2014

Using Backpropagation Neural Networks for the Prediction of Residual Shear Strength of Cohesive Soils

Luke Detwiler
University of Vermont, ldetwile@uvm.edu

University of Vermont

Using Backpropagation Neural Networks for the Prediction of Residual Shear Strength of Cohesive Soils

Thesis Review Committee:

Dr. Donna Rizzo	(Advisor)
Dr. Mandar Dewoolkar	(Co-advisor)
Alison Pechenick	

Luke Detwiler

Acknowledgements

I would like to thank Dr. Donna Rizzo and Dr. Mandar Dewoolkar for serving as advisors on this project. Dr. Rizzo supported me on all aspects of the coding and development of the backpropagation network, data analysis, and report editing performed in this project. Dr. Dewoolkar supported me on the geotechnical and data analysis aspects of this project. I am grateful to the both of them for providing support not only on this project, but in other aspects of my academic and personal development. Thank you to Alison Pechenick for serving on my thesis committee and employing me over the past two semesters. A special thanks to Trevon Noiva for two wonderful semesters of commiseration.

Abstract

A complex nonlinear relationship exists between indicative soil parameters (liquid limit, plastic limit, clay fraction, sand fraction, and normal stress loading) and drained secant residual friction angle ($ø_r$’). Academic literature provides various empirical models for the prediction of $ø_r$’, though their predictions generally suffer from insufficient representation of the nonlinear relationship between parameters. In this study, an artificial neural network was developed for the prediction of $ø_r$’. Artificial neural networks are computational learning algorithms that derive their structure and learning procedures from phenomena observed in biological nervous systems. Their complex, interconnected structures allow for successful mapping of nonlinear relationships between parameters. This motivated the development and application of a network known as a backpropagation neural network (BPNN) as an alternative predictive model for $ø_r$’.

The BPNN was trained to successfully map the relationship between indicative soil parameters and $ø_r$’ using a variety of soil datasets provided from academic literature and other sources. The BPNN’s performance was evaluated using a normalized root mean square error (RMSE) term. It was posited that the BPNN predictions could be improved by training individual networks on soil data subdivided by clay fraction ranges. Analysis showed that the division of data into subsets significantly reduced the BPNN’s predictive performance by limiting the amount of data available for individual network training.

Where other predictive models generally neglect sand fraction as a predictive parameter for $ø_r$’, this study attempted to evaluate its predictive value. Comparison between a BPNN that included sand fraction and one that did not proved inconclusive as the results from multiple RMSE analyses between the two models were not statistically different.

Correlation-based equations for the prediction of $ø_r$’ in Stark and Hussain (2013) are based on the subdivision of soil data by clay fraction ranges. A comparison of the BPNN predictive model to other empirical models was performed in order to evaluate the viability of a BPNN as an alternative to current predictive models. The BPNN outperformed a traditional, multivariate least-squares linear regression model as well as correlation-based equations from Stark and Hussain (2013). The normalized root mean square error for Stark and Hussain equation-based predictions of $ø_r$’ was 0.2270 in comparison to 0.1278 for the BPNN. Where Stark and Hussain’s equations performed well on the particular dataset used to create these empirical equations, it failed to accurately predict $ø_r$’ for other data sets provided in the literature. The BPNN model was more robust in its ability to predict $ø_r$’ over a variety of datasets in this study and suggests that the BPNN may provide a viable alternative to other predictive models for $ø_r$’.

Table of Contents

Table of Equations

For figures & tables, see attached Figures &Tables document.

Glossary

ANN	-	Artificial Neural Network
ASTM	-	American Society for Testing and Materials
BPNN	-	Backpropagation Neural Network
CF	-	Clay Fraction
D	-	Dewoolkar & Huzjak 2005 dataset
H	-	Hayden dataset
LL	-	Liquid Limit
PL	-	Plastic Limit
S	-	Commercial dataset
RANN	-	Recurrent Artificial Neural Network
SE	-	Stark and Eid 1994 dataset
SF	-	Sand Fraction
SOM	-	Self Organizing Map
TM03	-	Tiwari and Mauri 2003 dataset
TM05	-	Tiwari and Mauri 2005 dataset
$ø_r'$	-	Secant Residual Friction Angle
σ'	-	Effective Normal Stress
τ	-	Shear Stress

1. Introduction

Engineers are often faced with complex problems involving the mapping of two or more parameters that are nonlinear. A nonlinear relationship is a relationship between two or more parameters that cannot be satisfied through the rules of superposition. This occurs across a wide variety of disciplines, including chemical engineering, electrical engineering, process engineering, and flight control (Hoskins and Himmelblau 1988; Park, El-Sharkawi et al. 1991; Willis, Di Massimo et al. 1991; Kim and Calise 1997).

Where traditional modeling techniques fail to map nonlinear data relationships, Artificial Neural Networks (ANNs) have experienced success across disciplines (Chang-Chi and Sheng-Huoo 2007; Azami, Mosavi et al. 2013; Bevilacqua 2013; Tamilselvan and Wang 2013; Torabi, Shirazi et al. 2013). The Artificial Neural Network (ANN) structure is inspired by the complex parallel structure of the human nervous system. Biological neural networks function by processing numerous input signals through a complex network of neurons in order to generate some useful output.

1.1 Predicting Secant Residual Friction Angle: A Nonlinear Problem

In geotechnical engineering, the prediction of drained secant residual friction angle ($ø_r'$) for cohesive soils has been identified as a nonlinear problem (Stark and Eid 1994). This parameter is useful in slope stability analyses for clayey soils as it is indicative of the soil's residual shear strength. While a drained torsional ring-shear test allows geotechnical engineers to test directly for $ø_r'$, the test is costly both in time and money. Thus, geotechnical engineers have sought alternative methods to predict $ø_r'$ and numerous correlations and mathematical models have been developed. Chief among these are a set of correlation-based equations that have been discussed most recently in a publication by Stark and Hussain (2013). A portion of this study examines these equations, and brings into question the robustness of this predictive model. This study provides an alternative predictive method for secant residual friction angle using a nontraditional modeling technique called an artificial neural network.

1.2 General ANN Structure & Training

Artificial Neural Networks (ANNs) attempt to mimic the physical structure of biological neural networks by creating a linked network of nodes, analogous to biological neurons in our nervous system. A generalized schematic of a singular biological neuron is compared to an artificial neuron in Figure 1 (a) & (b) respectively. Each input node represents a neuron, characterized by real or binary number values, defining its state of activity. These input states are multiplied by their respective weights in an attempt to link them with subsequent nodes. The magnitude of these weights, initiated as random numbers between -1 and 1, are used to define the relevance of the input neuron's state to the linked neurons. This neuron's state is determined by some mathematical accumulation of all the weighted states of the neurons connected to it, and then passed through what is known as a transfer (or activation) function. The state of this singular node is then passed to all connected nodes in the network (e.g. Figure 1 (b)) and the process continues for each node in the network.

With the functionality of a single artificial neuron defined, a network can be established by generating a multitude of interconnected nodes tasked with manipulating input data to map to target outputs. While the entirety of the how and why is not fully understood, the functionality of biological neural networks can be considered to be governed by both the physical structure of the network and rules governing inter-neural relations. This knowledge is applied to artificial neural networks in the way that they are both physically and mathematically structured (Neyamadpour, Taib et al. 2009).

Different ANN algorithms vary in both their structure, as well as the mathematical rules that govern neural interactions. Many tend to follow a parallel, networked architecture similar to that of Figure 2, which shows a two-layer network that receives data, stored in an input vector, at the input layer. The links connecting the input layer neurons and the hidden layer neurons represent a weight matrix. These weights govern the transfer of data between the two layers.

Finally, data are passed through the network layers and an output vector is computed. In many networks, a set of input data are used to *train* the ANN learning algorithm such that the error between the output vector and the known output is utilized to update the network's weights. These networks are known as *supervised* ANNs. Through updating, this error decreases and the network better approximates the relationship between input and outputs. While this training strategy is very common, a diversity of these governing strategies has been developed in recent decades and have led to the emergence of a diverse number of ANN algorithms that span a wide variety of applications including medical and engineering applications (Specht 1991, Basma, Barakat et al. 2003, Baykan and Yilmaz 2010, Bevilacqua 2013, Cheng 2013, Liu, Tian et al. 2013, Tamilselvan and Wang 2013).

1.2.1 Backpropagation Neural Networks

A simple, supervised learning algorithm called a Backpropagation Neural Network (BPNN) has been proven to be successful in mapping nonlinear data relationships. This ability has allowed the BPNN and other ANNs to develop acclaim in data analysis across many disciplines including many facets of engineering (Basma, Barakat et al. 2003; Doris, Rizzo et al. 2008; Neyamadpour, Taib et al. 2009; Baykan and Yilmaz 2010; Ceryan, Okkan et al. 2013; Torabi, Shirazi et al. 2013).

A literature review describing the types of ANNs used in geotechnical engineering is provided below in order to support the hypothesis that a BPNN could be successful in developing a predictive model that could outperform traditional statistical and empirical models.

1.3 Goals and Objectives

Following the success of BPNNs in other fields, it was posited that a BPNN may be successfully implemented to provide a predictive model for the nonlinear secant residual friction angle problem. Thus, it was the goal of this thesis to develop a BPNN for the prediction of ϕ_r' and evaluate its predictive capability. In order to fully evaluate a BPNN's ability to predict ϕ_r', goals for this thesis included the variation of input data parameterization in order to optimize BPNN predictions. Further, it was decided that the comparison of BPNN predictions for ϕ_r' should be compared to other predictive models in order to fully evaluate the BPNN's viability as a predictive method.

2. Literature Review

The classification of ANNs is dependent upon their learning algorithm, which is often characterized by the level of supervision experienced in this algorithm's learning. In this case, supervision refers to outside influence involved in a learning process. Supervision, in the computational sense, involves a mathematical dialogue between an algorithm's prediction of some output and the target output. Supervision allows the algorithm to minimize error in its predictions through an iterative training procedure, similar to a biological neural network's development of muscle memory through repetitive training.

The supervised learning procedure utilizes the error between predicted and known output values in order to correct the initially random network weights and more accurately predict outputs on the next iterative step. Weights created through successful iterative training can be saved and used to create output predictions for data that is unused in the training process. The generalized structure of the supervised ANN, shown in Figure 2, accurately describes the structure of a BPNN as well. For the application of predicting ϕ_r', this structure would be augmented to fit predictive input parameters with a singular output node for predicting $ø_r'$. Supervised learning requires a set of training data that comprise an input and corresponding target output data.

Unsupervised learning considers only the input information in its learning process, instead of utilizing a comparison of the network's prediction to a target output dataset. Using mathematical processes, usually clustering algorithms, the unsupervised algorithm recognizes patterns within the dataset, which it uses to extract key features and arrive at a conclusion from that data. The ability of unsupervised learning procedures to predict patterns make them ideal tools in handwriting analysis (Larochelle, Bengio et al. 2009), health diagnosis (Tamilselvan and Wang 2013), and other classification problems (Martinez, Bengio et al. 2013; Wang and Wang 2013).

Different algorithms utilize different levels of supervision, allowing for the classification of a learning algorithm to be considered on a continuum. Some algorithms are considered to be between supervised and unsupervised; semi-supervised networks meld a level of supervision into what are otherwise considered unsupervised learning processes, or vice versa (Zhu).

ANNs have experienced success in most major fields of scientific study including computer vision (Bevilacqua 2013; El-Baz and Tolba 2013; Fan, Ma et al. 2013), function approximation (Cheng 2013; Ghoreyshi, Jirasek et al. 2013; Mosleh 2013), time series approximation (Alessandri, Cervellera et al. 2013; Liu, Tian et al. 2013; Wang, Yan et al. 2013), and classification problems (Azami, Mosavi et al. 2013; Mishra, Dwivedi et al. 2013; Tamilselvan and Wang 2013). However, the focus of this research is centered on the usage of an ANN to predict the geotechnical parameter secant residual friction angle. Thus, the scope of this literature review remains within the field of geotechnical engineering.

2.1 Artificial Neural Networks in Geotechnical Engineering

Most geotechnical engineering applications of ANNs utilize a supervised learning strategy because it is useful in many applications where complex nonlinear relationships arise between input and target output parameters. By exposing the ANN to a series of measured input/output training patterns, geotechnical engineers are successful at accurately generalizing or predicting a solution for input with unknown outputs. The ability to generalize these nonlinear relationships is desirable in many geotechnical engineering scenarios when developing an accurate mathematical model is challenging, or understanding the physical relationship within a system is not yet possible.

2.1.1 Backpropagation Neural Networks (BPNN)

The most commonly used ANN in geotechnical engineering is the supervised Back-Propagation Neural Network, or BPNN. Its worldwide success in generalizing for problems with highly nonlinear solutions has established BPNNs as a powerful tool in problem solving for a wide area of geotechnical engineering applications. These algorithms have been successful in electrical resistivity imaging of soils (Neyamadpour, Taib et al. 2009), mineral identification (Baykan and Yilmaz 2010), prediction of unconfined compressive strength of carbonate rocks (Ceryan, Okkan et al. 2013), estimation of shearing resistance angle in uniform sands (Sezer 2013), landslide susceptibility analysis (Ramakrishnan, Singh et al. 2013), evaluating tunnel boring machine performance (Torabi, Shirazi et al. 2013), and prediction of ground subsidence due to mining (Yang and Xia 2013).

The BPNN's success in developing models that map nonlinear relationships between input and output parameters makes it a desirable alternative to generalized regression analysis when attempting to generate mathematical models for specific problem sets (Ceryan, Okkan et al. 2013; Sezer 2013; Torabi, Shirazi et al. 2013). In all cases, a BPNN was deemed more accurate in modeling a dataset when compared against linear and nonlinear regression models. While different studies involving the comparison of a BPNN and regression model used a variety of statistical benchmarks to compare accuracies of artificial networks to regression models, these studies all use a common benchmark: the coefficient of determination (R^2) to compare the two. Using the R^2 measure, Table 1 describes the modeling accuracies of BPNNs compared to regression models.

Unlike generalized regression analysis, BPNNs do not identify an explicit mathematical model to relate input data and outputs. Instead, the mathematics is internalized within the connections (i.e. weights) linking each neuron in the network. Instead, the user would pass input data the algorithm was not trained for to a trained BPNN, run it, and estimate output values as a result.

The inability to extract an explicit mathematical relationship between input and output data has been viewed as a major weakness of the BPNN. While a network may be superior in its ability to model the relationship between input and output parameters, little information regarding the relationship between these parameters can be derived from the ANN's mappings (Tu 1996). Regression analysis has the capability to provide R^2 values and associated p values that describe each parameter's predictive strength with respect to an output. This information can be very useful in many scenarios, especially where users are not interested in complete reliance upon a physics-based model to understand a complex relationship.

While unable to provide the user with an explicit mathematical relationship between input and output parameters, BPNNs retain their desirability as an alternative to many function derivation techniques due to their ability to manage problems where input data experience nonlinear relationships (Ramakrishnan, Singh et al. 2013). These nonlinearities pose a significant problem for multiple linear regression analysis. The linear assumption becomes a noticeable source of error between target output and the output determined by this type of model (Table 1). The complex, layered architecture of a BPNN accounts for the network's ability to process nonlinear relationships between parameters, creating a more accurate model of output data than a regression model could.

Neyamadpour, Taib et al. (2009), trained a BPNN to solve the complex problem of inversion of 2D resistivity imaging data. Inversion of data generated through electrical resistive tomography is complicated by the fact that conventional inversion techniques are known to predict nonexistent structures, so a BPNN was implemented as an alternative method of inversion. A configuration of two hidden layers, sized 28 and 16 neurons, were used to generate a singular output. The BPNN was relatively successful in generalizing a conductive vertical column in the soil; it exhibited anomalous behavior unlike that observed in conventional inversion, bringing into question the validity of the BPNN's generalization. It is suspected that the BPNN learning algorithm was an inappropriate alternative to traditional inversion due to the fact that the network was tested for various sizings and none seemed to remove anomalous inversion behavior.

Bakyan and Yilmaz (2010) utilized a BPNN in mineral identification using image analysis. By feeding the BPNN information on color, hue, saturation, and value, the network was shown to experience training accuracies as high as 81-98%. Though, it showed some weaknesses when distinguishing like-colored minerals such as muscovite, biotite, and chlorite.

Ceryan, Okkan et al. (2013), successfully applied a BPNN to the prediction of the unconfined compressive strength of carbonate rock. Using only two input parameters (total porosity & P-wave velocity); the network was significantly more predictive than a multiple regression approach. Whereas Sezer (2013) used a BPNN to create a simple estimation model for the shearing resistance angle of uniform sands alongside another type of neural algorithm known as an Adaptive Neuro-Fuzzy Inference System and a multiple regression model. Provided with size and shape characteristics for the sand grains, both neural network models prevailed over the regression model. A BPNN structure using one, 10-node, hidden layer provided the most effective mapping of the relationship between sand characteristics and shearing resistance angle.

Ramakrishnan, Singh et al. (2013), created a BPNN-based model for landslide susceptibility in the Tawaghat area of India. By combining GIS data; slope, lithology, aspect, land use, and known areas of fracture, with known landslide triggering parameters: rainfall, seismicity, anthropogenic interference areas were categorized into zones of high, moderate and low landslide susceptibility. Landslides that occurred after the development of the model showed that 80% of the landslides occurred in zones designated as high susceptibility areas.

Torabi, Shorazi et al. (2013) used a BPNN to study the influence of geotechnical parameters on tunnel boring machine performance in a highway project. This study was successful in identifying that the main parameters related to boring machine downtime were not related to the rock, but instead the social aspects of the tunnel boring project. Though in this case, the rock material being tunneled through was described as 'moderate-to-weak structural condition', suggesting that the viability of a BPNN-driven study of geotechnical parameters for tunneling could increase with tougher rock.

Yang and Xia (2013) implemented a BPNN to predict mining subsidence under thin bedrock and unconsolidated layers. Combining known bedrock and unconsolidated soil layer parameters with information regarding the rate, depth, size, and management of mining operations, they were successful in measuring the horizontal movement and subsidence of nearby layers. Using a genetic algorithm (GA), the authors optimized the learning parameters and structure of the network to utilize one hidden layer, comprised of 32 nodes.

2.1.2 Recurrent Artificial Neural Networks: Time Sensitive BPNNs

Certain complex problems are convoluted by a path dependency. This dependency, often manifesting itself as a temporal component, outlines a relationship between input and output that is recurrent. In the recurrent case, the output for a given time step are key in determining the output for the following time step. For instance, a soil's level of swell at any given time step is heavily dependent upon the swell observed at a previous time step (Basma, Barakat et al. 2003). One common, path-dependent relationship experienced in geotechnical engineering involves the mechanical behavior of soils over time. This requires that a model of these behaviors be able to consider outputs from a previous time step as inputs for the following time step. While these problems experience nonlinearities for which a traditional BPNN may be well suited, they are not capable of considering this time dependency without adaptation of the network structure. The Recurrent Artificial Neural Network (RANN) is a BPNN adapted to include a recurrent, or time dependent component. The network considers certain predicted outputs (for time step t) as inputs for the following time step (time step t+1) as shown in Figure 3. The RANN creates a complex nonlinear mapping with consideration of a temporal, or path-dependent component.

The way in which a soil swells, shrinks, and responds to loading is time dependent in nature. By recurrently considering the output variables deviatoric stress (q), soil suction (U_a-U_w), and volumetric strain (ε_v) as input variables for the following time step, the RANN successfully maps the time dependent nature of an unsaturated soil (Johari, Javadi et al. 2011).

Basma, Barakat et al. (2003), developed a RANN for the purpose of creating a predictive model of swell in clay soil samples. This study was performed in comparison to a traditional BPNN and confirmed that the prediction accuracy of a RANN is greater than BPNNs for time dependent processes. The authors went on to conclude that a RANN may be highly successful in mapping relationships for other time dependent geotechnical applications such as prediction of soil consolidation, pore pressure dissipation, and seepage.

Zhu, Zaman, et al. (1998), used a RANN to predict soil properties for both fine-grained residual soil and coarse-grained dune sand. Their model proved successful in accurately characterizing these soils for various axial and volumetric strains, making the network robust in predicting hardening and softening behavior associated with loading-unloading-reloading conditions. This model was once again compared to and outperformed a traditional BPNN in the modeling of this time dependent problem.

Habibagahi & Bamdad (2003), implemented a time dependent mechanical soil behavior model and concluded that it was capable of modeling various parameters as well as simulating collapse phenomena related to unsaturated soils. The authors had success in sequentially simulating the deviatoric stress, volumetric strain, and change in suction that occurs in triaxial tests. The results are promising in that the network provided good performance for the prediction of the results of triaxial tests and understanding specific collapse phenomena.

Johari, Jadavi et al. (2011), implemented a RANN to predict the soil mechanics for unsaturated soils. They paired this approach with a genetic algorithm (GA) used to optimize the RANN learning parameters and architecture to provide the best mapping for this problem. This was compared to the RANN created by Habibagahi & Bamdad (2003) using the traditional trial and error paradigm common in optimizing many ANN structures. The GA optimized RANN prevailed in predictive accuracy for unsaturated soils, suggesting that ANN performances, while already acceptable in terms of predictive accuracy, can be better optimized through the use of genetic algorithms.

2.1.3 Unsupervised Self-Organizing Maps (SOMs): A Minority in Geotechnical ANN Applications

Use of unsupervised learning algorithms to classify data is rare in geotechnical engineering applications, though they have been used some. An ANN known as a Self-Organizing Map (SOM) is a nonparametric alternative to classical clustering statistical techniques. It classifies complex, multidimensional data into groups. This process, known as clustering, categorizes multidimensional datasets into distinct groupings and often displays them on a one or two-dimensional platform, useful for human interpretation (Figure 4). The clusters that are created also provide the user with a level of dimensional analysis, allowing them to understand which parameters are most important in the clustering process (Figure 5). Here, the a clustering process has been described in terms of each separate parameter passed to the network (i.e. sand %, depth (cm), $CaCO_3$ %) to better describe the relationship between the clusters that the SOM has generated, and the specific input parameters that generated the clusters.

Ferentinou, Hasiotis et al. (2012), used a SOM to classify marine sands in the Zakynthos canyon in Greece. Five distinct clusters were identified (CL1-CL5 of Figure 4), allowing soil scientists to understand how to classify certain areas of the Zakynthos region and make inferences about the sediment's interaction with the topology of the area. While it was difficult to predict trends in the vertical distribution of sediment, they were able to infer that the sediments observed in their sampling of the Zakynthos region were likely to occur laterally throughout the studied area.

3. Application: Prediction of Drained Secant Residual Friction Angle

3.1 What is Secant Residual Friction Angle?

The drained residual shear strength of cohesive soils is often needed for the determination of the susceptibility to landslides and other critical failures (Skempton 1985; Stark and Eid 1994). A cohesive soil's typical shear stress versus displacement response is depicted in Figure 6. After reaching a critical condition, a clayey soil's shear strength reaches a weaker residual condition. This residual condition occurs when originally jumbled clay platelet particles become aligned on a parallel plane, contributing to a plane of significantly reduced shear strength (Skempton 1985). This condition is considered to be a residual soil condition, in that once this plane has been produced, it will continue to exist as a plane of significantly reduced stability until it is subjected to a large enough loading to produce a failure. Therefore, it is important to quantify a clayey soil's residual properties to perform a complete stability analysis.

Figure 7 provides a generalized residual shear stress to normal stress envelope for a cohesive soil. Note that the envelope is typically nonlinear, making it very difficult to quantify a soil's failure envelope using a singular parameter. Instead, a soil's residual shear strength is matched to its corresponding normal stress loading. This relationship is commonly expressed as the secant of the shear stress divided by the normal stress and hence referred to as secant residual friction angle, or $ø_r$'.

Typically, a drained torsional ring shear test is used to determine the $ø_r$' of a remolded soil sample (Lupini, Skinner et al. 1981; Stark and Hussain 2013) for a given normal stress. The sample is subjected to a constant normal stress loading, while a constantly increasing shear stress is applied to the soil until a shear failure plane is developed. A general schematic, which describes the mechanistic procedure used in torsional ring shear stress testing, is shown in Figure 8.

