AIAA JOURNAL
Vol. 36, No. 11, November 1998

Air Data Sensing from Surface Pressure Measurements Using a Neural Network Method

Thomas J. Rohloff*
University of California, Los Angeles, Los Angeles, California 90095
Stephen A. Whitmore†
NASA Dryden Flight Research Center, Edwards, California 93523
and
Ivan Catton‡
University of California, Los Angeles, Los Angeles, California 90095

Neural networks have been successfully developed to estimate freestream static and dynamic pressures from an array of pressure measurements taken from ports located flush on the nose of an aircraft. Specific techniques were developed for extracting a proper set of neural network training patterns from an abundant archive of data. Additionally, the specific techniques used to train the neural networks for this project were reported, including the scheduled adjustments to learning rate parameters during the training process. The performance of the trained networks was compared to the accuracy of the aerodynamic model that is currently being applied to these flush air data sensing systems.

Nomenclature

dt = data sampling interval, s
f = neural network transfer function
$G(s)$ = filter transfer function
i_i = neural network input signal
i_{total} = weighted sum of inputs from the previous layer of the neural network
M = Mach number
o = neural network output
P_i = pressure measurement for port i of the flush air data sensing (FADS) instrumentation, psf
P_∞ = freestream static pressure, psf
q_c = freestream dynamic pressure, psf
w_i = neural network weight
$x(t)$ = measured value in a time series
$x_f(t)$ = expected value in a time series
α = momentum parameter for the training algorithm
α_∞ = angle of attack (pitch) of the aircraft relative to the freestream, deg
β_∞ = angle of side slip (yaw) of the aircraft relative to the freestream, deg
Δw = change in neural network weight
δ = neural network error signal
ϵ = filter threshold
η = learning rate parameter for the training algorithm
θ = neural network bias term
λ = cone angle of FADS pressure ports, deg
ξ = filter damping ratio
ϕ = clock angle of FADS pressure ports, deg
ω_n = filter natural frequency, rad/s

I. Introduction

ACCURATE measurements of air data parameters are important for both flight testing and control of aircraft. These parameters include the speed and direction of the air mass velocity relative to the aircraft, as well as the freestream static pressure. According to Gracy,[1] air data measurements are typically performed using intrusive booms that extend beyond the local boundary layer. These booms have been found to be excellent at making steady-state measurements at low to intermediate angles of attack. However, the performance of these instruments deteriorates during high angles of attack and highly dynamic maneuvers. They are also sensitive to vibration and alignment error and are susceptible to damage during both flight and maintenance.

Flush air data sensing (FADS) systems, described by Whitmore et al.,[2] were developed in response to problems associated with intrusive booms. These instruments infer the air data parameters from pressure measurements taken with an array of ports that are flush to the surface of the aircraft and are, thus, completely nonintrusive. However, because the locations of the pressure measurements are on the outer surface of the aircraft, locally induced flowfields can seriously complicate the calibration of these devices. Additionally, the semiempirical model used by other investigators to process the FADS pressure signals has experienced numerical instabilities that resulted in momentary degradations in the system performance. Current developments in FADS technology, described by Whitmore et al.,[3] have successfully improved algorithm stability but at the cost of increasing the complexity of the algorithm. This paper seeks to apply a neural network approach to develop a FADS estimation algorithm that is inherently stable and that is easier to calibrate and implement than the existing FADS system.

The need to improve the FADS interpolation algorithms provides an ideal opportunity for the application of artificial neural network techniques. FADS systems use an input vector comprising 11 pressure measurements P_i to estimate an output vector that includes four air data parameters, α_∞, β_∞, P_∞, and q_c. The relationship between these two vectors is complex and highly nonlinear. Computational fluid dynamics can be used to study the problem, but the need for a real-time invertible model makes the application of this approach infeasible. Neural networks, which require large quantities of training data, are very well suited to situations such as this, where the more traditional approaches are either insufficient or too complex but the empirical data are plentiful. These neural network systems allow the correlation of complex nonlinear systems without requiring explicit knowledge of the functional relationship that exists between the input and output variables of the system.

Compared with existing semiempirical FADS techniques, neural networks have the advantage of being easier to develop while providing a higher level of detail in the mapping between the two

Received Nov. 18, 1997; revision received July 14, 1998; accepted for publication July 29, 1998.

*Graduate Student, Mechanical and Aerospace Engineering Department, 405 Hilgard Avenue.
†Vehicle Dynamics Group Leader, Aerodynamics Branch, Mail Stop D-2033, P.O. Box 273.
‡Professor, Mechanical and Aerospace Engineering Department, 405 Hilgard Avenue.

vector spaces. The level of detail available with a neural network FADS (NNFADS) system is limited only by the detail available in the flight data and the computational time required to train those data to a network.

The first stage in the development of the NNFADS system, which was reported by Rohloff and Catton,[4] was to prove that the FADS air data estimation mapping could be adequately represented by a trained network. A set of training data, which consisted of a single FADS flight profile, was obtained from the bank of data available in the NASA Dryden Flight Research Center flight data access system. The transformation for the FADS pressure input space to the air data output space over the range of conditions considered, which included $0.1 < M < 0.9$, was successfully represented by a neural network with three hidden layers of 75 processing elements each. The network included 11 input elements corresponding to the 11 FADS pressure measurements and four output elements for the four air data parameters. These results demonstrated that neural networks could be trained to adequately represent the FADS transformation. Completion of the early stages of this project was facilitated by limiting the network training set to a single flight profile, which limited the applicability of the trained neural network to similar flight conditions. Subsequent stages in the development of this system, which are described subsequently, used a wider range of flight data pieced together from multiple flight tests to provide a comprehensive training data set across the entire flight envelope of the test aircraft.

