Neural network model of heat and fluid flow in gas metal arc fillet welding based on genetic algorithm and conjugate gradient optimisation

A. Kumar*[1] and T. DebRoy

Although numerical calculations of heat transfer and fluid flow can provide detailed insights into welding processes and welded materials, these calculations are complex and unsuitable in situations where rapid calculations are needed. A recourse is to train and validate a neural network, using results from a well tested heat and fluid flow model to significantly expedite calculations and ensure that the computed results conform to the basic laws of conservation of mass, momentum and energy. Seven feedforward neural networks were developed for gas metal arc (GMA) fillet welding, one each for predicting penetration, leg length, throat, weld pool length, cooling time between 800°C and 500°C, maximum velocity and peak temperature in the weld pool. Each model considered 22 inputs that included all the welding variables, such as current, voltage, welding speed, wire radius, wire feed rate, arc efficiency, arc radius, power distribution, and material properties such as thermal conductivity, specific heat and temperature coefficient of surface tension. The weights in the neural network models were calculated using the conjugate gradient (CG) method and by a hybrid optimisation scheme involving the CG method and a genetic algorithm (GA). The neural network produced by the hybrid optimisation model produced better results than the networks based on the CG method with various sets of randomised initial weights. The CG method alone was unable to find the best optimal weights for achieving low errors. The hybrid optimisation scheme helped in finding optimal weights through a global search, as evidenced by good agreement between all the outputs from the neural networks and the corresponding results from the heat and fluid flow model.

Keywords: Neural network, Gas metal arc welding, Optimisation, Fillet welds

Introduction

Since the geometry, composition and structure of welds are affected by the welding variables, these variables are often adjusted by trial and error to achieve defect-free, structurally sound and reliable welds. However, this approach is time consuming and expensive, and does not always provide optimum welds. Attempts have been made to obtain systematic correlations between welding variables and weld characteristics by the use of statistical regression analysis,[1] artificial neural networks[1–13] and phenomenological modelling.[14–24] In principle, regression analysis can relate weld pool geometry to welding variables with the use of a large volume of experimental data. However, this approach is difficult for gas metal arc (GMA) fillet welding, due to complex interactions between various physical processes, where each variable affects the weld pool dimensions, cooling rate and other parameters in a complex manner.[17–20] In recent years, numerical models of heat transfer and fluid flow in fusion welding have provided significant quantitative insights into both the welding processes and the welded materials. The computed temperature and velocity fields, cooling rates, weld pool geometry for various concentrations of surface active elements, concentrations of oxygen, nitrogen and hydrogen, and formation of defects, have been studied quantitatively using numerical models. Although these models are recognised as powerful tools for research, they are not extensively used in the welding industry, because the models are highly complex, require specialised training to develop and test, and consume a large amount of computer time.

The neural network models are capable of relating input variables such as welding process parameters and material properties to weld characteristics such as the weld geometry[1–5] and properties. Previous efforts to model the GMA fillet welding process using neural networks were based on training the network with experimental data. Since the volume of experimental

Department of Materials Science and Engineering, The Pennsylvania State University, University Park, PA, USA

*Corresponding author, email amitkumar@psu.edu

Published by Maney on behalf of the Institute
Received 12 August 2005; accepted 13 October 2005
DOI 10.1179/174329306X84319

data required to train a neural network depends on the number of input and output variables, most previous workers considered only few input parameters to keep the necessary volume of experimental data tractable.[1–4] For example, Kim *et al.*,[1] Smartt and Johnson,[2] Cook *et al.*[3] and Li *et al.*[4] developed neural network models of the GMA welding process in which the effects of process parameters such as welding speed, arc voltage and arc current were considered as inputs. Since the weld pool geometry depends on other welding variables, as well as various material properties, the effects of many of the welding variables and material properties cannot be determined from the available neural networks. Furthermore, the output variables used in the previous neural networks were also limited. For example, the existing neural network models do not provide any information about some of the important parameters, such as the cooling rate and peak temperature. A review of previous work indicates that what is needed is a framework for rapid calculation of weld pool geometry, cooling rate and peak temperature in the weld pool for GMA welding of various materials.

A neural network trained with the results of a numerical heat transfer and fluid flow model can correlate various output variables, such as the weld pool geometry, cooling rate, liquid velocities and peak temperatures, with all the major welding variables and material properties. Furthermore, such correlations satisfy the phenomenological laws. With the improvements in computational hardware in recent years, a large volume of training and validation data can be generated with a well tested numerical heat transfer and fluid flow model in a realistic time frame. The weights in the neural network were calculated using a hybrid optimisation scheme involving the conjugate gradient (CG) method and a genetic algorithm (GA). The neural network produced by the hybrid optimisation model produced better results than the networks based on the CG method. This paper seeks to document the problems, issues and lessons learnt in the development of a neural network model from the results of a heat transfer and fluid flow model that considers all the major input variables, including all important process variables and material properties, and correlates them with output variables.

Mathematical model

Heat transfer and fluid flow model of GMA fillet welding to generate database

The datasets for training, validation and testing of the neural network were generated by using a well tested heat transfer and fluid flow model that solves the equations of conservation of mass, momentum and energy in three dimensions. Because of the complexity of GMA fillet welding, the following simplifying assumptions[15,16,18,20] are made to make the computational work tractable:

(i) the thermophysical properties needed for calculations such as the thermal conductivity and specific heat of the workpiece material are assumed to be constant for simplicity

(ii) the liquid metal flow is assumed to be incompressible and Newtonian. The effect of turbulent flow in the weld pool is taken into account through the use of the enhanced thermal conductivity and viscosity of the liquid metal[15,16]

(iii) the heat transported from the filler metal droplets is taken into account using a time averaged volumetric heat source[14,15,17,25]

(iv) both the heat and current flux from the arc are assumed to have a Gaussian distribution at the weld top surface.[14,21] For example, heat flux is defined by

$$q_r = \frac{\eta IV}{\pi \sigma_j^2} \exp\left(-\frac{dr^2}{r_0^2}\right)$$

where q_r is the heat flux, η is the arc efficiency, I is the current, V is the voltage, d is the arc distribution factor, r is the radial distance from the arc location, and r_0 is the radius of the arc. The distributions of heat flux and current density are assumed to be unaffected by the deformation of the weld pool top surface.[14,15]

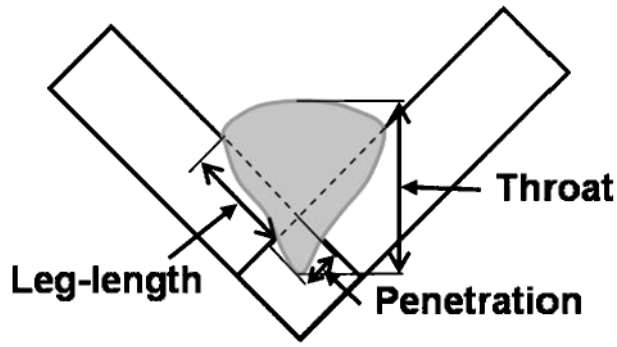
The thermofluid model takes into account the complex fillet joint shape, the deformation of the weld pool top surface, heat transfer by the hot metal droplets and the addition of the filler metal from the consumable electrode.[15,16,18,24] The calculations require the use of a deformable, curvilinear grid system for accurate calculation of heat transfer and fluid flow. Therefore, the governing equations are transformed from the Cartesian to the curvilinear coordinate system.[14,15] The discretised equations for enthalpy, three components of velocity and pressure are formulated by integrating the corresponding governing equations over all the interior control volumes in the computational domain.[26] These equations were solved to obtain the temperature and velocity fields using the modified SIMPLE algorithm.[26] Subsequently, the free surface profile of the weld pool was calculated based on the temperature field obtained in the previous step. After the solution of the free surface profile, the z locations of grids were adjusted to fit the surface profile, and the temperature and velocity fields were then recalculated in the fitted grid system. The calculation procedure was repeated until temperature and velocity fields converged. More details of the numerical model of heat transfer and fluid flow of GMA fillet welding are available in the literature[14–20,24] and are not repeated here. The original fillet geometry is transformed into a rectangular domain of length 450 mm, width 108 mm and depth 18 mm by using an appropriate coordinate transformation.[14,15] A $72 \times 66 \times 47$ grid system with finer grids near the heat source was used for maximum resolution of variables.

Neural network model

All the 22 input variables and their range of values used to develop the neural network are listed in Table 1. While most of these variables are easily understood, the following comments may be of interest. The input variables such as arc efficiency, arc power distribution factor and arc radius determine how heat is absorbed at various locations from the arc.[21] The droplet efficiency[15–18] is defined as the ratio of the total sensible heat input due to metal droplets Q_t and the total heat input IV, i.e. $\eta_d = Q_t/(IV)$, where I is arc current (A) and V is voltage (V). Since temperature independent thermophysical properties of the solid alloy are used in

the model, a question arises as to how to select their values. Since the heat flow in the solid region near the weld pool affects both the size and shape of the weld pool as well as the temperature field in the entire workpiece, it is appropriate to use thermophysical properties at a temperature closer to the melting point than to the ambient temperature. The effective thermal conductivity and viscosity are used to represent enhanced heat and momentum transfer within the weld pool because of the fluctuating components of velocities in a strong recirculating flow confined in a small weld pool. These variables represent system properties, and their values, determined by reverse modelling, are available in the literature[17–20] for GMA fillet welding for various heat input values. The effects of surface active elements have not been rigorously studied in the development of the neural network, to simplify calculations. Therefore, the results are valid for low concentrations of surface active elements.

All the important output variables from the model included in the neural network are listed in Table 2. The three output variables describing the weld cross-section are penetration, leg length and throat. These are shown in Fig. 1. In addition, the length of the weld pool is considered as an output of the neural network, as shown in Table 2. Other outputs of the neural network model include the peak temperature in the weld pool, maximum velocity in the weld pool and cooling time between 800°C and 500°C. The cooling time was calculated on the workpiece surface along the welding direction.



