Advances in Adaptive Data Analysis
Vol. 3, No. 3 (2011) 385–400

DOI: 10.1142/S1793536911000684

World Scientific
www.worldscientific.com

AN INITIALIZATION METHOD FOR FEEDFORWARD ARTIFICIAL NEURAL NETWORKS USING POLYNOMIAL BASES

THANASIS M. VARNAVA* and ANDREW J. MEADE, JR.†

Department of Mechanical Engineering and Materials Science,
William Marsh Rice University, 6100 Main Street,
Houston, Texas, 77005-1892, the USA
**thanasis@rice.edu*
†*meade@rice.edu*

We propose an initialization method for feedforward artificial neural networks (FFANNs) trained to model physical systems. A polynomial solution of the physical system is obtained using a mathematical model and then mapped into the neural network to initialize its weights. The network can next be trained with a dataset to refine its accuracy. We focus attention on an elliptical partial differential equation modeled using a feedforward backpropagation network. We present a numerical example and compare our method with other initialization methods. Our method converges nearly 90% faster compared to random weights, with higher probability of convergence to an acceptable local minimum.

Keywords: Neural networks; neurocomputing; mathematical modeling; function approximation; weight initialization; polynomial bases.

1. Introduction

In aerospace and other engineering disciplines, engineers try to use all available data in order to model the behavior of a physical system. Most commonly, these systems are described by partial differential equations. With the advent of computers, engineers can now predict performance of a physical system before it has been built [Keane and Nair (2005)]. Of course, solutions provided by even the most advanced solvers are still approximations of the response of the system being analyzed, and experimental results or solutions from different numerical solvers may need to be combined in order to get a better approximation of the system's actual response. This is true especially in computationally restricting cases.

An area of interest is the adaptive data-driven emulation and control of engineering systems by the multi-layer perceptron, a paradigm of a feedforward artificial neural network (FFANN). It is known that neural networks can approximate functions and mathematical operators arbitrarily, as the number of neurons in the network tends to infinity [Cybenko (1989); Girosi and Poggio (1990); Hornik *et al.*

(1989); Ito (1991)]. In this regard, FFANNs can be considered as "universal approximators" capable of describing input–output relationships of engineering systems. For engineers, the universal approximation capability of networks could be especially useful in the emulation of complex mechanical systems using multiple sources of information through supervised artificial neural network (ANN) training. During the training, the ANN is shown samples of the function at certain sampling points and then attempts to construct an approximation of the function. The downside is that with noisy experimental data, it can be over-trained and the solution might not have the appropriate smoothness. In addition, it has been found that even the most advanced training algorithms can show unreliable convergence [Saarinen *et al.* (1993)], may converge to poor local minima and may not provide a satisfactory solution [Hecht (1990)], requiring training to be restarted with a different set of weights. Clearly, this is a waste of computational resources especially with large datasets and multiple inputs.

To address these issues, we develop a strategy in which network modeling of physical systems utilizes all available engineering information in order to improve the generalization abilities of the network and to improve the speed and probability of convergence to an acceptable local minimum, where the network approximation of the function is of adequate engineering accuracy.

With this proposed method, the network is trained, so that the network accurately approximates the input–output relationship of the system's mathematical model. The method can be useful in obtaining preliminary bounds on the network performance and for estimating the accuracy of the network approximation of the function. For example, by exploiting the analogy between FFANN systems and polynomial approximations, the effect of augmenting networks with additional processing elements can be evaluated. Training time is reduced and convergence probability increases. Finally, initializing training with a mathematical model of sufficient fidelity significantly reduces problems with approximating a physical system with a small training dataset that might be very accurate but with not enough training samples to provide a good response everywhere in the domain of the problem. With this approach, the network weights are predetermined by a mathematical model leading to a reliable approximation of a system for which we have regions lacking experimental data. It also reduces the effects of noisy experimental data on the smoothness of the response and can be used with data from different sources with good generalization abilities.

2. Background

2.1. *Feedforward artificial neural networks*

Any approximation algorithm that uses a weighted combination of basis functions and can be mapped into a graph-directed representation can be treated as an ANN [Poggio and Girosi (1990b)]. In this context, the response of an FFANN $f_a(\bar{x}, \bar{\beta})$ can

be seen as a general scheme of experimental data approximation [Cybenko (1989); Hornik *et al.* (1989)]

$$f_a(\bar{x}, \bar{\beta}) = \sum_i \bar{\beta}_k^i \sigma_{(k-1)} \left(\cdots \sigma_1 \left(\sum_{m=1}^{d} \bar{\beta}_m^1 \eta_m \right) \right), \tag{1}$$

where $\bar{x} = (\eta_1, \ldots, \eta_d)$ is the d-dimensional input of the FFANN system, $\bar{\beta}$ represents the set of network weights, and σ_j is the nonlinear transfer function of neurons associated with the jth intermediate layer of the network. In Eq. (1), the value $k-1$ can be thought of as the number of intermediate layers in a graph-directed representation of the neural network. A single hidden layer of neurons (Fig. 1) is commonly selected in network applications unless a problem-related consideration indicates that several layers are needed and may have beneficial impact on the network behavior. An extension to the multi-output case is straightforward.

