INFORMATION TO USERS

This manuscript has been reproduced from the microfilm master. UMI films the text directly from the original or copy submitted. Thus, some thesis and dissertation copies are in typewriter face, while others may be from any type of computer printer.

The quality of this reproduction is dependent upon the quality of the copy submitted. Broken or indistinct print, colored or poor quality illustrations and photographs, print bleedthrough, substandard margins, and improper alignment can adversely affect reproduction.

In the unlikely event that the author did not send UMI a complete manuscript and there are missing pages, these will be noted. Also, if unauthorized copyright material had to be removed, a note will indicate the deletion.

Oversize materials (e.g., maps, drawings, charts) are reproduced by sectioning the original, beginning at the upper left-hand corner and continuing from left to right in equal sections with small overlaps.

Photographs included in the original manuscript have been reproduced xerographically in this copy. Higher quality 6" x 9" black and white photographic prints are available for any photographs or illustrations appearing in this copy for an additional charge. Contact UMI directly to order.
NOTE TO USERS

This reproduction is the best copy available.
RICE UNIVERSITY

THE ARTIFICIAL NEURAL NETWORK SOLUTION TO DOUBLE DIFFUSIVE CONVECTION EQUATIONS USING SPECTRAL METHODS

by

Powtawche Neengay Williams

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE Master of Science

APPROVED, THESIS COMMITTEE:

Dr. Andrew J. Meade Jr., Chairman
Associate Professor
Mechanical Engineering and Materials Science

Dr. S. Scott Collis
Assistant Professor
Mechanical Engineering and Materials Science

Dr. Richard A. Tapia
Noah Harding Professor
Computational and Applied Mathematics

Houston, Texas
May 2000
THE ARTIFICIAL NEURAL NETWORK SOLUTION TO DOUBLE DIFFUSIVE CONVECTION EQUATIONS USING SPECTRAL METHODS

Powtawche Neengay Williams

Abstract

Artificial neural networks have primarily been utilized to solve problems in pattern recognition, decision-making, signal analysis and controls. This thesis investigates the use of networks in modeling physical systems in fluid mechanics by using the governing partial differential equations to initialize the network parameters. The initialization 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. This approach is shown to be capable of incorporating smooth neuron transfer functions, such as the popular hyperbolic tangent. Attention is focused on the two-dimensional Navier–Stokes equations for Boussinesq convection that model two-dimensional double diffusive convection. The network used to model this example utilizes an approximation of the Gudermannian function and an application of the pseudospectral method for complete network initiation. Numerical examples are presented illustrating the accuracy and utility of the method.
Acknowledgments

I would like to express my sincerest gratitude to my advisor, Dr. Andrew Meade, Jr., for his kind patience, insight, and guidance during this work. I give great appreciation to my committee members, Dr. S. Scott Collis and Dr. Richard A. Tapia, for their counsel, motivation, and review of my thesis.

I am thankful for the advice and assistance from the members of my research group, especially Greg S. Lind and Alvaro A. Fernandez. I also wish to thank Dr. Anil Deane for his advice and the use of his double diffusive convection numerical code.

I would like to thank my parents, Byron C. and Geraldine B. Williams, my siblings, Ta-Shina, Craig and Tahlako, and all members of my extended family for their prayers, continuous love, and encouragement.

I must also give thanks to all my fellow graduate students, friends, and colleagues whose support never faltered.

Most of all, I give thanks and praise to God, the Almighty, who has blessed me with this experience and has made everything possible. Hoke!!

This research work was supported by the NASA-GEM Fellowship and the Rice University Department of Mechanical Engineering and Materials Science.
# Table of Contents

Abstract

Acknowledgments

Nomenclature

1 Introduction

1.1 System Simulation and Surrogates ........................................... 1

1.2 Thesis Organization .......................................................... 5

2 Background Information

2.1 Artificial Neural Network Development ................................. 7

2.2 Network Architecture and Training ....................................... 9

2.3 Conventional Construction of a ANN ................................... 14

3 Network Initiation and Basis Construction ............................ 16

3.1 Tikhonov Regularization ...................................................... 16

3.2 Construction of Global Bases from Transfer Functions .......... 19

3.3 Construction of Trigonometric Bases .................................. 21

4 Modeling Partial Differential Equations ............................... 27

4.1 Feedforward Artificial Neural Networks ............................... 27

4.2 Recurrent Artificial Neural Networks .................................. 29

4.3 FFANN/RANN Hybrid ......................................................... 30

5 The Double Diffusive Convection Problem ............................ 32
5.1 Double Diffusive Convection ........................................ 32
  5.1.1 Governing Equations ........................................ 34
5.2 Numerical Model .................................................. 37
5.3 Construction of the ANN .......................................... 39
5.4 Results ............................................................. 41

6 Conclusions and Future Work ................................... 47

Bibliography ............................................................ 50

A The ANN Modeling Subroutine .................................. 54
Nomenclature

\( d \) vertical height of diffusion  
\( g \) gravitational acceleration  
\( gd(\cdot) \) Gudermannian polynomial  
\( \vec{k} \) unit vector in the \( z \)-direction  
\( Pr \) Prandtl number  
\( R_S \) solute Rayleigh number  
\( R_T \) thermal Rayleigh number  
\( S_m \) solute mean  
\( S_{total} \) solute total  
\( S_x \) solute in \( x \)-direction  
\( \Delta S \) solute difference  
\( T_m \) mean temperature  
\( T_{total} \) total temperature  
\( \Delta T \) temperature difference  
\( u \) velocity component in the \( x \)-direction  
\( u_k \) summation vector component of \( \rho_m \) weights times the input  
\( V \) velocity  
\( w \) velocity component in the \( z \)-direction  
\( w_{kj} \) weight matrix component  
\( x \) spatial coordinate (horizontal)  
\( x_j \) ANN input vector component  
\( y_k \) ANN output vector component  
\( z \) spatial coordinate (vertical)  

Symbols

\( \alpha \) solute coefficient of expansion  
\( \beta \) temperature coefficient of expansion  
\( \delta_{ij} \) Kronecker delta  
\( \kappa_T \) temperature molecular diffusivity  
\( \kappa_S \) solute molecular diffusivity  
\( \varphi \) activation function  
\( \nu \) kinematic vorticity  
\( \Phi \) basis function  
\( \psi \) stream function  
\( \rho \) density distribution  
\( \tau \) ratio of diffusivities  
\( \theta_k \) threshold function  
\( \vec{\omega} \) vorticity vector

Subscripts
$i, j, k, m$  dummy indices

**Superscripts**

$n$  time level
Chapter 1

Introduction

1.1 System Simulation and Surrogates

In general, engineering can be described as the application of basic science to the design of systems that address physical problems in society. These systems can range from transportation of people and material (e.g., cars, locomotives, aircraft) to the control of local environments (e.g., heating, ventilation and air conditioning).

A fundamental and irreplaceable stage of the design of a system is its simulation. This stage of design can best be illustrated using a tool known as an information-flow diagram. As an example, the fluid-flow diagram of Figure 1.1 shows the components of a simple fire hydrant system in a petrochemical plant. The parameters \( p, w, \) and \( h \) are the pressures, volume flow rates, and pump elevation, respectively. A workable information-flow diagram is illustrated in Figure 1.2. Each block in Figure 1.2 represents a machine or physical component that can be represented with a map-

![Figure 1.1: Fire hydrant system and pump characteristics [30].](image-url)
Figure 1.2: Information-flow diagram for the fire hydrant system [30].

In automatic-controls each block represents a transfer function or plant. In field of multidisciplinary design optimization (MDO) the blocks are known as surrogates, which is the term we will use for the remainder of this text.

In MDO these surrogates can be continuous, discrete, deterministic, stochastic, steady, time–dependent, physical or even economic. The mapping function, \( f \), can be an analytic expression from theory, tabular data from experiments or even a system of algebraic equations from a computational mechanics algorithm. In the perspective of system simulation, one can say that engineering analysis is the determination of the mapping function of the surrogate when it represents a physical phenomena of interest. This can range from the performance characteristics of a pump to the lift and drag on a wing.

As previously mentioned, development of a surrogate, and hence the analysis of a physical phenomena of interest, includes the use of physical experimentation, numerical schemes or mathematical modeling. While using only one of these methods provides feasible approximations of the problem (i.e., mappings), solutions of better fidelity can be obtained through the combination of all three (Figure 1.3). This
Figure 1.3: Diagram of optimal engineering analysis.

combination also resolves any issues associated with modeling complex engineering problems. For example, ideally, experimental measurements should provide the most accurate description of a physical system. However, from an engineering standpoint, experimental studies lack the needed generality since measurements only describe the system behavior under a specific combination of input and system parameters. Also, errors in experimental measurements can induce a bias in the evaluation of the physical system. This may conceal important features in the system behavior. Most importantly, experimental measurements of complex physical systems may require significant, if not prohibitive, material and labor costs; describing complex engineering systems using sparse experimental data is an important and challenging engineering problem.

