Neuro hybrid model to predict weld bead width in submerged arc welding process

J Edwin Raja Dhas* and Somasundaram Kumanan2

1Department of Automobile Engineering, Noorul Islam University, Nagercoil 629 180, India
2Department of Production Engineering, National Institute of Technology, Tiruchirappalli 620 015, India

Received 09 November 2009; revised 09 February 2010; accepted 10 February 2010

This paper presents development of neuro hybrid model (NHM) to predict weld bead width in submerged arc welding. Experiments were designed using Taguchi’s principles and results were used to develop a multiple regression model. Data set generated from Multiple Regression Analysis (MRA) was utilized in ANN model, which was trained with backpropagation algorithm in MATLAB platform and used to develop NHM to predict quality of weld. NHM is flexible and accurate than existing models for a better online monitoring system.

Keywords: Bead width, Hybrid neuro model, Multiple regression model, Submerged arc welding

Introduction

Submerged arc welding (SAW) process1-7 is widely used in heavy fabrication industries due to its high quality, reliability, attaining deeper penetration and a smooth bead. Mathematical models were used to model welding process8-10. Fractional factorial technique11, statistical control technique12, curvilinear regression models13 and linear regression models14 have been used to model welding process. Response surface methodology (RSM) was used to predict weld bead quality in SAW of pipes15. Multiple regression analysis (MRA) technique16 has been applied to predict process parameters for gas metal arc welding. Taguchi method and regression analysis were used to determine weld process parameters17,18. Artificial neural network (ANN), fuzzy logic and expert system have emerged as competent tools for real time monitoring of weld process. ANNs with back-propagation learning algorithm19-21 has been used to model different arc welding processes but with no guarantee of an optimal solution. Genetic algorithm (GA), which gives near optimal solutions for any kind of problems22,23, guides weight updating process in ANN technique to build fast, accurate predictive networks. Hence, back propagation weight changing mechanism of ANN is replaced with GA operators in neuro hybrid system. A neurofuzzy (ANFIS) system has been used to predict weld bead width (WBW) in SAW process24. This study presents development of a neuro hybrid model (NHM) to predict WBW in SAW process for a given set of welding parameters.

Proposed Methodology [Neuro Hybrid Model (NHM)]

Data Acquisition

Experiment was carried out at M/s Ind Auto Products, Tiruchirappalli, India in semiautomatic SAW machine (SURARC of type XRPC 1200) with Taguchi’s design of experiments using L8 orthogonal array (factors, 4; levels, 2). Two mild steel plates (500 mm x 50 mm x 6 mm) of IS 2062 grade (Fe, 98.8%; C, 0.25%; Si, 0.20%; Mn, 0.75%) were welded with a square butt joint at a single pass in a SAW machine keeping electrode (diam, 3.15 mm) positive and perpendicular to plate. Samples (width, 10 mm) were cut from test piece. Then specimens were cleaned, polished and etched. Profile projector measured WBW (Table 1).

Multiple Regression Analysis (MRA)

MRA was performed using a statistical package. Relationship between WBW and its dependent variables (Fig. 1) is given as: 

\[ \text{WBW (mm)} = -34.833 + (6.667 \times 10^{-2} \times \text{welding current, amperes}) + (0.750 \times \text{arc voltage, volt}) + (1.25 \times 10^{-2} \times \text{welding speed, mm/min}) - (4.17 \times \text{welding speed, mm/min}) \]
Validity of this equation is confirmed by calculating R²-value (0.814), which has high coefficient of correlation.