Several types of transfer functions have been used in the neurons of ANNs, but the more popular training algorithms such as backpropagation require transfer functions to be continuous and bounded. As such, sigmoidal transfer functions are

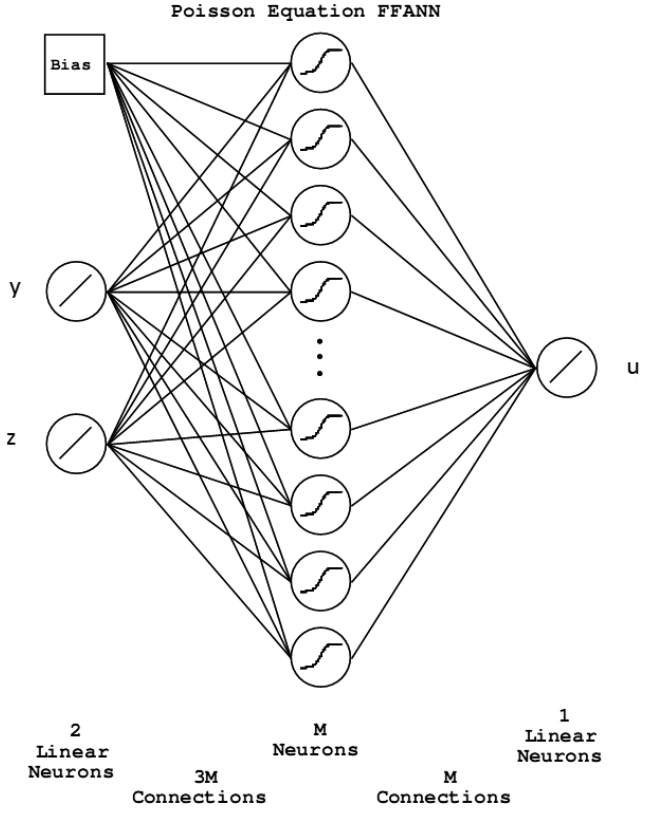

Fig. 1. Two-layer FFANN architecture with two inputs and one output.

often used. The hyperbolic transfer functional

$$\sigma(\xi) = \tanh(\xi). \tag{2}$$

is a paradigmic example of such sigmoidal functions and will be used in our ANN architecture for demonstrating our method.

2.2. *Training of FFANNs*

The training of an ANN system can be formulated as the procedure of selecting $\bar{\beta}$, so that the response of the network is "close" to the available experimental data, $f_e(\bar{x})$. Often, the network weights are obtained by minimizing the following error criterion

$$\varepsilon = \sum_{i=1}^{n} |f_e(x_i) - f_a, (x_i, \bar{\beta})|^2. \tag{3}$$

The network response $f_a(\bar{x}, \bar{\beta})$ depends linearly on only a few parameters, which are shown in Eq. (1) as the coefficients β_i^b of the expansion. The remaining parameters affect the network response in a nonlinear manner.

In terms of connectionist literature, numerical minimization of Eq. (3) by the steepest descent method constitutes the backpropagation algorithm [Rumelhart *et al.* (1986)]. The conjugate gradient method, the Levenberg–Marquardt method and other nonlinear optimization methods can be used for training [Saarinen *et al.* (1993)] each with its own merits and limitations.

ANN training requires these parameters to be specified beforehand, a process known as initialization. We will explore some common initialization techniques currently being used before we proceed with our method.

2.3. *Initialization schemes*

The initial values of the network weights, h_i, are commonly selected by random sampling initialization between the interval $(-0.05 < h_i < 0.05)$. This method provides a reference for other initialization techniques [Redondo and Espinosa (2001)]. If a satisfactory network response is not obtained after training due to convergence to a poor local minimum, the training procedure is restarted with new randomly selected values until a satisfactory network response is obtained [Schidt *et al.* (1993)]. Clearly this approach can require prohibitive computational resources for some practical problems, especially when dealing with large datasets or complex functions that are computationally expensive.

Kim and Ra [1991] proposed a minimum bound for their weight initialization scheme. The initial weights are chosen so that

$$|h_i| > \sqrt{\frac{\gamma}{d}} \tag{4}$$

where h_i are the initial weights, γ is the ANN, learning rate of the ANN, and d is the number of inputs of the network.

Alternatively, Lehtokangas *et al.* [1995] proposed that an orthogonal least squares algorithm be used for the initialization of the network weights. In this case, a large network is constructed by first randomly sampling the nonlinear network parameters and then determining the corresponding optimal values of the weights. Subsequently, the size of the network is reduced by pruning the neurons that do not significantly contribute to the network approximation. The deficiency in this approach is the necessity for determining a large intermediate network that can adequately emulate the system of interest; this can still require significant computational effort, and therefore, diminishes the applicability of the method. Moreover, the size of the network is not of primary concern for a significant class of practical problems if an accurate approximation can be obtained by adjusting only the weights. In this regard, Burrows and Niranjan [1993] proposed to initialize the network weights by linearizing the neuron transfer function, but this method is not satisfactory when emulating highly nonlinear input–output relationships. Husken and Goerick used evolutionary algorithms to select a good set of weights from a set of weights obtained *a priori* for similar problems [2000], but this technique adds complexity and evolutionary algorithms require additional resources. Castillo *et al.* proposed a linear least squares training approach where the nonlinearity of the transfer functions is also adapted for optimal performance [2002]. The downside to this method is that it is only applicable to single layer networks.

