A NEURAL NETWORK AERO DESIGN SYSTEM FOR ADVANCED TURBO-ENGINES

Jose M. Sanz
National Aeronautics and Space Administration
Glenn Research Center
Lewis Field
Cleveland, Ohio 44135

Abstract

An inverse design method calculates the blade shape that produces a prescribed input pressure distribution. By controlling this input pressure distribution the aerodynamic design objectives can easily be met. Because of the intrinsic relationship between pressure distribution and airfoil physical properties, a neural network can be trained to choose the optimal pressure distribution that would meet a set of physical requirements. The neural network technique works well not only as an interpolating device but also as an extrapolating device to achieve blade designs from a given database. Two validating test cases are discussed.

Introduction

Neural network systems have been attempted in the context of direct design methods of turbomachinery blading, References 1 and 2. From flow properties ascribed to a set of blades, the neural network is trained to infer the properties of an 'interpolated' blade shape. The difficulty with this approach is that in transonic regimes where we deal with intrinsically non-linear and ill-posed problems, small perturbations of the blade shape can produce very large variations of the flow parameters, i.e. the actual properties of the 'interpolated blade' can be very different from the intended. It is very unlikely that, under these circumstances, a neural network will be able to find the proper solution.

The unique situation in the present method is that the neural network can be trained to create the required input pressure distribution from a database of pressure distributions. An inverse method, Reference 3 and references therein, will compute the exact blade shape that corresponds to this 'interpolated' input pressure distribution. In other words, the interpolation process is transferred to a smoother problem, namely, finding a pressure distribution that would produce the required flow conditions. Once this is done, the inverse method finds the exact flow solution and corresponding blade shape.

The neural net acts as a versatile interpolation procedure. Once a database of pressure distribution and corresponding blade shape has been constructed, the neural net is trained according to certain rules. Each element (neuron) of the neural net is an operator ascribed with a weight and a bias. The training procedure consists of determining those weights and biases. Essentially, the process determines the relative weight for each element and how each weighted element influences and biases the system. A multilayered network is used to train the system for pattern association and classification. The training of the neural net system is unique to a given database, i.e., the system has to be retrained whenever the database is modified.

The implementation of the neural network is better achieved by adopting an object oriented paradigm that allows a more rational communication between the different modules involved in the design process. Under this paradigm, objects (blades) have properties assigned on which the neural network can act. The object oriented JAVA language has proven to be an excellent tool to write the neural net modules. Fortran modules are being embedded in a program interface that allows the execution on a central computer server and a graphical interface from any remote client.

The resulting system is a design procedure capable of a fast and automated way of producing the required blade designs from a given numerical database. The neural network is not only very efficient as an interpolating tool but seems to be capable of extrapolating results beyond the given database.

In this paper, the reader is led through the concept, training, use and testing of neural network methodology. The methodology is validated by analyzing two test cases.

Impact on the design cycle

The use of neural networks is, in this context, highly related to the use of proper optimization techniques. Optimization is used essentially as an automation procedure to force the input pressure distributions to achieve the required aero and structural design parameters. The automation and versatility of the procedure enables the design of an optimal mutistage aero configuration at preliminary design time by incorporating the actual blade geometries, rather than using a 'no blade' model, from the start of the design cycle.

In a typical design cycle, see Figure 1, the meanline and blade geometry calculations are reached at two quite different steps, and most commonly by two different groups of designers. The stream annulus shape, together with the hub to tip flow conditions, is the output of the meanline design in which the blade geometry designer is required to match the prescribed flow conditions. On the other hand, this stream annulus was determined with the assumption, backed by the designer database, that a blade geometry will fit in. Commonly, this stream annulus shape has been 'cast' by the time the blade designer receives it and very rarely will be modified. Even in very advanced designs, it is possible to find cases where an extra row of vanes has been included to resolve the peculiarities of a predetermined stream annulus geometry. The need for this row of vanes could have been eliminated with a more synergistic design process.

