Initialization of Artificial Neural Network Training Using Mathematical Models

Andrew J. Meade, Jr.[1], Gregory S. Lind and Boris A. Zeldin
Department of Mechanical Engineering
and Materials Science
Mail Stop 321
William Marsh Rice University
6100 South Main Street
Houston, Texas 77005-1892, USA
Phone: (713) 527-8101 ext. 3590
E-mail: meade@rice.edu

Submitted for publication to *Journal of Guidance, Control, and Dynamics*.

---

1. Author to whom correspondence should be addressed. meade@rice.edu
This work was supported under NASA grant NGT 51230 and Office of Naval Research grant N00014-95-1-0741.

Initialization of Artificial Neural Network Training Using Mathematical Models

Andrew J. Meade, Jr., Gregory S. Lind and Boris A. Zeldin
Department of Mechanical Engineering and Materials Science
Mail Stop 321
William Marsh Rice University
6100 South Main Street
Houston, Texas 77005-1892, USA

Keywords: Neural networks, neural computation, mathematical modelling, mechanical systems.

Abstract

An initiation method is developed for networks trained to model physical systems. 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 allows for mapping a polynomial approximation into an artificial neural network architecture. Attention is focused on time-dependent linear and non-linear partial differential equations modelled by the recurrent artificial neural network architecture, in conjunction with the popular hyperbolic tangent transfer function. Moreover, this approach is shown to be capable of incorporating other smooth neuron transfer functions. Numerical examples are presented illustrating the accuracy and utility of the method.

Initialization of Artificial Neural Network Training Using Mathematical Models

Andrew J. Meade, Jr.[2], Gregory S. Lind[3] and Boris A. Zeldin[4]
William Marsh Rice University, Houston, Texas 77005-1892, USA

1. Introduction

At present an area of increasing interest is the data driven emulation and/or control of engineering systems by artificial neural networks (ANNs). It is well known that neural networks can approximate functions and mathematical operators arbitrarily well as the number of neurons in the network tends to infinity [1-6]. In this regard, ANNs can be considered as 'universal approximators' capable of describing the input-output relationships of mechanical systems. For practicing engineers the universal approximation capability of ANNs would be especially useful in the emulation of complex mechanical systems.

The training of an ANN system can be formulated as the procedure of selecting the optimal network parameters so that the response of the neural network is 'close' to the available data; this is measured by an error criterion. In terms of the connectionist literature, numerical minimization of this error criterion by the steepest descent method constitutes the backpropagation algorithm [7]. The conjugate gradient method, the Levenberg-Marguardt method, and other non-linear optimization methods have also been used for network training. However, it has been found that even the most advanced optimization algorithms can show unreliable convergence in training [8]. Clearly, the convergence of training algorithms can be improved, to a great extent, by an effective selection of the initial values of the network parameters. Otherwise, even the most sophisticated optimization algorithms described in reference [8] may converge to a poor local minimum or may

2. Associate Professor. Member AIAA. Author to whom correspondence should be addressed. E-mail: meade@rice.edu.
3. Research Assistant.
4. Research Associate.

not reach convergence at all; the latter phenomenon is known as overtraining [9].

In neural network applications, the initial values for the network parameters are commonly selected by random sampling [10]. If a satisfactory solution is not obtained during training, the non-linear optimization procedure is restarted with new randomly selected values [11]. Clearly this approach can require prohibitive computational resources for some practical problems. Alternatively, Lehtokangas et al. [12] proposed that an orthogonal least squares algorithm [13] be used for the initiation of the network parameters. In this case, a large network is constructed by first randomly sampling the non-linear parameters and then determining the corresponding optimal values of the linear parameters. Consequently, the size of the network is reduced by removing neurons which do not significantly contribute to the network approximation. The deficiency in this approach is the necessity in determining a large intermediate network which can adequately emulate the system of interest; this can still require significant computational effort and, so, diminishes the applicability of the initiation method. Moreover, the size of the network is not of primary concern for a significant class of practical problem if an accurate network approximation can be obtained by adjusting only the linear network parameters. In this regard, Burrows and Niranjan [14] proposed to initialize the network parameters by linearizing the neuron transfer functions. Unfortunately, this method is also not satisfactory for emulating highly non-linear input-output relationships.

The objective of this paper is to develop a new initiation strategy for network modelling of physical systems by connectionist training algorithms. The proposed method is based on three assumptions. Firstly, an ANN system can be treated as a general approximation scheme [15] capable of approximating differential equations. Secondly, polynomial bases, which are a traditional method for approximating functions and analyzing experimental data [16, 17], can be mapped into neural network architectures. Thirdly, mathematical models of engineering systems, linear and non-linear, can be utilized to obtain initial values for the network parameters.

It is shown in this paper that mathematical models can be an efficient tool for initiating a training procedure in network simulations of physical systems. Note that the proposed approach allows the initiation of a connectionist training algorithm without data; this feature can be especially useful for on-line ANN modelling systems. Also, the method can be useful in obtaining preliminary bounds on the network performance and estimating the accuracy of network approximations. For example, by exploiting the analogy between ANN systems and polynomial approximations, the effect of augmenting networks with additional processing elements can be evaluated. Finally, initiating training with a mathematical model of sufficient fidelity significantly reduces problems associated with the inadequacy of a small training set in describing a physical system. With this approach the network parameters are pre-determined by a mathematical model leading to a reliable approximation of a physical system in parameter regions lacking data.

For the sake of brevity attention is focused on modelling time-dependent linear and non-linear partial differential equations by the recurrent artificial neural network (RANN) architecture with the hyperbolic tangent transfer function. However, the approach should be applicable to linear and non-linear elliptic equations and incorporate other smooth transfer functions.