When comparing weight initialization methods for function approximation, the speed of convergence and probability of successful convergence are important measures. The speed of convergence is measured by how many learning steps it takes for ANN to converge to an acceptable local minimum during training, described by a mean and standard deviation. The probability of convergence is the probability that the algorithm will converge to an acceptable local minimum with sufficiently small error rather than to a false local minimum, far away from the global minimum. In our simulations to prove the worthiness of our method, we compare it with the method of random initial weights, the standard for comparing initialization methods.

Before we proceed with the description of our method, we have to explore the similarity between FFANNs and polynomial approximations by considering basis functions that can be useful in approximating the response of a physical system.

2.4. *Basis functions*

We believe that initial values of the network parameters adequate for successful training can be partially determined by establishing a set of constraints on the neurons of the network such that the nonlinear neuron transfer functions form useful basis functions that adequately approximate the system. The response of

these types of neural networks can be written as a linear combination of N basis functions $\phi(\bar{x})$. Specifically,

$$f_a(\bar{x}, \bar{\beta}) = \sum_{i=1}^{N} \phi_i(\bar{x}) c_i. \tag{5}$$

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

(1) The bases should be assembled in a straightforward manner from ANN architectures with parameters that can be evaluated in a numerically stable manner.
(2) The bases should accurately approximate the desired functions.
(3) To evaluate the nonlinear 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.
(4) The bases should allow the user to control the accuracy of the network approximation.
(5) The developed bases should have the property of approximating desired functions using a small number of hidden layer neurons and requiring only a modest increase in the number of neurons to improve modeling accuracy (i.e., good convergence rate).

In this next section, global polynomial bases with the useful properties described are formed from linear combinations of network transfer functions. The weights associated with the bases are evaluated by linear optimization techniques.

3. Proposed Initialization Strategy

3.1. *Sigmoidal networks and polynomial bases*

The idea of mapping a conventional approximation scheme with well-established approximation properties into a neural network architecture has been explored in the literature [Girosi and Poggio (1990); Hornik *et al.* (1990)] for theoretical studies regarding the density and approximation properties of ANNs and the dependence of the network 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 methods will be mapped into a network architecture for the emulation of physical systems. Proceeding along these lines, attention is drawn to the formula

$$\psi_\rho(\bar{w}, \bar{x}) = \frac{\partial^{|p|}}{\partial w_1^{p_1} \cdots \partial w_s^{p_s}} \psi(\bar{w}^T \bar{x} + \theta) = \bar{x}^p \psi^{(|p|)}(\bar{w}^T \bar{x} + \theta), \tag{6}$$

where $(\cdot)^T$ denotes a vector transpose, $\bar{x}^p = \eta_1^{p_1} \cdots \eta_s^{p_s}$, p is a multi-index such that $|p| = p_1 + \cdots + p_s$, ψ is a nonlinear and nonpolynomial function, and superscript (p)

represents the order of the ordinary derivative ψ. Equation (6) shows that a polynomial $\bar{x}^p$ can be readily expressed as:

$$\bar{x}^p = (\psi^{(|p|)}(\theta))^{-1}\psi_p(0,\bar{x}), \tag{7}$$

where the bias θ is selected by the user. Therefore, by replacing the partial derivative $\psi_p(0,\bar{x})$ by an adequate finite difference approximation, one can accurately approximate the polynomial $\bar{x}^p$ by a finite linear combination of functions in the form $\psi(\bar{w}^T\bar{x}+\theta)$. The following finite difference scheme is given as an example

$$\psi_p(o,\bar{x}) \cong \mathrm{Y}(\bar{x}) = (2\Delta w)^{-|p|}\sum_{0\le\zeta\le p}(-1)^{|\zeta|}\binom{p}{\zeta}\psi((2\zeta-p)^T\Delta w\cdot\bar{x}+\theta), \tag{8}$$

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

$$\binom{p}{\zeta} = \prod_{j=1}^{s}\binom{p_j}{r_j}. \tag{9}$$

Note that the accuracy of the finite difference scheme of Eq. (8), within a bounded domain of Ω, is given by

$$\max|\mathrm{Y}(\bar{x}) - \psi_p(0,\bar{x})| \le C(\Delta w)^2. \tag{10}$$

By combining Eqs. (10) and (11) and replacing the function $\psi(\bar{x})$ by the neuron transfer function $\sigma(\bar{x})$, it can be shown that the polynomial $\bar{x}^p$ can be approximated by a network of $M = \prod_{j=1}^{s}(p_j+1)$ neurons. Specifically,

$$\bar{x}_a^p = (\sigma^{(|p|)}(\theta))^{-1}(2\Delta w)^{-|p|}\sum_{0\le\zeta\le p}(-1)^{|\zeta|}\binom{p}{\zeta}\sigma((2\zeta-p)^T\Delta w\cdot\bar{x}+\theta). \tag{11}$$