II. Current FADS Technology

A prototype real-time FADS (RT-FADS) system has been developed at the NASA Dryden Flight Research Center.[2] This system was implemented and tested on the NASA F/A-18B systems research aircraft (SRA) over the entire nominal flight envelope of the F-18 from takeoff to landing ($M < 1.6$, $\alpha_\infty < 50$ deg, $-25 < \beta_\infty < +25$ deg), and the resulting FADS system was shown to be robust to noise in the measured pressures. However, algorithm instabilities were encountered during the development of the RT-FADS system for certain flight conditions, which caused momentary degradations in the system performance. A new adaptation of the FADS system currently being developed at NASA Dryden Flight Research Center for the X-33 single-stage-to-orbit launch vehicle, described by Whitmore et al.,[3] has successfully improved algorithm stability but at the cost of increasing the complexity of the algorithm.

The hardware for the RT-FADS system is located in a modified radome of the SRA. The radome and the RT-FADS instrumentation are shown in Fig. 1. The system consists of the FADS pressure port matrix and the associated measurement transducers. A matrix of 11 pressure orifices was integrated into a composite nose cap and attached in place of the nose boom. The locations of the ports on the nose cap are shown in Fig. 2 and are defined in terms of clock and cone coordinate angles ϕ and λ, respectively. The pressures at these locations are sensed by 11 digital absolute pressure transducers located on a palette inside the SRA radome. Pressure transducers at the FADS ports are sensed by 11 digital absolute-pressure transducers. The transducers are reported by Whitmore et al.[2] to have a repeatability that exceeds 0.01% of full scale with a measurement range from 1.50 to 40.00 psia.

The reference air data source reported by Whitmore et al.[2] was generated by combining information from various sources. These measurements included the onboard inertial navigation system attitudes, rates, and accelerations; radar tracking velocity and position data[5]; and rawinsonde weather balloon sounding data.[6]

The weather balloon sounding data were verified in flight by flying 360 deg at a constant speed and performing a level turn before and after each maneuver. When the indicated airspeed was averaged over the course of the 360-deg turn, the effects of the winds were eliminated. The difference between the averaged airspeed and the averaged radar-derived groundspeed was the velocity error for that airspeed caused by the static source error. The velocity error provided an accurate point on the position error curve.

The local wind direction and speed were evaluated by adding this static source velocity error to the indicated airspeed reading and then plotting the groundspeed and corrected airspeed as a function of time. Velocity data were converted to Mach numbers using temperature values obtained from the rawindsonde balloon soundings and radar-derived geometric altitude. Local ambient pressure values were evaluated using balloon soundings and radar-derived geometric altitude.

The accuracy of the reference data source techniques was reported in Refs. 2 and 6. The differences between the true and expected values were shown to be $\Delta M < 0.02$, $\Delta\alpha < 0.4$ deg, $\Delta\beta < 0.4$ deg, $\Delta q_c < 12$ psf, and $\Delta P_\infty < 17$ psf.

III. NNFADS Training Set Compilation

The FADS instrumentation has been used to generate large amounts of data from scores of flight tests. Data files include time histories of the raw pressure readings from the FADS sensor along

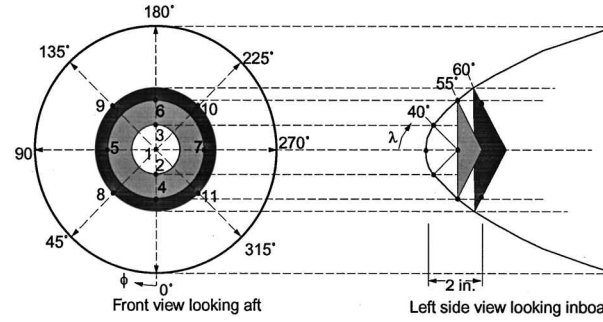

Fig. 2 FADS pressure port configuration.

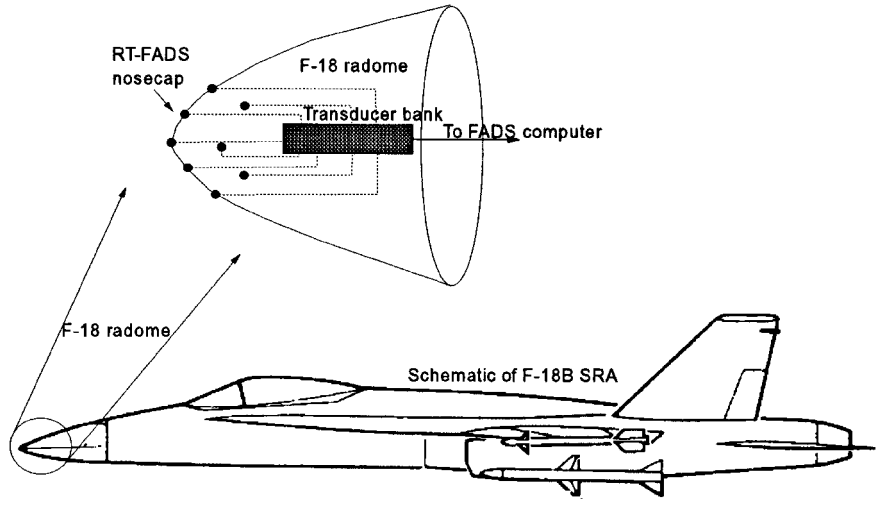

Fig. 1 FADS hardware.

with the corresponding air data estimates compiled from the array of flight test instrumentation. In most cases, the air data estimates from the onboard air data computer (ADC) were used as the training set. These estimates were shown to agree with the reference data set within the expected level of precision ($\Delta M < 0.02$, $\Delta\alpha < 0.4$ deg, $\Delta\beta < 0.4$ deg, $\Delta q_c < 12$ psf, and $\Delta P_\infty < 17$ psf). The data archives included a total of 40 relevant time histories, but before the training process could proceed, two problems had to be addressed. First, many of the data sets included blocks of time where data were suddenly outside physically reasonable limits due to momentary losses in the telemetry data. Second, there were simply too many data to be processed all at once.