2. Technical Background

2.1 Recurrent Neural Networks

A recurrent artificial neural network (RANN) consists of several layers of processing elements as illustrated in Fig. 1. The variables $u_i(t), \quad i = 1, \ldots, n$ denote the input states of the RANN at the time t while the variables $v_i(t), \quad i = 1, \ldots, N$ denote the output states of the network at the time t which in turn are fed back as some of the input variables to determine the RANN response at the next time period $t + 1$. The first l components of the input state vector $\bar{u}(t)$ are selected as the input variables of the physical system, $l < n$. Similarly, the first L components of the vector $\bar{v}(t)$ specify the response of the corresponding physical system, $L \leq N$. The RANN consists of k layers of non-linear processing elements. Thus, the RANN output at time $t + 1$ can be determined by the equation

$$v_j(t+1, \bar{\beta}) = \sigma_k\left(\sum_l \beta^j_{(k-1),l}\sigma_{k-1}\left(\dots\sigma_2\left(\sum_i \beta^p_{1,i}\sigma_1(u_i(t))\right)\right)\right), \tag{1}$$

where the vector $\bar{\beta}$ represents the set of network parameters, and σ_j is the non-linear transfer function of neurons associated with the *j*-th intermediate layer of the network. A simple RANN architecture with two layers of non-linear processing elements is quite common in network applications. However, a problem-oriented consideration may indicate that several layers can have a beneficial impact on the overall behavior of the network. In particular, the proposed approach shows that 3-layer RANN systems, which are utilized in this study, can be readily associated with some conventional approximation procedures.

Several alternative functions have been used as the processing elements in RANNs, but the more popular training procedures, such as backpropagation, require that the functions σ_k be continuously differentiable and bounded. In this regard, sigmoidal neurons are often selected for network applications. The corresponding functions are continuous and are characterized by the following limits

$$\lim_{\xi \to \infty} \sigma(\xi) = 1, \qquad \lim_{\xi \to -\infty} \sigma(\xi) = -1 \tag{2}$$

The function

$$\sigma(\xi) = \tanh \xi \tag{3}$$

is a paradigmatic example of such sigmoidal functions.

The training of the RANN system is performed by adjusting the network parameters in the vector $\bar{\beta}$. Specifically,

$$\bar{\beta}^{\text{new}} = \bar{\beta}^{\text{old}} - \alpha \nabla_{\tilde{\beta}} \varepsilon(\bar{\beta}), \tag{4}$$

where α is the learning rate constant selected by emulating a corresponding non-linear optimization procedure. Further, $\varepsilon(\bar{\beta})$ is the training objective function that designates the measure of discrepancy between the network response and the observed data. Commonly, $\varepsilon(\bar{\beta})$ is calculated by the equation

$$\varepsilon(\bar{\beta}) = \sum_{i=1}^{L} \sum_{t} \left| f_i(t) - v_i(\bar{\beta}, t) \right|^2, \tag{5}$$

where $f_i(t)$ represent the observed response of the physical system. The objective function taken in the form of equation (5) readily yields an analytical expression for the gradient $\nabla_{\tilde{\beta}} \varepsilon(\bar{\beta})$ needed in equation (4) [9].

As it has been noted previously, the success in training by equation (4) depends, to a great extent, on an appropriate selection of the initial values of the network parameters; in this regard, a new procedure is developed in the following section.

3. RANN Parameter Initiation by Using Mathematical Models.

3.1 Linear versus Non-Linear Training

Figure 1 presents the general architecture of a RANN. However, in an actual application the user must specify the number of neuron layers as well as the number of neurons in each layer and their connections. These user specifications may be based on intrinsic considerations. In this study 3-layer networks are used. Also, the transfer functions of the first and the last layer are taken as linear

$$\sigma_i(x) = x, \quad i = 1, 3, \tag{6}$$

whereas the transfer function of the second intermediate layer is taken as the sigmoidal function of equation (3). The response of this type of RANN can be written as

$$v_j(t+1) = \sum_l \beta_{2,l}^j \tanh\left(\sum_i \beta_{1,i}^l u_i(t-1) + \theta_i\right), \tag{7}$$

where θ_i are biases associated with the neurons of the intermediate layer. Note that the network response depends linearly only on the parameters $\beta_{2,l}^j$, other parameters affect the network response in a highly non-linear manner. In this regard, initial values of the network parameters adequate for successful network training can be determined efficiently by establishing a set of constraints on the non-linear parameters. The response of these types of neural networks can then be seen as a linear combination of basis functions $\varphi_l(\bar{u})$. Specifically,

$$v_j(t+1) = \sum_l c_l^j \varphi_l(\bar{u}). \tag{8}$$

For engineering applications, bases with the following properties could be especially useful:

- The bases should be assembled in a straightforward manner from ANN architectures with parameters that can be evaluated uniquely and in a numerically stable manner.
- The bases should accurately approximate the desired function.
- To evaluate the linear coefficients, the bases should allow for utilizing a variety of computational techniques familiar to the engineering community such as finite difference, finite elements and spectral methods.
- The bases should allow the user to control the accuracy of the network approximation.

- The developed bases should have the property of approximating desired functions using a small number of hidden layer neurons and of requiring only a modest increase in the number of neurons to improve modelling accuracy (i.e., good rate of convergence).

In this paper global bases will be formed from linear combinations of transfer functions. The linear coefficients (weights) associated with the bases will be evaluated by linear optimization techniques. This approach readily yields reliable estimates of the initial values of network parameters for subsequent training by non-linear optimization algorithms.