Development of ANN for Weld Quality Prediction

Back propagation neural network (BPNN) is to map a set of input patterns to a corresponding set of output patterns by learning from a series of input – output data set. Then network performance is tested for patterns to predict appropriate outputs. In SAW, WBW is changed due to complicated welding conditions, and accurate mapping is needed to produce desired WBW according to welding parameters. Architecture of developed ANN (Fig. 2) is feed forward BPNN trained with Levenberg-Marquardt back propagation algorithm in MATLAB platform (Appendix 1). Number of samples for training and testing were 51 and 5 respectively. Learning function was gradient descent algorithm with momentum weight and bias learning function. Number of hidden layers and neurons were determined through a trial and error method, in order to accommodate converged error. Structure of proposed ANN is 4-12-9-1 (4 neurons in input layer, 12 neurons in 1st hidden layer, 9 neurons in 2nd hidden layer and 1 neuron in output layer). With a learning rate of 0.55 and a momentum term of 0.9, ANN was trained for 10000 iterations. Error between desired and actual outputs was < 0.001 after training ANN.

Development of NHM for Weld Quality Prediction

Flowchart (Fig. 3) depicts stages of NHM for weld quality prediction. Topology of ANN model developed
Results

Bead width

Network prediction

Network weight matrix

Network with optimal weight set

Evaluation of objective function

Results

Bead width

Generation of optimized weights

Data sets from experimentation/MRA

Cross over

Mutation

Fig. 2 — ANN architecture

Fig. 3 — Schematic diagram of proposed neuro hybrid model (NHM)
remained same. Weights of ANN were updated using GA to achieve optimum weight set. Data set required for training and testing NHM was obtained from experimental results and MRA. Values of different network parameters were as follows: number of training data sets, 43; number of testing data, 8; fitness function, mean squared error; population size, 40; maximum number of iterations, 250; selection, best half of the population; crossover probability, 80%; and mutation probability, 0.5%. Weight updation is explained as follows:

**Encoding Weights**

Weights and bias of network are represented in the form of chromosomes. Each chromosome represents one neural weight bias set. Initial population of chromosomes were randomly generated. Length of chromosome is number of links between neurons. For four input layer neurons, four hidden layer neurons and one output neuron, there were 20 links thus length of chromosome was 20 to correspond 20 weights.

**Fitness Function**

Fitness function for evaluating chromosome’s performance estimates performance of a given ANN. Training set obtained through performing experiments (Table 1) was presented to ANN, and mean squared error was calculated. GA finds a set of weights that minimizes mean squared errors (difference between target and predicted WBW).

**Ranking**

Individuals (chromosome) in population were evaluated and ranked. Since number of individuals in each population was kept constant, for each new individual an old one having worst fitness value was discarded.

**Selection**

Selection of individuals for crossover and mutation was biased towards fittest chromosomes. Top half of ranked chromosomes were selected for production of new offspring. All above steps are repeated till stopping criteria are reached. When ANN reaches with a best chromosome, optimal weights are stored. NHM was trained and tested for different conditions. Results obtained from BPNN and NHM are compared (Table 2) with results from MRA.

**Validation of NHM for Weld Quality Prediction**

Confirmatory experiments are done to validate NHM to predict WBW (Table 3). Error (%) = [(Experimental value – Predicted value) / Predicted value] X 100. Error (%) between results predicted by NHM and experimental values were not found significant.

**Results and Discussion**

Experiments were conducted in SAW machine using Taguchi’s principles. Results from experimentation were used to generate more sets of data from MRA to train proposed ANN. Time elapsed by NHM to predict weld