In order to ensure that a reliable test is performed, the shear displacement is applied at relatively slow increments. With rates as slow as 0.02 mm/min, a test may take weeks to reach a residual condition. Not only is the drained torsional ring shear test a slow process, both the apparatus and testing are financially costly. These disadvantages have prompted many geotechnical engineers to seek alternative methods to the ring shear test. It has been determined that, especially for clay soils, there is a relationship between various common geotechnical soil parameters and $ø_r$'. These relationships have the potential to be exploited to indirectly predict $ø_r$' without use of traditional torsional ring shear tests, saving both time and money for the interested party.

3.2 Predictive Soil Parameters

Previous studies have related drained residual friction angle ($ø_r$') to a soil's index properties (Skempton 1985; Collotta, Cantoni et al. 1989; Stark and Eid 1994; Mesri and Shahien 2003; Wesley 2004; Tiwari and Marui 2005). A soil's $ø_r$' is known to be affected by its mineralogy, composition, and magnitude of effective normal stress applied during soil testing (Dewoolkar and Huzjak 2005). Thus, certain soil parameters that quantitatively describe one or more of these soil characteristics have been related to $ø_r$'. These predictive soil parameters are provided in Table 2.

3.2.1 Mineralogy: Atterberg Limits – LL & PL

Atterberg limits are common geotechnical soil parameters that are indicative of a soil's mineralogy and ability to endure stress loadings and thus, of direct concern to $ø_r$' (Stark and Eid 1994; Wesley 2004; Stark, Choi et al. 2005). Atterberg limits are significant in geotechnical engineering as they provide information related to a soil's behavior at critical water contents. These limits are important for soils that contain a significant clay mineralogical component because clay-water interactions heavily dictate the mechanical properties of a soil (Moore 1991; Stark and Eid 1994). A soil that contains a significant clay mineralogical component experiences polar interactions between clay particles contributing to cohesion and plastic behavior between soil particles (Moore 1991). Such a soil is said to be cohesive.

Originally, studies concerned with determining a relationship between $ø_r'$ and other soil parameters considered liquid limit (LL) without plastic limit (PL) (Stark and Eid 1994; Wesley 2004). More recently, studies have considered combining both basic Atterberg limit parameters using the lumped plasticity index (PI). Defined as the difference between LL and PL, PI has been correlated to $ø_r'$. (Tiwari and Marui 2005; Stark and Hussain 2013). The predictive capacity of PI suggests that the consideration of PL as an input parameter provides some predictive capacity beyond what is supplied by LL alone.

3.2.2 Composition: CF & SF

Because the quantity and type of soil particles determine a soil's strength characteristics, the composition of a cohesive soil also contributes to a soil's $ø_r'$. As is apparent when considering Atterberg limits, clay composition has a strong effect on a soil's mechanical properties. Where Atterberg limits may indirectly inform an analyst about the mineralogy and quantity of clay within a soil sample, the clay fraction (CF) quantifies the amount of clay material by mass. Expressed as a percentage, an increase in CF is known to cause a decrease in $ø_r'$ (Dewoolkar and Huzjak 2005). Equations developed between $ø_r'$ and other geotechnical parameters commonly implement CF as a predictive parameter within their models (Skempton 1985; Collotta, Cantoni et al. 1989; Stark and Hussain 2013).

While not commonly used in studies that correlate $ø_r'$ and indicative soil parameters, sand fraction (SF) shows potential to be a predictive parameter. Unlike clay particles, sand particles exhibit no cohesive properties. In large enough proportion, these particles can exhibit stability characteristics dissimilar to those caused by clay particles (Nelson 2013). Because sand particles are expected to react differently to shear loadings than clays, it is hypothesized that a model that incorporates SF as an input parameter may experience added predictive capability.

3.2.3 Magnitude of Normal Stress Loading: σ

Applying a normal stress loading (σ’) to a soil sample is an integral part of the ring shear test that traditionally determines $ø_r'$. This is because a well-established relationship exists between shear stress (τ) parameters and σ’, where σ’ is the effective normal stress loading:

$$\boldsymbol{\tau = \sigma' tan(ø_r')} \tag{1}$$

Thus, a soil's shear response, commonly characterized through $ø_r'$, is dependent upon σ’. As a result, some studies that establish relationships between $ø_r'$ and common geotechnical parameters include σ’ in their correlations and predictive models (Collotta, Cantoni et al. 1989; Mesri and Shahien 2003; Wesley 2004; Dewoolkar and Huzjak 2005; Tiwari and Marui 2005; Stark and Hussain 2013).

3.3 Prediction of Secant Residual Friction Angle

Given that a torsional ring shear test is both time and money intensive, alternative methods for determining $ø_r'$ have been explored for decades. The observed relationships between the soil parameters (Table 2) and $ø_r'$ provide grounds for developing predictive models for $ø_r'$. Over the past five decades, studies have developed correlations between $ø_r'$ and various input parameters seen in Table 2.

While useful empirical equations that relate indicative soil parameters and $ø_r'$ have been created, a physics-based mathematical relationship has not been derived. Most studies consider a multitude of parameters in developing predictive models for $ø_r'$ (Collotta, Cantoni et al. 1989; Mesri and Shahien

2003; Wesley 2004; Tiwari and Marui 2005; Stark and Hussain 2013) and despite significant progress, the predictive ability of many of these models are suspect. A discussion of Stark and Hussain's (2013) equation, one of the more widely accepted predictive models, is provided. Failure to fully understand the physics behind this mathematical relationship is due to the complex nonlinear relationship between $ø_r'$ and the soil parameters described in section 3.2.

A brief glimpse into the complex nonlinear relationship between these parameters is outlined in Figure 9. In this study, $ø_r'$ is a function of LL for varying ranges of CF and σ' loadings. Where soils with CF≤20% exhibit a strong linear relationship, soils with CF≥50% exhibit nonlinear behavior across different normal stress loadings between the 60-120% LL range. While this may not seem like a very complex nonlinear relationship, it is important to note that this analysis and two-dimensional representation does not consider other predictive parameters such as PL and SF. The complexity increases with the addition of these other useful parameters. A multivariate least-squares regression analysis, described in 4.1.2, supports this, suggesting that a more mathematically complex model is necessary.

3.3.1 Current Predictive Methods

Currently, predictive methods for $ø_r'$ exist mainly in the form of empirical equations developed from soil datasets acquired from the field and some manufactured or altered in the laboratory setting. As discussed above, these relations are sometimes based on a single parameter. These are often incapable of generalizing across a wide range of soil types (Stark and Eid 1994; Wesley 2004).

A commonly used multivariate model for the prediction of $ø_r'$ are equations that have been developed over the past 20 years and presented in a series of three publications. First posited in the Stark and Eid (1994) publication, these equations have been revisited both in Stark et al. (2005) and most recently Stark and Hussain (2013). These studies provide equations for the prediction of $ø_r'$ using LL and σ' as input data across the three distinct CF ranges shown in Figure 9. As discussed in the results section 5.3.2, even these equations can be shown to be limited in their prediction of $ø_r'$ across a wide range of soil types. It may be argued that even this multidimensional consideration of input soil parameters fails to address the complex nonlinearity of the multidimensional relationship between input soil parameters and $ø_r'$. The shortcomings experienced by these predictive methods provide grounds for seeking an alternative method of prediction. The following section concerns the development of a BPNN for the prediction of $ø_r'$.

4. Methodology

Similar to a majority of the ANN geotechnical applications provided in the literature review, this study utilizes a BPNN to develop a predictive model for $ø_r'$. Prior to the development of this model, soil data were acquired; and the soil datasets used in this study were prepared and subjected to preliminary statistical analyses. These statistical analyses provided correlations between individual input parameters and $ø_r'$, to help inform the parameterization of the BPNN. In addition, a multivariate linear regression model was developed to provide an alternative model for comparison to the BPNN. Further, code was developed to use Stark and Hussain's 2013 equations to predict $ø_r'$ for the data in this study, providing a second model to compare BPNN performance to.

4.1 Methods for Soil Data Acquisition

Three major testing procedures were required for the acquisition of soil parameters considered in this study. These tests were not performed in this study, but were performed by the researchers who provided these data. $ø_r$’ angles were determined using a drained torsional ring shear test conducted at certain σ’ (Figure 8). The American Society for Testing and Materials (ASTM) provides standard methods for the torsional ring shear test in ASTM D6467 – 13. Often soil samples are remolded and pre-sheared to expedite the process of attaining a residual shear condition in the soil. Both LL and PL soil parameters can be determined by following ASTM standard method: ASTM D4318 – 10e1. Both CF and SF parameters are measured by performing a simple soil gradation analysis (ASTM D6913 – 04), which generally includes sieve and hydrometer analyses.

4.2 Data Preparation

In order to conduct this study, six viable soil datasets were acquired from various academic and commercial sources. These datasets come from various soil parameter studies, most related to developing models and correlations for the prediction of $ø_r$’. While this study enjoys the advantage of a large pool of data, the fact that these data come from various sources requires careful consideration and pre-processing.

In terms of data quality, the datasets may be divided into two major categories: internally consistent and inconsistent. Internally consistent datasets use data for soil samples that have been tested using the same methodology, equipment, and ideally, personnel. By ensuring that soil parameters are determined in an internally consistent manner, one can be confident that, relative to the specific dataset, data are free from systematic experimental error and skew. In this way, the quality of internally consistent data is superior and preferred to its internally inconsistent counterpart.

Though four of the six datasets considered are internally consistent, two internally inconsistent datasets were used in this study to increase the size and breadth of data used to train and test a BPNN. All of these datasets included LL, PL, CF, and σ’, though some lacked SF information. The total dataset was divided into two major subsets; those that include SF data and those that do not. Datasets used in this study are listed in Table 3 with their sources, abbreviated labels, and sample size provided.

Despite the fact that both the Commercial and the Dewoolkar dataset are potentially internally inconsistent, they provide a significant amount of data to both the subset of data with and without SF. Despite the fact that the Stark and Eid (1994) dataset is an amalgamation of soil data from various sources, testing methods remained internally consistent. Further, this dataset is vital due to its use in developing the Stark and Hussain (2013) predictive equations for $ø_r$’.

Each dataset was individually trained using a unique BPNN. Further, the two larger data subsets (D, H, S, TM03 and D, H, S, SE, TM03, TM05) were trained to the BPNN to compare the predictive capacity of SF. Some of these datasets were further subdivided due to their internal consistency as is discussed in the results. Prior to BPNN training, soil data was subjected to statistical analyses to inform BPNN parameterization.

4.3 Multivariate Statistical Analysis of Soil Datasets

Statistical analyses were performed on the soil datasets in question to acquire knowledge about various soil parameters’ ability to predict $ø_r$’, as well as develop a benchmark comparison between a BPNN and more common parametric statistical methods. To provide a robust dataset for statistical analysis, all available datasets were combined into one large dataset. This dataset includes Dewoolkar, Hayden, Commercial, Stark & Eid, Tiwari 2003, and Tiwari 2005 datasets. All correlation and multiple regression statistical analyses were performed using JMP Pro 10 software.

4.4 Development of a BPNN for Prediction of Secant Residual Friction Angle

Preliminary statistical analysis of the soil data informed the development of a BPNN with the purpose of predicting $ø_r$' using LL, PL, CF, SF and σ'. Noticeable correlations were identified between $ø_r$' and each parameter. Two BPNNs with distinct input parameter scenarios were considered and generalized schematics for each network are provided in Figure 10. In one network, a soil's LL, PL, CF, SF, and σ' were provided as input parameters (Figure 10 (a)). The other BPNN considered LL, PL, CF, and σ' without the use of SF (Figure 10 (a)). Further, the use of these two networks allowed for the comparison of two predictive models in an attempt to determine whether SF is an effective or necessary parameter in the determination of $ø_r$'.

4.4.1 BPNN Structure: Configuration & Sizing

The generalized structure of the BPNNs used in this study (Figure 10) each consist of an input layer where input data parameters are passed to the network, a hidden layer where input information is processed and passed to a singular terminal node in the output layer. This node holds the network's prediction of $ø_r$' and the quality of this prediction is partially dependent on the network's structure; defined by its configuration and sizing.

In this context, configuration pertains to the number of layers within the network, while sizing pertains to the number of nodes in each of these layers. Both of these criteria determine the level of mathematical complexity in the predictive relationship developed by the BPNN. While the size and configuration of the input and output layers are determined by the application at hand, these criteria may vary for the hidden layer(s). BPNNs have the potential to utilize a configuration that contains multiple hidden layers to allow for higher levels of complexity. Similarly, the sizing of each of these layers may be varied. Generally, an increase in either configurations or sizing generates a network capable of developing relationships of higher mathematical complexity.

A mathematical theorem of Irie and Miyake (1988) suggests a network configuration that incorporates two hidden layers (Figure 11), for large enough node sizings, has the capability to map any mathematical relationship between inputs and outputs. Despite this, the ability of a network with two hidden layers to outperform a single hidden layer network is suspect. Table 4 shows the performance summary for BPNNs of various size and configuration. Note that the minimum RMS error for a one layered network is not improved upon by adding a second hidden layer. Clearly, the variation of both of these parameters can make the optimization of network structure complicated and subjective.

To avoid excessive complexity in the network structure optimization process, this study focuses on varying only network sizing, and utilizes one hidden layer in its configuration. The optimization process shown in the next section proves that this structure is sufficient to develop a BPNN of sufficient mathematical complexity for this problem. A backpropagation neural network's input layer is comprised of an artificial neuron or node per input parameter considered in the neural network. The values for these input parameters are passed to the network's input layer and sequentially passed through to the output layer to generate predictions.

4.4.2 Data Pre-Processing

BPNN network training occurs most efficiently when network calculations remain bounded between 0 and 1. To ensure that network training occurs efficiently, input values that may originally occur along any continuous scale, are first normalized to values between 0 and 1 as shown in Equation 2, where x is the parameter data value, x_{norm} is the normalized data value, $x_{max}$is the maximum data value, and x_{min} is the minimum data value:

2

$$x_{norm} = \frac{x - x_{min}}{x_{max} - x_{min}}$$

4.4.3 Training and Prediction

For a given dataset, training occurs on a random subset of the data while the remainder is withheld for testing. This is done to ensure that the network is mapping to the general relationship between input data, rather than learning specific data values. If all data within a dataset were trained to a network, it would be possible for the network to map the mathematical relationship between the specific input and output data provided without learning underlying general relationship between parameters. By testing the network's performance on a withheld data subset, one may ensure that a BPNN is generalizing by observing the model's predictive performance for the data that was untrained to the network.

In backpropagation training, initial network weights are randomly assigned values between -1 and 1. Input parameters are then multiplied through to subsequent network layers by their corresponding weight matrices. In order to ensure that the values remain between 0 and 1, the values calculated for the subsequent layer are passed through what is called a logistic squashing function. Seen mathematically in Equation 3 and visually in Figure 12, a logistic squashing function forces calculated node values to fit between 0 and 1. In a process called forward propagation, these operations are sequentially applied to input values through the network until operations terminate on the final layer. Equation 3 is as follows, where x is the nodal data, and e is the exponential function:

3

$$f(x) = \frac{1}{1 + e^{-x}}$$

Values calculated at the final layer are said to be the network's output predictions. Note that these values have been normalized between 0 and 1 and must be un-normalized to correctly represent the predictions made by the network. This is accomplished using a simple un-normalization function seen in Equation 4, where x is the parameter data value, x_{norm} is the normalized data value, $x_{max}$is the maximum data value, and x_{min} is the minimum data value:

4

$$x = x_{norm}(x_{max} - x_{min}) + x_{min}$$

Once a set of predicted outputs is correctly scaled to match known outputs, the backpropagation updating procedure begins.

Where forward propagation runs from input layer to output layer, backpropagation runs in reverse, hence the name. Backpropagation begins with a calculation of error between the known and predicted output values, which informs the updates applied to all weights in the network. The first error calculation (Equation 5) is a function of the difference between known and predicted output, where δ_{out} is the output layer error, $f'(x)$ is the derivative of the logistic squashing function, o_k is the known output vector, and o_p is the predicted output vector:

5

$$\boldsymbol{\delta_{out} = f'(x)(o_k - o_p)}$$

Subsequent error calculations precipitate from this (Equation 6), where δ_{hid} is hidden layer error, $f'(x)$ is the derivative of the logistic squashing function, δ_{hid} is the output layer error, and w_{out} are the output weights:

6

$$\boldsymbol{\delta_{hid} = f'(x)[\delta_{out} w_{out}]}$$

The weight adjustments are a function of these error calculations following the general form shown in Equation 7, where Δw_{ij} is the change in weight matrix between layers i and j, η is the updating coefficient (generally 0.5), δ_j is the error for forward layer j, and x_i are the current values for nodes at backward layer i:

7

$$\Delta \boldsymbol{w_{ij} = \eta \delta_j x_i}$$

The network's new weight matrices are then calculated by simply adding the changes in weights to the previous weight states (Equation 8), where w_{ij}^{new} is the new weight matrix between layers i and j, w_{ij} is the current weight matrix between layers i and j, and Δw_{ij} is the change in weight matrix between layers i and j:

8

$$\boldsymbol{w_{ij}^{new} = w_{ij} + \Delta w_{ij}}$$

4.4.4 Training Criteria: Root Mean Square Error

Through iterative application, the backpropagation procedure is successful in lowering the error between known and predicted output. A simple error metric, known as Root Mean Square Error (RMSE), is used to evaluate the BPNN training progress (see Figure 13). A user-defined threshold RMSE is commonly used to decide when training ends. At the end of any iteration, if the RMSE is less than or equal to the threshold, training terminates and network weights are saved.

During this BPNN training, a threshold RMSE is used in conjunction with a maximum iteration criterion. Since a network may never reach the threshold RMSE, a 10,000-iteration limit was applied to ensure that network training would occur for a finite amount of time.

4.4.5 Network Improvements: Noise, Momentum & Bias

The BPNN used in this study includes three major improvements to the general BPNN structure described in Figure 10. These improvements called noise, momentum, and bias, allow the network to train more efficiently by allowing the BPNNs error descent to occur over less iterations.

Noise

A major problem with training a BPNN is the issue of specification versus generalization. Even for a large datasets of training patterns, iterative BPNN training has the potential to create a set of weights which have been overfit to a training set. In this way, a network's weights may have developed the information necessary to replicate patterns used in training very accurately, but poorly predict untrained values. The purpose of the network is to be able to generalize. To aid in generalization during the training process, randomness is introduced into the training patterns passed to the network. This randomness, or noise, causes the BPNN to become less inclined to specify, because it will never train on exactly the same inputs (Reed and Marks 1998). This noise can be applied in various ways, though in this study, a singular input value is randomly chosen from a training pattern and varied during each training iteration. The values are only changed by ±0.1% of their original value to ensure that training patterns retain their relevance to the values they represent.

Momentum

Iterative BPNN training can be costly in terms of both time and computation and techniques that can speed up the training process have been sought after and developed. Momentum attempts to utilize information in a previous training step to quickly force the BPNN's descent toward lower error. Within the weight updating procedure in the backpropagation process, the weight adjustment, Δw_{ij} is supplemented with a momentum term (Equation 9), where Δw_{ij} is the change in weight matrix between layers i and j, η is the updating coefficient (generally 0.5), δ_j is the error for forward layer j, x_i are the current values for nodes at backward layer i, α is a momentum coefficient (generally 0.9), and Δw_{ij}^{prev} is the change in weight matrix from the previous training step:

9

$$\Delta \boldsymbol{w_{ij}} = \boldsymbol{\eta \delta_j x_i} + \boldsymbol{\alpha} \Delta \boldsymbol{w_{ij}^{prev}}$$

By adding a term that considers the weight changes used in the previous iteration, changes in weights have the potential to occur more quickly, allowing the training procedure to gain 'momentum'. If the newly calculated and previous changes are of the same sign, changes in the coming iteration experience a more significant change than they would without momentum. If the signs of these two terms conflict, this momentum dissipates.

Momentum provides a secondary utility in helping the network avoid and escape from error 'ruts'. If one were to consider an RMSE training curve for a network, like the one in Figure 14, one would notice that there could be local minima between the initial network weight states and a more global minimum. Without momentum, a network in training would be more likely to get stuck in these local minima, and fail to find a weight state that provides even lower RMSE.

Bias

Another way to improve a BPNN's training speed is by applying bias to the network. Analogous to an intercept in a linear equation, a unique bias term is added to each hidden node within a network. The addition of the bias term in the forward propagation procedure is known to increase the training speed of a network, as it shifts the logistic squashing function (Figure 13). The bias node is connected via weights to all other hidden nodes within the BPNN. The bias node always has a value of 1, though the weights associated with it are updated following the same backpropagation procedure as all other weights in the network. A schematic diagram of a BPNN equipped with bias is shown in Figure 15.

4.5 Optimization of BPNN

A BPNN's structure must be optimized to ensure the best possible predictive model. In the successful training of a BPNN, the RMSE associated with the error between known and predicted outputs descends to a desirably low value. In varying the number of nodes in the BPNN's hidden layer, one may be able to find the node sizing that provides an optimally low RMSE.

Since the BPNN is developed for prediction of $ø_r$', the network's predictive performance must be evaluated. A subset of data must be withheld from the training process for testing and validation and a BPNN may be biased toward data used in the training process. To ensure the network's ability to accurately predict, the data subset that is withheld from training is used in a testing procedure. At the end of each training iteration, the network should be able to accurately predict the values of the testing subset. This prediction error, called testing RMSE, is iteratively calculated alongside training RMSE to provide training performance metrics. A BPNN that is training correctly experiences a testing RMSE that follows a similar trend to that observed in the training RMSE. Though the testing RMSE follows the same trend as training RMSE, it is systematically higher, due to the predictive bias provided to trained data. Optimal network sizing can be determined by identifying the network sizing that generates with the lowest terminal value of testing RMSE.

Networks with a smaller number of hidden nodes than the optimal BPNN may not be capable of mapping the mathematical relationship between input and output parameters, leading to an undesirably high terminal RMSE. It is desirable to keep the network's hidden layer as small as possible to avoid overtraining. Overtraining occurs when a network's structure provides a level of mathematical complexity that exceeds what is necessary to map a relationship. This mathematical complexity conflicts with the network's ability to accurately map the relationship between input and output parameters, resulting in high RMSE.

An example of the structural optimization performed in this study is shown in Figures 16-21. These figures show the notable iterative RMSE testing values as the number of hidden nodes varies from 3 to 25. The optimal hidden layer nodes sizing for this particular model and training data set was determined to be a 6 node BPNN (Figure 19) due to its low training RMSE of 0.1351 at training termination. Networks smaller than this were incapable of reaching such a low testing RMSE and networks larger than this were incapable of improving on this error. In Figure 21, a 13 node BPNN's testing RMSE departs significantly from the training RMSE. This shows an extreme example of overtraining, where the network's ability to correctly train is inhibited by its mathematical complexity. The same optimization procedure was used for all BPNNs developed in this study.

4.6 Stark & Hussain (2013) – Equations for the Prediction of $ø_r$'

An alternative to the BPNN model developed in this study is the correlation-based model provided by Stark and Hussain (2013). Using the Stark and Eid dataset, these equations were developed for specific CF and PL data ranges at discrete σ' loadings. The equations are shown in Figure 22.

Note that there are gaps between CF data ranges. These equations were adapted to encompass all data by redefining the low, middle, and high CF ranges to CF≤23, 23≤CF≤48, CF>48. These adapted equations were embedded into a MATLAB code that enables interpolation between σ' loadings to predict $ø_r$'. This along with all other code relevant to this study can be found in Appendix B.