The error of this approximation can be made arbitrarily small by the selection of Δw and θ. For example, the linear function y can be approximated by y_a, where

$$y_a = (2\Delta w\sigma^{(1)}(\theta))^{-1}[\sigma(\Delta wy+\theta) - \sigma(-\Delta wy+\theta)]. \tag{12}$$

Similarly, to construct products of dependent variables,

$$\begin{aligned} y_a z_a = {} & (4\Delta w_y\Delta w_z\sigma^{(2)}(\theta))^{-1} \\ & \times\begin{bmatrix}\sigma(\Delta w_y y+\Delta w_z z+\theta)+\sigma(-\Delta w_y y-\Delta w_z z+\theta) \\ \quad -\sigma(-\Delta w_y y+\Delta w_z z+\theta)-\sigma(\Delta w_y y-\Delta w_z z+\theta)\end{bmatrix}, \end{aligned} \tag{13}$$

where Δw_y and Δw_z are constants specific to variables y and z, and multilayer approximations

$$y_{a2} = (2\Delta w\sigma^{(1)}(\theta))^{-1}[\sigma(\Delta wy_{a1}+\theta) - \sigma(-\Delta wy_{a1}+\theta)], \tag{14}$$

where y_{a1} is the approximation from Eq. (12) and y_{a2} is the result from the second layer of neurons.

Note that the developments of this subsection have been utilized by Leshno *et al.* [1993] and Mhaskar [1995] for establishing the approximation properties of some neural networks architectures. In the next subsection, we will combine mathematical models of mechanical systems to obtain a reliable approximation scheme for initializing a feedforward neural network.

3.2. *Developing a regularization functional using* a priori *mathematical model*

In constructing a regularization functional using an *a priori* mathematical model, a large class of mechanics problems can be described by the equations

$$Lf_0(\bar{x}) = g(\bar{x}), \quad \bar{x} \in \Omega \quad \text{and} \quad Bf_0(\bar{x}) = 0, \quad \bar{x} \in \partial\Omega, \tag{15}$$

where f_0 is the response of the mathematical model, L is a linear self-adjoined differential operator, $g(\bar{x})$ is some given function, and B is the boundary operator. The solution to Eq. (15) can also be determined by minimizing the following quadratic (energy) form:

$$\Lambda(f) = \frac{1}{2} l\langle f, f\rangle - \langle g, f\rangle, \tag{16}$$

where f is an arbitrary function satisfying the boundary operator B exactly and a quadratic symmetry energy form $l\langle\cdot,\cdot\rangle$ can be associated with the differential operator L. For example,

$$l\langle f(\xi), f(\xi)\rangle = \int_\Omega f(\xi) L f(\xi) d\xi, \tag{17}$$

where Ω represents the domain of interest for the dummy variable ξ. Since for any f that satisfies the boundary conditions

$$l\langle f_0, f\rangle = \langle g, f\rangle, \tag{18}$$

Then, substituting Eq. (18) into Eq. (16) yields

$$\Lambda(f) = \frac{1}{2} l\langle f - f_0, f - f_0\rangle - \frac{1}{2} l\langle f_0, f_0\rangle = \frac{1}{2} l\langle f - f_0, f - f_0\rangle + \text{ constant}. \tag{19}$$

From a physical perspective, the functional $\Lambda(f)$ penalizes the mechanical energy of the discrepancy between the function f and the response of the mathematical model f_0.

With the use of f and the energy functional of Eq. (6), regularized modeling can be formulated as the minimization of the objective function

$$\varepsilon(f) = \sum_{i=1}^{N} |f_e(x_i) - f(x_i)| + \lambda \cdot \left(\frac{1}{2} l\langle f, f\rangle - \langle g, f\rangle \right), \tag{20}$$

which combines information from experimental data and the *a priori* mathematical model. Note that λ represents the degree of reliability of the mathematical model compared to that of the experimental data.

To determine $f_a(\bar{x}, \bar{\beta})$, Eq. (20) is augmented since an arbitrary $\bar{\beta}$ will not satisfy the boundary conditions. Specifically,

$$\varepsilon(f_a) = \sum_{i=1}^{N} |f_e(x_i) - f(x_i, \bar{\beta})| + \lambda \cdot \left(\frac{1}{2} l\langle f_a, f_a \rangle - \langle g, f_a \rangle + b\langle f_a, Q \rangle \right), \tag{21}$$

where $b\langle \cdot, \cdot \rangle$ may be defined as $b\langle f_a, Q \rangle = \int_{\partial\Omega} (B f_a(\xi)) Q(\xi) d\xi$ on the domain boundary $\partial\Omega$.