The neural net system will attempt to integrate these two steps into a single procedure which will optimize the stream-annulus/blade-shape design process. The inverse design code can produce, with the present computational capabilities, a blade cross section in one second of CPU time for a given pressure distribution. This blade shape candidate will guarantee shock free, non separated flow at the design point and will supply the design point loss. This information is supplied back to the meanline code to optimize the stream tube shape. The neural network chooses at each step the proper pressure distribution to feed into the inverse design code.

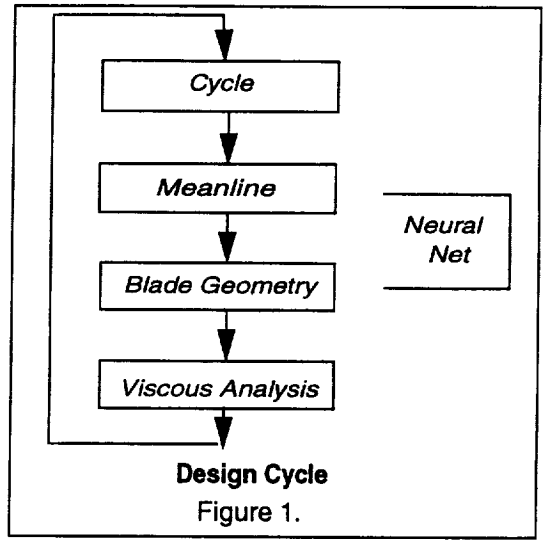


Figure 1.

A static deflection analysis and a natural frequency vibration analysis are used to thin out aerodynamic shapes that cannot be accepted from a structural point of view and to serve as a first iteration of the multidisciplinary design cycle.

The appealing natural extension of this work is to merge cycle and viscous analyses into the present process. A powerful cluster of today's workstations makes possible the timely production of an off-design blade map that can be fed back into the design cycle. The hurdle to overcome is not CPU time, but synergy and format compatibility.

The neural network concept

A neural network, as an emulation of its biological counterpart, consists of a multilayered array of simple computational elements called neurons. A neuron receives a set of *input* values, multiplies each by a *weight* factor and adds a *bias* to form a *net* input which is supplied to a *transfer* operator yielding an output value. If **p** and **w** are r-dimensional input and weight vectors, respectively, and b is a scalar bias, the transfer operator F produces a scalar output:

$$a = F(\boldsymbol{w} \cdot \boldsymbol{p} + b)$$

The multiple inputs are weighted in a similar manner to the way the dendrites condition the different electrochemical inputs in a biological neu-

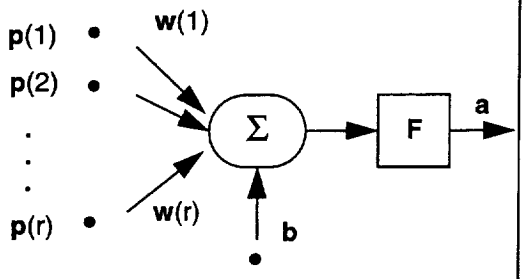
ron. The bias can be considered a weight for a constant input of 1. The transfer function transforms the net input $n = w \cdot p + b$ into the output element, as the axon does in its biological equivalent.

The transfer function most commonly used is the Heaviside (step) function. A fixed response is obtained if the net input surpasses a given threshold. A neuron with this transfer function is called a perceptron. It very much resembles the dynamics of the biological neuron, and is useful for data classification.

The Heaviside function is not differentiable but can be replaced by a differentiable function with similar properties. The sigmoid function

$$\sigma = 1/(1+e^{-x})$$

can be used to produce similar results after a transient, differentiable, period. Other transfer functions commonly used are the arctanh, trigonometric functions in general, a Gauss (error) function as well as linear functions. Figure 2 represents a schematic of a neuron.



Schematic of a Neuron

Figure 2.

An r-input neuron is uniquely determined by r weights, one bias and a transfer function acting on the net input n and yielding one single output value.

Assume now that the same r-input vector is presented to a *layer* of s neurons with the same or different transfer functions. The relationship

$$p \rightarrow n = w \cdot p + b \rightarrow a = F(n)$$

establishes a new matrix equation, where **p** is an r-column vector, **w** is an (s,r) matrix and **b** and **a** are now s- column vectors. The neuron layer takes r input values and transforms them into s output values. It has r times s weights and s biases to be determined for s given transfer functions.The type of problem will determine the number of layers and transfer function. In the JAVA programing language, in which the neural net has been programed, transfer functions can be added very easily as new methods to the class of neurons.

