}$ is the array of predicted response variables.

All the three algorithms, selected for this study, are trained in a batch training mode (Howard & Beale, 1998; Mukherjee & Ray, 2008c). The details of the three algorithms selected are discussed below.

2.1. Gradient descent algorithm-based backpropagation with modified performance function, and adaptive learning rate (A-L BPNN)

In this approach, the error performance function is slightly modified from the simple MSE calculation used for backpropagation-based network design. A regularization approach is used to improve the generalization of the final network architecture, and prevent over-fitting (or over-training). A modified performance function, so-called 'MSEREG' (Howard & Beale, 1998; Mukherjee & Ray, 2008c) is selected, and expressed as

$$\mathrm{MSEREG}(t) = \gamma_i \times [\mathrm{MSE}(t)] + (1-\gamma_i) \times [\mathrm{MSW}(t)], \quad (4)$$

where t is the number of training epoch completed, and mean sum of squared error, (MSE) is expressed as

$$\mathrm{MSE}(t) = \frac{1}{Np}\sum_{i=1}^{N}\sum_{k=1}^{k=p}\left[\left(d_k^i\right)^2 - \left(y_k^i\right)^2\right], \quad (5)$$

and sum of square of network weights (MSW) is expressed as,

$$\mathrm{MSW}(t) = \frac{1}{n}\sum_{ij=1}^{n} w_{ij}^2, \quad (6)$$

where γ_i is a performance weighted ratio, and n is the total number of interconnected weights. Use of this performance function causes the network to have smaller weights and biases. It also forces the network responses to be smoother and less likely to over-fit (Howard & Beale, 1998).

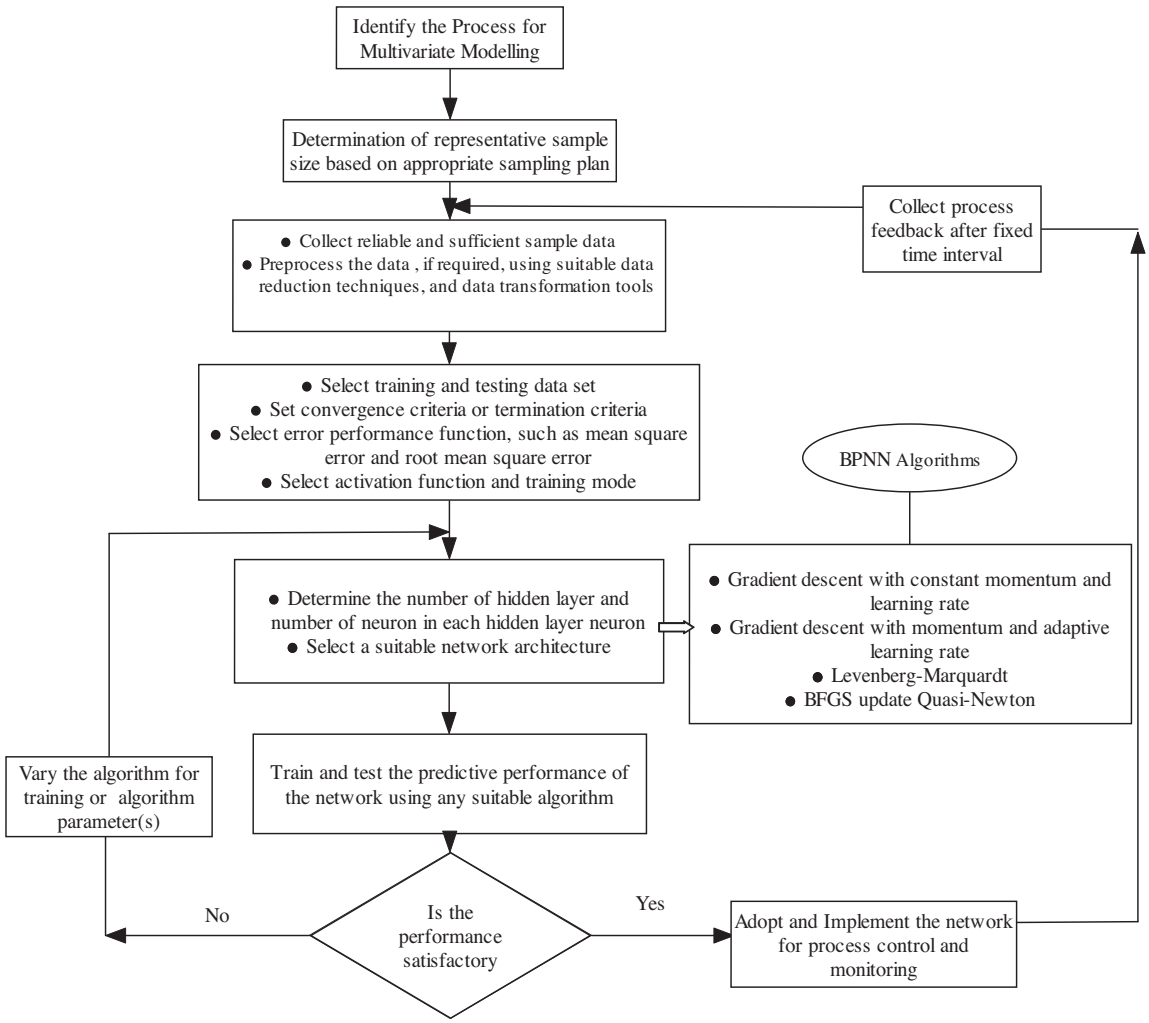


Fig. 2. A typical flow diagram for nonlinear multivariate process modelling using BPNN algorithm.

Table 1
Correlation matrix for the two-pass finish honing process responses.

