AIAA JOURNAL
Vol. 33, No. 6, June 1995

Technical Notes

TECHNICAL NOTES are short manuscripts describing new developments or important results of a preliminary nature. These Notes cannot exceed 6 manuscript pages and 3 figures; a page of text may be substituted for a figure and vice versa. After informal review by the editors, they may be published within a few months of the date of receipt. Style requirements are the same as for regular contributions (see inside back cover).

Neuromorphic Approach to Inverse Problems in Aerodynamics

R. K. Prasanth*
Purdue University, West Lafayette, Indiana 47906
and
Kevin W. Whitaker†
University of Alabama, Tuscaloosa, Alabama 35487

Introduction

COMPUTATIONAL paradigms based on the structure of a brain, such as artificial neural networks, and their application to real world problems have rapidly flourished in recent years. The present paper explores the possibility of solving inverse design problems in aerodynamics using neural networks. We will begin with a brief overview of neural networks and then present two examples, each of which is representative of a different class of problems encountered in engineering aerodynamics.

Time-invariant problems, such as those to be considered in this paper, can be solved using static neuromorphic systems called feed-forward neural networks. A neural network is based on the concept of the adaptive linear element first studied in detail by Widrow and Angell[1] in the early 1960s. Neural networks consist of nothing more than simple, highly interconnected processing elements called neurons. Neurons by themselves are not particularly interesting, but their interconnection creates a powerful device that can approximate arbitrary functions. Each input signal to a neuron is amplified or dampened by a weighting factor associated with the path from the signal source to the neuron. The neuron collects all of the weighted inputs and sums them to form a total weighted input. This weighted input is passed to an activation threshold function, typically the mathematical function called the sigmoid. If the weighted input exceeds a certain threshold, the neuron fires sending a signal along its output path to another neuron. Additional descriptions of neuron function can be found in Refs. 2 and 3.

Neurons within a network can be arranged in any number of ways. In fact, much research has been conducted into determining the orientation of neurons, neuron interconnection schemes, and the interconnection weights. The goal is always to develop an architecture that minimizes some measure of error between the network output and the function to be approximated. Unfortunately, parameters such as the number of neurons, the interconnection schemes, and the network learning-rate strategy elude a priori assignment and require something of a black art to determine them. Thus, most neural network research centers around finding these network parameters.[2] In practice, a neural network architecture is frequently assumed to reduce the complexity of the design process.

Neurons can be arranged into vertical stacks called layers and a network can have any number of layers. A typical arrangement can be seen in Fig. 1. The leftmost and the rightmost layers are called the input and output layers, respectively, and the layers between them are called hidden layers because they do not interact with the outside. These definitions are more a matter of convention than of necessity, and any neuron in any layer can be used as an input or output channel. The terminology feed-forward has its roots in the fact that with such a network configuration, neuronal connections do not run from right (output side) to left (input side) or laterally. In other words, each neuron in any layer receives inputs from all of the neurons in the previous layer and from no other neuron. This assumption does not diminish the approximation capability of the network, although a larger network might be needed to achieve a given degree of accuracy.[4,5]

Neural networks solve problems by adapting to the nature of the data they receive. This is done primarily through supervised training. In supervised training, a network is told whether its answer (output) to a given input is correct or, if incorrect, what the magnitude of the error is. If the network's answer is not correct, the network begins to adjust the weights along each and every neuronal interconnection to reduce the error between the predicted output and the desired output. The most popular supervised training method is called backpropagation which was initially described by Werbos[6] in 1974, and later by Rummelhart et al.[7] in 1987. In short, backpropagation dictates how error at the output layer is fed backwards through the network, adjusting the interconnection weights along the way. Through repeated applications of a set of training data and subsequent back propagation of error, the weights eventually converge to an optimal configuration that allows the neural network structure to identify patterns in any new data presented to it. (It is tacitly assumed that the new data lie within or near a domain spanned by the training data.)

Applications

In this section, two simple examples that demonstrate the ability of neural networks to solve inverse aerodynamic problems are presented. The first example is an airfoil design problem wherein the shape of an airfoil that gives rise to a specified pressure distribution is sought. This is an unconstrained optimization problem in the sense that it involves a search over a well-defined class of airfoils without additional constraints on that class. The second example presented is a minimum length supersonic nozzle design. In this case, the solution requires searching for a nozzle of minimum length over the set of all nozzle shapes that are constrained to satisfy specified exit conditions. Each example concludes with results from the neural network design verification.