Similarly the output s-dimensional vector of an s-neuron layer can be fed to a new s-input layer with, say, s2 neurons to produce an s2-dimensional output vector.

Although a single neuron can not do much, the combination of layers of neurons is a very powerful tool. The combination of a sigmoid function layer feeding into a linear layer can approximate almost any function with an arbitrary number of discontinuities and without showing a Gibbs (overshooting) phenomenon.

Once a neural net has been established by determining a sequence of layers and choosing the transfer functions and dimensions of each layer (r-inputs into s-neurons) the process of determining the weights and biases can start. This process is called the training of the neural net, and is unique to every database on which the net is going to act.

Training the neural network

Assume we a have a multilayered net with an r-dimensional input vector feeding the first layer, some intermediate layers with an arbitrary number of neurons always equal to the number of inputs for the next layer, and a final s-dimensional output vector. We would like to establish a one-to-one relationship between **q** *training* vectors of length r and **q** *target* vectors of length s. In other words, we want the neural network to associate, in a one to one manner, the properties of each one of the **q** r-input vectors to each of the **q** s-output vectors. If the neural net 'learns' that the target vectors contain the properties that we associate with the training inputs, a new input vector will be associated

with an output vector whose properties will follow a similar pattern.

The neural net can be seen, essentially, as an interpolation procedure which is carried out in a very versatile manner. In the elementary case of a single linear neuron, if we choose the number of training vectors q to be equal to r+1 (q constraints, r+1 degrees of freedom) the problem would be fully determined (provided that the vectors are linearly independent) and a linear solver would suffice as training.

To train a general neural network we adopt the following method, known as the Widrow-Hoff technique, Reference. 6. For a given set of weights **w** and biases **b,** let **p** be an input vector and **a = F(w.p+b)** its corresponding output vector. If **t** is the target vector of **p** for each of the **q** training vectors, we want to find the weights and biases that minimize the sum of the squared errors, for the error vector **e = t-a**

$$\|e\|^2 = \sum_{k=1}^{q} (t(k) - a(k))^2$$

A gradient steepest descent method can be used to solve this problem. By incorporating the biases into the weights matrix and augmenting the inputs **p** with a row of 1, the Lagrangian derivatives

$$\frac{\partial}{\partial w_{ij}}(\|e\|^2) = \frac{\partial}{\partial w_{ij}}(t_i - F^i(w_{ij}p_j))^2$$

$$= -2e_i \frac{dF^i}{dn} p_j$$

can be obtained for each weight.

The weights matrix can then be incremented in the opposite direction of the increasing errors

$$\Delta w = \lambda e\left(\frac{dF}{dn} \cdot transp(p)\right)$$

with λ being a positive adjustable parameter. A back propagation process is used for a multilayered network by finding first the weights of the last layer and working back towards the first layer. The method guarantees only local rather than global error minimum. The number of neurons in the intermediate layers is at our disposal and can be modified if the error achieved is not satisfactory.

A neural net for the inverse design system

The inverse design method solves the problem of finding a blade shape with a prescribed surface pressure distribution. For the problem to be uniquely determined, three additional constants related to the inlet Mach No., inlet air angle and cascade solidity have to be prescribed. These three constants together with a B-spline representation of the surface pressure distribution form the *core input* for the inverse method. An automated Newton algorithm acts on this *core input* to achieve the design objective vector

$$[M_1, \beta_1, \Delta\beta, \sigma, Th_e]$$

formed by the target inlet Mach number, inlet air angle, air flow turning, solidity and trailing edge thickness. The trailing edge thickness is included as part of the objective vector due to the large impact that it can have over the (finite) trailing edge speed.

An existing database of inverse design blades, comprising approximately one hundred test cases, contains the *core inputs* with their corresponding *objective vectors*, blade geometries and associate flow properties. The task at hand is to train the neural net using this database. Figure 3 is a sketch of the neural net flow process. The output of the neural net then provides a pressure distribution for the inverse design method. The inverse design method will then generate the blade geometry.