Using the first part of the proposed initialization strategy, bases are formed from the transfer functions, so that $f_a$can be described by Eqs. (4) and (21) can be minimized with respect to the remaining unknowns, c_i, resulting in

$$\sum_{j}^{M} \left(l\langle \phi_j(\bar{x}), \phi_k(\bar{x}) \rangle + \frac{2}{\lambda} \sum_{i=1}^{N} \phi_j(x_i) \phi_k(x_i) \right) c_j + b\langle \phi_k(\bar{x}), Q \rangle$$

$$= \langle g, \phi_k(\bar{x}) \rangle + \frac{2}{\lambda} \sum_{i=1}^{N} f_e(x_i) \phi_k(x_i) \quad \text{for } k = 1, \ldots, M \tag{22}$$

and

$$\sum_{j}^{M} c_j b\langle \phi_j(\bar{x}), P \rangle = 0, \tag{23}$$

where P and Q belong to the appropriate finite dimensional subspace.

In order to initialize the neural network weights with an *a priori* mathematical model, the available data $f_e(x_i)$ are neglected by allowing $\lambda \to \infty$, so that

$$\sum_{j}^{M} l\langle \phi_j(\bar{x}), \phi_k(\bar{x}) \rangle c_j + b\langle \phi_k(\bar{x}), Q \rangle = \langle g, \phi_k(\bar{x}) \rangle, \quad \text{for } k = 1, \ldots, M. \tag{24}$$

The values of c_j that satisfy Eqs. (23) and (24) can then be determined. That is $f_0(\bar{x})$ is approximated by the network to an acceptable engineering accuracy. In the computational literature, Eqs. (23) and (24) are known as the Galerkin method [Xiao *et al.* (2006)]. This observation provides a useful link between the initialization of FFANNs and methods in computational mechanics.

Extending this link with computational mechanics, Eq. (24) can be rewritten with greater generality, and without loss of accuracy, as

$$\sum_{j}^{N} \langle \Psi_k(\bar{x}), R(\bar{x}) \rangle + b\langle \Psi_k(\bar{x}), Q \rangle = 0, \quad \text{for } k = 1, \ldots, N$$

$$\text{and } R(\bar{x}) = L\phi_j(\bar{x}) c_j - g, \tag{25}$$

where Ψ_k is an arbitrary weighting function and $R(\bar{x})$ is the equation residual. Equation (25) facilitates the utilization of the method of weighted residuals [Finlayson (1972)], which encompasses many of the popular conventional computational methods.

With the solution of the coefficients c_j, and the preselected values of $\bar{\beta}$, the approximation f_a can be realized as a two-layer FFANN. The details will be clarified with an example below.

4. Numerical Example

A numerical example is presented in order to demonstrate the usefulness and applicability of the proposed initialization method for mechanical applications. This example involves modeling fully developed flow in a square duct, described by the Poisson equation with homogeneous boundary conditions.

4.1. *Problem formulation*

Assume that we wish to use a neural network to emulate and/or control steady incompressible viscous flow through a square cross section as shown in Fig. 2.

Far from the inlet or exit of the channel, the flow does not vary in the streamwise direction η, the crossflow velocity components are zero and the pressure variation is constant (α). Using a Cartesian coordinate system, the fluid flow for this example is governed by the momentum nonlinear partial differential equation [Canto *et al.* (1988)],

$$\frac{\partial^2 u}{\partial y^2} + \frac{\partial^2 u}{\partial z^2} = -\frac{1}{\mu}\frac{\partial p}{\partial x} = -\alpha, \tag{26}$$

where u, ρ, p, and μ represent the streamwise component of fluid velocity, the fluid density, pressure, and viscosity, respectively. This kind of flow can, therefore, be described by the Poisson equation for the dimensionless velocity $u^* = u(\alpha/L^2)$ and the dimensionless normalized lengths $y^* = y/L$ and $z^* = z/L$,

$$\frac{\partial^2 u}{\partial y^2} + \frac{\partial^2 u}{\partial z^2} + 1 = 0, \tag{27}$$

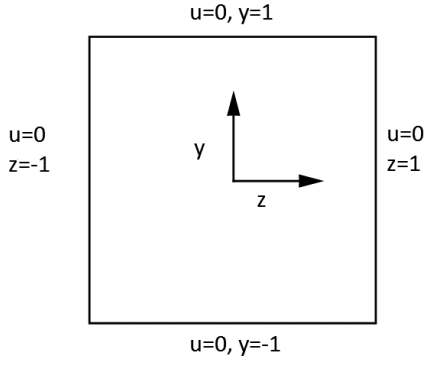

Fig. 2. Problem domain and boundary conditions for flow through a square duct problem.

where the stars have been dropped for notational convenience. Further, Eq. (5) is augmented by the following boundary conditions

$$u = 0 \quad \text{on } y = 1, \quad y = -1, \quad z = 1 \quad \text{and} \quad z = -1, \tag{28}$$