3.2 Sigmoidal Networks and Polynomial Bases Approximation

The idea of mapping a conventional approximation scheme, with well established approximation properties, into neural network architecture has been pursued in the literature [3-5, 18, 19] for theoretical studies regarding the density and approximation properties of ANNs and the dependence of the network approximation error on the number of neurons. In this paper, the mapping approach is used with a new and practical intent. Specifically, the well known method of polynomial approximation commonly used in data analysis and computational mechanics applications will be mapped into a network architecture for the emulation of mechanical systems.

In this regard, attention is drawn to the following formula

$$\varphi_{\boldsymbol{p}}(\bar{w}, \bar{x}) = \frac{\partial^{|\boldsymbol{p}|}}{\partial w_1{}^{p_1} \dots \partial w_s{}^{p_s}} \varphi(\bar{w}^T \bar{x} + \theta) = \bar{x}^{\boldsymbol{p}} \varphi^{(|\boldsymbol{p}|)}(w\bar{x} + \theta), \tag{9}$$

where $(\circ)^T$ denotes a vector transpose, $\bar{x}^{\boldsymbol{p}} = x_1^{p_1} \cdot \dots x_s^{p_s}$, $\boldsymbol{p}$ is a multi-index such that $|\boldsymbol{p}| = p_1 + \dots p_s$, and φ is a non-linear non-polynomial function. Equation (9) shows that a polynomial $\bar{x}^{\boldsymbol{p}}$ can be readily expressed as

$$\bar{x}^{\boldsymbol{p}} = (\varphi^{(|\boldsymbol{p}|)}(\theta))^{-1} \varphi_{\boldsymbol{p}}(0, x), \tag{10}$$

where the bias θ is selected so that $\varphi^{(|\boldsymbol{p}|)}(\theta)$ is large. Therefore, by replacing the derivative $\varphi_{\boldsymbol{p}}(0, \bar{x})$ by an adequate finite difference approximation, one can accurately approximate the polynomial $\bar{x}^{\boldsymbol{p}}$ by a finite linear combination of functions in the form $\varphi(\overline{w}_i^T \bar{x} + \theta)$. It has been determined that the following finite difference scheme is adequate for practical applications

$$\varphi_{\boldsymbol{p}}(0, \bar{x}) \approx \Phi_{\boldsymbol{p}}(\bar{x}) = (2\Delta w)^{-|\boldsymbol{p}|} \sum_{0 \le r \le p} (-1)^{|\boldsymbol{r}|} \binom{\boldsymbol{p}}{\boldsymbol{r}} \varphi((2\boldsymbol{r} - \boldsymbol{p})^T \Delta w \cdot \bar{x} + \theta), \tag{11}$$

where Δw is a small positive constant and the multi-integer binomial is defined as

$$\binom{\boldsymbol{p}}{\boldsymbol{r}} = \prod_{j=1}^{s} \binom{p_j}{r_j}. \tag{12}$$

Note that the accuracy of the finite difference scheme of equation (11), within a bounded domain of $\bar{x}$, is given by the following error bound

$$max\left|\Phi_{\boldsymbol{p}}(\bar{x}) - \varphi_{\boldsymbol{p}}(0, \bar{x})\right| \le C(\Delta w)^2. \tag{13}$$

Finally, combining equations (10) and (11) and replacing the function $\varphi(x)$ by the intermediate neuron transfer function, $\tanh x$, one can show that the polynomial $\bar{x}^{\boldsymbol{p}}$ can be approximated by a network with $\prod_{j=1}^{s} (p_j + 1)$ neurons in the intermediate layer. The error of this approximation can be made arbitrarily small by selecting an arbitrarily small value for Δw. For example, the linear function x can be approximated as

$$x \approx \frac{\tanh(\Delta w \cdot x + \theta) - \tanh(-\Delta w \cdot x + \theta)}{2\Delta w \cdot \tanh'(\theta)}. \tag{14}$$

where the prime denotes first-order differentiation.

Note that the developments of this subsection have been previously utilized in references [18, 19] for establishing the approximation properties of some types of neural network architectures. However, in the next subsection this approach will be combined with mathematical models of physical systems to obtain a reliable approximation scheme for initiating RANN training.

3.3 Developing Analytical RANN Models of Dynamic Systems

Let an a-priori mathematical model of a mechanical system be described by the following partial differential equation

$$\frac{\partial}{\partial t}y(x, t) - L(y(x, t)) = q(x, t) \tag{15}$$

augmented with appropriate boundary and initial conditions. Here, y and q denote response and excitation functions, respectively, and L is the spatial operator, possibly non-linear. Note that equation (15) describes the continuous time variation of all points of the mechanical system. Clearly, since a finite ANN system can describe variation of only a discrete-time, finite-state system, equation (15) must be appropriately discretized both in the temporal and spatial domains.

In this regard, the response of the mechanical system can be approximated as

$$y(x, t) \approx \sum_{i=1}^{N} v_i(t)\phi_i(x) , \tag{16}$$

where $\phi_i(x)$ are a set of basis functions in the spatial domain and $v_i(t)$ are the time-dependent coefficients of this expansion. Note that $v_i(t)$ can be associated with a dynamic behavior of a finite number of nodes within the domain of the mechanical system. Clearly, the approximation of equation (16) does not automatically satisfy the governing differential equation (15). Substituting equation (16) into equation (15) results in a non-zero equation residual $R(x, t)$ where

$$R(x, t) = \sum_{i=1}^{N} \frac{\partial}{\partial t} v_i(t)\phi_i(x) - L\left(\sum_{i=1}^{N} v_i(t)\phi_i(x)\right) - q(x, t) . \tag{17}$$

The method of weighted residuals [20, 21] is utilized to obtain a set of equations involving the coefficients $v_i(t)$ and equation residual $R(x, t)$. Specifically, the residual $R(x, t)$ is forced to be orthogonal to a pre-selected set of *N* weighting functions $w_i(x)$. This is written as,

$$\langle R, w_j\rangle = \sum_{i=1}^{N} \frac{\partial}{\partial t} v_i(t)\langle \phi_i, w_j\rangle - \langle L\left(\sum_{i=1}^{N} v_i(t)\phi_i\right), w_j\rangle - u_j(t) = 0 , \quad j = 1, \ldots N , \tag{18}$$

where $u_j(t)$ denotes the discretized excitation term $\langle q, w_j\rangle$ and the inner product is defined as $\langle \varphi, \psi\rangle = \int \varphi(x)\psi(x)dx$.