1 During the GMA fillet welding process, the weld geometry is commonly specified by three quantities that affect the joint properties: penetration, throat and leg length

Figure 2 shows the structure of each neural network, which contains an input layer, a hidden layer and an output layer. The input layer comprises all of the 22 input variables, which are connected to neurones in the hidden layer through the weights assigned for each link. The number of neurones in the hidden layer is found by optimising the network. In mathematical terms, we describe the output from a neurone y at each node (represented by circles in Fig. 2) as follows[27–29]

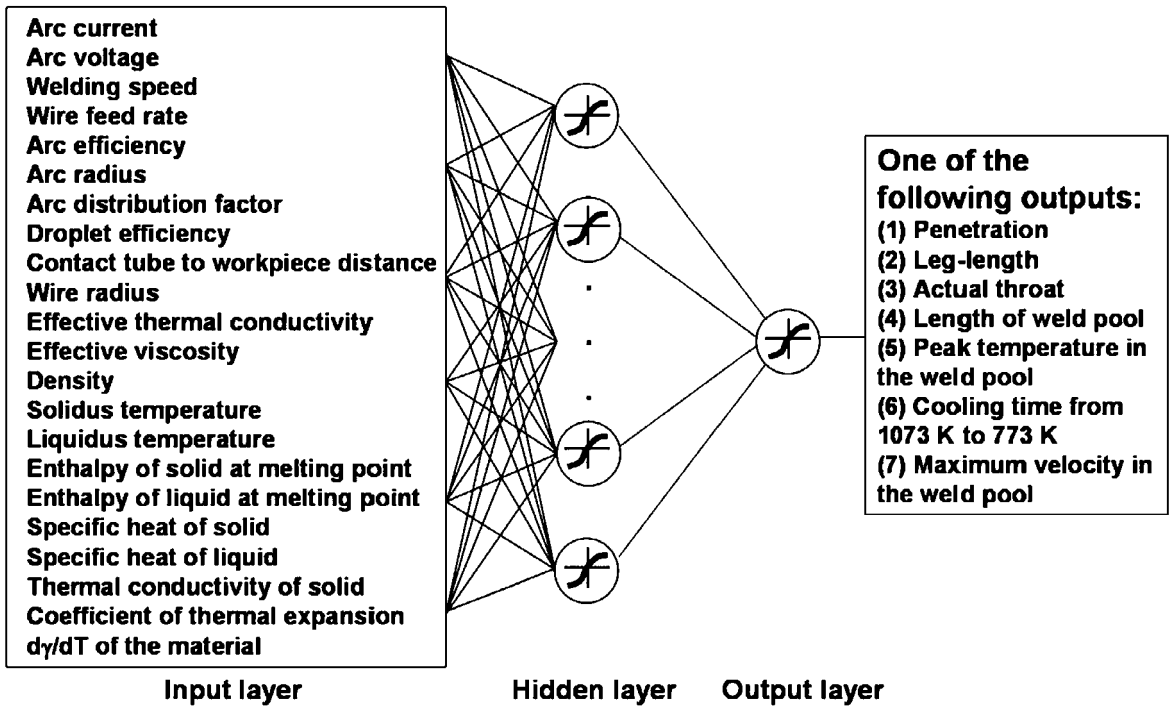
$$y = f\left(\sum_{i=0}^{N} w_i x_i\right) \quad (1)$$

Table 1 The range of input variables used in the generation of training, validation and testing datasets

Variables	Minimum	Maximum	Mean
Arc current, A	200·0	410·0	326·3
Arc voltage, V	25·0	42·0	33·8
Welding speed, mm s^{-1}	4·2	8·5	6·4
Wire feed rate, mm s^{-1}	120·0	290·0	199·6
Arc efficiency	0·4	0·7	0·5
Arc radius, mm	4·0	6·5	5·1
Arc distribution factor	0·5	3	1·4
Droplet efficiency	0·1	0·2	0·13
Contact tube to workpiece distance, mm	17·5	30·0	23·4
Wire radius, mm	0·5	0·9	0·7
Effective thermal conductivity, J m^{-1} s^{-1} K^{-1}	83·6	543·4	298·5
Effective viscosity, kg m^{-1} s^{-1}	$2{\cdot}0\times10^{-2}$	$21{\cdot}0\times10^{-2}$	$7{\cdot}9\times10^{-2}$
Density, kg m^{-3}	7000·0	8500·0	7742·1
Solidus temperature, K	1690·0	1790·0	1741·7
Liquidus temperature, K	1745·0	1815·0	1784·6
Enthalpy of solid at melting point, kJ kg^{-1}	731·5	1149·5	1002·4
Enthalpy of liquid at melting point, kJ kg^{-1}	1045·0	1463·0	1280·2
Specific heat of solid, J kg^{-1} K^{-1}	543·4	794·2	677·0
Specific heat of liquid, J kg^{-1} K^{-1}	689·7	919·6	789·7
Thermal conductivity of solid, J m^{-1} s^{-1} K^{-1}	14·6	40·5	26·9
Coefficient of thermal expansion, 1/K	0·0	$1{\cdot}7\times10^{-5}$	$9{\cdot}1\times10^{-6}$
$d\gamma/dT$ of alloy without any surface active element, N m^{-1} K^{-1}	$-5{\cdot}5\times10^{-4}$	$-2{\cdot}5\times10^{-4}$	$-4{\cdot}2\times10^{-4}$

Table 2 Root mean square error for different output variables

Output variable	Training data	Validation data	Testing data
Leg length, mm	0·01	0·02	0·14
Actual throat, mm	0·02	0·18	0·29
Penetration, mm	0·02	0·08	0·11
Length of the weld pool, mm	0·16	0·18	0·19
Peak temperature in the weld pool, K	4·27	4·58	4·82
Cooling time between 800°C and 500°C, s	0·04	0·06	0·06
Maximum velocity in the weld pool, mm s^{-1}	2·27	2·53	1·85



2 Architecture of the neural network model used in this work. The input layer comprises 22 variables, which are connected to a hidden layer. The output of the network is either penetration, leg length, throat, weld pool length, cooling time between 800°C and 500°C, maximum velocity or peak temperature in the weld pool

where x_1, x_2, x_3, …, x_N are the input signals to the neurone and w_1, w_2, w_3, …,w_N are the synaptic weights that embody the non-linear relationships between the input and the output variables. The combination of a fixed input $x_0=1$ and an extra input weight w_0 accounts for the bias input. The activation function, denoted by f, captures the non-linear interactions of various welding variables such as the arc current, voltage, wire feed rate, welding speed and material properties such as weld geometry, cooling rate and peak temperature in the weld pool during GMA fillet welding. The following hyperbolic tangent function (which is a symmetrical sigmoid function) is used as the activation function

$$y=\tanh\left(a\sum_{i=0}^{N} w_i x_i\right) \tag{2}$$

where a is the slope parameter of the sigmoid function. By varying the parameter a, we can obtain sigmoid functions of different slopes.[27–29] An increase in the value of a increases the slope of the activation function and vice versa. A very high value of the slope makes the curve close to a step function, while a low value retards the convergence rate. Based on the findings of previous studies, a value of 1·5 was used to achieve rapid convergence.[29,30] Furthermore, the use of the tan$h$ function in equation (2) as the activation function helps in keeping the problem reasonably well conditioned. An attractive feature of the hyperbolic tangent function is that its derivative, given by $f'=1-f^2$, does not increase computational volume significantly.[27–29]

To find the weights **W**, a modified back propagation algorithm[31] is used for the training of these neural networks.[27–29] The algorithm tries to minimise the objective function, i.e. the least square error between the predicted and the target outputs, and is given by

$$E=\frac{1}{2}\sum_{p}\left[d_o^{(p)}-y_o^{(p)}\right]^2 \tag{3}$$

where p represents the number of training datasets and o represents the number of output nodes, which is 1 in this work. The desired outputs of the network, such as weld penetration, leg length, throat, cooling rate and peak temperature, are dependent on input welding conditions, material properties and network parameters such as the weights. The working procedure of the back propagation algorithm is explained in Appendix A. The basic or the original back propagation algorithm adjusts the weights in the steepest descent direction (negative of the gradient).[27–29,31] This is the direction in which the error decreases most rapidly. Since this algorithm requires a learning rate parameter to determine the extent to which the weights change in an iteration, i.e. the step sizes, its performance depends on the choice of the value of the learning rate.[27–29,31] A slight modification of the back propagation algorithm includes a momentum term. The momentum term ensures that the previous changes in the weights are considered in determining the current direction of changes of weights. Although there is some guidance for the selection of these parameters, they are more oriented towards specific problems such as pattern recognition, and their performance varies with the type of problem.[27–29,31] Owing to difficulty in the selection of the learning rate parameter and momentum term, the original back propagation algorithm was modified by replacing the steepest gradient method with the CG method for optimising the weights.[31]