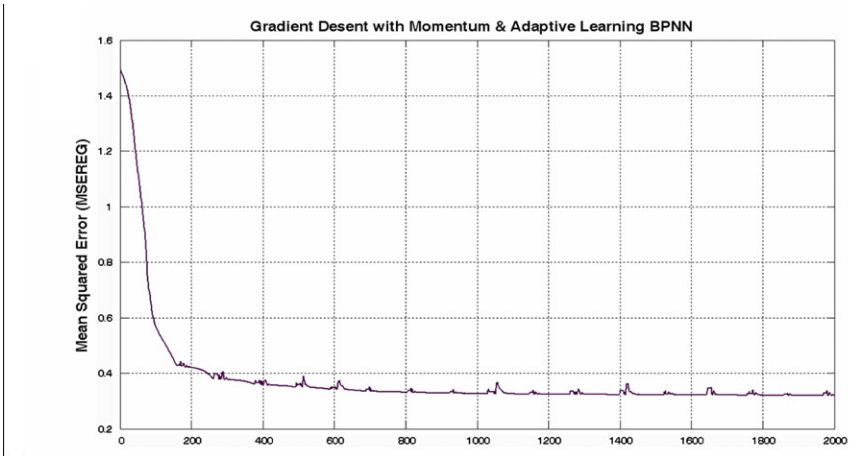
Response	Liner bore surface finish	Liner bore honing angle	Liner bore diameter	Liner bore ovality	Liner bore taper
Liner bore surface finish	1	0.440*	0.227*	−0.029	−0.067
Liner bore honing angle	0.440*	1	0.277*	−0.102	−0.236*
Liner bore diameter	0.227*	0.277*	1	−0.104	−0.123
Liner bore ovality	−0.029	−0.102	−0.104	1	0.540*
Liner bore taper	−0.067	−0.236*	−0.123	0.540*	1

* Pearson correlation is significant at 0.01 confidence level (2-tailed test).

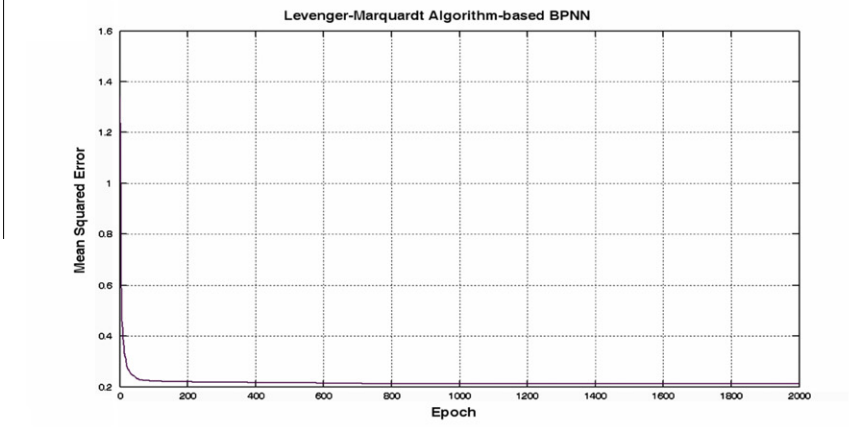
2.1.1. Variable or adaptive learning rate

Learning rate limits or expands the extent of weight adjustments in a training cycle. A high learning rate reacts quickly to input changes, and can make networks unstable. The changes can be too extreme, and cripple the network's prediction ability. In the contrary, if the learning rate is too low, the network training time is substantially increased. A high learning rate is useful to accelerate learning until the weight adjustments begin to reach plateau. However, the higher learning rate increases the risk that the weight search may jump over a minimum error condition. This could jeopardize the integrity of the network and cause back-propagation learning to reach global optima. The consensus among researchers is that adaptive learning rates stabilize risk of failure and accelerate the training process. A constant learning rate is inefficient as back-propagation needs a gradually declining rate of convergence. A low rate from the beginning is also time consuming. The correct setting for the learning rate is application dependent, and typically chosen by experiment. It can also be time-varying, or gradually decreases as the algorithm progresses to optimal solution.

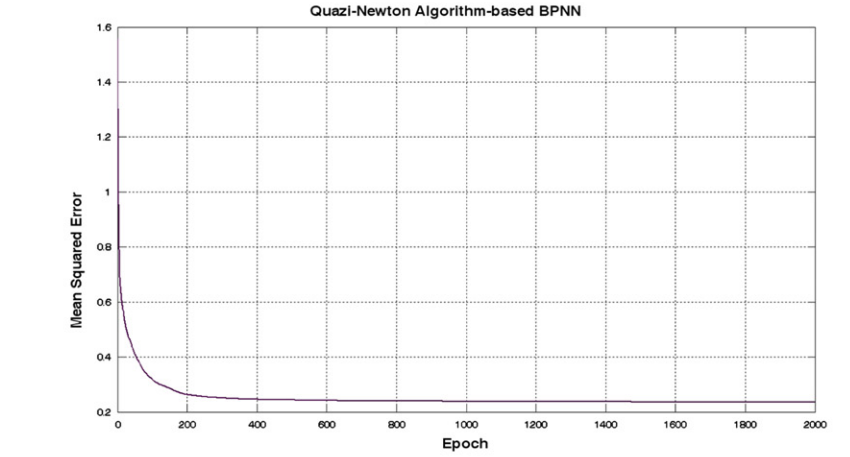
In addition, determining optimal setting of learning rate is near to impossible before training starts. It is in this context that increasing or decreasing the learning rate based on slope (flat or steep) of the error response surface may speed up the training and convergence of steepest decent algorithm. The variable or adaptive learning rate (Howard & Beale, 1998; Mukherjee & Ray, 2008c) increases the learning rate only to an extent that the network can learn without large error increase. It also decreases the learning rate as error decreases or until stable learning is resumed. In this particular study, A-L BPNN is implemented with the help of a built-in function, 'traingdx' provided in Matlab 6.5 programming environment. The variable or adaptive learning rate (Hagan et al., 2002; Howard & Beale, 1998) as selected at tth stage of epoch is determined based on two simple steps,



(a) Progress of Error Performance Measure during training for A-L BPNN



(b) Progress of Error Performance Measure during training for L-M BPNN



(c) Progress of Error Performance Measure during training for Q-N BPNN

Fig. 3. Progress of error performance measure using different algorithm for BPNN modelling.

(i) calculating the current network error at *i*th stage of iteration,
(ii) if ratio of new and old error exceeds a predefined value $e_{i(\max)}$, weights and biases of the network are discarded and reinitialized. In addition, the current learning rate is decreased by a multiple of $m_{i(decrease)}$, or the new weights calculated are retained. If the new error is less than the old error, the learning rate is increased by a multiple of $m_{i(increase)}$.

In this study, the values of $e_{i(\max)}$, $m_{i(decrease)}$, and $m_{i(increase)}$ are selected as 1.04, 0.7, and *1.05*, respectively (Hagan et al., 2002; Howard & Beale, 1998).