The first step in improving the quality and reducing the quantity of data sets was to graph the time histories of all of the relevant parameters from each of the 40 candidate sets. Often the data from a particular set were very limited, or the data were outside of a physically reasonable domain. These sets were simply removed from the group of candidates. Causal inspection of the remaining sets revealed that a vast majority of the available data was recorded at what could be called moderate conditions, where the angle of attack was less than 15 deg and the Mach number was less than 0.9. To further reduce the overwhelmingly large quantity of candidate data, some of the less interesting data sets were simply discarded. Care was taken to keep any data where either $\alpha_\infty > 15$ deg or $M > 0.9$. The final step in this preliminary filtering process was to exclude portions of the time histories that were recorded either before takeoff or after landing. After this process, the total group of candidates was reduced to a more manageable 24 sets, with a total of 1.5×10^6 frames or points of data.

IV. Training Preprocessing

A. Source of the Air Data Parameters

Time histories were analyzed at 20 samples per second. The extracted parameters included the FADS pressure measurements from the nose of the aircraft, along with a best estimate of the associated air data parameters P_∞, q_c, α_∞, and β_∞. The best estimate, however, was not always obtained from the onboard ADC. The onboard instrumentation was applicable for angles of attack of only 25 deg or less. Above that, flow separation caused the aircraft's instrumentation to fail. A best estimate of the true air data state for high angles of attack had to be obtained from the output of the RT-FADS algorithms. Independent measurements, reported by NASA Dryden Flight Research Center, indicated that RT-FADS agreed with the reference data set within the expected level of accuracy ($\Delta M < 0.02$, $\Delta\alpha < 0.4$ deg, $\Delta\beta < 0.4$ deg, $\Delta q_c < 12$ psf, and $\Delta P_\infty < 17$ psf). In addition to high-angle-of-attack data, all estimates of the angle of side slip were based on the output of the RT-FADS algorithm because the onboard ADC did not measure this parameter. The source of each parameter is summarized as follows: 1) P_i always used FADS instrumentation pressure data; 2) P_∞, q_c, and α_∞ with low angle of attack ($\alpha_\infty < 25$ deg) then used onboard ADC estimates and with high angle of attack ($\alpha_\infty \geq 25$ deg) then used RT-FADS estimates; and 3) β_∞ always used RT-FADS estimates.

B. Data Filtering

Despite the initial screening of the data sets, occasional spikes, which resulted from lapses in communication between the ground-based tracking system and the onboard telemetry, were present in the parameter time histories. When these spikes occurred, the value of one or more of the parameters would change drastically from one time frame to the next and would return back to the original values shortly thereafter. The intense time rates of change associated with these spikes indicated that they were physically unreasonable.

To remove these spikes, the time histories were processed through a low-pass filter. This filter used the information available from two successive time frames to generate an expected value for the third in the series. If the recorded value of any of the parameters of a given time frame varied by more than a given threshold ϵ from the expected value predicted by the low-pass filter, then that entire frame of data was removed from the data set. Otherwise, the original recorded values were maintained. Note that the expected values from the filter were used only to identify corrupted data and not to replace them.

Corrupt data frames were identified using a second-order low-pass filter similar to that described by Brown.[7] The transfer function for this filter is given by

$$G(s) = \frac{\omega_n}{s^2 + 2\xi\omega_n s + \omega_n^2} \tag{1}$$

where ξ is the damping ratio and ω_n is the natural frequency of the filter. The filter was discretized using a bilinear transform similar to that found in Ref. 7.

When starting this technique with a new series of data, the expected value for the first two frames had to be initialized. This was accomplished by setting the expected value $x_f(t) = x(t)$ for the first two time frames ($t = 0$ and $t = \mathrm{d}t$).

Applying this technique to the FADS data, the filter tuning parameters were optimized heuristically. Setting $\xi = \sqrt{2}/2$, $\omega_n = 3\pi$, and $\epsilon = 1\%$ guaranteed that all of the physically unreasonable data spikes were removed with only a 5% reduction in the data set. A heuristic approach is justified here because the filter was used only to identify suspect data frames and not to replace them.

V. Determination of the Boundaries of the Training Set

Trained with a given set of data, neural networks are excellent tools for interpolating between the original examples. However, the performance of a neural net falls substantially when applied to data outside the domain of the original set. In other words, neural networks are ineffective at extrapolating outside the original training domain. It is, therefore, very important to use all of the points coincident with the envelope around the data set when constructing a subset to be used in the actual training process. One technique for determining the outermost points of a set of data is to examine the convex hull polytope (CHP) of the set, as described by Courrieu.[8]

The CHP associated with a set of n-dimensional data is defined as the minimum convex surface that encompasses the entire set. An example of a CHP for a two-dimensional data set is included in Fig. 3. A region is considered convex if, for any two points in the region, all of the points on the line segment joining the two are also in that region. Thus, the convex hull of a set of data includes the line segments between every combination of two points from the set. Additionally, all of the points along each of these line segments exhibit the properties of a convex set, i.e., the segments joining these points are also elements of the convex hull. These connections continue until the convex set completely fills the region delimited by the polytope, or n-dimensional polyhedron, defined by vertices coinciding to the outermost points of the original data set.