Alternatively, utilizing only mathematical models for engineering analysis can yield general results which can then provide the foundation for a parametric study. However, even the most sophisticated mathematical model can only approximate the physical system. In this regard, the fidelity of mathematical models and its applicability for describing system behavior ultimately requires model validation using
experimental measurements. In addition, the majority of sophisticated mathematical models do not have analytical solutions and must be simplified (low-fidelity) or solved by high-speed digital computers combined with accurate numerical algorithms. Unfortunately, even with the use of numerical computation only a limited number of parameter combinations can be investigated and, like the experimental approach, the theoretical approach loses its generality.

Though it is known in the engineering community that successful analyses rest upon the proper balance of all three approaches, it is noted that few attempts have been made in uniting experimental, theoretical and numerical methods in the literature. At present, an area of increasing interest is the construction of physically-based surrogates by adaptive data-driven emulation using artificial neural networks (ANNs).

It is known that ANNs, also known as neural networks, can approximate functions and mathematical operators arbitrarily well as the number of neurons in the network tends to infinity [4, 12]. In this regard, ANNs can be considered to be "universal approximators" capable of describing the input-output mapping of physical systems. For engineers, the universal approximation capability of networks would be especially useful in the emulation of complex mechanical systems through ANN training.

Unfortunately, ANNs have not found wide acceptance among engineers since to emulate a physical system through conventional training algorithms an engineer must either follow a rule based approach, train the ANN with data from a similar, though less complex or expensive, physical system, or train with data from a numerical simulation. However, the ANN need not learn the physical relationships to map input to output. The engineer may wish to use functional relationships from the physical problem to derive additional information of interest (e.g., velocity from displacement). In addition, ANN training, which will be discussed later in the thesis, is an ill-posed nonlinear optimization problem and as such has problems with local minima [11],
slow convergence or nonconvergence [27], and stability with respect to errors in data measurements. Lastly, major concerns shared by all users of ANNs in both hard and soft computing are: (1) the number of neurons and neuron layers needed for emulation (ANN architecture), (2) the interpretation of the connection weight matrix, (3) the amount of data sufficient to model desired behavior, and (4) the reliability of ANNs in interpolating from training data.

It is believed one approach to addressing all of these concerns is to determine the ANN architecture and connection weights by incorporating a mathematical model of the physical system (of some fidelity) and reinforce the ANN with observational data by regularization. Consequently, the objective of this thesis is to both show how the problems of data-driven ANN implementation can be resolved by replacing the Tikhonov regularization functional of Poggio and Girosi [24] with a functional corresponding to a mathematical model of the mechanical system of interest and to show that well-behaved polynomial bases can be approximated through constraints on the nonlinear network parameters and nonlinear transfer functions.

Since Meade and Zeldin have already investigated the use of Tikhonov regularization in merging mathematical models with experimental data and the construction of local bases from ANN transfer functions [22, 33], this thesis continues progress made on forming global bases proposed in Meade and Moreno [21]. In particular, this thesis concentrates on the emulation of a two-dimensional double diffusive convection (DDC) fluid system using a hybrid feedforward/recurrent ANN initiated with the Navier–Stokes equations for Boussinesq convection and the pseudospectral method.

1.2 Thesis Organization

Chapter 2 includes background information on artificial neural networks and their training. Chapter 3 discusses the Tikhonov regularization and global basis construc-
tion methods used to constrain nonlinear parameters in an ANN for the solution of engineering problems. Chapter 4 describes the ANN topologies which model partial differential equations (PDEs) of elliptic, parabolic and hyperbolic form and also discusses a method for combining canonical ANN topologies. Chapter 5 presents the physical phenomenon of two-dimensional double diffusive convection and derives the construction of the ANN solution method used to solve the problem. Results and conclusions of this research are discussed in Chapters 5.4 and 6, respectively. The ANN modeling subroutine is listed in the appendix.
Chapter 2

Background Information

The following presents general background information on ANNs and their construction. The first section introduces the physical model of the biologically-inspired ANN. The second section discusses the two major types of ANN topologies used with supervised training and the third section maps out the procedure used in training a ANN.

2.1 Artificial Neural Network Development

Developed by McCulloh and Pitts in 1943, the processing unit of an artificial neural network is modeled after the basic cell of the biological nervous system which is the neuron (Figure 2.1). The three components of a biological neuron are dendrites (input receivers), soma (neuron body), and axon (output transmitter). As an information processor, the neuron receives input signals from dendrites, then results are summed and compared to an excitatory boundary limit. If the threshold value is met, an information transfer occurs by firing an impulse within the neuron. The information transfer can be classified as an excitatory or inhibitory synapse that determines if the information will be sent out of the neuron and evaluated by the rest of the biological neural network.

Information is processed similarly in an artificial neural network which is comprised of many neurons containing a summing junction, $u_k$, that multiplies inputs, $x_k$, with synaptic weights, $w_{kj}$ (Figure 2.2).
Figure 2.1: Model of a biological neuron.

Figure 2.2: An artificial neuron whose input signals, activation function, and output are modeled respectively after the dendrites, soma, and axon of a biological neuron.
The activation function, \( \varphi \), also known as the squashing function because it squashes or limits the amplitude range of the output, determines the neuron's output by comparing the summing junction result with a threshold function, \( \theta_k \). This procedure is written as

\[
    u_k = \sum_{j=1}^{p} w_{kj} x_j \quad \text{and} \quad y_k = \varphi(u_k - \theta_k),
\]

where \( y_k \) represents the output state.

As a complex device, whose interconnections have not yet been directly mapped, biological neural responses are approximated in ANNs as bounded, continuous activation functions. There are three types of activation functions used to approximate the neural response; they include the hard limiter, piecewise linear sigmoid, and smooth sigmoid functions. Functions within the smooth sigmoid activation function taxonomy, such as hyperbolic tangents, are most commonly used because they exhibit smoothness, asymptotic characteristics, and assume a continuous range of values from 0 to 1. This thesis applies the hyperbolic tangent activation function to approximate a response for the neurons. These neurons make up the network architecture for the problem model; information on network architecture is described in the following section.

### 2.2 Network Architecture and Training

In general, ANNs are parallel, non-linear, adaptive systems that possess potential for fast computation due to architecture or topology of the network. ANN architectures are defined by the number of neuron layers and interconnections. Specifically, a network architecture is composed of input (receptor), hidden and output (effector) layers and are classified as single or multi-layered recurrent and feedforward lattice structures. The recurrent artificial neural network (RANN) has a cyclic structure where the output of one neuron serves as the input to next neuron (Figure 2.3).
Figure 2.3: An example of a recurrent network.

Neurons may or may not be arranged in layers, however, there must be a connection between preceding and current neurons. Information processed in the ANN system evolves with time, as the adaptation of synaptic weights and synaptic responses are evaluated within this architecture.

In contrast, feedforward artificial neural network architecture leads information through one pathway, and can contain several hidden layers of interconnected neurons (Figure 2.4). Unlike RANN, FFANN contains no connections between proceeding and

Figure 2.4: An example of a 4-3-2-1 multilayer feedforward network.
current neurons. The perceptron is the simplest feedforward network since it only contains a single layer. Conversely, the feedforward or multi-layered neural network with backpropagation learning process is most commonly used to model problems. The network architecture of the problem studied in this thesis includes a hybrid of a multi-layered RANN and FFANN, which is discussed in Chapter 4.

Equally important to an ANN’s topology is the training used to build the engineering model. Training is classified as either supervised or unsupervised. Excluding unsupervised training from this thesis, the supervised training of an ANN can be formulated as the procedure of selecting \( \beta \) so that the response of the neural network is “close” to the available data, \( f_\epsilon(x) \). Often, the network parameters are found by minimizing the following error criterion (training objective function)

\[
\epsilon = \sum_{i=1}^{N} \left| f_\epsilon(x_i) - f_{ANN}(x_i, \beta) \right|^2 .
\]  

(2.1)

where

\[
f_{ANN}(x, \beta) = \sum_i \beta_i^k \varphi(k-1) \left( \varphi_1 \left( \sum_{m=1}^{p} \beta_m^1 x_m \right) \right),
\]  

(2.2)

and where \( x = (x_1, ..., x_p) \) is the p-dimensional input of the ANN system, \( \beta \) represents the set of network parameters, and \( \varphi_j \) is the non-linear transfer function of neurons associated with the \( j \)-th intermediate layer of the network. The network response \( f_{ANN}(x, \beta) \) depends linearly on only a few parameters, which are shown in Equation (2.2) as the coefficients \( \beta_i^k \) of the expansion. The remaining parameters affect the network response in a non-linear manner. Correspondingly, supervised training of neural networks is based on non-linear optimization procedures. In terms of the connectionist literature, numerical minimization of Equation (2.1) by the steepest descent method constitutes the backpropagation algorithm [26], which is the most popular in the literature. The conjugate gradient method and other nonlinear optimization methods have also been used for network training [27].
In most training schemes the initial values for the network parameters, $\bar{\theta}$, are commonly selected by random sampling [14]. If a satisfactory solution is not obtained during training, the non-linear optimization procedure is restarted with new randomly selected values [28]. Clearly this approach can require a great deal of computational resources for some practical problems. Engineers have continued to make progress in studying minimization procedures and have developed new training algorithms to better represent complex problems. For example, in 1989, Ersoy and Chen [9] considered the complexity of real world problems by presenting the formulation of transform techniques in the learning algorithm to reduce computation cost and to simplify the network. They did so by using a fast transform a-priori in the ANN learning scheme so that the initial parameters are based on the best approximation of the desired transform. The procedure not only minimizes the number of iterations necessary to compute the solution, but gave better generalization and increased quality of the problem modeled.