2.2. The Levenberg–Marquardt algorithm-based BPNN (L–M BPNN)

The Levenberg–Marquardt (L–M) algorithm (Bazaraa, Sherali, & Shetty, 2004; Madsen, Nielsen, & Tingleff, 2004; Marquardt, 1963)

outperforms simple gradient descent and many other conjugate gradient methods in a wide variety of problems (Ampazis & Perantonis, 2000; Dombayci & Golcu, 2009; Nalbant, Gokkaya, Toktas, & Sur, 2009; Torrecilla, Otero, & Sanz, 2007). L–M is a blend of local search properties of Gauss–Newton with consistent error decrease provided by gradient descent algorithm. The training of feed forward network based on L–M is considered as an unconstrained optimization problem. The main disadvantage of L–M algorithm is its increased memory requirement to calculate the Jacobian matrix of the error function. Determining the inverse of the matrix with dimensions equal to the number of the weights of neural network is cumbersome. Another disadvantage of L–M is that it does not always guarantee global optimum for a unconstrained optimization problem. The whole training process should be restarted, when the solution is unacceptable. In case of gradient descent algorithm, a momentum term is incorporated that helps to overshoot local minima. For a multiple output feed forward network, the mean square error $E(t)$ objective function is expressed as,

$$E(t) = \frac{1}{Np}\sum_{i=1}^{N}\sum_{k=1}^{k=p}\left[\left(d_k^i\right)^2 - \left(y_k^i\right)^2\right], \tag{7}$$

where y_k^i is the kth desired or target output activation of ith training sample. d_k^i is the kth actual output from the ith training pattern. u is the column vector containing all the weights, and thresholds of the network.The main idea is to use second order Taylor series expansion method. By this approach, the local approximation of the cost function is assumed quadratic, and expressed as

$$E(u_t + du_t) = E(u_t) + \nabla E(u_t)^T du_t + \frac{1}{2} du_t^T \nabla^2 E(u_t) du_t, \tag{8}$$

where $\nabla E(u_t)$ and $\nabla^2 E(u_t)$ are the gradient vector and matrix of second order derivative of cost function, respectively. The optimal step (or Newton step) du_t is obtained by the first optimality condition and expressed as

$$du_t = -[\nabla^2 E(u_t)]^{-1} \nabla E(u_t). \tag{9}$$

Due to the special form of Eq. (8), the second order derivative or Hessian matrix can also be written as,

$$\nabla^2 E(u_t) = \left(J_t^T J_t + S_t\right), \tag{10}$$

where $J_t$is the Jacobian matrix of first derivatives of the residuals $\left(d_k^i - y_k^i\right)^2$, and S_t denotes the second order derivative information of$\nabla^2 E(u_t)$. Ignoring the S_t term in Eq. (10) leads to Gauss–Newton method. In actual, S_t approaches zero as the algorithm converges (Ampazis & Perantonis, 2000). Thus, Gauss–Newton method can achieve the quadratic convergence of Newton's method by using only the information of its first derivatives. However, when algorithm is far away from optima, the term S_t cannot be negligible. Such approximation of the Hessian matrix can give poor results with slow convergence rate. The L–M method is based on the assumption that such an approximation is valid only within a trust region with small radius. This type of approximation for the Hessian matrix (Bazaraa et al., 2004) simplifies Eq. (10) as

$$\nabla^2 E(u_t) = \left(J_t^T J_t + \mu_t I\right), \tag{11}$$

where I is the identity unit matrix and μ_t is a scalar that (indirectly) controls the size of the trust region. When $\mu_t = 0$, the method becomes equivalent to the Gauss–Newton method. Whereas, for large μ_t, the L–M algorithm tends towards steepest descent algorithm. A common approach to implement L–M for training of neural networks is by selection a small μ_0. In subsequent iteration, μ_t can be adjusted according to following logic:

(i) If a successful step is taken [i.e. $E(u_t + du_t) < E(u_t)$], then μ_t is decreased by a factor of 0.1. This will lead the iteration towards Gauss–Newton direction.
(ii) if the step is unsuccessful [e.g. $E(u_t + du_t) > E(u_t)$], then μ_t is increased by a factor 10, until a successful step can be obtained.

It has been shown that L–M method does not guarantee global optimal and it is just a heuristic, that works extremely well in practical problems (Bazaraa et al., 2004). The main drawback of this algorithm is its computational complexity to calculate matrix inversion with variable size of few thousand. Generally, inverse of Hessian is implemented by pseudo-inverse method or singular value decomposition approach.

The 'trainlm' function in Matlab 6.5 is used to implement the L–M algorithm-based BPNN for this particular study. The value of μ_0 is selected as 0.001 and maximum value of μ_t is assumed to be 10^{10} for this study.

2.3. Boyden, Fletcher, Goldfarb and Shanno (BFGS) update Quasi-Newton algorithm-based BPNN (Q-N BPNN)

The gradient descent search requires only information of first partial derivative but can often gives poor numerical performance (Rardin, 2003). Gauss–Newton (G–N) algorithm yields improved convergence but require second derivative information and also to solve system of linear equations in each step.

Quasi-Newton algorithm (Brezinski, 2003; Oren, 1976; Robitaille, Marcos, Veillejte, & Payre, 1996; Shanno & Kettler, 1970; Xu & Zhang, 2001) is a blend of these two algorithms, and is based on the concept of conjugacy. If the objective function is unconstrained and quadratic in nature, then searching alone the conjugate directions of Hessian matrix can give a minimum point in n steps.

The Newton step size (Δx) for a second order Taylor series approximation of $f(x)$ at any current point x_i is obtained from the equation

$$H(x_i)\Delta x = -\nabla f(x_i). \tag{12}$$

Δx is further used to calculate x_{i+1}, and the equation is

$$x_{i+1} = x_i + \Delta x. \tag{13}$$

The search will continue until $\|\nabla f(x_i)\| < \varepsilon$, where ε is the stopping criteria or tolerance. In this study, ε is selected as 10^{-6}.

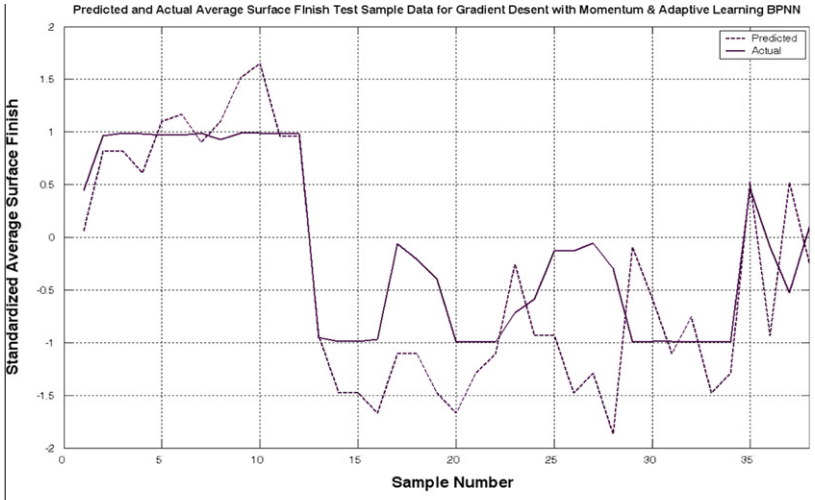
Assuming that the Hessian matrix is non-singular, Eq. (12) can also be rewritten as

