AIAA JOURNAL
Vol. 31, No. 2, February 1993

Specifying Exhaust Nozzle Contours with a Neural Network

Kevin W. Whitaker,* Ravi K. Prasanth,† and Robert E. Markin‡
University of Alabama, Tuscaloosa, Alabama 35487

Thrust vectoring is continuing to become an important issue in future military aircraft system designs. A recently developed concept of vectoring aircraft thrust makes use of flexible exhaust nozzles. Subtle modifications in the nozzle wall contours produce a nonuniform flowfield containing a complex pattern of shock and expansion waves. The end result, due to the asymmetric velocity and pressure distributions, is vectored thrust. Specification of the nozzle contours required for a desired thrust vector angle (an inverse design problem) has been achieved with genetic algorithms. However, this approach is computationally intensive, preventing nozzles from being designed on demand, which is necessary for an operational aircraft system. An investigation was conducted into using genetic algorithms to train a neural network in an attempt to obtain, in real time, two-dimensional nozzle contours. Results show that genetic-algorithm-trained neural networks provide a viable, time-efficient alternative for designing thrust vectoring nozzle contours. Thrust vector angles up to 20 deg were obtained within an average error of 0.0914 deg. The error surfaces encountered were highly degenerate and thus the robustness of genetic algorithms was well suited for minimizing global errors.

Introduction

FUTURE military aircraft will rely heavily on two- and three-dimensional thrust vectoring engines to boost their maneuverability and provide enhanced performance spanning their large operating envelopes. Current new technology engines use postexit vanes or large moveable surfaces to redirect engine exhaust to yield the desired thrust vectoring. Although this method has proven to be effective, penalties must be paid. For example, most thrust vectoring devices are heavy, primarily due to structural requirements involving the impinging exhaust flow. The devices must also be designed to withstand the extreme temperatures of the engine exhaust gases impinging on them. Control of the vectoring apparatus is complex and adds even more weight to the aircraft. Furthermore, the installation of typical thrust vectoring devices tends to mandate large clearance gaps to allow surface movement, and there is little opportunity for aerodynamic fairing. These and other factors can combine to yield higher overall drag forces on the aircraft.

A novel concept of vectoring engine thrust that addresses these concerns has been developed and shown to be viable.[1] The concept makes use of flexible nozzles where engine exhaust gases are turned not by some postexit apparatus but by *subtle* changes in the contour of the nozzle walls. The contour modifications produce a complex shock and expansion wave pattern in the nozzle flowfield and the end result is vectored engine thrust. Through judicious tailoring of the nozzle contour, a large range of thrust vector angles may be achieved. Theoretical pitch vectoring of ±20 deg has already been demonstrated with this concept. Full three-dimensional vectoring (pitch, yaw, and roll) is currently being investigated and could possibly eliminate the need for any tail control surfaces on future aircraft. This elimination would result in a tremendous savings in weight and drag, as well as a significant reduction in radar cross section.

Received April 21, 1992; presented as Paper 92-3328 at the AIAA/SAE/ASME/ASEE 28th Joint Propulsion Conference, Nashville, TN, July 6–8, 1992; revision received Aug. 17, 1992; accepted for publication Aug. 21, 1992.

*Assistant Professor, Department of Aerospace Engineering. Senior Member AIAA.

†Graduate Research Assistant, Department of Aerospace Engineering; currently, Graduate Student, Purdue University, West Lafayette, IN. Student Member AIAA.

‡Undergraduate Research Assistant, Department of Aerospace Engineering. Student Member AIAA.

This novel approach to thrust vectoring is based entirely on modifying the contour of the exhaust nozzle. For the technique to be useful in an operational aircraft system, the nozzle contour must be alterable on demand. Structural concerns aside, the challenge is to specify a nozzle contour for a pilot-requested thrust vector angle. This need suggests that modification of the nozzle contour would be tied to the flight control system of the aircraft. What is necessary for the success of such a thrust vectoring system is the real-time solution of an inverse design problem. (In this context, real time refers to the rate at which nozzle contours must be supplied to the vehicle's flight control system to effect attitude changes when requested by the pilot.)

