Artificial Neural Networks Applied to Arc Welding Process Modeling and Control

KRISTINN ANDERSEN, GEORGE E. COOK, FELLOW, IEEE, GABOR KARSAI, MEMBER, IEEE, AND KUMAR RAMASWAMY

Abstract—**Artificial neural networks have been studied for applicability for modeling and control of physical processes. Some basic concepts relating to neural networks are explained as well as how they can be used to model weld bead geometry in terms of the equipment parameters selected to produce the weld. Approaches to utilization of neural networks in process control are discussed as well. The need for modeling transient as well as static characteristics of physical systems for closed-loop control is pointed out, and an approach to achieving this is presented. The performance of neural networks for modeling is presented and evaluated using actual welding data. It is concluded that the accuracy of neural network modeling is fully comparable with the accuracy achieved by more traditional modeling schemes.**

Introduction

WELD MODELING is important for advancing knowledge on the mechanics of weld processes and how they can be best controlled and utilized. Modeling of the weld bead geometry has turned out to be nontrivial, however, and therefore a number of approaches have been devised to attack the problem. Generally, the arc welding processes are substantially nonlinear, in addition to being highly coupled multivariable systems. Frequently, not all the variables affecting welding quality are known, nor may they be easily quantified. Examples include contamination, workpiece heat absorption along the weld, and various environmental conditions. All this perplexity contributes to the difficulties of designing reliable welds and the equipment used to produce them. Frequently, the experience and knowledge of the human welder provides the last steps toward a reliable weld. The approach of neural network methodologies presented here is proposed to aid the weld designer and the welder in attaining the required weld specifications with minimal experimentations.

Gas tungsten arc welding (GTAW) will be used in the remainder of this paper to exemplify modeling of a welding process using a neural network. An arc is initiated and sustained between a pointed tungsten electrode and the surface of the welded workpiece. Inert gas, such as argon or helium, is conducted coaxially down around the arc and thus it shields the molten weld pool from the atmosphere. Usually filler wire is fed into the weld pool where it melts and adds to the molten workpiece material. The torch normally moves with the filler wire guide along the surface of the workpiece, tracing the joint to be welded.

The physical geometry of the molten pool is a major factor in determining the structural adequacy of the weld, such as its strength. Such factors, which characterize the finished weld, will be referred to here as direct weld parameters (DWP's) [1]. They include the penetration of the weld pool, the bead width, the transverse cross-sectional area, and the height of the reinforcement. These direct weld parameters are governed by numerous factors, e.g., welding voltage, current, torch travel speed, wire feed rate, electrode tip angle, and shielding gas type and flow rate. Collectively, they are referred to as indirect weld parameters (IWP's). The task of the human welder is to select and control the IWP's in such a way as to obtain some desired DWP values. This task is complicated by two challenges. First, the multivariable system is closely coupled, making adjustment of any single DWP difficult without affecting the others. For critical applications this usually requires substantial experimentations. Secondly, not all output variables can be observed in real time. For example, while bead width can be monitored while welding is in progress, the pool penetration is usually unknown on-line. A practical approach to estimating DWP's that cannot be sensed is using a reliable model of the weld pool describing the pool geometry in terms of the heat input and other controllable parameters. This is equivalent to a mapping from a set of indirect weld parameters to another one of direct weld parameters. In addition to providing estimates of the DWP's a model can be beneficial in decoupling the system parameters.

Models of the weld pool are primarily based on the physics of the weld or empirical data. Frequently models combine both approaches to some extent; for example physics-derived models often include "efficiency" or other parameters that may be adjusted to fit the model to actual welding data. An alternative classification of weld models defines them as either static or dynamic. The static models describe the relations between process variables after they have reached equilibrium while the dynamic models provide additional analysis of the process during transitions. Therefore, the dynamic models are preferable for real-time control purposes. An overview of current models of the GTAW process

Paper IUSD 89-8, approved by the Industrial Control Committee of the IEEE Industry Applications Society for presentation at the 1989 Industry Applications Society Annual Meeting, San Diego, CA, October 1-5. Manuscript released for publication February 20, 1990.

K. Andersen, G. E. Cook, and G. Karsai are with the Department of Electrical Engineering, P.O. Box 1824 B, Vanderbilt University, Nashville, TN 37235.

K. Ramaswamy was with the Department of Electrical Engineering, Vanderbilt University, Nashville, TN. He is now with Rensselaer Polytechnic Institute at 2235 14th Street, Troy, NY 12180.