Next, the temporal domain is discretized by the Euler forward difference scheme, yielding the following time stepping rule

$$v_i(t) = v_i(t-1) + \Delta t \sum_{j=1}^{N} [C^{-1}]_{ij} \langle L\left(\sum_{k=1}^{N} v_k(t-1)\phi_i\right), w_j\rangle + \Delta t \sum_{j=1}^{N} [C^{-1}]_{ij} u_j(t-1) , \tag{19}$$

where C is a matrix with elements $[C]_{ij} = \langle \phi_i, w_j\rangle$. Note that more accurate time integration procedures, such as the MacCormack method [22], can also be applied. Next, the non-linear term in equation (19) can be accurately approximated by polynomials, resulting in the following recursive scheme for the determination of the system response at time *t*

$$v_i(t) = \sum_{j=1}^{N} \alpha_{ij}^1 v_j(t-1) + \sum_{j,k=1}^{N} \alpha_{ijk}^2 v_j(t-1) v_k(t-1) + \ldots \Delta t \sum_{j=1}^{N} [C^{-1}]_{ij} u_j(t-1), \qquad (20)$$

where $\alpha_{i,j,\ldots}^l$ are parameters determined by the operator L.

As shown in the preceding subsection, the recurrent time stepping scheme of equation (20) can be realized as a 3-layer RANN. The details of the developed procedure will be clarified by means of specific examples.

4. Numerical Results

Two numerical examples are presented in this section to illustrate the usefulness and applicability of the proposed training initiation method for mechanical applications. These examples involve a one-dimensional heat-conductive system and a scalar representation of the Navier-Stokes equations.

4.1 Heat Equation as a Model of a Dynamic System

The one-dimensional heat-conductive system is described by the following equation

$$\frac{\partial}{\partial t} T(x,t) - \alpha \frac{\partial^2}{\partial x^2} T(x,t) = 0, \qquad (21)$$

where $T(x,t)$ is the temperature of the system at time t and at the location x, α is the system conductivity ($\alpha = 1$) and external heat sources are assumed to be equal to zero. Equation (21) is a simple parabolic equation which describes time-varying heat conduction in an isotropic medium. Further, equation (21) is augmented by the following boundary and initial conditions

$$T(0,t) = 0, \quad T(1,t) = 1, \qquad (22)$$

$$T(x, 0) = T_0(x) = \sin(\pi x) + x. \tag{23}$$

The analytical solution of this heat conductivity problem is available in the literature

$$T(x, t) = \sin(\pi x) e^{-\pi^2 t} + x. \tag{24}$$

Following the developments of Section 3.3 the temperature $T(x, t)$ is approximated as

$$\tilde{T}(x, t) = \sum_{i=1}^{N} b_i(t)\phi_i(x), \tag{25}$$

where the bases $\phi_i(x)$ are taken as 'hat' functions (or B_1 splines) associated with the grid nodes $x_i = (i-1)\Delta x = (i-1)/(N-1)$, $i = 1, \ldots N$,

$$\phi_i(x) = \begin{cases} (x - x_{i-1})/\Delta x, & x_i \geq x \geq x_{i-1} \\ (x_{i+1} - x)/\Delta x, & x_{i+1} \geq x \geq x_i \; ; \\ 0, & \text{otherwise.} \end{cases} \tag{26}$$

Using these basis functions, the coefficients $b_i(t)$ directly represent the temperature of the conductive system at the grid nodes x_i. Further, substituting equation (25) into equation (21) and forcing the corresponding equation residual to be orthogonal to the set of functions $\phi_i(x)$, $i = 1, \ldots, N$ one obtains the following system of differential equations

$$C_1 \frac{d}{dt}\bar{b}(t) + \alpha C_2 \bar{b}(t) = 0, \tag{27}$$

where the matrices C_1 and C_2 are defined by the equations

$$[C_1]_{ij} = \int \phi_i(x)\phi_j(x)dx\,, \qquad [C_2]_{ij} = \int \phi_i'(x)\phi_j'(x)dx\,. \tag{28}$$

Utilizing the MacCormack method of time-integration, equation (27) gives

$$\bar{b}(t) = (I - \alpha\Delta t C_1^{-1} C_2 + (\alpha\Delta t C_1^{-1} C_2)^2)\bar{b}(t-1)\,. \tag{29}$$

where I is the identity matrix.

The RANN system that implements the recurrent relationship of equation (29) is shown in Fig. 2. The response of this RANN system is plotted in Fig. 3for $N = 11$ and $\Delta t = 0.025$. The initial values $\bar{b}(0)$ corresponds to the initial temperatures of the nodal points given by equation (23). The error distribution of the network approximation is plotted in Fig. 4. The network emulates this multi-state mechanical system adequately, providing for an efficient initialization of the training procedure.