In 1996, Citterio et al. [3] developed a training method using a new spectrum-based scheme which allows the network to learn at a faster rate. Specifically, the authors recognized that the network's output spectrum is a linear combination of the hidden neuron's output spectra, therefore, they can use this information to minimize the error between the function solution and the network. Using this learning technique, functions can be approximated using spectral analysis of three-layered feedforward artificial neural networks (FFANNs).

Then in 1997, Lee et al. [16] devised a scheme which includes a scaled conjugate gradient-learning algorithm. This teaching method was used in a neural network to simulate the correlation between wall-shear stresses and wall actuation for drag reduction in turbulence control. The hidden layer consisted of an activation function
constructed by hyperbolic tangents. This scheme proved to be effective in modeling a physical engineering problem with the use of an ANN architecture.

Burrows and Niranjan [2] proposed to initialize the network parameters by linearizing the neuron transfer functions. Unfortunately, this method is not satisfactory for emulating highly non-linear input-output relationships. In 1997 the application of Chebyshev polynomial bases (CPB) expansions as the ANN activation function was studied by Lee and Jeng [17, 18], who found that CPB has a faster learning speed as compared to other conventional methods that do not use bases expansions. Specifically, they derive a relationship between single-layer and multi-layer perceptron neural networks and show that the CPB unified model neural network can be represented as a functional link network based on Chebyshev polynomials. In comparison to other networks which use the recursive least square method learning algorithm, CPB shared similar capabilities and was compatible in solving the control of nonlinear $H_{\infty}$ problems [15].

A drawback found in all training methods is that sparse or incomplete data sets generate noise in the network. This is known as overfitting. Additionally, poor convergence due to the use of nonlinear optimization methods can cause what is known as overtraining. Overtraining and overfitting are commonly encountered when modeling physical systems. Both of these problems are a result of the minimization of the objective function of Equation (2.1), which is an ill-posed problem; an infinite number of possible functions $f_{ANN}$ can be used to satisfy the equation by fitting the data. One common method to address this problem is to add random noise to the training data. Another remedy is the use of Tikhonov regularization, which will be discussed in Chapter 3.
2.3 Conventional Construction of a ANN

To better illustrate the problems that may be encountered by a typical user, the conventional construction of a network that emulates a physical system is outlined:

1. Choose the topology of the network (e.g., feedforward, recurrent, or Hopfield networks).
2. Choose the type transfer functions (e.g., sigmoids or radial bases).
3. Choose the number of layers.
4. Choose the training method (e.g., supervised or unsupervised).
5. Assemble the data, normalize it, and separate training from testing data.
6. Choose the number of nodes in the input, output, and hidden layer(s).
7. Train until convergence.
8. Compare ANN response to the testing data set.

If the comparison between the ANN response and testing data is unsatisfactory then the following can be done either in sequence or in parallel:

1. Alter the number of nodes and/or the number of layers.
2. Alter the type of transfer functions.
3. Redistribute data between training and testing sets.
4. Add random noise to data in training set.

As noted by this outline, the conventional construction and training of a ANN can be a time consuming trial and error process. In addition, even if the construction and
training are successful there is no guarantee than the ANN has mapped the input and output is a physically consistent manner. For example, there is no guarantee that with the successful mapping of displacement with time that one will be able to retrieve the speed and acceleration by differentiating $f_{ANN}$.

In an effort to alleviate (or even eliminate) the previously mentioned problems, this thesis presents a method by which conventional computational mechanics computer codes that approximate the solutions to mathematical models can be used to build the ANN architecture and determine the initial values of the network parameters. This initialized network can then be further improved by experimental data, with or without network parameter constraints. The proposed strategy can also allow the initiation of a connectionist training algorithm without data; this feature can be especially useful for on-line ANN modeling 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 describing a physical system with an inadequately small training set. 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. In presenting the method we start with a discussion of Tikhonov regularization in Chapter 3.
Chapter 3

Network Initiation and Basis Construction

This chapter will discuss the reasoning and details involved in the proposed ANN construction and weight initiation. We will begin with a discussion of the Tikhonov regularization approach and then progress to the construction of well-behaved bases from transfer functions.

3.1 Tikhonov Regularization

Initially adapted to ANN systems by Poggio and Girosi [24], Tikhonov regularization imposes constraints on the ANN parameters by minimizing the objective function of Equation (2.1) to reduce the overtraining mentioned in Section 2.2. This is accomplished with the Tikhonov regularization functional \( \Lambda[f_{ANN}(x, \bar{\beta})] \) which controls the smoothness of the ANN response, \( f_{ANN} \). The objective function is

\[
\epsilon(\lambda, f_{ANN}) = \sum_{i=1}^{N} |f_{exp}(x_i) - f_{ANN}(x_i, \bar{\beta})|^2 + \lambda \Lambda[f_{ANN}(x, \bar{\beta})]
\]  

(3.1)

where the regularization parameter and functional are denoted by \( \lambda \) and \( \Lambda \) respectively.

The Tikhonov functional chosen by Poggio and Girosi minimizes the curvature of the ANN response. This will not necessarily satisfy the smoothness requirements of a physical system. In an effort to remedy this drawback, an a-priori mathematical model has been used in place of the smoothness functional along with a linear combination of transfer functions [20, 22]. The functional representing the a-priori mathematical model can be derived by starting with the following equation which can describe a
large class of mechanics problems,

\[ L[f_0] = g(x), \quad x \in \Omega, \quad \text{and} \quad B[f_0(x)] = 0, \quad x \in \partial \Omega. \]  

(3.2)

The differential and boundary operators are denoted by \( L[\cdot] \) and \( B[\cdot] \) respectively. The solution function, \( f_0 \), and forcing function \( g \) are also represented in this equation.

Defining the equation residual as

\[ R_{ANN} = L[f_{ANN}] - g(x), \quad x \in \Omega \]

and the inner product as

\[ \langle u, v \rangle = \int_{\Omega} u \, v \, d\Omega, \]

where \( u \) and \( v \) are arbitrary functions, then the solution of the mechanical problem becomes the minimization of the energy functional,

\[ \Lambda(f_{ANN}) = \frac{1}{2} \left( R_{ANN}, R_{ANN} \right) + \frac{1}{2} \left( B[f_{ANN}], B[f_{ANN}] \right). \]

This combination of the augmented objective function can be written as

\[ \epsilon(\lambda, f_{ANN}) = \sum_{i=1}^{N} |f_{\exp}(x_i) - f_{ANN}(x_i, \beta)|^2 \]

\[ + \frac{\lambda}{2} \left( \left( R_{ANN}, R_{ANN} \right) + \left( B[f_{ANN}], B[f_{ANN}] \right) \right). \]  

(3.3)

Equation (3.3) includes both the mathematical model and experimental data. Overall, this process allows the user to merge the mathematical model and experimental information in the construction of the ANN in a mathematically consistent manner that can be controlled and monitored by the user.

Unfortunately, we see that the minimization of Equation (3.3) still requires nonlinear optimization. However, the construction of local or global basis from transfer functions can be used to constrain the nonlinear parameters in particular. By
constraining the nonlinear parameter, it is known that ANNs can represent general local or global basis functions, as compiled by Duch et al. [8]. That is, recalling Equation (2.2) we can rewrite $f_{ANN}$ as

$$f_{ANN}(x, \beta) = \sum_i \beta_i \varphi_i(x) = \sum_j \Phi_j(x)c_j,$$

and we note the only remaining unknowns are the coefficients $c_i$. Returning to Equation (3.3) and assuming $f_{ANN}$ can be described by a weighted linear combination of $M$ well-behaved basis functions, $\Phi$, then the minimum of Equation (3.3) gives:

$$\left\langle R_{ANN} \left( \sum_{j=1}^{M} \Phi_j c_j \right), \Psi_k(x) \right\rangle + \frac{2}{\lambda} \sum_{j=1}^{M} \sum_{i=1}^{N} (\Phi_j(x_i)\Psi_k(x_i)) c_j$$

$$+ \left\langle B \left( \sum_{j=1}^{M} \Phi_j c_j \right), q_k(x) \right\rangle = \frac{2}{\lambda} \sum_{i=1}^{N} (f_{exp}(x_i)\Psi_k(x_i)) .$$

(3.4)

where $\Psi_k(x)$ and $q_k(x)$ are equal to linearized forms of $\frac{\partial R_{ANN}}{\partial c_k}$ and

$$\frac{\partial B[f_{ANN}]}{\partial c_k},$$

respectively.