It is important to note that although each example case required a separate neural network to be designed, trained, and validated, some features were common to all networks used. First, all training data were generated numerically. A neural network is not sensitive to how training data are obtained, and large amounts of numerically generated data were easier to obtain than experimental data. Second, the activation used in the input layers was affine but all other neurons, including those of the output layer had sigmoidal activation. This is a standard technique with a proven record of success.[2]

Airfoil Design Problem

This problem involved computing an airfoil geometry that generates a specified pressure coefficient distribution. Considered was two-dimensional ideal flow over NACA four-digit airfoils. Also, only the pressure coefficient distribution on the lower surface of the airfoil was considered, and it was assumed that the angle of attack

Received Oct. 4, 1993; revision received June 2, 1994; accepted for publication June 2, 1994. Copyright © 1994 by the American Institute of Aeronautics and Astronautics, Inc. All rights reserved.

*Graduate Student, School of Aeronautics and Astronautics. Student Member AIAA.

†Associate Professor, Department of Aerospace Engineering. Senior Member AIAA.

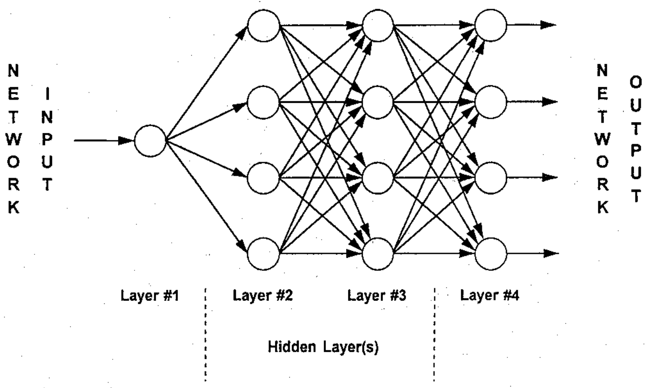

Fig. 1 Schematic of typical neural network architecture.

was fixed at 5 deg. These assumptions were made to simplify the problem and to aid our purpose of illustrating the power of neural networks. Larger neural networks can be used to solve this problem without the stated assumptions in much the same way as described subsequently.

The first step in the solution process was to determine the network architecture. Since the inverse problem solver must output a NACA four-digit airfoil, the neural network must have three neurons in the output layer. (Recall that the NACA four-digit code only contains three pieces of information.) The pressure coefficient distributions were described by polynomial expressions and, thus, the number of neurons in the input layer was fixed by the degree of the polynomial used. For this example, polynomials of degree 9 (i.e., 10 coefficients and thus 10 neurons) were used to described the pressure distributions. Finally, a hidden layer with three neurons was added to the network resulting in 16 total neurons with a total of 39 neuronal interconnection weights to be computed during the training process.

The objective of the training process, in this case, was for the network to learn to associate pressure coefficient distributions with airfoil geometries. Training data representing this mapping had to be obtained, and the easiest way to do this was to solve the forward problem. The forward problem has well-known solutions, and a traditional vortex-lattice method written by the authors was used. This phase involved randomly generating NACA four-digit airfoils and running the vortex-lattice code to compute the corresponding pressure coefficient distribution for an angle of attack of 5 deg. The pressure coefficient distributions were then approximated using polynomials of degree 9. Thus, the polynomial coefficients along with the three numbers (four digits) that described the NACA airfoil formed a training sample. This study used 200 such training samples and took 1 h to generate on a 486 PC running at 66 MHz.

The neural network was then trained through repeated presentation of the training data and subsequent adjustment of the neuronal interconnection weights using backpropagation. Convergence was determined by minimizing the square error between then values at each of the output neurons and what the four-digit airfoil code should actually be. This training process took approximately 4 h on an IBM RS/6000 workstation.

To verify the network, 40 pressure coefficient distributions described in terms of polynomial coefficients were randomly generated. These distributions were then used as inputs to the neural network, and the network computed the digits signifying a particular NACA four-digit airfoil. Illustrating the power of neural networks, the trained network was able to make the 40 requested predictions in less than 2 s of CPU time on an IBM RS/6000 workstation. Pressure coefficient distributions over the airfoils predicted by the neural network were then computed using the vortex-lattice code. The error between the specified pressure distribution and that resulting from the network predicted airfoil was computed using

$$e_{cp} = \frac{1}{50}\sum_{i=1}^{50}[Cp_{\text{spec}}(x_i) - Cp_{nn}(x_i)]^2 \qquad (1)$$

where $\{x_i\}_{i=1}^{50}$ are points along the surface of the airfoil, and $Cp(x_i)$ is the pressure coefficient at that location. A total of 50 points along the airfoil surface were considered. Figure 2 summarizes the results

