SOLUTION OF NONLINEAR ORDINARY DIFFERENTIAL EQUATIONS BY FEEDFORWARD NEURAL NETWORKS

Andrew J. Meade, Jr.* and Alvaro A. Fernandez
Rice University
Department of Mechanical Engineering
and Materials Science
Houston, Texas, 77251-1892, USA
Phone: (713) 527-8101 ext. 3590
email: meade@rice.edu

To appear in Mathematical and Computer Modelling, 1994

ABSTRACT

It is demonstrated, through theory and numerical examples, how it is possible to directly construct a feedforward neural network to approximate nonlinear ordinary differential equations without the need for training. The method, utilizing a piecewise linear map as the activation function, is linear in storage, and the L_2 norm of the network approximation error decreases monotonically with the increasing number of hidden layer neurons.

The construction requires imposing certain constraints on the values of the input, bias, and output weights, and the attribution of certain roles to each of these parameters. All results presented used the piecewise linear activation function. However, the presented approach should also be applicable to the use of hyperbolic tangents, sigmoids, and radial basis functions.

* Recipient of correspondence, email: meade@rice.edu
This work is supported under NASA grant NAG 1-1433.

SOLUTION OF NONLINEAR ORDINARY DIFFERENTIAL EQUATIONS BY FEEDFORWARD NEURAL NETWORKS

Andrew J. Meade, Jr.* and Alvaro A. Fernandez
Department of Mechanical Engineering
and Materials Science
Rice University
Houston, TX 77251-1892, USA

1 INTRODUCTION

The rapidly growing field of connectionism is concerned with parallel, distributed, and adaptive information processing systems. This includes such tools as genetic learning systems, simulated annealing systems, associative memories, and fuzzy learning systems. However, the primary tool of interest is the artificial neural network.

The term Artificial Neural Network (ANN) refers to any of a number of information processing systems that are biologically inspired. Generally they take the form of directed-graph type structures [1] whose nodes perform simple mathematical operations on the data to be processed. Information is represented within them by means of the numerical weights associated with the links between nodes. The mathematical operations performed by these nodes, the manner of their connectivity to other nodes, and the flow of information through the structure are patterned after the general structure of biological nervous systems. Much of the terminology associated with these systems is also biologically inspired; thus networks are commonly said to be "trained," not programmed, and to "learn" rather than to model.

ANNs have proven to be versatile tools for accomplishing what could be termed higher order tasks such as sensor fusion, pattern recognition, classification, and visual processing. All these applications are of great interest to the engineering field, however, present ANN applications have created the opinion that networks are unsuited for use in tasks requiring accuracy and precision, such as mathematical modelling and the physical analysis of engineering systems. Certainly the biological underpinnings of the neural network concept suggest networks would perform best on tasks at which biological systems excel and worse or not at all at other tasks.

Contrary to this opinion the authors believe that continued research into the approximation capabilities of networks will show that ANNs can be as numerically accurate and predictable as conventional computational methods. It is also believed that in viewing ANNs as numerical tools, advances in training and analyses of existing architectures can be made. In a more immediate sense, benefits of this approach will enable the neural network paradigm, with all of its richness of behavior and adaptability, to be mated to the more purely computational paradigms of mathematically oriented programming used by engineers in modelling and analysis.

Presently, the most popular application of ANNs in science and engineering is the emulation of physical processes by a feedforward artificial neural network (FFANN) architecture using a training algorithm. Because this paradigm requires exposure to numerous input-output sets, it can become memory intensive and time consuming. Considerable effort may be saved if the mathematical model of a physical process could be incorporated directly and accurately into the FFANN architecture without the need for examples, thereby shortening or eliminating the learning phase.

Previous work has viewed mathematical modeling, or connectionist equation solving, as a new area in which to apply conventional connectionist paradigms. For example, Takeda and Goodman [2] and Barnard and Casasent [3] use the optimization network paradigm, specifically variations on the Hopfield network, to solve linear systems of equations. The linear systems to be solved are expressed as the minimization of a quadratic function, and the weights of the Hopfield net are trained until the function is minimized. Lee and Kang [4] extend this use of Hopfield-type nets to the solution of differential equations. Wang and Mendel [5] use a more or less standard FFANN architecture with the backpropagation algorithm to solve linear algebra problems by repeatedly exposing a net to the solutions, with the connection structure constrained by the nature of the problem being solved.

* Assistant Professor. Author to whom all correspondence should be addressed. email: meade@rice.edu
The authors would like to thank technical monitor M. Tracy of NASA Langley Research Center for her helpful suggestions and assistance. This work is supported under NASA grant NAG 1-1433.

In this paper, the authors take a different approach by investigating the possibility of drawing parallels between the network and certain mathematical objects from function approximation theory. In this manner it will be shown that FFANNs can be constructed to model given nonlinear ordinary differential equations without the need for training or the creation of datasets. The weights of these networks can be determined by solving systems of both linear and nonlinear algebraic equations. The resulting nets are architecturally identical to those that use conventional training techniques, yet they possess monotonically decreasing errors for increasing number of processing elements.

Progress in this approach should also be of interest to the connectionist community, since the approach may be used as a relatively straightforward method to study the influence of the various parameters governing the FFANN. Additionally, applying an FFANN to the solution of a differential equation can effectively separate the influences of the quality of data samples, network architecture, and transfer functions from the network approximation performance.

For numerical examples, a simple FFANN using a single input with multiple output neurons, with a single hidden layer of processing elements utilizing a piecewise linear map as the activation function, has been constructed to approximate accurately the solution to two benchmark problems from the field of fluid mechanics. The first benchmark problem is a third order nonlinear ordinary differential equation known as the Falkner-Skan equation. The second problem is a system of three coupled nonlinear second and third order ordinary differential equations, also from the field of fluid mechanics. In addition, it will be demonstrated how the error of the constructed FFANNs can be controlled.

2 FUNCTION APPROXIMATION

Function approximation theory, which deals with the problem of approximating or interpolating a continuous, multivariate function, is an approach that has been considered by other researchers [6]-[8] to explain supervised learning and ANN behavior in general. With this in mind, consider that the most common method of function approximation is the functional basis expansion. A functional basis expansion represents an arbitrary function $y(x)$ as a weighted combination of linearly independent basis functions (Υ) that are functions of the independent variable x,

$$y(x) = \sum_{q=1}^{T} \Upsilon_q(x)\ c_q \tag{1}$$

where T is the total number of basis functions and c_q are the basis coefficients. A common example of a functional basis expansion is the Fourier series, which uses trigonometric basis functions. Other basis functions can be used as well, but the values of the coefficients c_q will of course vary depending on the basis functions selected.

Examination of Eq.(1) reveals it to be a scalar product of the basis functions and expansion coefficients. Similarly, the mathematical operations performed by the simple FFANN of Fig. 1 can be interpreted as a scalar product of the transfer function output values and the output weights.

By appealing to function approximation theory, we can intelligently choose the transfer functions and weight arrangements that will optimize the FFANN's performance in representing functional relationships. Further, we can assign roles to the various parameters in the net. For example, the output weights can act as the expansion coefficients for the basis and should in principle be computable by a host of conventional techniques. Similar insights are found in Cybenko [9].

We have selected the piecewise linear map [10] of Fig. 2 as the activation function for the hidden layer, since it is one of the simplest functions to implement on both digital computers and electronic hardware [11]. The piecewise linear activation function can be modelled by the following equations:

$$\begin{aligned} \Upsilon_q &= -1.0 && \text{for} \quad \xi_q < -1.0 \\ \Upsilon_q &= \xi_q(x) && \text{for} \quad -1.0 \le \xi_q \le 1.0 \\ \Upsilon_q &= +1.0 && \text{for} \quad \xi_q > 1.0 \end{aligned} \tag{2}$$

where inspection of Fig. 1 shows that the transfer function operates on the sum of the weighted input and bias values ($\xi_q = \alpha_q x + \theta_q$), i.e., a linear transformation of the independent variable x. This notation is similar to the unscaled shifted rotations of Ito [12].

Basis functions in function approximation literature are said to either be local (exhibiting a nonzero value over a limited part of the domain only) or global (nonzero mostly everywhere in the domain they are to span). Piecewise linear activation functions may be viewed as global linear polynomial basis functions defined in a piecewise manner with the aid of the transformation into ξ_q. All global basis functions have similar weaknesses including ill-conditioning when solving for expansion coefficients and poor approximation between discretization points ("knots") [13].

One may suspect that many of the common problems associated with FFANN representation could arise simply from the use of a mathematically ill-suited basis approximation. In function approximation theory, the disadvantages of global basis functions are avoided by using piecewise polynomial splines.

The most common polynomial spline is the Chapeau, or "hat" function, as shown in Fig. 3. It is defined as

$$\begin{aligned}
\Phi_i &= \frac{(x - x_{i-1})}{(x_i - x_{i-1})} \qquad \text{for} \qquad x_{i-1} \le x \le x_i \\
\Phi_i &= \frac{(x_{i+1} - x)}{(x_{i+1} - x_i)} \qquad \text{for} \qquad x_i \le x \le x_{i+1} \\
\Phi_i &= 0 \qquad \text{for} \qquad x < x_{i-1} \quad \text{or} \quad x > x_{i+1}
\end{aligned} \tag{3}$$

Note that the polynomial varies between the values of 0 and 1.

Figure 4 shows, for a problem domain $[x_1, x_N]$ discretized with $N = 3$ knots (x_1, x_2, and x_3), how the hat functions must be distributed to provide a linear interpolation between the knots. The algebraic system generated to evaluate the expansion coefficients for this particular polynomial spline is very sparse and well-conditioned. Since the maximum value of the spline is 1, the values of the expansion coefficients are identical to the values of the approximated function at the knots. One drawback, however, is that the approximation is discontinuous in the first derivative.

3 TRANSFORMING PIECEWISE LINEAR ACTIVATION FUNCTIONS TO SPLINES BY WEIGHT CONSTRAINTS

All the mathematical advantages of the hat functions suggest it would be advantageous to transform somehow the piecewise linear map FFANN into a hat-function-based representation. Fortunately the choice of the piecewise linear map as the activation function makes this simple.

Consider a one-dimensional domain as in Fig. 5. If the function centered between x_i and x_{i+1} is multiplied by -1 and added to the second unaltered function (Fig. 6), the result will be a hat function with a maximum value of 2. The transformation of piecewise linear activation functions to hat functions can be described by the following equation:

$$\Upsilon_i^A - \Upsilon_i^B = 2\Phi_i, \quad (i = 1, 2, ..., N) \tag{4}$$

where the superscripts of the activation functions A and B refer to adjoining intervals $x_{i-1} \le x \le x_i$ and $x_i \le x \le x_{i+1}$, respectively. The functions Υ_i^A and Υ_i^B are defined as zero at the midpoints of their respective intervals. A consequence of this formulation is that the number of activation functions can be linked to the number of hat functions desired. Our approach requires twice as many piecewise linear functions as hat functions ($T = 2N$).