Existing Jacobian-based methods for solving an inverse problem of this type are fraught with numerical difficulties and usually require an intense computational effort. A non-Jacobian method based on genetic algorithms can be used to compute the required nozzle contour for a requested thrust vector angle, as was proven in a recent study by King et al.[2] However, the specification of the nozzle contours still could not be accomplished in real time using genetic algorithms due to the computational requirements. Genetic algorithms, although they can routinely solve the inverse nozzle design problem (a definite advantage over many Jacobian-based methods), still require numerous flowfield evaluations to do so.

The hypothesis of the work presented here was that the inverse design problem could be solved on demand if a non-Jacobian-based method (genetic algorithms) was coupled with a neural network. Neural networks are biologically inspired computing systems with the phenomenal ability to grasp topological invariances that underlie inverse transformations. The power of a neural network stems from its ability to recognize patterns through experience (called training), rather than by deduction or recollection of rules. In fact, neural networks are often referred to as adaptive pattern recognition devices, but because of their flexibility and ability to generalize, their applicability typically extends well outside the established realm of simple pattern recognition. Thus, a neural network has the potential to be trained by a genetic algorithm and then, after sufficient training, to quickly solve the inverse nozzle design problem.

It is important to note that there is no intention to dismiss Jacobian methods; in fact the coupling of a Jacobian method with a neural network to design exhaust nozzles is currently under investigation by the authors. The work presented here, however, demonstrates that by using genetic algorithms, neural networks can be designed to provide an alternative with remarkable dexterity and computational ease for the real-time specification of thrust vectoring exhaust nozzles.

Genetic Algorithm Overview

Genetic algorithms are increasing in popularity as a search and optimization technique but are still unknown to a large portion of the scientific community; thus a brief description is in order. Genetic algorithms (GAs) are search algorithms based on the mechanics of genetics, using operations found in natural genetics to guide their trek through a search space. Their main strength lies in their ability to perform efficiently across a broad spectrum of search problems, including problems that are large, noisy, and poorly behaved. Two empirical investigations in the early 1970s demonstrated the technique's efficiency in function optimization.[3,4] Subsequent application of GAs to the search problems of pipeline engineering, microchip layout, structural optimization, job-shop scheduling, medical image processing, propulsion system component design, and machine learning adds considerable evidence to the claim that GAs are broadly based and robust.

GAs consider many points in a search space simultaneously and, therefore, have a reduced chance of converging to a local optimum. In most conventional search techniques, a single point is considered based on some decision rule. These methods can be dangerous in multimodal (many peaked) search spaces because they can converge to local optima. However, GAs generate entire populations of points, test each point independently, and then combine qualities from existing points to form a new population containing improved points. Aside from conducting a more global search, the GA's simultaneous consideration of many points makes it highly adaptable to parallel processors, because the evaluation of each point is an independent process.

GAs require the natural parameter set of the problem to be coded as a finite length string of characters. This requirement is actually true of all operations performed on a computer at the machine level, however the GA requires this coding on the *local* level. The user must represent possible solutions to the search problem as character strings. This task may at first seem imposing, but there have been a number of techniques developed for coding solutions to search problems.[5] Since GAs work directly with a coding of the parameter set and not the parameters themselves, they are difficult to fool because they are not dependent on continuity of the parameter space. A GA only requires information concerning the quality of the solution produced by each parameter set. This requirement differs from many optimization methods which require derivative information or, worse yet, complete knowledge of the problem structure and parameters. Because GAs do not require such problem-specific information, they are more flexible than most search methods.

It is important to note that GAs differ from a number of search and optimization techniques in that they use random choice to guide their search. Although chance is used to define their decision rules, GAs are by no means "random walks" through the search space. They use random choice efficiently in their exploitation of prior knowledge to locate optimal solutions rapidly.

Neuromorphic Approaches to Inverse Problems

Before presenting the results of this study, it is necessary to discuss the justification for solving an inverse problem using a non-Jacobian, genetic-algorithm-trained neural network approach. Of fundamental importance in solving inverse problems is the classic Stone-Weierstrass theorem.[6] Using the Stone-Weierstrass theorem it can be shown that under certain conditions nonlinear operators, such as the one encountered in fluid flow problems, can be represented using the well-known Volterra and Wiener series, thereby allowing computation of an approximate solution to the inverse problem. The theoretical works of Volterra, Wiener, and Urysohn (see Ref. 7) on the characterization and approximation of nonlinear operators find their full expression in neuromorphic approaches to inverse problem solving.