4.2 Burger's Equation as a Model of a Dynamic System

The viscous Burger's equation,

$$\frac{\partial}{\partial t}u(x, t) + u\frac{\partial}{\partial x}u(x, t) = \upsilon\frac{\partial^2}{\partial x^2}u(x, t) \tag{30}$$

represents the scalar form of the time-dependent Navier-Stokes equations. Here, $u(x, t)$ denotes the velocity of the fluid at the location x and at time t and υ acts as the kinematic viscosity, which is assumed constant ($\upsilon = 0.2$). With the boundary and initial conditions given by the equations

$$u(-2, t) = 1\,, \quad u(2, t) = 0 \tag{31}$$

$$u(x,0) = u_0(x) = \begin{cases} 1, & x_{(N-1)/2-1} \geq x \geq -2 \\ (x_{(N-1)/2+1} - x)/2\Delta x, & x_{(N-1)/2-1} \geq x \geq x_{(N-1)/2+1}\,, \\ 0, & \text{otherwise} \end{cases} \tag{32}$$

it is seen that Burger's equation simulates a one-dimensional disturbance propagating through a viscous fluid.

As with the preceding example, the velocity is approximated by the equation

$$\tilde{u}(x,t) = \sum_{i=1}^{N} b_i(t)\phi_i(x)\,, \tag{33}$$

where the basis functions $\phi_i(x)$ are the hat functions of equation (26). In this example, the functions $b_i(t)$ denotes the time variation of the velocity at the grid node location x_i. The squared velocity can be approximated as

$$\tilde{u}^2(x,t) = \sum_{i=1}^{N} b_i^2(t)\phi_i(x)\,. \tag{34}$$

Substituting equations (33) and (34) into equation (30) and using the method of weighted residuals yields the following system of ordinary differential equations

$$C_1\frac{d}{dt}\bar{b}(t) + \upsilon C_2\bar{b}(t) + \frac{1}{2}C_3\bar{b}^2 = 0\,, \tag{35}$$

where $\bar{b}^2$ denotes a vector with components $b_i^2(t)$. The matrices C_1 and C_2 are given by equation (28) and

$$[C_3]_{ij} = \int \phi_i(x)\phi_j{}'(x)dx\,. \tag{36}$$

Finally, applying the forward difference scheme to the differential equation (27) yields the following time stepping rule

$$\bar{b}(t) = (I - \upsilon\Delta t C_1^{-1}C_2)\bar{b}(t-1) - \frac{1}{2}\Delta t C_1^{-1}C_3\bar{b}^2(t-1)\,. \tag{37}$$

A RANN system that implements the recurrent relationship of equation (37) is shown in Fig. 5. The response of this RANN system is plotted in Fig. 6 for $N = 41$ and $\Delta t = 4 \times 10^{-4}$. The initial values $\bar{b}(0)$ corresponds to the initial velocity distribution at the nodal points given by equation (32). The error distribution of the network approximation is plotted in Fig. 7. As in the previous example, a RANN of relatively modest size emulates this non-linear multi-state mechanical system with acceptable accuracy.

5. Conclusion

The adaptive data-driven emulation and control of mechanical systems are popular applications of artificial neural networks in engineering. However, because ANN training is non-linear optimization problem it is difficult to apply this attractive modelling tool to engineering applications. As a remedy, a new initiation method for connectionist training algorithms has been presented. The method is based on forming global polynomial bases from linear combinations of smooth transfer functions and then mapping the polynomial bases into a network architecture. Mathematical models of engineering systems have been, then, used to obtain initial values for the network parameters. It is noted that the proposed method allows for the initialization of a connectionist on-line training algorithm without data. Also, since polynomial bases are well known in conventional computational mechanics, the method can be useful in obtaining preliminary bounds on the network performance and estimating the accuracy of network approximations. With this approach the network parameters are pre-determined by a mathematical model leading to a reliable approximation of a physical system in parameter regions lacking data.

For the sake of brevity attention has been focused on modelling time-dependent linear and non-linear partial differential equations by the recurrent artificial neural network architecture with the hyperbolic tangent transfer function. Numerical examples have illustrated the accuracy and utility of the initialization method. Since the approach needs only smooth transfer functions to form conventional polynomial bases, it should also be applicable to linear and non-linear elliptic equations with other smooth transfer functions and it can be used for training feed-forward artificial neural networks.

Acknowledgments

This work was supported under NASA grant NGT 51230 and Office of Naval Research grant N00014-95-1-0741.

References

1. Cybenko, G., "Approximation by Superposition of a Sigmoidal Function," *Mathematics of Control, Signals, and Systems*, Vol. 2, 1989, pp. 303-314.

2. Girosi, F., and Poggio, T., "Networks and the Best Approximation Property," *Biological Cybernetics*, Vol. 63, 1990, pp. 169-176.

3. Hornik, K., Stinchcombe, M., and White, H., "Multilayer Feedforward Networks are Universal Approximators, *Neural Networks*, Vol. 2, 1989, pp. 359-366.

4. Hornik, K., Stinchcombe, M., and White, H., "Universal Approximation of an Unknown Mapping and its Derivatives Using Multilayer Feedforward Networks," *Neural Networks*, Vol. 3, 1990, pp. 551-560.

5. Ito, Y., "Representation of Functions by Superpositions of a Step or Sigmoidal Function and

Their Applications to Neural Network Theory," *Neural Networks*, Vol. 4, 1991, pp. 385-394.

6. Park J., and Sanberg, I.W., "Universal Approximation Using Radial-Basis-Function Networks," *Neural Computation* 3, 1991, pp. 246-257.

7. Rumelhart, D.,Hinton, G.E., and Williams, R.J., "Learning Internal Representations by Error Propagation," in *Parallel Distributed Processing: Exploration in the Microstructure of Cognition*, D. Rumelhart, J.L. McClelland, and the PDP Research Group, eds., Ch. 8, 1986, MIT Press, Cambridge, MA, pp. 318-362.