An algorithm for determining the vertices of a CHP is presented by Courrieu.[8] This approach is based on measuring the Euclidean distance, or exteriority, between a given point and the closest point in the convex set. If the point lies within the CHP, then it is a member of the convex set, and the exteriority is identically zero.

Vertices of the polytope will delimit the domain of the training set. Any new point that lies within this domain will be within the domain of interpolation of the training data. The final set of data used to train the neural networks should, therefore, include the vertices of the polytope.

VI. Training Set Density Distribution

Neural networks are most successful when they are used for interpolation, and including the vertices of the convex hull of the available data will guarantee that the trained network will be operating in the interpolation mode. However, if the domain being mapped by the network is wide or internally complex, then a training set composed solely of the exterior points will not be enough to ensure adequate performance of the neural net. The training set must also include enough internal data to properly represent the entire operating domain. Ideally, the data set will be fairly evenly distributed across the domain without any major holes. Techniques, therefore, have been developed to examine the density distribution of a set of training data. Areas in which the data are too sparse can then be identified, and efforts can be taken to obtain additional data to fill in the holes.

A. Constructing Density Graphs

The method used to determine the number density distribution of a given set of n-dimensional data relies on the fact that the

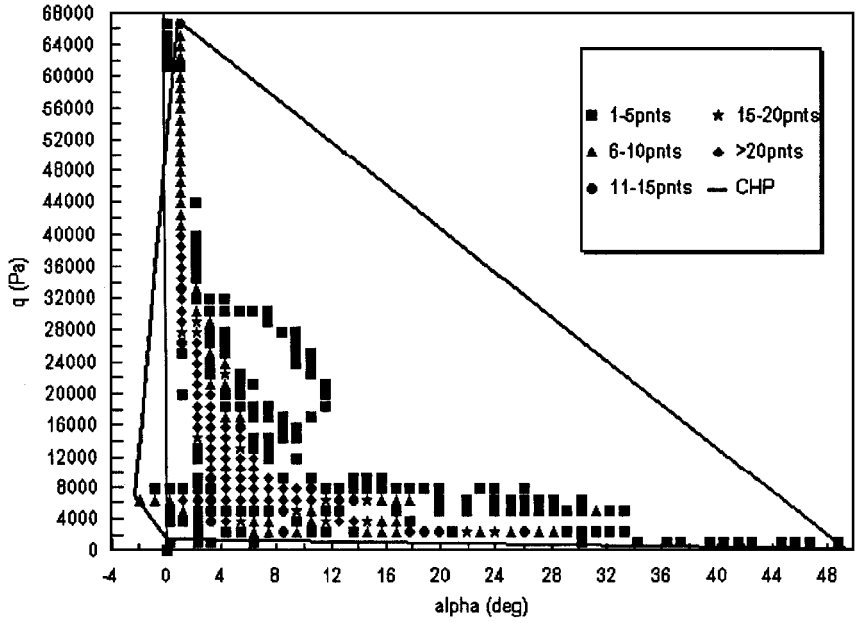

Fig. 3 Number density distribution q_c vs α_∞.

n-dimensional space can be taken two parameters at a time to generate planar projections of the original set. Graphical demonstration of the overall density distribution is achieved by projecting the n-dimensional vectors onto each of the possible two-dimensional coordinate planes. The following steps were used to generate the density graphs:

1) Given an n-dimensional data set, there are $(n-1)!$ combinations of two-dimensional planes in which the original vectors are projected. Each plane is considered separately.

2) These planes are divided into equal bins, and the number of data points within each bin is recorded.

3) The coordinates of each bin are recorded along with the number of points in that bin.

4) The distribution of points is then graphed and evaluated visually.

These graphs give an indication of the number density distribution of the data set. This process is not guaranteed to find all of the holes in the data set, but the larger voids should be uncovered.

The density graphs produced by the preceding procedure were used to visually check for holes in the full four-dimensional data set. Figure 3 gives an example of a two-dimensional projection in the output space of density distribution of the candidate set projected in the α_∞–q_c plane. Note that the data are relatively thin for high angles of attack and high dynamic pressure. Previous versions of this same graph revealed that the original data density of these regions was too thin. Therefore, before the graph in Fig. 3 was prepared, the data archives were scoured in search of any and all data for $\alpha_\infty > 15$ deg and $M > 0.9$ (high M corresponding to high values of q_c). Even after this comprehensive search, these regions are still thin, but they now include all of the available data for that region.

B. Extraction of Evenly Distributed Training Set

Once the entire set of 1.5×10^6 data frames was checked for holes, a smaller set had to be extracted for use in training the neural nets. The construction of this smaller set had to meet three criteria.

1) The vertices of the convex hull polytope had to be included.

2) The data had to be uniformly extracted from the domain of the set.

3) The final subset of training data had to be much smaller.

The binning techniques used to construct the density graphs were adapted such that an equal number of points could be extracted from each four-dimensional bin. The actual number being taken from the bins was varied until the extraction process gave a reasonably sized subset of training data of about 40,000 points. The points that composed the vertices of the convex hull were added to this subset, and the final training data set was complete.

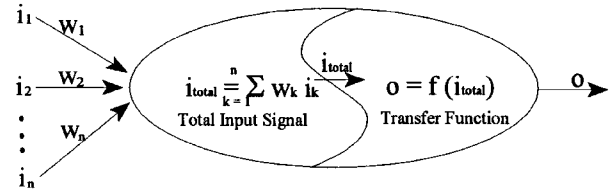

Fig. 4 Single processing element.