5. Results

Analysis of the predictive performance of the BPNNs developed in this study is discussed below. Major areas of analysis included:

1. Preliminary statistical data analysis
2. CF distribution across BPNN prediction space
3. BPNN performance with and without the SF input parameter
4. BPNN predictive capability as compared to:

a. Linear multivariate regression analyses
b. Equations developed in Stark & Hussain's 2013 publication

5.1 Preliminary Statistical Analysis

4.1.1 Correlation Analysis

First, a correlation analysis of soil parameters was performed to determine which input parameters (independent variables: LL,PL,CF,SF,σ') showed with Secant Residual Friction Angle ($ø_r'$). Figure 23 demonstrates this analysis, individually comparing each parameter against the remaining parameters. Of greatest concern is the comparison between input parameters and $ø_r'$, as is seen along the first row and column. Correlation coefficients are provided for each parameter to mathematically describe these relationships. The strength of the correlation between the two parameters increases as the absolute value of the correlation coefficient approaches 1. These correlations are also visualized through the ellipsoids transposed correlation plots. A slanted and thin ellipsoid surrounding data points indicates a strong correlation between the two parameters. This analysis shows that liquid limit, plastic limit, and clay fraction are all highly correlated with $ø_r'$, while sand fraction is identified as slightly less correlated. It is clear that effective normal stress is not as predictive as the other parameters, though $ø_r'$s dependence upon normal stress loading, seen in Equation 1, has been well established in geotechnical engineering (Tiwari, Brandon et al. 2005). It is therefore important to retain the normal stress loading parameter when trying to establish predictive models for $ø_r'$.

In the literature, BPNNs are commonly compared to multivariate linear regression models to provide a benchmark for comparison of predictive capacity (Ceryan, Okkan et al. 2013; Sezer 2013; Torabi, Shirazi et al. 2013). The next section describes the least squares multivariate linear regression model of secant residual friction angle as a function of input soil parameters (σ', LL, PL, CF, and SF) performed using JMP Pro 10 software.

4.1.2 Least Squares Multivariate Linear Regression

Two least squares regressions were performed on the dataset. The first version included all five input parameters for a total of 195 data points from four datasets (Dewoolkar, Hayden, Commercial, and Tiwari 2003). A larger dataset was available for $ø_r'$ which did not include sand fraction, allowing for a total of 388 data points from 6 datasets (Dewoolkar, Hayden, Commercial, Stark & Eid, Tiwari 2003, and Tiwari 2005). Least squares models were generated for each of these datasets individually, and as a whole to understand the predictive relationship between these variables.

In all cases, the models' p-values are extremely low ($p<0.0001$), suggesting that, from a linear statistical sense, the models have successfully defined a linear model between secant residual friction angle and the input soil parameters. However, the 1 to 1 plots of actual vs. predicted $ø_r'$ in Figure 24 show these models poorly estimate the relationship between soil parameters and $ø_r'$. This lack of fit can be quantitatively described by considering the adjusted R^2 parameter associated with the predictive model error. For multivariate analysis, the adjusted R^2 value is considered because it accounts for the number of explanatory terms (input variables) in the model. Although the adjusted R^2 values **(**ranging from 0.413 to 0.426) suggest some relationship between explanatory soil parameters and $ø_r'$, the linear regression model does not provide a robust predictive method for determining $ø_r'$ (Figure 24 (a) & (b)).

Models for this dataset were created with and without consideration of the input soil parameter, sand fraction, to demonstrate that sand fraction provides useful predictive capacity to these predictive models, as is seen by the decrease in adjusted R^2 value when sand fraction is no longer considered as an input parameter (Figure 24).

While multivariate linear regression does not successfully develop a robust predictive model, it is successful at quantifying a level of importance for each input parameter on $ø_r'$, making the regression model useful, despite its predictive shortcomings.

5.1 Analysis of CF distribution across BPNN Prediction Space

Analysis of the predictive performance of the BPNNs developed in this study is discussed in this section. In various models, including the widely accepted equations provided in Stark & Hussain (2013), prediction of $ø_r'$ requires that soil data be divided into data ranges. Stark and Hussain's predictive equations vary for different ranges of CF. Within each of these CF ranges there are equations for 3 or 4 discrete σ' loadings that consider LL as an input parameter.

It was posited that in the same way that the predictive ability of Stark and Hussain's equations depends on this subgrouping of input variables, the BPNN may benefit from division of data as well. It was decided that multiple BPNNs could be developed for various CF ranges. By doing this, the BPNNs may have been more streamlined for prediction of the different phenomena observed in different CF data ranges. It is clear that various levels of nonlinearity exist for the three data ranges graphically described in the ellipses of Figure 25. While this is a simplification of the complex, multidimensional nonlinear relationship between $ø_r'$ and the input parameters considered in this study, this figure shows the potential for distinct subgroupings of data.

The subgroupings are implemented, using CF distributions modified from the three used in Stark and Hussain's 2013 publication (Figure 25). The success of this method was investigated by comparing it to a BPNN that used all provided data and BPNNs that were trained on data subsets determined by these CF ranges shown in bold in Table 5.

The CF distribution analysis began with the consideration of a successfully trained BPNN's CF distribution across its entire prediction space. Division of data three into distinct CF-based subsets significantly reduces the size of the data available to the network during training. Therefore, this analysis used the largest data subset available in this study (n=388). This analysis includes data from D,H,S,T03,T05,SE datasets which incorporate LL, PL, CF, and σ' as input parameters (no SF). Figure 26 shows that these three CF ranges occupy relatively distinct areas of the $ø_r'$ prediction space. This 1 to 1 plot shows the predicted values of $ø_r'$ on the y axis and known values of $ø_r'$ on the x axis with a black line running along the 45° which signifies a perfect ratio between predicted and known values. These spatial distinctions support the idea that BPNN models could be successfully developed for the distinct CF ranges described in Table 5.

Predictions of drained secant residual friction angle, used the comprehensive BPNN, and were trained on the entire dataset, and each BPNN trained on CF subsets of this data are shown in Figures 27-29. Figure 27 shows the comprehensive BPNN predictions for compared to the $CF \leq 23$ data range, Figure 28 shows same as compared to the $23 < CF \leq 48$ data range, and Figure 29 shows the same compared to $CF > 48$ range. The terminal testing RMSE associated with these networks is used for evaluating performance and is shown in the top left of the 1 to 1 prediction plots shown on the first row of figures. A more qualitative analysis may be made by observing the RMSE plots in the second row of each figure.

In Figure 27, it is clear that the RMSE associated with the training of the comprehensive BPNN outperformed the BPNN that only considered data in the $CF \leq 23$ range. Not only did terminal RMSE remain lower for the comprehensive BPNN, the RMSE for both testing and training was substantially lower throughout the training process. Figure 28 shows a similar trend for data in the $23 < CF \leq 48$ range.

The BPNN trained on data in the CF>48 range (Figure 29) is more successful than the comprehensive BPNN. The terminal RMSE for the subgrouped BPNN was significantly lower than the comprehensive BPNN. Further, it is visually apparent that predictions created by this network are more accurate. The comprehensive BPNN's predictions for the CF>48 range, shown in black in the top left window, are clearly overestimated. These data are systematically biased above the 45° black line, showing a tendency to overpredict. The top-right window shows that the network trained only on the CF>48 range is more successful in mapping the relationship between input parameters and $ø_r$'.

Results from these analyses suggest that subdivision of data before developing a predictive BPNN could be useful in enhancing the predictive capabilities of BPNN models developed for this application. Despite this subdivision's inability to reduce predictive error for all CF data ranges, significant reduction in prediction error was developed for the CF>48 range. As was recognized at the beginning of this analysis, it is possible that there were not enough data for the BPNN to successfully map the complex relationships observed in the lower CF ranges. Further, data subgrouping may have been improved through more sophisticated data subdivision techniques. Insights provided by data clustering, similar to analysis capabilities of the SOM described in 2.1.3.

5.2 Analysis of BPNN Performance with and without SF

While most other studies do not include SF in their models, this study considered BPNN models that incorporated SF as an input parameter along with models that neglected that parameter. The choice to incorporate this parameter was motivated by a multivariate linear correlation analysis. This correlation analysis, shown in Figure 23 provided a correlation coefficient of 0.3118 between $ø_r$' and SF. While this is a weaker correlation coefficient as compared to various other input parameters (LL,PL,CF), it provides substantive evidence that there may be a relationship between $ø_r$' and SF.

In order to conduct a controlled analysis of a BPNN's predictive capacity with and without SF, identical datasets needed to be compared. Only datasets that included SF data were considered in this study, limiting this model to D,H,S, and T03 datasets. Due to concerns related to internal consistency, the D and T03 datasets were initially neglected from this analysis. First, a BPNN was trained to the H and S datasets with and without consideration of SF as an input parameter. A schematic of the two networks used in this comparison are provided in Figure 10.

The resultant optimally structured networks from both BPNN input parameter scenarios are shown in Figure 30. It is notable that the network including SF has an optimal sizing of 3 input nodes while the other is optimally sized at 4 input nodes. This is due the fact that the different input parameter scenarios provide different data loadings to the network. Since this compares the best-case predictive models for the two input parameter scenarios, the variation in these network sizings is not concerning. Comparison of the terminal testing RMSE values for the networks suggests that the BPNN may perform better without SF being considered.

This first analysis used only 120 data points, of which only 80 were used for training. Due to concern about the validity of an analysis that utilized such a small data subset, further analyses were employed to develop a more robust understanding of the predictive contribution of SF. Since the low testing RMSE values validated that the BPNNs shown in Figure 30 were training correctly, the network was retrained to the entire dataset with no data withheld for testing. Figure 31 shows the results of these network trainings for H and S datasets. In order to further increase the amount of training data the T03 dataset was added to H and S. Following similar positive trends in testing for this new dataset, the results of withholding no data for testing are shown in Figure 32.

The analyses described in Figure 31 & 32 both provide results contrary to those initially observed in Figure 30. The training RMSE would suggest that the optimal networks that include SF as an input parameter outperform those that do not. Despite this trend, it would be unwise to draw such a conclusion without further study. Error margins between these two networks are no larger than 0.0210 in normalized RMSE, and it could be argued that the difference in performance between these two networks is trivial. A more exhaustive study would require a significantly larger dataset to come to a concrete conclusion about the contribution of SF to the prediction of $ø_r$'.

5.3 BPNN Performance

The predictive performance of the BPNN models developed in this study was evaluated to determine their viability as an alternative to other models employed to predict $ø_r$'. The performance of the BPNN was evaluated in comparison to two alternative predictive models for $ø_r$'. RMSE was utilized as the quantitative metric used in these comparisons. First the BPNN was compared against multivariate linear regression models before being compared against equations described in Stark and Hussain (2013).

5.3.1 BPNN vs. Multivariate Linear Regression Models

A common metric of BPNN performance is the comparison of a network to a linear regression (REG) model (Ceryan, Okkan et al. 2013; Sezer 2013; Torabi, Shirazi et al. 2013). The multivariate linear regressions used in this study were developed using JMP Pro 10 software, though figures were created in MATLAB. As was discussed in 4.1.2, these linear REG models do not successfully develop a robust predictive model, and BPNN models historically outperform them. This is apparent in Figure 33 & 34 where optimal BPNNs are compared to linear regression models of best fit.

Where data tends to follow the 1 to 1 error lines in the BPNN model, the data in the linear REG model follows no such trend. The spread of these data show that the linear REG model fails to accurately predict $ø_r$', especially when compared to the performance of the BPNN. A quantitative measure of this is shown by the differential in RMSE between the two models. In both cases, the BPNN models outperform the linear REG models by a normalized RMSE of at least 0.1. From this analysis, it is clear that the BPNN provides a relatively successful model for prediction of $ø_r$', continuing the historical trend of BPNNs outperforming linear REG models.

5.3.2 BPNN vs. Stark & Hussain Equations (2013)

The BPNNs outperformance of linear REG models warranted comparison to an established predictive model for $ø_r$'. As discussed in 4.4, Stark and Hussain (2013) provides a series of equations that allow for the prediction of $ø_r$' given input data related to LL, PL, CF, and σ'. These equations were adapted and written into a MATLAB code to compare predictions from this method to BPNN models. Because SF is not a parameter required by this model, these equations were applied to all six datasets included in this study. Figure 35 shows a 1 to 1 comparison of prediction results to known values for all 6 datasets. Datasets are color coordinated and broken into CF range subgroups by different marker shapes. Alternatively, Figure 36 shows these predictions on the same $ø_r$' vs. LL plot used in Stark & Hussain (2013). Predictive equations are plotted as black lines with $ø_r$' predictions coded in color by dataset, and in shape by CF range. In comparison to the data distribution shown in Figure 25, it is clear that there is significantly more scatter for $ø_r$' predictions than for the dataset used in the development of these equations.

Figure 22 shows that Stark and Hussain's equations are only available for discrete σ' loadings that fall within a specific PL range for each CF range subset. This code interpolates between the discrete σ' loadings described in these equations to provide predictions to a larger range of soil data values. Despite this, some data which did not fall within the required PL ranges for their CF subset, were omitted from prediction. These omissions are indicated in the figure's title and legend, which show the number of data points which were used for prediction versus the total number available. While the predicted versus known axes allow for easy visual interpretation, the RMSE is shown in the figure title allowing for quantitative analysis of predictive performance.

The results of the $ø_r$' predictions created by the adapted Stark and Hussain equations were compared to BPNN predictions. The optimal BPNN predictions for all 6 datasets are compared to those produced using the Stark and Hussain equations in Figure 37. As is visually and quantitatively apparent, the BPNN outperforms these equations in accuracy of prediction. In comparing various data subsets by color, some datasets are more accurately predicted by Stark and Hussain than others. One such dataset is the Stark and Eid 1994 dataset shown in turquoise. A comparison for this data subset is shown in Figure 38 and an exhaustive list of figures for all datasets is found in Appendix C.

In comparison to all other datasets, the prediction accuracy for the SE dataset is significantly higher. This makes sense, given that the dataset was used to develop the Stark and Hussain 2013 equations. Despite this, the BPNN trained to the SE dataset did achieve a slightly lower RMSE. Though the difference in RMSE values may be trivial, their closeness shows that the BPNN provides a comparable model to these equations.

Observe that this higher prediction accuracy for the SE dataset is also seen for the BPNN model. In comparison to the Commercial and Dewoolkar datasets, shown in Figure 39 & 40, the RMSE for the BPNN trained to the SE dataset is significantly lower. Where the RMSE for the Stark and Eid BPNN is 0.0671, the RMSE for Commercial and Dewoolkar are 0.1625 and 0.1934; roughly a magnitude of 10 larger. This is suggestive of a dataset that is, perhaps intentionally, less dimensionally complex. The fact that, in comparison to others, the BPNN trains so well to the SE dataset suggests the mapping developed for these data is simpler than those needed to map to other datasets.

It is clear from the left plot on Figure 37 that the BPNN predictions are highly successful for $ø_r$' values less than 10°, and greater than 20°. The relatively high prediction accuracy of $ø_r$' in these regions can be explained by observing the predictive empirical models of Figure 36, which show that for these data ranges, the relationship between LL and $ø_r$' is quite linear. This linearity is indicative of a reduced mathematical complexity between $ø_r$' and all other variables, leading to a more successful mapping of the complex nonlinear relationship in the $ø_r$'<10° and $ø_r$'>20° data ranges. This irregularity in predictive accuracy further supports the subdivision of soil data by CF range. This subdivision could lend accuracy to the middle range values that experience a higher level of mathematical complexity in the 10°<$ø_r$'<20° range (Figure 36).

The predictions of $ø_r$' by the Stark and Hussain equations are significantly less accurate for datasets other than SE. This is especially apparent for the Dewoolkar dataset, where most equation-based predictions don't fall within the 30% error line. This incongruity in prediction accuracy suggests that the Stark and Hussain equations may be capable of providing accurate equations for soil data that is dimensionally similar to the SE dataset, but the model was produced using data that is not representative of a larger soil population.

The superiority of BPNN predictive accuracy is shown both visually by the comparison using 1 to 1 plots and quantitatively using RMSE. Further, the BPNN model allows for consideration of input parameters along a continuum. While adapted to be applied more continuously by interpolating between σ', the Stark and Hussain equations were originally only applied to discrete loadings across limited CF and PL ranges that provide significant data gaps.

While RMSE provided a basis to quantitatively compare the Stark and Hussain equations to the BPNN models developed in this study, this comparison is not comprehensive enough to definitively state that one predictive model is better than the other. A more conclusive study would require a significantly large, internally consistent soil dataset from multiple sites, allowing for a more meticulous evaluation of performance for both BPNN and Stark and Hussain (2013) models.

A major shortcoming of the BPNN is that it provides a ‘black box’ in that there is no equation that can be provided to the user. Instead, the user must provide input values to the BPNN and accept the output response, trusting the weight matrices that connect a digital structure to deliver a once again tangible output. This mathematical intangibility is a quality that could be undesirable in that it requires a suspension of disbelief and trust in a nontraditional model. Conversely, the Stark and Hussain equations are given in the form of tangible mathematical equations. In this way, a user may actively perform calculations by his or herself in a way that lends trust through its traditional mathematical structure. It is clear that there are distinct advantages and disadvantages to each modeling technique, though the RMSE error comparisons across various datasets suggest that the BPNN modeling approach provides a more robust and accurate predictive model than Stark and Hussain’s correlation-based equations.

6. Conclusions and Future Work

6.1 Conclusions

With six different soil datasets that include input soil parameters LL, PL, CF, σ’, and sometimes SF, various BPNN models were developed to predict $ø_r$’. The subdivision of soil data by CF ranges specified in Stark and Hussain (2013) equations was considered in an attempt to lend predictive accuracy to the model by grouping like data. While the clustering observed in Figure 26 makes this division attractive, analyses show that the reduction in data size caused by dividing these data does not allow for the improvement of predictive ability in this case. Further, prediction accuracy varies for different ranges of $ø_r$’. $ø_r$’<10° and $ø_r$’>20° have higher prediction accuracies than the middle range, suggesting that data subdivision may supply increased prediction accuracy for the middle range. Thus, the subdivision of data by CF range is suggested for a more robust dataset with substantial amounts of data to map a BPNN to each range.

Where other studies neglect SF as a predictive parameter for $ø_r$’, this study considered the predictive capacity lent to a BPNN model by incorporating SF. Comparisons between BPNN models that incorporate SF versus those that don’t are inconclusive. While one RMSE comparison suggests that $ø_r$’ is better predicted by a network that doesn’t include SF, two subsequent comparisons suggest the opposite. In all three cases, the RMSE between the two models is not statistically different.

The analyses performed on the BPNN models developed in this study suggest that the BPNN outperformed the other models. Where Stark and Hussain’s equations were inconsistent in their ability to reliably predict $ø_r$’ for datasets other than SE, the BPNN consistently exhibited relatively low RMSE across all datasets. This suggests that a BPNN model provides a robust mapping the characteristic nonlinear relationship between input soil parameters and $ø_r$’. In comparison, the Stark and Hussain equations fail to accurately predict $ø_r$’ for datasets other than the SE dataset (the dataset used in the development of these equations). This can only be said for the data provided, and there are concerns related to internal consistency of datasets and the combination of these data. It is posited that provided with a large and internally consistent dataset subdivided by CF ranges, BPNN models would be able to improve on the predictive performance observed in the models developed in this study and serve as a robust predictive tool for the indirect determination of $ø_r$'.

6.2 Future Work

A more complicated analysis of data clusters, beyond the division of soil data by CF ranges, may provide even higher predictive accuracy to a predictive BPNN model. Unsupervised algorithms such as the SOM discussed in the literature review are a powerful means by which to identify these multidimensional subgroups of data. Unsupervised algorithms such as this one are beyond the scope of this honors thesis research, though future work should include clustering techniques such as a SOM to provide robust data subdivisions to improve the predictive capacities of BPNN models.

Not included in this study is an analysis of the predictive capacity of PL in a BPNN model. Given that LL and PL are both functions of clay mineralogical composition, and thus related, it is possible that a network that only considers one Atterberg limit may experience increased predictive accuracy due to minimized mathematical complexity within the network. Future studies involving the parameterization of a BPNN for the prediction of $ø_r$' should involve a study of predictive performance of a BPNN with and without PL, similar to what was performed in this study for SF.

Alessandri, A., C. Cervellera, et al. (2013). "Predictive Control of Container Flows in Maritime Intermodal Terminals." Ieee Transactions on Control Systems Technology **21**(4): 1423-1431.

Azami, H., M. R. Mosavi, et al. (2013). "Classification of GPS Satellites Using Improved Back Propagation Training Algorithms." Wireless Personal Communications **71**(2): 789-803.

Basma, A. A., S. A. Barakat, et al. (2003). "Modeling time dependent swell of clays using sequential artificial neural networks." Environmental & Engineering Geoscience **9**(3): 279-288.

Baykan, N. A. and N. Yilmaz (2010). "Mineral identification using color spaces and artificial neural networks." Computers & Geosciences **36**(1): 91-97.

Bevilacqua, V. (2013). "Three-dimensional virtual colonoscopy for automatic polyps detection by artificial neural network approach: New tests on an enlarged cohort of polyps." Neurocomputing **116**: 62-75.

Ceryan, N., U. Okkan, et al. (2013). "Prediction of unconfined compressive strength of carbonate rocks using artificial neural networks." Environmental Earth Sciences **68**(3): 807-819.

Cheng, K. H. (2013). "Self-structuring fuzzy-neural backstepping control with a B-spline-based compensator." Neurocomputing **117**: 138-149.

Collotta, T., R. Cantoni, et al. (1989). "A CORRELATION BETWEEN RESIDUAL FRICTION ANGLE, GRADATION AND THE INDEX PROPERTIES OF COHESIVE SOILS." Geotechnique **39**(2): 343-346.

Dewoolkar, M. M. and R. J. Huzjak (2005). "Drained residual shear strength of some claystones from front range, Colorado." Journal of Geotechnical and Geoenvironmental Engineering **131**(12): 1543-1551.

Doris, J. J., D. M. Rizzo, et al. (2008). "Forecasting vertical ground surface movement from shrinking/swelling soils with artificial neural networks." International Journal for Numerical and Analytical Methods in Geomechanics **32**(10): 1229-1245.

El-Baz, A. H. and A. S. Tolba (2013). "An efficient algorithm for 3D hand gesture recognition using combined neural classifiers." Neural Computing & Applications **22**(7-8): 1477-1484.

Fan, F. H., Q. Ma, et al. (2013). "Prediction of texture characteristics from extrusion food surface images using a computer vision system and artificial neural networks." Journal of Food Engineering **118**(4): 426-433.

Flom, P. L. C., David L. (2007). "Stopping Stepwise: Why stepwise and similar selection methods are bad, and what you should use." NESUG.

Ghoreyshi, M., A. Jirasek, et al. (2013). "Computational approximation of nonlinear unsteady aerodynamics using an aerodynamic model hierarchy." Aerospace Science and Technology **28**(1): 133-144.

Hoskins, J. C. and D. M. Himmelblau (1988). "Artificial neural network models of knowledge representation in chemical engineering." Computers & Chemical Engineering **12**(9–10): 881-890.

Irie, B. and S. Miyake (1988). Capabilities of three-layered perceptrons. Neural Networks, 1988., IEEE International Conference on.

Johari, A., A. A. Javadi, et al. (2011). "Modelling the mechanical behaviour of unsaturated soils using a genetic algorithm-based neural network." Computers and Geotechnics **38**(1): 2-13.

Kim, B. S. and A. J. Calise (1997). "Nonlinear Flight Control Using Neural Networks." Journal of Guidance, Control, and Dynamics **20**(1): 26-33.

Larochelle, H., Y. Bengio, et al. (2009). "Exploring Strategies for Training Deep Neural Networks." Journal of Machine Learning Research **10**: 1-40.

Liu, H., H. Q. Tian, et al. (2013). "An experimental investigation of two Wavelet-MLP hybrid frameworks for wind speed prediction using GA and PSO optimization." International Journal of Electrical Power & Energy Systems **52**: 161-173.

Lupini, J. F., A. E. Skinner, et al. (1981) The drained residual strength of cohesive soils. Geotechnique **31**, 181-213

Martinez, H. P., Y. Bengio, et al. (2013). "Learning Deep Physiological Models of Affect." Ieee Computational Intelligence Magazine **8**(2): 20-33.