The problem has now been sufficiently constrained so that the only remaining unknowns are the linear coefficient of $f_{ANN}$. Depending on the value of $\lambda$ and the form of $R_{ANN}$, the coefficients $c_j$ may be determined directly or by a nonlinear optimization technique. In the case of nonlinear optimization, if the basis $\Phi_j$ are well-behaved basis functions, then existing computational methods can be used to solve for the coefficients $c_j$.

For $\lambda \to 0$ the mathematical model is fitted through each of the data points [33] while for the case of $\lambda \to \infty$, the experimental data becomes negligible. Since countless papers have been written on the fitting of data by ANNs, we turn our
attention to the mathematical model, i.e., investigation of Equation (3.4) with $\lambda \rightarrow \infty$. To accomplish this we investigate the linearization of the ANN response $f_{ANN}$ through the use of global basis functions.

As previously mentioned, by constraining the nonlinear parameters it is known that ANNs can represent general local or global basis functions. However, there is no indication that these bases are well-behaved. On the other hand, it has been shown by Meade and Zeldin [22] and Meade and Fernandez [10] that with the proper constraints popular ANN transfer functions can accurately represent conventional local and global basis functions (e.g. Chebyshev polynomials, B-splines, Fourier series) so that the ANN output weights become the basis expansion coefficients for the basis function. In this thesis we will concentrate on forming global bases from $\varphi = \tanh$.

3.2 Construction of Global Bases from Transfer Functions

Bases with the following properties could be especially useful for engineering applications:

- 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 modeling accuracy (i.e., good rate of convergence).

To form global bases we start by constructing polynomial with the following formula:

\[ \varphi_p(w, x) = \frac{\partial^{\mid \varphi \mid}}{\partial w_1^{p_1} \cdots \partial w_s^{p_s}} \varphi \left( w^T x + \theta \right) = x^p \varphi^{(\mid \varphi \mid)} \left( w^T x + \theta \right), \]  
(3.5)

where \((\cdot)^T\) denotes a vector transpose, \(x^p = \eta_1^{p_1} \cdots \eta_s^{p_s}\), \(\varphi\) is a multi-index such that \(\mid \varphi \mid = p_1 + \cdots + p_s\), and superscript \((\cdot)\) represents the order of the ordinary derivative of \(\varphi\). Equation (3.5) shows that a polynomial \(x^p\) can be readily expressed as

\[ x^p = \left( \varphi^{(\mid \varphi \mid)}(\theta) \right)^{-1} \varphi_p(0, x), \]

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

\[ \varphi_p(0, x) \approx \Upsilon(x) = (2\Delta w)^{-\mid \varphi \mid} \sum_{0 \leq \zeta \leq \varphi} (-1)^{\mid \zeta \mid} \left( \begin{array}{c} \varphi \\ \zeta \end{array} \right) \varphi \left( (2\zeta - \varphi)^T \Delta w x + \theta \right), \]

(3.6)

where \(\Delta w\) is a small positive constant and the multi-integer binomial is defined as

\[ \left( \begin{array}{c} \varphi \\ \zeta \end{array} \right) = \prod_{j=1}^{s} \left( \begin{array}{c} p_j \\ r_j \end{array} \right). \]

(3.7)

Note that the accuracy of the finite difference scheme of Equation (3.6), within a bounded domain of \(\Omega\), is given by

\[ \max |\Upsilon(x) - \varphi_p(0, x)| \leq C (\Delta w)^2 \]  
where \(C\) is a positive constant.
By combining Equations (3.6) and (3.7) it can be shown that the polynomial \( x^p \) can be approximated by a network layer of \( M = \prod_{j=1}^4 (p_j + 1) \) neurons [23]. Specifically,

\[
x_{ANN}^p = (\varphi^{(i_\cdot)}(\theta))^{-1} (2\Delta w)^{-|p|} \sum_{0\leq \zeta \leq p} (-1)^{|\zeta|} \left( \begin{array}{c} \varphi \\ \zeta \end{array} \right) \varphi \left( (2\zeta - \varphi)^T \Delta wx + \theta \right).
\]

The error of this approximation can be made arbitrarily small by the selection of \( \Delta w \) and \( \theta \). For example, the linear function \( y \) can be approximated by \( y_{ANN} \), where

\[
y_{ANN} = (2\Delta w \varphi^{(1)}(\theta))^{-1} [\varphi (\Delta wy + \theta) - \varphi (-\Delta wy + \theta)]. \tag{3.8}
\]

This approach is also valid for constructing products of dependent variables,

\[
y_{ANN} z_{ANN} = \left(4\Delta w \varphi^{(2)}(\theta)\right)^{-1} \left[\varphi (\Delta wy + \Delta wz + \theta) + \varphi (-\Delta wy - \Delta wz + \theta) - \varphi (-\Delta wy + \Delta wz + \theta) - \varphi (\Delta wy - \Delta wz + \theta)\right],
\]

where \( \Delta w_y \) and \( \Delta w_y \) are constants specific to variables \( y \) and \( z \), and multilayered approximations

\[
y_{ANN2} = (2\Delta w \varphi^{(1)}(\theta))^{-1} [\varphi (\Delta wy_{ANN1} + \theta) - \varphi (-\Delta wy_{ANN1} + \theta)],
\]

where \( y_{ANN1} \) is the approximation from Equation (3.8) and \( y_{ANN2} \) is the result from the second layer of neurons.

### 3.3 Construction of Trigonometric Bases

Since the DDC problem of this thesis is governed by partial differential equations that are solved by a pseudospectral scheme, the basis expansion requires the use of polynomials compatible to this numerical scheme. Specifically, the global basis is constructed by polynomials based on the Gudermannian function [1]. The Gudermannian
function is defined as $\gamma = gd(x)$, and its relation to the hyperbolic and trigonometric functions in the spectral scheme is

$$\tanh(x) = \sin(gd(x)).$$

Solving for $x$, we find

$$x = gd^{-1}(\gamma)$$

where

$$gd^{-1}(\gamma) \approx \gamma + \frac{1}{6} \gamma^3 + \frac{1}{24} \gamma^5 + \frac{61}{5040} \gamma^7 + \frac{277}{72576} \gamma^9 + \frac{50521}{39916800} \gamma^{11}$$

for $|\gamma| < \frac{\pi}{2}$

and so

$$\sin(\gamma) \approx \tanh \left( \gamma + \frac{1}{6} \gamma^3 + \frac{1}{24} \gamma^5 + \frac{61}{5040} \gamma^7 + \frac{277}{72576} \gamma^9 + \frac{50521}{39916800} \gamma^{11} \right)$$

for $|\gamma| < \frac{\pi}{2}$. (3.9)

The approximation of the sine can be realized in a FFANN subnetwork with three hidden layers using a distribution of piecewise linear sigmoidal transfer functions in the first hidden layer to form a sawtooth distribution [10]. This sawtooth distribution, varying between $\pm \pi/2$, maps the input $x$ into $\gamma$. The output from the linear combination of piecewise linear sigmoids are then fed into the second hidden layer consisting of hyperbolic tangents, with appropriate output weights, that approximate $gd^{-1}(\gamma)$. This in turn is fed into a third layer with a single hyperbolic tangent of output weight equal to 1. Figure 3.1 shows the RMS of the error between the sine and its subnetwork approximation versus the number of hyperbolic tangents required in the second hidden layer using two piecewise linear (first hidden layer) and 1 hyperbolic tangent (third hidden layer) for a $\Delta w = 1.5625 \times 10^{-2}$. The number of
hyperbolic tangents are directly related to the order of the polynomials used in the approximation of $g d^{-1}(\gamma)$: 2 for $\gamma$, 4 for $\gamma^3$, 6 for $\gamma^5$, 8 for $\gamma^7$, 10 for $\gamma^9$, and 12 for $\gamma^{11}$. Details can be found in the appendix. We see that this approach to building bases for engineering applications satisfies all of the desirable properties listed in Section 3.2. Figure 3.2 compares the sine to a 2-42-1 subnetwork approximation of Equation (3.9). Figure 3.3 displays the error distribution between the two functions of Figure 3.2. The approximation seems satisfactory.

For the DDC problem our network possess two inputs ($x$ and $z$) whose inputs range from zero to $5\pi/2$ for $x$ and zero to $3\pi/2$ for $z$. We have used an architecture of six piecewise linear (first hidden layer), 42 hyperbolic tangents (second hidden layer), and 1 hyperbolic tangent (third hidden layer) with $\Delta w = 1.5625 \times 10^{-2}$ for each input. The output of the subnetwork approximation is then added to the contribution from other sine approximations in the full ANN output layer whose nodes are simple linear functions. Figure 3.4 is a sketch of the full ANN assembled to approximate the solution to the DDC problem. If we define the subnetwork required to approximate
Figure 3.2: Comparison between $\sin(x)$ and the ANN approximation of Equation (3.9).