The coefficients of the B-spline representation for the pressure distribution contain a wealth of information about the flow characteristics of the associated blade. The distinguishing characteristic of the inverse method is that the variation of these coefficients acts almost linearly on the flow properties of the blade. For instance, controlling the B-spline coefficient that sets the slope of the pressure distribution at the stagnation point of a turbine blade, which directly relates to the leading edge curvature of the airfoil, can, almost linearly, modify

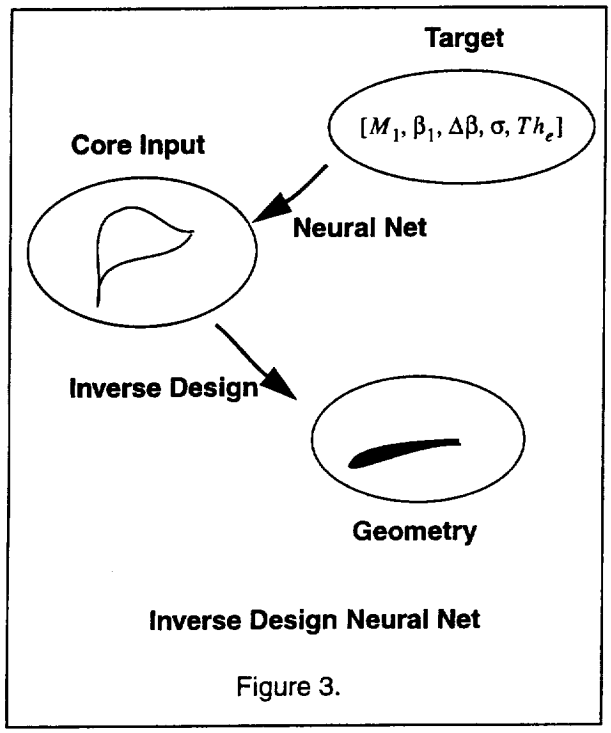


Figure 3.

the aft-loading characteristic of that blade. Each pressure distribution B-spline can, in most cases, be represented by six control points on each side with a fourth order spline. The size of the database file is less than 200 kilobytes.

The variation of a single parameter is seen to have a direct impact on the blade performance. We want the neural net to learn that this leading edge stagnation point slope coefficient has a given influence on the aft-loading of the blade, and compares it to the impact that its modification will have on the overall loading of the blade. The coefficients that control the overall loading (sustained area) will be influenced by the necessary modifications since one target to be met is the flow turning angle.

A neural net formed by a sigmoid layer and a linear layer has been trained to produce the core input data for a required objective vector. The resulting neural net, because of the good approximating properties of the combination sigmoid-linear layer, seems to be sufficient for the purpose of picking up proper input pressure distributions for the inverse design code. More layers will probably be necessary to address other types of problems related to overall performance of engine components in which more complex dependencies are to be explored.

Testing the neural net

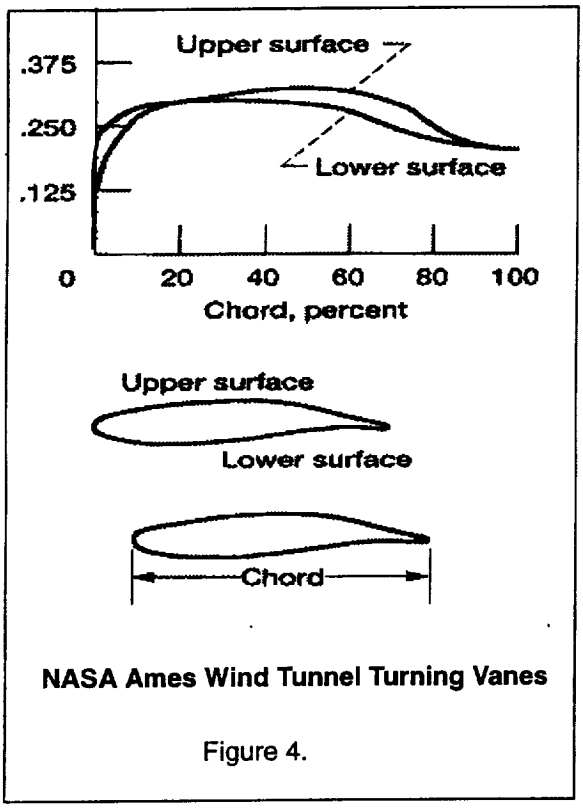
Two experiments have been carried out to test the neural net system. The first experiment was to produce new designs whose objective vectors fall fully within the design database envelope. The new designs conform naturally to the database. If the pressure distributions for the database elements surrounding the objective vector are of a 'flat top' type, the new pressure distribution inherits this property. Similarly, behavior at the leading edge is inherited from neighboring cases. In essence, the quality and characteristics of the new blade design are determined by the database characteristics.