$$\Delta x = -H(x_i)^{-1} \nabla f(x_i), \tag{14}$$

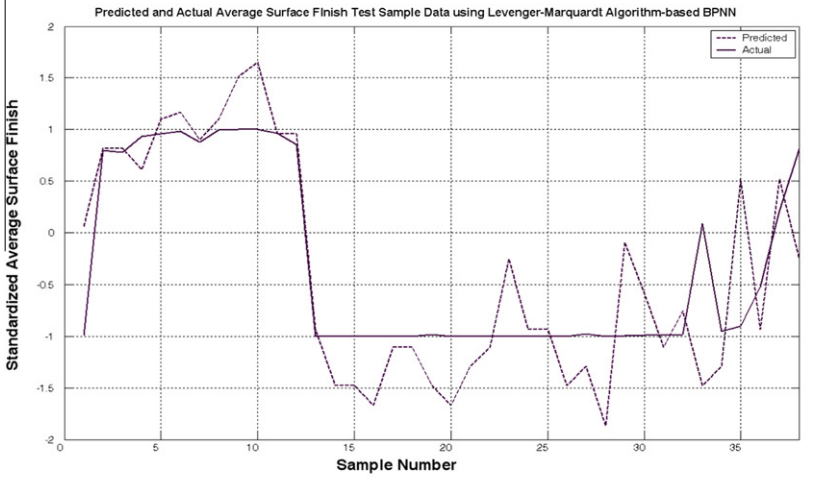
where $D_i = H(x_i)^{-1}$ is so-called the 'deflection matrix'.

Table 2
Mean square error and network training time for BPNN algorithms.

BPNN algorithm	Mean square errors	Network training time (sec) per epoch
Gradient descent with momentum and adaptive learning (A-L)	0.3186	0.00203
Levenberg–Marquardt (L–M)	0.2117	0.1022
BFGS update Quasi-Newton (Q-N)	0.2356	0.0234



(a) Correlation Co-efficient $r = 0.834$ using A-L BPNN



(b) Correlation Co-efficient $r = 0.845$ using L-M BPNN

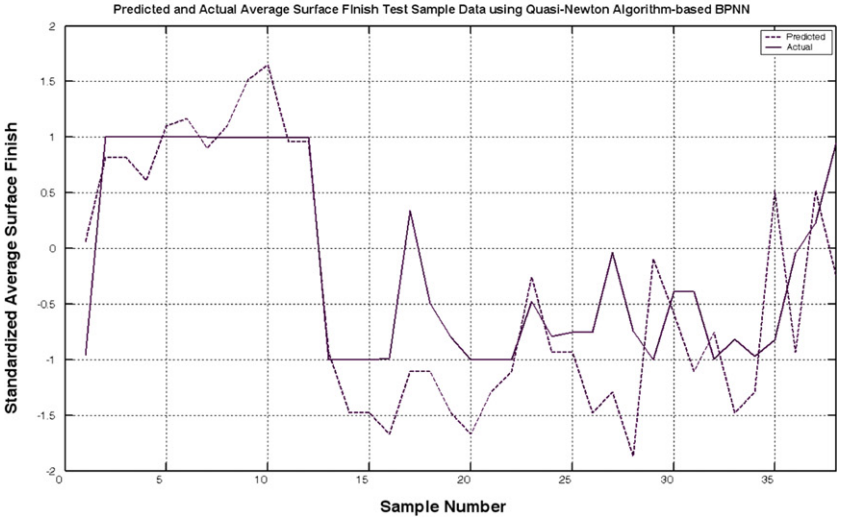


Fig. 4. Plot of actual and predicted test sample responses using (a) A-L-based BPNN, (b) L–M-based BPNN, (c) BFGS update Q-N-based BPNN.

The deflection matrix should assure improved directions by keeping D_i positive definite for a minimization problem. Various updating formula exist, that can meet the Quasi Newton requirement. One such expression is,

$$D^{*}_{i+1}g = d, \tag{15}$$

where d is the difference between x_{i+1} and x_i, and g is the difference between $\nabla f(x_{i+1})$ and $\nabla f(x_i)$. However, the most efficient update is proposed by Broyden, Fletcher, Goldfarb, and Shanno, so called BFGS formula (Rardin, 2003). In BFGS, the updated deflection matrix is expressed as,

$$D_{i+1} = D_i + \left(1 + \frac{g \cdot D_i \cdot g}{d \cdot g}\right)\frac{d \cdot d^T}{d \cdot g} - \frac{D_i \cdot g \cdot d^T + d \cdot g^T \cdot D_i}{d \cdot g}. \tag{16}$$

The BFGS method for minimization problem employs identity deflection matrix for initial iteration and is expressed as

$$\Delta x^{(1)} = -I \cdot \nabla f(x^0), \tag{17}$$

where I is the identity matrix. Subsequently, it proceeds by line search approximation and computes the step size,λ_{i+1} by minimizing $f(x_i + \lambda_i \Delta x_{i+1})$. It updates by using the following equation,

$$x_{i+1} = x_i + \lambda_{i+1}\Delta x_{i+1}. \tag{18}$$

The 'trainbfg' function in Matlab 6.5 is used to implement the BFGS update Q-N algorithm in this study.

A typical flow diagram for implementing back propagation algorithm-based neural network for nonlinear multiple response process modelling is illustrated in Fig. 2.

A typical grinding (honing) real life case is selected for the comparative study of the three algorithms. The details of the case and analysis are given in the following section.

3. Case example

The performances of algorithms are compared based on multivariate modelling of a two-pass, hydraulic oil pressure-based finish grinding (honing) process of six-cylinder automobile engine liner bore. The process data was collected from a machining unit of a largest automobile manufacturing plant, located in eastern India. The manufacturer produces wide varieties of automobiles, ranging from passenger cars to heavy duty commercial vehicles.