IEEE Log Number 9037064.

is given by Andersen *et al.* [2]. An early static model derived from the physics of heat flow was that by Rosenthal [3] and related works have been carried out by Nunes [4] and Tsai [5]. These models described the weld pool contour as a function of heat input, electrode velocity, and material properties, all of which were assumed to be constant. The model by Nunes was verified by Prasad *et al.* [6] through comparison with actual welding data. Bates and Hardt [7] used Tsai's model to devise a decoupling and modeling scheme to control pool width and penetration independently. Yet another static physics-based models is the one proposed by Smartt and Key [8], which assumes hemispherical pool shape and ignores the effects of a nonzero electrode speed. A totally different approach was taken by Hunter *et al.* [9] who devised a static weld model primarily based on empirical data. Weld parameters obtained from experiments were accumulated and fitted to a mathematical equation by multiple regression, and the resulting equations formed the weld model. Smartt *et al.* [10] devised a static weld model for gas metal arc welding (GMAW) which relies on both the physics of the process and regression of experimental data. Finally, Hardt *et al.* [11] have developed a cylindrical model of the weld pool, applying only to full-penetration GTAW. Unlike the models cited before, this one described the dynamics of the pool shape. Experimental results were used to verify the model. In addition to welding-specific models, such as the ones outlined earlier, finite-element algorithms provide relatively accurate information about the weld pool. One such numerical model is the discrete-element model developed by Eraslan *et al.* [12]. Although this and other models using temporal and spatial discretization are more general and frequently more accurate than the application-specific models listed earlier, they usually require far more computations. Therefore, they are better suited for off-line simulations than real-time control purposes.

The modeling technique presented here using neural networks is empirically based. The model is based on recorded welding data rather than the underlying heat flow physics. As illustrated at the end of the following section this approach has certain advantages as well as disadvantages as compared to more traditional models.

Neural Networks

Recent successes in employing artificial neural network models for solving various computationally difficult problems have inspired renewed research in the area. Early work by McCulloch [13] and Widrow [14] focused largely on mathematical modeling, while more recent research has augmented theoretical analysis with computer simulations and implementation demonstrations. Numerous variants of pattern classifiers using neural networks have been studied by Hopfield [15] and Lippmann [16]. Introductory texts on the subject may be found in [17] and [18].

As the name indicates, a neural network resembles, to a certain degree, biological nervous systems as we currently understand them. While most traditional computers rely on a single or few computational units to perform their tasks, the neural network typically consists of a relatively large number of computational units, connected with an even larger number of communication links. The underlying principle aims to examine numerous hypotheses simultaneously and process data in a distributed fashion. In essence, the neural network is a self-adaptive structure which incrementally alters its inner workings until it achieves the desired performance.

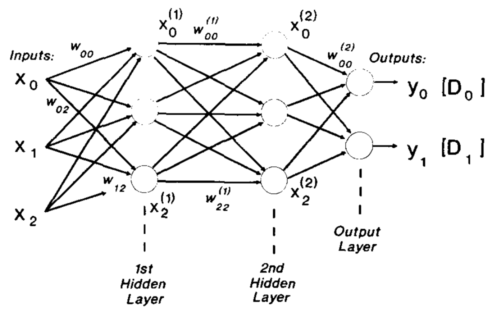


Fig. 1. Three-input, two output neural network, using two hidden layers of three nodes each.

A neural network and its adaptation procedure using back propagation is best illustrated by an example. Fig. 1 shows a small neural network consisting of eight *nodes*, arranged in two *hidden layers* of three nodes each and one *output layer* of two nodes. Each node i in the first hidden layer produces a single numeric output which we denote as $x_i^{(1)}$. Similarly, the nodes of the second hidden layer produce output values labeled $x_0^{(2)}$ through $x_2^{(2)}$. Specifically, the three inputs to the network are labeled x_0 through x_2. The two outputs of the neural net are y_0 and y_1. Each node of the network accepts numeric data through a number of input links, each of which multiplies the input data with a *weight* factor. The weight factor associated with the link from $x_i^{(1)}$ to the node producing $x_j^{(2)}$ is annotated as $w_{ij}^{(1)}$ and a similar convention holds for the links between other layers. Each node calculates its output by summing its weighted inputs and using the result s as the argument of a nonlinear function associated with the node. For our application this function is the same for all nodes:

$$f(s) = \left[1 + \exp\left[-(s - c)\right]\right]^{-1} \tag{1}$$

where s is the sum of the node inputs and c is an internal offset value. Clearly, the node output will be confined to the range $0 < f(s) < 1$. Because the limiting values 0 and 1 will only be approached as s approaches $+/-$ infinity, all input and output data are scaled so that they are confined to a subinterval $[0 \cdots 1]$. A practical region for the data is chosen to be $[0.1 \cdots 0.9]$. In this case each input or output parameter p is normalized as p_n before being applied to the neural network according to

$$p_n = \left[(0.9 - 0.1)/(p_{\max} - p_{\min})\right](p - p_{\min}) + 0.1 \tag{2}$$

where $p_{\max}$ and $p_{\min}$ are the maximum and minimum values, respectively, of data parameter p. The network starts calculating its output values by passing the weighted inputs to the nodes in the first layer. The resulting node outputs of that layer are passed on, through a new set of weights, to the second layer, and so on until the nodes of the output layer compute the final outputs.

Before practical application, the network has to be *trained* to perform the mapping of the three input parameters to the two output parameters. This is done by repeatedly applying training data to its inputs, calculating the corresponding outputs by the network, comparing them to the desired outputs, and altering the internal parameters of the network for the next round. The training starts by assigning small random values to all weights w_{ij} and node offsets c_j in the network. The first three input data values are presented to the network which in turn calculates the two output values. Because the initial weights and node offsets are random, these values will generally be quite different from the desired output values D_0 and D_1. Therefore, the differences between the desired and calculated outputs have to be utilized to dictate improved network values, tuning each weight and offset parameter through back propagation. The weights preceding each output node are updated according to

$$w_{ij}(t+1) = w_{ij}(t) + \eta d_j x_i^{(2)} \tag{3}$$

where η is a *correction gain* and d_j is the *correction factor*:

$$d_j = y_j(1 - y_j)(d_j - y_j). \tag{4}$$

Clearly, each weight will be increased if the calculated output from its node is less than the desired value and vice versa. The correction factors used to update weights preceding hidden layer modes are updated according to

$$d_j = x_j \cdot (1 - x_j) \sum_k (d_k \cdot w_{jk}) \tag{5}$$

where k applies to the node layer succeeding the one currently being updated. The offset parameter c of each node is treated as an additional weight factor and updated in the same manner.

The weights and offsets of the neural network are recalculated during the back propagation as outlined before. Then the network repeats calculation of output values based on the same input data, compares them to the desired output values, and readjusts the network parameters through yet another back propagation phase. This cycle is repeated until the calculated outputs have converged sufficiently close to the desired outputs or an iteration limit has been reached. Once the neural network has been tuned to the first set of input/output data, additional data sets can be used for further training in the same way. To ensure concurrent network adaptation to all sets of data, the entire training process may be repeated until all data transformations are adequately modeled by the network. This requires, of course, that all the data sets were obtained from the same process and therefore the underlying input/output transformation is consistent.

As noted before, the training iteration process may be terminated either by a convergence limit or simply by limiting the total number of iterations. In the former case we use an error measure e defined as following

$$e = \max_{k=1 \cdots K} \left\{ \sum_{m=0}^{M-1} (d_{k,m} - y_{k,m})^2 \right\} \tag{6}$$

where K is the number if input/output data sets used for training, M is the number of network output parameters in each data set, and $(d_{k,m} - y_{k,m})$ is the error in the network calculation of parameter m in data set k. The error measure e changes after each round of network weight adjustments. In the long run e decreases as the network is refined by training iterations. Using this indicator one can program the network to terminate the iterative tuning process as soon as e reaches some threshold value e_0. Alternatively, a given network may not be able to reduce the error measure down to the specified e_0. In that case the iterations may be terminated by simply specifying a maximum number for them.

The training mode, as described earlier, is a precondition for actually applying the neural network in the application mode. In this mode, entire new input data are presented to the network which, in turn, predicts new outputs based on the transfer characteristics learned during the training. If these new data are obtained from the same local region of operation of the process as during the training phase, data from the input/output relations should be governed by the same underlying process and the neural network should perform adequately. The neural network is not updated in the application mode.