The Icing Research Tunnel Heat Exchanger Modification Project required the design of new turning vanes at the corners No. 3 and 4 of this wind tunnel to accommodate the change in cross section area imposed by the new heat exchanger. The neural net was able to pick up an alternative solution to the standard one being proposed. The experimental results, Reference 7, confirm the quality of the design. Besides being far more efficient, which will largely improve the flow quality of the wind tunnel, these turning vanes have produced a substantial manufacturing saving because approximately only half the vanes are needed. Besides, due to their characteristic thickness, the new vanes have been manufactured out of fiber glass rather than steel. Operational and maintenance costs will be also lowered by incorporating this design.

Obviously this type of design could have been achieved without the neural net, but now the system finds the solution in an automated way, in a matter of minutes, rather than having to tailor a specific pressure distribution for that particular case. If the problem at hand is the design of a compressor stage including its stream tube, new blade shapes corresponding to changing flow conditions can be automatically generated, adding a whole new operational mode to the blade design process.

The inverse design database contains a number of cases that demonstrate the capability of the method to produce very unique designs. The second type of experiment was devised to see if, by suppressing some of the more unique data sets from the database and training the neural net without them, the system would be capable of reproducing those cases.

The first case, Figure 4 is a staggered cascade of turning vanes with near zero lift and a working range of inlet air angle of near 60 degrees, Reference 4.



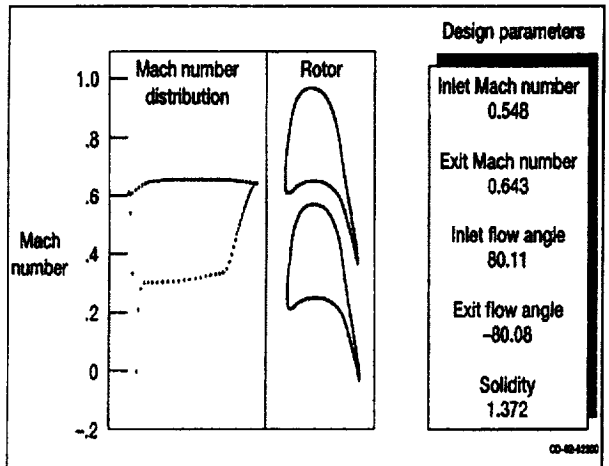
NASA Ames Wind Tunnel Turning Vanes

Figure 4.

The neural net system is confronted with finding an airfoil that is not symmetric, but has near zero lift. It generates a pressure distribution that inherits the needed properties from neighboring blades. The inverse design method finds an airfoil that is strikingly similar to the original design for the NASA Ames 40X80X120 wind tunnel that has been suppressed from the database. Because this was a two point design, the design inlet air angle was set to minus 16 degrees and the turning to nine degrees to accommodate the stagger angle at both operating conditions.

The second case of this experiment was to try to reproduce a turbine blade design with 160 degrees of flow turning originally designed for the Advanced Turbopump Turbine of the Space Shutle Main Engine, Reference 5. Such a degree of turning did not exist in the industry database at the time of the design, and as a matter of fact, the literature does not indicate that it has ever been reproduced. A turbine blade with a 100 degrees of turning was considered a very high turning blade.