The final step in generating the NNFADS training set was to break the set of 40,000 points into a uniformly distributed training set and a complementary test set. Points for the training set were extracted randomly from the larger set. The neural networks were trained to represent the data in the training set and were subsequently applied to the test set to measure their ability to generalize to new data. The optimum proportions of the two sets had to be determined heuristically. If the training set was too small, then the network would not be able to represent the complex mapping existent in the FADS data. On the other hand, if the training set was too large, then the training algorithm would have trouble converging down to a reasonable level of error within an acceptable period of time. Typically, the optimum training set size required for the neural networks in the NNFADS processor was found to be in the range from 1000 to 5000 points. Note that the training set always included the points of the CHP.

VII. NNFADS Training Process

A. Neural Network Modeling

Artificial neural networks comprise many interconnected processing elements (PEs), an example of which is shown in Fig. 4. As shown in Fig. 4, each of the input signals i_i impinging on the unit has a corresponding weight w_i. The total input to the PE, i_{total}, is given by the sum of the products of the inputs and the weights. This net signal is subsequently passed on to an activation function, or transfer function $f(i_{\text{total}})$, which is typically nonlinear. This activation function is used to transform the net input signal to the output signal o, which is passed forward to other units in the network. A transfer function that is fairly common in neural network applications and that was used in this study, is the logistic activation function. Output signals for nodes using this type of function are given by

$$o = \frac{1}{1 + \exp\left[-\left(\sum_\theta w_\theta i_\theta + \theta\right)\right]} \tag{2}$$

where θ is a bias term and i is summed over the connections from the previous layer. The term $w_i i_i$ is the weighted output of a node from the previous layer.

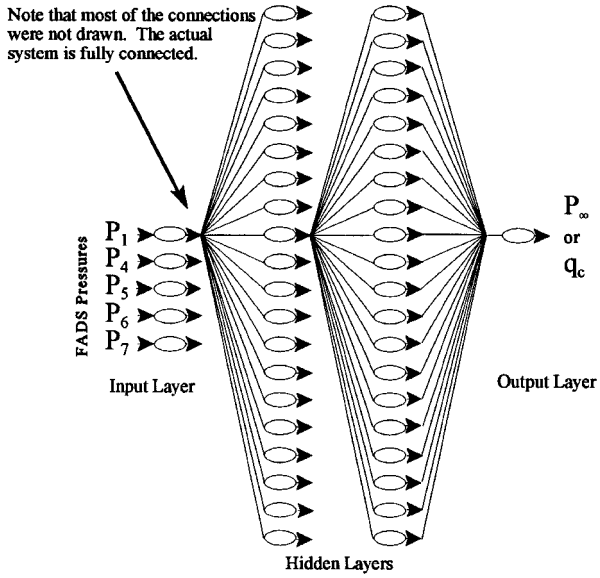

Fig. 5 Neural network architecture.

Complete networks are constructed by connecting several of these PEs into a series of layers, as shown in Fig. 5. The first layer, referred to as the input layer, simply receives and distributes the input signals without any sort of transfer function. The input signals are passed along through the network to the last layer, or output layer, which produces the final network outputs. All of the layers in between these two, which have no direct external contact, are called hidden layers. There is no deterministic means of defining the appropriate network architecture, and a heuristic approach must be employed. As can be seen from Fig. 5, there can be many more connections in a network than there are PEs. It is this array of weighted connections that contains the information of the trained neural network.

The neural network training algorithm that is used in this work is a variation of the general backpropagation of errors method.[9] This procedure starts by defining an error function and then uses steepest-descent methods to find a set of network weights that minimizes the function and that optimizes the performance of the network for the particular task.

The training procedure involves the presentation of a set of pairs of input and output patterns. The input pattern is used by the neural network to calculate its own output vector. These calculated outputs are then compared to the desired output pattern, or target vector. If there is no significant difference, then no training is required. Otherwise, the weights in the network are changed to reduce the error. The change in weights after a new pattern presentation, $\Delta w(n+1)$, is proportional to the product of an error signal δ and the output of the unit from which that node is receiving input i_i, added with a fraction of the previous update $\Delta w(n)$,

$$\Delta w_i(n+1) = \eta \delta i_i + \alpha \Delta w_i(n) \tag{3}$$

where η is the constant of proportionality or the learning rate parameter and α is the momentum parameter. A detailed derivation of the training process can be found in Ref. 9.

At the start of the training procedure, the weight space has to be initialized. A random set of weights is typically assigned to the network to prevent any symmetries in the initial space. During the training procedure, any symmetries that might exist would be maintained during the weight updates, and there would be no relative change within the space. This symmetry, in effect, would prevent the network from being trained.

The training rules described called for weight updates after each presentation of an input/output pattern. This procedure is sometimes referred to as continuous updating. Networks can sometimes be trained faster if updating occurs after each cycle through all of the available training data. One entire cycle though all of the training patterns is sometimes referred to as a single epoch, and summing the error seen by each weight over an entire epoch before making adjustments is, consequently, referred to as epochwise updating.

B. Training NNFADS Neural Networks

The tuning parameters of the training process had to be adapted to the current task using a trial-and-error approach because there were not any set rules for choosing the optimum values. In fact, a great deal of time was required to systematically vary the parameters to find acceptable and/or optimal values and settings. For example, a high level of success was experienced when the original weight state was randomly distributed between values of -0.5 and $+0.5$. During the actual training process, epochwise updating with a fanning in of the learning rate parameter η was found to work well. Fanning in the learning rate parameter just means that η is divided by the number of nodes in the layer before it is applied to updating the weights in that layer.