---

**Table 2—Weld bead width predicted by MRA, ANN and neuro hybrid model (NHM)**

<table>
<thead>
<tr>
<th>Welding current A</th>
<th>Arc voltage V</th>
<th>Welding speed mm/min</th>
<th>Electrode extension mm</th>
<th>Weld bead width, mm</th>
<th>Error MRA</th>
<th>Error ANN</th>
<th>Error NHM</th>
</tr>
</thead>
<tbody>
<tr>
<td>360</td>
<td>25</td>
<td>410</td>
<td>25</td>
<td>12.0007</td>
<td>12.0036</td>
<td>12.0011</td>
<td>0.0029</td>
</tr>
<tr>
<td>360</td>
<td>26</td>
<td>420</td>
<td>25</td>
<td>12.6257</td>
<td>12.6244</td>
<td>12.6214</td>
<td>0.0013</td>
</tr>
<tr>
<td>370</td>
<td>25</td>
<td>410</td>
<td>25</td>
<td>12.7925</td>
<td>12.7926</td>
<td>12.7922</td>
<td>0.0001</td>
</tr>
<tr>
<td>370</td>
<td>26</td>
<td>420</td>
<td>21</td>
<td>12.6674</td>
<td>12.3703</td>
<td>12.4215</td>
<td>0.2971</td>
</tr>
<tr>
<td>370</td>
<td>26</td>
<td>410</td>
<td>25</td>
<td>13.5842</td>
<td>13.9179</td>
<td>13.7491</td>
<td>0.3337</td>
</tr>
<tr>
<td>380</td>
<td>25</td>
<td>400</td>
<td>19</td>
<td>13.2924</td>
<td>13.5806</td>
<td>13.332</td>
<td>0.2882</td>
</tr>
</tbody>
</table>

**Table 3—Results from confirmatory experiments**

<table>
<thead>
<tr>
<th>Welding current A</th>
<th>Arc voltage V</th>
<th>Welding speed mm/min</th>
<th>Electrode extension mm</th>
<th>Bead width, mm</th>
<th>Error Experimentation</th>
<th>Error NHM</th>
<th>Error %</th>
</tr>
</thead>
<tbody>
<tr>
<td>390</td>
<td>26</td>
<td>410</td>
<td>19</td>
<td>14.7</td>
<td>14.900</td>
<td>0.0004</td>
<td>-0.13</td>
</tr>
<tr>
<td>370</td>
<td>25</td>
<td>420</td>
<td>21</td>
<td>12.5</td>
<td>12.508</td>
<td>0.0043</td>
<td>-0.06</td>
</tr>
<tr>
<td>380</td>
<td>26</td>
<td>400</td>
<td>25</td>
<td>14.0</td>
<td>14.209</td>
<td>0.0003</td>
<td>-1.47</td>
</tr>
<tr>
<td>360</td>
<td>26</td>
<td>400</td>
<td>19</td>
<td>13.7093</td>
<td>13.962</td>
<td>0.6667</td>
<td>2.527</td>
</tr>
</tbody>
</table>
quality was 32% less than that of BPNN. Computing machine used was Intel Pentium IV 1.8 GHz processor, 512 MB RAM and 80 GB Hard Disk Drive.

Conclusions
This paper presented development of welding quality predictor using neuro hybrid model. Taguchi method was successfully employed to model weld quality using L₈ orthogonal array.

Acknowledgement
Authors thank Ministry of Human Resource Development [MHRD], Govt of India for sponsoring experimental work under its R & D program.

Appendix 1 — MATLAB code
Load input and output data sets
Set input and output range
Declare
IN = iprange, % declaring input range
OUT = oprange, % declaring output range
IP = input, % declaring input matrix
OP = output, % declaring output matrix
S1 = size(IP), % size of the input matrix
p = S1(1), % number of input parameters (i.e. no. of rows)
q = S1(2), % number of training sets (no. of columns)
S2 = size(OP), % size of the output matrix
y = S2(1), % number of output parameters
r = 2, % number of layers
h = 14, % number of neurons in the hidden layer
G = 5000, % number of generation
P = 40, % number of population

Normalization of input data
For i=1:p
For j=1:q
IP(i,j) = 0.1+0.8*((IP(i,j)-min(IN(i,:)))/(max(IN(i,:))-min(IN(i,:))))
End
End

Normalization of output data
For i=1:y
For j=1:q
OP(i,j) = 0.1+0.8*((OP(i,j)-min(OP(i,:)))/(max(OP(i,:))-min(OP(i,:))))
End
End