The dimensionless dependent variable u is approximated in this problem by

$$\begin{aligned} u_a(y,z) \approx u_{\text{poly}} &= \sum_{i=1}^{N1} \sum_{j=1}^{N2} (y^{(i-1)} - y^{(i+1)})(z^{(j=1)} - z^{(j+1)}) c_r \\ &= \sum_{r=1}^{N1 \cdot N2} \phi_r(y,z) c_r, \end{aligned} \tag{29}$$

where u_{poly} satisfies the boundary conditions identically and $r = N1 \cdot (j-1) + i$. Note that since the FFANN can only approximate polynomials, we must distinguish between the FFANN approximation u_a and the approximation by polynomial expansion u_{poly}. Substitution of Eq. (29) into Eq. (27) results in

$$\sum_{r=1}^{N1 \cdot N2} \left(\frac{\partial^2 \phi_r}{\partial y^2} + \frac{\partial^2 \phi_r}{\partial z^2} \right) c_r + 1 = R(y,z), \tag{30}$$

where $R(y,z)$ is the residual of the equation.

Using the method of weighted residuals Eq. (30) can be rewritten as a series of algebraic equations. MATLAB's PDE toolbox was used to obtain the finite difference solution using 676 grid points and a second order polynomial approximation to the square duct problem was obtained as

$$\begin{aligned} u_{\text{poly}} = &-0.187y^2 - 0.187z^2 - 3.25 \times 10^{-4} yz - 1.77 \\ &\times 10^{-4} y - 1.77 \times 10^{-4} z + 0.0267. \end{aligned} \tag{31}$$

The polynomial approximation u_{poly} was then mapped into an FFANN using the technique introduced in Sec. 3.1 with $\Delta w_y = \Delta w_z = 0.02$ and $\theta = -0.695$.

4.2. *FFANN formulation*

A two-layer perceptron with two input neurons, one hidden layer consisting of 16 neurons, and one output neuron was developed, using hyperbolic tangent transfer functions in the hidden layer and a linear output transfer function. The response of the initialized network is shown in Fig. 3 and the absolute error is shown in Fig. 4. Note that the ANN approximation does not satisfy the boundary conditions exactly, and most of the error is distributed near the domain boundaries. Since the boundary conditions are known, we can replace the ANN approximation at the boundary with the correct known values.

The same numerical solution used to obtain the polynomial approximation u_{poly} was used as a dataset in order to compare our initialization method with random initial weights. The neural network was trained 200 times with initialization and

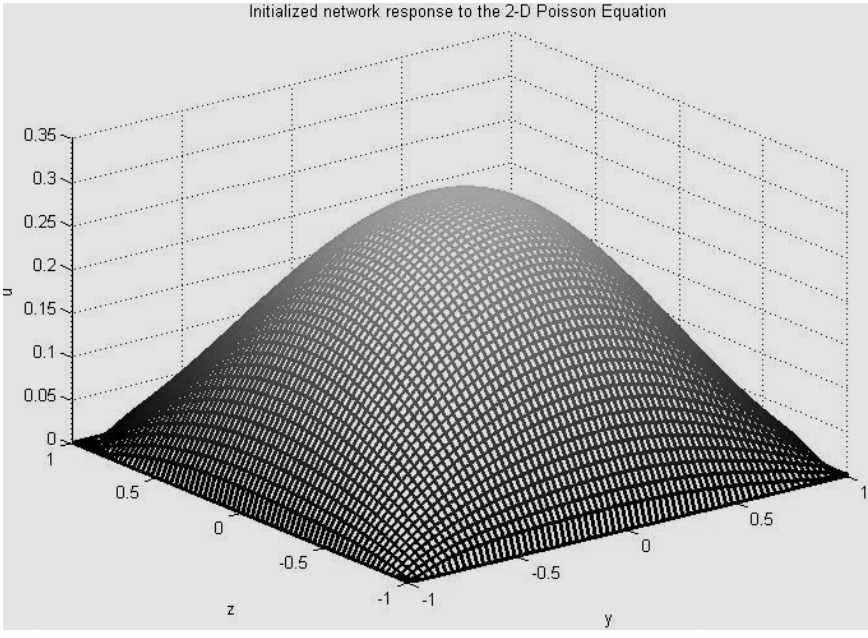

Fig. 3. Initialized FFANN response to the 2D Poisson equation.

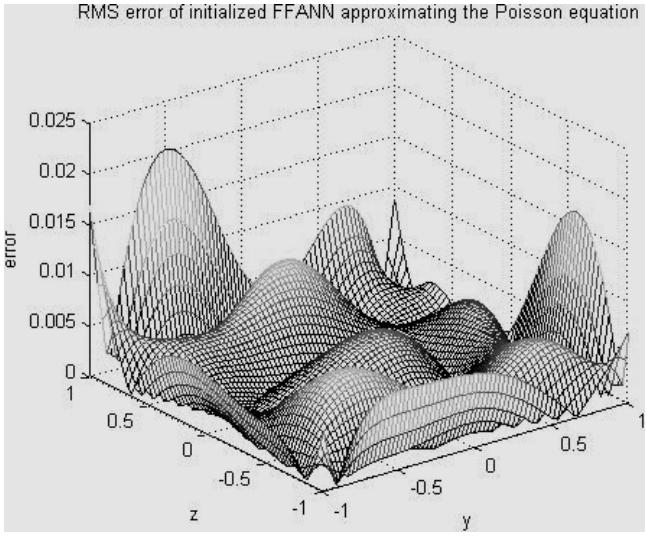

Fig. 4. Error distribution of initialized FFANN approximating the Poisson equation.