8. Saarinen, S., Bramley, R., and Cybenko, G., "Ill-Conditioning in Neural Network Training Problems," *SIAM J. Sci. Comput*. Vol. 14, No. 3, 1993, pp. 693-714.

9. Hecht-Nielsen, R., *Neurocomputing*, Addison-Wesley, Inc., New York, 1990.

10. Igelnik, B., and Pao, Y.-H., "Stochastic Choice of Basis Function in Adaptive Function Approximation and the Function-Link Net", *IEEE Transactions on Neural Networks*, Vol. 6, No. 6, 1995, pp. 1320-1329.

11. Schidt, W.F., Raudys, S., Kraaijveld, M.A., Skurikhina, M., and Duin, R.P.W., "Initializations, Back-Propagation and Generalization of Feed-Forward Classifiers", *Proceedings of IEEE International Conference on Neural Networks, ICNN-93*, Vol 1, 1993, pp. 598-604.

12. Lehtokangas, M., Saarinen, J., Kaski, K., and Huuntanen, P., (1995). "Initializing Weights of a Multilayer Perceptron Network by Using the Orthogonal Least Squares Algorithm", *Neural Computation*, Vol. 7, 1995, pp. 982-999.

13. Chen, S., Cowan, C.F.N., and Grant, P.M., "Orthogonal Least Squares Learning Algorithm for Radial Basis Function Networks", *IEEE Transactions on Neural Networks*, Vol. 2, No. 2, 1991, pp. 302-309.

14. Burrows, T.L., and Niranjan, M., *The Use of Feed-Forward and Recurrent Neural Networks*

for System Identification, Technical Report CUED/F-INFENG/TR158, Cambridge University Engineering Department, Cambridge, England, 1993.

15. Poggio, T., and Girosi, F. “Networks for Approximation and Learning,” *Proceedings of the IEEE*, Vol. 78, No. 9, 1990, pp. 1481-1497.

16. Canto, C., Hussaini, M.Y., Quarteroni, A., Zang, T.A., *Spectral Methods in Fluid Dynamics*, Springer-Verlag, Berlin Heidelberg, 1988, pp. 31-75.

17. Box, G.E.P., and Draper, N.R. *Empirical Model-Building and Response Surface*, Wiley, New York, 1987.

18. Leshno, M., Lin, V.Y., Pinkus, A., and Schocken, S., “Multilayer Feedforward Networks with a Nonpolynomial Activation Function Can Approximate Any Function”, *Neural Networks*, Vol. 6, 1993, pp. 861-867.

19. Mhaskar, H.N., “Neural Networks for Optimal Approximation of Smooth and Analytic Functions’’, *Neural Computation*, Vol. 8, 1995, pp. 164-177.

20. Finlayson, B.A., *The Method of Weighted Residuals and Variational Principles*, Academic Press, New York, 1972.

21. Fletcher, C.A.J., *Computational Galerkin Methods*, Springer, New York, 1984.

22. Hoffmann, K.A., *Computational Fluid Dynamics for Engineers*, Engineering Education System, Austin, 1989.

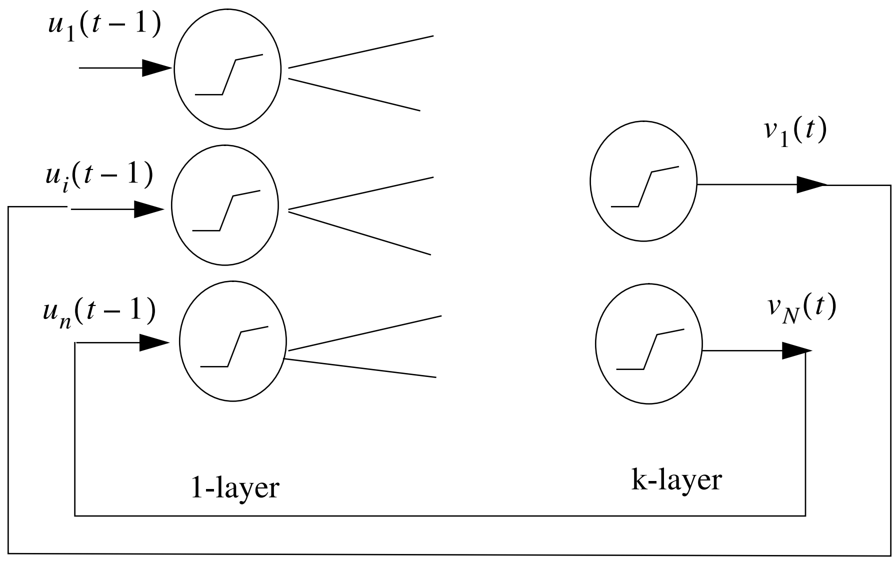

Figure 1. Basic RANN architecture.