Modified back propagation algorithm

In the CG method,[17,18,20,27,31] the weights are updated after each iteration, based on the error calculated using equation (3) for the entire training dataset. For minimising the objective function, the method requires calculation of the step size or change in weights and the search direction. The step size is calculated at each iteration by using Brent's algorithm,[27–29,31] whereas the search direction is calculated by conjugating the previous direction with the current gradient of the objective function. Brent's iterative line search algorithm[19,24] utilises the parabolic interpolation and Golden section search method[27–29,31] to locate the line minima. At the first iteration, since there is no previous direction, the search is performed in the direction of the steepest descent. The directions of search at subsequent iterations are calculated by conjugating the previous direction with the current gradient. These conjugate directions are actually calculated on the assumption that the error surface is quadratic.[27–29] However, if the algorithm discovers that the current line search direction is not downhill, it simply calculates the line of steepest descent and restarts the search in that direction. Once a point close to a minimum is found, the quadratic assumption holds true and the minimum can be located very quickly using the Golden section search method.[27–29,31] The steps involved in the calculation are as follows.[22,23,28,31] A line search is performed to determine the optimal distance to move along the current search direction. If we let p_k denote the direction vector at iteration k of the algorithm, then the weights are updated using the following rule[27–29,31]

$$W^{k+1} = W^k + \Delta W^k = W^k - \eta_k \left(\frac{\partial E}{\partial W}\right)_0 = W^k + \eta_k p_k \quad (4)$$

where η_k is the learning rate or search step size calculated at each iteration by using Brent's method.[32,33] For the first iteration, the initial direction p_0 is the same as the steepest descent direction (negative of the gradient)

$$p_0 = -g_0 = -\left(\frac{\partial E}{\partial W}\right)_0 \quad (5)$$

where g is the gradient of error with respect to the weight of the link connecting any two consecutive layers. Then the next search direction is determined so that it is conjugate to previous search directions. The general procedure for determining the new search direction is to combine the new steepest descent direction with the previous search direction[22,23]

$$p_k = -g_k + \beta_k p_{k-1} \quad (6)$$

The various versions of CG are distinguished by the manner in which the constant β_k is computed. For the Polak–Ribiére update, the constant β_k is computed from[17,18,20,22,23,32] the following equation

$$\beta_k = \frac{(g_k - g_{k-1})^T g_k}{g_{k-1}^T g_{k-1}} \quad (7)$$

where T refers to transposition of the matrix, i.e. rows changed to columns. This is the product of the previous change in the gradient, with the current gradient divided by the norm squared of the previous gradient. The gradient between the hidden node h and the output node o is given by[27–29,31]

$$\mathbf{g} = g_{oh} = \frac{\partial E}{\partial w_{oh}} = -\sum_p \delta_o^{(p)} y_h^{(p)} \quad (8)$$

$$\delta_o^{(p)} = \left[d_o^{(p)} - y_o^{(p)}\right] f'\left[x_o^{(p)}\right] \quad (9)$$

where $y_h^{(p)}$ and $y_o^{(p)}$ represent the output at hidden and output nodes, respectively, and $x_o^{(p)}$ is the input. For the input-to-hidden connections, the gradient between hidden node h and input node i is given by[27–29,31]

$$g_{hi} = \frac{\partial E}{\partial w_{hi}} = -\sum_p \delta_h^{(p)} y_i^{(p)} \quad (10)$$

$$\delta_h^{(p)} = f'[x_h^{(p)}] \sum_o \delta_o^{(p)} w_{oh} \quad (11)$$

The value of output nodes o is 1 in equations (8), (9) and (11), since only one output is produced by each of the neural networks. In the gradient descent algorithms, calculations are started at some point on the error function defined over the weights, and an attempt is made to move to the global minimum of the function. The gradient based methods can easily get trapped in local minima. Stochastic optimisation techniques are capable of finding the global minima and avoiding local minima.[34] Therefore, a genetic algorithm (GA)[34] is used along with the CG method to find the optimal global weights in the present work. A parent centric recombination (PCX) operator based generalised generation gap (G3) GA model[24,35–38] was used in the present study. The PCX operator is a steady state, elite preserving, scalable and computationally fast population alteration model.[35] This model was chosen because it has been shown to have a higher convergence rate on standard test functions as compared to other evolutionary algorithms and classic optimisation algorithms, including other real parameter GAs with the unimodal normal distribution crossover (UNDX) and the simplex crossover (SPX) operators, the correlated self-adaptive evolution strategy, the covariance matrix adaptation evolution strategy (CMA-ES), the differential evolution technique, and the quasi-Newton method.[35] Detailed description of this model is available in the literature[24,35–38] and is not included here. The various terms used to describe GA are explained in Table 3. Specific application of GA for determination of the optimised weights of the neural network is given in Appendix B.

Calculation procedure

Number of hidden layers in the network

A large database is required for training, validation and testing of the neural network. The number of training datasets should be more than the number of weights connecting different nodes. The number of hidden layers in a neural network depends on the type of the problem and the relationships between the input and the output variables. Theoretically, any continuous variation of output with respect to input can be represented by a single hidden layer.[39,40] Two hidden layers are needed when the relationships between the input and the output variables are discontinuous.[39,40] The use of more than

the optimal number of hidden layers in the network may result in undesirable overfitting of the data.[27–30,39,40] A single hidden layer was used, since the outputs are continuous in nature in GMA fillet welding. For a single hidden layer network, the number of weights is given by the following

$$\text{Number of weights in the network} = (n_i+1)n_h+(n_h+1)n_o \quad (12)$$

where n_i is the number of input variables, i.e. 22 in the present work, n_h is the number of nodes in the hidden layer, and n_o is the number of output variables, i.e. 1. Since the number of weights increases with the increase in the number of nodes, an optimal number of nodes is required.

Database generation based on design of experiments

A database for training of the neural networks was generated on the basis of the design of experiments to capture the effects of all the welding parameters and material properties.[41] The original L_{81} (3^{40}) orthogonal array contains 81 rows and 40 columns of variables, each having three levels of values for capturing the interactions among all 40 variables.[41] However, in the case of GMA fillet welding, we have 22 input variables. In the L_{81} (3^{40}) array, only 22 columns are used and the remaining 18 remain vacant. Since only three levels of weld process parameters are not capable of capturing the complex interactions among variables, the array is modified to L_{81} ($9^6 \times 3^{16}$) based on linear graph theory,[41] as explained in Appendix C. In the modified array, the six most important variables, current, voltage, welding speed, wire feed rate, effective thermal conductivity and effective viscosity, are assigned nine levels, and the remaining 16 variables are assigned three values each. The rest of the variables, such as the efficiency, arc radius, arc power distribution factor, contact tube to workpiece distance (CTWD) and material properties, such as density, specific heat of the solid, specific heat of the liquid and latent heat, are kept at three levels. This procedure increases the degrees of freedom and helps to capture the effects of the variables, which have a large influence on the weld geometry and cooling rate. Two L_{81} ($9^6 \times 3^{16}$) arrays for each of the Fe 1005, Fe 1045 and A36 steels were used. For each steel, we used two different sets of nine levels of values of current, voltage, welding speed, wire feed rate, effective thermal conductivity and effective viscosity to capture the interactions. Similarly, material properties and other parameters such as arc efficiency, arc radius, arc power distribution factor and CTWD are different in each array. Therefore, we have 18 levels of all the input variables in the form of six L_{81} ($9^6 \times 3^{16}$) arrays, or 486 datasets for training. For validation and testing of the neural network, an additional 50 and 25 different datasets were generated using three-dimensional heat transfer and a fluid flow model. These datasets for validation and testing were generated randomly by selecting the values of variables that are different from the training dataset. The ranges of all the 22 input variables used for the generation of datasets are shown in Table 1. The different levels or values of the variables are decided based on their sensitivity to weld geometry. In the database, variables such as arc current, arc voltage, welding speed, wire feed rate, effective thermal conductivity and effective viscosity, which have a major influence on weld geometry, are taken at many levels compared to the other remaining variables, as shown in Fig. 3. The material properties in the database were selected around their corresponding values for Fe 1005 steel, Fe 1045 steel and A36 steel.

Normalising inputs and outputs

The values of the input and output variables vary significantly. The vastly different scales of inputs and bias values lead to ill-conditioning of the problem.[27–29] Whereas large inputs cause ill-conditioning by leading to very small weights, large outputs do so by leading to very large weights.[27–29] To eliminate the ill-conditioning problem, the data were normalised using the following formula[29]

$$x' = 2\left(\frac{x - x_{\min}}{x_{\max} - x_{\min}}\right) - 1 \quad (13)$$

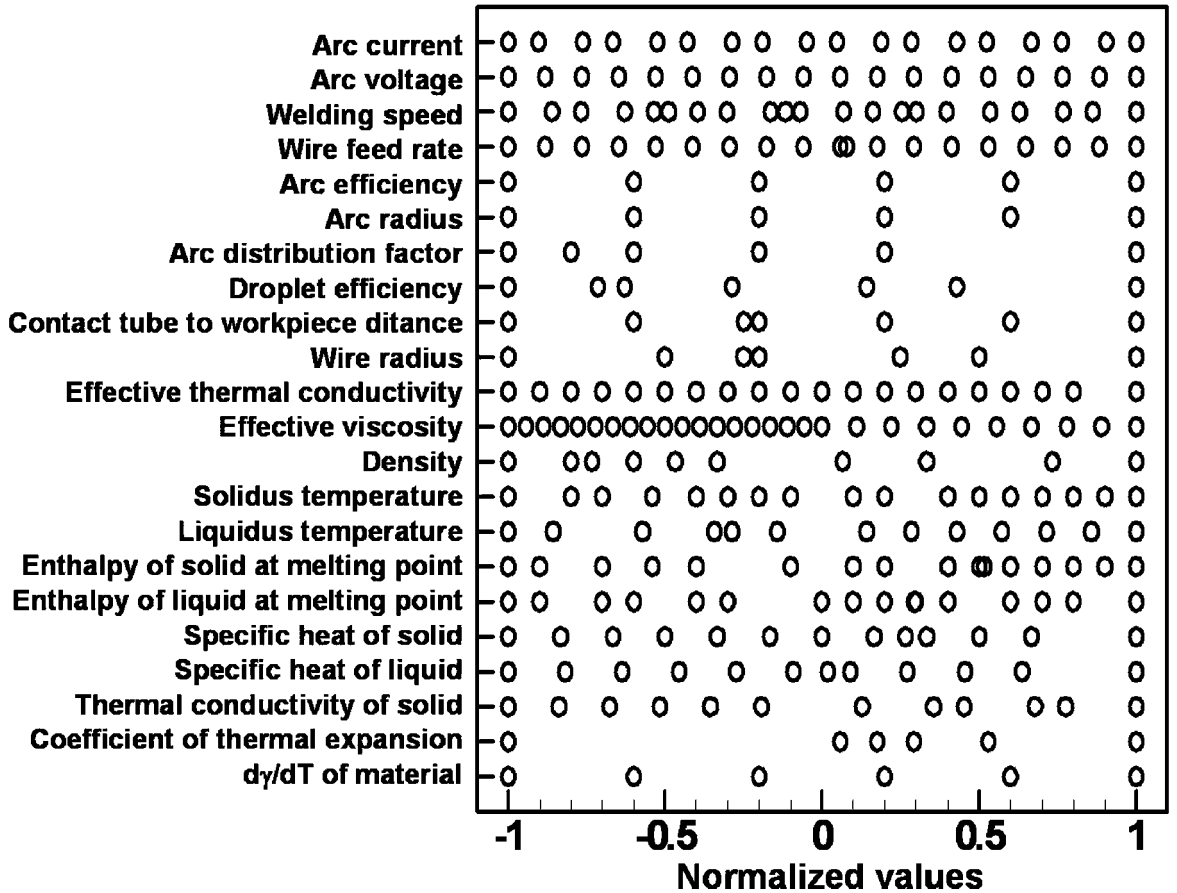
where x is the original value of the variable, x' is the normalised value, and $x_{\min}$ and $x_{\max}$ represent the minimum and maximum values of the variable in the whole dataset. Equation (13) normalises the data in the range -1 to 1. The range of values of all input and output parameters from -1 to $+1$ implies that the standard deviation cannot exceed 1, and its symmetry about zero means that the mean will typically be relatively small. Furthermore, its maximum derivative is also 1·5, so that back propagated errors will be neither magnified nor attenuated more than necessary.[29]