3.1 Constraint of Input, Bias, and Output Weights

We will now determine the remaining constraints required to equate the activation function representation (Eq.(1)) to one using the hat functions. Let us discretize the problem domain [a,b] using the knots x_i in the following manner:

$$a = x_1 < x_2 < ... < x_{N-1} < x_N = b$$

We will label this discretization as the problem mesh. To generate the appropriate hat functions at the boundaries, two auxiliary knots x_0 and x_{N+1} are required such that $x_0 < a$, and $x_{N+1} > b$. Using the constraint, $T = 2N$, we may rewrite Eq.(1) as

$$\sum_{q=1}^{T=2N} \Upsilon_q(x)\ c_q = \sum_{q=1}^{N} \Upsilon_q(x)\ c_q\ + \sum_{q=N+1}^{2N} \Upsilon_q(x)\ c_q \tag{5}$$

where the summations of the right hand side can be rewritten without loss of generality as

$$\sum_{q=1}^{N} \Upsilon_q\ c_q = \sum_{i=1}^{N} \Upsilon_i^A\ u_i \quad \text{and} \quad \sum_{q=N+1}^{2N} \Upsilon_q\ c_q = \sum_{i=1}^{N} \Upsilon_i^B\ v_i \tag{6}$$

Therefore

$$\sum_{q=1}^{T} \Upsilon_q(x)\ c_q = \sum_{i=1}^{N} \Upsilon_i^A\ u_i + \sum_{i=1}^{N} \Upsilon_i^B\ v_i \tag{7}$$

It can then be shown that the basis expansion in terms of piecewise linear activation functions can be transformed into a basis expansion in terms of hat functions by selecting the following values for the FFANN weights of Fig. 1 [14]. For the input weights and bias weights:

$$\begin{aligned} \alpha_i^A &= \frac{2}{(x_i - x_{i-1})} \quad , \quad & \alpha_i^B &= \frac{2}{(x_{i+1} - x_i)} \\ \theta_i^A &= -\frac{2x_{i-1}}{(x_i - x_{i-1})} - 1 \quad , \quad & \theta_i^B &= -\frac{2x_i}{(x_{i+1} - x_i)} - 1 \end{aligned} \tag{8}$$

where α_i^A, θ_i^A, α_i^B, and θ_i^B correspond to the activation functions of Eq.(4). For the output weights:

$$\begin{aligned} c_i = u_i &= \frac{w_i}{2}, \quad (i = 1, 2, ..., N) \\ c_{i+N} = v_i &= \frac{-w_i}{2}, \quad (i = 1, 2, ..., N) \end{aligned} \tag{9}$$

where the numbering of the weights with respect to the actual net architecture can be in any order.

The application of Eqs.(4)-(9) permits the following equivalence:

$$y_a(x) = \sum_{q=1}^{T} \Upsilon_q(x)\ c_q = \sum_{i=1}^{N} \Phi_i(x)\ w_i \tag{10}$$

The hat functions Φ_i are defined as having a value of 1 at the knot x_i, as in Eq.(3) and w_i are the basis coefficients associated with the hat functions.

While the weights have been constrained in this approach, the architecture of the network has been preserved. Figure 7 shows the resulting architecture for six processing elements used to form three hat functions for three knots plus two auxiliary knots, as in Fig. 4.

Note that there is no "training" involved in this approach. The values of the input and bias weights are fixed for a given number of knots in the problem domain and are effectively uncoupled from the determination of the output weights. The construction of the network has been reduced to a question of finding the appropriate values of coefficients w_i and then using w_i to determine the values of the pairs of output weights u_i and v_i. The computation of the expansion coefficients w_i can be made from a variety of numerical methods. We apply a procedure stemming from the method of weighted residuals [15].

4 DETERMINATION OF OUTPUT WEIGHTS: THE METHOD OF WEIGHTED RESIDUALS

Given a differential equation represented as $L(y) = g(x)$, where $L(y)$ is some differential operator acting on the dependent variable y, an approximate expansion y_a is substituted for y where

$$y_a(x) = \sum_{i=1}^{N} \Phi_i(x)\ w_i \tag{11}$$

and where $\Phi_i(x)$ are basis functions. Therefore $L(y_a) - g(x) = R(w_1, ..., w_N, x)$, where R is some function of nonzero value that can be described as the error or the differential equation residual.

In the method of weighted residuals, the residual is multiplied or weighed by a function $F(x)$ (the test function), integrated over the problem domain $[a, b]$, and set to zero,

$$\int_a^b FR\, dx = (F, R) = 0 \tag{12}$$

where (F, R) is the inner product of functions F and R. Note that even though the integral is set to zero the differential equation residual R approaches zero only in the mean and only as $x_{i+1} - x_i$ approaches zero. In a connectionist scheme, this is equivalent to letting the number of processing elements (T) in the hidden layer approach infinity. Equation (12) is equivalent then to

$$(F, L(y_a) - g(x)) = 0, \quad \text{or} \quad (F, L(y_a)) = (F, g) \tag{13}$$

Many of the more popular methods of function approximation such as finite difference, subdomain methods, finite elements, collocation, and spectral methods can be derived from Eq.(13) with the appropriate choice of the function $F(x)$ [16]. For this paper, we will use the Petrov-Galerkin technique [16] where $F(x)$ is a modification of the basis functions used to describe the approximation y_a

$$F(x) = F_k(x) = a(x)\ \Phi_k(x), \quad k = 1, 2, ..., N \tag{14}$$

where $a(x)$ is chosen, for instance, to help satisfy boundary conditions of the problem in question. The Petrov-Galerkin technique was chosen for the determination of the output weights because it facilitates treatment of highly nonlinear or discontinuous problems. It should be noted that the Petrov-Galerkin technique will work only if the basis functions to be used are linearly independent, such as hat functions.

With $F(x)$ so defined, evaluation of the integrals leads to a system of algebraic equations. This system is usually linear if the differential equation to be approximated is linear. Otherwise the algebraic equations are nonlinear. These equations can be evaluated for a unique solution through standard techniques [17].

5 NUMERICAL EXAMPLES

5.1 Example 1: The Falkner-Skan Equation

For the first numerical example we constructed a single input, multiple output FFANN to approximate the solution, and the first and second derivatives of the solution, for the following nonlinear ordinary differential equation:

$$\frac{d^3 f}{d\eta^3} + f\frac{d^2 f}{d\eta^2} + \beta\left[1 - \left(\frac{df}{d\eta}\right)^2\right] = 0 \quad \text{for} \quad 0 \le \eta \tag{15}$$

with associated boundary conditions

$$\begin{aligned} f = \frac{df}{d\eta} &= 0 \quad \text{at} \quad \eta = 0 \\ \frac{df}{d\eta} &\to 1 \quad \text{as} \quad \eta \to \infty \end{aligned} \tag{16}$$

Additionally, from the previous boundary conditions

$$\frac{df}{d\eta} \to 1 \quad \text{as} \quad \eta \to \infty \quad \text{then} \quad \frac{d^2 f}{d\eta^2} \to 0 \quad \text{as} \quad \eta \to \infty \tag{17}$$

Equation (15) is known as the Falkner-Skan equation and is derived from a similarity transformation of the system of partial differential equations that model a steady, two-dimensional, and incompressible momentum boundary layer. Central to the similarity transformation are the following restrictions

$$0 \leq x \text{ and } 0 \leq y \;, \quad \overline{u}(x,y) = \overline{U}(x)\, \frac{df}{d\eta}(\eta) \;,$$
$$\overline{U}(x) = Kx^{\beta/(2-\beta)} \;, \quad \eta = y\sqrt{\frac{\overline{U}}{(2-\beta)\nu x}} \tag{18}$$

where x and y are the spatial dimensions in the streamwise and surface normal directions, respectively. The variable $\overline{u}(x,y)$ is the dimensional streamwise boundary layer velocity, $\overline{U}(x)$ is the dimensional outer-layer velocity, K is a proportionality constant with a value of +1, ν is the kinematic viscosity of the fluid, and $\eta(x,y)$ is the similarity variable. A thorough discussion on the derivation and properties of Eq.(15) can be found in reference [18].

The solutions to Eq.(15) demonstrate many boundary layer phenomena such as strong favorable gradients, boundary layer separation, velocity overshoot, and non-uniqueness. The parameter β is a measure of the pressure gradient and can be directly related to the boundary layer about wedges and expansion corners of turning angles θ where

$$\theta = \frac{\beta\pi}{2}$$

If we restrict our attention to cases in which Eq.(15) can model physically valid fluid flows and where $\overline{u}(x,y)$ increases monotonically with y, then $-0.19884 \leq \beta \leq +\infty$. As a consequence of Eq.(18) and $\overline{u}(x,y)$, $df/d\eta$ increases monotonically with η for the restricted β range. It should be noted that for $-0.19884 \leq \beta \leq 0$ reference [20] demonstrates that there are at least two solutions of Eq.(15) for any given β, though only one of the solutions gives the monotonically increasing $\overline{u}(x,y)$. For this example problem we will investigate three values of β

$\beta = -0.15$: flow about an expansion corner of turning angle $\theta = 13.5°$

$\beta = 0$: flow along a flat plate

$\beta = +0.5$: flow about a wedge of half-angle $\theta = 45°$

In our approach to solve the Falkner-Skan equation we note that the independent variable η in Eq.(15) is semi-infinite and the equation itself has final conditions applied at infinity. However, for our restricted values of β a simple asymptotic analysis of Eq.(15), using the constraint of a monotonically increasing $df/d\eta$, determines that the dependent variable f, as well as its second and third derivative, vary monotonically with η. Therefore, it should be possible to transform Eq.(15) into a form where the dependent variable, or its derivatives, act as the independent variable. As previously mentioned, the hat functions used as bases in the approximate solution are linear in nature which means that the second, third, and higher derivatives of the bases are identically zero. In order to obtain a non-trivial solution we not only introduce a change of independent variable from η to u, but also a change of dependent variable from f to τ, where

$$u = \frac{df}{d\eta} \quad \text{with} \quad 0 \leq u \leq 1 \quad \text{and} \quad \tau = \frac{d^2 f}{d\eta^2} \tag{19}$$