Figure 3.3: Distribution of error between $\sin(x)$ and the ANN approximation of Equation (3.9).
Figure 3.4: ANN used to approximate the DDC solution.
the sine as a processing element, then the DDC problem requires a full ANN of 768 processing elements with two input nodes for $x$ and $z$, and four output nodes for the approximation of the $x$ and $z$ velocity components ($u$ and $w$), the temperature ($T$), and the stream function ($\psi$).

Chapter 5 presents the numerical method and the hybrid FFANN/RANN based scheme used to solve the DDC. But first, the following chapter examines the ANN approximation of PDE solutions.
Chapter 4

Modeling Partial Differential Equations

The governing equations of most complex problems incorporate ordinary and partial differential equations (ODEs and PDEs). This thesis employs a pseudospectral method for the solution of the DDC's governing equations. The pseudospectral method is based on the use of Galerkin and collocation numerical schemes which are in turn members of the general numerical method known as the Method of Weighted Residuals (MWR). This chapter will demonstrate how the solution of governing PDEs through the MWR and transfer function constraints dictate the architecture of the ANN. In other words, the bases used to solve the governing PDEs, and the governing PDEs themselves, build the network for the user.

4.1 Feedforward Artificial Neural Networks

To begin with, we expect the FFANN architecture to adequately approximate the solutions to elliptic equations since they only provide spatial mappings. However, the equations for double diffusive convection are temporal as well as spatial. In this regard, the MWR is used to decompose the problem into separate spatial and temporal contributions. Specifically, the MWR converts the spatial operator of the differential equation into a system of algebraic equations. Various procedures can be used to solve these equations. To start, recalling Equation (3.2), the MWR is constructed from the spatial operator $L$. An approximation of $f_0$, $f_{ANN}$, is made to
be

\[ f_{ANN}(x) = \sum_{i=1}^{N} \Phi_i(x)c_i. \]

where again, the basis function \( \Phi_i(x) \) can either be defined by global polynomials or locally defined piecewise polynomials.

To satisfy the boundary conditions automatically, a selection of \( \Phi_i(x) \) must be made or another function \( \Gamma(x) \) multiplied by the approximation can be applied to satisfy the boundary conditions of the problem. In this case, the approximation is

\[ f_{ANN}(x) = \Gamma(x) \sum_{i=1}^{N} \Phi_i(x)c_i. \]

Using the spatial operator in Equation (3.2), the approximating function becomes

\[ L[f_{ANN}] - g(x) = R \neq 0. \]

If the selected approximation is an exact solution, then the residual error, \( R \), will vanish everywhere.

In the MWR, a weighted integral of the following form is set to zero,

\[ \langle R(x), W_k(x) \rangle = \int_{\Omega} R(x)W_k(x)dx = 0, \quad k = 1, \ldots, N \]

where \( W_k(x) \) are the weighting functions.

The collocation method forces the residual to zero at specific values of \( x \) by choosing the weighting functions \( W_k(x) \) to be equal to the distribution \( \Delta \) where,

\[ W_k(x) = \Delta(x - x_k) = \begin{cases} 0 & x \neq x_k \\ \infty & x = x_k \end{cases}. \]

The Galerkin method, on the other hand, forces the equation residual to be orthogonal to the function \( W_k = \Psi_k \) which is a function of the basis function, \( \Phi \) (i.e., \( \Psi_k = f(\Phi_k) \)),

\[ \langle R(x), W_k(x) \rangle = \langle R(x), \Psi_k(x) \rangle = 0, \quad k = 1, \ldots, N. \]
4.2 Recurrent Artificial Neural Networks

Recurrent artificial neural networks (RANN) are time-marching topologies for time-dependent mappings and lend themselves to the solution of time-dependent PDEs. It has been demonstrated by Meade and Moreno [21] that any noniterative time integration scheme can be realized as a RANN. We will present the forward difference technique as an example.

Assuming a time-dependent problem can be described by the equation,

$$\frac{\partial f_0}{\partial t} - L[f_0] + g(x) = 0$$  \hspace{1cm} (4.1)

and if we substitute $f_{ANN} \approx f_0(x, t)$, then when forward time stepping is used the above equation becomes

$$\frac{\partial f_{ANN}}{\partial t} \approx \frac{f_{ANN}^{n+1} - f_{ANN}^{n}}{\Delta t}.$$  

Functions $f_{ANN}^{n}$ and $f_{ANN}^{n+1}$ give approximations at the current and next time step, respectively. Both functions are dependent on the size of the time step $\Delta t$. Using Equation (4.1), the differential operator $L[f_{ANN}]$ and $g$ are evaluated by the forward difference scheme,

$$\frac{f_{ANN}^{n+1} - f_{ANN}^{n}}{\Delta t} - L[f_{ANN}^{n}] + g = 0.$$  

This can be written to solve for $f_{ANN}$ at the $n+1$ time step,

$$f_{ANN}^{n+1} = f_{ANN}^{n} + \Delta t \left( L[f_{ANN}^{n}] - g \right).$$

Our solution technique used for spatial dependence and applied to a nonlinear spatial operator will result in a system of nonlinear algebraic equations. These dependent variable polynomials can be approximated by linear combinations of the ANN transfer functions discussed in Section 3.2.
4.3 FFANN/RANN Hybrid

The previous section described the types of equations whose solutions can be approximated by FFANN and RANN architectures. In summary, elliptic equations are modeled with a feedforward network, whereas time-dependent hyperbolic and parabolic equations can be modeled with RANN structures. However, with the inclusion of a feedback mechanism, feedforward networks can be used as a time series generator. This is accomplished with a hybrid of FFANN and RANN known as a Sequence Generator [25].

The Sequence Generator (SGen) is a degenerate form of the time-delay neural network (TDNN). TDNN applies the theory of state space reconstruction of a dynamical system using delay coordinates. SGen is a FFANN that applies the response of the network's past output, as in RANN architecture. Figure 4.1 illustrates SGen as a FFANN/RANN combination where the updates are described by the following:

\[
S_{i}^{t+1} = S_{out}; \quad S_{j}^{t+1} = S_{j-1}^{t}, j = 2, ..., N.
\]

The input values are \( S_{j}^{t} \) at time step \( t \). The hidden layers shown in Figure 4.1 refer to the number of layers in the given multi-layer feedforward network. Note that if this were a perceptron, there would be no hidden layers.

Like the Sequence Generator, the ANN architecture employed in this thesis resembles the dynamic response of a feedforward network with a recurrent loop that sends output at time \( t \) to the input layer at \( t + 1 \).
Figure 4.1: Diagram of the Sequence Generator (SGen).
Chapter 5

The Double Diffusive Convection Problem

The specific objective of this thesis is to show that Deane’s code [6] can be used to generate a neural network, with connection weights, whose response approximates the solution to the two-dimensional double diffusive convection problem. The purpose of this is to illustrate that ANN can be built straightforwardly and can be applied to hard computing problems with confidence. The resulting ANN can then be modified by the user to incorporate experimental data through the Tikhonov regularization approach, discussed in Chapter 3. This chapter shows how the double diffusive convection problem is solved using the pseudospectral method with polynomials in the ANN initialization scheme. Information on the fluid mechanics problem is presented along with the modifications made to Deane's numerical code. To start, the following will give a general description on the double diffusive convection problem.

5.1 Double Diffusive Convection

Double diffusive convection has been studied for several decades and was first referred to as thermohaline or thermosolutal convection due to the presence of salinity and temperature gradient in the ocean. Once believed to be a phenomenon of little scientific significance, double diffusion has developed into a subdiscipline of fluid mechanics. Besides oceanography, double diffusion is studied in other areas such as astrophysics and chemical engineering [5, 13].

The double diffusive convection model was formulated in 1956 when Stommel, Aarons and Blanchard [31] proposed an oceanographic curiosity relating to the phys-
ical nature of the tropical and sub-tropical regions of the ocean. In these regions, surface layers include a high content of salinity and heat, whereas the oceanic bottom is comprised of fresh cool water. With the placement of a long conducting pipe which extends vertically from the ocean surface to the floor, Stommel et al. projected the existence of a large saline and temperature gradient. They proposed that water pumped through the pipe flows continuously in either direction even after the pump is shut-off, hence the term perpetual salt-fountain (Figure 5.1).