Let f and θ be Lebesgue integrable functions representing the spatio-temporal evolution of nozzle geometry and temporal evolution of thrust vector angle, respectively. The complex cause-effect structure that relates nozzle geometry and thrust vector angle can be written as

$$\theta = \boldsymbol{T}(f) \tag{1}$$

where $\boldsymbol{T}:\boldsymbol{E}\rightarrow\boldsymbol{F}$ is a mapping between appropriately defined Banach spaces $\boldsymbol{E}$ and $\boldsymbol{F}$. The inverse problem is to determine the map $\boldsymbol{T}^{-1}:\boldsymbol{F}\rightarrow\boldsymbol{E}$ such that

$$f = \boldsymbol{T}^{-1}(\theta) \tag{2}$$

Except in certain cases of little practical interest, the precise nature of the operator $\boldsymbol{T}$ is usually not known. To solve the inverse problem, we must first characterize the class of Banach space operators to which $\boldsymbol{T}$ belongs. But even when $\boldsymbol{T}$ is known to belong to a certain class, $\boldsymbol{T}^{-1}$ may not exist as a unique map resulting in an infinite number of solutions to the inverse problem. Thus, traditional approaches to this inverse problem are necessarily iterative.

A neuromorphic approach is entirely different. The basic idea is to somehow capture the inverse functional relation between thrust vector angle and nozzle shape in the synaptic interconnections of a neural network. Let $\boldsymbol{S}:\boldsymbol{F}\rightarrow\boldsymbol{E}$ represent a neural network. The particular representation of $\boldsymbol{S}$ depends on a finite number of synaptic weights W, a finite number of interconnections i, and an activation function ρ, i.e., $\boldsymbol{S}=\boldsymbol{S}(W, i, \rho)$. The neural network that solves the inverse problem at hand is obtained by finding W, i, and ρ that minimizes

$$\| \theta - \boldsymbol{TS}\theta \| \tag{3}$$

where $\| \cdot \|$ is a suitably defined spatial norm and θ is a thrust vector angle belonging to the bounded interval $[a, b]$. Commonly used spatial norms are of the L^p type defined on the space $\boldsymbol{F}$ as illustrated by:

$$\begin{aligned} L^2 \text{ norm: } \|f\|_2 &= \int_a^b \|f(x)\|^2 \mathrm{d}x \\ L^\infty \text{ norm: } \|f\|_\infty &= ess\sup |f(x)| \end{aligned} \tag{4}$$

In some sense, we are trying to find a right inverse of the operator $\boldsymbol{T}$; however, the design problem can also be formulated as the computation of a left inverse of $\boldsymbol{T}$. Note also that, depending on the activation function used, the neural network $\boldsymbol{S}$ will have some *nice* properties.

Neural networks with certain types of activation functions can approximate any function of the L^p class. This property, called *universal approximation*, has been studied extensively. For example, Funahashi,[8] Hornik et al.,[9] and Stinchcombe and White[10] have all studied continuous activation functions. Cotter[11] evaluated various activation functions using the Stone-Weierstrass theorem. Of prime importance to the present study, Cybenko[12] proves that multilayered neural networks with sigmoidal activation functions can approximate universally. The universal approximation capability of single-layer neural networks using integrable activation functions has been studied by Park and Sandberg.[13]

Although the results cited assure that, in principle, neural networks can be used to solve inverse problems, the actual design of a network is somewhat different. Because of the real-time requirements of the present study, we can only afford a finite number of interconnections and weights and, in most cases, an a priori estimate of the effectiveness of a finite neural network design is not available. The standard procedure is to choose an activation function with the universal approximation property (there are uncountably many functions) and to assume feed-forward full interconnection (each node in a layer receives inputs from all of the nodes in the previous layer). The number of layers and the number of nodes per layer are then chosen, based on experience, to provide a balance between accuracy, amount of training, and complexity of the network. Obviously, larger networks give better approximations, but more computations are needed for off-line training and on-line operation. These and other issues related to the practical design of neural networks are dealt with in recent studies.[14–16]