Mesri, G. and M. Shahien (2003). "Residual Shear Strength Mobilized in First-Time Slope Failures." Journal of Geotechnical and Geoenvironmental Engineering **129**(1): 12-31.

Mishra, S. V. N., R. Dwivedi, et al. (2013). "Gases/Odors Identification With Artificial Immune Recognition System Using Thick Film Gas Sensor Array Responses." Ieee Sensors Journal **13**(8): 3039-3045.

Moore, R. (1991) The chemical and mineralogical controls upon the residual strength of pure and natural clays. Geotechnique **41**, 35-47

Mosleh, M. (2013). "Fuzzy neural network for solving a system of fuzzy differential equations." Applied Soft Computing **13**(8): 3597-3607.

Nelson, S. (2013). "Slope Stability, Triggering Events, Mass Movement Hazards."

Neyamadpour, A., S. Taib, et al. (2009). "Using artificial neural networks to invert 2D DC resistivity imaging data for high resistivity contrast regions: A MATLAB application." Computers & Geosciences **35**(11): 2268-2274.

Park, D. C., M. A. El-Sharkawi, et al. (1991). "Electric load forecasting using an artificial neural network." Power Systems, IEEE Transactions on **6**(2): 442-449.

Ramakrishnan, D., T. N. Singh, et al. (2013). "Soft computing and GIS for landslide susceptibility assessment in Tawaghat area, Kumaon Himalaya, India." Natural Hazards **65**(1): 315-330.

Reed, R. D. and R. J. Marks (1998). Neural Smithing: Supervised Learning in Feedforward Artificial Neural Networks, MIT Press.

Sezer, A. (2013). "Simple models for the estimation of shearing resistance angle of uniform sands." Neural Computing & Applications **22**(1): 111-123.

Skempton, A. W. (1985). "RESIDUAL STRENGTH OF CLAYS IN LANDSLIDES, FOLDED STRATA AND THE LABORATORY." Geotechnique **35**(1): 3-18.

Stark, T. D., H. Choi, et al. (2005). "Drained shear strength parameters for analysis of landslides." Journal of Geotechnical and Geoenvironmental Engineering **131**(5): 575-588.

Stark, T. D. and H. T. Eid (1994). "DRAINED RESIDUAL STRENGTH OF COHESIVE SOILS." Journal of Geotechnical Engineering-Asce **120**(5): 856-871.

Stark, T. D. and M. Hussain (2013). "Empirical Correlations: Drained Shear Strength for Slope Stability Analyses." Journal of Geotechnical and Geoenvironmental Engineering **139**(6): 853-862.

Tamilselvan, P. and P. F. Wang (2013). "Failure diagnosis using deep belief learning based health state classification." Reliability Engineering & System Safety **115**: 124-135.

Tiwari, B., T. L. Brandon, et al. (2005). "Comparison of residual shear strengths from back analysis and ring shear tests on undisturbed and remolded specimens." Journal of Geotechnical and Geoenvironmental Engineering **131**(9): 1071-1079.

Tiwari, B. and H. Marui (2005). "A new method for the correlation of residual shear strength of the soil with mineralogical composition." Journal of Geotechnical and Geoenvironmental Engineering **131**(9): 1139-1150.

Torabi, S. R., H. Shirazi, et al. (2013). "Study of the influence of geotechnical parameters on the TBM performance in Tehran-Shomal highway project using ANN and SPSS." Arabian Journal of Geosciences **6**(4): 1215-1227.

Tu, J. V. (1996). "Advantages and disadvantages of using artificial neural networks versus logistic regression for predicting medical outcomes." Journal of Clinical Epidemiology **49**(11): 1225-1231.

Wang, M. L., X. D. Yan, et al. (2013). "Spatiotemporal prediction for nonlinear parabolic distributed parameter system using an artificial neural network trained by group search optimization." Neurocomputing **113**: 234-240.

Wang, Y. and D. Wang (2013). "Towards Scaling Up Classification-Based Speech Separation." Ieee Transactions on Audio Speech and Language Processing **21**(7): 1381-1390.

Wesley, L. D. (2004). "Residual strength of clays and correlation using Atterberg limits." Geotechnique **54**(7): 503-504.

Willis, M. J., C. Di Massimo, et al. (1991). "Artificial neural networks in process engineering." Control Theory and Applications, IEE Proceedings D **138**(3): 256-266.

Yang, W. and X. Xia (2013). "Prediction of mining subsidence under thin bedrocks and thick unconsolidated layers based on field measurement and artificial neural networks." Computers & Geosciences **52**: 199-203.

Zhu, X. "Semi-Supervised Learning." University of Wisconsin-Madison: 1-10.

Using Backpropagation Neural Networks for the Prediction of Residual Shear Strength of Cohesive Soils

Luke Detwiler

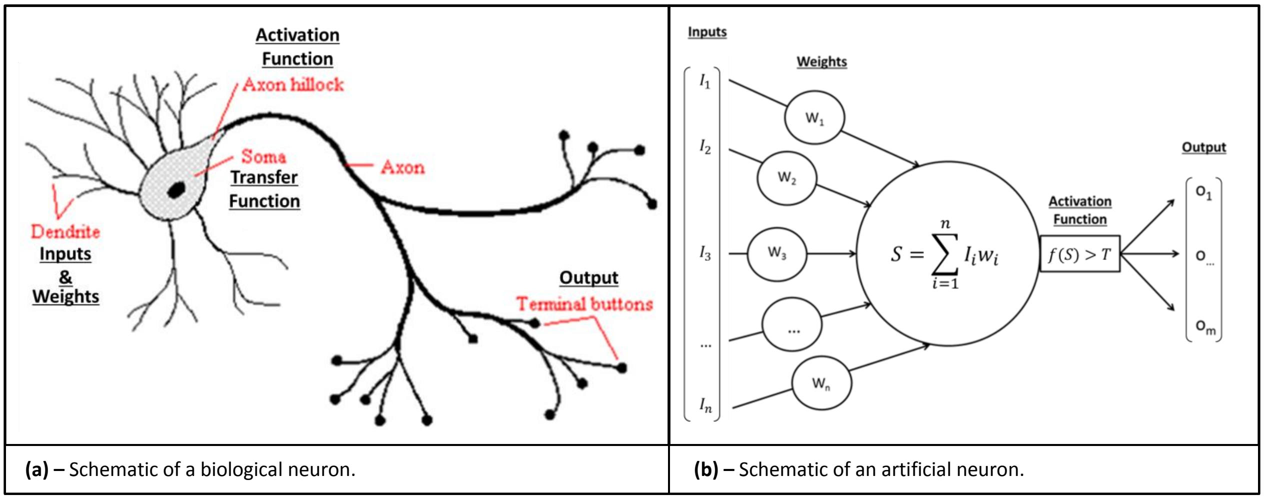
Figures, Tables, & Equations

Figures

Tables

Equations

Figures



(a) – Schematic of a biological neuron.

(b) – Schematic of an artificial neuron.

Figure 1: Schematics of (a) biological vs. (b) artificial neurons.

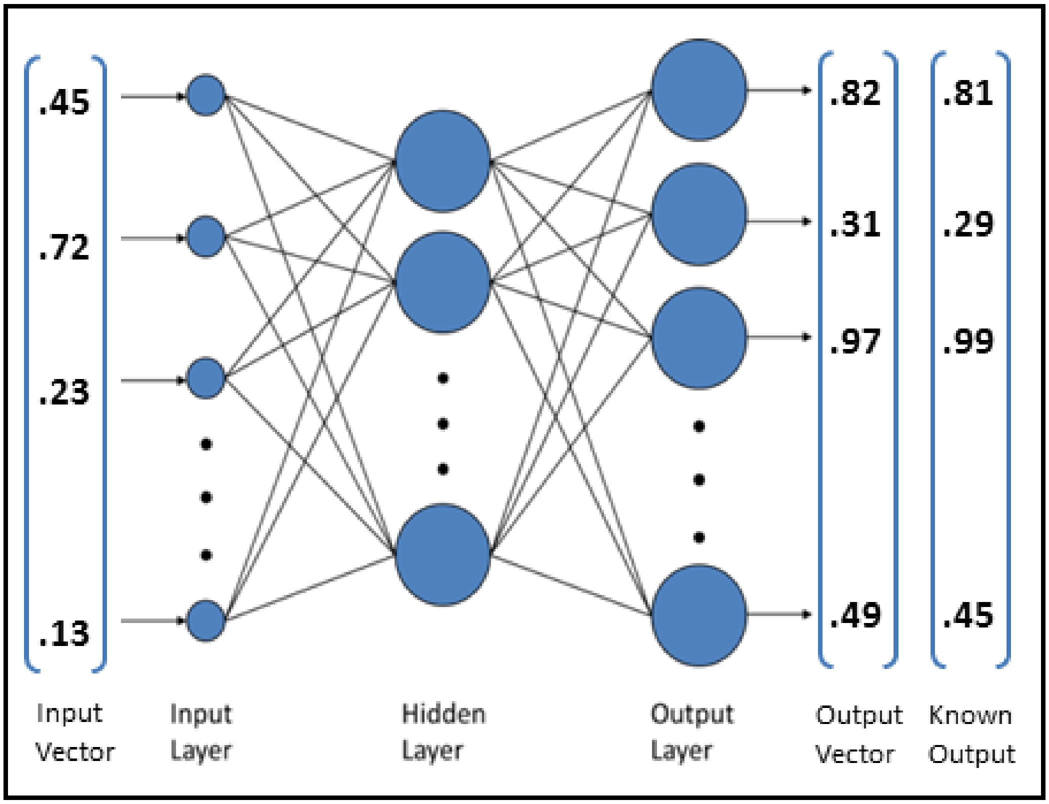


Figure 2: Generalized schematic of a supervised ANN structure showing sample input and output vectors.

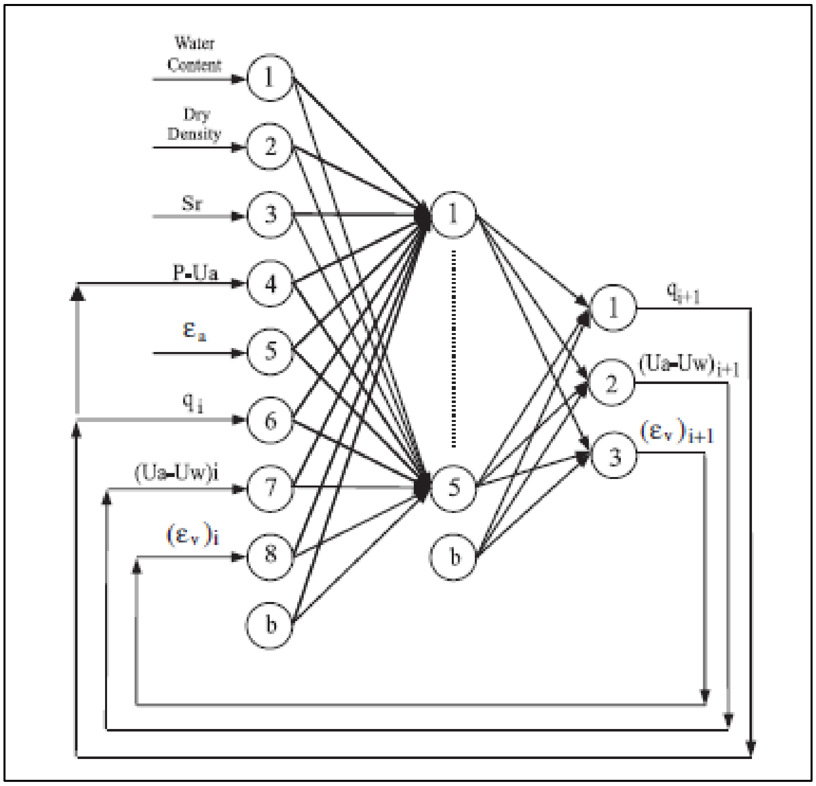


Figure 3: A RANN architecture showing outputs as inputs for recurrent calculation (Johari, Javadi et al. 2011).

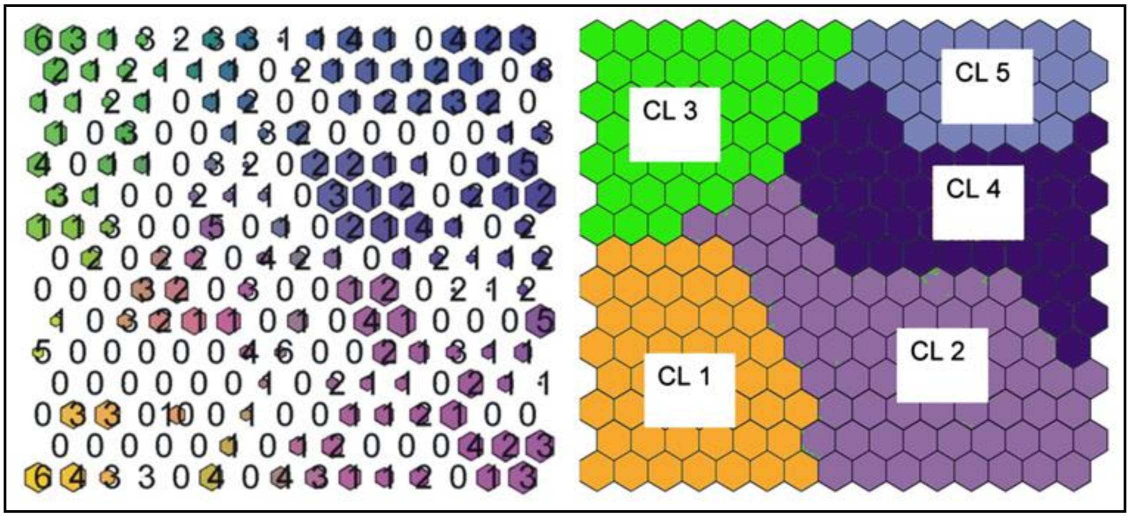


Figure 4: Results of a SOM's clustering results for marine sediment data from the Zakynthos canyon area in Greece (Ferentinou, Hasiotis et al. 2012).

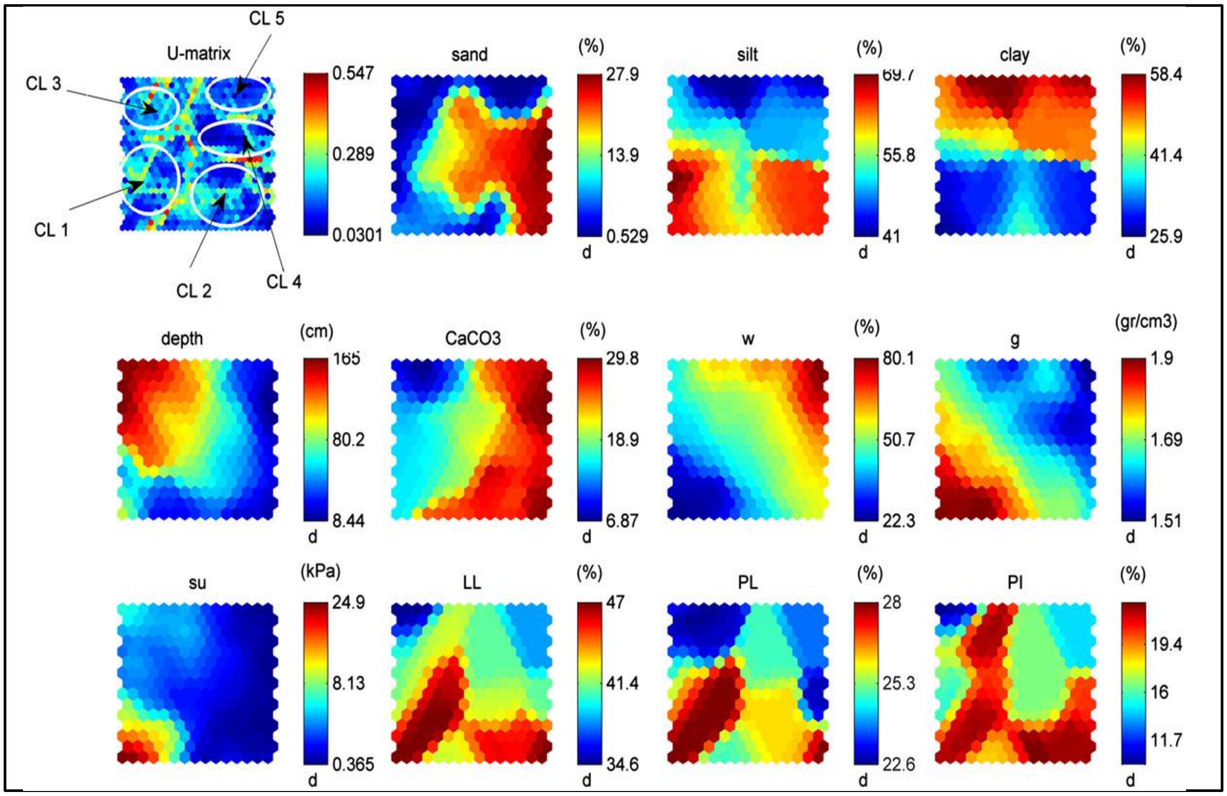


Figure 5: A SOM's cluster distribution by input parameters (Ferentinou, Hasiotis et al. 2012).

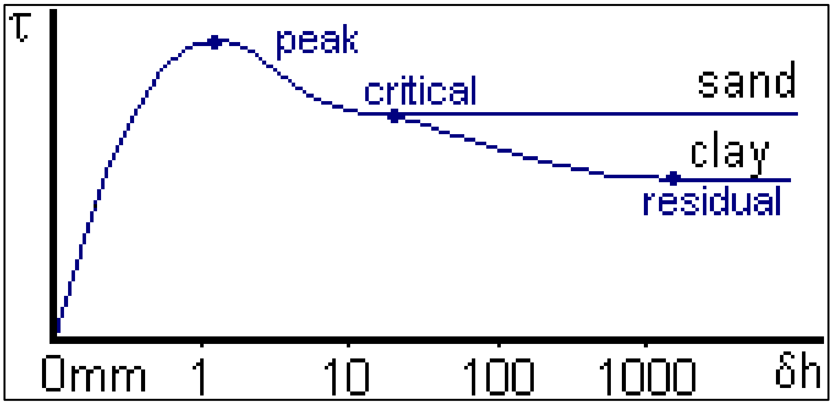


Figure 6: Residual condition of clayey soil (Nelson 2013).

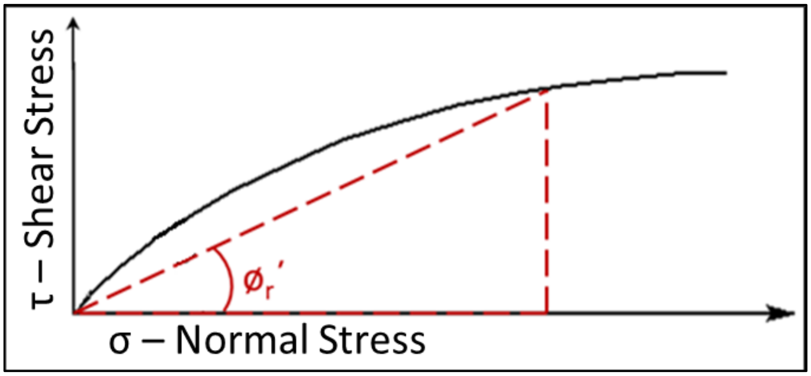


Figure 7: Secant residual friction angle on a soil failure envelope (Nelson 2013).

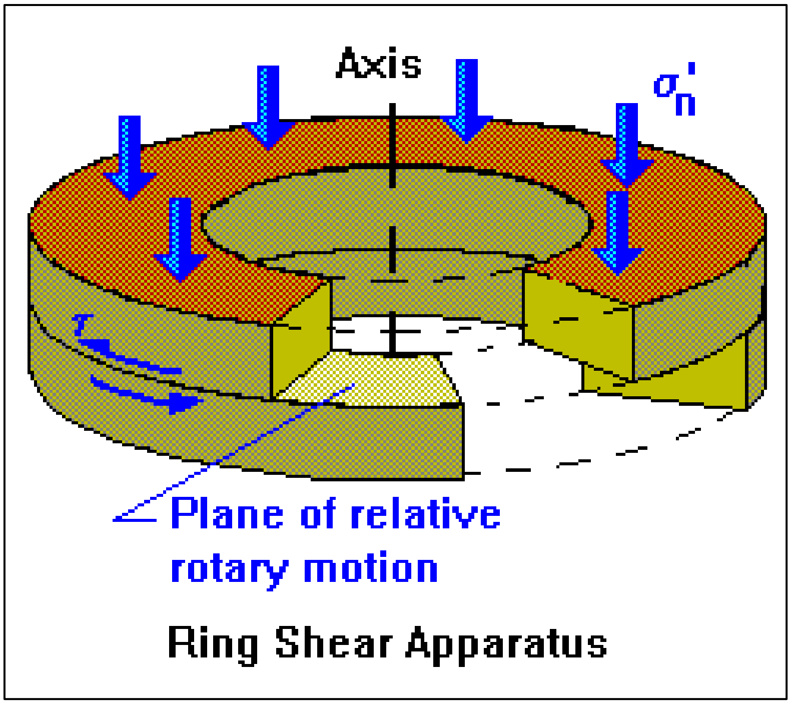


Figure 8: Generalized schematic of the torsional ring shear stress procedure (England 1992).

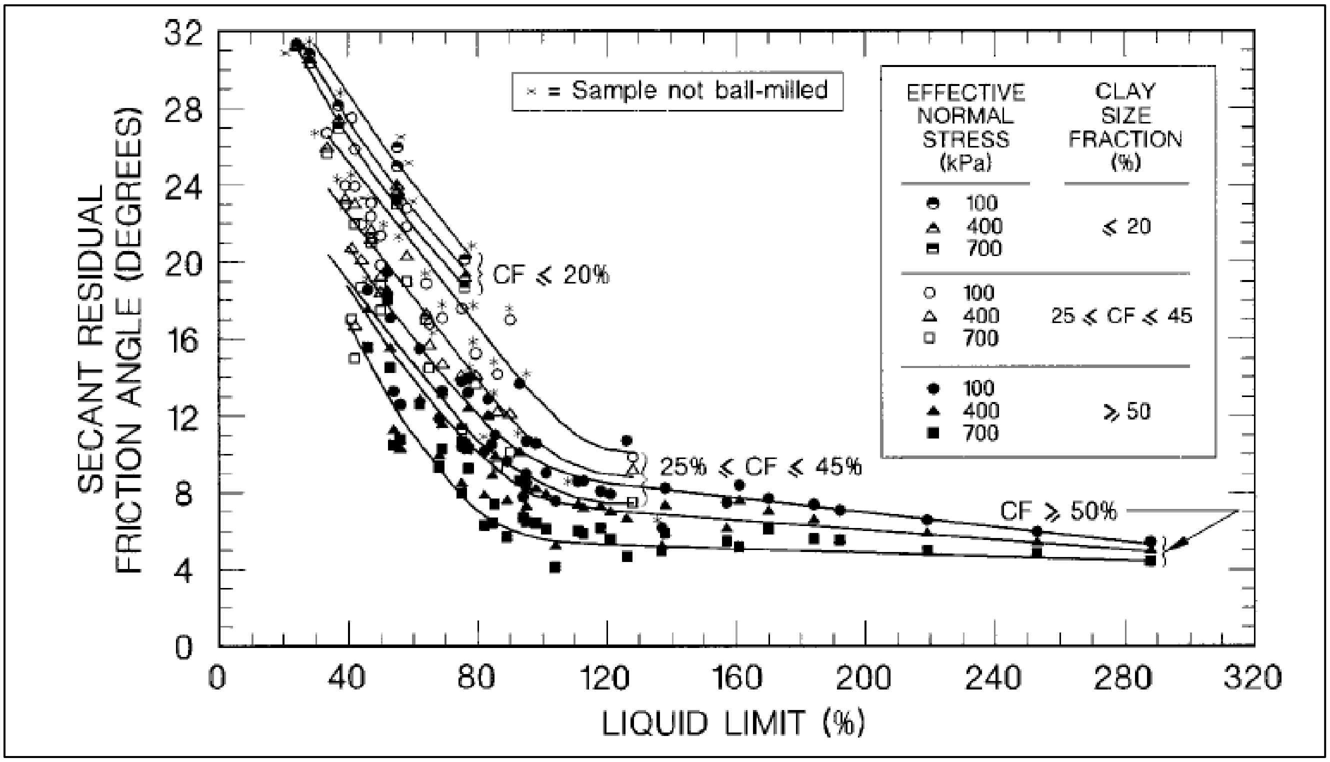


Figure 9: The complex nonlinear relationship between secant residual friction angle and various common soil parameters (Stark, Choi et al. 2005).