With the application of Eqs.(19) and the chain rule, the Falkner-Skan equation reduces to

$$\frac{d\tau}{du} + f + \beta\frac{(1-u^2)}{\tau} = 0 \tag{20}$$

Though the highest order derivative in the equation has been reduced, Eq.(20) still contains f in explicit form. To eliminate f we take the derivative of Eq.(20) with respect to u and apply the chain rule, which results in a modified Falkner-Skan equation:

$$\frac{d^2\tau}{du^2} + (1-2\beta)\frac{u}{\tau} + \beta(1-u^2)\frac{d}{du}\left(\frac{1}{\tau}\right) = 0 \quad \text{for} \quad 0 \leq u \leq 1 \tag{21}$$

Note that the third order nonlinear ordinary differential equation on an unbounded problem domain has been transformed through a change of variables into a more manageable second order differential equation on a unit domain. The boundary conditions in Eqs. (16) and (17) are rewritten in terms of the transformed variables:

$$\begin{aligned} u &= 0 \quad \text{for} \quad \eta = 0 \quad \text{and} \quad u \rightarrow 1 \quad \text{for} \quad \eta \rightarrow \infty \\ \tau &= 0 \quad \text{for} \quad u = 1 \end{aligned}$$

Applying Eq.(12) to Eq.(21), the weak form is obtained,

$$\left(F_k \ , \ \frac{d^2\tau}{du^2}\right) = -(1-2\beta)\left(F_k \ , \ \frac{u}{\tau}\right) - \beta\left(F_k \ , \ (1-u^2)\frac{d}{du}\left(\frac{1}{\tau}\right)\right), \quad k = 1, 2, ..., N \tag{22}$$

where, for example,

$$\left(F_k \ , \ \frac{d^2\tau}{du^2}\right) = \int_0^1 F_k \frac{d^2\tau}{du^2} \ du$$

Integration by parts must be applied to the left hand side of Eq.(22) to reduce the order of the derivative of τ. This results in:

$$\left(\frac{dF_k}{du} \ , \ \frac{d\tau}{du}\right) = (1-2\beta)\left(F_k \ , \ \frac{u}{\tau}\right) + \beta\left(F_k \ , \ (1-u^2)\frac{d}{du}\left(\frac{1}{\tau}\right)\right) + \left[F_k \frac{d\tau}{du}\right]_0^1 \tag{23}$$

Let us now introduce the approximate solution τ_a. As mentioned in section 3.1, the problem domain for the acting independent variable u is discretized into N knots,

$$0 = u_1 < u_2 < ... < u_N = 1$$

plus two auxiliary knots outside the problem domain. Using the input and bias weights, the piecewise linear activation functions of the FFANN are constrained to form hat functions centered at each knot in the problem domain. However, instead of approximating τ with a simple basis expansion as in Eq.(11), we use

$$\tau_a = (1-u)\sum_{i=1}^{N} \Phi_i \ b1_i \tag{24}$$

Note that the approximation of τ is zero at $u = 1$, thus automatically satisfying the transformed boundary condition. Rather than use the reciprocal of Eq.(24) for the approximation of $1/\tau$ we will utilize the product approximation method [19] and introduce a separate approximation:

$$\frac{1}{\tau_a} = \frac{1}{(1-u)}\sum_{i=1}^{N} \Phi_i \ b2_i \tag{25}$$

The basis expansion for $1/\tau$ is automatically unbounded at $u = 1$ and like Eq.(24), the basis coefficients are not needed to satisfy the boundary condition.

The relation between the set of basis coefficients can be determined by equating $1/\tau_a$ and the reciprocal of τ_a at each knot u_j for $j = 1, ..., N$:

$$\frac{1}{\tau_a(u_j)} = \frac{1}{(1-u_j)}\sum_{i=1}^{N} \Phi_i(u_j) \ b2_i = \frac{1}{(1-u_j)\sum_{i=1}^{N} \Phi_i(u_j) \ b1_i} \tag{26}$$

For the hat function

$$\Phi_i(u_j) = 1, \ j = i \quad \text{and} \quad \Phi_i(u_j) = 0, \ j \neq i \tag{27}$$

Therefore, after replacing the dummy subscript j with i in Eq.(26) we have,

$$b2_i = \frac{1}{b1_i} \quad \text{for} \quad i = 1, ..., N \tag{28}$$

which provides the additional set of equations required.

The product approximation method is also applicable to problems with more than one dependent variable by assigning each multiplicative group of variables separate basis expansions. The application of the product approximation method does not adversely affect the accuracy of the problem solution but does increase computational efficiency while greatly simplifying the treatment of nonlinear problems by the weighted residual method [16].

Substituting the approximations and their derivatives into the weak formulation results in

$$\sum_{i=1}^{N}\left(\frac{dF_k}{du}\ ,\ (1-u)\frac{d\Phi_i}{du}-\Phi_i\right)b1_i=(1-2\beta)\sum_{i=1}^{N}\left(F_k\ ,\ \frac{u}{1-u}\Phi_i\right)b2_i$$
$$+\beta\sum_{i=1}^{N}\left(F_k\ ,\ \frac{(1-u^2)}{(1-u)^2}\left((1-u)\frac{d\Phi_i}{du}+\Phi_i\right)\right)b2_i+\left[F_k\frac{d\tau}{du}\right]_0^1 \tag{29}$$

Note that since the term $1/(1-u)$ is present it will cause difficulty in the evaluation of the inner products unless the term is somehow eliminated. This can be done by applying the Petrov-Galerkin technique to Eq.(29). Say that

$$F_k=(1-u)\Phi_k$$

Therefore, Eq.(29) becomes

$$\sum_{i=1}^{N}\left((1-u)\frac{d\Phi_k}{du}-\Phi_k\ ,\ (1-u)\frac{d\Phi_i}{du}-\Phi_i\right)b1_i=$$
$$(1-2\beta)\sum_{i=1}^{N}\left(\Phi_k\ ,\ u\Phi_i\right)b2_i+\ \beta\sum_{i=1}^{N}\left(\Phi_k\ ,\ (1+u)\left((1-u)\frac{d\Phi_i}{du}+\Phi_i\right)\right)b2_i$$
$$+\left[(1-u)\Phi_k\frac{d\tau}{du}\right]_0^1 \tag{30}$$

If we define the following matrices,

$$\begin{aligned}
\mathbf{M1}=M1_{ki} &= \left((1-u)\frac{d\Phi_k}{du}-\Phi_k\ ,\ (1-u)\frac{d\Phi_i}{du}-\Phi_i\right)\\
\mathbf{M2}=M2_{ki} &= \left(\Phi_k\ ,\ u\Phi_i\right)\\
\mathbf{M3}=M3_{ki} &= \left(\Phi_k\ ,\ (1+u)\left((1-u)\frac{d\Phi_i}{du}+\Phi_i\right)\right)
\end{aligned} \tag{31}$$

then the algebraic system formed by Eq.(30) can be written as

$$\sum_{i=1}^{N}M1_{ki}\ b1_i=(1-2\beta)\sum_{i=1}^{N}M2_{ki}\ b2_i+\beta\sum_{i=1}^{N}M3_{ki}\ b2_i+\left[(1-u)\Phi_k\frac{d\tau}{du}\right]_0^1 \tag{32}$$

for $k=1,...,N$ with

$$b2_i=\frac{1}{b1_i}\quad \text{for}\quad i=1,...,N \tag{33}$$

The nature of the hat functions as bases reveals that matrices **M1**, **M2**, and **M3** are positive definite and tridiagonal and therefore of order N ($O(N)$) in storage [16]. Let us now consider the flux term

$$\left[(1-u)\Phi_k\frac{d\tau}{du}\right]_0^1=\left[(1-u)\Phi_k\frac{d\tau}{du}\right]_1-\left[(1-u)\Phi_k\frac{d\tau}{du}\right]_0$$

Since $u = 0 = u_1$ and $u = 1 = u_N$, the application of Eq.(27) shows

$$\left[(1-u)\Phi_k \frac{d\tau}{du}\right]_0 = \Phi_k(0)\frac{d\tau}{du}(0) = 0 \quad \text{for} \quad k = 2, ..., N \tag{34}$$

but

$$\left[(1-u)\Phi_k \frac{d\tau}{du}\right]_0 = \frac{d\tau}{du}(0) \quad \text{for} \quad k = 1 \tag{35}$$

and

$$\left[(1-u)\Phi_k \frac{d\tau}{du}\right]_1 = 0 \quad \text{for} \quad k = 1, ..., N \tag{36}$$

Evaluating Eq.(20) at $u = 0$ and recalling that $f(0) = 0$,

$$\frac{d\tau}{du}(0) = -\left[f + \beta\frac{(1-u^2)}{\tau}\right]_{u=0} = -\frac{\beta}{\tau(0)}$$

Therefore, the flux term can be written as

$$\left[(1-u)\Phi_k \frac{d\tau}{du}\right]_0^1 = \frac{\beta}{\tau(0)} = \beta\sum_{i=1}^{N}\Phi_i(0)\ b2_i = \beta\ b2_1 \quad \text{for} \quad k = 1 \tag{37}$$

Let us define a matrix **M4** such that

$$M4_{ki} = (1-2\beta)M2_{ki} + \beta M3_{ki} + \beta \quad \text{for} \quad k = 1 \quad \text{with} \quad i = 1, ..., N$$

and

$$M4_{ki} = (1-2\beta)M2_{ki} + \beta M3_{ki} \quad \text{for} \quad k = 2, ..., N \quad \text{with} \quad i = 1, ..., N \tag{38}$$

where **M4** is positive definite and tridiagonal. The use of Eqs.(32), (33), (37), and (38) results in the algebraic system:

$$\sum_{i=1}^{N} M1_{ki}\ b1_i - \sum_{i=1}^{N} M4_{ki}\ \frac{1}{b1_i} = 0 \quad \text{for} \quad k = 1, .., N \tag{39}$$

The approximation of the dependent variable τ will be complete with the solution of Eq.(39).

5.1.1 Solution Procedure

The system described by Eq.(39) can be solved by any of the standard nonlinear equation solvers. We resort to a calculus method, specifically Powell's method. This has been implemented using the routine DNEQNJ from the IMSL numerical library [21].

The algorithm minimizes the scalar function:

$$\epsilon\ (b1_1, b1_2, ..., b1_N) = \sum_{k=1}^{N}(e_k)^2$$

where

$$e_k = \sum_{i=1}^{N} M1_{ki}\ b1_i - \sum_{i=1}^{N} M4_{ki}\ \frac{1}{b1_i}$$

As with any multidimensional search algorithm, Powell's method requires the Jacobian matrix of the system. In this example problem the Jacobian **J** was supplied analytically to the algorithm to increase computational efficiency. The Jacobian is calculated in the following manner:

$$\mathbf{J} = J_{kj} = \frac{de_k}{db1_j} = M1_{kj} + M4_{kj}\left(\frac{1}{b1_j}\right)^2 \tag{40}$$

Note that like the matrices **M1** and **M4**, the Jacobian is also positive definite and tridiagonal.