To further illustrate the advantages of a non-Jacobian method, consider a Jacobian-based solution technique to our simple problem involving no unsteady effects. The goal is to find a static nozzle geometry f so as to minimize

$$J = (\theta - \theta^*)^2 \tag{5}$$

subject to

$$Tf = \theta \tag{6}$$

where J represents the difference between the calculated and desired thrust vector angles. Under certain assumptions on $\boldsymbol{T}$, variational calculus provides the necessary conditions for computing an optimal nozzle contour. In general, a numerical solution can then be found iteratively from

$$f_{\text{new}} = f_{\text{old}} + K\nabla J \tag{7}$$

where K is a gain and ∇J is the gradient of J evaluated at f_{old}. However, the disadvantages of this solution are:

1) Every time a thrust vector angle is demanded, the flow equations must be solved at every iteration until convergence to evaluate the gradient. This requires exceptional computing power for real-time applications.

2) The cost surface is highly degenerate and has a multitude of troughs. There is no guarantee that the iteration will converge to an acceptable solution.

3) Perhaps the most important limitation is that the optimal solution depends on the particular assumptions made regarding nozzle flow. The operator $\boldsymbol{T}$ that describes nozzle flow must be known explicitly for any numerical implementation. Thus, experimental data cannot be used.

Consequently, the use of genetic algorithms for neural network design is justified. As alluded to previously, this justification does not mean that genetic algorithms are the only viable network design tool. For example, designing a network using a Jacobian-based back-propagation method is possible, but its robustness may be questionable.

Neural Network Design

Designing a neural network for nozzle contour specification involves two phases: a supervised training phase and a verification phase. Supervised training entails embedding the topological invariances in the synaptic weight space through repeated presentations of training samples that characterize the relationship between nozzle contour and thrust vector angle. Although a single network with a large number of synaptic interconnections can be designed to span a wide range of in-flight thrust vector angle requirements, it is not ideally suited for real-time applications. Instead, designing several small neural networks with fewer real-time computations, each for a specified overlapping range of thrust vector angle, is more appropriate. Outputs from two neural networks that span the overlap containing the demanded thrust vector angle can then be linearly interpolated to provide nozzle shapes. In addition to maintaining network simplicity, this approach has the significant advantage of allowing the two neural networks to be run in parallel, thereby reducing computational requirements. We will provide additional motivation for using smaller networks shortly.

Supervised training, or off-line training, is actually an optimization problem. To see this, recall that our aim is to design a network that takes as an input a thrust vector angle and gives as an output a nozzle shape that deflects the flow sufficiently to produce the demanded thrust vector angle. Since the nozzle shape is a function

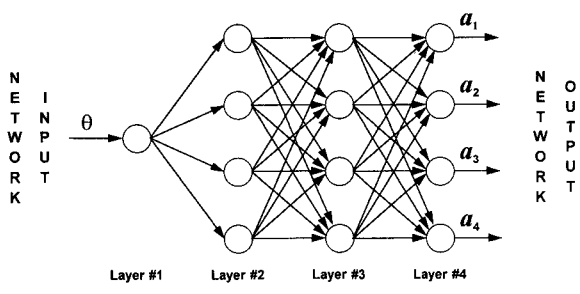

Fig. 1 Schematic of the neural network used.

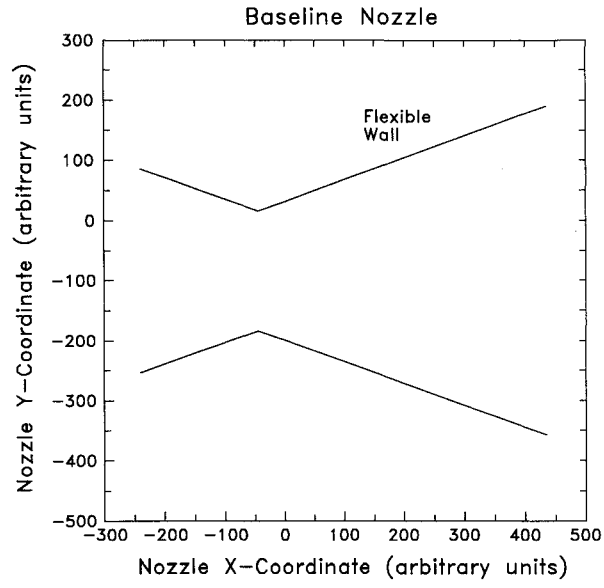

Fig. 2 Baseline nozzle geometry.

of spatial variables, we parameterize it so that the neural network can output these parameters instead of the shape. Accordingly, the input to the neural network is the desired thrust vector angle θ and outputs from the network are four polynomial coefficients $\{a_i, i = 1, 4\}$ that define the contour of the nozzle's upper wall as

$$f(x) = \sum_{i=1}^{4} a_i (x-x_0)^i (x-x_f)^i + g(x) \tag{8}$$

where x_0 and x_f are x coordinates of the fixed ends of the baseline geometry $g(x)$. The neural network outputs thus define an incremental geometry referenced about the baseline.