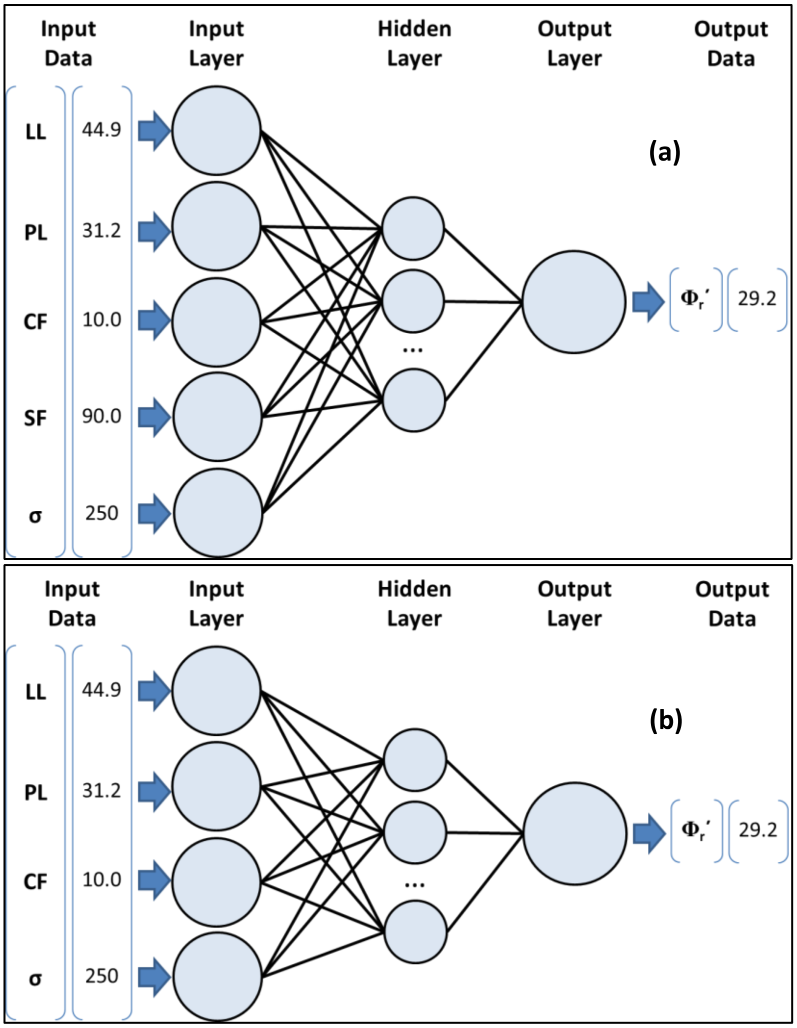


Figure 10: BPNN schematics for training with (a) and without SF (b).

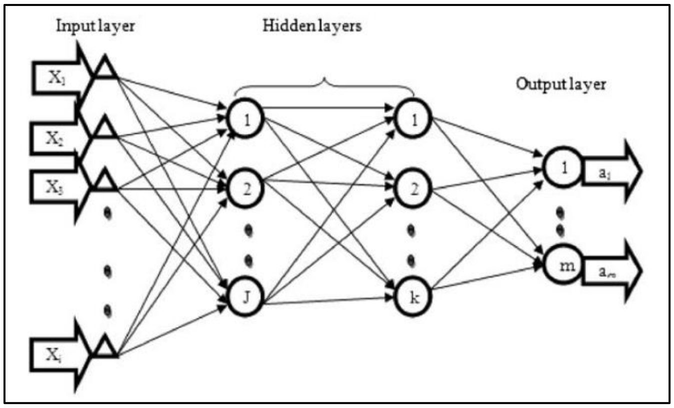


Figure 11: Example of a BPNN with multiple hidden layers (Neyamadpour, Taib et al. 2009).

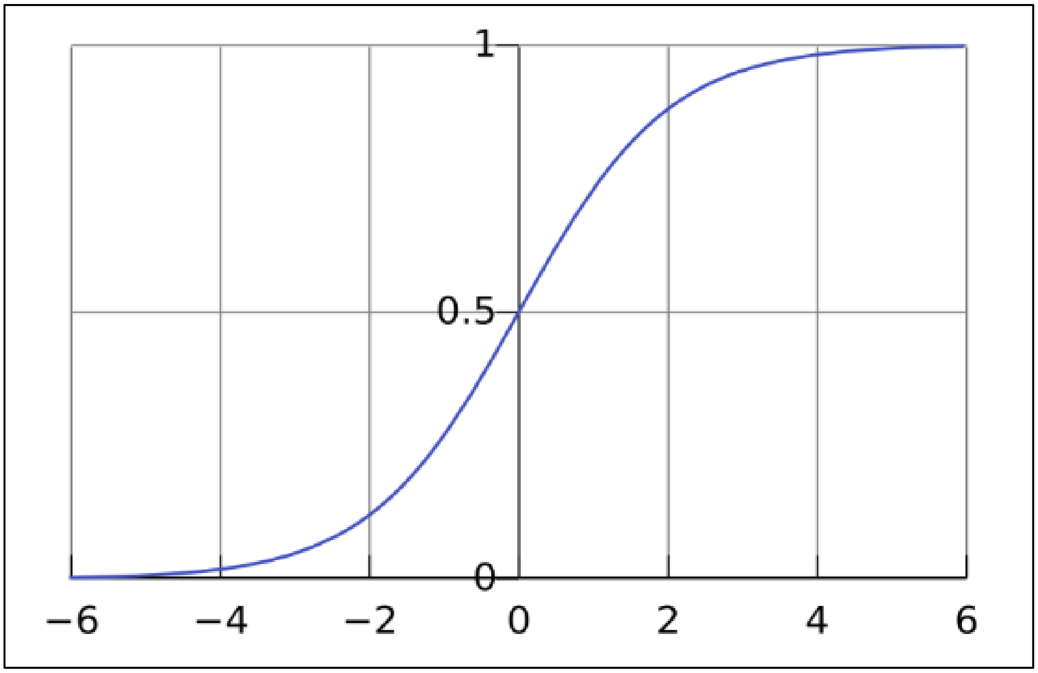


Figure 12: Logistic squashing function.

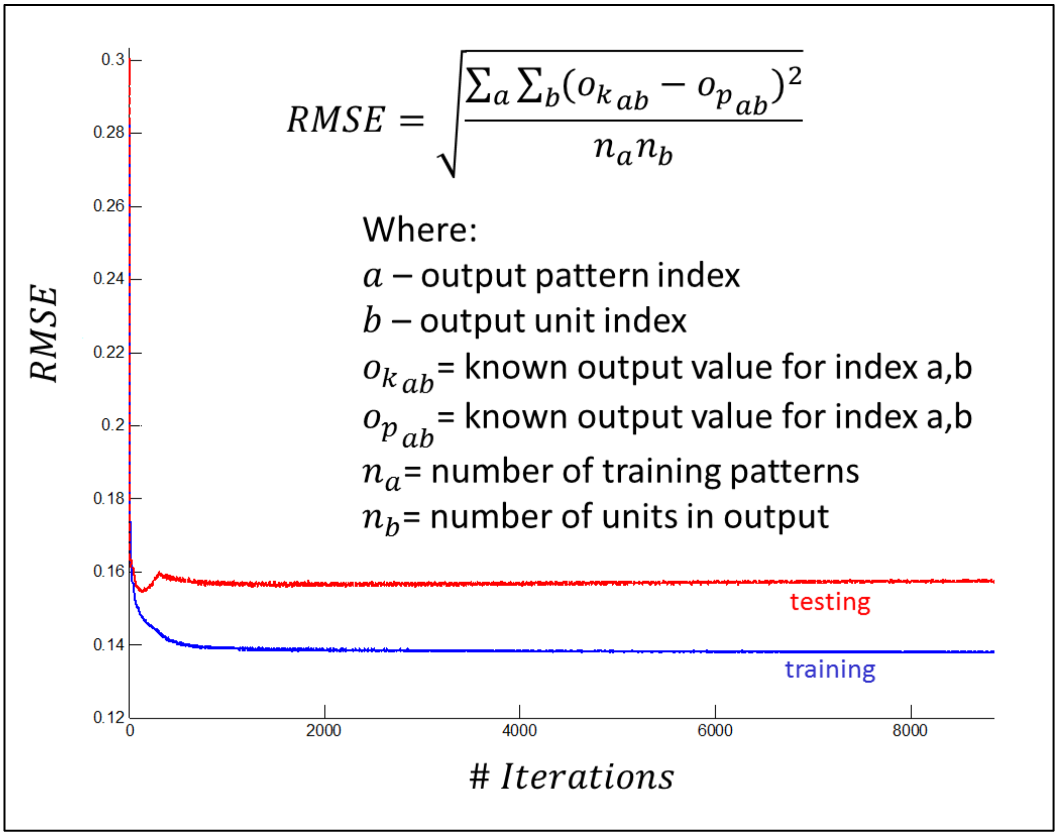


Figure 13: Iterative RMSE calculation for BPNN output.

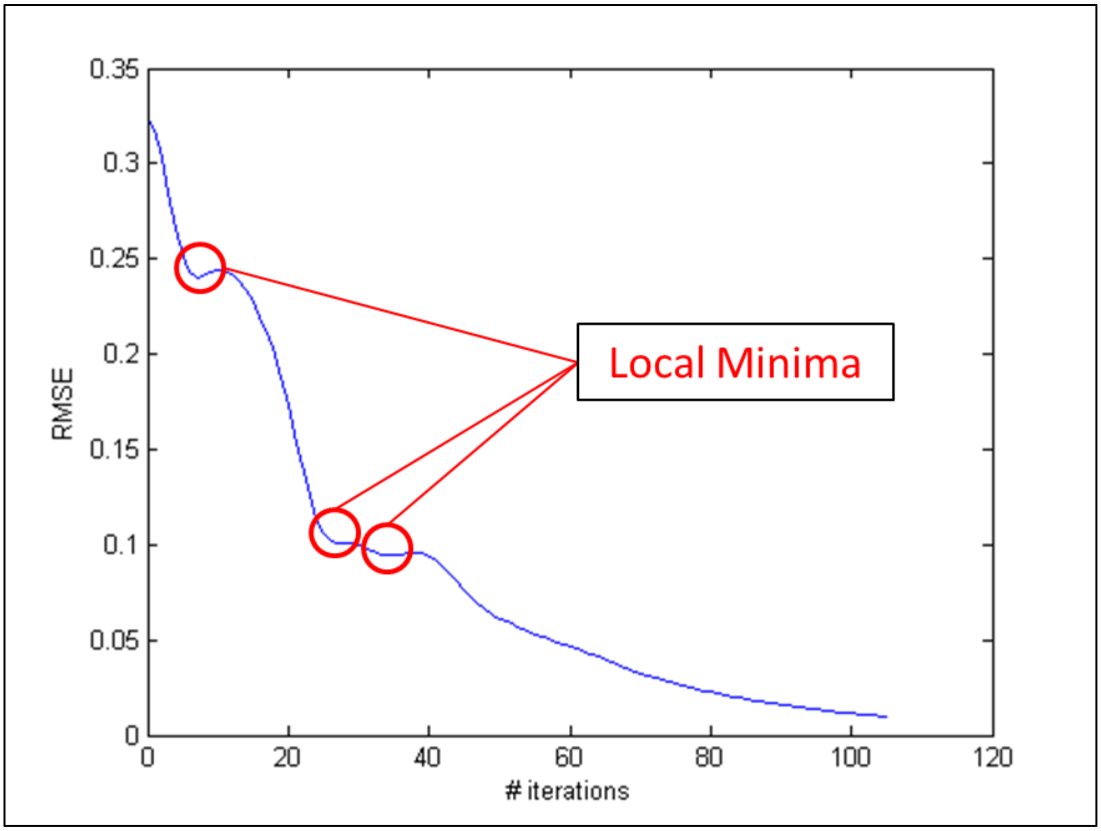


Figure 14: Avoiding local error minima in BPNN training via momentum.

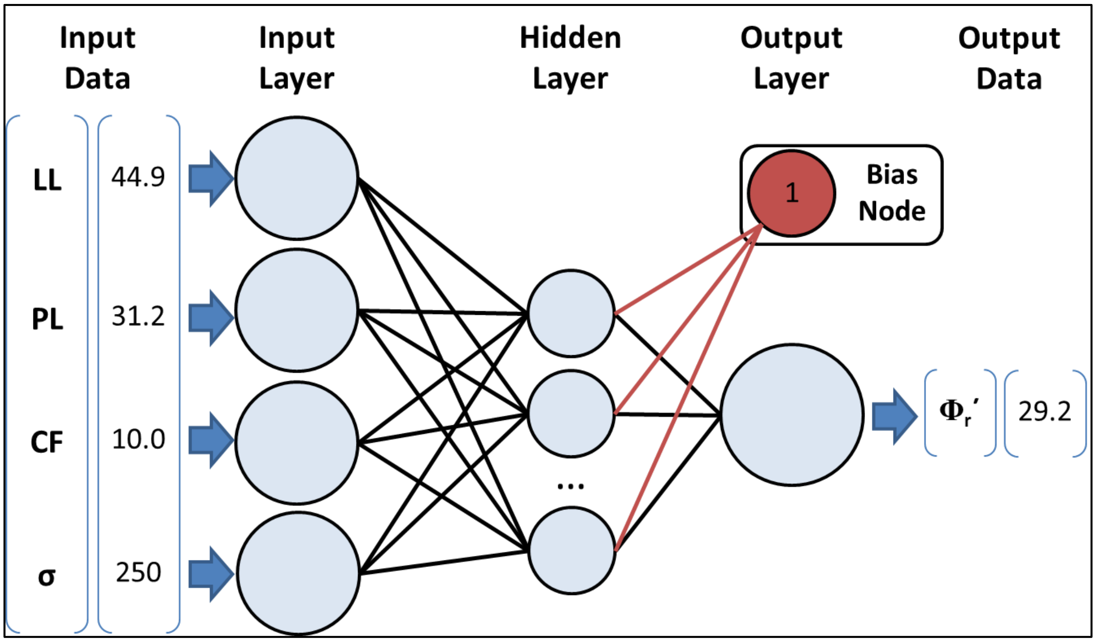


Figure 15: Biased BPNN.

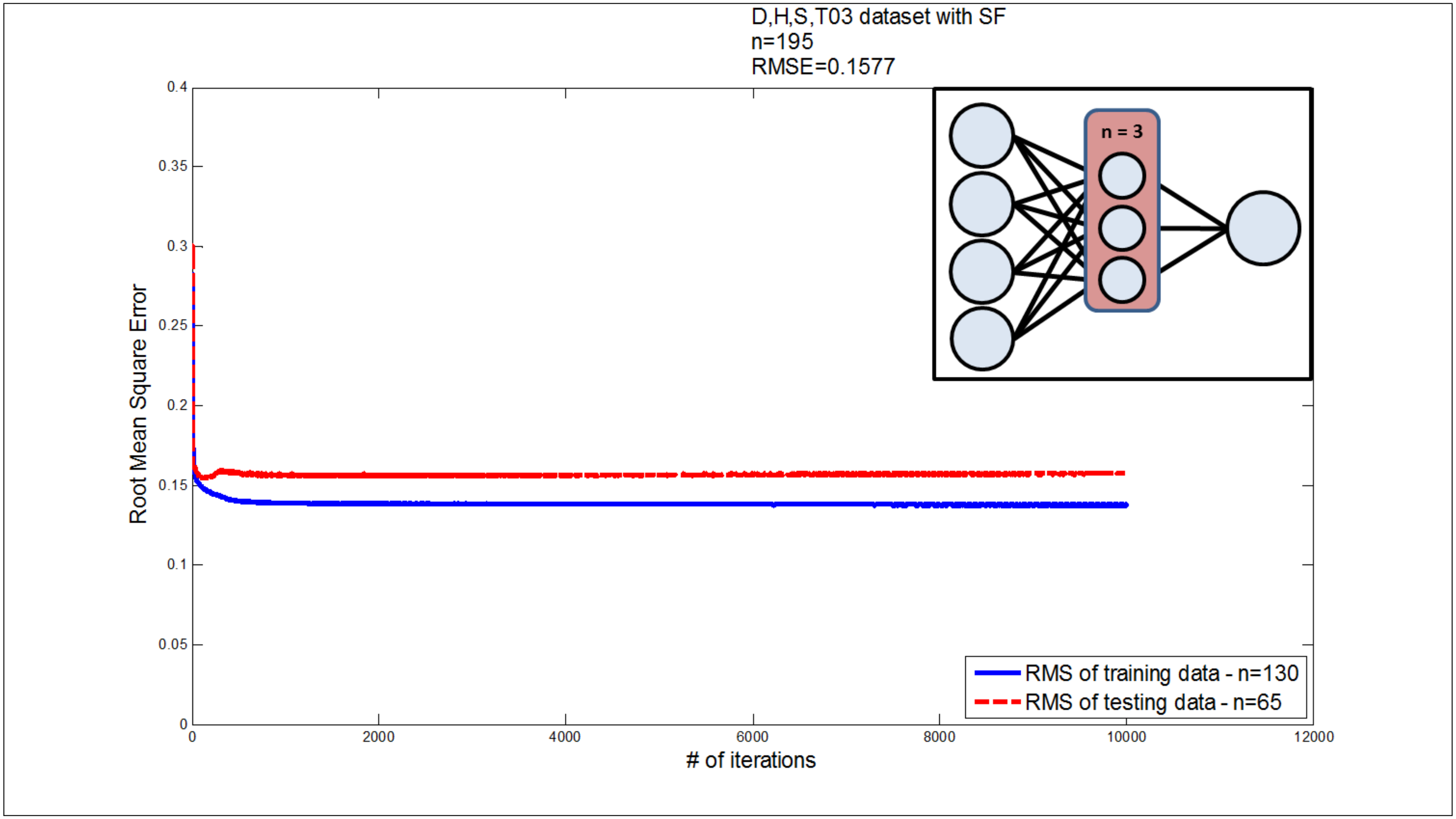


Figure 16: 3 node BPNN model tested for optimization.

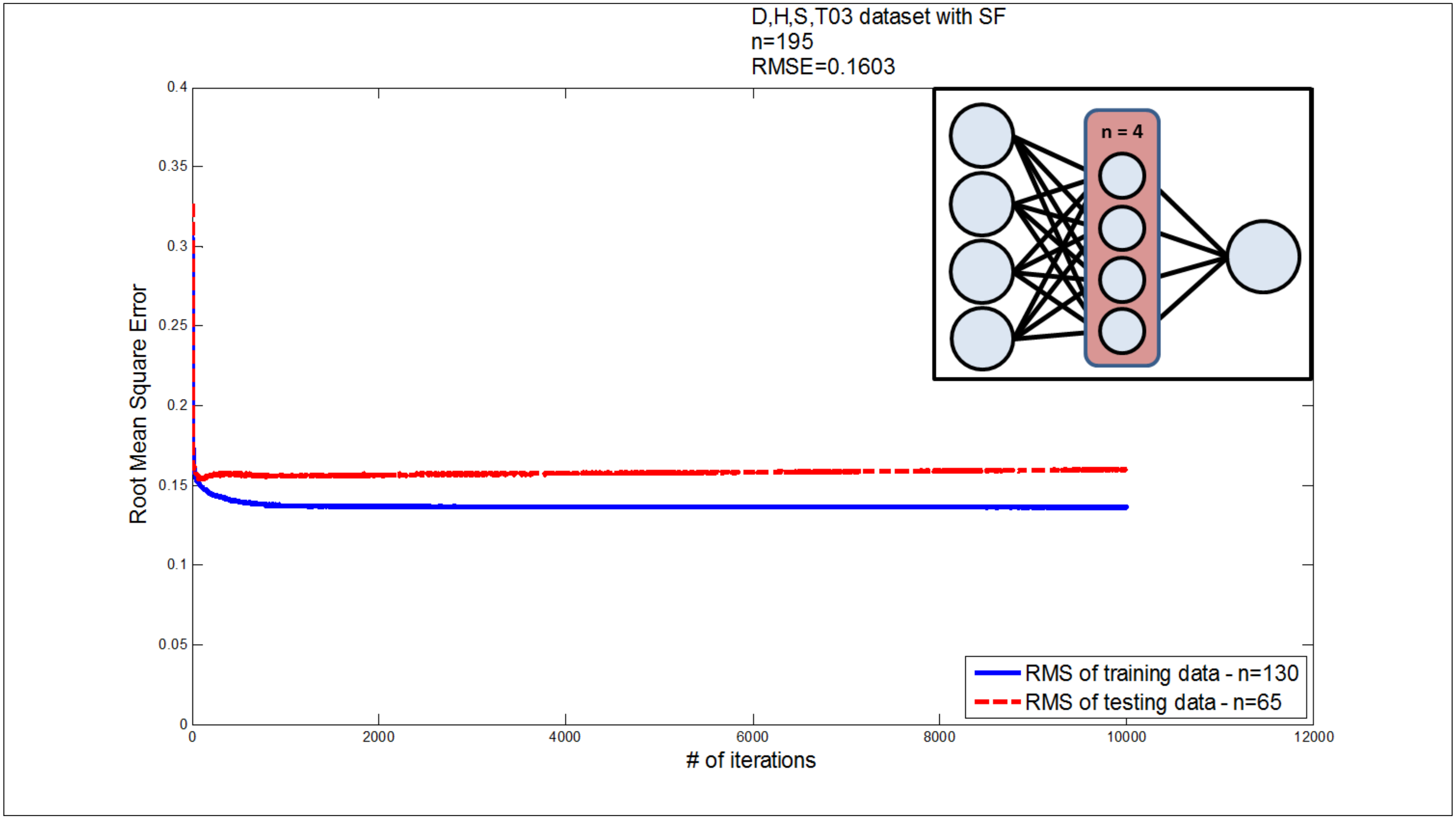


Figure 17: 4 node BPNN model tested for optimization.