When compared to other modeling methodologies, neural networks have certain drawbacks as well as advantages. The most notable drawback is the lack of comprehension of the physics of the process. Relating the qualitative effects of the network structure or parameters to the process parameters is usually impossible. On the other hand, most physical models resort to substantial simplifications of the process and therefore trade accuracy for comprehensibility. The advantages of neural models include relative accuracy, as illustrated in the following section, and generality. If the training data for a neural network are general enough, spanning the entire ranges of process parameters, the resulting model will capture the complexions of the process, including nonlinearities and parameter cross couplings, over the same ranges. Model development is much simpler than for most other models. instead of theoretical analysis and development for a new model, the neural network tailors itself to the training data. The network can be refined at any time with the addition of new training data. Finally, the neural network can calculate its result relatively quickly, as the input data are only propagated once through the network in the application mode.

Application Examples for Neural Networks

The foregoing review of neural networks revealed how they can learn and implement the mapping between a number of input and output parameters. Note that these parameter sets can be selected arbitrarily. For the purpose of welding control, a specific neural network can be trained to derive DWP's from IWP's while another network can yield IWP's from DWP's. The previous application is the weld pool modeling problem, where the neural network derives the geometrical attributes of the weld pool from the parameters of the welding equipment. The latter application can be directly used to determine the equipment parameters necessary to achieve a certain pool geometry. A practical imple-

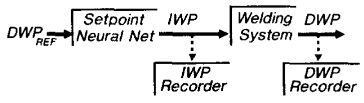


Fig. 2. Indirect weld parameter selector using neural network.

mentation scheme for such an application will be discussed now.

Selecting the appropriate indirect weld parameters for a given weld geometry is a nontrivial task. Generally, each parameter of the welding equipment affects a number of the geometrical attributes of the welding pool. Current, for instance, affects both weld width and penetration. The human welder or weld designer usually selects the indirect weld parameters based on previous experience, handbook recommendations, etc., and then fine tunes the selection with trial experiments. This approach can be complemented or replaced by the neural network scheme illustrated in Fig. 2. The user specifies the required weld geometry by the reference or setpoint DWP's ($\mathrm{DWP_{REF}}$). Assuming that the network has been properly trained off-line with previous sets of actual weld parameters, the network produces a set of IWP's that can be used as weld equipment settings. If the network parameters include the ones having principal roles in controlling the weld pool, the welder can achieve the required DWP's with minimal experimentations or experience. While the weld is performed, the IWP's may be recorded in the *IWP recorder*, and similarly, the resulting DWP's can be determined and stored in the *DWP recorder* after welding. Once these data are available the neural network can be refined by additional off-line training with the new data, and thus its characteristic are continuously updated as future welds are produced.

It is emphasized that this scheme applies to static parameter settings only. In this case dynamic control of the welding process is left to the various controllers of the welding equipment, such as the automatic voltage controller (AVC), power source (usually constant current for GTAW), etc. Thus, it is the responsibility of these controllers to maintain the parameters, selected by the neural network, constant in presence of noise or process perturbations.

An approach to closed-loop control, using neural networks, is illustrated in Fig. 3. As before, the set of IWP's, determined by the setpoint network, is used as a reference by the welding equipment. In parallel to the actual welding hardware a second neural network is fed with the same IWP's. This network has been trained off-line to model the welding process. As a result, it calculates its own set of DWP's, which are estimates of the actual DWP's created by the welding process. Combining the observable DWP's from the weld sensors with the remaining, estimated, DWP's from the modeling network, an on-line estimate of the entire DWP vector is obtained. The combined measured and estimated output DWP's of the entire DWP vector is obtained. The combined measured and estimated output DWP's of the process are compared against the reference DWP's, and an error vector is produced. This vector is entered to a proportional-integral-derivative (PID) controller which in turn provides the control signals to the setpoint neural network.

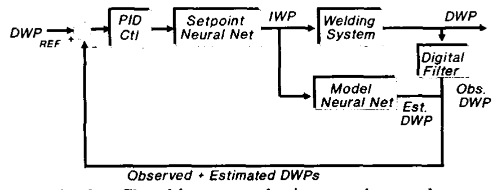


Fig. 3. Closed-loop control using neural networks.

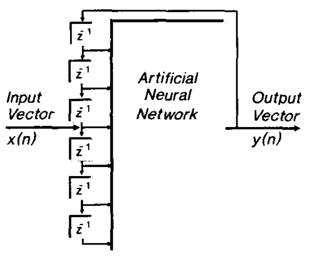


Fig. 4. Neural network scheme capturing dynamics of physical process.

A few observations and notes are in order here. First, using the setpoint neural network eliminates the need for explicit decoupling of the process parameters. Given a set of required DWP's the network directly finds the corresponding IWP settings. Secondly, the neural networks are taught by examples, which eliminates the need for explicit mathematical modeling or similar system analysis.