The solution procedure can be summarized as follows:

1. Numerically evaluate the integrals of Eq.(31) to form the matrices **M1**, **M2**, and **M3**.
2. Select a value for β and compute the components of matrix **M4**.
3. Select initial values for $b1_i$. For this example $b1_i = 1$ for $i = 1, ..., N$.
4. Apply the Powell's method algorithm to minimize ϵ, evaluating J_{kj} and e_k at every iteration. Convergence is achieved when the change in e_k between successive iterations is less than some predetermined value.

5.1.2 Post-Processing of Solution

With the values for $b1_i$ known, the expansion for τ given in Eq.(24) is complete. The construction of basis expansions for the unknowns f, η, and $d^2f/d\eta^2$, using τ and u, is relatively straightforward.

Assume that the approximation of f may be written as

$$f_a(\eta) = f_a(u) = \sum_{k=1}^{N} \Phi_k(u)\ w1_k \tag{41}$$

where N and Φ are the same as those used for the approximation of τ.

Using the coordinate transformation of Eq.(19) we show:

$$u = \frac{df}{d\eta} \quad \text{so} \quad df = u\ d\eta \quad \text{and} \quad \tau = \frac{du}{d\eta} \quad \text{so} \quad d\eta = \frac{1}{\tau}\ du$$

Therefore we may write

$$df = \left(\frac{u}{\tau}\right)\ du \tag{42}$$

Using the knots we can integrate Eq.(42)

$$\int_{u_1}^{u_k} df = f(u_k) - f(u_1) = \int_{u_1}^{u_k} \left(\frac{u}{\tau}\right)\ du = f_a(u_k) - f_a(u_1) \tag{43}$$

From the boundary conditions $f(u_1 = 0) = 0 = f_a(0)$ and from the properties of the hat functions $f_a(u_k) = w1_k$, Eq.(43) becomes

$$w1_k = \int_0^{u_k} \left(\frac{u}{\tau}\right) du \tag{44}$$

After substituting the appropriate expansion,

$$w1_k = \int_0^{u_k} \frac{u}{(1-u)} \sum_{i=1}^{N} \Phi_i\ b2_i\ du = \sum_{i=1}^{N} \left(\int_0^{u_k} \frac{u}{(1-u)} \Phi_i\ du\right) b2_i = \sum_{i=1}^{N} G1_{ki}\ b2_i \quad \text{for} \quad k = 1, 2, ..., N-1 \tag{45}$$

where

$$G1_{ki} = \int_0^{u_k} \frac{u}{(1-u)} \Phi_i\ du \quad \text{for} \quad k = 1, 2, ..., N-1 \tag{46}$$

If at $k = N$ we use the knot $u_N = 1$, then the value of $w1_N$ would become infinite. Therefore, for our example we set

$$w1_N = \sum_{i=1}^{N} \left(\int_0^{0.99} \frac{u}{1-u} \Phi_i \; du \right) b2_i = \sum_{i=1}^{N} G1_{Ni} \; b2_i \tag{47}$$

With the evaluation of $w1_k$ the basis expansion for f is complete.

To approximate the original independent variable η in terms of the variable u we write

$$\eta_a = \eta_a(u) = \sum_{k=1}^{N} \Phi_k(u) \; w2_k \tag{48}$$

Recall

$$\tau = \frac{du}{d\eta} \quad \text{so} \quad d\eta = \left(\frac{1}{\tau}\right) du$$

Integrating $d\eta$ and acknowledging that $\eta(0) = 0 = \eta_a(0)$, we have

$$\eta_a(u_k) = \int_0^{u_k} \left(\frac{1}{\tau}\right) \; du = w2_k$$

Therefore,

$$w2_k = \sum_{i=1}^{N} G2_{ki} \; b2_i \quad \text{for} \quad k = 1, 2, ..., N \tag{49}$$

where

$$G2_{ki} = \int_0^{u_k} \frac{1}{(1-u)} \Phi_i \; du \quad \text{for} \quad k = 1, 2, ..., N-1 \quad \text{and} \quad G2_{Ni} = \int_0^{0.99} \frac{1}{(1-u)} \Phi_i \; du \tag{50}$$

It remains to approximate the second derivative of f with respect to η in terms of the acting independent variable u. We write

$$\frac{d^2 f_a}{d\eta^2}(u) = \sum_{k=1}^{N} \Phi_k(u) \; w3_k \tag{51}$$

To determine $w3_k$, equate the approximation of τ with the approximation of $d^2 f/d\eta^2$ at the knots in the problem domain.

$$\tau_a(u_k) = (1 - u_k) \sum_{i=1}^{N} \Phi_i(u_k) \; b1_i = \frac{d^2 f_a}{d\eta^2}(u_k) = \sum_{k=1}^{N} \Phi_i(u_k) \; w3_k$$

This results in

$$w3_k = (1 - u_k) \; b1_k \tag{52}$$

Since the discretization is the same for all of the basis expansions, application of Eq.(9) to $w1_k$, $w2_k$, and $w3_k$ will allow the construction of the single input, multiple output feedforward network of Fig. 8. For a given value of $df/d\eta$, the network will output values for η, f, and $d^2 f/d\eta^2$.

5.1.3 Results

For Example 1, the Falkner-Skan equation, the input, bias, and output weights for the network were computed via a conventional Fortran code. All computations were done in double precision. As mentioned previously, the IMSL subroutine DNEQNJ was used to solve the system of nonlinear algebraic equations. The subroutine DQDAG was used for numerical integration of the components of the coefficient matrix.

The Falkner-Skan equation, though relatively simple in appearance, has no analytic solution at present. However, Eq.(15) does yield to numerical methods. In order to give some idea of the accuracy of the constructed FFANN, a solution for the nonlinear differential equation was sought using a variable-step fourth/fifth order accurate Runge-Kutta algorithm and an iterative shooting method.

The feedforward network was constructed using a single hidden layer of 42 processing elements for the case $\beta = -0.15$ and 22 hidden processing elements for $\beta = 0$ and 0.5. This corresponds to an even spacing, within the problem domain, of 21 and 11 knots, respectively. The variable step size Runge-Kutta algorithm marched from $\eta = 0$ to $\eta = 5$, in 89 steps for $\beta = -0.15$, 103 steps for $\beta = 0$, and 128 steps for $\beta = 0.5$. The Runge-Kutta solution satisfied the boundary conditions with an error less than 1.0×10^{-6} for all cases.

Figures 9 - 11 compare the output of the constructed feedforward neural network with the Runge-Kutta solution (solid line) at 10 equispaced sample points ($S = 10$) for $\beta = -0.15$, $\beta = 0$, and $\beta = 0.5$. The primes denote differentiation of the dependent variables with respect to η. The number of iterations (I) required for solution of the basis coefficients were: $I = 9$ for $\beta = -0.15$ with $\epsilon = 2.336 \times 10^{-13}$, $I = 7$ for $\beta = 0$ with $\epsilon = 4.611 \times 10^{-18}$, and $I = 8$ for $\beta = 0.5$ with $\epsilon = 8.596 \times 10^{-16}$. The values of the root mean square of the difference (RMS) between the Runge-Kutta and the network output are given in the individual figure descriptions.

The program ran on a standard SUN Microsystems Sparcstation 2 with a run time on the order of two seconds for the test cases of Figs. 9 - 11.

5.1.4 Convergence

The error bounds derived for the Petrov-Galerkin method lead us to expect that the L_2 norm of the error will be bounded by a quadratic term in the spacing of the knots (grid spacing) when approximating a linear ordinary differential equation [22]. This standard result from numerical analysis was confirmed in reference [14] for network solution of linear ordinary differential equations. When approximating a nonlinear differential equation, this quadratic convergence rate degrades and cannot be predicted easily.

For uniform grid spacing

$$u_{i+1} - u_i = u_i - u_{i-1} = h = \frac{1}{(N-1)}$$

So, for the error E in the variable $d^2f/d\eta^2$ the L_2 norm of E is defined as

$$||E|| = \sqrt{\int (E)^2 du} = \sqrt{\int \left(\frac{d^2 f}{d\eta^2} - \frac{d^2 f_a}{d\eta^2} \right)^2 du} \tag{53}$$

In practice, and for this paper, the discrete L_2 norm of the error is used so as to avoid having to perform the actual integration in Eq.(53):

$$||E|| = \sqrt{h \sum_{i=1}^{N} \left(\frac{d^2 f}{d\eta^2}(u_i) - \frac{d^2 f_a}{d\eta^2}(u_i) \right)^2}$$

where the integral has been approximated by a scaled summation. We assume that the L_2 norm of the error is directly proportional to some order A of the grid spacing h and write

$$||E|| \approx h^A \quad \text{or} \quad ||E|| \leq Ch^A \tag{54}$$

where C is a constant and where A is some real number. The exponent A should be $\leq$ 2 for nonlinear differential equations.

The variable $d^2f/d\eta^2$ is chosen for the convergence study since numerical experimentation has shown this function to be well-behaved and its convergence rate should provide an upper bound on the convergence of the other variables.

Since there is no exact solution to the Falkner-Skan equation, the convergence rate has been tested by successive grid refinement through the following steps:

1. Select an initial spacing h_0 and set $k = 0$.
2. Solve the equation on a grid spacing of $h_k = h_0/2^k$ and $h_{(k+1)} = h_0/2^{(k+1)}$.
3. Find the discrete L_2 norm of the difference between the solutions at common knots.
4. Increment k and return to Step 2.

The purpose of this procedure is to measure the rate at which the approximation is approaching some "exact" solution at discrete points.

If we take the log of $||E||$ in Eq.(54) then

$$\log(||E||) \leq \log(C) + A\log(h) \tag{55}$$

Considering the form of Eq.(55), it would be reasonable to expect that a log plot of the L_2 norm of the error versus the log of the grid spacing will yield a straight line of slope A.

A suitable convergence plot of this type is shown in Fig. 12 for the three values of β. The dashed line represents a purely linear convergence rate. The size of the problem mesh was varied from $h = 0.1$ to $h = 0.0125$. The value of the exponent A was determined for $\beta = -0.15$, 0, and 0.5 to be 0.892, 0.679, and 0.598 respectively. Note the highest convergence rate corresponded to the negative valued β.