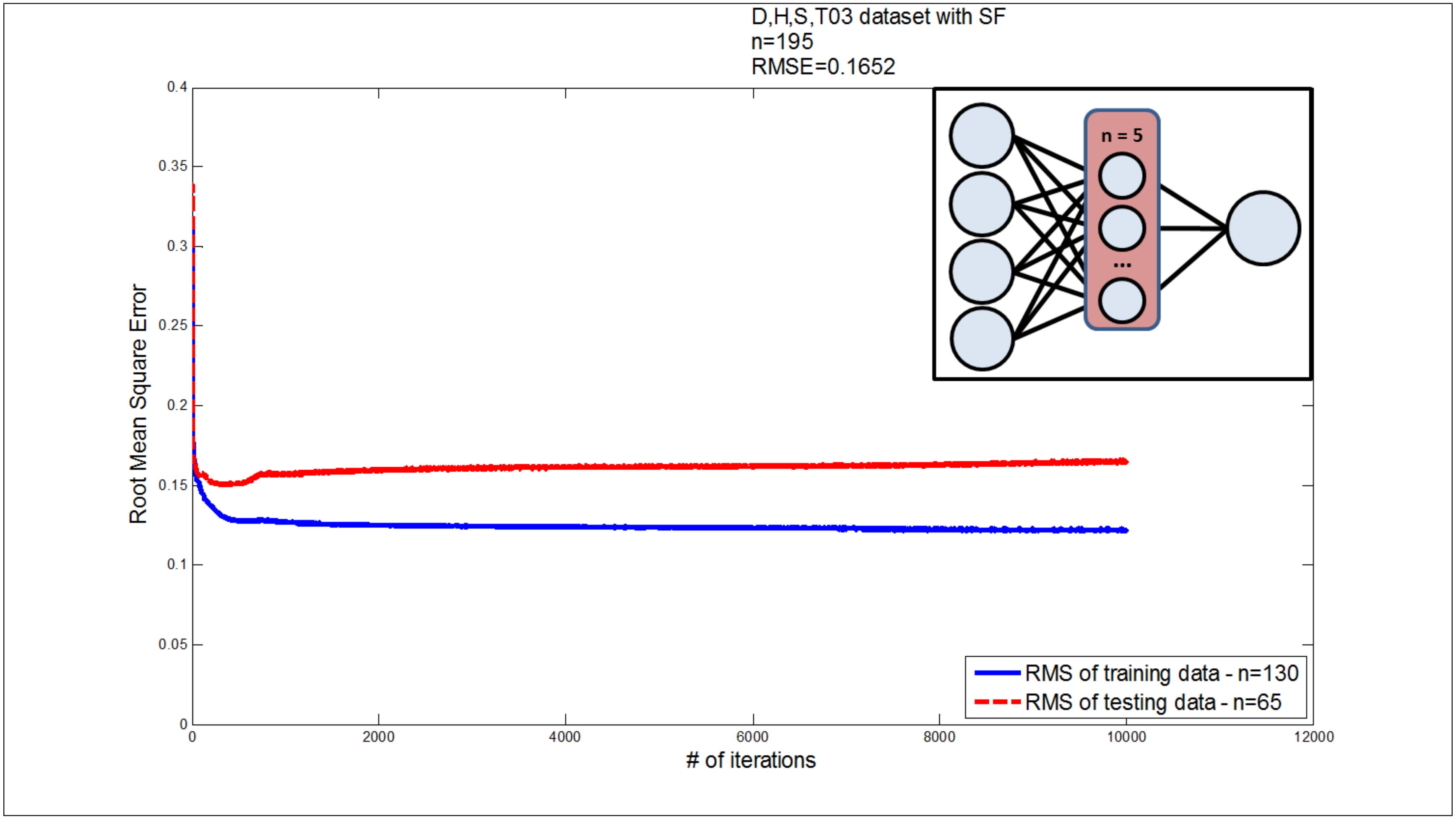


Figure 18: 5 node BPNN model tested for optimization.

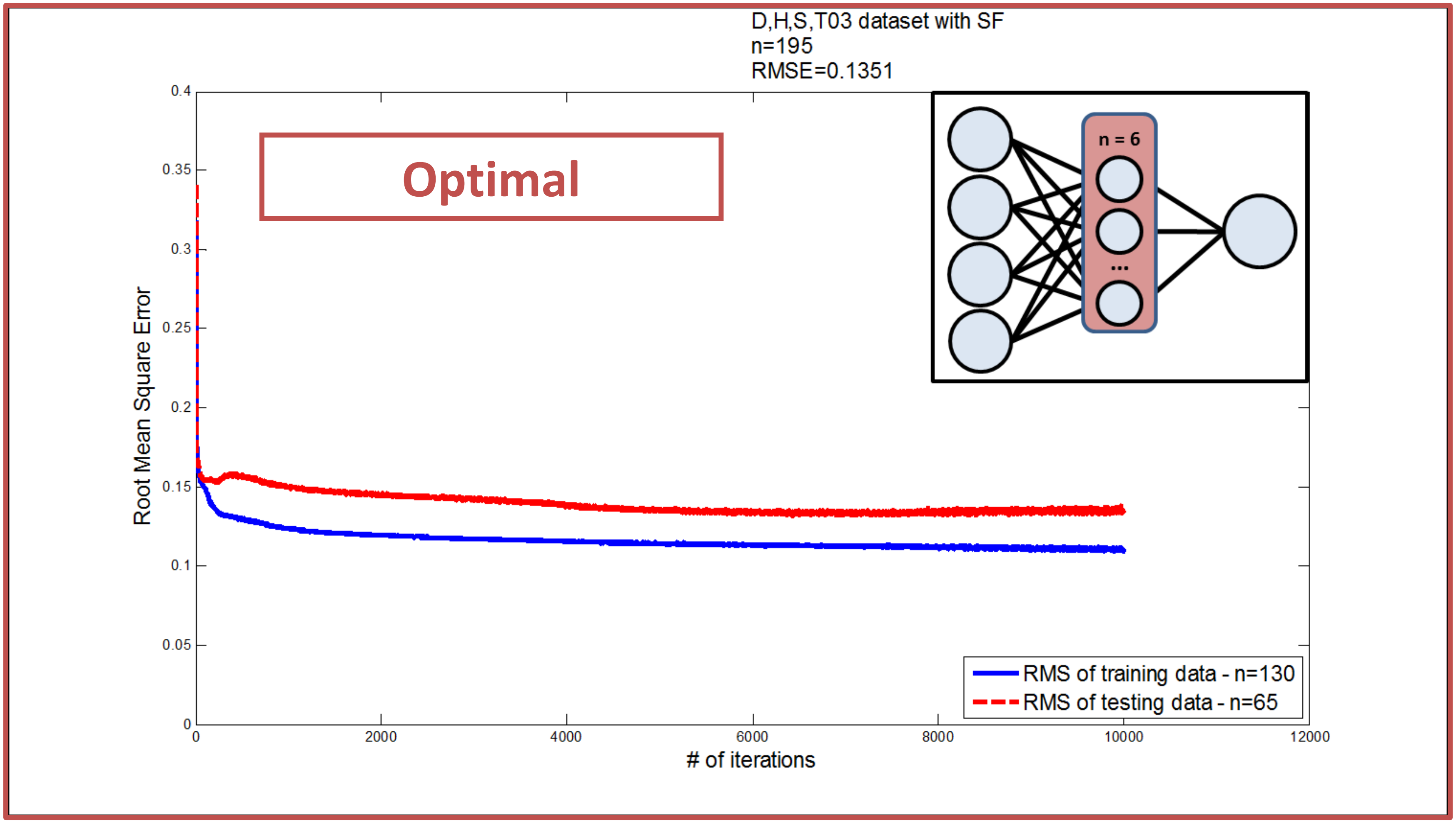


Figure 19: 6 node BPNN model tested for optimization.

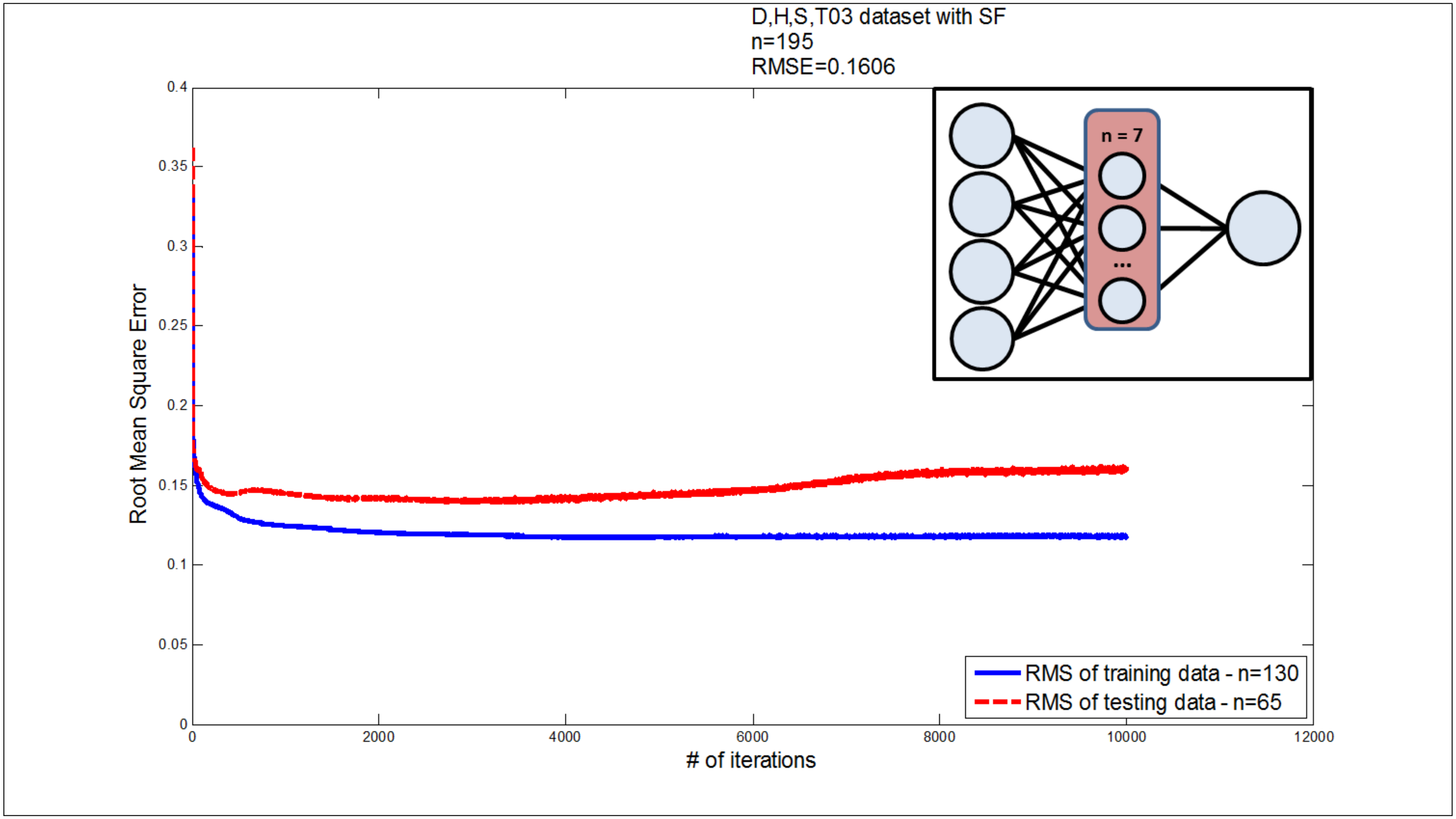


Figure 20: 7 node BPNN model tested for optimization.

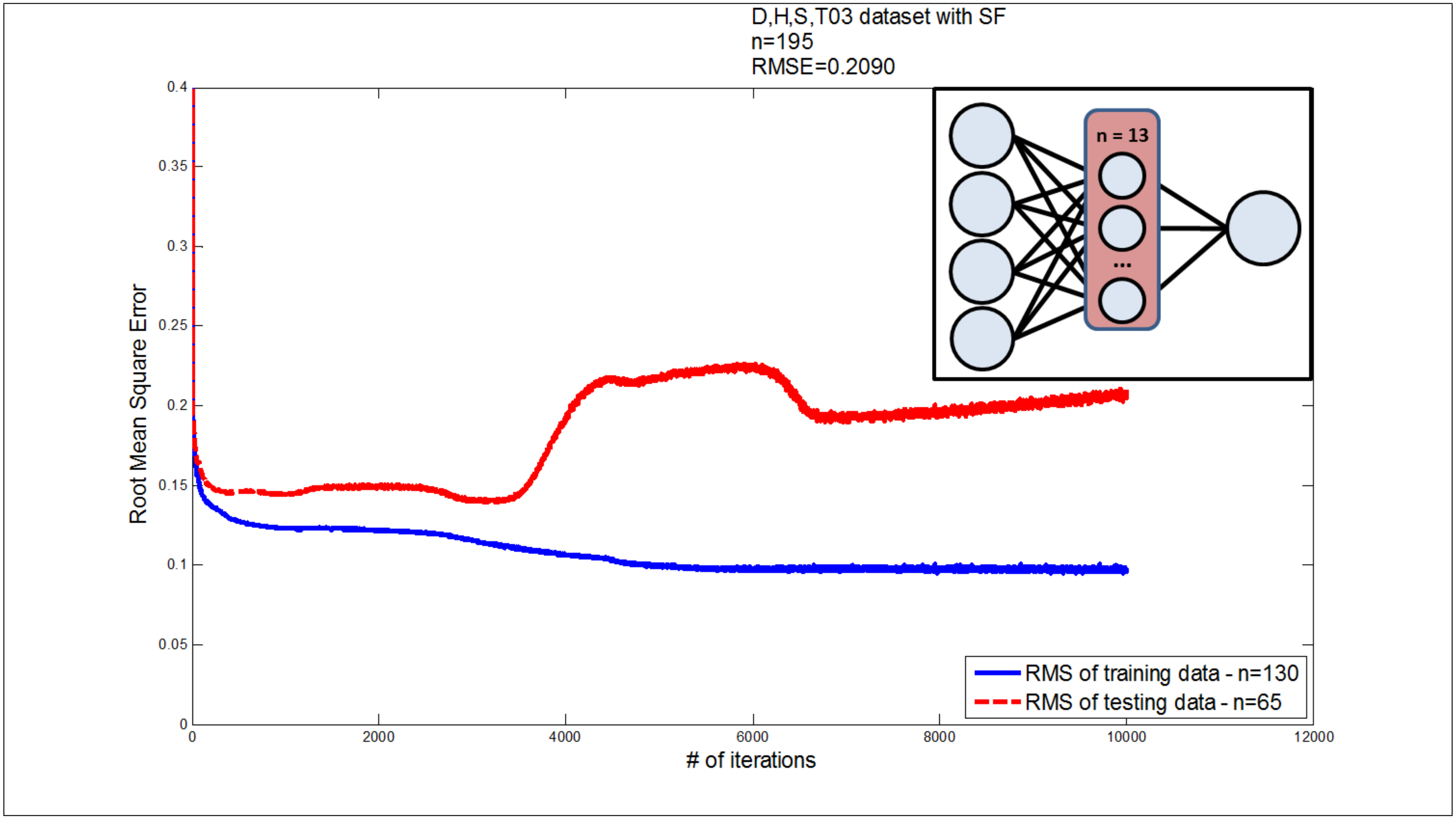
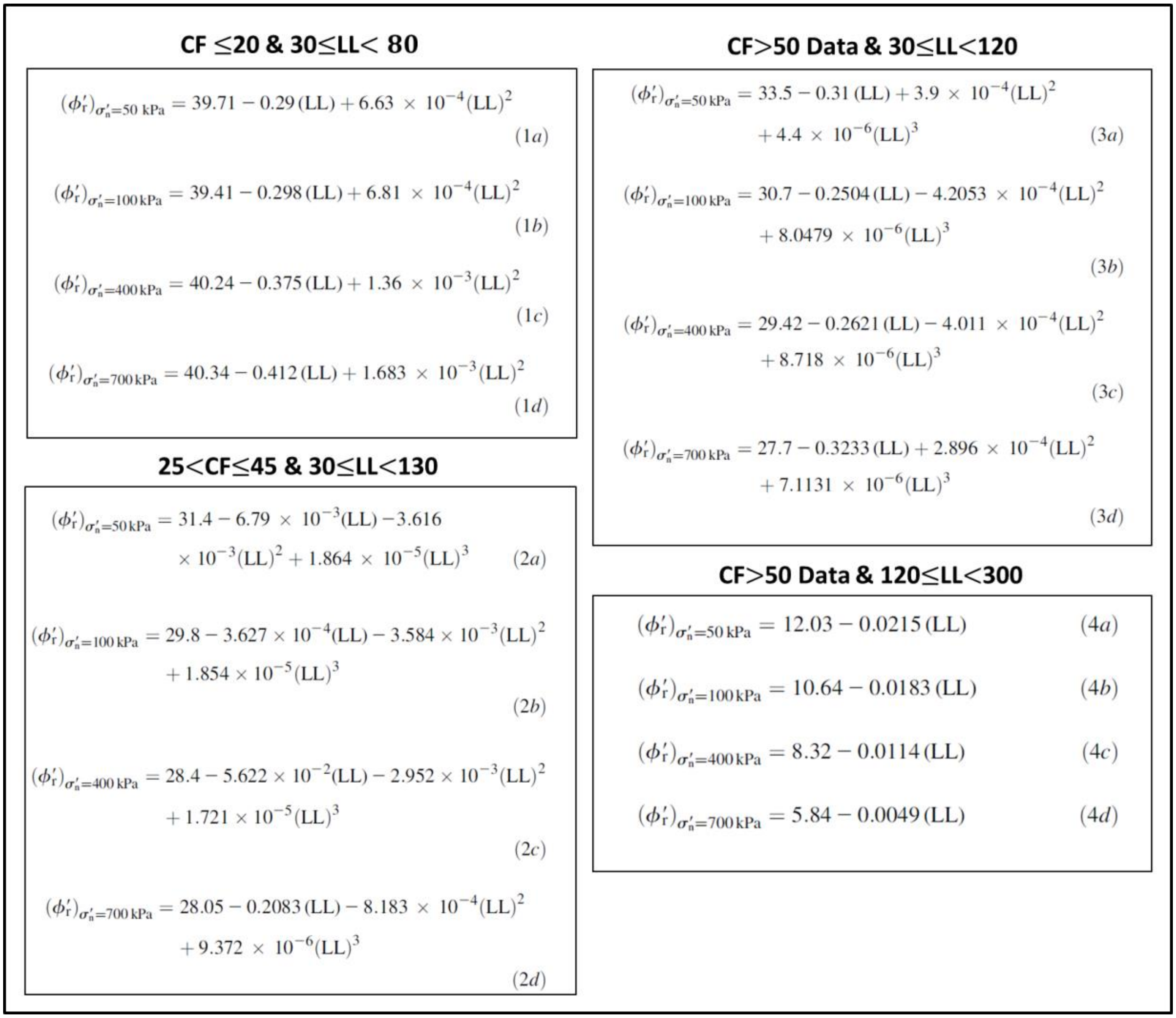


Figure 21: 13 node BPNN model tested for optimization.



CF ≤20 & 30≤LL< 80

$$(\phi'_\mathrm{r})_{\sigma'_\mathrm{n}=50\ \mathrm{kPa}} = 39.71 - 0.29\,(\mathrm{LL}) + 6.63 \times 10^{-4}(\mathrm{LL})^2 \tag{1a}$$

$$(\phi'_\mathrm{r})_{\sigma'_\mathrm{n}=100\,\mathrm{kPa}} = 39.41 - 0.298\,(\mathrm{LL}) + 6.81 \times 10^{-4}(\mathrm{LL})^2 \tag{1b}$$

$$(\phi'_\mathrm{r})_{\sigma'_\mathrm{n}=400\,\mathrm{kPa}} = 40.24 - 0.375\,(\mathrm{LL}) + 1.36 \times 10^{-3}(\mathrm{LL})^2 \tag{1c}$$

$$(\phi'_\mathrm{r})_{\sigma'_\mathrm{n}=700\,\mathrm{kPa}} = 40.34 - 0.412\,(\mathrm{LL}) + 1.683 \times 10^{-3}(\mathrm{LL})^2 \tag{1d}$$

25<CF≤45 & 30≤LL<130

$$(\phi'_\mathrm{r})_{\sigma'_\mathrm{n}=50\,\mathrm{kPa}} = 31.4 - 6.79 \times 10^{-3}(\mathrm{LL}) - 3.616 \times 10^{-3}(\mathrm{LL})^2 + 1.864 \times 10^{-5}(\mathrm{LL})^3 \tag{2a}$$

$$(\phi'_\mathrm{r})_{\sigma'_\mathrm{n}=100\,\mathrm{kPa}} = 29.8 - 3.627 \times 10^{-4}(\mathrm{LL}) - 3.584 \times 10^{-3}(\mathrm{LL})^2 + 1.854 \times 10^{-5}(\mathrm{LL})^3 \tag{2b}$$

$$(\phi'_\mathrm{r})_{\sigma'_\mathrm{n}=400\,\mathrm{kPa}} = 28.4 - 5.622 \times 10^{-2}(\mathrm{LL}) - 2.952 \times 10^{-3}(\mathrm{LL})^2 + 1.721 \times 10^{-5}(\mathrm{LL})^3 \tag{2c}$$

$$(\phi'_\mathrm{r})_{\sigma'_\mathrm{n}=700\,\mathrm{kPa}} = 28.05 - 0.2083\,(\mathrm{LL}) - 8.183 \times 10^{-4}(\mathrm{LL})^2 + 9.372 \times 10^{-6}(\mathrm{LL})^3 \tag{2d}$$

CF>50 Data & 30≤LL<120

$$(\phi'_\mathrm{r})_{\sigma'_\mathrm{n}=50\,\mathrm{kPa}} = 33.5 - 0.31\,(\mathrm{LL}) + 3.9 \times 10^{-4}(\mathrm{LL})^2 + 4.4 \times 10^{-6}(\mathrm{LL})^3 \tag{3a}$$

$$(\phi'_\mathrm{r})_{\sigma'_\mathrm{n}=100\,\mathrm{kPa}} = 30.7 - 0.2504\,(\mathrm{LL}) - 4.2053 \times 10^{-4}(\mathrm{LL})^2 + 8.0479 \times 10^{-6}(\mathrm{LL})^3 \tag{3b}$$

$$(\phi'_\mathrm{r})_{\sigma'_\mathrm{n}=400\,\mathrm{kPa}} = 29.42 - 0.2621\,(\mathrm{LL}) - 4.011 \times 10^{-4}(\mathrm{LL})^2 + 8.718 \times 10^{-6}(\mathrm{LL})^3 \tag{3c}$$

$$(\phi'_\mathrm{r})_{\sigma'_\mathrm{n}=700\,\mathrm{kPa}} = 27.7 - 0.3233\,(\mathrm{LL}) + 2.896 \times 10^{-4}(\mathrm{LL})^2 + 7.1131 \times 10^{-6}(\mathrm{LL})^3 \tag{3d}$$

CF>50 Data & 120≤LL<300

$$(\phi'_\mathrm{r})_{\sigma'_\mathrm{n}=50\,\mathrm{kPa}} = 12.03 - 0.0215\,(\mathrm{LL}) \tag{4a}$$

$$(\phi'_\mathrm{r})_{\sigma'_\mathrm{n}=100\,\mathrm{kPa}} = 10.64 - 0.0183\,(\mathrm{LL}) \tag{4b}$$

$$(\phi'_\mathrm{r})_{\sigma'_\mathrm{n}=400\,\mathrm{kPa}} = 8.32 - 0.0114\,(\mathrm{LL}) \tag{4c}$$

$$(\phi'_\mathrm{r})_{\sigma'_\mathrm{n}=700\,\mathrm{kPa}} = 5.84 - 0.0049\,(\mathrm{LL}) \tag{4d}$$

Figure 22: Equations for prediction of ø$_r$’ subdivided by CF (Stark and Hussain 2013).