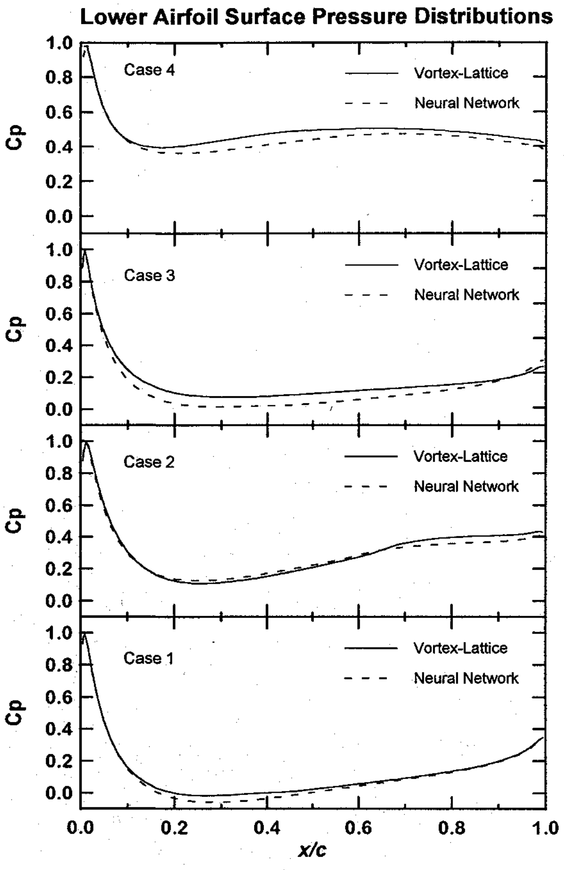

Fig. 2 Comparison of pressure coefficient distributions.

of this experiment by showing the assumed pressure coefficient distribution along with the distribution from the neural network predicted airfoil for four of the 40 test cases. The four cases shown are typical of the results obtained. Agreement was quite good in all cases with an average error of 5.1%.

Minimum Length Nozzle Problem

This problem here was to compute a nozzle contour of minimum length that would give rise to certain exit conditions uniform flow and a specified Mach number. It was assumed that the throat and exit areas were fixed with the throat Mach number equal to one. It was also assumed that the nozzles were symmetric and planar so that only one-half of each nozzle needed to be considered. To model the wall shapes in this study, it was decided (a purely arbitrary decision) to use fourth-order polynomials of the form

$$s(x) = s_{\text{ref}}(x) + \sum_{k=1}^{4} a_k x^k (L - x)^k \qquad 0 \le x \le L \qquad (2)$$

where S_{ref} is a baseline nozzle contour, which was assumed to be parabolic, and L is the length of the nozzle. Since this was a constrained optimization problem, the unknowns are the four polynomial coefficients and the minimum length. The neural network that solves for the minimum length nozzle (MLN) will clearly have the exit Mach number as input and the four polynomial coefficients that define the wall shape as outputs.

Two neural networks that jointly solved the MLN problem were designed. The first network received exit Mach number as input and gave the minimum nozzle length L^* as output. The second network took exit Mach number as an input and produced as outputs the coefficients of the normalized polynomial. The nozzle shape could then be computed from these outputs using

$$s(x) = \sum_{k=1}^{4} a_k^* x^k (L^* - x)^k \qquad 0 \le x \le L^* \qquad (3)$$

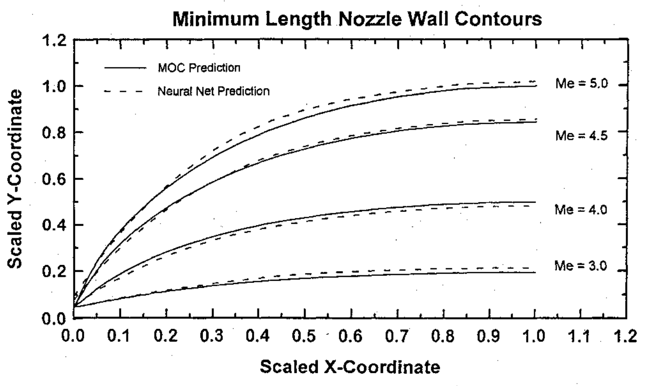

Fig. 3 Comparison of nozzle wall shapes.

Since the relation between exit Mach number and minimum length is a simple one-dimensional map, a polynomial approximation was used as the first network. The second network had a single neuron input layer, two hidden layers with four neurons each, and an output layer with four neurons.

The traditional MLN problem can be solved using method of characteristics (MOC) procedures and a program to that end was developed by the authors and used to generate training data. This phase involved randomly generating 200 exit Mach numbers in the range [1.05, 5]. The MOC program was then run for each Mach number and a fourth-order polynomial of the presented form was used to fit the shape output from the MOC program. The polynomial coefficients along with the corresponding exit Mach number and minimum length formed the training sample set. Network training was accomplished through repeated presentation of the training samples and backpropagation of error. Training both networks took 6 h on an IBM RS/6000 workstation.