Although the value of the error cannot be evaluated before the solution of Eq.(39), the error is bounded. In addition Fig. 12 shows that the error can be controlled by altering the number of processing elements in the hidden layer of the network.

5.2 Example 2: Coupled Nonlinear Ordinary Differential Equations

For the second numerical example we constructed a single input, multiple output FFANN to approximate the solution, and derivatives, of the following coupled system of nonlinear third and second order ordinary differential equations:

$$\frac{d^3f}{d\eta^3} - (A_1 f + A_2 g)\frac{d^2f}{d\eta^2} = 0 \tag{56}$$

$$\frac{d^3g}{d\eta^3} - (A_1 f + A_2 g)\frac{d^2g}{d\eta^2} - A_3\frac{df}{d\eta}\frac{dg}{d\eta} - A_4\left(\frac{dg}{d\eta}\right)^2 - A_5\left(\frac{df}{d\eta}\right)^2 - A_6\lambda - A_7 = 0 \tag{57}$$

$$\frac{d^2\lambda}{d\eta^2} - Pr\,(A_1 f + A_2 g)\frac{d\lambda}{d\eta} - A_8\left(\frac{d^2f}{d\eta^2}\right)^2 - A_8\,(A_1 f + A_2 g)\frac{df}{d\eta}\frac{d^2f}{d\eta^2} = 0 \tag{58}$$

where $0 \leq \eta$. The coefficients A_1, ..., A_7 are constants and represent combinations of various physical parameters and the coefficient A_8 is a function of the parameter Pr. The associated boundary conditions for Eqs.(56)-(58) are

$$f = 0\ ,\ \frac{df}{d\eta} = 0\ ,\ g = 0\ ,\ \frac{dg}{d\eta} = 0\ ,\ \lambda = 0 \ \text{ for } \ \eta = 0$$
$$\text{and} \quad \frac{df}{d\eta} \to 1 \quad \text{as} \quad \eta \to \infty \tag{59}$$

Additionally from the boundary conditions, since

$$\frac{df}{d\eta} \to 1 \quad \text{as} \quad \eta \to \infty \quad \text{then} \quad \frac{d^2f}{d\eta^2} \to 0 \quad \text{as} \quad \eta \to \infty \tag{60}$$

As in the previous example, Eqs.(56) - (58) are of practical interest in the field of fluid mechanics. The ordinary differential equations are derived from a similarity transformation of the system of partial differential equations that model both a three-dimensional momentum and thermal boundary layer about an inclined right circular cone in supersonic flow [23] - [25]. If Eqs.(56) - (58) are to model physically meaningful situations, $df/d\eta$ must increase monotonically with η. This may be accomplished by restricting the values of the physical parameters, which is equivalent to restricting A_1 through A_8. For a cone with a half angle of 15°, in air with a freestream Mach number of 7, and freestream Reynolds number of 5×10^5, we have the following numerical values of A_1 through A_7 for an angle of attack of 10° :

$$\begin{aligned} A_1 = -0.960428 \ , \quad A_2 = -0.395063 \ , \quad A_3 = 0.640285 \ , \quad A_4 = 0.395063 \\ A_5 = 0.946661 \ , \quad A_6 = -0.665229 \ , \quad A_7 = -0.665229 \end{aligned} \tag{61}$$

We must also employ the constraint $0.7 \leq Pr \leq 1.0$, which corresponds to $0 \leq A_8 \leq 0.86$. A detailed discussion of the derivation of Eqs.(56) - (58), the necessity of monotonically increasing $df/d\eta$, and the evaluation of coefficients A_1 through A_8 can be found in reference [26].

To construct the FFANN that models the coupled system, we must transform the governing equations into a more manageable form. Our approach to solving Eqs.(56) - (58) will be quite similar to the approach used for the Falkner-Skan problem. Since $df/d\eta$ varies monotonically with η for the previously stated restrictions of A_1 through A_8, the derivative can be used as the acting independent variable. We introduce the following change of variables:

$$\begin{aligned} u = \frac{df}{d\eta} \quad \text{with} \quad 0 \leq u \leq 1 \ , \ \tau = \frac{d^2 f}{d\eta^2} \\ r = \frac{dg}{d\eta} \ , \quad s = \frac{d\lambda}{d\eta} \ , \quad p = \frac{d\tau}{d\eta} \ , \quad q = \frac{dr}{d\eta} \end{aligned} \tag{62}$$

In our transformation of the coupled equations we must eliminate explicit dependence on f and g. Rearrangement of Eq.(56) gives

$$(A_1 f + A_2 g) = \frac{d^3 f}{d\eta^3}\left(\frac{d^2 f}{d\eta^2}\right)^{-1} \tag{63}$$

Substitution of Eq.(63) into the derivative of Eq.(56), and directly into Eq.(57) and Eq.(58) results in, respectively,

$$\frac{d^4 f}{d\eta^4} = \left(A_1 \frac{df}{d\eta} + A_2 \frac{dg}{d\eta}\right)\frac{d^2 f}{d\eta^2} + \frac{d^3 f}{d\eta^3}\left(\frac{d^2 f}{d\eta^2}\right)^{-1}\frac{d^3 f}{d\eta^3} \tag{64}$$

$$\frac{d^3 g}{d\eta^3} = \frac{d^3 f}{d\eta^3}\left(\frac{d^2 f}{d\eta^2}\right)^{-1}\frac{d^2 g}{d\eta^2} + A_3 \frac{df}{d\eta}\ \frac{dg}{d\eta} + A_4\left(\frac{dg}{d\eta}\right)^2 + A_5\left(\frac{df}{d\eta}\right)^2 + A_6\lambda + A_7 \tag{65}$$

and

$$\frac{d^2\lambda}{d\eta^2} = Pr\frac{d^3 f}{d\eta^3}\left(\frac{d^2 f}{d\eta^2}\right)^{-1}\frac{d\lambda}{d\eta} + A_8\left(\frac{d^2 f}{d\eta^2}\right)^2 + A_8\frac{d^3 f}{d\eta^3}\ \frac{df}{d\eta} \tag{66}$$

With substitution of the transformed dependent variables and use of the chain rule we have for Eqs.(64) - (66)

$$\frac{dp}{du} = A_1 u + A_2 r + \left(\frac{p}{\tau}\right)^2 \tag{67}$$

$$\frac{dq}{du} = \left(\frac{p}{\tau}\right)\left(\frac{q}{\tau}\right) + A_3 u\left(\frac{r}{\tau}\right) + A_4\left(\frac{r^2}{\tau}\right) + A_5 u^2\left(\frac{1}{\tau}\right) + A_6\left(\frac{\lambda}{\tau}\right) + A_7\left(\frac{1}{\tau}\right) \tag{68}$$

and

$$\frac{ds}{du} = Pr\left(\frac{p}{\tau}\right)\left(\frac{s}{\tau}\right) + A_8\tau + A_8 u\left(\frac{p}{\tau}\right) \tag{69}$$

Use of the chain rule on the variables of Eq.(62) completes the transformation by reducing the original system of three coupled nonlinear mixed second and third order differential equations to a system of six first order equations:

$$\frac{dp}{du} - A_1 u - A_2 r - \left(\frac{p}{\tau}\right)^2 = 0$$

$$\frac{dq}{du} - \left(\frac{p}{\tau}\right)\left(\frac{q}{\tau}\right) - A_3 u\left(\frac{r}{\tau}\right) - A_4\left(\frac{r^2}{\tau}\right) - A_5 u^2\left(\frac{1}{\tau}\right) - A_6\left(\frac{\lambda}{\tau}\right) - A_7\left(\frac{1}{\tau}\right) = 0$$

$$\frac{ds}{du} - Pr\left(\frac{p}{\tau}\right)\left(\frac{s}{\tau}\right) - A_8\tau - A_8\ u\left(\frac{p}{\tau}\right) = 0$$

$$\frac{d\tau}{du} - \left(\frac{p}{\tau}\right) = 0, \qquad \frac{dr}{du} - \left(\frac{q}{\tau}\right) = 0, \qquad \frac{d\lambda}{du} - \left(\frac{s}{\tau}\right) = 0 \tag{70}$$

It now remains to find the new boundary conditions for the transformed system. From the evaluation of Eqs.(56)-(58) at $\eta = 0$ and $\eta \to \infty$, coupled with Eq.(60) and Eq.(62), it becomes apparent that

$$\begin{aligned} p = r = \lambda = 0 \quad &\text{for} \quad u = 0 \\ \tau = q = s = 0 \quad &\text{for} \quad u = 1 \end{aligned} \tag{71}$$

We can now use the basis expansions to approximate the transformed dependent variables. Utilizing the product approximation method introduced in Example 1, the following approximations can be written for the dependent transformation variables

$$\tau_a = (1-u)\sum_{i=1}^{N}\Phi_i\ b1_i\ , \quad r_a = \sum_{i=1}^{N}\Phi_i\ b2_i\ , \quad \lambda_a = \sum_{i=1}^{N}\Phi_i\ b3_i,$$
$$p_a = \sum_{i=1}^{N}\Phi_i\ b4_i\ , \quad q_a = (1-u)\sum_{i=1}^{N}\Phi_i\ b5_i\ , \quad s_a = (1-u)\sum_{i=1}^{N}\Phi_i\ b6_i$$

and for the multiplicative groups,

$$\frac{p_a}{\tau_a} = \frac{1}{(1-u)}\sum_{i=1}^{N}\Phi_i\ b7_i\ , \quad \frac{q_a}{\tau_a} = \frac{1}{(1-u)}\sum_{i=1}^{N}\Phi_i\ b8_i\ , \quad \frac{s_a}{\tau_a} = \frac{1}{(1-u)}\sum_{i=1}^{N}\Phi_i\ b9_i$$
$$\left(\frac{p_a}{\tau_a}\right)^2 = \frac{1}{(1-u)^2}\sum_{i=1}^{N}\Phi_i\ b10_i\ , \quad \left(\frac{p_a}{\tau_a}\right)\left(\frac{q_a}{\tau_a}\right) = \frac{1}{(1-u)^2}\sum_{i=1}^{N}\Phi_i\ b11_i$$
$$\frac{{r_a}^2}{\tau_a} = \frac{1}{(1-u)}\sum_{i=1}^{N}\Phi_i\ b12_i\ , \quad \frac{1}{\tau_a} = \frac{1}{(1-u)}\sum_{i=1}^{N}\Phi_i\ b13_i\ , \quad \frac{\lambda_a}{\tau_a} = \frac{1}{(1-u)}\sum_{i=1}^{N}\Phi_i\ b14_i$$
$$\left(\frac{p_a}{\tau_a}\right)\left(\frac{s_a}{\tau_a}\right) = \frac{1}{(1-u)^2}\sum_{i=1}^{N}\Phi_i\ b15_i\ , \quad \frac{r_a}{\tau_a} = \frac{1}{(1-u)}\sum_{i=1}^{N}\Phi_i\ b16_i \tag{72}$$

where N is the number of knots in the problem domain. Note that the multiplication and division of the basis expansions by $(1-u)$ help enforce boundary conditions.