Dynamic modeling, using neural nets directly or other alternatives, is vital for complete process control and research on temporal networks is already ongoing. The capture of dynamic responses requires memory of past network states. Two rudimentary ways of accomplishing this are by means of realtime clock input, and by autoregressive feedbacks. In the former case, a single input parameter, time, is added to the network, while in the latter several inputs are added, each one accepting one of the output variables delayed by a fixed time lag (refer to Fig. 4). Research on these and similar structures is continuing [19].

As far as accuracy is concerned, static neural networks appear to yield quite satisfactory results. This will be demonstrated by the simulations described in the following section.

Neural Network performances

To evaluate the accuracy of neural networks for weld modeling, a number of simulations were carried out. Actual GTA welding data [6] were used for this purpose. The data consisted of values for voltage, current, electrode travel speed, and wire feed speed, and the corresponding bead width, penetration, reinforcement height, and bead cross-sectional area. In all, 42 such data sets were used, of which 31 data sets were selected at random and used for training purposes while the remaining 11 data sets were presented to the trained networks as new application data for evaluation purposes. Thus the networks were evaluated using data that had not been used for training.

Before training the networks for IWP–DWP mapping, decisions had to be made about fixed network parameters, such as the correction gain n (see (3)), and the error criterion, e_0. Preliminary simulation tests were carried out to

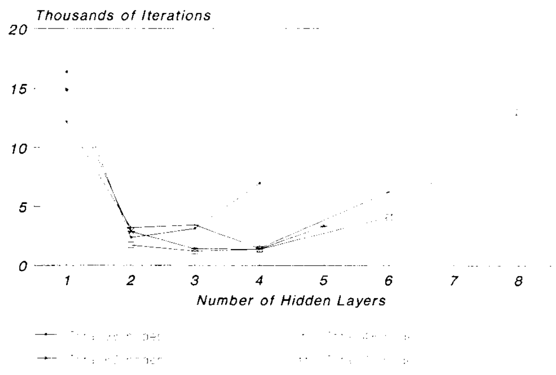


Fig. 5. Number of iterations required for convergences of various networks. Error threshold e_0 is 0.09 and correction gain η is 0.1 for all networks.

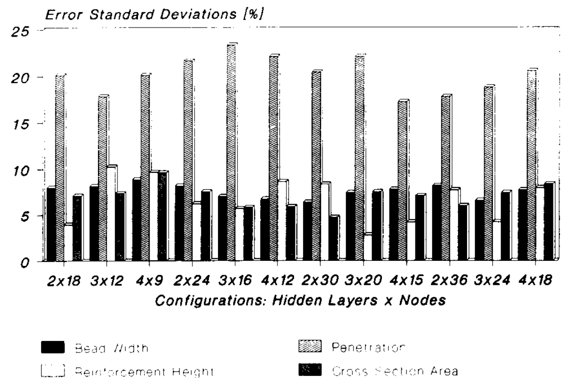


Fig. 6. Standard deviations of mapping errors (%) for various networks, shown for all four DWP's.

examine how these parameters affected training convergence rate and to determine feasible values for the remaining tests. A value of 0.09 was found to be suitable for e_0; substantially larger values yielded inaccurate network mappings while smaller error criteria required considerably more training iterations. Similarly, preliminary tests showed a correction gain of about 0.1–0.2 to be suitable. The trade-off there was slow training convergence at low gains against convergence instability at which correction gains.

Using $n = 0.1$ and $e_0 = 0.09$, numerous network configurations for mapping the IWP's to DWP's were tested and compared. Both the required iteration numbers and mapping performances were examined for these networks. Because the error criterion was fixed, all networks were expected to perform the mapping with a comparable accuracy, and therefore the first objective was to determine which configurations required the least number of training iterations. Networks consisting of 36, 48, 60, and 72 nodes were built, and for any total number of nodes various numbers of hidden layers were tested as well. For example, networks of 60 nodes were built with 36 nodes in a single hidden layer, 18 nodes in each of two layers, 12 nodes in each of three layers, and nine nodes in each of four layers. The 31 sets of welding data were used for training each of the networks. The number of training iterations require to reduce the convergence measure e down to 0.09 is illustrated in Fig. 5. A training iteration is defined here as one round of network adaptation to all sets of training data. Fig. 5 shows that, depending on network configuration, the number of training iterations ranges from just over 16 000 down to about 1200. For each total number of hidden layer nodes there appears to be an optimum number of hidden layers that requires the least amount of training iterations. Although the training iterations take place off-line and are therefore irrelevant to on-line modeling, they can be fairly time consuming. CPU time for a typical case of 1500 training iterations was about 1 h 50 min on a MicroVAX computer. Although the program was not optimized for speed and performed various auxiliary functions, it is clear that training time optimization is advantageous for extensive experimentations.