3.1. Background of the case

Honing is a critical mechanical abrasion machining process, which generally coming under the broad category of grinding. It can provide required dimensional accuracy and surface quality that are created by processes of lesser accuracy, such as casting, boring, and turning. Details of honing process and its critical process characteristics can be found in various literatures (El-Hofy, 2005; Fishcher, 1986, chap. 13). During honing, the geometrically undefined and random directional multi-edge abrasive stone carries out cutting operation. Stone expansion, rotation, and stroking provide the required surface finish, crosshatch lay direction, and dimensional accuracy of a typical engine cylinder bore. However, despite its importance and popularity, honing still remains one of the most difficult and least-understood processes (Feng et al., 2002).

In this particular case, the diesel engine manufacturing unit of the automobile industry is currently facing intense pressure from its competitors to deliver engine of higher quality and prolonged economic life. Also the industry wants to reduce smoke emission, as standardized by the international regulating authorities. In this context, finished cylinder block liner is one of the most critical engine components characterized by its internal surface quality having a bearing on the level of smoke emission, lubrication oil consumption, and longevity of diesel engine (Feng et al., 2002). Internal surface quality characteristics, such as surface finish, honing angle (or cross-hatch angle), surface diameter, ovality and taper also affect the lubrication oil retention property at high engine combustion temperature. The surface characteristics also influence the resistance to wear.

However, preliminary sample data analysis of the cylinder liners show that the engine division is facing acute problem of simultaneously maintaining and controlling the desired surface finish, cross-hatch angle, and dimensional accuracy of the liner. This is resulting in harmful side effects, such as high smoke emission, and lubrication oil consumption. Engine lubrication oil consumption remains at a level 20% more than the defined target value. Excessive rejects and scraps (5–7% of total production quantity in a month) is also an issue of concern. Data analysis also reveals that there are significant correlations between the selected five critical response characteristics. The Pearson correlation coefficient matrix, using 2-tailed test statistic, for response variables of two-pass finish honing process is provided in Table 1.

From the table, it is clear that there is a presence of both negative and positive correlation between the responses. Therefore, multivariate response surface modelling of grinding behaviour can be attempted.

In this case, the objective is to find the empirical relationship between five response characteristics (viz. average surface finish, average honing angle, average diameter, ovality, and taper of liner bore), three input variables (viz. initial liner bore average diameter liner bore ovality and liner bore taper), and seven in-process variables (viz. rough-pass hydraulic pressure, finished-pass hydraulic pressure, rough-pass honing time, finished-pass honing time, cutting oil temperature, vertical stroke speed, and dog length). All the independent and dependent variables are continuous in this case study. A brief discussion on data collection approach is given in the following subsection.

3.2. Data collection

The input variables, in-process parameters, and responses of the honing processes are identified for the process after thorough discussion with the concerned personnel, reference to the relevant documents and standard operating practices of the machine centres. Pertinent and reliable real-time production data are collected through direct observations, and based on appropriate sampling plan. It was assumed that the production environment do not experience long-term process shift and there is no deficit with regard to certain ranges or interactions. Data are collected at different time points based on *ANSI/ASQC Z* 1.4 attribute sampling plan (Mitra, 2001; Montgomery, 1991).The data are collected following the principal of rational sub grouping. Various data prepossessing techniques (data transformation and data scaling) are applied to facilitate subsequent empirical model development. Interested readers can find a detailed discussion on these techniques and application in Mukherjee and Ray (2008a).

4. Neural network architectures design and comparative analysis of algorithms

The simulation run of the three algorithms is performed in a personal computer having configuration as Intel Pentium (IV) processor, 1.8 GHz CPU, and compatible to 256 MB RAM. While implementing all the three algorithm, the total sample data set is divided into training and testing (untrained) half in a 80–20% ratio (Coit et al., 1998; Mukherjee & Ray, 2008b). The training data set is selected in such a way, so that test data lies within the state space boundary of training data set (Basheer & Hajmeer, 2000). For MSEREG function, γ_i is selected as 0.5 (Basheer & Hajmeer, 2000). Matlab (version 6.5) toolbox is used to train and test the proposed network. The single BPNN model architecture selected for the two-pass finish honing process is having 10 input layer (inputs and in-

Table 3
Correlation coefficient (r) of predicted outputs with actual outputs using test sample.

Algorithm	Avg. surface finish (Y_1)	Avg. honing angle (Y_2)	Avg. liner bore diameter (Y_3)	Liner bore ovality (Y_4)	Liner bore taper (Y_5)
Responses					
A-L BPNN	0.834	0.715	0.779	0.247	0.042
L–M BPNN	0.845	0.625	0.341	0.283	0.42
BFGS update Q-N BPNN	0.818	0.446	0.687	0.376	0.193

Table 4a
Summary statistics of r-values.

Summary statistic	Count	Average	Variance
A-L BPNN	5	0.5234	0.1266
L–M BPNN	5	0.5028	0.0533
BFGS update Q-N BPNN	5	0.5040	0.0621
Avg. surface finish (Y_1)	3	0.8323	0.0001
Avg. honing angle (Y_2)	3	0.5953	0.0187
Avg. diameter (Y_3)	3	0.6023	0.0533
Ovality (Y_4)	3	0.3020	0.0044
Taper (Y_5)	3	0.2183	0.0362

Table 4b
A two-factor analysis of variance without replication using r-values.

Source of variation	Sum of square	Degree of freedom	Mean sum of square	F-calculated	P-value	F-tabulated
Algorithms	0.0013	2	0.0007	0.023**	0.976**	4.458
Response variables (Y's)	0.7441	4	0.1860	6.629*	0.011*	3.837
Error	0.2244	8	0.0281			
Total	0.9699	14				

* Statistically significant at $\alpha = 0.05$ (level of significance).
** Not statistically significant at $\alpha = 0.05$ (level of significance).

Table 5a
Error performance measures for individual responses using test sample data and A-L BPNN.

Training algorithm	Different measures of error	Based on test sample data				
		Avg. surface finish (Y_1)	Avg. honing angle (Y_2)	Avg. liner bore diameter (Y_3)	Liner bore ovality (Y_4)	Liner bore taper (Y_5)
A-L BPNN	AAD	0.5048	1.1333	1.5834	2.5112	3.4866
	MSE	0.1474	0.7404	0.3757	0.2541	0.0117
	RMSE	0.3839	0.8605	0.6129	0.5041	0.1081
	MPE (%)	0.6893	0.8422	4.9332	1.3916	2.5700
	MAPE (%)	0.6903	0.8304	0.7643	0.8689	0.9782
	MRE (%)	1.2856	2.6991	4.1717	5.9822	8.0052

process parameters), 5 output layer (responses), and 20 neurons in single hidden layer. Selection of number of hidden layer and neurons in any particular hidden layer is purely based on trial runs, and can be debated. The network adopted for this comparative study is purely based on trial-by-trial method. Different algorithms with varied number of neurons are tested to achieve a level of model accuracy. There is no single approach that can be used to determine the best network for all situations. The data analysis

Table 5b
Error performance measures for individual responses using test sample data and L–M BPNN.