Of the basis coefficients, only $b1_i$ through $b6_i$ are linearly independent. The relation between the remaining set of basis coefficients can be determined by making use of the same equivalence arguments made in Example 1 (Eqs. (26) - (28)):

$$b7_i = b4_i\ b13_i\ , \quad b8_i = b5_i\ b13_i\ , \quad b9_i = b6_i\ b13_i$$
$$b10_i = (b4_i\ b13_i)^2\ , \quad b11_i = b4_i\ b5_i\ (b13_i)^2\ , \quad b12_i = (b2_i)^2\ b13_i$$
$$b13_i = \frac{1}{b1_i}\ , \quad b14_i = b3_i\ b13_i\ , \quad b15_i = b4_i\ b6_i(b13_i)^2\ , \quad b16_i = b2_i\ b13_i \tag{73}$$

for $i = 1, ..., N$.

To formulate the algebraic systems whose solution will yield the expansion coefficients of the transformed problem Eq.(70), we apply the Petrov-Galerkin method with $F_k(u) = (1-u)^2\Phi_k(u)$:

$$\sum_{i=1}^{N} M1_{ki}\ b4_i - A_1 v_k - A_2 \sum_{i=1}^{N} M3_{ki}\ b2_i - \sum_{i=1}^{N} M4_{ki}\ b10_i = 0$$

$$\sum_{i=1}^{N} M7_{ki}\ b5_i - \sum_{i=1}^{N} M2_{ki}(A_4\ b12_i + A_7\ b13_i + A_6\ b14_i) - \sum_{i=1}^{N} M4_{ki}\ b11_i - A_3 \sum_{i=1}^{N} M5_{ki}\ b16_i - A_5 \sum_{i=1}^{N} M6_{ki}\ b13_i = 0$$

$$\sum_{i=1}^{N} M7_{ki}\ b6_i - Pr\sum_{i=1}^{N} M4_{ki}\ b15_i - A_8\left(\sum_{i=1}^{N} M8_{ki}\ b1_i - \sum_{i=1}^{N} M5_{ki}\ b7_i\right) = 0$$

$$\sum_{i=1}^{N} M7_{ki}\ b1_i - M2_{ki}\ b7_i = 0$$

$$\sum_{i=1}^{N} M1_{ki}\ b2_i - M2_{ki}\ b8_i = 0$$

$$\sum_{i=1}^{N} M1_{ki}\ b3_i - M2_{ki}\ b9_i = 0 \tag{74}$$

where the matrices **M1, ..., M8** and the vector **v** are defined as

$$\begin{aligned} M1_{ki} &= \left((1-u)^2\Phi_k\ ,\ \frac{d\Phi_i}{du}\right)\ ,\ M2_{ki} = ((1-u)\Phi_k\ ,\ \Phi_i)\ ,\ M3_{ki} = \left((1-u)^2\Phi_k\ ,\ \Phi_i\right) \\ M4_{ki} &= (\Phi_k\ ,\ \Phi_i)\ ,\ M5_{ki} = (u(1-u)\Phi_k\ ,\ \Phi_i)\ ,\ M6_{ki} = \left(u^2(1-u)\Phi_k, \Phi_i\right) \\ M7_{ki} &= \left((1-u)^2\Phi_k\ ,\ \left((1-u)\frac{d\Phi_i}{du} - \Phi_i\right)\right)\ ,\ M8_{ki} = \left((1-u)^3\Phi_k, \Phi_i\right) \\ v_k &= \left((1-u)^2\Phi_k, u\right) \end{aligned} \tag{75}$$

The boundary conditions of Eq.(71) give

$$\begin{aligned} b2_1 = b3_1 = b4_1 = 0 \quad (u=0) \\ b1_N = b5_N = b6_N = 1 \quad (u=1) \end{aligned} \tag{76}$$

As in the Falkner-Skan equation example, these constraints are incorporated into the system of algebraic equations by the replacement of the appropriate rows of the matrices.

5.2.1 Solution Procedure

The six sets of equations in Eq.(74) must be solved for the coefficients $b1_i, b2_i, ..., b6_i$. As in Example 1, we again resort to Powell's method to solve the nonlinear equations. In this case the algorithm minimizes the following scalar function:

$$\epsilon\ (b1_1, ..., b1_N,\ b2_1, ..., b2_N,\ ...,\ b6_1, ..., b6_N) = \sum_{k=1}^{N}\left((e_k^{(1)})^2\ +\ (e_k^{(2)})^2 +\ ...\ + (e_k^{(6)})^2\right)$$

where

$$
\begin{aligned}
e_k^{(1)} &= \sum_{i=1}^{N} M1_{ki}\ b4_i - A_1\ v_k - A_2 \sum_{i=1}^{N} M3_{ki}\ b2_i - \sum_{i=1}^{N} M4_{ki}\ b10_i \\
e_k^{(2)} &= \sum_{i=1}^{N} M7_{ki}\ b5_i - \sum_{i=1}^{N} M2_{ki}\ (A_4\ b12_i + A_7\ b13_i + A_6\ b14_i) - \sum_{i=1}^{N} M4_{ki}\ b11_i - \\
&\qquad A_3 \sum_{i=1}^{N} M5_{ki}\ b16_i - A_5 \sum_{i=1}^{N} M6_{ki}\ b13_i \\
e_k^{(3)} &= \sum_{i=1}^{N} M7_{ki}\ b6_i - Pr \sum_{i=1}^{N} M4_{ki}\ b15_i - A_8 (\sum_{i=1}^{N} M8_{ki}\ b1_i - \sum_{i=1}^{N} M5_{ki}\ b7_i) \\
e_k^{(4)} &= \sum_{i=1}^{N} M7_{ki}\ b1_i - M2_{ki}\ b7_i \\
e_k^{(5)} &= \sum_{i=1}^{N} M1_{ki}\ b2_i - M2_{ki}\ b8_i \\
e_k^{(6)} &= \sum_{i=1}^{N} M1_{ki}\ b3_i - M2_{ki}\ b9_i
\end{aligned}
\tag{77}
$$

As in the previous example, the Jacobian ($\mathbf{J}$) is evaluated analytically for increased computational efficiency. In this case the Jacobian is of size $(6N) \times (6N)$. It is composed of 36 partitions, each of which can be considered an $N \times N$ sub-Jacobian. For example, if a partition is defined as

$$
J^{(AB)} = \frac{de_k^{(A)}}{dbB_j}, \quad \text{where} \quad A, B = 1, ..., 6
$$

then

$$
\mathbf{J} = \begin{bmatrix} J^{(11)} & \cdots & J^{(16)} \\ \vdots & \ddots & \vdots \\ J^{(61)} & \cdots & J^{(66)} \end{bmatrix}
$$

Each partition of the Jacobian matrix is tridiagonal, owing to the properties of the hat function. As a result, the full Jacobian matrix is sparse.

5.2.2 Post-Processing of Solution

With the coefficients $b1_i, b2_i, ..., b6_i$ known, then $b7_i, b8_i, ..., b16_i$ can be calculated. The approximations of the dependent transformation variables are complete. Using $u = df/d\eta$ as the input variable to the network, it now remains to determine the output weights for the original variables η, f, $d^2f/d\eta^2$, g, $dg/d\eta$, $d^2g/d\eta^2$, λ, and $d\lambda/d\eta$.

We write the basis expansions for the approximations as follows:

$$
\begin{aligned}
\eta_a = \sum_{i=1}^{N} \Phi_i(u)\ w1_i\ , \quad f_a = \sum_{i=1}^{N} \Phi_i(u)\ w2_i\ , \quad \frac{d^2 f_a}{d\eta^2} = \sum_{i=1}^{N} \Phi_i(u)\ w3_i \\
g_a = \sum_{i=1}^{N} \Phi_i(u)\ w4_i\ , \quad \frac{dg_a}{d\eta} = \sum_{i=1}^{N} \Phi_i(u)\ w5_i,\ , \quad \frac{d^2 g_a}{d\eta^2} = \sum_{i=1}^{N} \Phi_i(u)\ w6_i \\
\lambda_a = \sum_{i=1}^{N} \Phi_i(u)\ w7_i\ , \quad \frac{d\lambda_a}{d\eta} = \sum_{i=1}^{N} \Phi_i(u)\ w8_i
\end{aligned}
\tag{78}
$$

Our approach to determining the basis coefficients $w1_i$, ..., $w8_i$ will be identical to that used in Example 1. Therefore from Eqs.(42)-(52) we may immediately write:

$$w1_k = \sum_{i=1}^{N} G2_{ki}\ b13_i\ , \quad w2_k = \sum_{i=1}^{N} G1_{ki}\ b13_i\ , \quad w3_k = (1 - u_k)\ b1_k$$

$$w4_k = \sum_{i=1}^{N} G2_{ki}\ b16_i\ , \quad w5_k = b2_k\ , \quad w6_k = (1 - u_k)\ b5_k$$

$$w7_k = \sum_{i=1}^{N} G2_{ki}\ b9_i\ , \quad w8_k = (1 - u_k)\ b6_k \tag{79}$$

where $k = 1, ..., N$. With the evaluation of the output weights $w1_k$ through $w8_k$, and the application of Eq.(9), the FFANN model of Eqs.(56)-(58) is complete.

5.2.3 Results

As with Example 1, this system was solved using a variable-step Runge-Kutta iterative shooting technique for two runs, corresponding to $Pr = 1.0$ ($A_8 = 0.0$) and $Pr = 0.7$ ($A_8 = 0.853837$), and the results compared with the output from the FFANN constructed using the Petrov-Galerkin technique (Fig. 13 - 16). The constructed network used 162 processing elements in the hidden layer, corresponding to an even spacing of 81 knots and two auxiliary knots. The results were compared at 10 sample points ($S = 10$), with only five sample points shown in the figures for clarity. The variable step size Runge-Kutta algorithm marched from $\eta = 0$ to $\eta = 10$ in 40 steps for $Pr = 1.0$ and 46 steps for $Pr = 0.7$, for an error less than 1.0×10^{-6},

Physically $Pr = 1.0$ indicates that the momentum and thermal boundary layer are identical. Mathematically with $Pr = 1$ only Eq.(56) and Eq.(57) should be linearly independent as it can be shown that in this situation we should have $\lambda = df/d\eta$, and so $s = \tau$.