Because the error criterion for all networks was the same, their performances were comparable. Table I illustrates the modeling performance of the 2×18 node network which serves here as an example of a typical case. The DWP's estimated by the network are listed with the corresponding error percentages in Table II. The standard deviation of the bead width, penetration, reinforcement height, and cross-sectional area errors are 8.01, 20.18, 3.97, and 7.10%, respectively. Fig. 6 summarizes the performances of all networks for the four DWP's. The bars show the standard deviations of the modeling errors for the various networks using the 11 tested data sets. Most of the errors are on the order of 5–10%, with penetration errors typically in the 20% range.

The foregoing network configurations were tested again with randomly selected data from the original training data set. Interestingly, the errors were generally not smaller than those obtained from the untrained data. Using the 2×18 network again for a comparison the standard deviations of the four DWP's are, in the same order as before, 2.86, 7.75, 7.13, and 6.72%. This observation underscores the fact that a network of a finite size can only model given data to a limited degree. In most practical cases, network adaptation is terminated when a small residual mapping error is reached. Whether additional mappings, using new data, will be more accurate or not depends on the nature of the data.

Conclusion

In summary, models of the welding processes and related neutral network applications have been discussed. Neural networks are very general in the sense that they can be constructed with empirical data rather than analytical formulations. All fine tuning of the model to the data is built into the network structure, and additional data can be used for further refinements at any time. Although training of the neural network requires considerable computations, due to the iterations, application of the network takes only a single forward pass and is therefore relatively fast.

The simulations discussed indicate that relatively simple neural sets can yield fairly accurate results for static model-

TABLE I
DATA SETS USED FOR TRAINING (1–31) AND TESTING (32–42) THE NEURAL NETWORKS

Weld Number	Arc Current (A)	Arc Voltage (V)	Travel Speed (mm/s)	Wire Speed (mm/s)	Bead Width (mm)	Penetration (mm)	Reinforcement Height (mm)	Cross Section Area (mm^2)
1	150	9.6	1.91	2.12	6.95	2.60	0.37	1.87
2	150	9.6	1.91	4.23	7.04	2.70	0.64	2.87
3	150	9.6	1.91	6.35	6.93	2.63	0.86	4.13
4	150	9.6	1.91	8.47	6.87	2.47	1.02	4.94
5	100	9.0	1.19	2.12	5.49	1.75	0.59	2.55
6	100	9.0	1.19	4.23	5.23	1.77	0.88	3.33
7	100	9.0	1.19	6.35	5.51	1.50	1.43	5.34
8	100	9.0	1.19	10.58	5.47	1.18	2.10	8.68
9	100	9.0	1.82	2.12	4.60	1.74	0.60	2.05
10	100	9.0	1.82	6.35	4.64	1.62	1.07	3.51
11	100	9.0	1.82	8.47	4.68	1.29	1.25	4.18
12	150	9.6	2.88	4.23	6.81	1.93	0.47	2.12
13	150	9.6	2.88	6.35	7.23	2.00	0.60	2.55
14	150	9.6	2.88	8.47	7.05	1.83	0.78	3.51
15	150	9.6	1.44	2.12	7.45	3.21	0.42	2.42
16	150	9.6	1.44	4.23	7.58	3.10	0.63	3.47
17	100	9.0	0.89	2.12	5.59	1.29	0.66	2.75
18	100	9.0	0.89	4.23	5.66	1.85	1.09	4.53
19	100	9.0	0.89	6.35	5.53	1.36	1.64	6.59
20	200	10.5	4.19	2.12	7.66	1.33	0.21	0.92
21	200	10.5	4.19	2.12	6.64	1.51	0.21	0.83
22	200	10.5	4.19	4.23	8.00	1.50	0.34	1.58
23	200	10.5	4.19	10.58	7.86	1.06	0.59	2.98
24	200	10.5	4.19	12.70	8.12	1.11	0.69	3.66
25	200	10.5	4.19	16.93	8.66	0.72	0.82	4.73
26	200	10.5	4.19	21.17	8.75	0.55	0.94	5.61
27	200	10.5	2.12	8.47	9.76	2.50	0.78	4.93
28	200	10.5	2.12	16.93	11.09	1.11	1.18	8.51
29	200	10.5	2.79	12.70	9.45	1.58	0.83	5.11
30	200	10.5	2.79	16.93	9.58	0.89	1.05	6.52
31	200	10.5	2.79	8.47	9.31	2.28	0.63	3.64
32	100	9.0	1.19	8.47	5.35	1.43	1.89	7.27
33	100	9.0	1.82	4.23	5.70	1.73	0.69	2.64
34	150	9.6	2.88	10.58	6.06	1.94	0.97	4.23
35	150	9.6	1.44	6.35	7.55	2.86	0.95	5.06
36	100	9.0	0.89	8.47	5.78	1.44	2.13	9.44
37	200	10.5	4.19	6.35	7.80	1.58	0.44	1.85
38	200	10.5	4.19	8.47	8.09	1.80	0.53	2.59
39	200	10.5	4.19	14.82	8.27	0.87	0.75	4.00
40	200	10.5	2.12	2.12	10.08	2.16	0.42	2.55
41	200	10.5	2.12	12.70	10.24	1.76	0.97	6.48
42	200	10.5	2.79	2.12	9.35	2.05	0.30	1.74