M = (x*h)+(h*y)+h+y, % X = number of elements in a population
For i = 1:M
R(i,1) = -1, % R = Range of values
R(i,2) = 1
End
W = cell(P,2), % weight initialization cell
HOP = cell(P,2), % hidden layer output initialization
crom = initp(PR), % chromosome initialization
chroml = chrom, % where chroml is initially generated chromosomes or wt. set
n=1, % n = iteration number

Determining objective values for all populations
For j = 1:P
W(j,1) = reshape((chrom(j,1:x*h)),h,x), % reshaping into first weight matrix
W(j,2) = reshape((chrom(j,1+(x*h):(M-h-y))),y,h), % reshaping into second weight matrix
K(j,1) = W(j,1)*IP
b(j,1) = reshape((chrom(j,(1+(x*h)+(h*y)):(M-y))),h,1), % reshaping into first layer bias matrix
b(j,2) = reshape((chrom(j,(1+(M-y)):M)),n,1), % reshaping into second layer bias matrix
For r=1:q
b1 = K(j,1(:,r))+b(j,1);
K(j,1(:,r)) = b1;
End

HOP(j,1) = tansig(K(j,1)), % output data from hidden layer
K(j,2) = W(j,2)*HOP(j,1)
For r=1:q
bias2 = K(j,2(:,r))+b(j,2)
K(j,2(:,r)) = bias2
HOP(j,2) = purelin(K(j,2)); % output data from output layer
End

Calculation of objective function (mean square error)
objV(j,1) = (sum(sum((OP-HOP(j,2)).^2))/q);
For i = 1:G
minerror(i) = min(objV);
gen = 1;
while (objV(gen)~=minerror(i))
gen = gen+1;
% disp('string of least error in generation')
disp(n)
disp(gen)
disp(minerror(i))
WLE(i,:) = chrom(gen,:); % where WLE (wt. matrix for least error)
end
% Condition for stopping the iteration in between generation
if minerror(i) < 0.0001
break, end
if n==G
break, end
n=n+1
[xx,index] = sortrows(objV)
newchromes = index(1:P/2,:)
newchrom1 = chrom (newchromes,:)
ewchrom2 = xovshrs (newchrom1,0.8), % functions for crossover
newchrom = mutate(newchrom2,0.08), % function for mutation
newchrom = mutbga(newchrom2,R,[0.08 1 16])
For j = 1:P/2
W[j,1] = reshape((newchrom(j,1:x*h)),h,x); % reshaping into first weight matrix
W[j,2] = reshape((newchrom(j,1+(x*h)):(M-h-y)),y,h); % reshaping second weight matrix K[j,1] = W[j,1]*IP;
b[j,1] = reshape((newchrom(j,(1+i)+(h*y)):(M-y)),h,1); % reshaping first layer matrix
b[j,2] = reshape((newchrom(j,(1+(M-y)):M)),y,1); % reshaping second layer bias matrix
For r=1:q
    bias1 = K[j,1](:,r)+b[j,1]
    K[j,1](:,r) = bias1
    HOP[j,1] = tansig(K[j,1]);
    K[j,2] = W[j,2]*HOP[j,1];
End

For r=1:q
    bias2 = K[j,2](:,r)+b[j,2];
    K[j,2](:,r) = bias2;
    HOP[j,2] = purelin(K[j,2]);
End

**Calculation of mean squared error**

\[
\text{objv2}(j,1) = \frac{\text{sum}(\text{sum}((\text{OP}-\text{HOP}[j,2]).^2))}{q}
\]

\[
\text{chrom} = [\text{newchrom1};\text{newchrom}]
\]

\[
\text{objV} = [\text{objv1};\text{objv2}];
\]

% display of minimum error
for z=1:n disp(minerror(z))

**List minimum error of all generations**

\[
\text{least} = \min(\minerror(z))
\]

\[
\text{gen} = 1
\]

while(minerror(gen)==least)
    gen = gen + 1
End

disp('least error in generation no = ')
disp(gen)
disp('weight set corresponding to least error is')
wm = WLE(gen,:);
% wm is the matrix with least error

**References**