The number of nodes in the input and output layer of the network are obviously fixed by the problem at hand. The remaining network design parameters are chosen based on prior experience. Thus, the fully interconnected feed-forward neural network topology selected for use in this study consists of the sigmoidal function ρ, which satisfies universality,[12] a single-node input layer, a four-node output layer, and two hidden layers, each with four nodes. A schematic representation of the network can be seen in Fig. 1. When operating, the output from the kth node ($1 \leq k \leq 4$) in the (i + 1)th layer ($1 \leq i \leq 3$) is given by

$$\rho\left(\sum_{j=1}^{4} W_{ij,k} u_{ij}\right) \tag{9}$$

where u_{ij} is the output from the jth node in the ith layer and $W_{ij,k}$ is the weight for the interconnection from the jth node in the ith layer to the kth node in the (i+1)th layer. Note that the single node in the input layer has linear output elements, whereas all other nodes, including those of the output layer, have a sigmoidal activation function. Therefore, there are 36 (4+16+16) total weights that need to be selected to finish the network design. These weights are computed during training by minimizing the square (L^2) error

$$\text{error} = \sum_{i=1}^{N} \sum_{j=1}^{4} [S_j(\theta_i) - a_{ji}]^2 \tag{10}$$

where $\{\theta_i, a_{1i}, a_{2i}, a_{3i}, a_{4i}\}_{c=1}^{N}$ is a set of N training samples consisting of thrust vector angles θ_i and the corresponding nozzle shape coefficients a_{1i}, a_{2i}, a_{3i}, and a_{4i}. $S_j(\theta)$ is the output from the jth node of the output layer when θ is used as an input to the neural network.

In an actual design, only a finite number of training samples can be used. This raises the question of generalizability. Generalizability refers to the neural network's ability to output accurate nozzle shapes for thrust vector angles that were not present in the training

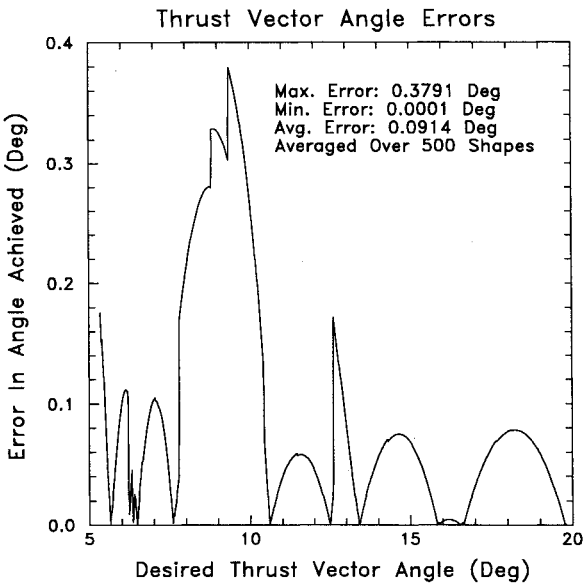

Fig. 3 Error in network specified angles.

set. A neural network that lacks this ability is no better than a look-up table. In going from small thrust vector angles to large angles, the flowfield properties change significantly. And to encode all of the complexities of a wide range of angle-shape relations, a large neural network is needed. On the other hand, in any small range of thrust vector angles flowfield properties do not change much and hence smaller networks can generalize. Moreover, most of a network's on-line computing consists of evaluating the nonlinear activation function present in each node. Thus, using small networks in parallel is better suited for the present study, where thrust vector angles are demanded at a high rate. Although the total time to train all of the small networks may not be lower than that for a single network, the small networks can be trained and even replaced independently.