TABLE II
DIRECT WELD PARAMETER ESTIMATES AND ERRORS FROM THE 2 × 18 NEURAL NETWORK

	Bead Width		Penetration		Reinforcement Height		Cross Sectional Area	
Weld Number	Estimated (mm)	Error (%)	Estimated (mm)	Error (%)	Estimated (mm)	Error (%)	Estimated (mm^2)	Error (%)
32	5.36	0.19	1.32	7.69	1.75	7.41	6.90	5.09
33	4.98	12.63	1.65	4.62	0.69	0.00	2.24	15.15
34	6.60	8.91	1.74	10.31	0.96	1.03	4.45	5.20
35	7.65	1.32	2.94	2.80	0.91	4.21	4.78	5.53
36	5.84	1.04	1.28	11.11	1.96	7.98	8.31	11.97
37	7.90	1.28	1.30	17.72	0.41	6.82	1.87	1.08
38	8.09	0.00	1.22	32.22	0.51	3.77	2.43	6.18
39	8.38	1.33	0.93	6.90	0.76	1.33	4.14	3.50
40	8.37	16.96	3.14	45.37	0.38	9.52	2.20	13.73
41	10.00	2.34	1.81	2.84	0.95	2.06	6.77	4.48
42	7.80	16.58	2.59	26.34	0.31	3.33	1.57	9.77

ing. Further work, addressing the dynamics of the welding processes is in progress.

References

[1] G. E. Cook, "Feedback and adaptive control in automated arc welding systems," *Metal Construction*, vol. 13, no. 9, pp. 551-556, Sept. 1981.

[2] K. Andersen, R. J. Barnett, G. E. Cook, K. Ramaswamy, and T. Prasad, "Intelligent gas tungsten arc welding control," *Phase I SBIR Final Report to NASA-MSFC, Sept.* 1987.

[3] D. Rosenthal, "Mathematical theory of heat distribution during welding and cutting," *Welding J.*, vol. 20, no. 5, pp. 220s-234s, 1941.

[4] A. C. Nunes, "An extended Rosenthal weld model," *Welding J.*, pp. 165s-170s, June 1983.

[5] N. Tsai, "Heat distribution and weld bead geometry in arc welding," Ph.D. dissertation, Dept. of Mater. Sci., Mass. Inst. Technol., Cambridge, Apr. 1983.

[6] T. Prasad *et al.*, "Computer implementation and study of a weld model," in *Proc. IEEE Southeastcon 89*, vol. 2, Columbia, SC, Apr. 9-12, 1989, pp. 517-521.

[7] B. E. Bates and D. E. Hardt, "A realtime calibrated thermal model for closed loop weld bead geometry control," *Trans. ASME J. Dynamic Syst., Meas. Contr.*, pp. 25-33, Mar. 1985.

[8] H. B. Smartt and J. F. Key, "An investigation of factors controlling GTA weld bead geometry," presented at Trends in Welding Research in the U.S., Idaho, Nov. 16-18, 1981, EGG M10581.