The networks were then verified by randomly generating 40 Mach numbers in the range [1.05, 5] and running both the MOC code and the neural networks for each exit Mach number and obtaining the corresponding shapes. Errors were computed using

$$e_{\text{shape}} = \sum_{i=1}^{50} [s_{nn}(x_i) - s_{\text{moc}}(x_i)]^2 \tag{4}$$

where $\{x_i\}_{i=1}^{50}$ are points along the length of each nozzle. Typical results can be seen in Fig. 3. Good agreement was obtained between the neural network predicted shapes and the MOC predictions over the range of Mach numbers considered. The average error in shape was 3.9% and the average error in exit Mach number was never more than 3.1%. It must be noted that the 40 nozzle shapes were predicted by the neural networks in only 2.4 s of CPU time on an IBM RS/6000 workstation.

Conclusions

Experiments with two design problems have demonstrated the ability of neural networks to accurately and rapidly solve inverse aerodynamic problems. Once trained, a neural network is far superior computationally than any standard technique. In situations where inverse problems must be solved in real time, neural networks are certainly beneficial. Practical difficulties with this approach are associated with training data generation, sizing of the network, and choice of training algorithms. Problems more complex than those discussed here may require large amounts of computer time to solve the forward problem and, thus, it would be difficult to generate sufficient training data. Some complex problems may not have an analytic forward solution available and would need to resort to expensive experimentally generated training data. Robust neural networks designed with minimal training are needed and is an area suggested for further research.

Acknowledgments

We are grateful to Rob Markin, Karen Harwell, and Dedra Caddell, undergraduate students in the Department of Aerospace Engineering, for providing programming help on numerous occasions.

References

[1]Widrow, B., and Angell, J. B., "Reliable, Trainable Networks for Computing and Control," *Aerospace Engineering*, Vol. 21, 1962, pp. 78–123.

[2]Hecht-Nielsen, R., *Neurocomputing*, Addison–Wesley, Reading, MA, 1990.

[3]Rumelhart, D. E., and McClelland, J. L., *Parallel Distributed Processing*, MIT Press, Cambridge, MA, 1986.

[4]Hornik, M., Stinchcombe, M., and White, H., "Multilayer Feed-Forward Networks are Universal Approximators," *Neural Networks*, Vol. 2, 1989, pp. 359–366.

[5]Stinchcombe, M., and White, H., "Universal Approximation Using Feed-Forward Networks with Non-Sigmoidal Hidden Layer Activation Functions," *Proceedings of the IEEE—INS Conference on Neural Networks*, Vol. 1, 1989, pp. 185–191.

[6]Werbos, P., "Beyond Regression: New tools for Prediction and Analysis in the Behavioral Sciences," Ph.D. Dissertation, Committee on Applied Mathematics, Harvard Univ., Boston, MA, Nov. 1974.

[7]Rummelhart, D. E., Hinton, G. E., and Williams, R. J., "Learning Internal Representations by Error Propagation," *Nature*, Vol. 323, 1986, pp. 318–362.

Grid Combination Method for Hyperbolic Grid Solver in Regions with Enclosed Boundaries

Yih Nen Jeng* and Yi-Liang Shu†
National Cheng Kung University,
Tainan 70101, Taiwan, Republic of China

Introduction

THE hyperbolic equation method of grid generation[1–6] developed by Steger and Chaussee[2] is widely applied to external flow problems because it is very fast and can provide approximate grid orthogonality. However, grid oscillations frequently occur whenever the distribution of the Jacobian is not smooth[7], particularly in regions around a sharp convex or concave corner. In 1992, Chan and Steger[5] introduced several enhanced methods to eliminate these oscillations.

In Ref. 7, Tai et al. pointed out that some grid oscillations are due to the insufficient or excess damping of numerical equations proposed in Refs. 2 and 5. Subsequently, they employed the first-order upwind scheme[8] to approximate the grid equations. However, oscillation along the marching direction may still be present. Recently, a predictor-corrector method was proposed in Refs. 9 and 10, where the numerical procedure of Ref. 7 is considered as the predictor to recalculate the Jacobian. In Ref. 10, the restriction that the opposite boundary can not be specified in hyperbolic grid systems was partially alleviated by introducing a one-dimensional combination technique for the one-through type internal flow problems. This study employs the smoothing effect of the modified hyperbolic grid solver and modifies the combination technique developed in Refs. 10–12. Consequently, a hyperbolic grid solver for regions with prescribed grid points along all of the boundaries is developed.

Formulation

Steger and Chaussee[2] considered the following two equations

Received Aug. 12, 1994; revision received Dec. 6, 1994; accepted for publication Dec. 10, 1994; presented as Paper 95-0857 at the AIAA 33rd Aerospace Sciences Meeting, Reno, NV, Jan. 9–12, 1995. Copyright © 1995 by the American Institute of Aeronautics and Astronautics, Inc. All rights reserved.

*Associate Professor, Institute of Aeronautics and Astronautics. Member AIAA.

†Ph.D. Candidate, Institute of Aeronautics and Astronautics.