The number of iterations (I) required for solution of the basis coefficients were: $I = 19$ for $Pr = 1.0$ with $\epsilon = 1.834 \times 10^{-11}$, and $I = 36$ for $Pr = 0.7$ with $\epsilon = 3.711 \times 10^{-10}$. The values of the root mean square of the difference between the Runge-Kutta and the network output are given in the individual figure descriptions.

The program was run on a SUN Microsystems Sparcstation 2, with a run time on the order of 200 CPU seconds for $Pr = 1.0$ and 700 seconds for $Pr = 0.7$, for the 81 knot case.

5.2.4 Convergence

A logarithmic convergence plot of the variable $d^2f/d\eta^2$ (Fig. 17) was obtained by successive grid refinement. The value of the exponent for $Pr = 1.0$ is approximately 0.899. The convergence curve for $Pr = 0.7$ does not have a constant slope, though its average would appear to be the same as for the $Pr = 1.0$ case.

6 CONCLUSION

It is possible to construct FFANNs, with piecewise linear activation functions, to model the solutions to non-linear ordinary differential equations without the need for training in the conventional connectionist sense. The construction requires imposing certain constraints on the values of the input, bias, and output weights, the attribution of certain roles to each of these parameters, and the use of well-established mathematical techniques from function approximation theory. This approach should also be applicable to the use of hyperbolic tangents, sigmoids, and radial basis functions.

Explicit formulae for the input and bias weights were formulated and the method of weighted residuals (specifically, one of its variations, the Petrov-Galerkin method) was introduced as a mathematical algorithm to determine the values of the output weights of the FFANN. Two example problems drawn from fluid mechanics were approached and solved in this manner and the results used to construct neural networks. The output of the networks were compared with solutions obtained via an iterative shooting technique using the Runge-Kutta marching algorithm.

Like training methods, the algorithms used in this paper were iterative in nature. However, the algebraic system of equations is well-conditioned and as the number of processing elements increases, the error in the network output monotonically decreases. In addition, the algorithms are fairly conventional, requiring no special assumptions and running on conventional hardware in a reasonable amount of time for the problems presented.

7 CURRENT AND FUTURE RESEARCH

Since the quality of the basis function affects the interpolation, storage, speed, and convergence properties of the FFANN construction method, work is progressing on generating higher order splines using the hyperbolic tangent, sigmoid, and radial basis functions. The higher order splines being investigated include Lagrange, Hermite, and B-splines.

The approach outlined in this paper has allowed the investigation of suitable algorithms for the construction of FFANNs, Sigma-Pi, and recurrent networks, to approximate linear and nonlinear partial differential equations.

Work is progressing in applying the FFANN construction approach to the problem of supervised learning. A non-iterative "training" algorithm is being developed which will allow the supervised learning problem to be solved directly for the needed weights and number of hidden layers. The algorithm will require only the unique solution of a linear algebraic system of equations.

References

[1] J. Freeman and D. Skapura, *Neural Networks: Algorithms, Applications, and Programming Techniques*, Addison Wesley, New York, (1991).

[2] M. Takeda and J. Goodman, Neural Networks for Computation: number representation and programming complexity, *Applied Optics*, **25** (18), 3033, (1986).

[3] E. Barnard and D. Casasent, New Optical Neural System Architectures and Applications, *Optical Computing 88*, **963**, 537 Toulon, France, (1988).

[4] H. Lee and I. Kang, Neural Algorithms for Solving Differential Equations, *Journal of Computational Physics*, **91**, 110, (1990).

[5] L. Wang and J.M. Mendel, Structured Trainable Networks for Matrix Algebra, *IEEE International Joint Conference on Neural Networks*, **2**, 125 San Diego, June (1990).

[6] S. Omohundro, Efficient Algorithms with Neural Network Behaviour, *Complex Systems*, **1**, 237, (1987).

[7] T. Poggio and F. Girosi, A Theory for Networks for Approximation and Learning, *Artificial Intelligence Laboratory, Massachusetts Institute of Technology*, A.I. Memo No. 1140, July (1989).

[8] F. Girosi and T. Poggio, Networks for learning: a view from the theory of approximation of functions, *Proceedings of the Genoa Summer School on Neural Networks and Their Applications*, Prentice Hall, (1989).

[9] G. Cybenko, Approximation by Superposition of a Sigmoidal Function, *Math. Control Signals Systems*, **2**, 303, (1989).

[10] A. Maren, C. Harston and R. Pap, *Handbook of Neural Computing Applications*, Academic Press, New York, (1990).

[11] L.O. Chua, C.A. Desoer and E.S. Kuh, *Linear and Nonlinear Circuits*, McGraw Hill, New York, (1987).

[12] Y. Ito, Approximation of Functions on a Compact Set by Finite Sums of a Sigmoid Function Without Scaling, *Neural Networks*, **4**, 817, (1991).

[13] P.M. Prenter, *Splines and Variational Methods*, Wiley, New York, (1989).

[14] A.J. Meade, Jr. and A.A. Fernandez, The Numerical Solution of Linear Ordinary Differential Equations by Feedforward Neural Networks, *Mathematical and Computer Modelling*, **19** (12), 1, (1994).

[15] Finlayson, B. A. and Scriven, L. E., The Method of Weighted Residuals - A Review, *Applied Mechanics Review*, **19** (9), 735, (1966).

[16] C.A.J. Fletcher, *Computational Galerkin Methods*, Springer-Verlag, New York, (1984).

[17] G. Strang, *Linear Algebra and Its Applications*, Second Edition, Academic Press, New York, (1980).

[18] Frank M. White, *Viscous Fluid Flow*, McGraw-Hill Book Company, New York, (1974).

[19] I. Christie, D.F. Griffiths, A.R. Mitchell, and J.M. Sanz-Serna, Product Approximation for Non-linear Problems in the Finite Element Method, *Journal of Numerical Analysis* **1**, 253, (1981).

[20] K. Stewartson, Further solutions of the Falkner-Skan equation, *Proc. Cambr. Phil. Soc. Journal of Numerical Analysis* **50**, 454, (1954).

[21] *IMSL User's Manual, MATH/LIBRARY*, Version 2, (1991).

[22] C. Johnson, *Numerical Solution of Partial Differential Equations by The Finite Element Method*, Cambridge University Press, Cambridge, (1990).

[23] F.K. Moore, Three-Dimensional Compressible Laminar Boundary-Layer Flow, NACA TN 2279, (1951).

[24] F.K. Moore, Laminar Boundary Layer on Cone in Supersonic Flow at Large Angle of Attack, NACA TR 1132, (1952).

[25] E. Reshotko, Laminar Boundary Layer with Heat Transfer on a Cone at Angle of Attack in a Supersonic Stream, NACA TN 4152, (1957).

[26] A.J. Meade Jr., Semi-Discrete Galerkin Modelling of Compressible Viscous Flow Past a Circular Cone at Incidence, Ph.D Thesis, University of California, Berkeley, (1989).

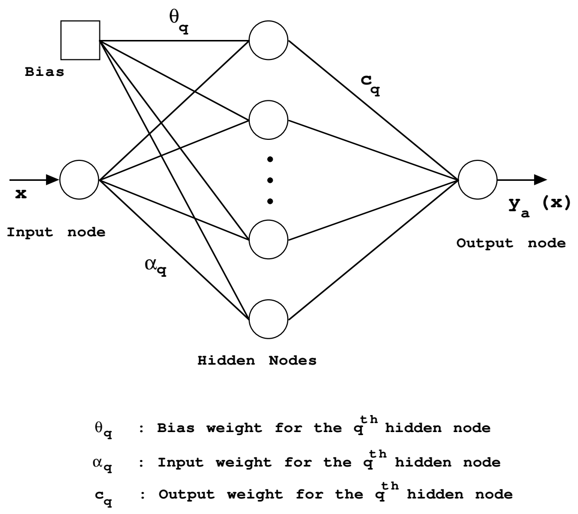

Figure 1: Standard feedforward neural network architecture.

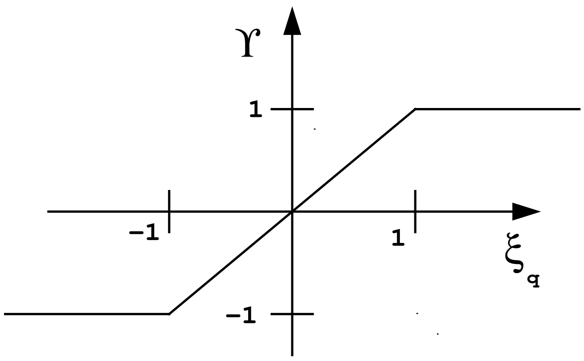

Figure 2: Hard limit transfer function.

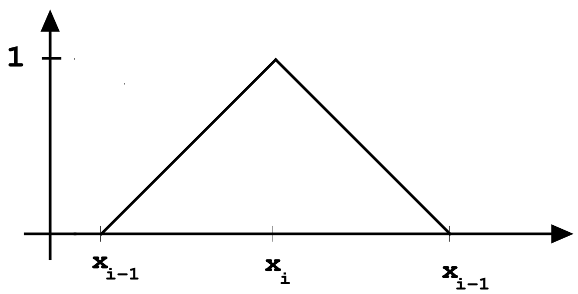

Figure 3: The Chapeau or "hat" function.

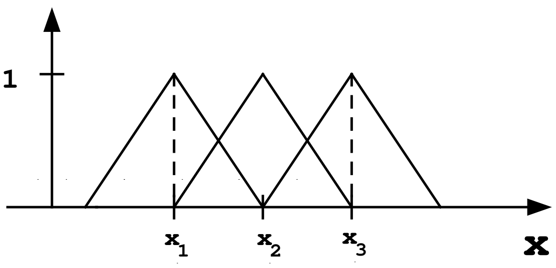

Figure 4: Proper distribution of hat functions along the problem domain ($x_1 \leq x \leq x_3$).

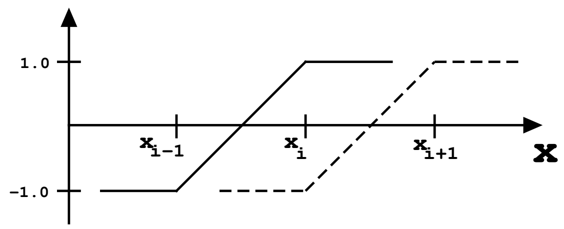

Figure 5: Constrained distribution of hard limits along x axis using the input and bias weights.