![Diagram](image)

**Figure 5.1: A salt fountain.**

This phenomenon occurs because as the water slowly travels through the pipe, heat, not salinity, is transferred to the ambient ocean environment, where the saline density component contributes to buoyancy properties of the water. Specifically, if the water flows in an upward direction, the density inside the pipe is less than that of the outside surroundings and so the water inside will have greater buoyancy at the same vertical location. As an example, consider a small particle traveling in the vertical direction. The molecular diffusivity for temperature, \( \kappa_T = 1.5 \times 10^{-3} \text{ cm}^2/\text{sec} \), is greater than salt, \( \kappa_S = 1.3 \times 10^{-5} \text{ cm}^2/\text{sec} \); therefore the particle loses temperature at a greater rate than salinity providing energy for Boussinesq convective motion.
Stommel et al. conducted experiments with a salt fountain on a smaller scale and found that after an initial priming of water in a glass tube, the flow continued until the experimental conditions broke down. The authors surmise that in the ocean, continuous flow could also be maintained if climatic influences were consistent.

In 1960, Stern [29] suggested that future studies related to motion amplitude and turbulence would determine whether the proposed mechanism is significant in the vertical mixing of the sea. He studied the salt fountain’s convective motion and defined the mechanism of vertical transport. To evaluate this oceanic phenomenon, he considered an isolated system consisting of an convection chamber where the surface maintained a higher temperature and saline content than the bottom, with small perturbations imposed on the system. It is revealed that when the denser liquid is above, this system resembles Rayleigh-Bénard convection.

Stern studied this concept further by evaluating the stratification of unbounded regions of the ocean. Even without boundaries, double diffusive convection effects exist within the fluid where regions called salt fingers are formed. Oscillatory motion also occurs if diffusive conditions are unstable. In this case, stability analysis must be performed to determine the fluid’s properties. In this regard, the following section derives the governing equations and linear stability of double diffusive convection.

**5.1.1 Governing Equations**

In general, convective motion is described as a displaced particle in a fluid whose density concentrations have opposing vertical gradients that cause diffusion and transfer of energy. In double diffusive convection, buoyancy forces become the source of energy for fluid motion. Included are the Boussinesq approximation, which only allows density difference in the buoyancy terms, along with the existence of a solute component to be considered. By deriving the Boussinesq equations from the conservation
of mass, momentum, thermal energy, solute, and the linearized equation of state, the non-dimensionalized equations are formed [32].

The model problem for double diffusive convection considers two-dimensional time-dependent fluid motion between two horizontal planes, with the lower plane, \(z = 0\) having a higher content of salinity and temperature than the upper plane \(z = d\). The environment for the diffusive model includes free conductive boundaries. Corresponding values of temperature and solute are described by linear and convective redistribution regimes:

\[
T_{\text{total}} = T_m - \Delta T \frac{z}{d} + T,
\]

\[
S_{\text{total}} = S_m - \Delta S \frac{z}{d} + S.
\]

The density distribution is written as a linear approximation,

\[
\rho = \rho_m (1 - \alpha T + \beta S)
\]

where \(\rho_m\) is the mean density.

Boussinesq convection equations used for double diffusion are derived from the conservation of mass,

\[\nabla \cdot \mathbf{V} = 0,\]

conservation of momentum,

\[
\frac{\partial \mathbf{V}}{\partial t} + \mathbf{V} \cdot \nabla \mathbf{V} = -\frac{1}{\rho_m} \nabla p + g(\alpha T - \beta S)\mathbf{k} + \nu \nabla^2 \mathbf{V}, \tag{5.1}\]

conservation of thermal energy,

\[
\frac{\partial T}{\partial t} + \mathbf{V} \cdot \nabla T - w \frac{\Delta T}{d} = \kappa_T \nabla^2 T, \tag{5.2}\]

and conservation of solute,

\[
\frac{\partial S}{\partial t} + \mathbf{V} \cdot \nabla S - w \frac{\Delta S}{d} = \kappa_S \nabla^2 S, \tag{5.3}\]
where the acceleration due to gravity and the velocity vector are \( g \) and \( \mathbf{V} = (u, w) \), respectively.

The four non-dimensional parameters that are used in the vorticity equations include the Prandtl number,

\[
Pr = \nu / \kappa_T,
\]
the Lewis number,

\[
\tau = \kappa_S / \kappa_T,
\]
the thermal Rayleigh number, \( R_T \),

\[
R_T = \frac{g \alpha \sigma T \rho d^4}{\nu \kappa_T},
\]
and solute Rayleigh number, \( R_S \),

\[
R_S = \frac{g \beta \sigma S \rho d^4}{\nu \kappa_S},
\]
where \( \nu, \kappa_S, \kappa_T \) are kinematic viscosity, saline and temperature diffusivity, respectively. The values of \( \alpha \) and \( \beta \) are

\[
\alpha = -\left( \frac{1}{\rho \partial T} \right)_s, \quad \beta = \left( \frac{1}{\rho \partial S} \right)_T,
\]
and the stream function is defined as

\[
u = \frac{\partial \psi}{\partial z}, \quad \text{and} \quad w = -\frac{\partial \psi}{\partial x}.
\]

The boundary conditions to the vorticity equations are:

\[
\psi = \frac{\partial^2 \psi}{\partial z^2} = T = S = 0 \text{ at } z = 0, 1.
\]

Deane, Knobloch, and Toomre [7] describe the use of periodic boundary conditions in the horizontal to model a fluid layer of large aspect ratio. They model traveling
waves and chaos and show numerical results for Hopf bifurcation in thermosolutal convection in two dimensions. Hopf bifurcation includes double diffusive systems and convection in binary mixtures, and the boundary conditions are periodic in the horizontal. In this regard, the double diffusion convection model presented in this thesis will consider the complex phenomena of traveling and standing waves.

5.2 Numerical Model

Deane's computer code [6] was written to solve the two-dimensional Navier-Stokes equations for Boussinesq convection by the pseudospectral method and models the occurrence of standing and traveling convection waves.

In deriving the numerical model we start by representing the dimensionless dependent variables as,

\[
\hat{u} = \frac{1}{N_x N_z} \sum_{k_x k_z} u e^{i k_x 2\pi x / N_x} \cos(\pi k_z z / N_z) \tag{5.4}
\]

\[
\hat{T} = \frac{1}{N_x N_z} \sum_{k_x k_z} T e^{i k_x 2\pi x / N_x} \sin(\pi k_z z / N_z) \tag{5.5}
\]

\[
\hat{S} = \frac{1}{N_x N_z} \sum_{k_x k_z} S e^{i k_x 2\pi x / N_x} \sin(\pi k_z z / N_z) \tag{5.6}
\]

\[
\hat{w} = \frac{1}{N_x N_z} \sum_{k_x k_z} w e^{i k_x 2\pi x / N_x} \sin(\pi k_z z / N_z). \tag{5.7}
\]

where \(N_x\) and \(N_z\) represent the number of terms in the \(x\) and \(z\) directions, respectively and \(k = [k_x, k_z]^T\). We also define \(\zeta_j\) as \(\zeta_j \equiv (V \times \vec{\omega})_j\) and \(q \equiv p + \frac{1}{2} V \cdot V\), where the subscript \(j = 1, 2, 3\) represents the component in the \(x-, y-, z-\) directions, respectively.

If the curl of the dimensionless momentum equation is applied to continuity, then

\[- \frac{\partial \zeta_j}{\partial x_j} = - \frac{\partial^2 q}{\partial x_j^2} + \text{Pr}(\frac{\partial T}{\partial t} + \frac{\partial S}{\partial t}) \delta_{j3}\] for \(j = 1, 2, 3\). \tag{5.8}
Applying the Fourier transformation to Equation (5.8) yields,

\[-i k_j \hat{\zeta}_j = (k \cdot k) \hat{q} + i \text{Pr} k_j (\hat{T} + \hat{S}) \delta_{j3}\]

and solving for \( \hat{q} \) results in

\[\hat{q} = \frac{1}{(k \cdot k)} (-i k_j \hat{\zeta}_j - i \text{Pr} k_j (\hat{T} + \hat{S}) \delta_{j3}).\]

Applying the Fourier transform to the dimensionless diffusion equations gives

\[\frac{\partial \hat{V}_j}{\partial t} - \hat{\zeta}_j = -i k_j \hat{q} + \text{Pr} (\hat{T} + \hat{S}) \delta_{j3} - (k \cdot k) \hat{T} \quad j = 1, 2, 3,\]

\[\frac{\partial \hat{T}}{\partial t} + \hat{\nabla} \cdot \nabla T = R_T \hat{w} - (k \cdot k) \hat{T},\]

\[\frac{\partial \hat{S}}{\partial t} + \hat{\nabla} \cdot \nabla S = -R_S \hat{w} - (k \cdot k) \hat{S}.\]

In pseudospectral methods, we evaluate the nonlinear term in physical space (collocation) and the remainder in spectral space (Galerkin).