A complication in the training process arose from the need to change some of the parameters dynamically over time. Parameter values that worked well during the initial stages of training caused instabilities in the weight space convergence during the final stages. Some of the parameters were, therefore, adjusted for the different stages of the training process. These parameters included the learning rate parameter η and the momentum parameter α. These parameters were adjusted during the three main stage using the following schedule:

1) The first stage lasted for about the first 100 to 500 epochs. During this phase, the root mean square error of the neural network predictions oscillated at fairly high levels before starting to decay. The following set of parameters was found to initiate the decay of the rms error: $\eta = 0.15$–0.75 and $\alpha = 0.05$.

2) After the rms error started to decay, the settings $\eta = 0.05$ and $\alpha = 0.95$ were found to accelerate the convergence process without causing instabilities. The rms error dropped drastically during the next 1000–5000 epochs before leveling off.

3) The final stage consists of a very slow, stable decline in the rms error over 50,000–200,000 epochs, parameters $\eta = 0.05$ and $\alpha = 0.99$.

A sample training curve, shown in Fig. 6, was generated to demonstrate the advantage of staging the training parameters instead of just setting them to constant values. Two neural nets were trained to represent static pressure over the entire flight envelope, and the rms error of the predictions was recorded as a function of training time, i.e., the number of epochs. It can be seen from the graph (Fig. 6) that the staged process converges to a minimum error faster, and to an overall lower level of error, than the straight process.

VIII. Results

The outlined data sets and training techniques were applied in the development of a pair of neural networks. The first network translates the signals from the FADS instrumentation into an estimate of static pressure, and the second gives an estimate of dynamic pressure. (Results for α_∞ and β_∞ are not available at this time.) The fully connected feedforward architecture of both networks includes 5 input nodes, two hidden layers of 20 nodes each, and a single output node, as shown in Fig. 5. The five input nodes correspond to 5 of the 11 FADS pressure signals, and the output nodes of the two networks provide the estimates of P_∞ and q_c, respectively. The size of these networks was optimized heuristically during the early stages of their development. The trial-and-error search demonstrated that networks with too few nodes in the hidden layer were incapable of representing the system and networks with too many nodes required too much computational training time. Thus, the ideal network included the minimum number of nodes that could still adequately represent the system.

There are several important differences between the current results and those presented by Rohloff and Catton.[4] First, and most importantly, the current neural networks were trained to cover the entire flight envelope of the F-18 SRA, including $0.1 < M < 1.6$, whereas the previous results were limited to a much smaller range. Second, the air data parameters are now being estimated from smaller independent networks instead from a large, single network with multiple outputs. Splitting the networks resulted in an increase in the accuracy of the air data estimates while reducing the computational burden required to process each set of FADS pressure signals. Finally, only a fraction of the available pressure signals is now being used for each network. Referring to the index notation presented in

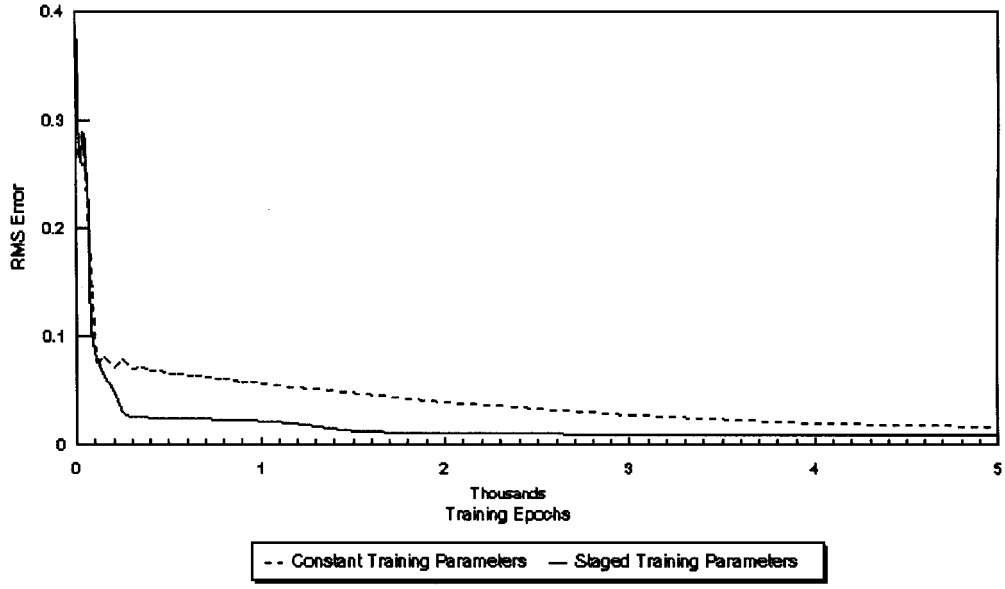

Fig. 6 Sample training convergence curves.

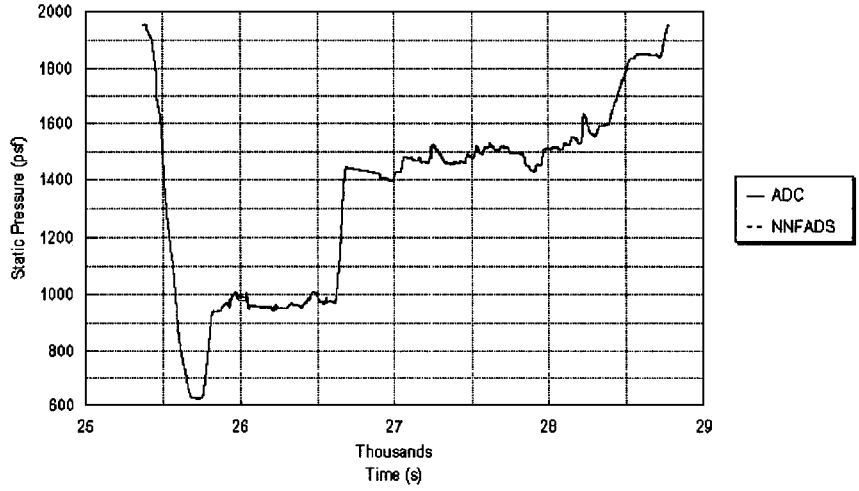

Fig. 7 Neural network estimate of static pressure P_∞.