Selection of initial weights

In the back propagation algorithm, the magnitude of the error propagated backwards through the network is proportional to the value of the weights. If all the weights are the same, the back propagated errors will be

Table 3 Terminology used in genetic algorithm

Biological terms	Equivalent neural network variables and representation in genetic algorithm
Genes: units containing hereditary information	In the form of weights of the network variables w_1, w_2, …, w_n, e.g. $w_1=-0{\cdot}10$; $w_2=0{\cdot}17$; $w_n=0{\cdot}26$
Chromosome/individual: a number of genes folded together	A set of values of weights taken together, i.e. (−0·10, 0·17, …, 0·26)
Population: collection of many chromosomes/individuals	Collection of multiple sets of weights: (−0·10, 0·17, …, 0·26), (0·15, 0·27, …, 0·24), (0·33, −0·14, …, 0·43)
Parents: chromosomes/individuals participating in creating new individuals (or offspring)	Parents, e.g. (−0·10, 0·17, …, 0·26), (0·15, 0·27, …, 0·24)
Fitness value: value of fitness function determines if a chromosome/individual survives or dies	Objective function or fitness function: calculated for each set of input variables using equation (3)



3 Range of input variables in the database used for training, validation and testing of the network. The normalised value of the variables was obtained using equation (13) and corresponding minimum and maximum values listed in Table 1. Various combinations of these 22 input variables were generated using the modified L_{81} ($9^6 \times 3^{16}$) array in the database

the same, and consequently all of the weights will be updated by the same amount.[27–29] To avoid this symmetry problem, the initial weights of the network were selected randomly. Furthermore, to avoid the premature saturation of the network, the initial values of the weights were distributed inside a small range of values, i.e. in the interval −0·5 to 0·5. When the weights are small, the neurones operate in the linear regions of the activation function, and consequently the activation function does not saturate.

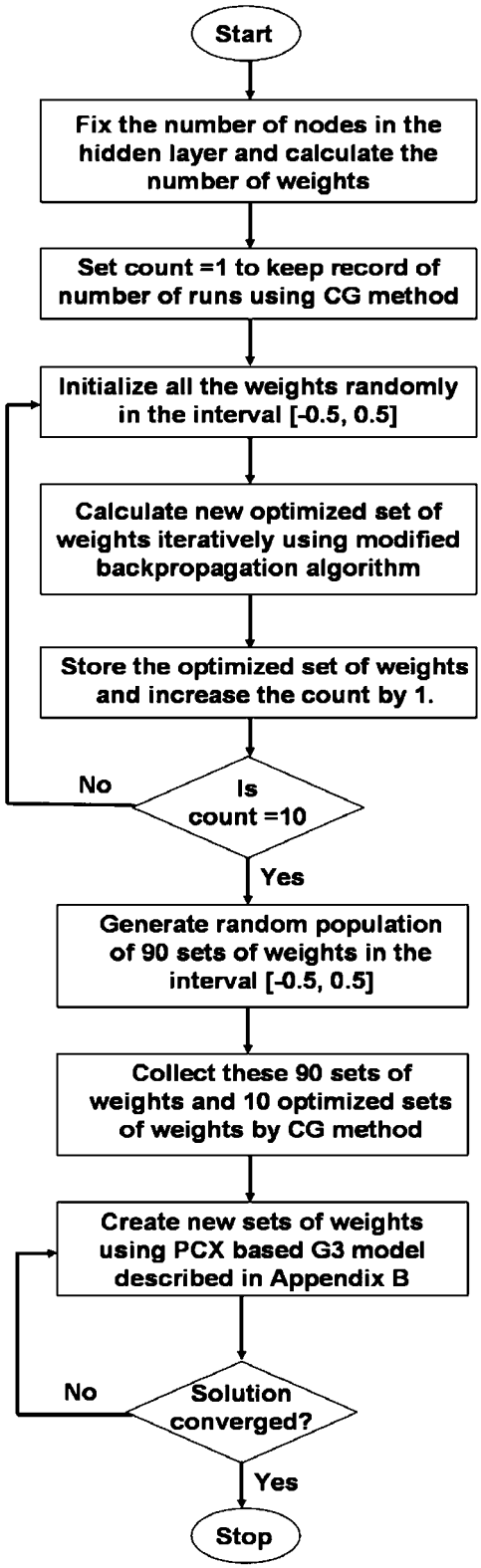
The calculation starts with the selection of the number of nodes in the hidden layer. The total number of weights in the network is calculated on the basis of the number of nodes in the hidden layer. The weights are then initialised randomly in the interval −0·5 to 0·5, as described in the previous section. In the next step, a modified back propagation algorithm is used to minimise the error on the training dataset. The weights calculated by the CG method are stored as one possible set of weights. This process is repeated 10 times with different randomly selected initial weights for fixed values of nodes in the hidden layer. All of these 10 sets of weights are provided as input to the GA. The final aim of the GA is to find the weights in the network through a systematic global search that will give the least error between the neural network prediction and heat transfer and fluid flow calculations. The flowchart of the calculation scheme is presented in Fig. 4. The convergence is based on the error in training and validation data. When the error during validation starts to increase, the calculation is stopped to avoid overfitting, even if the error in the training dataset decreases with iteration.

Results and discussion

The number of nodes in the hidden layer was varied to get an optimum number of nodes that resulted in a minimum mean square error (MSE), as shown in Fig. 5. The MSE is defined as follows

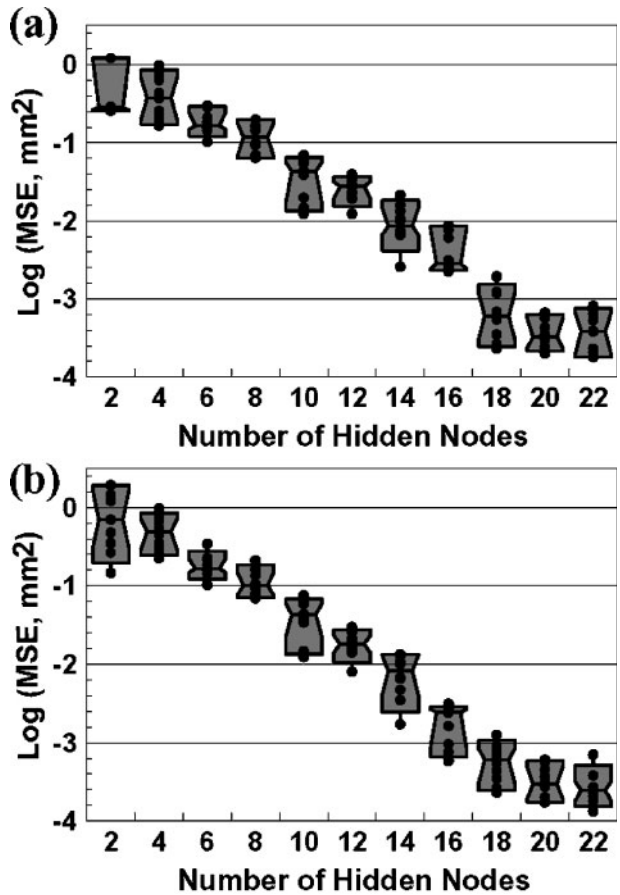
$$\mathrm{MSE} = \frac{2 \cdot E}{p} = \frac{1}{p} \sum_{\mathrm{p}} \left[d_{\mathrm{o}}^{(\mathrm{p})} - y_{\mathrm{o}}^{(\mathrm{p})} \right]^2 \tag{14}$$

where E is the objective function represented by equation (3). Figure 5*a* shows that log(MSE) decreases almost linearly with increase in the number of hidden nodes. The results are shown for penetration as the output variable. Other output variables also showed the same trend. These runs were conducted using the CG method with 10 different randomly selected initial sets of weights in the neural network to avoid any local optimal solution. Figure 5*b* shows a similar trend for the variation in log(MSE) for leg length. The log(MSE) for penetration and leg length becomes almost constant for more than 19 hidden nodes in the network. Figure 6 shows that most of the runs for 19 hidden nodes gave very small values of error within 15 000 iterations. However, the convergence rate depended on the choice of initial set of weights. This is the main difficulty in using the CG method alone to find the optimal weights.