Using a leap-frog method for time integration gives

\[\frac{\hat{V}^{n+1} - \hat{V}^{n-1}}{2 \Delta t} = D_{NL},\]

where \( D_{NL} \) represents nonlinear terms, and using the Crank-Nicolson scheme for the diffusion terms results in

\[\frac{\hat{V}^{n+1} - \hat{V}^{n-1}}{2 \Delta t} = \frac{1}{2} (\nabla^2 \hat{V}^{n+1} + \nabla^2 \hat{V}^{n-1}).\]

The discrete equations are

\[\frac{\hat{V}^{n+1}_j - \hat{V}^{n-1}_j}{2 \Delta t} - \hat{\zeta}^n_j = i k_j \hat{q}^n + \text{Pr} (\hat{T}^n + \text{Pr}(\hat{S}^n)) \delta_{j3} - \frac{1}{2} \text{Pr} (k \cdot k)(\hat{V}_j^{n+1} - \hat{V}_j^{n-1})\]

\[\frac{\hat{T}^{n+1} - \hat{T}^{n-1}}{2 \Delta t} + i k_j \hat{V} \cdot \nabla T^n = R_T w^n - \frac{1}{2} (k \cdot k)(\hat{T}^{n+1} - \hat{T}^{n-1})\]

\[\frac{\hat{S}^{n+1} - \hat{S}^{n-1}}{2 \Delta t} + i k_j \hat{V} \cdot \nabla S^n = -R_S w^n - \frac{\tau}{2} (k \cdot k)(\hat{S}^{n+1} - \hat{S}^{n-1}).\]
Let
\[
\hat{\alpha}_j = V_j^{-1}(1 - \Pr(k \cdot k) \Delta t) + 2\Delta t \hat{\omega}_j^n + 2\Pr \Delta t (\hat{T}^n + \hat{S}^n) \delta_{j3},
\]
so that
\[
\hat{V}_j^n(1 + \Pr \Delta t (k \cdot k)) = \hat{\alpha}_j - \frac{k \cdot \hat{\alpha}}{(k \cdot k)} k_j.
\]

These are the governing equations for two-dimensional double diffusive convection numerical model used by Deane. Recall that the motivation of this work is to allow the numerical scheme that approximates the DDC solution to dictate the form of the architecture of the ANN surrogate. The following section describes how the ANN is constructed within the existing numerical scheme.

5.3 Construction of the ANN

Since we are able to construct a wide variety of traditional local and global bases, we see that we have an option other than utilizing trial and error architectures, slow or nonconvergent nonlinear optimization schemes and synthetic data for training. Through the construction of traditional bases from the linear combinations of transfer functions we have access to any of the numerical techniques that can be generated by the MWR; we merely have to replace the basis in an existing computational mechanics code with an ANN approximation. The computational mechanics code determines the type and number of bases and their respective coefficients. Knowing the values of coefficients and the nonlinear parameter constraints of the ANN and the number and type of transfer functions, the ANN is constructed.

To modify Deane's code to form the neural network that solves Equations (5.4) through (5.7), the Gudermannian function is utilized to build polynomials which lead to the trigonometric basis, as discussed in Chapter 3. The actual implementation
required only replacing the sine and cosine calls with the ANN trigonometric subroutine. To employ the ANN architecture illustrated in Figure 5.2, Deane's code evokes

Figure 5.2: ANN processing element, where $\Sigma$ is the data input summing junction and $\phi$ is the transfer function.

the structure of a neural network processing element where the problem's input information is first processed by a FFANN. This input information is subjected to data summation and transfer function computation. For the time-dependent component, the DDC solution output is used as the FFANN initial conditions for the next time step (Figure 5.3). As a result, Deane's code was used to construct a ANN in the SGen

Figure 5.3: FFANN architecture embedded into computer code.

topology discussed in Chapter 4. This network was then used to generate accurate
spatial and temporal responses to the DDC problem. Section 5.4 compares results generated from the SGen response to published results.

5.4 Results

As previously discussed, Deane's code [6] solves the double diffusive convection PDEs with a pseudospectral method. Specifically, this entails the evaluation of the nonlinear term in physical space, with collocation, while the remainder is calculated in spectral space with the Galerkin method. Since the boundary conditions for the DDC are periodic, the initial conditions are a combination of sinusoidal perturbations, which were modified for the ANN-based scheme. The responses of both the SGen network developed in Section 5.3 and Deane's DDC code using a $16 \times 48$ mesh, with a spatial period $l = 3$, $\tau = 10^{-1/2}$, $\sigma = 1$, and a time step of $\Delta t = 0.001$, are compared against the solutions of Deane et al. [7] for a traveling wave ($R_T = 14000$ and $R_S = 14585.4$) and a standing wave ($R_T = 10800$ and $R_S = 14585.4$).

When comparing the stream function contours from the ANN approximation of the traveling wave in Figure 5.4 (a) against the solutions of reference [7] in Figure 5.5, for the times indicated, we see that the convection cells have the proper rotation and are drifting to the left. Unfortunately, the cell size of the SGen ANN response seems too large; Deane et al. give a cell width of $\Delta x_{cell} \approx 1.5$, while the ANN has $\Delta x_{cell} > 2$. Also, the drift speed is obviously less than that in reference [7]. However, the ANN response is identical to the solutions given by Deane's pseudospectral code in Figure 5.4 (b).

When comparing the contours from the standing wave ANN approximation of Figure 5.6 (a) against the solutions of reference [7] in Figure 5.7, for the times as indicated, we see that the convection cells have the proper rotation and reversal at the correct times. Again, the cell size of the SGen ANN response seems too large.
Also, the ANN was unable to reproduce the streamline contours of Figure 5.7 at $t = 25.206$. For the ANN, the positions of the cells remain constant but the rotation reversal at $t = 25.206$ comes with a slight shift in cell positions to the right. Like the case shown in Figure 5.4, the ANN response is identical to the solutions given by Deane's pseudospectral code in Figure 5.6 (b).

Since the SGen ANN yields results that are identical to Deane's DDC code for both standing and traveling wave cases, that in turn are similar to the published results of reference [7], we conclude that the ANN is accurate in its emulation of the numerical model. Unfortunately, Deane's DDC code seems to have a problem in matching the conditions of reference [7]. Though the comparisons between the SGen ANN and the results of Deane et al. are not entirely satisfactory, Figures 5.4 and 5.6 demonstrate the capability of an ANN-based scheme to approximate a complex temporal–spatial fluid mechanics problem without training.
Figure 5.4: Stream function contours from the traveling wave for \( R_S = 14585.4 \) (a) ANN approximation and (b) Deane’s DDC code. Bottom to top, \( t = 45.360, 45.416, 45.472, 45.528, 45.584 \).
Figure 5.5: Deane et al. [7] traveling wave results ($\psi$) for $R_s = 14585.4$. Bottom to top, $t = 45.360$, 45.416, 45.472, 45.528, 45.584.
Figure 5.6: Stream function contours from the standing wave for $R_s = 14585.5$ (a) ANN approximation and (b) Deane’s DDC code. Bottom to top, $t = 25.180$, 25.193, 25.206, 25.219, 25.232.
Figure 5.7: Deane et al. [7] standing wave results ($\psi$) for $R_s = 14585.5$. Bottom to top, $t = 25.180$, 25.193, 25.206, 25.219, 25.232.
Chapter 6

Conclusions and Future Work

Although the traditional modeling of physical systems with a ANN can be a time consuming and complicated procedure, this thesis proves that it can in fact be quite straight-forward and convenient for the engineering community. Recalling the conventional construction procedure listed in Section 2.3 we see that with our method:

1. Mathematical models determine the network topology, not the user.

2. Selection of the basis function family helps limit our choice of transfer function types that can be used.

3. Constraining transfer functions to form bases determines the number of layers, not the user.

4. Forming bases initializes the values of the nonlinear parameters.

5. Constraining the transfer functions to form bases allows the user to determine the effect of increasing the number of nodes on the ANN accuracy (rates of convergence).

6. The connection weight matrix can be easily interpreted.

7. Mathematical models, combined with Tikhonov regularization, allow the use of conventional and well understood methods from computational mechanics in the initialization of the parameters not already determined by the formation of the bases.
8. Mathematical models with Tikhonov regularization allow the user to control the quality of the ANN response with respect to the training data through $\lambda$ of Equation (3.1).

The ANN initiation method applied to the two-dimensional double diffusive convection problem yielded a hybrid feedforward/recurrent neural network whose response compared favorably to existing numerical solutions. Specifically, the method modifies an existing numerical scheme to construct an ANN without the need for training. The governing equations of the DDC are used as the Tikhonov functional while the construction of well-behaved global bases from the piecewise linear and hyperbolic tangent transfer functions were used in the approximation of the problem. Results indicate the success of this method by the similarities between the stream function contour plots of the ANN-based code and the published results. Additionally, this work suggests that with the ANN-based procedures more effective solutions can be gained for the analysis of other engineering problems. This is accomplished by including analytical, numerical, and experimental methods which provide information about a physical problem. Moreover, this method can be used as a physical surrogate which can be included in a larger framework (e.g., MDO).