The baseline nozzle developed for use in this study is shown in Fig. 2. The nozzle type selected was a symmetric, dual expansion-ramp nozzle. Concerns factored into the design were minimum length (to minimize weight) and reduced line of sight into the engine hot section to address observable characteristics. The baseline geometry was obtained after a number of iterations to insure the best performing nozzle was being used as a reference. The thrust vector angle of the baseline is 0 deg with a gross thrust coefficient of 0.983. It was assumed that the on-design conditions for the nozzle would be a nozzle pressure ratio of 10, a flight Mach number of 1.5, and a fluid-specific heat ratio of 1.15. Being essentially a proof-of-concept, this study was also restricted to a two-dimensional (planar) nozzle. A thrust vector angle range of 5–20 deg was selected, based on results with nozzles previously obtained by King et al.[2] (In this study, a positive thrust vector angle corresponds to a vehicle nose-up pitching moment.) Also, only the upper nozzle wall was selected for modification to further simplify this initial analysis.

The weight optimization was accomplished using codes developed for this investigation by the authors. Because the governing flowfield equations solve the forward problem, it is not possible to prescribe a thrust vector angle and find the true nozzle shape using them. Instead we must randomly choose a large number of nozzle shapes (i.e., the four polynomial coefficients), solve the flow equations, and compute the true thrust vector angles. The flowfield solver used in this study was a code based on the inverse method of characteristics.[17] This code, developed at the University of Alabama, allows for the analysis of supersonic flowfields internal to a nozzle as well as the supersonic exhaust plume. The code has been extensively validated with experimental data from NASA and industry. To initiate a solution, the code requires the nozzle on-design conditions and an initial value line located slightly downstream of the nozzle throat. The initial value line for this study used a Mach number distribution of 1.05 with all flow starting in the streamwise direction. Although a numerical code was used in this investigation, it is important to realize that the network design procedure does not depend on how the training samples are obtained and any method—numerical or experimental—can be used.

After randomly generating nozzle shapes and thrust vector angles, 500 training samples whose thrust angles ranged between 4.5 and 7.5 deg were chosen to span the interval [5 deg, 7 deg]. Similarly, seven other training sets, each with thrust vector angles in a 3-deg interval between 7.5 and 21.5 deg, were formed. Thus, a total of eight neural networks, each spanning a 2-deg interval in [5 deg, 20 deg], were designed by minimizing the L^2 performance index [Eq. (10)] using a genetic algorithm with a population size of 100. The set of weights displaying the minimum performance index over 25 generations was considered the optimal set of weights. Training all eight networks required 6 h on an IBM 3090.

Each of the eight neural networks can take any thrust vector angle as input and give shape coefficient as outputs. But the shapes predicted are more accurate in the interval for which a network was designed. It was observed that each neural network had a thrust vector angle in its range where the error became virtually zero. For example, the neural network that spans [9 deg, 11 deg] achieves this at 10.6 deg (see Fig. 3). That this center is not the midpoint of the interval is due to the nonuniformity in the training set. This observation also allows us to interpolate between networks to get better nozzle shapes. If we denote these centers by $\{\gamma_i\}_{i=1}^{8}$ then for any force angle $\theta \in [5 \text{ deg}, 20 \text{ deg}]$, the shape coefficients $\{a_i\}_{i=1}^{4}$ are obtained as follows:

1) Determine k such that $\gamma_k \leq \theta \leq \gamma_{k+1}$.

2) Input θ to the kth and $(k+1)$th neural networks and obtain $\{a_{1i}\}_{i=1}^{4}$ and $\{a_{2i}\}_{i=1}^{4}$ as the respective outputs.

3) Compute $\{a_i\}_{i=1}^{4}$ using:

$$a_i = a_{1i} + \left[\frac{\theta - \gamma_k}{\gamma_{k+1} - \gamma_k}\right](a_{2i} - a_{1i}) \tag{11}$$

It should be noted that in an operational version of the complete neural network, inputs would be a function of time, whereas during the training phase, constant values of thrust vector angles constituted the training samples. This fact brings about a significant advantage of transforming what, in general, is a dynamic optimization problem to a static network design problem. However, it is valid only on neglecting unsteady fluid flow effects caused by dynamic changes in nozzle contour. When the unsteady flow