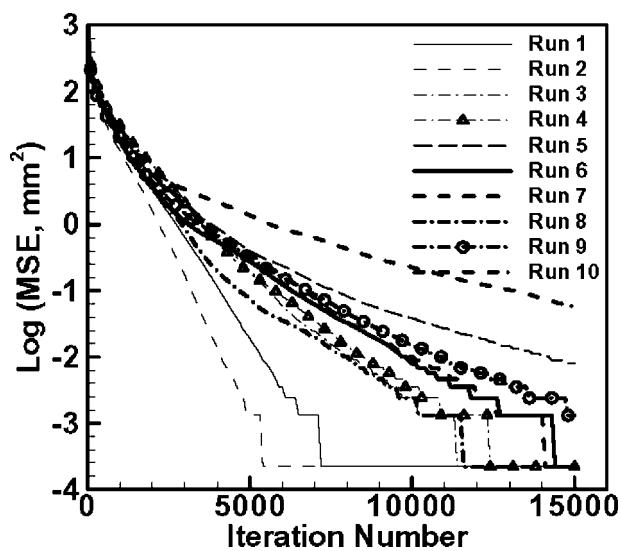


4 Flowchart of the modified back propagation algorithm using the hybrid optimisation model after coupling of the generalised generation gap (G3) genetic algorithm with the conjugate gradient method

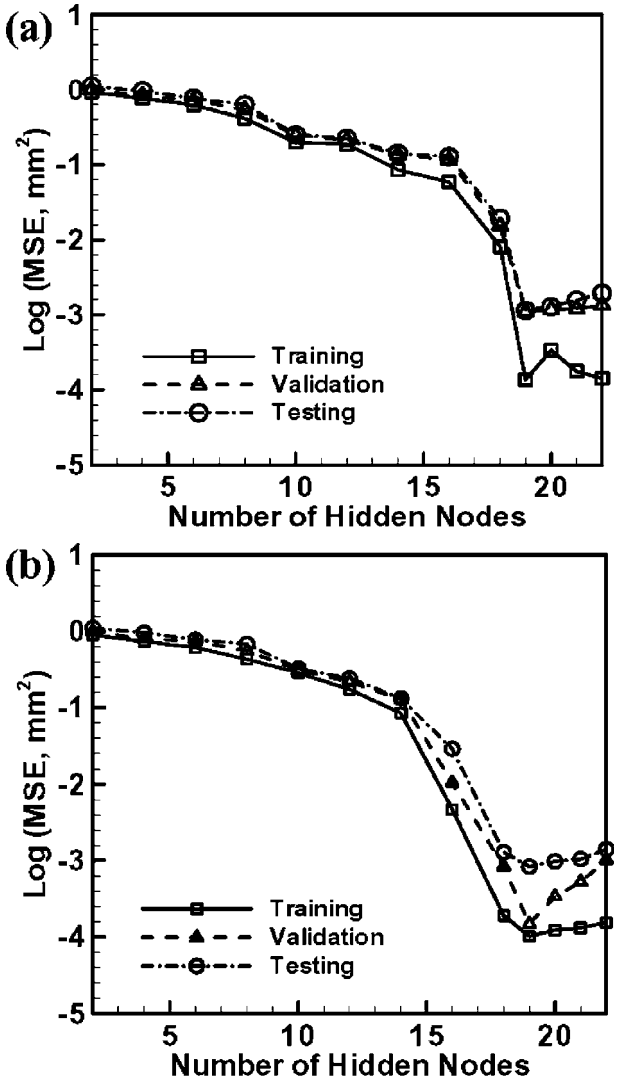
In order to solve this problem and obtain globally optimised weights, the weights obtained by using the CG method for 10 different runs were included in the input to a GA based optimisation model. Since the GA requires a population of at least 100 individuals (or different sets of weights)[24,34,35] to start the calculations,



5 *a* Variation of log(MSE) for penetration with number of hidden nodes for training data by using the CG optimisation method after 50 000 iterations. Dots represent log(MSE) obtained in 10 different runs, which were taken to examine the effect of initial weights on the final converged solution. The box whisker plot shows the variation and the mean of the log(MSE) for different numbers of hidden nodes. Seventy-five percent of the data are inside the shaded box. *b* Variation of log(MSE) for leg length with number of hidden nodes for training data by using the CG optimisation method



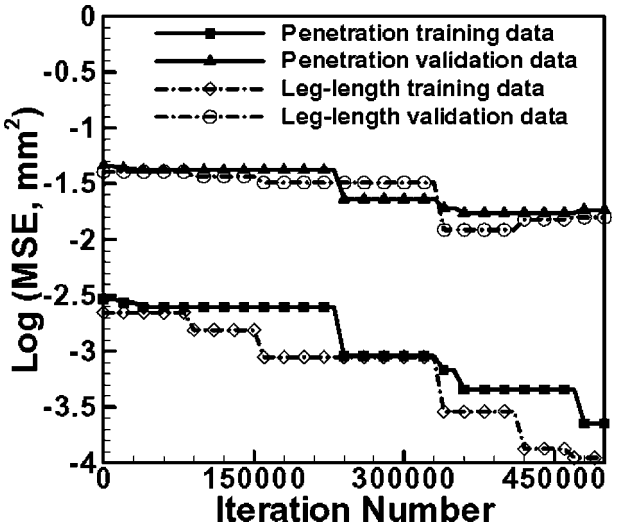
6 Variation of log(MSE) for penetration training data with number of iterations for 19 hidden nodes by using the CG optimisation method



7 Variation of log(MSE) in *a* penetration and *b* leg length for training, validation and testing data by using the hybrid optimisation scheme after 50 000 iterations; the 19 nodes in the hidden layer provide less error on all three datasets

the rest of the 90 sets of weights were generated randomly in the interval −0·5 to 0·5.

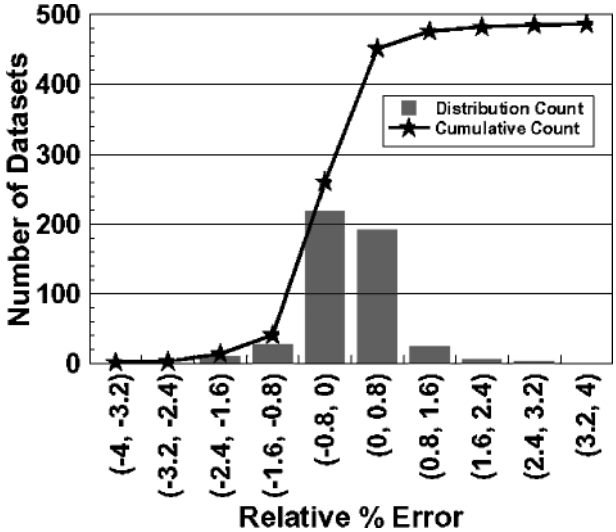
Figure 7*a* shows the log(MSE) for penetration with the best optimised set of weights for various numbers of hidden nodes with the hybrid optimisation scheme. For the first 19 hidden nodes, the error decreased continuously with the increase in number of nodes, and then started to increase slightly with the increase in nodes. The lower value of MSE may also be due to overfitting of the network. The performance of the network was tested using the validation and testing datasets. The network was trained using only the training data. The validation data were randomly generated independently of the training data. During training, if the network learns the effects of input variables on the output, the MSE on the validation set improves with the iterations. However, if the network minimises the MSE on the training data by overfitting, the MSE on the validation dataset will increase. This behaviour indicates that the interactions between different input variables of the training dataset are accurately modelled for the training data but not for all possible values of input variables.



8 Variation of averaged log(MSE) in penetration and leg length for all the 100 individual members in the population by using the hybrid optimisation method for 19 nodes in the hidden layer. The calculation was stopped when the error on the validation data started to increase

This behaviour also means that the performance of the network may vary significantly for training, validation and testing datasets. To avoid overtraining (or overfitting), the training was stopped after some iterations when the performance of the network for the validation data was optimal. The testing data were used to check the overall performance of the network.

Figure 7*a* shows that 19 hidden nodes provide the lowest value of log(MSE) for penetration for the training, validation and testing data. Similarly, the minimum value of log(MSE) for leg length was found for 19 hidden nodes, as shown in Fig. 7*b*. Since a neural network with 19 hidden nodes showed low errors for all other output variables, 19 hidden nodes were selected for all the variables for simplicity. Figure 8 shows the variation of log(MSE) in penetration and leg length with iterations using the hybrid method for 19 nodes in the hidden layer. Initially, the error for both training and validation data decreases with iterations. However, once the network gets the optimal weights, the error becomes almost constant. The calculation was stopped when the error for validation data started to increase with change in weights or iterations. Furthermore, the number of iterations depended on the output variables and was not the same for all variables. For example, for leg length and penetration, the calculations were stopped after 42 000 and 48 000 iterations, respectively, based on the results in Fig. 8. Table 2 shows the root mean square (rms) error for all the output variables. The rms errors in the training data for penetration, leg length and peak temperature in the weld pool were 0·02 mm, 0·01 mm and 4·27 K, respectively. These rms errors were quite small in comparison to the magnitude of these output variables. The rms errors in training data for other outputs were also very small. However, the rms errors were higher for both the validation and testing data. Leg length showed a good match for all three datasets. This is because leg length depends mainly on the heat input and is not significantly influenced by the impingement of droplets.[17–19] Furthermore, the rms error for