Table 1. Convergence rate and convergence probability comparison between the proposed method and random weights.

	Proposed method	Random weights
Convergence rate	89	721+/−69
Convergence probability	100%	92.5%

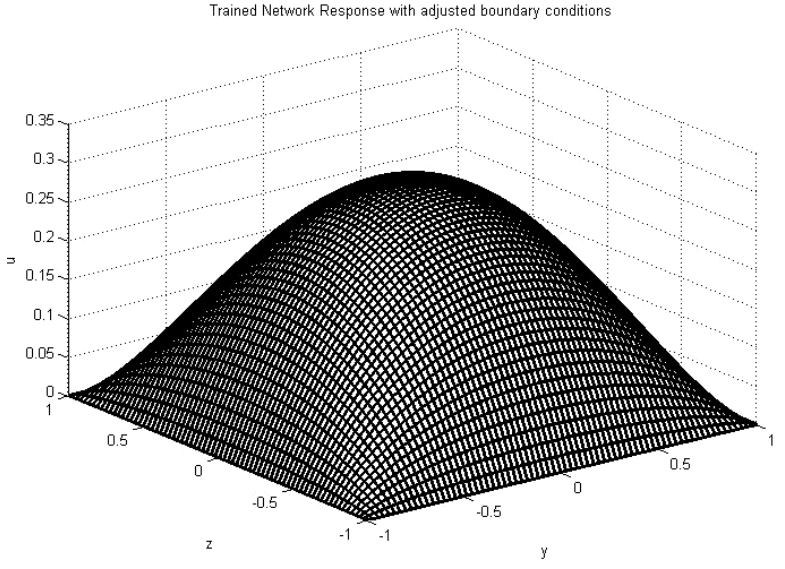

Fig. 5. Response of the initialized network after training with data with adjusted boundary conditions.

200 times with random weights. The same learning rate was used. Probability of convergence was measured as the number of training runs that converged to a mean square error of less than 3×10^3 divided by the total number of runs. Convergence rate was measured by the number of epochs needed for the ANN to converge to an MSE value of less than 3×10^3 if it successfully did so. Each epoch is defined as the presentation of the entire dataset to the ANN during learning. The results can be seen in Table 1.

Clearly the network achieves convergence much faster than with the standard method of random initial weights and has a higher probability of converging to an acceptable local minimum. The minimum RMS error achieved with initialized weights was 1.78×10^3 and 1.81×10^3 with random weights. The response of the trained network is shown in Fig. 5.

5. Conclusion

The adaptive data-driven emulation and control of mechanical systems are popular applications of ANNs in engineering. Since ANN training is a nonlinear optimization

problem, it is difficult to apply this attractive modeling tool to engineering applications. As a remedy, a new initialization method for connectionist algorithms has been presented. The method is formulated on building global polynomial bases from linear combinations of smooth transfer functions and then mapping the polynomial bases into an FFANN architecture. *A priori* mathematical models of engineering systems can then be used to obtain initial values of the network weights. The method we propose allows for the initialization of the ANN without training data. In addition, 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 method, the network parameters are predetermined by a mathematical model leading to a reliable approximation of a physical system in regions lacking data. To prove the usefulness and accuracy of this method, an elliptical differential equation was solved using a two-layer perceptron with hyperbolic tangent transfer functions in the processing elements and compared with the standard method of random initial weights. The initialized network converged to an acceptable solution much faster than with random weights, and had a higher probability of convergence.

Acknowledgments

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

References

Burrows, T. L. and Niranjan, M. (1993). *The Use of Feedforward and Recurrent Neural Networks for System Identification*. Technical Report CUED/F-INFENG/TR158, Cambridge University.

Canto, C., Hussaini, M. Y., Querteroni, A. and Zang, T. A. (1987). *Spectral Methods in Fluid Dynamics*, Springer-Verlag, Berlin Heidelberg.

Castillo, E., Fontenla-Romero, O., Alonso-Betanzos, A. and Guijjaro-Berdinas, B. (2002). A global optimum approach for one-layer neural networks. *Neural Computat.*, **14**: 1429–1449.

Cybenko, G. (1989). Approximations by superposition of sigmoidal functions. *Math. Cont. Sign. Syst.*, **2**: 303–314.

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

Girosi, F. and Poggio, T. (1990). Networks and the best approximation property. *Biol. Cybern.*, **63**: 169–176.

Girosi, F., Jones, M. and Poggio, T. (1995). Regularization theory and neural network architectures. *Neural Comput.*, **7**: 219–269.