The implication of this test case was to see if the neural net is capable of extrapolation as well as interpolation. By suppressing from the database this blade with 160 degrees of turning, the objective vector is now completely outside the database. No neighboring cases exist that can interpolate the needed conditions. Still the neural net finds that to meet the turning demands it has to flatten the pressure distribution to maximize the sustained area and again finds a solution which is very close to the original design. Figure 5, shows the original design of the Advanced Turbopump Turbine.



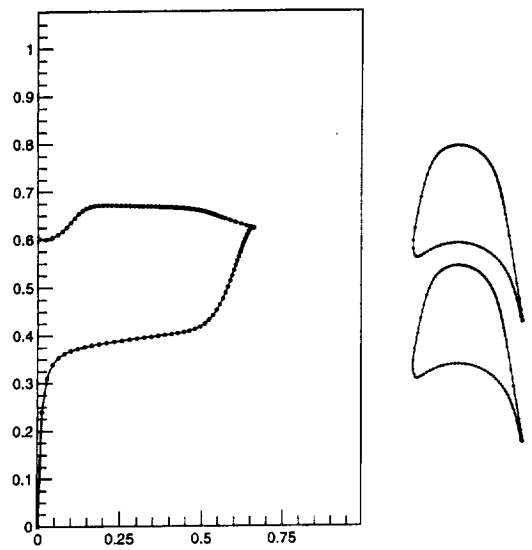
Advanced Turbopump Turbine

Figure 5.

This original design has a specific wedge angle requirement at the trailing edge. The neural net does not find the same wedge angle because this is an external requirement not included in the database and there are no neighboring cases in the database to influence the design. The input pressure distribution found by the neural net has to be manually changed and a few more iterations of the design code are necessary to match the original design. Figure 6 shows the design that the neural net finds before adjusting for the imposed wedge angle. The new pressure distribution is somewhat different to the original because of this extra condition, but both meet the design objective equally well.

Conclusion

A neural network has been designed and



Inlet Mach No. 0.549 *Pitch 0.742*
Inlet Air Ang. 80.02 *Axl Pitch 1.080*
Exit Mach No. 0.633 *Max Thck 0.751*
Exit Air Ang. -80.10 *T.E.Thck 0.010*
Turning 160.12 *D.T.E. 0.000*

Figure 6

implemented as a complementary tool for the design of turbomachinery blading. The initial experimentation with the system indicates that, indeed, the neural net can learn association and pattern behavior. The system offers to automate the blade design process to an extent that blade geometries, with their particular performance, can be incorporated very early in the design process. The experiments seem to indicate that the neural net is not only a good interpolating tool for the design within the current database but also a good extrapolating tool for the design outside this database.

References

1. M.M. Rai and N.K. Madavan"Application of Artificial Neural Networks to the design of Turbomachinery Airfoils. 36th Aerospace Sciences Meeting and Exhibit. Jan 12-15, 1998. Paper # AIAA 98-1003.

2. H. Lee, S. Goel, S.S. Tong, B. Gregory and S. Hunter."Towards Modeling the Concurrent Design of Aircraft Engine Turbines" ASME Paper 93-GT-1993.

3. J.M. Sanz. "Automated Design of Controlled Diffusion Blades". Journal of Turbomachinery, Vol. 110, October 1988. pp 540-544.

4.J.M. Sanz et al. "Design and Performance of a Fixed, Non-Accelerating, Guide Vanes Cascade that Operates Over an Inlet Flow angle Range of 60 Degrees." Journal for Gas Turbines and Power, Vol. 107 NO.2, April 1985, pp 247-254.

5. J.M. Sanz. "On the Impact of Inverse Design Methods to Enlarge the Aero Design Envelope for Advanced Turbo-Engines." Proceedings of the Sixth International Symposium on Computational Fluid Dynamics, Sept. 1995, Val III, pp. 1044-1051.

6.Introduction to the Theory of Neural Computation. J. Hertz, A. Krogh and R. Palmer. Addison-Wesley, 1995.

7. V. A. Canacci et al. "Flow quality Studies of the Scale Model Icing Research Tunnel and Projections to the Full-Scale Modified IRT." 37th Aerospace Sciences Meeting & Exhibit. AIAA-99-0307.