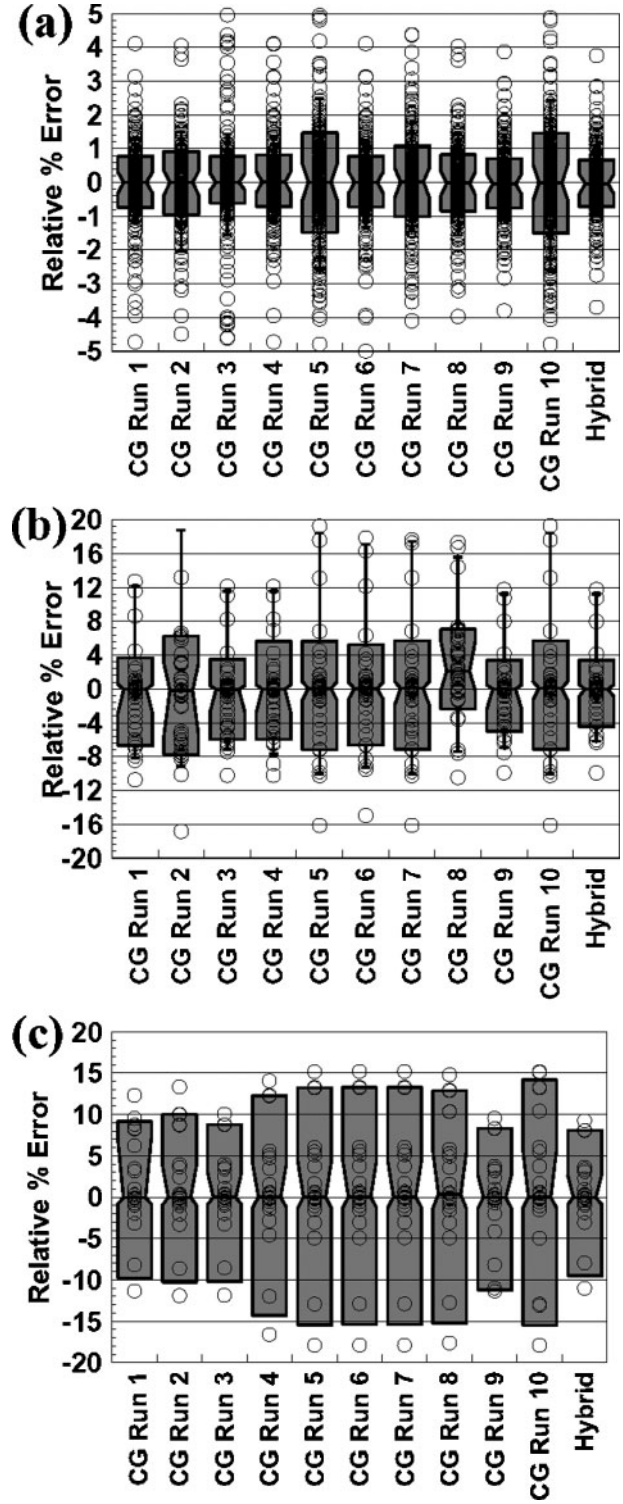
9 Frequency plot showing number of training datasets of penetration lying in different ranges of relative percentage error

penetration was large because its calculation involves complex interactions between various welding process parameters such as arc current, welding speed and wire feed rate. The rms error in testing data was generally higher because the testing data were selected near the extreme range of input variables, with the exception of maximum velocity, for which the errors in the testing data are smaller. To further check the accuracy of the result predicted by the neural network, the relative errors were calculated as follows

$$(\text{Relative \% error})_i = 100 \times \left(\frac{d_i - y_i}{d_i}\right) \quad (15)$$

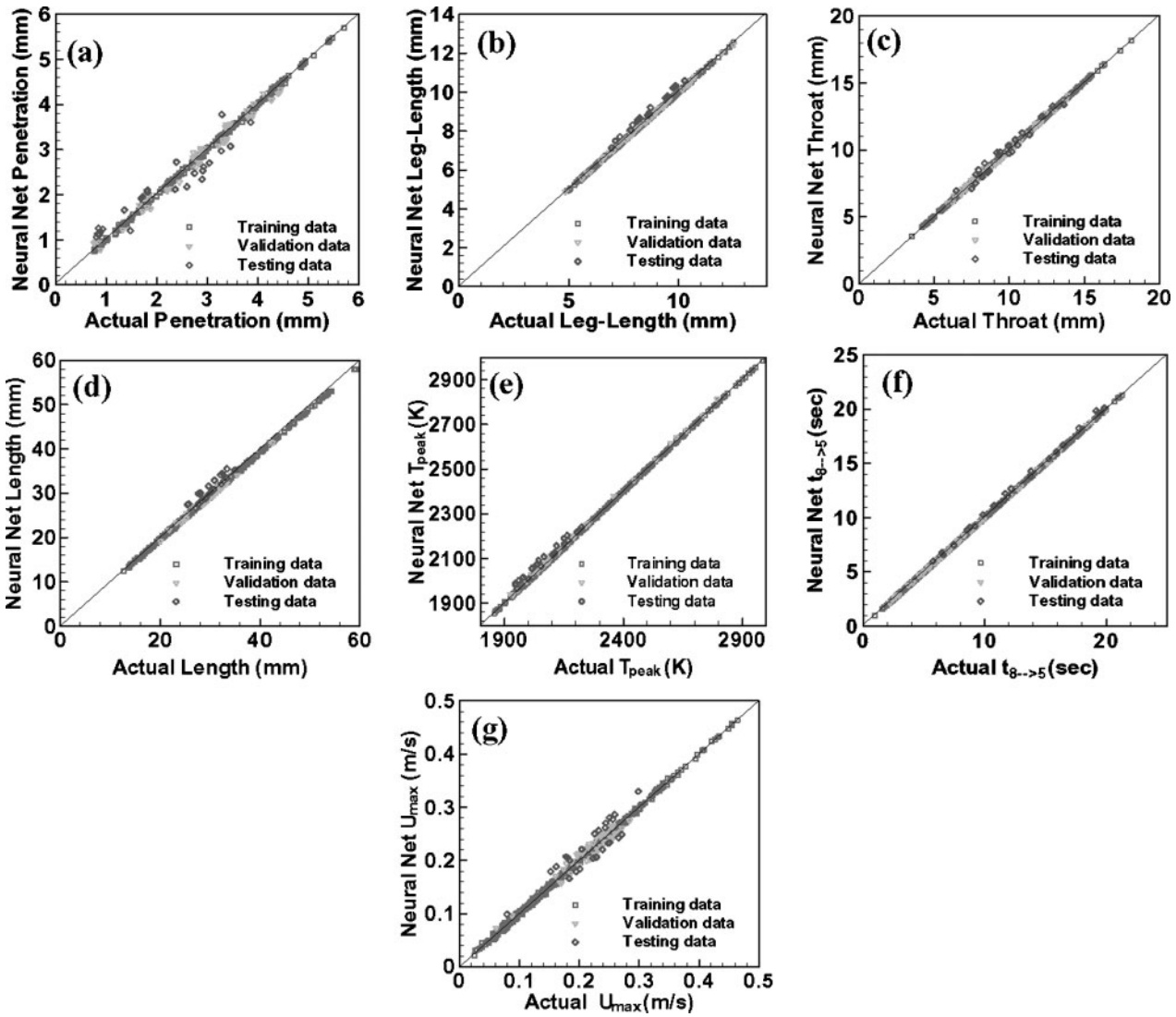
where d_i is the desired value of the penetration for the ith data point, and y_i is the calculated value of the penetration by the neural network for the same dataset. The results are presented as typical frequency v. relative percentage error for penetration in Fig. 9. This error has a classic Gaussian distribution, with the centre around zero. Furthermore, more than 90% of the data has an error of less than 0·8% in the training dataset. This low percentage error indicates the accuracy of the neural network.

Various runs were conducted using the hybrid optimisation method with different randomly selected initial sets of weights in the population to obtain the optimised values of all weights. The hybrid method gave either a smaller or the same error as the CG method. The computations took approximately 6 h for 19 hidden nodes on a 3·06 GHz Intel P4 CPU with 512 Mb PC2700 DDR-SDRAM memory. Figure 10 compares the relative percentage error for training, validation and testing data for different best sets of optimal weights obtained using the CG method alone and the hybrid method involving CG and GA. The results show that the best set of weights obtained using the hybrid method provides less error than the best sets obtained using the CG method. Furthermore, the relative percentage error with the CG method varied significantly with the initial values of the weights. As a result, the CG method alone is not capable of finding the best optimal network. The relative percentage errors in training data with the CG



10 Comparison of relative percentage error in penetration for *a* training data, *b* validation data, and *c* testing data, calculated by taking 10 different runs of the CG method alone in the neural network and the hybrid optimisation method. These box whisker plots show that the hybrid optimisation method always produced less error than the CG method

method in runs 4, 6 and 8 were low. However, all of these runs resulted in a large relative percentage error in validation and testing data (Fig. 10*b* and *c*). Use of the CG method to determine the weights produced suboptimal solutions. With use of the optimal weights



11 Comparison of output variables, i.e. *a* penetration, *b* leg length, *c* throat, *d* length of weld pool, *e* peak temperature in the weld pool, *f* cooling time between 800°C and 500°C, and *g* maximum velocity U_{max} in the weld pool, calculated from the heat transfer and fluid flow model (*x*-axis) with corresponding values predicted by the neural network model of GMA fillet weld. The diagonal lines in each plot show that, ideally, all the points should lie on this line. The training data, validation data and test data comprise 486, 50 and 25 datasets, respectively

produced by the hybrid method, the average relative percentage errors in training data, validation data and testing data were 0·5%, 2·4% and 3·8%, respectively, for the predicted value of the penetration. Table 4 shows that the average relative percentage errors in training data for other output variables were less than 0·5 %, except for maximum velocity, which had a maximum error of 0·7%.