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

Hornik, K., Stinchcombe, M. and White, H. (1989). Multilayer feedforward networks are universal approximators. *Neural Networks*, **2**: 359–366.

Hornik, K., Stinchcombe, M. and White, H. (1990). Universal approximation of an unknown mapping and its derivatives using multilayer feedforward networks. *Neural Networks*, **3**: 551–560.

Husken, M. and Goerick, C. (2000). Fast learning for problem classes using knowledge based network initialization. *Proceedings of International Joint Conferecne on Neural Networks*, pp. 619–624.

Ito, Y. (1991). Representation of functions by superpositions of a step or sigmoidal function and their applications to neural network theory. *Neural Networks*, **4**: 385–394.

Keane, A. J. and Nair, P. B. (2005). *Computational Approaches for Aerospace Design*: *The Pursuit of Excellence*, Wiley, UK.

Kim, Y. K. and Ra, J. B. (1991). Weight value initialization for improving training speed in the backpropagation network. *Proceedings of International Joint Conference of Neural Networks*, **3**: 2396–2401.

Lehtokangas, M., Saarinen, J., Kaski, K. and Huuntanen, P. (1995). Initializing Weights of a multilayer perceptron network by using orthogonal least squares algorithm. *Neural Comput.*, **7**: 982–999.

Leshno, M., Lin, V. Y., Pinkus, A. and Schocken, S. (1993). Multilayer feedforward networks with a non-polynomial activation function can approximate any function. *Neural Networks*, **6**: 861–867.

Mhaskar, H. N. (1996). Neural networks for optimal approximation of smooth and analytic functions. *Neural Comput.*, **8**: 164–177.

Poggio, T. and Girosi, F. (1990a). Networks for approximation and learning. *Proc. IEEE*, **78**(9): 1481–1497.

Poggio, T. and Girosi, F. (1990b). Regularization algorithms for learning that are equivalent to multilayer networks. *Science*, **247**: 978–982.

Redondo, F. M. and Espinosa, C. H. (2001). Weight initialization methods for multilayer feedforward networks. *ESANN'2001 Proceedings — European Symposium on Artificial Neural Networks*, Bruges (Belgium), pp. 25–27

Rumelhart, D., Hinton, G. E. and Williams, R. J. (1986). Learning internal representations by error propagation. *Parallel Distributed Processing: Exploration in the Microstructure of Cognition*, eds. D. Rumelhart, J. L. McClelland and the PDP Research Group, MIT Press, Cambridge, MA, pp. 318–362.

Saarinen, S., Bramley, R. and Cybenko, G. (1993). Ill-conditioning in neural network training problems. *SIAM J. Sci. Comput.*, **14**(3): 693–714.

Schidt, W. F., Raudys, S., Kraaijveld, M. A., Skurikhina, M. and Duinn, R. P. W. (1993). Initializations, backpropagation and generalization of feedforward classifiers. *Proceedings of IEEE International Conference on Neural Networks, ICNN-93*, **1**: 598–604.

Xiao, R., Wu, C. and Feng, W. (2006). Neural network method for solving elastoplastic finite element analysis. *J. Zhejiang Univ. Sci. A*, **7**(3): 378–382.

Appendix

Nomenclature

Symbols	Explanation
$\mathrm{B}(\cdot)$	Boundary operator of the mathematical model
$b\langle\cdot,\cdot\rangle$	Symmetric form associated with the boundary operator B
c_m	Basis expansion coefficients
$\bar{c}$	Vector with components c_m
d	Data dimension (number of input neurons)
F	Response of the physical system
f	Arbitrary function satisfying the boundary operator B exactly

$\underline{f_0}$	Response of the mathematical model
$\overline{f_0}$	Vector with components $f_0(x_j)$
f_a	Network response
$\underline{f_e}(x_i)$	Experimental measurements of $F(x_i)$
$\overline{f_e}$	Vector with components $f_e(x_i)$
$G(\cdot,\cdot)$	Green's function of the mathematical model
$\mathbf{G}$	Matrix with components $G(x_k, x_m)$
g	Algebraic function
$\mathbf{I}$	Identity matrix
$L(\cdot)$	Self-adjoined differential operator
$l\langle\cdot,\cdot\rangle$	Quadratic symmetric energy form associated with L
R^d	Dimensional Euclidean space
s	Parameter related to the operator L where $s > 0$
$x = (\eta_1, \ldots, \eta_d)$	d-dimensional spatial coordinate
$\langle\cdot,\cdot\rangle$	Inner product

Greek	**Symbols**
$\bar{\beta}$	Set of general ANN weights/set of basis function expansion coefficients
γ	ANN learning rate
$\varepsilon(\cdot)$	Objective function
$\Lambda(\cdot)$	Smoothness based Tikhonov regularization functional
ξ	Arbitrary variable
ρ	Positive constant
σ	Basis function
$\Phi(\cdot)$	Energy functional of the mathematical model
ϕ	Arbitrary function
ψ	Arbitrary function
Ω	Spatial domain of the mathematical model
$\partial\Omega$	Spatial boundary of the mathematical model