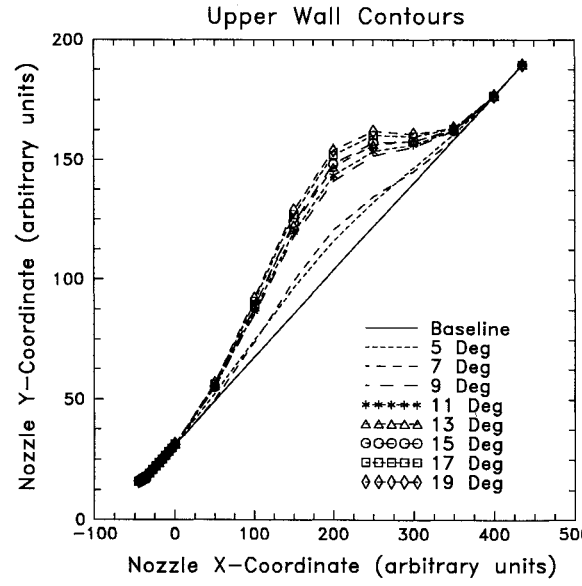

Fig. 4 Representative nozzle contours specified by the neural network.

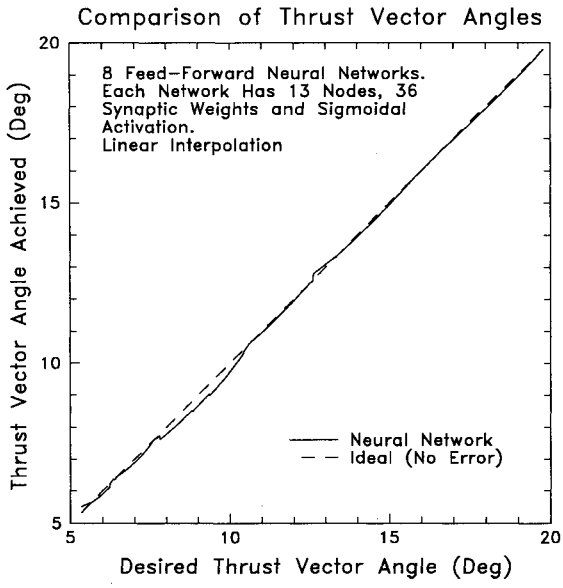

Fig. 5 Comparison of requested angles with angles achieved.

effects neglected are of importance, a recurrent (or dynamic) neural network is appropriate.

Neural Network Verification

A Monte Carlo simulation was performed to verify the neural network design. (From this point on, neural network refers to the eight small networks operating in parallel.) A total of 500 thrust vector angles between 5 and 20 deg were randomly generated. Then each thrust vector angle was presented to the neural network as an input. The four polynomial coefficients obtained as outputs from the neural network for each requested thrust vector angle were used to define a nozzle contour. The neural network could generate 16 nozzle shapes/s on an IBM-compatible PC with an 80486 microprocessor running at 25 MHz. Representative samples of nozzle contours obtained can be seen in Fig. 4.

To determine how well the neural network specified the nozzle contour, the method of characteristics code was run to find the actual thrust vector angle for each of the neural network specified contours. Figure 5 compares the requested thrust vector angle with the thrust vector angle obtained from the method of characteristics code. Figure 3 shows the error in the network-predicted thrust vector angles. The performance of each of the eight small networks used also can be clearly seen in Fig. 3. Thrust vector angles of up to 20 deg were obtained within an average error of 0.0914 deg by affecting modifications to the upper nozzle wall only. Modifying both upper and lower walls would cause a very complex flow structure and could possibly expand the vectoring angle envelope. The maximum error in the thrust vector angle was 0.3791 deg, which would be negligible in an operational aircraft system. It is thought that further improvements in vectoring performance could be expected to occur by using an L^{∞}-type performance index and running the genetic algorithm in the training phase with an increased number of generations.

Conclusions

It has been shown that a genetic-algorithm-trained neural network provides a viable alternative to straight Jacobian-based solution methods for inverse problems. The neural network developed in this study requires very few on-line computations while maintaining accuracy and retaining design simplicity. In addition, although genetic algorithms may not be ideal for solving the inverse problem of thrust vectoring directly, their utility is demonstrated by their ability to train a neural network to do so.