Training algorithm	Different measures of error	Based on test sample data				
		Avg. surface finish (Y_1)	Avg. honing angle (Y_2)	Avg. liner bore diameter (Y_3)	Liner bore ovality (Y_4)	Liner bore taper (Y_5)
L–M BPNN	AAD	0.4199	1.0531	1.8424	2.7004	3.4103
	MSE	1.1260	0.9792	0.0145	0.2612	0.0080
	RMSE	1.0611	0.9895	0.1204	0.5111	0.0897
	MPE (%)	0.1313	1.5917	0.2107	1.4178	1.1153
	MAPE (%)	0.4619	0.5638	0.6968	0.7645	0.7683
	MRE (%)	2.0218	4.0437	6.0656	8.0875	10.1093

Table 5c
Error performance measures for individual responses using test sample data and BFGS update Q-N BPNN.

Training algorithm	Different measures of error	Based on test sample data				
		Avg. surface finish (Y_1)	Avg. honing angle (Y_2)	Avg. liner bore diameter (Y_3)	Liner bore ovality (Y_4)	Liner bore taper (Y_5)
BFGS update Q-N BPNN	AAD	0.5157	1.3498	1.8358	2.5695	3.4453
	MSE	1.0688	1.1275	0.0146	0.2612	4.3668
	RMSE	1.0338	1.0618	0.1211	0.5111	2.0897
	MPE (%)	0.5969	1.6419	5.1572	0.1608	1.3165
	MAPE (%)	0.6472	0.8262	0.7701	0.7865	0.8306
	MRE (%)	2.1858	4.3717	6.5576	8.7434	10.9293

and interpretation of results after training and testing of all the networks is given below.

4.1. Training and testing of network architectures

The network is trained in batch training mode, and the progress of the performance measure, mean square error, for A-L BPNN, L–M BPNN, and BFGS update Q-N BPNN algorithm is shown in Fig. 3(a–c).

4.2. Comparative results and analysis

Table 2 provides the mean square error measure, and network training time per epoch for BPNN model using the three different algorithms.

From Table 2 and Fig. 3, it is clear that back propagation learning using L–M algorithm converges faster than A-L and Q-N algorithm. The training *MSE* values also indicate that L–M BPNN provides a more accurate nonlinear predictive model, followed by Q-N method, and lastly A-L BPNN. However, A-L BPNN takes the minimum time for network training. BFGS update Q-N required about 10 times and L–M requires about 50 times more computational time than A-L-based BPNN.

Fig. 4(a–c) provides the plot of actual and predicted test sample response for the three different algorithms.

Table 3 provide the correlation co-efficient (r) for all the five responses using test sample data and any specific algorithm.

The individual r-values of five responses do not provide a clear indication about the superiority of any particular algorithm. A-L BPNN predicts Y_1, Y_2, and Y_3 comparatively better than L–M or Q-N-based BPNN. L–M predicts Y_1 and Y_5 better than A-L BPNN and Q-N BPNN.

Statistical analysis can provide better insight to identify if there is any significant difference in the predictive performance of these three algorithms. A two-factor analysis of variance (ANOVA) with-

Table 6
A brief summary of the training algorithm.

Intrinsic details\algorithm	A-L BPNN	L–M BPNN	BFGS update Q-N BPNN
Optimization technique	Gradient descent search	Blend of gradient descent and Gauss–Newton based search	Conjugate gradient based search and BFGS update rule for deflection matrix
Performance function	A modified performance, so called MSEREG	Mean square error	Mean square error
Mode of training	Batch	Batch	Batch
Convergence	Fast	Fastest	Faster
Network training time per Epoch	Minimum	Maximum	Medium
Rate/size	Variable learning rate	A scalar, μ_t, that (indirectly) controls the size of the trust region	Step size and deflection matrix
Memory storage requirement	Small	Large	Large
Suitability for multiple response problem	Suitable	Suitable	Suitable
Online dynamic process control	Best as learning is very fast	Can be used if computational speed of learning is high	Can be used if computational speed of learning is high

out replication is performed on the *r*-values. The summary statistics and two-way ANOVA results are provided in Tables 4a and 4b.

The two-way ANOVA analysis indicates that the correlation coefficient (r) does not show a significant difference when different algorithms are used predicting the given set of test data. In other words, all the three algorithms (A-L BPNN, L–M BPNN, and BFGS update Q-N BPNN) are having same efficiency to predict test data set. However, the statistically significant result on *r*-values for response variables (Y's) is obvious, as *MSE* values will vary for different Y's. However, seeing the comparative *r*-values for Y_1, Y_2 and Y_3, and time taken to training the network, A-L-based BPNN can be preferred. In addition, understanding the A-L BPNN may be easier as compared to L–M and BFGS update Q-N-based BPNN. In a dynamic online process control environment, A-L BPNN may be preferred.

Tables 5a–5c also provides various error performance measures for each responses, using the test sample data. The different error measures reported in the table are expressed as,

$$\text{Average Absolute Deviation}: AAD = \frac{1}{N}\sum_{t=1}^{N}|y(t)-y_d(t)|, \tag{19}$$

$$\text{Mean Square Error}: MSE = \frac{1}{N}\sum_{t=1}^{N}(y(t)-y_d(t))^2, \tag{20}$$

$$\text{Root Mean Square Error}: RMSE = \sqrt{MSE}, \tag{21}$$

$$\text{Mean Percentage Error}: MPE = \frac{1}{N}\sum_{i=1}^{N}\frac{[y(t)-y_d(t)]}{y_d(t)}, \tag{22}$$

$$\text{Mean Absolute Percentage Error}: MAPE = \frac{1}{N}\sum_{i=1}^{N}\left|\frac{[y(t)-y_d(t)]}{y_d(t)}\right|, \text{ and} \tag{23}$$

$$\text{Mean Relative Error}: MRE = \frac{100}{N}\sum_{i=1}^{N}\frac{|y(t)-y_d(t)|}{y_d(t)}. \tag{24}$$

Here, $y(t)$ is the actual output, $y_d(t)$ is the predicted output of the trained network, and N is the total number of patterns used for training.

All the measures are provided as there is no single consensus on which measure is most appropriate to compare different algorithms. Based on the overall theoretical understanding and case analysis using the three algorithms, a summary is provided in Table 6.