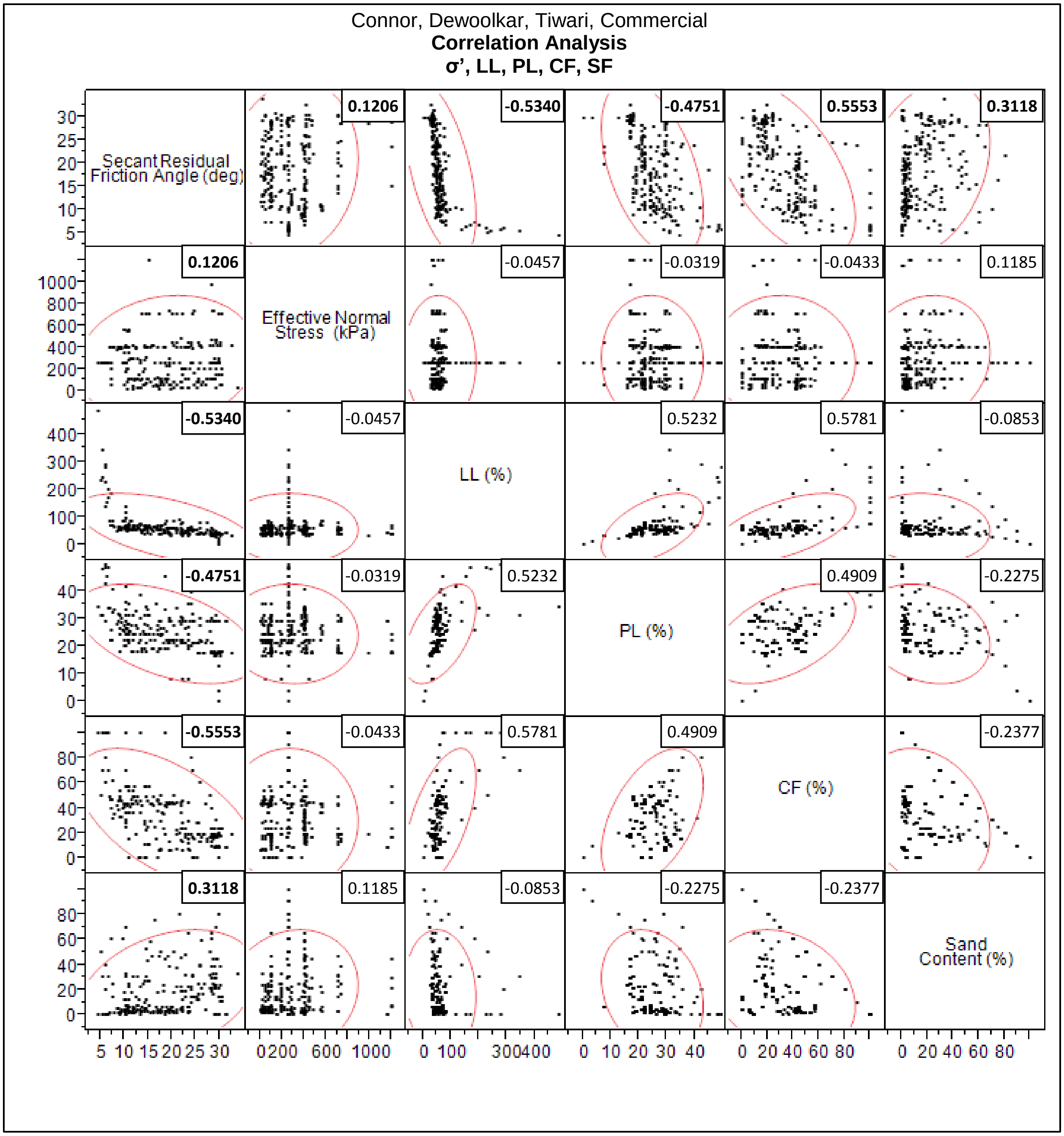
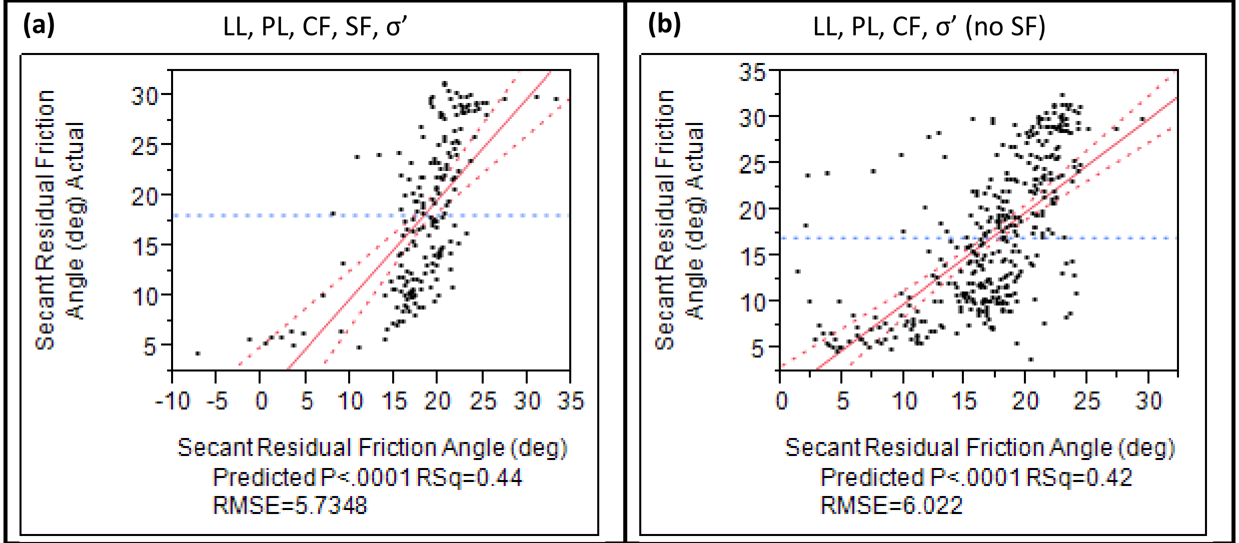


Figure 23: Scatter plots showing correlations between all available of soil parameters; correlations are provided in the upper right corner of each plot.

Connor, Dewoolkar, Tiwari 2003, Commercial
Least Squares Multivariate Linear Regression Model



Number of data points	195	Number of data points	388
RSquare	0.440805	RSquare	0.419844
RSquare Adj	**0.426011**	**RSquare Adj**	**0.413785**
P value (prob>F)	**<0.0001**	**P value (prob>F)**	**<0.0001**

Figure 24: Summary of least-squares regression for soil datasets with SF ((a) n=195) and without SF ((b) n=388).

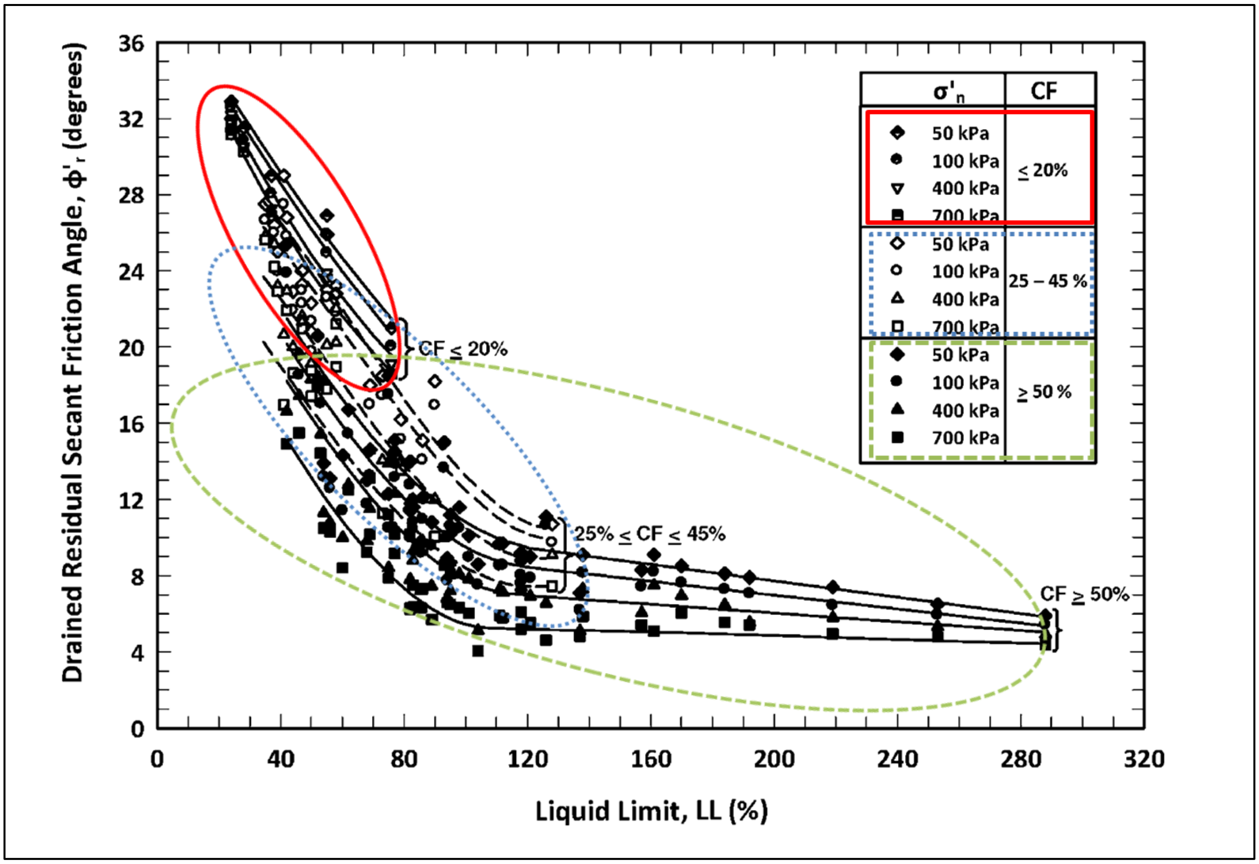


Figure 25: Relationship of friction angle with LL across a large dataset (Stark and Hussain 2013) with three CF regions identified corresponding to three BPNN developed in this thesis.

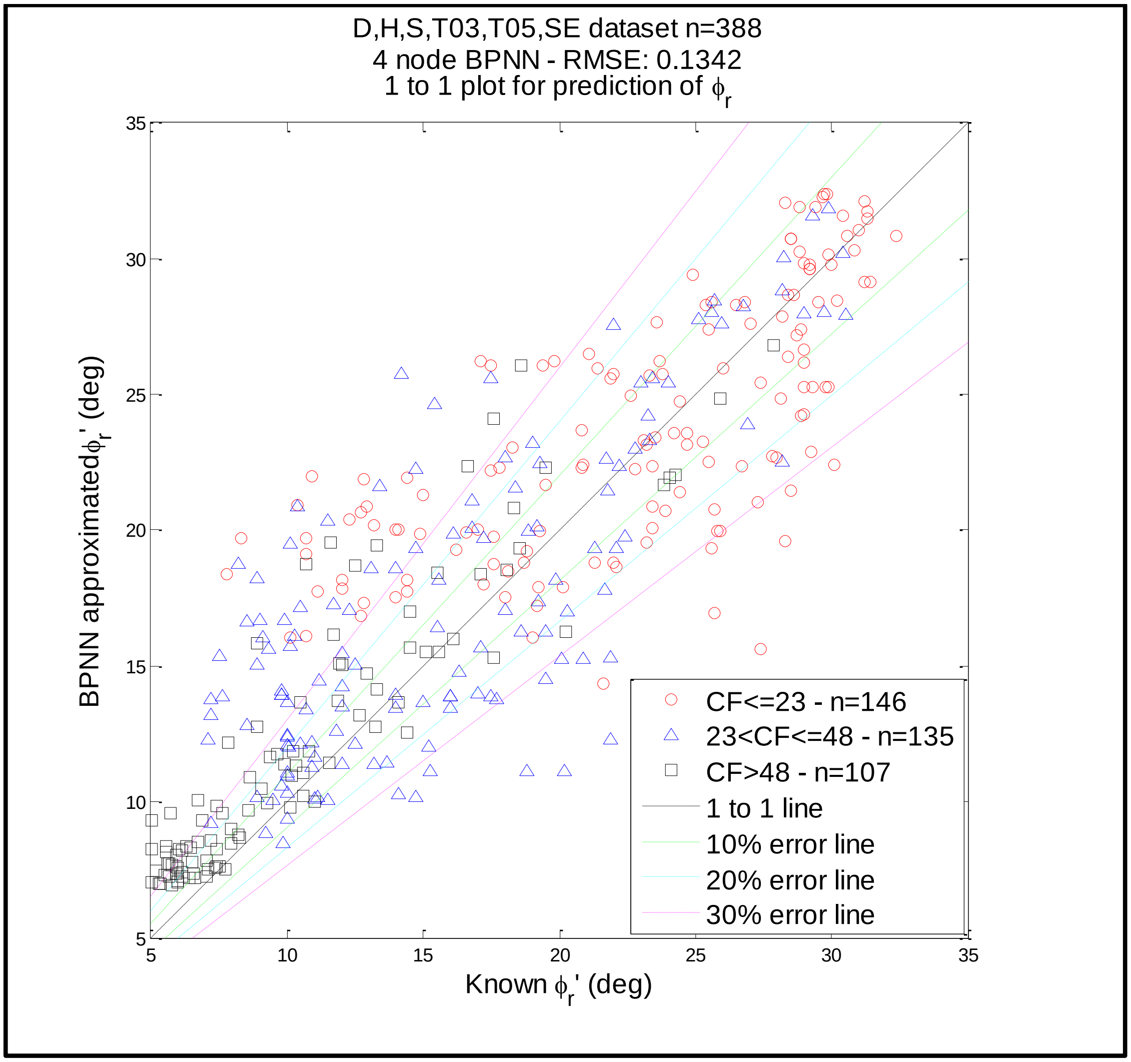


Figure 26 – Predicted ϕ_r' values using a 4 node BPNN plotted against known values with the CF distribution shown.

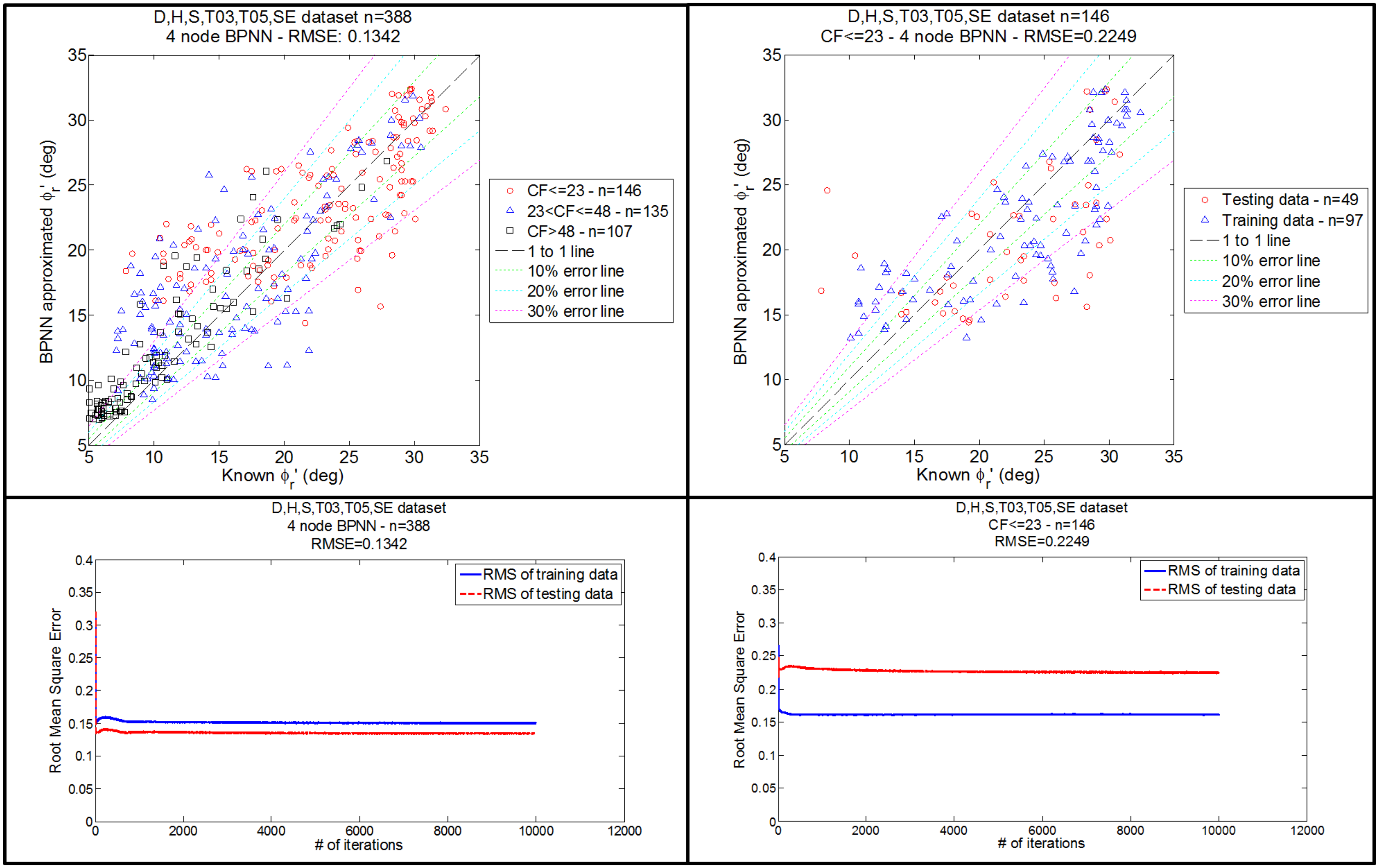


Figure 27: Typical BPNN vs. low CF range BPNN for same dataset.

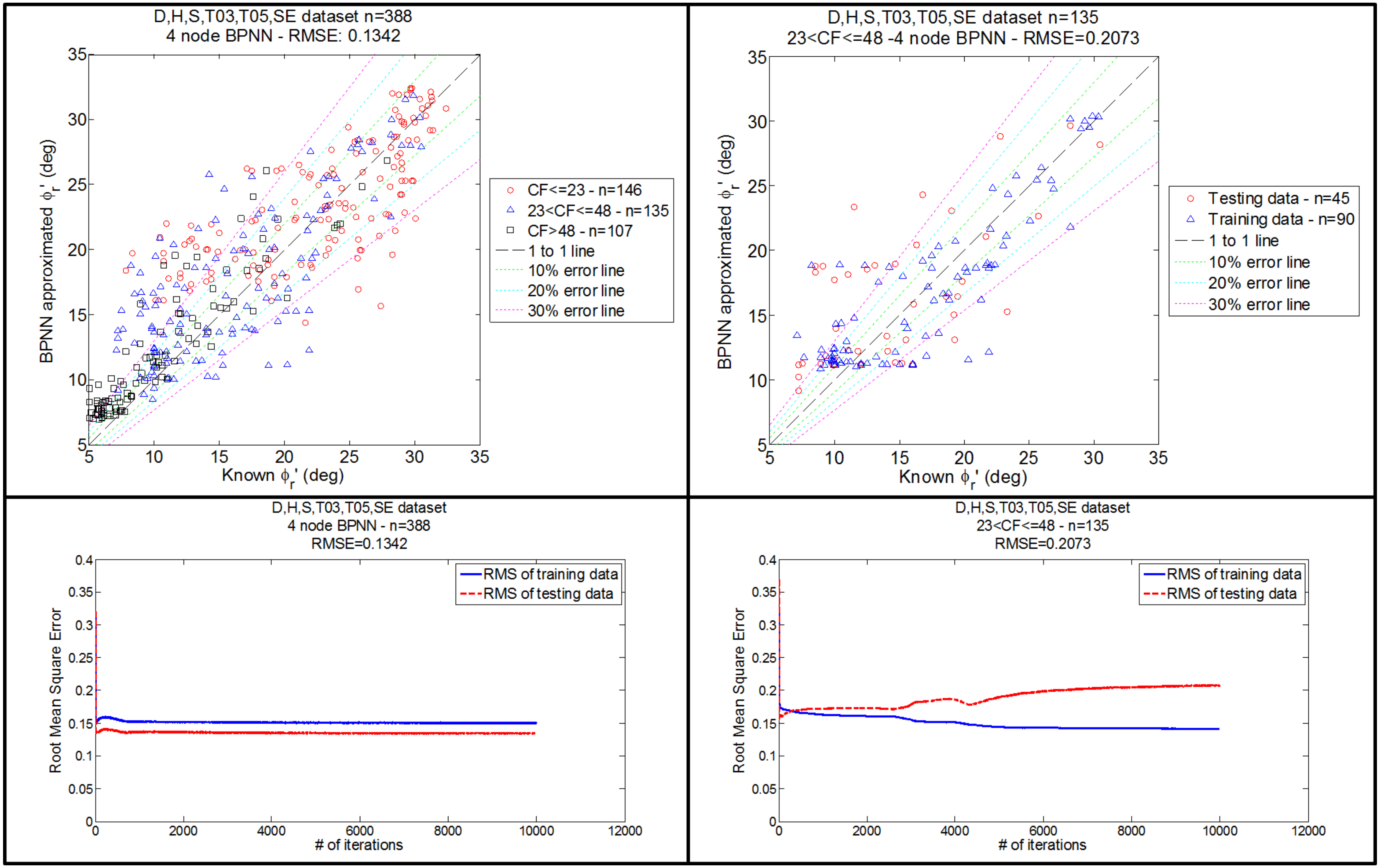


Figure 28: Typical BPNN vs. middle CF range BPNN for same dataset.

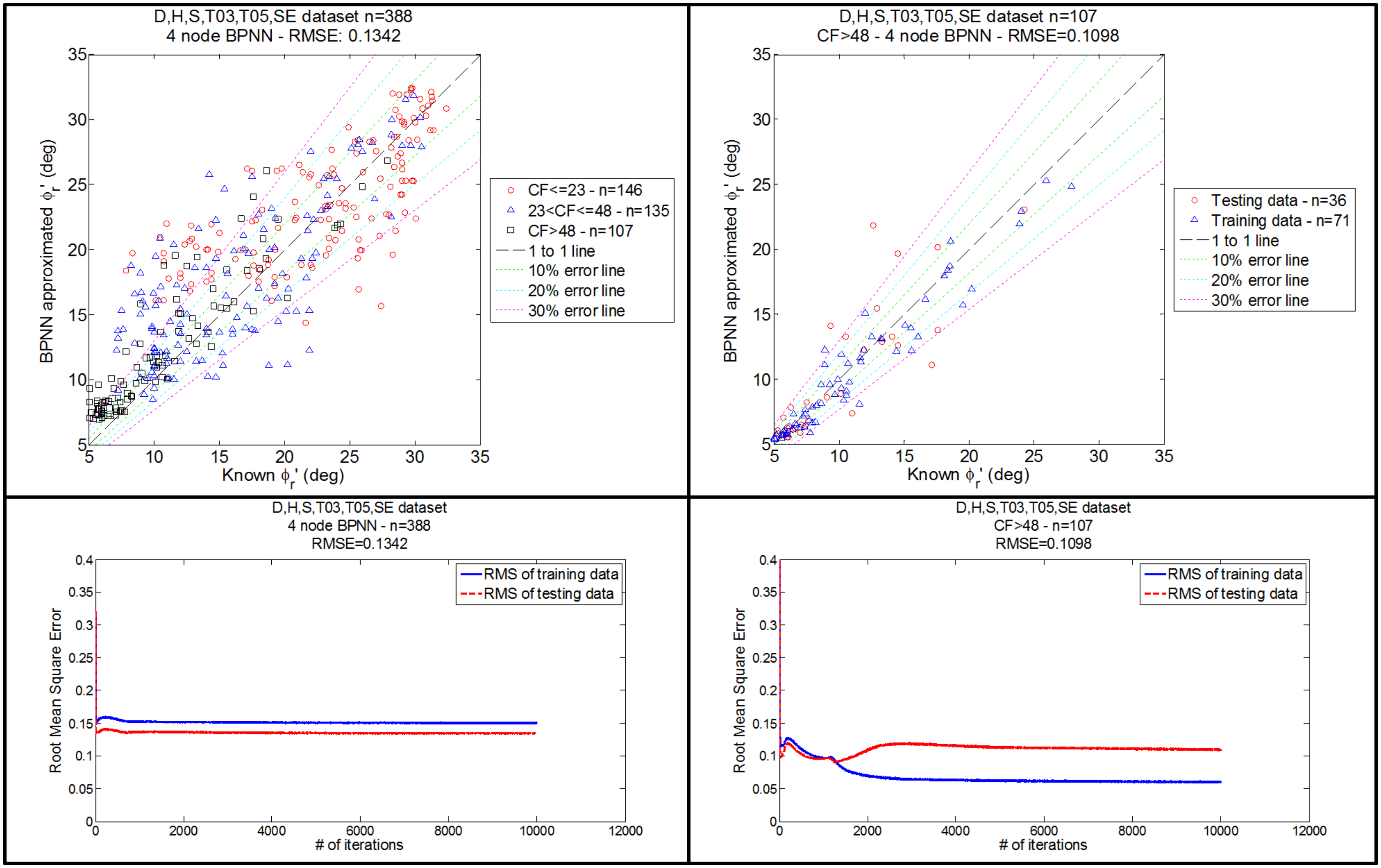


Figure 29: Typical BPNN vs. high CF range BPNN for same dataset.