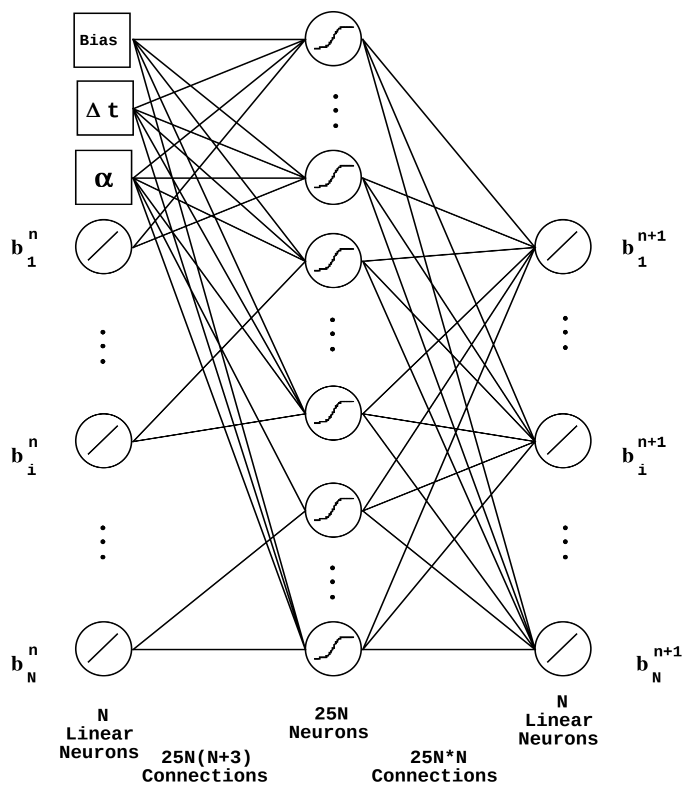

Figure 2. RANN architecture for the unsteady heat equation

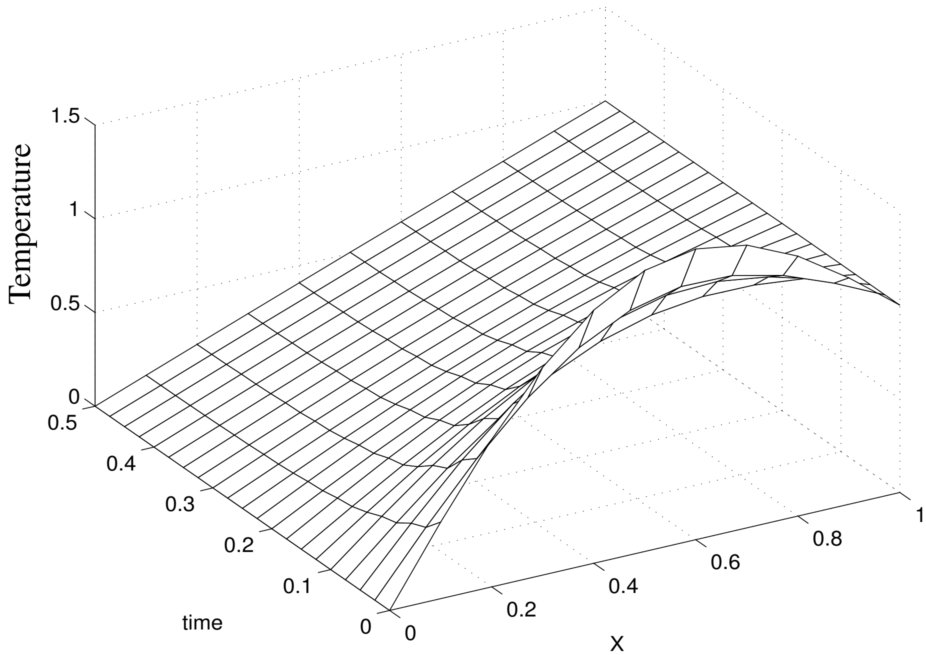

Figure 3. The response of the RANN model of the heat-conductive system

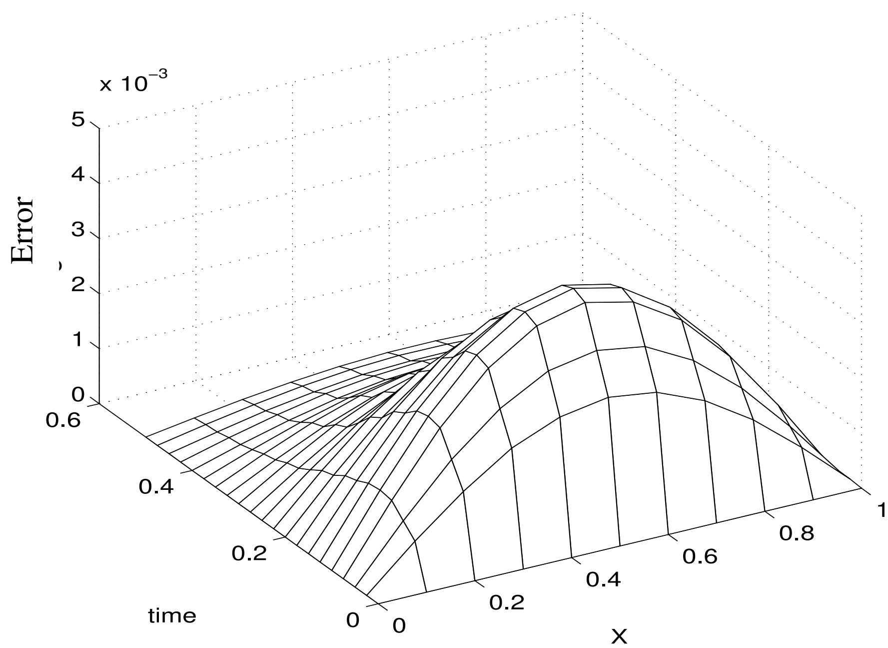

Figure 4. Error distribution of the RANN model of the heat-conductive system