Computed values of penetration, actual throat, leg length, length of weld pool, peak temperature, cooling time and maximum velocity in the weld pool obtained from the neural network model were compared with their corresponding values obtained from the heat transfer and fluid flow model to examine the accuracy of the neural network. Figure 11 shows all the output data obtained using different sets of welding variables used during the training, validation and testing of the network. All the neural networks of GMA fillet weld comprised only one hidden layer containing 19 nodes for each of the seven output variables, i.e. penetration, actual throat, leg length, length of weld pool, peak temperature, cooling time and maximum velocity in the weld pool. Figure 11 shows the predicted values of outputs from the neural network and the corresponding values calculated using the heat transfer and fluid flow model. The plots show that all points lie on or very close to the diagonal line, and the results obtained from the neural network agree well with the values calculated using the heat transfer and fluid flow model. Table 4 shows the average absolute value of the relative percentage error in all the output variables for training, validation and testing data. The maximum average value of the relative percentage error in all outputs was 1·1, 2·4 and 6·0 in the training, validation and testing data, respectively. The results indicate that the neural network can be used for simulations with predetermined good accuracy.

Summary and conclusions

A set of multiple feedforward neural networks was developed for GMA fillet welding to calculate penetration,

Table 4 Average absolute value of the relative percentage error for different output variables

Output variable	Training data	Validation data	Testing data
Leg length	0·1	0·1	1·2
Actual throat	0·1	1·0	1·8
Penetration	0·5	2·4	3·8
Length of the weld pool	0·4	0·4	0·5
Peak temperature in the weld pool	0·1	0·2	0·2
Cooling time between 800°C tand 500°C	0·2	0·2	0·7
Maximum velocity in the weld pool	0·7	0·8	0·8

leg length, throat, weld pool length, cooling time between 800°C and 500°C, maximum velocity and peak temperature in the weld pool. These neural network models used 22 input variables, including welding process parameters and material properties. A hybrid optimisation scheme involving a CG and a GA for finding global optimal weights resulted in low errors in training, validation and testing data. The results obtained using the hybrid scheme were better than those obtained with the CG method alone. For each output variable, a separate neural network model was developed. This approach provided superior results and greater flexibility than one neural network for all the output variables. These neural network models can replace the output of complex heat transfer and fluid flow models with significant computational economy.

Appendix A

Back propagation algorithm

The back propagation training consists of two passes of computation, a forward pass and a backward pass.[27–31] In the forward pass, an input vector (i.e. set of welding variables) is applied to the sensory nodes of the network. The signals from the input layer propagate to the units in the first layer, and each unit produces an output according to equation (2). The outputs of these units are propagated to units in subsequent layers, and this process continues until the signals reach the output layer, where the actual response of the network (i.e. weld geometry parameters such as penetration, throat, leg length and cooling time) to the input vector is obtained. During the forward pass, the synaptic weights of the network are fixed. During the backward pass, on the other hand, the synaptic weights are all adjusted in accordance with an error signal, which is propagated backwards through the network. The basic idea is that the objective function (given by equation (3)), which is a discrepancy between the desired solution and the predicted values, has a particular surface over the weight space, and therefore an iterative process such as the CG method can be used for its minimisation. In short, the basic back propagation algorithm is as follows:[27–29,31]

(i) initialise the input layer
(ii) propagate signals forward
(iii) calculate the error in the output layer
(iv) back propagate the error
(v) update the weights.

Appendix B

PCX based G3 genetic algorithm

The steps involved in calculating the optimal values of weights are as follows:

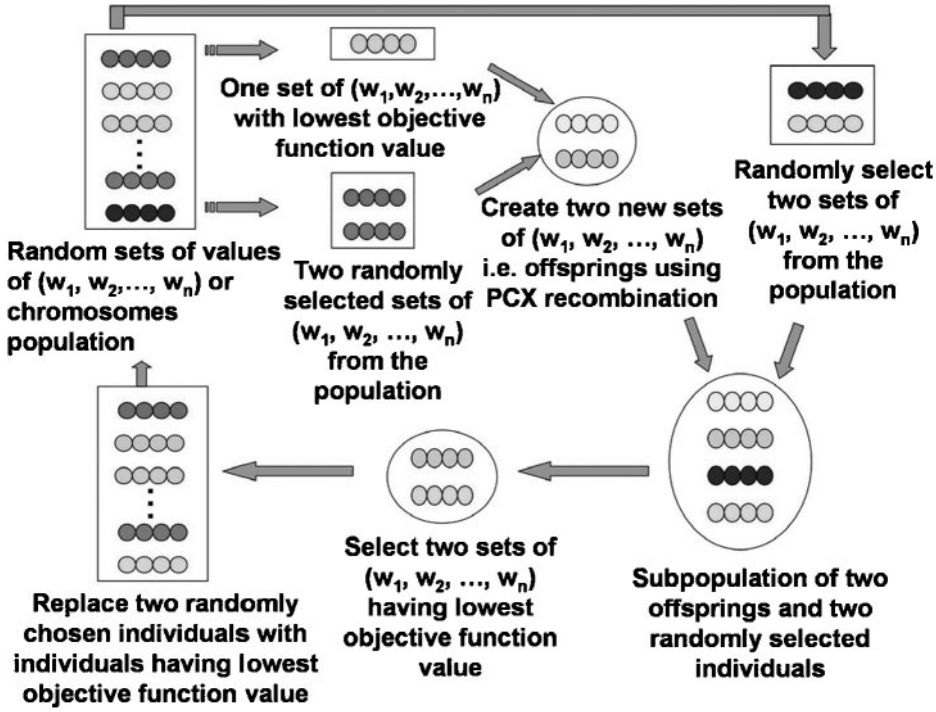
(i) a population is a collection of many individuals, and each individual represents a set of randomly chosen values of all the weights. A parent refers to an individual in the current population. The best parent is the individual that has the best fitness, i.e. gives the minimum value of the objective function, defined by equation (3), in the entire population. The best parent and two other randomly selected parents are chosen from the population
(ii) from the three chosen parents, two offspring or new individuals are generated, using a recombination scheme. PCX based G3 models are known to converge rapidly when three parents and two offspring are selected. A recombination scheme is a process for creating new individuals from the parents. The recombination scheme (step (ii)) used in the present model is based on the PCX operator[24,34–38]
(iii) two new parents are randomly chosen from the current population of the individuals
(iv) a subpopulation of four individuals that includes the two randomly chosen parents in step (iii) and two new offspring generated in step (ii) is formed
(v) the two best solutions, i.e. the solutions having the least values of the objective function, are chosen from the subpopulation of four members created in step (iv). These two individuals replace the two parents randomly chosen in step (iii).

The calculations are repeated from step (i) again until convergence is achieved, as shown in Fig. 12. The recombination scheme (step (ii)) used in the present model is based on the PCX operator. A brief description of this operator, tailored to the present problem, is as follows.

The first three parents, i.e. $(w_1^0,w_2^0,w_3^0,\ldots,w_n^0)$, $(w_1^1,w_2^1,w_3^1,\ldots,w_n^1)$, $(w_1^2,w_2^2,w_3^2,\ldots,w_n^2)$, are randomly selected from the current population. Here, the subscripts represent the number of weights in the network, and the superscripts denote the parent identification number. The mean vector or centroid

$$\vec{g}=\left(\frac{w_1^0+w_1^1+w_1^2}{3},\frac{w_2^0+w_2^1+w_2^2}{3},\frac{w_3^0+w_3^1+w_3^2}{3},\cdots,\frac{w_n^0+w_n^1+w_n^2}{3}\right)$$

of the three chosen parents is computed. To create an offspring, one of the parents, say $\vec{x}^{(p)}=(w_1^0,w_2^0,w_3^0,\ldots w_n^0)$, is chosen randomly. The direction vector $\vec{d}^{(p)}=\vec{x}^{(p)}-\vec{g}$ is next calculated from the selected parents to the mean vector or centroid. Thereafter, from each of the other two parents, i.e. $(w_1^1,w_2^1,w_3^1,\ldots w_n^1)$, and $(w_1^2,w_2^2,w_3^2,\ldots w_n^2)$, perpendicular



12 The working principle of the GA based on the generalised generation gap (G3) model and using the PCX operator

distances D_i to the direction vector $\vec{d}^{(p)}$ are computed and their average $\overline{D}$ is found. Finally, the offspring, i.e. $\vec{y} = \left(w_1', w_2', w_3', \ldots w_n'\right)$, is created as follows[24,34–38]

$$\vec{y} = \vec{x}^{(p)} + w_\zeta \left|\vec{d}^{(p)}\right| + \sum_{i=1, i\neq p}^{n} w_\eta \overline{D}\, \vec{h}^{(i)}$$

where $\vec{h}^{(i)}$ are the orthonormal bases that span the subspace perpendicular to $\vec{d}^{(p)}$, and w_ζ and w_η are randomly calculated zero mean normally distributed variables.

Appendix C

The original L_{81} (3^{40}) orthogonal array was modified to L_{81} ($9^6 \times 3^{16}$) on the basis of linear graph theory. The columns in the L_{81} (3^{40}) orthogonal array have two degrees of freedom.[41] Since any variable in the modified L_{81} ($9^6 \times 3^{16}$) array with nine levels requires eight degrees of freedom,[41] it was necessary to determine the appropriate four columns from the L_{81} (3^{40}) orthogonal array to which to assign this variable. The procedure for converting the standard orthogonal array to a multilevel L_{81} ($9^6 \times 3^{16}$) array includes the following steps.