[9] J. J. Hunter, G. W. Bryce, and J. Doherty, "On-line control of the arc welding process," *Developments in Mechanised Automated and Robotic Welding*, WI, 1980.

[10] H. B. Smartt, C. J. Einerson, A. D. Watkins, and R. A. Morris, "Gas metal arc process sensing and control," in *Proc. 1985 ASM Int. Welding Congress*, Toronto, ON, Canada, Oct. 1985.

[11] D. E. Hardt, D. A. Garlow, and J. B. Weinert, "A model of full penetration arc welding for control system design," *Trans. ASME, J. Dynamic Syst., Meas., Contr.*, pp. 40-46, Mar. 1985.

[12] A. H. Eraslan, T. Zacharia, and D. K. Aidun, "WELDER: A computer code for simulating fast-transient, three-dimensional, three-phase, flow, temperature and material composition conditions during welding," Dep. of MIE, Clarkson Univ., Potsdam, NY, Report MIE-142, Oct. 1986.

[13] C. A. McCulloch and W. Pitts, "A logical calculus of the ideas of imminent in nervous activity," *Bull. Math. Biophys.*, vol. 5, pp. 115-133, 1943.

[14] B. Widrow and M. E. Hoff, "Adaptive switching circuits," in *1960 IRE WESCON Conv. Rec.,* pt. 4, Aug. 1960, pp. 96-104.

[15] J. J. Hopfield, "Neural networks and physical systems with emergent collective computational abilities," *Proc. Nat. Acad. Sci.*, vol. 79, pp. 2554-2558, Apr. 1982.

[16] R. P. Lippmann, B. Gold, and M. L. Malpass, "A comparison of Hamming and Hopfield neural nets for pattern classification," MIT Lincoln Lab., Tech. Rep., TR-769.

[17] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, *Parallel Distributed Processing*, vol. 12. Cambridge, MA: MIT Press, 1986.

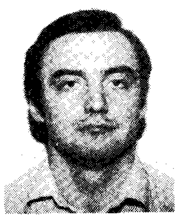
[18] R. P. Lippmann, "An introduction to computing with neural nets," *IEEE ASSP Mag.*, pp. 4-22, Apr. 1987.

[19] G. Karsai, K. Andersen, G. E. Cook, and K. Ramaswamy, "Dynamic modeling and control of nonlinear processes using neural network techniques," in *Proc. IEEE '89 Symp. Intelligent Control*, Albany, NY, Sept. 25-26, 1989, pp. 280-286.

Kristinn Andersen was born in Reykjavik, Iceland, in 1958. He received the degree in electrical engineering from the University of Iceland, Reykjavik, in 1982. He is currently working toward the Ph.D. degree at Vanderbilt University, Nashville, TN, on a Fullbright Scholarship.

He is employed part-time at Mid-South Engineering in Nashville, where he has been responsible for various research on industrial automation, including arc welding control and robotics for NASA. His current research interests include applied artificial intelligence, system modeling and control, and robotics. He has authored or coauthored over 30 papers on his research.

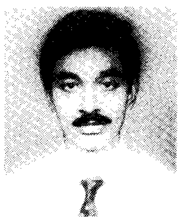
George E. Cook (F'88) is Associate Dean for Research and Professor of Electrical Engineering at Vanderbilt University School of Engineering, Nashville, TN. His research interests are robotics and automation with emphasis on welding automation.



Gabor Karsai (M'87) received the B.Sc. and M.Sc. degrees in electrical engineering from the Technical University of Budapest, Hungary, and the Ph.D. degree from Vanderbilt University, Nashville, TN, in 1982, 1984, and 1988, respectively.

He is currently Research Associate in the Department of Electrical Engineering at Vanderbilt University. His research interests include the design and implementation of advanced software systems for real-time, intelligent control, programming tools for graphical programming of process control systems, and the application of neural networks for solving difficult control problems.

Dr. Karsai is a member of the International Neural Network Society.



Kumar Ramaswamy was born on July 9, 1965. He received the B.E. degree (with honors) from the Regional Engineering College, Tiruchirapalli, India, and the M.S. degree in electrical engineering from Vanderbilt University, Nashville, TN, in 1986 and 1988, respectively. He is currently working toward the Ph.D. degree at Rensselaer Polytechnic Institute, Troy, NY.

Till early 1987, he was employed with the Center for the Development of Telematics, Bangalore, India, and was involved with the development of digital switching equipment. He was a Research Assistant while at Vanderbilt University. His interests include system identification, neural networks and digital communication theory.