5. Conclusions

In this paper, two different training algorithms, viz. Levenberg–Marquardt (L–M), and BFGS update Quasi-Newton is compared with gradient descent algorithm to develop a BPNN model for nonlinear modelling multiple response honing process. The comparative analysis is based on actual production data collected over a period of time. The key finding that came out from this study is:

(i) L–M and Q-N algorithm-based BPNN networks are equally efficient as A-L algorithm-based BPNN network to predict the behaviour of multiple response grinding process.
(ii) L–M algorithm has fastest network convergence rate, followed by BFGS update Q-N and A-L algorithm.
(iii) A-L-based BPNN learns faster than BFGS update Q-N, and L–M takes the maximum time for training, and
(iv) A-L algorithm is relatively easy-to-understand and implement as compared to L–M or BFGS update Q-N algorithm for online process control.

The future scopes of research are:

(i) Verifying the suitability of L–M and BFGS update Q-N algorithm for predicting varied other cutting situations.
(ii) Incorporating experimental data for robust variability modelling and compare the algorithms performances.
(iii) Develop a framework to determine optimal network configuration for such type of comparative studies.

Acknowledgements

The authors of this paper would like to acknowledge the support and assistance provided for data collection by Quality Assurance (Engine Division), and Production (Engine Division) Department of Tata Motors Ltd., Jamshedpur, India.

References

Ampazis, N., & Perantonis, S. J. (2000). Levenberg–Marquardt algorithm with adaptive momentum for the efficient training of feedforward networks. In *Proceedings of IEEE & INNS international joint conference on neural networks, IJCNN 2000, paper no. NN0401, Como, Italy, (CD proceedings)* (pp. 126–131).
Basheer, I. A., & Hajmeer, M. (2000). Artificial neural networks: Fundamentals, computing, design, and application. *Journal of Microbiological Methods, 43*, 3–31.
Bates, D. M., & Watts, D. G. (1988). *Nonlinear regression analysis and its applications*. New York: John Wiley and Sons.
Bazaraa, M. S., Sherali, H. D., & Shetty, C. M. (2004). *Nonlinear programming. Theory and algorithms* (2nd ed.,). India: Wiley.
Box, G. E. P., & Draper, N. R. (1987). *Empirical model-building and response surface*. New York: John Willey and Sons.
Brezinski, C. (2003). A classification of quasi-Newton methods. *Numerical Algorithms, 33*, 123–135.
Chen, H., Lin, J., Yang, Y., & Tsai, C. (2010). Optimization of wire electrical discharge machining for pure tungsten using a neural network integrated simulated annealing approach. *Expert Systems with Applications, 37*, 7147–7153.

Chen, Y. T., & Kumara, S. R. T. (1998). Fuzzy logic and neural network for design of process parameters: A grinding process application. *International Journal of Production Research, 36*(2), 395–415.

Choi, T. J., Subrahmanya, N., Li, H., & Shin, C. Y. (2008). Generalized practical models of cylindrical plunge grinding processes. *International Journal of Machine Tools and Manufacture, 48*, 61–72.

Coit, D. W., Jackson, B. T., & Smith, A. E. (1998). Static neural network process models: Considerations and case studies. *International Journal of Production Research, 36*(11), 2953–2967.

Cus, F., Zuperl, U., & Milfelner, M. (2006). Dynamic neural network approach for tool cutting force modelling of end milling operations. *International Journal of General Systems, 35*(5), 603–618.

Dombayci, O. A., & Golcu, M. (2009). Daily means ambient temperature prediction using artificial neural network method: A case study of Turkey. *Renewable Energy, 34*(4), 1158–1161.

El-Hofy, H. (2005). *Advanced machining processes: Nontraditional and hybrid machining processes*. New York: MacGraw-Hill.

Farago, F. T. (1976). *Abrasive methods engineering*. NY: Industrial Press Inc.

Feng, C., Wang, X., & Yu, Z. (2002). Neural networks modelling of honing surface roughness parameter defined by ISO 13565. *SIAM Journal of Manufacturing Systems, 21*(8), 1–35.

Fishcher, H. (1986). In R. I. King & R. S. Hahn (Eds.), *Honing in handbook of modern grinding technology* (pp. 301–336). New York: Chapman and Hall.

Fu, L. (2003). *Neural network in computer intelligence*. India: Tata McGraw Hill.

Govindhasamy, J. J., MacLoone, S. F., Irwin, G. W., French, J. J., & Doyle, R. P. (2005). Neural modelling, control and optimization of an industrial grinding process. *Control Engineering Practice, 13*, 1243–1258.

Haber, R. E., & Alique, A. (2003). Intelligent process supervision for predicting tool wear in machining processes. *Mechatronics, 13*, 825–849.

Hagan, M. T., Demuth, H. B., & Beale, M. (2002). *Neural network design*. India: Thomson Learning.

Howard, D., & Beale, M. (1998). *Neural network toolbox: For use with Matlab; user's Guide, version 3*. USA: The Mathworks Inc.

Jain, R. K., Jain, V. K., & Kalra, P. K. (1999). Modelling of abrasive flow machining process: A neural network approach. *Wear, 231*, 242–248.

Krajnik, P., Kopac, J., & Sluga, A. (2005). Design of grinding factors based on response surface methodology. *Journal of Materials Processing Technology*, 162–163. 629-636.

King, R. I., & Hahn, R. S. (1986). *Handbook of modern grinding technology*. New York: Chapman and Hall.

Kumar, S., & Choudhury, S. K. (2007). Prediction of wear and surface roughness in electro-discharge diamond grinding. *Journal of Materials Processing Technology, 191*, 206–209.

Kwak, J. (2005). Application of Taguchi and response surface methodologies for geometric error in surface grinding process. *International Journal of Machine Tools and Manufacturing, 45*, 327–334.

Luong, L. H. S., & Spedding, T. A. (2002). A neural-network system for predicting machining behaviour. *Journal of Materials Processing Technology, 52*, 585–591.

Madsen, K., Nielsen, H. B., & Tingleff, O. (2004). *Methods for non-linear least squares problems* (2nd ed.). Technical University of Denmark.

Maksoud, T. M. A., & Atia, M. R. (2004). Review of intelligent grinding and dressing operations. *Machine Science and Technology, 8*(2), 263–276.

Malkin, S. (2007). Thermal analysis of grinding. *Annals of CIRP, 56*(2), 760–782.

Malkin, S. (1984). Grinding of metals: Theory and application. *Journal of Applied Metalwork, 3*, 95–109.