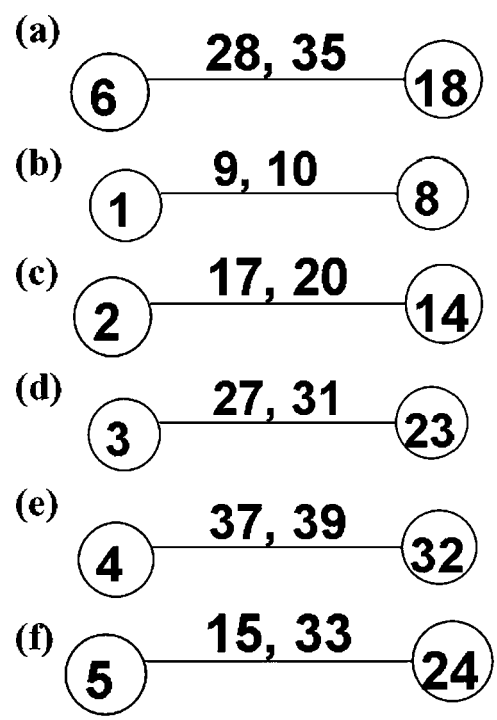
Step 1: Establish the standard L_{81} (3^{40}) orthogonal array.

Step 2: Choose a suitable linear graph. For example, a linear graph joining four columns in the L_{81} (3^{40}) orthogonal array is shown in Fig. 13*a*. Based on this graph, four columns, i.e. columns 6, 18, 28 and 35, were selected from the standard L_{81} (3^{40}) orthogonal array as shown in Table 5.

Step 3: The linear graph was modified with different combinations of 1s, 2s and 3s from columns 6, 18 and 28 without having to consider the 35th column, as shown in Table 6. In Table 6, only a few rows of the array are shown, to save space.

Step 4: Columns 6, 18, 28 and 35 in the original L_{81} (3^{40}) orthogonal array were replaced with the modified nine level column.

The process was repeated with the other linear graphs shown in Fig. 13 to achieve six columns, each with nine levels of values, in the original orthogonal array. The first few rows of the resulting multilevel L_{81} ($9^6 \times 3^{16}$) array are shown in Table 6.



13 Linear graphs of L_{81} (3^{40}) orthogonal array for columns *a* 6, 18, 28 and 35, *b* 1, 8, 9 and 10, *c* 2, 14, 17 and 20, *d* 3, 23, 27 and 31, *e* 4, 32, 37 and 39, and *f* 5, 15, 24 and 33, to capture interactions among variables

Table 5 Columns of L_{81} (3^{40}) orthogonal array selected on the basis of the linear graph shown in Fig. 13*a*

Column case	6	18	28	Modified	35
1	1	1	1	1	1
2	1	2	2	2	2
3	1	3	3	3	3
4	2	1	2	4	3
5	2	2	3	5	1
6	2	3	1	6	2
7	3	1	3	7	2
8	3	2	1	8	3
9	3	3	2	9	1

Table 6 First nine rows of the modified multilevel L_{81} ($9^6 \times 3^{16}$) orthogonal array

Variable	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22
Case 1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
Case 2	1	2	2	2	2	2	1	1	1	2	2	2	2	2	2	2	2	2	2	2	2	2
Case 3	1	3	3	3	3	3	1	1	1	3	3	3	3	3	3	3	3	3	3	3	3	3
Case 4	2	1	2	3	4	4	2	2	2	1	1	1	1	2	2	2	2	3	3	3	3	3
Case 5	2	2	3	1	5	5	2	2	2	2	2	2	2	3	3	3	3	1	1	1	1	1
Case 6	2	3	1	2	6	6	2	2	2	3	3	3	3	1	1	1	1	2	2	2	2	2
Case 7	3	1	3	2	7	7	3	3	3	1	1	1	1	3	3	3	3	2	2	2	2	2
Case 8	3	2	1	3	8	8	3	3	3	2	2	2	2	1	1	1	1	3	3	3	3	3
Case 9	3	3	2	1	9	9	3	3	3	3	3	3	3	2	2	2	2	1	1	1	1	1

Acknowledgements

This research was supported by a grant from the US Department of Energy, Energy Programs Office under DE-FC07-011D14204. Mr Amit Kumar gratefully acknowledges the award of a Fellowship from the American Welding Society. The authors thank Dr J. M. Vitek of Oak Ridge National Laboratory and Mr S. Mishra of Penn State for many helpful comments. We thank Ms Marie Quintana of Lincoln Electric and Dr S. S. Babu of Oak Ridge National Laboratory for their interest in the work.

References

1. I. S. Kim, S. H. Lee and P. K. D. V. Yarlagadda: *Sci. Technol. Weld Joining*, 2003, **8**, (5), 347–352.
2. H. B. Smartt and J. A. Johnson: in Proceedings of 'Artificial neural networks in engineering, 'ANNIE'91', St Louis, MO, (ed. C. H. Dagli, S. R. T. Kumara and Y. C. Shin), 711–716; New York, ASME, 1991.
3. G. E. Cook, K. Andersen, G. Karsai and K. Ramaswamy: *IEEE Trans Indust. Applic.*, 1990, **26**, 824–830.
4. X. Li, S. W. Simpson and M. Rados: *Sci. Technol. Weld. Joining*, 2000, **5**, (2), 71–79.
5. H. K. D. H. Bhadeshia: *ISIJ Int.*, 1999, **39**, 966–979.
6. J. M. Vitek, Y. S. Iskander and E. M. Oblow: *Weld. J.*, 2000, **79**, (2), 33–40.
7. S. H. Lalam, H. K. D. H. Bhadeshia and D. J. C. Mackay: *Sci. Technol. Weld. Joining*, 2000, **5**, (3), 135–147.
8. E. A. Metzbower, J. J. Deloach, S. H. Lalam and H. K. D. H. Bhadeshia: *Sci. Technol. Weld. Joining*, 2001, **6**, (2), 116–124.
9. I. S. Kim, Y. J. Jeong, C. W. Lee and P. K. D. V. Yarlagadda: *Int. J. Advanced Manufacturing Technol.*, 2003, **22**, (9–10), 713–719.
10. D. S. Nagesh and G. L. Datta: *J. Mater. Process. Technol.*, 2002, **123**, (2), 303–312.
11. K. H. Christensen, T. Sorensen and J. K. Kristensen: *Sci. Technol. Weld. Joining*, 2005, **10**, (1), 32–43.
12. A. De, J. Jantre and P. K. Ghosh: *Sci. Technol. Weld. Joining*, 2004, **9**, (3), 253–259.
13. C. S. Wu, T. Polte and D. Rehfeldt: *Sci. Technol. Weld. Joining*, 2000, **5**, (5), 324–328.
14. C. H. Kim, W. Zhang and T. DebRoy: *J. Appl. Phys.*, 2003, **94**, 2667–2679.
15. W. Zhang, C. H. Kim and T. DebRoy: *J. Appl. Phys.*, 2004, **95**, 5210–5219.
16. W. Zhang, C. H. Kim and T. DebRoy: *J. Appl. Phys.*, 2004, **95**, 5220–5229.
17. A. Kumar and T. DebRoy: *Int. J. Heat Mass Transfer*, 2004, **47**, 5793–5806.
18. A. Kumar, W. Zhang and T. DebRoy: *J. Phys. D Appl. Phys.*, 2005, **38**, 119–126.
19. A. Kumar and T. DebRoy: *J. Phys. D Appl. Phys.*, 2005, **38**, 127–134.
20. A. Kumar, W. Zhang, C. H. Kim and T. DebRoy: in 'Mathematical modelling of weld phenomena 7', Graz, Austria, (ed. H. Cerjak, H. K. D. H. Bhadeshia and E. Kozeschnik), 1–35; 2005, London, Maney Publishing.
21. A. Kumar and T. DebRoy: *J. Appl. Phys.*, 2003, **94**, (2), 1267–1277.
22. A. De and T. DebRoy: *Weld. J.*, 2005, **84**, (7), 101–112.
23. A. De and T. DebRoy: *J. Phys. D Appl. Phys.*, 2004, **37**, 140–150.
24. A. Kumar and T. DebRoy: *Metall. Mater. Trans A.*, 2005, **36A**, 2725–2735.
25. S. Kumar and S. C. Bhaduri: *Metall. Trans B*, 1994, **25**, 435–441.
26. S. V. Patankar: 'Numerical heat transfer and fluid flow', 41–71; 1980, New York, NY, Hemisphere.
27. S. Haykin: 'Neural network: a comprehensive foundation', 2nd edn., 156–245; NJ, Prentice Hall.
28. M. T. Hagan: 'Neural network sesign', 1st edn., 145–178; Boston, PWS.
29. T. Masters: 'Practical neural network recipes in C++', 64–127; Boston, Academic Press.
30. B. L. Kalman and S. C. Kwasny: *Proc. Int. Joint Conf. Neural Networks*, 1992, **4**, 578–581.
31. C. Charalambus: *IEE Proc. G*, 1992, **139**, 301–310.
32. W. H. Press, B. P. Flannery, S. A. Teukolsky and W. T. Vetterling: 'Numerical recipes in C', 2nd edn., 274–305; 1997, Cambridge, Cambridge University Press.
33. R. P. Brent: 'Algorithms for minimization without derivatives', 86–124; 1993, NJ, Prentice-Hall.
34. D. E. Goldberg: 'Genetic algorithm in search, optimization and machine learning', 45–78; 1989, Reading, MA, Addison-Wesley.
35. K. Deb, A. Anand and D. Joshi: *Evolutionary Computation*, 2002, **10**, 371–395.
36. A. Kumar, S. Mishra, J. W. Elmer and T. DebRoy: *Metall. Mater. Trans A.*, 2005, **36**, 15–22.
37. S. Mishra and T. DebRoy: *J. Phys.D: Appl. Phys.*, **38**, 2977–2985.
38. S. Mishra and T. DebRoy: *J. Appl. Phys.*, **98**, (4).
39. R. P. Lippman: *IEEE ASAP Magazine*, 1987, **4**, 4–22.
40. J. Makhoul, A. El-jaroudi and R. Schwartz: *Proc. Int. Joint Conf. Neural Nets*, 1989, **1**, 455–460.
41. G. S. Peace: 'Taguchi methods: a hands-on-approach', 364–389; 1993, Reading, MA, Addison-Wesley.