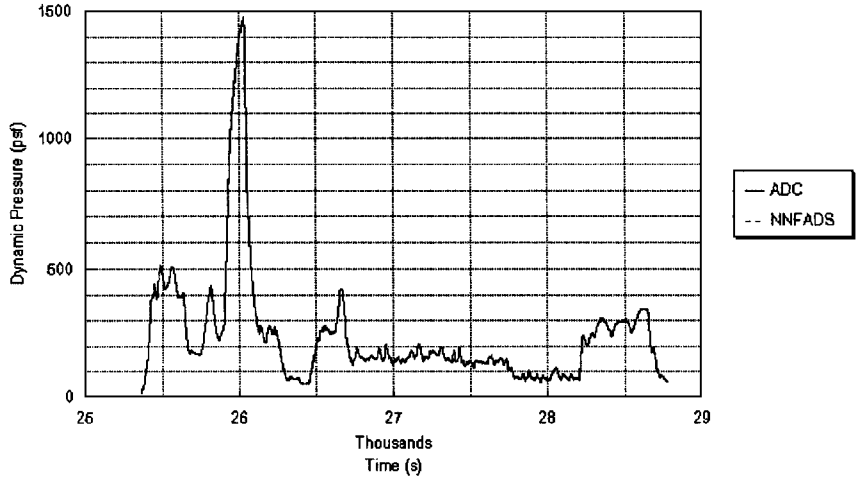

Fig. 8 Neural network estimate of dynamic pressure q_c.

Fig. 2, ports 1 and 4–7 were used as inputs to the current networks. This configuration was determined by systematically reducing the number of inputs before training and by later observing the accuracy of the trained network. The accuracy was found to deteriorate if fewer then five inputs were used, but no more then six inputs were needed to still produce excellent results. Note that the symmetry of the original configuration was maintained in each of the reduced input sets.

The performance of the two trained neural nets was demonstrated for 1 of the 24 flight profiles described. The neural net predictions of static and dynamic pressures are graphed as a function of time in Figs. 7 and 8, respectively. The Mach number was calculated from these estimated values and is graphed as a function of time in Fig. 9. The estimates from the ADC are included in the three graphs. Notice that neural network values follow closely those of the ADC across the entire flight profile. Over most of the graphs, the two curves are

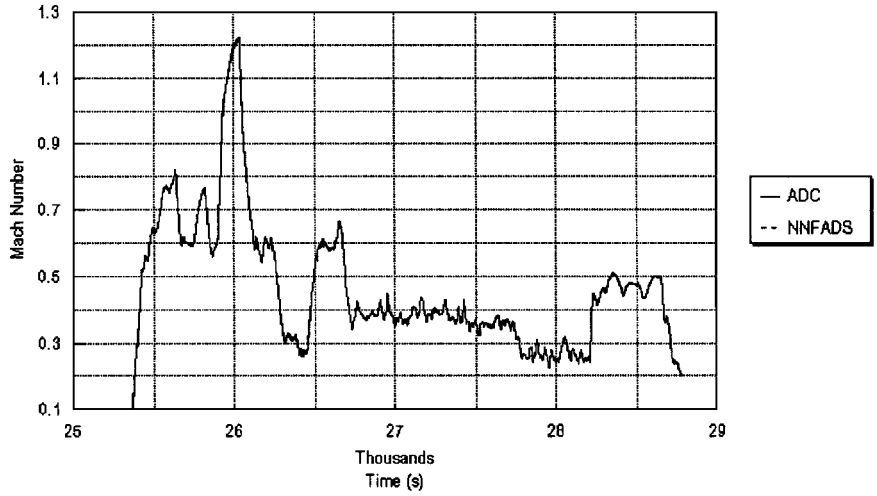

Fig. 9 Calculated estimate of Mach number *M*.

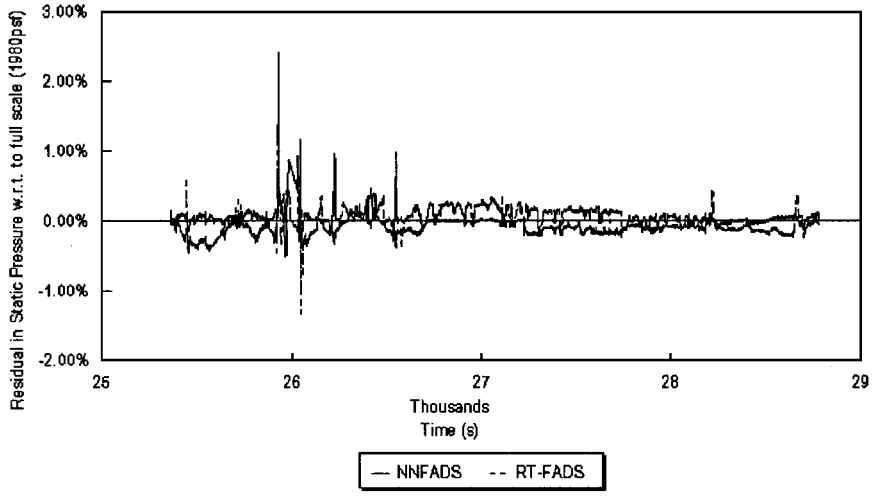

Fig. 10 Residuals in static pressure.