The procedure presented here for designing neural networks and the subsequent design of thrust vectoring nozzles has advantages and disadvantages. Error surfaces encountered while designing thrust vectoring nozzles are highly degenerate, and, therefore, a robust optimization scheme such as a genetic algorithm is required for global error minimization. But in problems of high dimensionality, there are numerical difficulties in using genetic algorithms, limiting the complexity of the simulated function and the size of the network that can be trained. Jacobian-based back propagation, for example, may reduce the training period significantly and would have no difficulty handling large dimensions. However, as with other gradient techniques, back propagation is prone to converge to a local minimum, thereby converging to an incorrect network design or not converging at all. Further study is recommended to put these concerns to rest.

Finally, there are two competing aspects to neural network design—accuracy and generalizability—which need to be addressed. Accuracy has to do with how close the approximation obtained using the neural network is. Generalizability means that a neural network can interpolate and extrapolate beyond problem instances spanned by the training set. The performance index used in this study does not reflect generalizability. It would be of considerable interest to develop performance indices that provide a balance between the two aspects and then redesign *adaptive* neural networks for thrust vectoring. In this study an off-line design method, wherein the entire training set is presented to the neural network at one time, was used; an on-line or adaptive neural network capable of learning while in operation would be better suited for practical applications.

References

[1]Whitaker, K. W., Gowadia, N. S., and Fordyce, S. C., "Thrust Vectoring Using Non-Axisymmetric Nozzles With Flexible Contours," AIAA Paper 91-2368, June 1991.

[2]King, E. G., Freeman, L. M., Whitaker, K. W., and Karr, C. L., "Two-Dimensional Thrust Vectoring Nozzle Optimization Techniques," AIAA Paper 91-0473, Jan. 1991.

[3]Holland, J. H., "Genetic Algorithms and the Optimal Allocations of Trials," *SIAM Journal of Computing*, Vol. 2, No. 2, 1973, p. 88–94.

[4]Foo, N. Y., and Bosworth, J. L., "Algebraic, Geometric, and Stochastic Aspects of Genetic Operators," NASA CR-2099, Aug. 1972.

[5]Goldberg, D. E., *Genetic Algorithms in Search, Optimization, and Machine Learning*, Addison-Wesley, Reading, MA, 1989, pp. 80–84.

[6]Zimmer, R. J., "Essential Results of Functional Analysis," *Chicago Lectures In Mathematics*," Univ. of Chicago Press, Chicago, 1990.

[7]Rugh, W. J., *Non-Linear System Theory: The Volterra-Wiener Approach*, Johns Hopkins Univ. Press, Baltimore, MD, 1981.

[8]Funahashi, K., "On the Approximate Realization of Continuous Mappings by Neural Networks," *Neural Networks*, Vol. 2, No. 3, 1989, pp. 183–192.

[9]Hornik, M., Stinchcombe, M., and White, H., "Multilayer Feed-Forward Networks are Universal Approximators," *Neural Networks*, Vol. 2, No. 4, 1989, pp. 359–366.

[10]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, Washington, DC, 1989, pp. 185–191.

[11]Cotter, N. E., "The Stone-Weierstrass Theorem and its Application to Neural Networks, " *IEEE Transactions on Neural Networks*, Vol. 1, Dec. 1990, pp. 290–295.

[12]Cybenko, G., "Approximation by Superposition of Sigmoidal Functions," *Mathematical Control, Signals and Systems*, Vol. 2, No. 4, 1989, pp. 303–314.

[13]Park, J., and Sandberg, I. W., "Universal Approximation Using Radial Basis Function Networks," *Neural Computation*, Vol. 3, No. 2, 1991, pp. 246–257.

[14]Moody, J., and Darken, C. J., "Fast Learning in Networks of Locally Tuned Processing Units," *Neural Computation*, Vol. 1, No. 2, 1989, pp. 281–294.

[15]Jones, R. D., et al., "Nonlinear Adaptive Networks: A Little Theory, A Few Applications," Los Alamos National Lab. Rept. LA-UR-91-273, Los Alamos, NM, 1991.

[16]Jones, R. D., Lee, Y. C., Barnes, C. W., Flake, G. W., Lee, K., Lewis, P. S., and Qian, S., "Function Approximation and Time Series Prediction with Neural Networks," Los Alamos National Lab. Rept. LA-UR-90-21, Los Alamos, NM, 1990.

[17]Whitaker, K. W., and Cates, J. E., "A User-Friendly Exhaust Nozzle Design Program Based on the Method of Characteristics," AIAA Paper 90-2029, July 1990.