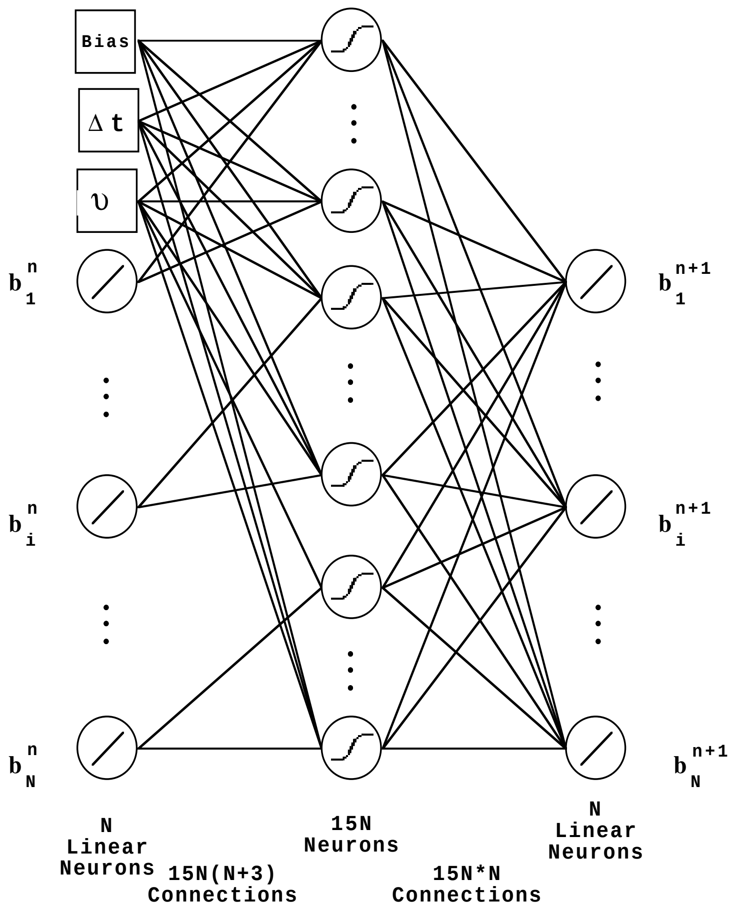

Figure 5. RANN architecture for the Burger's equation

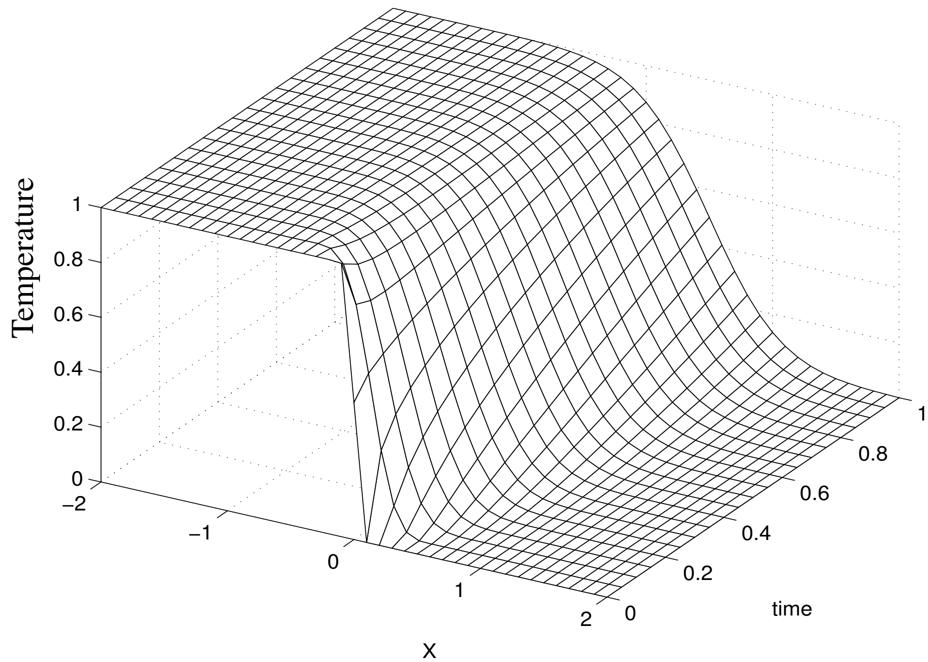

Figure 6. Response of the RANN model of the Burger's equation

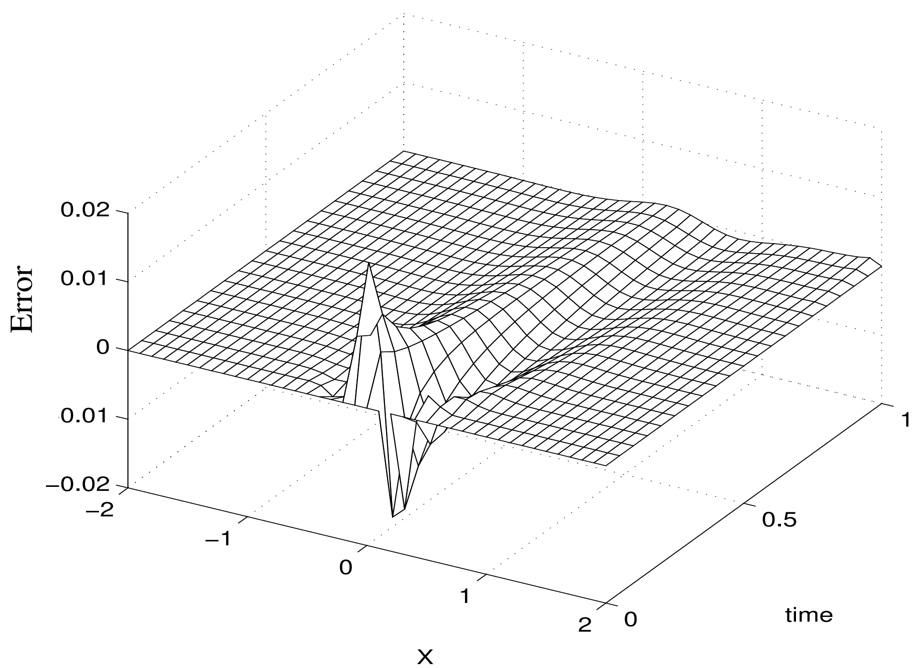

Figure 7. Error distribution of the RANN model of the Burger's equation