Suggestions for future research on this problem include the following:

1. The DDC problem can be examined again using sequential function approximation (SFA) to model the problem's PDEs. SFA is a meshless and matrix–free function approximation method which reduces the number of bases required while producing comparable solutions for ODEs and PDEs [19].

2. Although analytical and numerical information are used in this thesis, perhaps data from DDC experiments can be included in the regularization of nonlinear
parameters. These results can then be compared with solutions computed by other numerical schemes.

3. It is proposed that this method be applied to other problems governed by ODEs and PDEs by imposing an ANN architecture to conventional numerical codes.
Bibliography


Appendix A

The ANN Modeling Subroutine

The following lists the ANN subroutine that approximates the sine function in Deane’s double diffusive convection simulation code [6].

c This program is added to the main program of Deane’s
c double diffusive convection code.
c The following code replaces the sine and cosine
c functions in the main program for the values of
c cur, cwr, ctr, csr.
c Note that the ANN cosine function for cur
c is constructed in Deane’s main
c program by subtracting pi/2 from the ANN sine
c approximation.

c

******************************************************************************
c ANN sine approximation
******************************************************************************

    real function annsin(x1)
    implicit none
    real x1
    double precision pi,b,x,w
double precision phi1,phi2,phi3,phi4,phi5,phi6,phi7,phi8,phi9
double precision phi10,phi11
double precision theta1,theta2,theta3,theta4,theta5,theta6
double precision theta7,theta8,theta9,theta10,theta11
double precision k
double precision hold1,hold2,hold3,hold4,hold5,hold6
double precision hold7,hold8,hold9,hold10,hold11
double precision hold12,hold13,weight
double precision holdt1,holdt2,holdt3,holdt4,holdt5,holdt6
integer a,ht

c Begin program section
   x=dble(x1)
   w = 1.5625d-02

c Constants (values for phi and theta are determined
c with central differencing)
   pi=3.141592653589793d0
   phi1=1.0d0
   phi2=0.76980035823721d0
   phi3=0.666666666622090d0
   phi4=4.08588549669786d0
   phi5=16.0d0
   phi6=52.26595295364843d0
   phi7=1.553495389774902d2
   phi8=1.223720519250494d3
   phi9=7936.0d0
   phi10=4.557203537396648d4
   phi11=2.411338687378368d5
   theta1=0.0d0
theta2 = -6.585000000000001d-01
theta3 = -1.146200d0
theta4 = 4.213000000000000d-01
theta5 = theta1
theta6 = -3.125000000000000d-01
theta7 = -5.717000000000000d-01
theta8 = 2.488000000000000d-01
theta9 = theta1
theta10 = -2.068000000000000d-01
theta11 = -3.872000000000000d-01

c Convert values of x to be between -2 Pi and 0
if ((x.lt.0.0d0).and.(x.ge.(-2.0d0*pi))) then
  a = 0
  b = x + dble(a)*2.0d0*pi
elseif (x.lt.0.0d0) then
  ht = int(x/(2.0d0*pi))
  a = -ht
  b = x + dble(a)*2.0d0*pi
elseif ((x.ge.(0.0d0)).and.(x.le.(2.0d0*pi))) then
  ht = int(x/(2.0d0*pi))
  a = ht + 1
  b = x - dble(a)*2.0d0*pi
else
  ht = int(x/(2.0d0*pi))
  a = ht + 1
  b = x - dble(a)*2.0d0*pi
endif

c Now convert to -pi/2 to pi/2

if ( (b.ge.(-pi/2.0d0)).and.(b.le.(pi/2.0d0)) ) then
  k=1.0d0
  x=b
elseif((b.ge.((-3.0d0*pi)/(2.0d0))).and.(b.lt.(-pi/2.0d0))) then
  k=-1.0d0
  x=b+pi
elseif((b.ge.(-2.0d0*pi)).and.(b.lt.((-3.0d0*pi)/(2.0d0))))then
  k=1.0d0
  x=b+2.0d0*pi
else
  write(*,*)'stopping code'
  stop
endif

c Using the Gudermannian equation, the following calculates

c the function expansion for tanh

c *******************************************************
c Function x**1

c 2 tanh

  weight= 1.0d0/((2.0d0*w)**1*phi1)
  hold1 = tanh(theta1 -w*x)
  hold2 = -1.0d0*tanh(theta1 + w*x)
  holdt1=-weight*(hold1+hold2)

c *******************************************************
c Function x**3
c 4 tanh

weight = 1.0d0/(8.0d0*(w)**3*phi3)
hold1= tanh(theta3 - 3.0d0 * w * x)
hold2= -3.0d0 * tanh(theta3 - w * x)
hold3= 3.0d0 * tanh(theta3 + w*x)
hold4= - tanh(theta3 + 3.0d0*w*x)
holdt2=-weight*(hold1+hold2+hold3+hold4)

c ****************************************
c Function x**5

c 6 tanh

weight= 1.0d0/((2.0d0*w)**5.0d0*phi5)
hold1=tanh(theta5 - 5.0d0*w*x)
hold2= - 5.0d0*tanh(theta5 - 3.0d0*w*x)
hold3= 10.0d0*tanh(theta5 - w*x)
hold4= - 10.0d0*tanh(theta5 + w*x)
hold5= 5.0d0*tanh(theta5 + 3.0d0*w*x)
hold6= - tanh(theta5 + 5.0d0*w*x)
holdt3=-weight*(hold1+hold2+hold3+hold4+hold5+hold6)

c ****************************************
c Function x**7

c 8 tanh

weight= 1.0d0/((2.0d0*w)**7*phi7)
hold1=tanh(theta7 - 7.0d0*w*x)
hold2= - 7.0d0*tanh(theta7 - 5.0d0*w*x)
hold3= 21.0d0*tanh(theta7 - 3.0d0*w*x)
hold4= - 35.0d0*tanh(theta7 - w*x)
hold5 = 35.0d0*tanh(theta7 + w*x)
hold6 = -21.0d0*tanh(theta7 + 3.0d0*w*x)
hold7 = 7.0d0*tanh(theta7 + 5.0d0*w*x)
hold8 = -tanh(theta7 + 7.0d0*w*x)
hold9 = hold5 + hold6 + hold7 + hold8
holdt4 = -1.0d0*weight*(hold1 + hold2 + hold3 + hold4 + hold9)

C *************************************************
C Function x**9
C 10 tanh

weight = 1.0d0/((2.0d0*w)**9*phi9)
hold1 = tanh(theta9 - 9.0d0*w*x)
hold2 = -9.0d0*tanh(theta9 - 7.0d0*w*x)
hold3 = 36.0d0*tanh(theta9 - 5.0d0*w*x)
hold4 = -84.0d0*tanh(theta9 - 3.0d0*w*x)
hold5 = +126.0d0*tanh(theta9 - w*x)
hold6 = -126.0d0*tanh(theta9 + w*x)
hold7 = 84.0d0*tanh(theta9 + 3.0d0*w*x)
hold8 = -36.0d0*tanh(theta9 + 5.0d0*w*x)
hold9 = 9.0d0*tanh(theta9 + 7.0d0*w*x)
hold10 = -tanh(theta9 + 9.0d0*w*x)
hold11 = hold6 + hold7 + hold8 + hold9 + hold10
holdt5 = weight*(hold1 + hold2 + hold3 + hold4 + hold5 + hold11)

C *************************************************
C Function x**11
C 12 tanh

weight = 1.0d0/((2.0d0*w)**11*phi11)
hold1 = tanh(theta11 - 11.0d0*w*x)
hold2 = -11.0d0*tanh(theta11 - 9.0d0*w*x)
hold3 = +55.0d0*tanh(theta11 - 7.0d0*w*x)
hold4 = -165.0d0*tanh(theta11 - 5.0d0*w*x)
hold5 = +330.0d0*tanh(theta11 - 3.0d0*w*x)
hold6 = -462.0d0*tanh(theta11 - w*x)
hold7 = 462.0d0*tanh(theta11 + w*x)
hold8 = -330.0d0*tanh(theta11 + 3.0d0*w*x)
hold9 = +165.0d0*tanh(theta11 + 5.0d0*w*x)
hold10 = -55.0d0*tanh(theta11 + 7.0d0*w*x)
hold11 = 11.0d0*tanh(theta11 + 9.0d0*w*x)
hold12 = -tanh(theta11 + 11.0d0*w*x)
hold13 = hold6 + hold7 + hold8 + hold9 + hold10 + hold11 + hold12
holdt6 = -weight*(hold1 + hold2 + hold3 + hold4 + hold5 + hold13)

C *****************************************************************************

hold1 = holdt1 + (1.0d0/6.0d0)*holdt2 + (1.0d0/24.0d0)*holdt3
hold2 = (61.0d0/504)0.0d0)*holdt4 + (277.0d0/72576.0d0)*holdt5
hold3 = (50521.0d0/39916800.0d0)*holdt6
annsin = sngl(k*tanh(hold1 + hold2 + hold3))

return

end

C *****************************************************************************