Markos, S., Viharos, Zs. J., & Monostori, L. (1998). Quality-oriented, comprehensive modelling of machining processes. In *6th ISMQC IMEKO symposium on metrology for quality control in production* (pp. 67–74).

Marquardt, D. W. (1963). An algorithm for least-squares estimation of nonlinear parameters. *Journal Society Industrial Applied Mathematics, 11*(2), 431–441.

Mitra, A. (2001). *Fundamentals of quality control and improvement* (2nd ed.). India: Pearson Education Asia, Inc.

Montgomery, D. C. (1991). *Introduction to statistical quality control* (2nd ed.). New York: John Wiley and Sons.

Montgomery, D. C. (2001). *Design and analysis of experiment* (5th ed.). New York: Wiley and sons Inc.

Montgomery, D. C., & Peck, E. A. (1992). *Introduction to linear regression analysis* (2nd ed.). New York: John Wiley and Sons, Inc.

Moulik, P. N., Yang, H. T. Y., & Chandrasekhar, S. (2001). Simulation of thermal stresses due to grinding, international. *Journal of Mechanical Society, 43*, 831–851.

Mukherjee, I. (2007). *Modelling and optimization of abrasive metal cutting processes*. Ph.D. Thesis, Department of Industrial Engineering and Management. Indian Institute of Technology, Kharagpur, India.

Mukherjee, I., & Ray, P. K. (2006). A review of optimization techniques in metal cutting processes. *International Journal of Computers and Industrial Engineering, 50*, 15–34.

Mukherjee, I., & Ray, P. K. (2008a). A systematic solution methodology for inferential multivariate modelling of industrial grinding process. *International Journal of Materials Processing Technology, 196*, 379–392.

Mukherjee, I., & Ray, P. K. (2008b). Technology management of grinding process using quality control methods: A multivariate regression and neural network application. In *Innovation and Technology Management Advanced Research Series* (pp. 238–246). India: Macmillan.

Mukherjee, I., & Ray, P. K. (2008c). A modified tabu search strategy for multiple-response grinding process optimization. *International Journal of Intelligent Systems Technologies and Applications, 4*(1/2), 97–122.

Nalbant, M., Gokkaya, H., Toktas, I., & Sur, G. (2009). The experimental investigation of the effects of uncoated, PVD- and CVD-coated cemented carbide inserts and cutting parameters on surface roughness in CNC turning and its prediction using artificial neural networks. *Robotics and Computer-Integrated Manufacturing, 25*(1), 211–223.

Oren, S. S. (1976). On quasi-Newton and pseudo-Newton algorithms. *Journal of Optimization Theory and Applications, 20*(2), 155–170.

Petri, K. L., Billo, R. E., & Bidanda, B. (1998). A neural network process model for abrasive flow machining operations. *Journal of Manufacturing Systems, 17*(1), 52–64.

Rardin, R. L. (2003). *Optimization in operations research*. India: Pearson Education.

Rencher, A. C. (1995). *Method of multivariate analysis*. USA: John Wiley and Sons, Inc.

Rencher, A. C. (1998). *Multivariate statistical inference and applications*. New York: John Wiley and Sons, Inc.

Rumelhart, D. E., Hilton, G. E., & Williams, R. J. (1986). Learning representations by back-propagating errors. *Nature, 323*(9), 533–536.

Robitaille, B., Marcos, B., Veillejte, M., & Payre, G. (1996). Modified quasi-Newton methods for training neural networks. *Computers and Chemical Engineering, 20*(9), 1133–1140.

Rowe, B. W., Yan, L. I., Inasaki, I., & Malkin, S. (1994). Applications of artificial intelligence in grinding. *Annals of CIRP, 43*(2), 521–531.

Sathyanarayanan, G., Lin, I. J., & Chen, M. (1992). Neural network modelling and multiobjective optimization of creep feed grinding of superalloys. *International Journal of Production Research, 30*(10), 2421–2438.

Sedighi, M., & Afshari, D. (2010). Creep feed grinding optimization by an integrated GA-NN system. *Journal of Intelligent Manufacturing, 21*(6), 657–663.

Shaji, S., & Radhakrisnan, V. (2003). Analysis of process parameters in surface grinding with graphite as lubricant based on Taguchi method. *Journal of Material Processing Technology*, 1–9.

Shanno, D. F., & Kettler, P. C. (1970). Optimal conditioning of quasi-newton methods. *Mathematics of Computation, 24*(111), 657–664.

Shin, Y. C., & Vishnupad, P. (1996). Neuro-fuzzy control of complex manufacturing processes. *International Journal of Production Research, 34*(12), 3291–3309.

Sorensen, P. H., Norgaard, M., Ravn, O., & Poulsen, N. K. (1999). Implementation of neural network based non-linear predictive control. *Neurocomputing, 28*, 37–51.

Tawakoli, T., Rabiey, M., & Lewis, M. (2009). Neural Network prediction of dry grinding by CBN grinding wheel using special conditioning. *International Journal of Materials and Product Technology, 35*(1/2), 118–133.

Torrecilla, J. S., Otero, L., & Sanz, P. D. (2007). Optimization of an artificial neural network for thermal/pressure food processing: Evaluation of training algorithms. *Computers and Electronics in Agriculture, 56*, 101–110.

Tsai, K., & Wang, P. (2001). Predictions of surface finish in electrical discharge machine based upon neural network models. *International Journal of Machine Tools and Manufacture, 41*, 1385–1403.

Urbaniak, M. (2004). Evaluation system for the cutting properties of grinding wheels. *Proceedings of the Institution of Mechanical Engineers, Part B: Journal of Engineering Manufacture, 218*(11), 1491–1498.

Vafaeesefat, A. (2009). Optimum creep feed grinding process conditions for rene 80 supper alloy using neural network. *International Journal of Precision Engineering and Manufacturing, 10*(3), 5–11.

Wasserman, P. D. (1993). *Advanced methods in neural computing*. New York: Van Nostrand Reinhold.

Widrow, B., & Lehr, M. A. (1990). 30 Years of adaptive neural networks: Perceptron, Madaline, and back propagation. *Proceedings of IEEE, 78*(9), 1415–1442.

Xu, C., & Zhang, J. (2001). A survey of quasi-Newton equations and quasi-newton methods for optimization. *Annals of Operations Research, 103*, 213–234.

Zadeh, L. A. (1965). Fuzzy sets. *Information and Control, 8*, 338–353.

Zhang, H. C., & Huang, S. H. (1995). Application of neural network in manufacturing – a state of art survey. *International Journal of Production Research, 33*(3), 705–728.