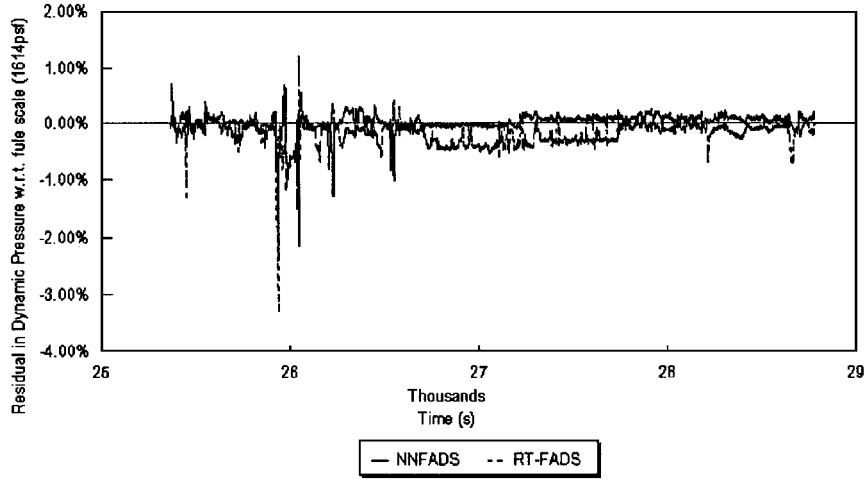

Fig. 11 Residuals in dynamic pressure.

indistinguishable. The only significant deviation occurred in Fig. 7 for the estimate of static pressure. At around 26,000 s, during the high-Mach-number maneuver, the two curves varied by as much as 1.5% with respect to full scale (1980 psf). Fortunately, this is still within an acceptable range of error.

The accuracy of the RT-FADS system relative to the ADC estimates is being used as a standard of performance for the NNFADS system. The differences between the neural network estimates and those of the ADC must not exceed the difference between the RT-FADS system and the ADC. The residual in the air data estimates is defined as the difference between the estimated values and the values taken from the ADC. The ratio of these residuals to the maximum (full-scale) values of the air data parameters are graphed for static and dynamic pressures in Figs. 10 and 11, respectively. The residuals from both the NNFADS and RT-FADS estimates are included in these graphs. Close inspection of these graphs shows that

both the maximum and mean error of the neural networks are lower than those of the RT-FADS system. The accuracy of the RT-FADS system was already shown to be of the same order as the reference data set ($\Delta M < 0.02$, $\Delta\alpha < 0.4$ deg, $\Delta\beta < 0.4$ deg, $\Delta q_c < 12$ psf, and $\Delta P_\infty < 17$ psf). Therefore, the accuracy of the neural networks is within the acceptable range.

IX. Conclusions

Neural networks have been successfully developed to estimate freestream static and dynamic pressures from an array of pressure measurements taken from ports located flush on the nose of an aircraft. The accuracy of these networks has been shown to meet the accuracy of other aerodynamic-model-based systems over a wide range of flight conditions from subsonic to supersonic speeds. Unlike the aerodynamic models, the neural networks are implemented using a set of explicit calculations and are, thus, unsusceptible to computational instabilities. However, the fault tolerance and extrapolation capability of this system still need further evaluation to quantify uncertainty, and many of the details of the final algorithm still need to be worked out. At this time, a neural-network-based FADS system promises to be a viable alternative to currently available systems.

Acknowledgment

This work was supported under Grant NCC 2-374 from NASA Dryden Flight Research Center, Edwards Air Force Base, California.

References

[1]Gracey, W., "Measurement of Aircraft Speed and Altitude," NASA Reference Publication 1046, May 1980.

[2]Whitmore, S. A., Davis, R. J., and Fife, J. M., "In-Flight Demonstration of a Real-Time Flush Airdata Sensing (RT-FADS) System," NASA TM-104314, Oct. 1995.

[3]Whitmore, S. A., Cobleigh, B. R., and Haering, E. A., "Architecture, Algorithms, and Calibration of the X-33 Flush Airdata Sensing (FADS) System," AIAA Paper 98-0201, Jan. 1998.

[4]Rohloff, T. J., and Catton, I., "Development of a Neural Network Flush Airdata Sensing System," *Proceedings of the 1996 ASME International Mechanical Engineering Congress and Exposition*, FED-Vol. 242, American Society of Mechanical Engineers, Atlanta, GA, 1996, pp. 39–43.

[5]Haering, E. A., Jr., and Whitmore, S. A., "FORTRAN Program for Analyzing Ground-Based Radar Data: Usage and Derivations, Version 6.2," NASA TP-3430, 1994.

[6]Ehernberger, L. J., Haering, E. A., Jr., Lockhart, M. G., and Teets, E. H., "Atmospheric Analysis for Airdata Calibration on Research Aircraft," AIAA Paper 92-0293, Jan. 1992.

[7]Brown, R. G., *Introduction to Random Signal Analysis and Kalman Filtering*, Wiley, New York, 1983, pp. 140–174.

[8]Courrieu, P., "Three Algorithms for Estimating the Domain of Validity of Feedforward Neural Networks," *Neural Networks*, Vol. 7, No. 1, 1994, pp. 169–174.

[9]Rumelhart, D. E., *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, MIT Press, Cambridge, MA, 1986, pp. 318–362.

G. Laufer
Associate Editor