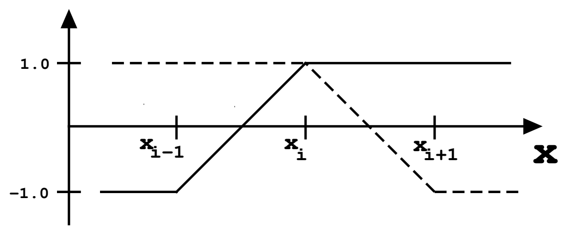

Figure 6: Right side hard limit (dashed) rotated by change of output weight sign. Addition of the two hard limits creates a hat function. Refer to Figure 3.

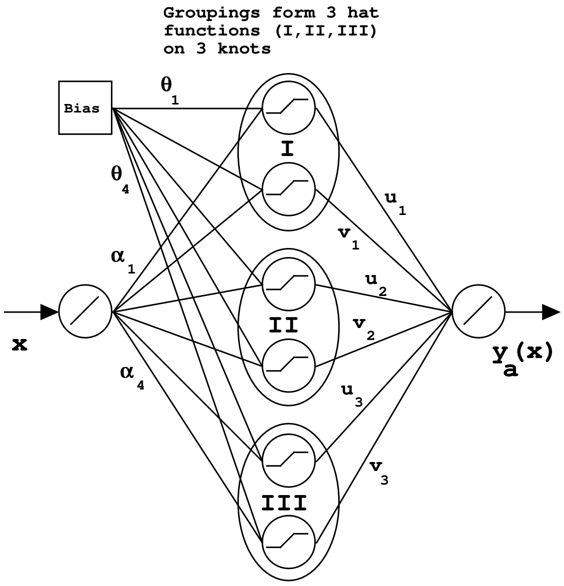

Figure 7: Single hidden layer feedforward network with constrained weights ($T = 6$, $N = 3$).

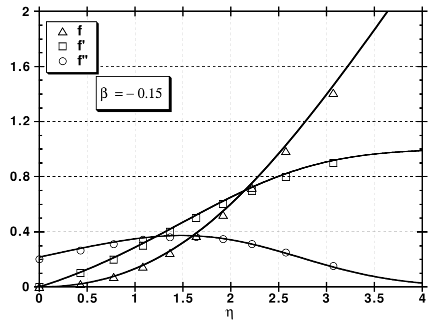

Figure 9: $\beta = -0.15$: Comparison of R-K solutions and network outputs with $T = 42$ processing elements. $f : RMS = 3.033 \times 10^{-2}$; $df/d\eta : RMS = 7.200 \times 10^{-2}$; $d^2f/d\eta^2 : RMS = 1.067 \times 10^{-2}$

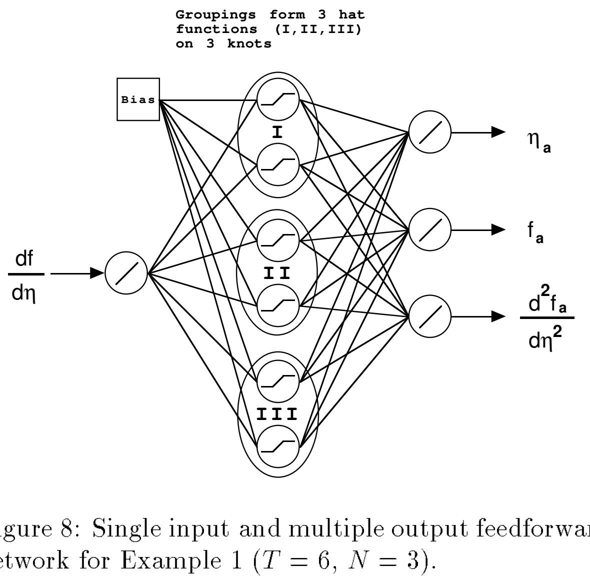

Figure 8: Single input and multiple output feedforward network for Example 1 ($T = 6$, $N = 3$).

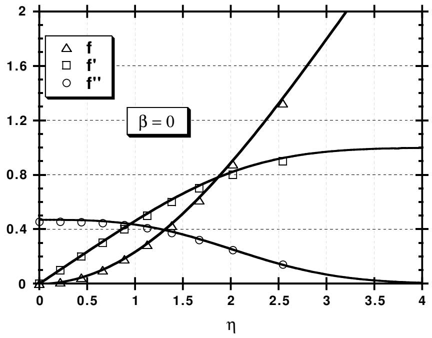

Figure 10: $\beta = 0$: Comparison of R-K solutions and network outputs with $T = 22$ processing elements. f : $RMS = 3.563 \times 10^{-2}$; $df/d\eta : RMS = 6.215 \times 10^{-2}$; $d^2f/d\eta^2 : RMS = 1.788 \times 10^{-2}$

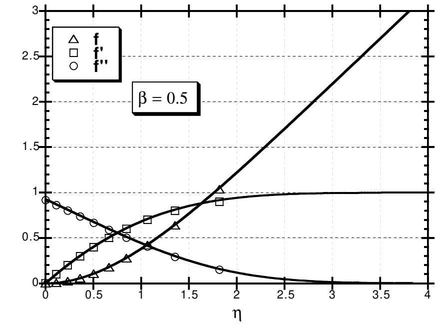

Figure 11: $\beta = +0.5$: Comparison of R-K solutions and network outputs with $T = 22$ processing elements. $f : RMS = 2.554 \times 10^{-2}$; $df/d\eta : RMS = 3.534 \times 10^{-2}$; $d^2f/d\eta^2 : RMS = 1.741 \times 10^{-2}$

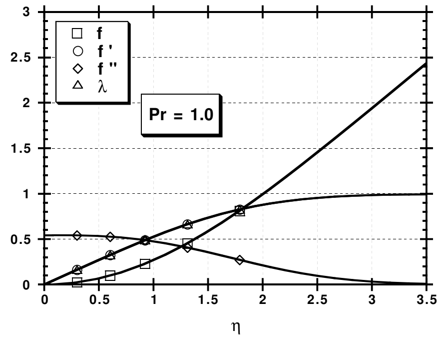

Figure 13: $Pr = 1.0$: Comparison of R-K solutions and network outputs with $T = 162$ processing elements (5 sample points shown). $f : RMS = 1.683 \times 10^{-2}$; $df/d\eta : RMS = 1.907 \times 10^{-2}$; $d^2f/d\eta^2 : RMS = 2.740 \times 10^{-3}$; $\lambda : RMS = 1.223 \times 10^{-6}$

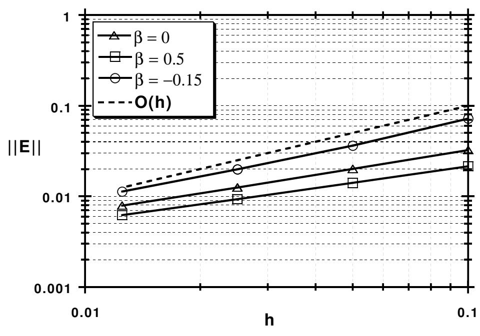

Figure 12: Logarithmic convergence plot for $d^2f/d\eta^2$. $\beta = -0.15$: *slope* $= 0.892$; $\beta = 0$: *slope* $= 0.679$; $\beta = +0.5$: *slope* $= 0.598$.

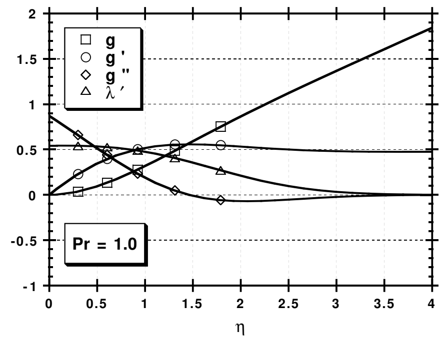

Figure 14: $Pr = 1.0$: Comparison of R-K solutions and network outputs with $T = 162$ processing elements (5 sample points shown). $g : RMS = 1.178 \times 10^{-2}$; $dg/d\eta : RMS = 4.153 \times 10^{-3}$; $d^2g/d\eta^2 : RMS = 1.524 \times 10^{-3}$; $d\lambda/d\eta : RMS = 2.739 \times 10^{-3}$

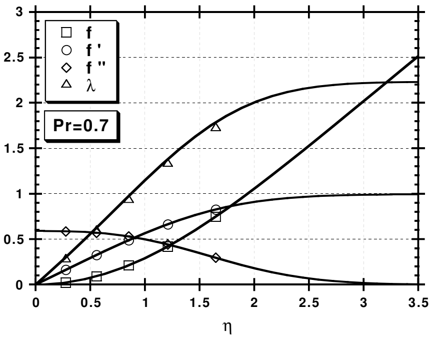

Figure 15: $Pr = 0.7$: Comparison of R-K solutions and network outputs with $T = 162$ processing elements (5 sample points shown). f : $RMS = 1.628 \times 10^{-3}$; $df/d\eta$: $RMS = 2.854 \times 10^{-3}$; $d^2f/d\eta^2$: $RMS = 1.102 \times 10^{-3}$; λ : $RMS = 3.050 \times 10^{-2}$

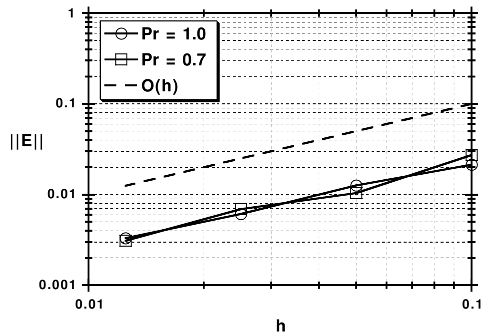

Figure 17: Logarithmic convergence plot for $d^2f/d\eta^2$. $Pr = 1.0$: $slope = 0.899$; $Pr = 0.7$: $avg.slope \approx 0.899$.

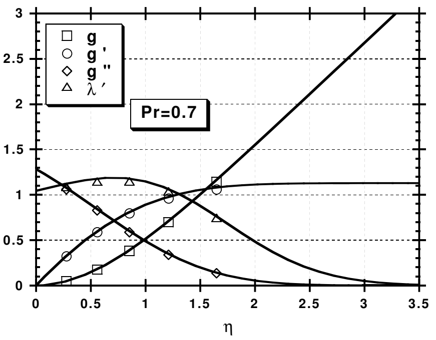

Figure 16: $Pr = 0.7$: Comparison of R-K solutions and network outputs with $T = 162$ processing elements (5 sample points shown). g : $RMS = 1.274 \times 10^{-2}$; $dg/d\eta$: $RMS = 1.325 \times 10^{-2}$; $d^2g/d\eta^2$: $RMS = 1.385 \times 10^{-2}$; $d\lambda/d\eta$: $RMS = 2.706 \times 10^{-2}$