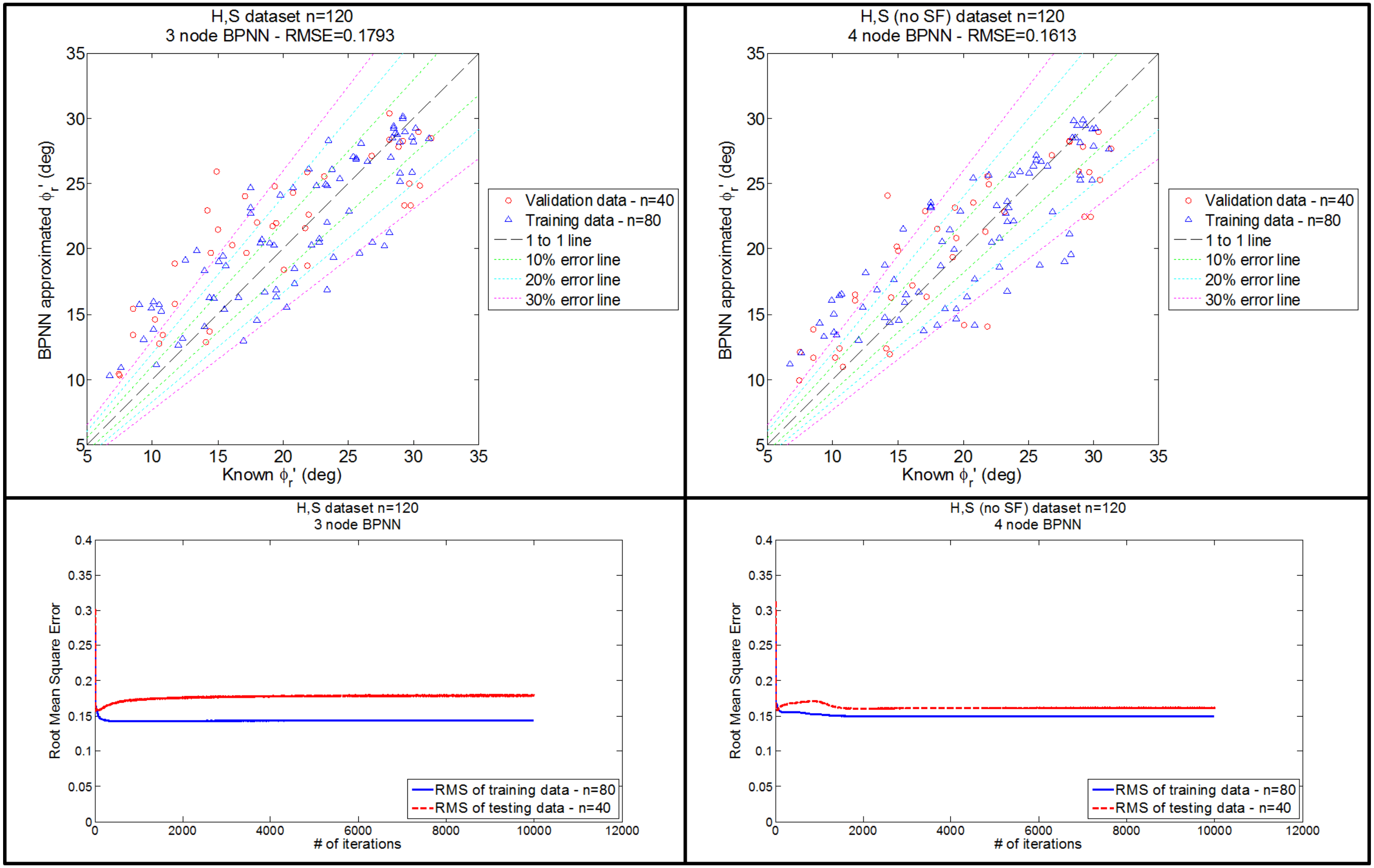


Figure 30: Initial comparison of BPNN performance with and without SF.

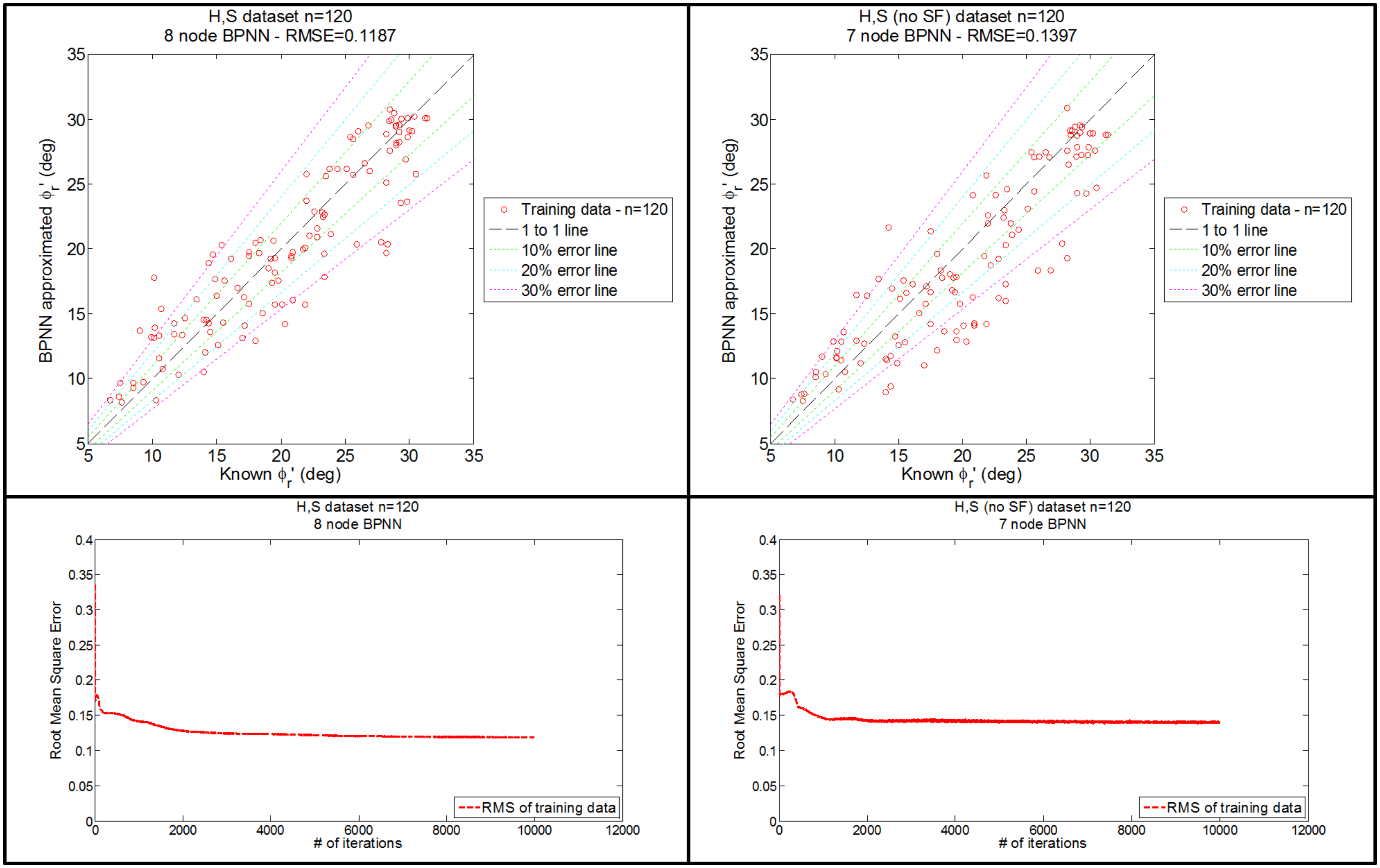


Figure 31: Training only comparison of BPNN performance with and without SF for H & S datasets.

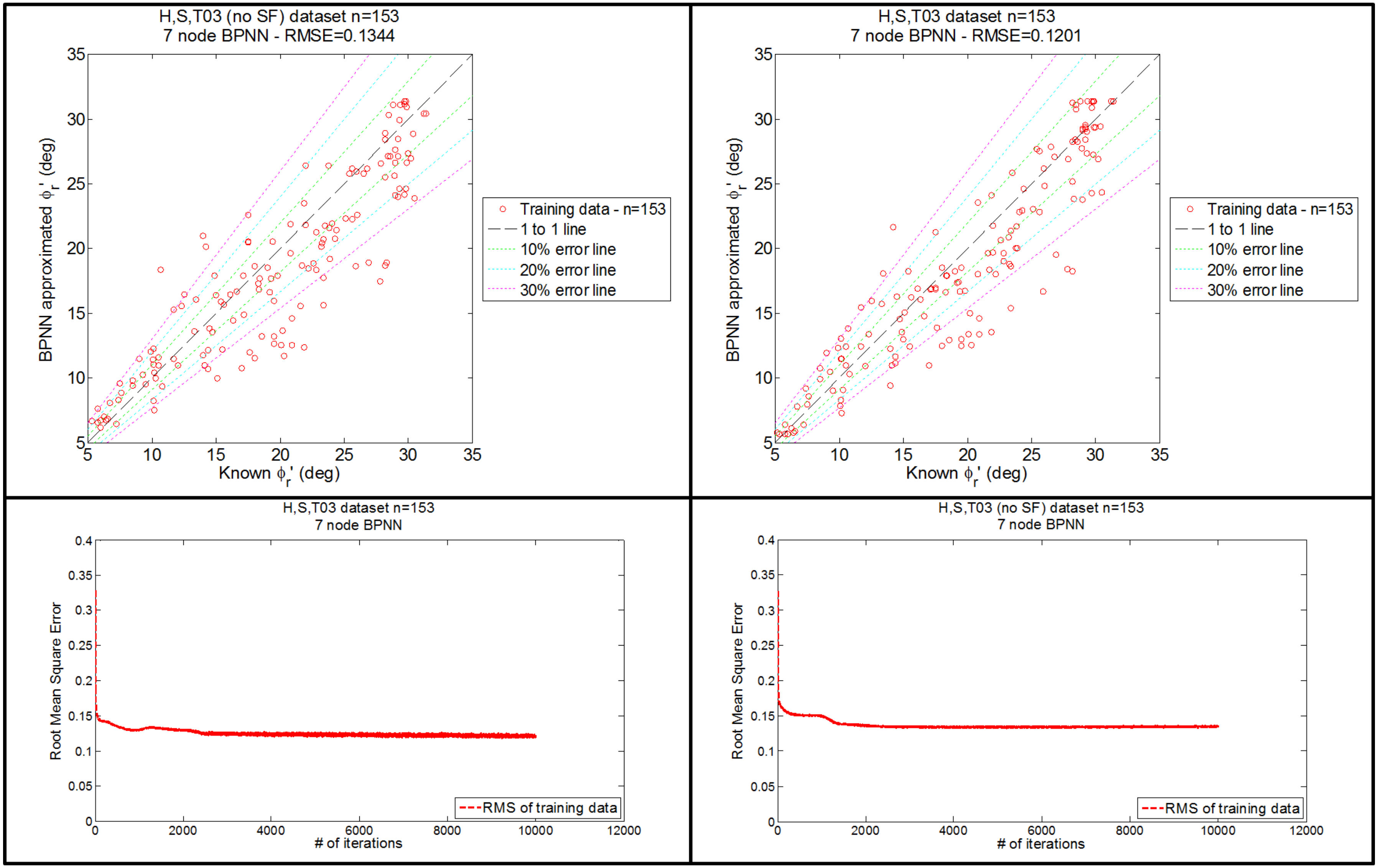
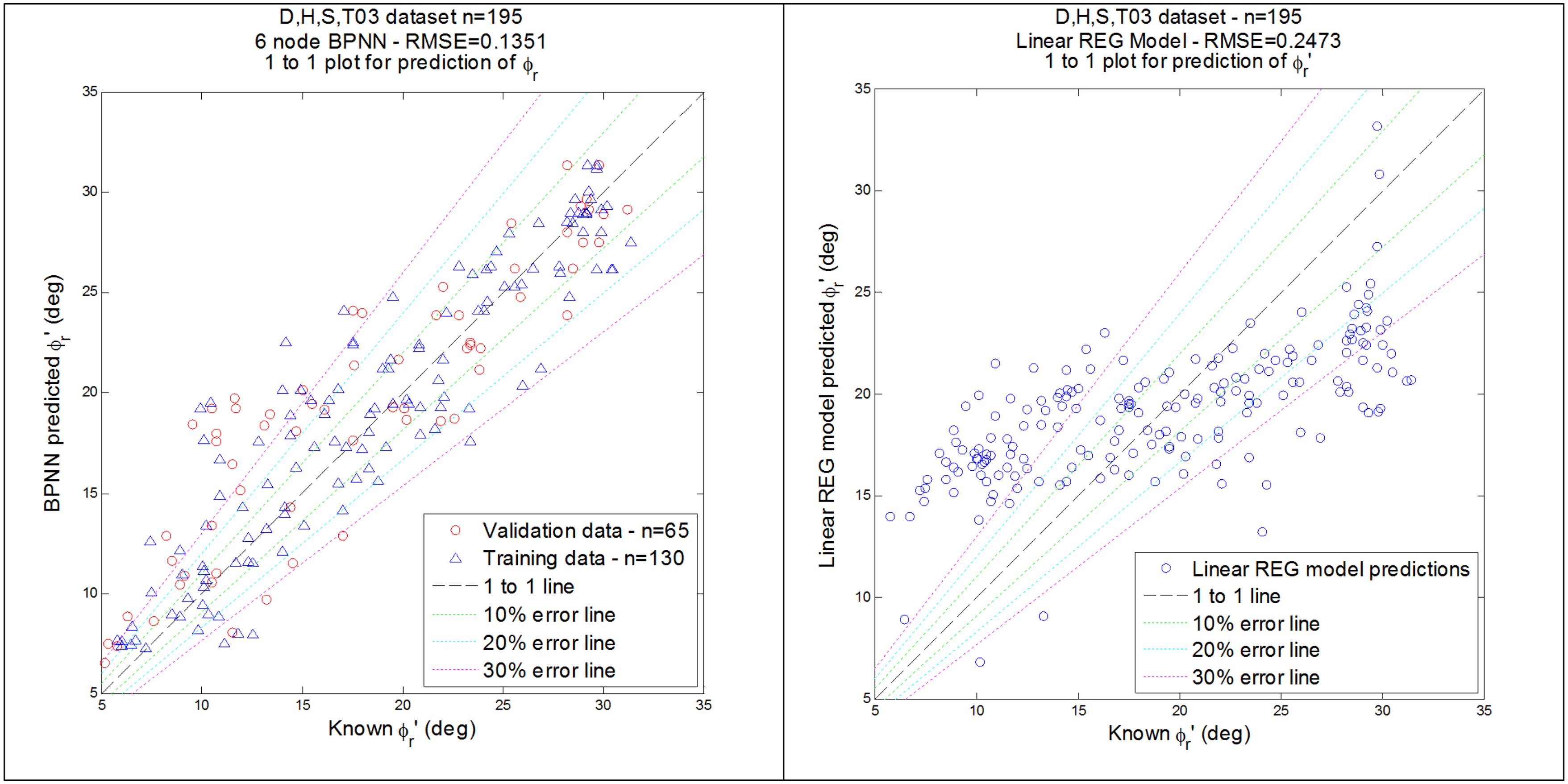


Figure 32: Training only comparison of BPNN performance with and without SF for H, S, & T03 datasets.



D,H,S,T03 dataset n=195
6 node BPNN - RMSE=0.1351
1 to 1 plot for prediction of ϕ_r

D,H,S,T03 dataset - n=195
Linear REG Model - RMSE=0.2473
1 to 1 plot for prediction of ϕ_r'

Figure 33: Comparison of BPNN to REG model for input parameters: CF, SF, LL, PL, & σ'.

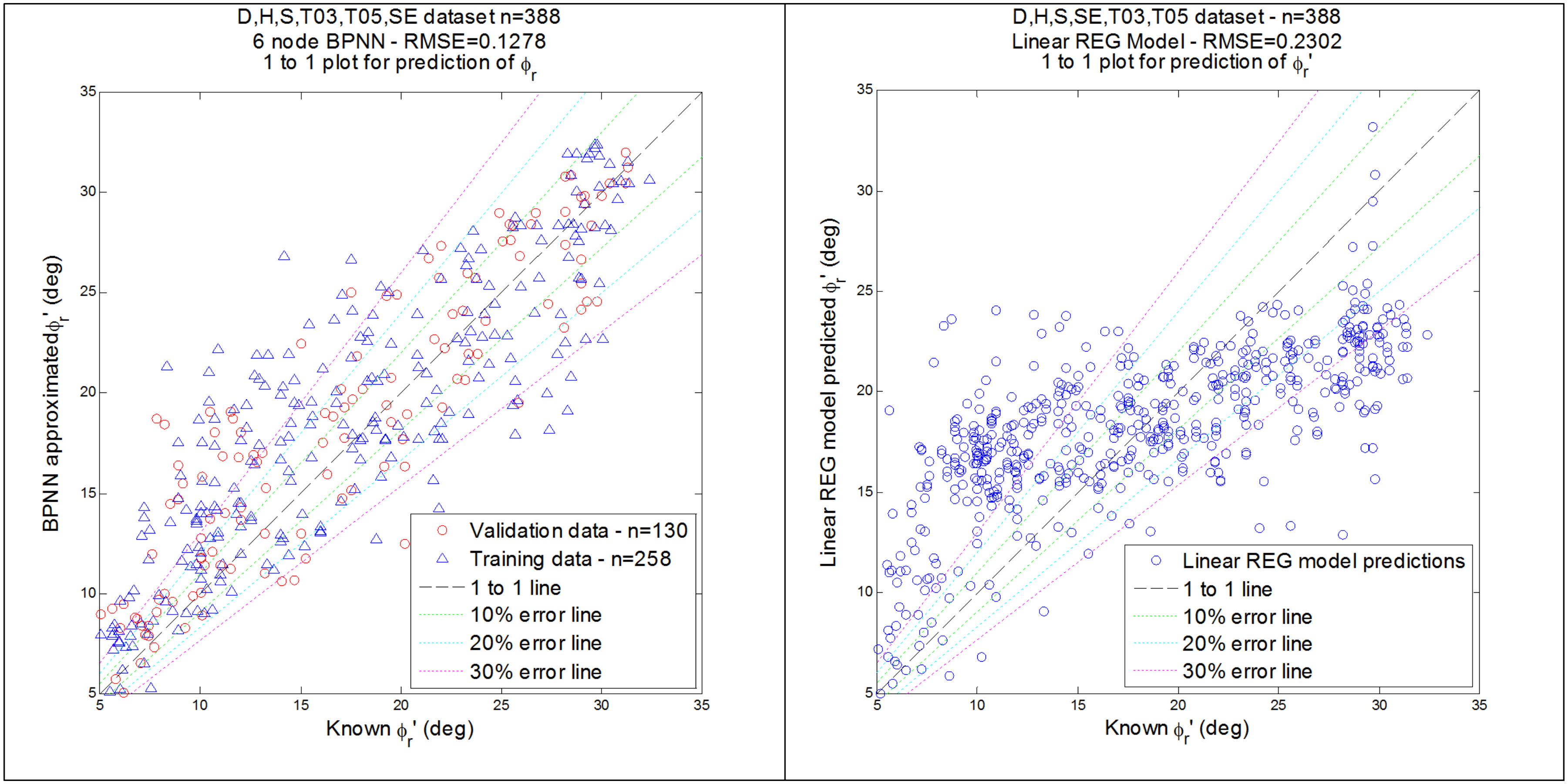


Figure 34: Comparison of BPNN to REG model for input parameters: CF, LL, PL, & σ' (no SF).

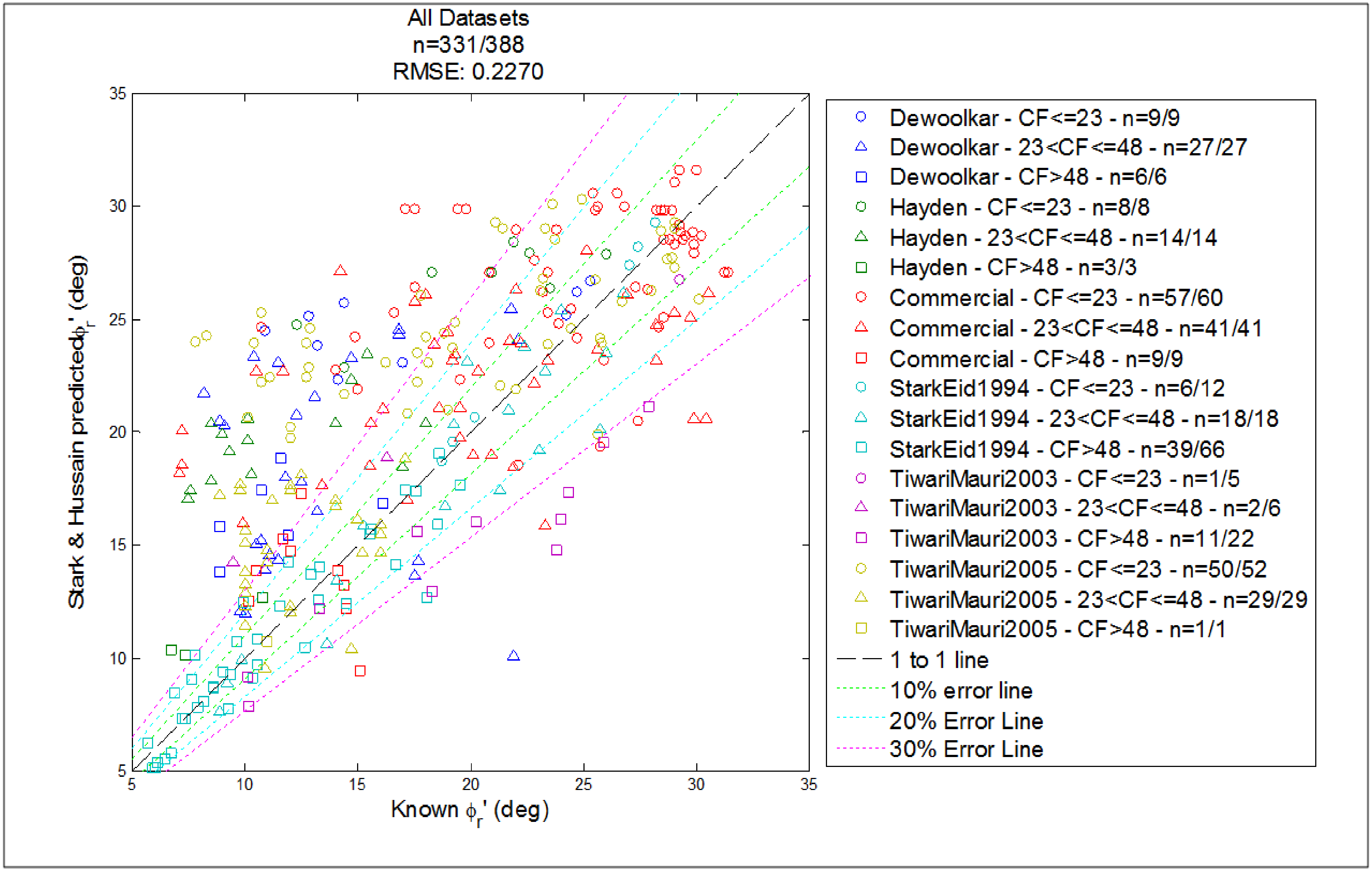


Figure 35: ø$_r$' predictions for all data using equations from Stark & Hussain 2013.

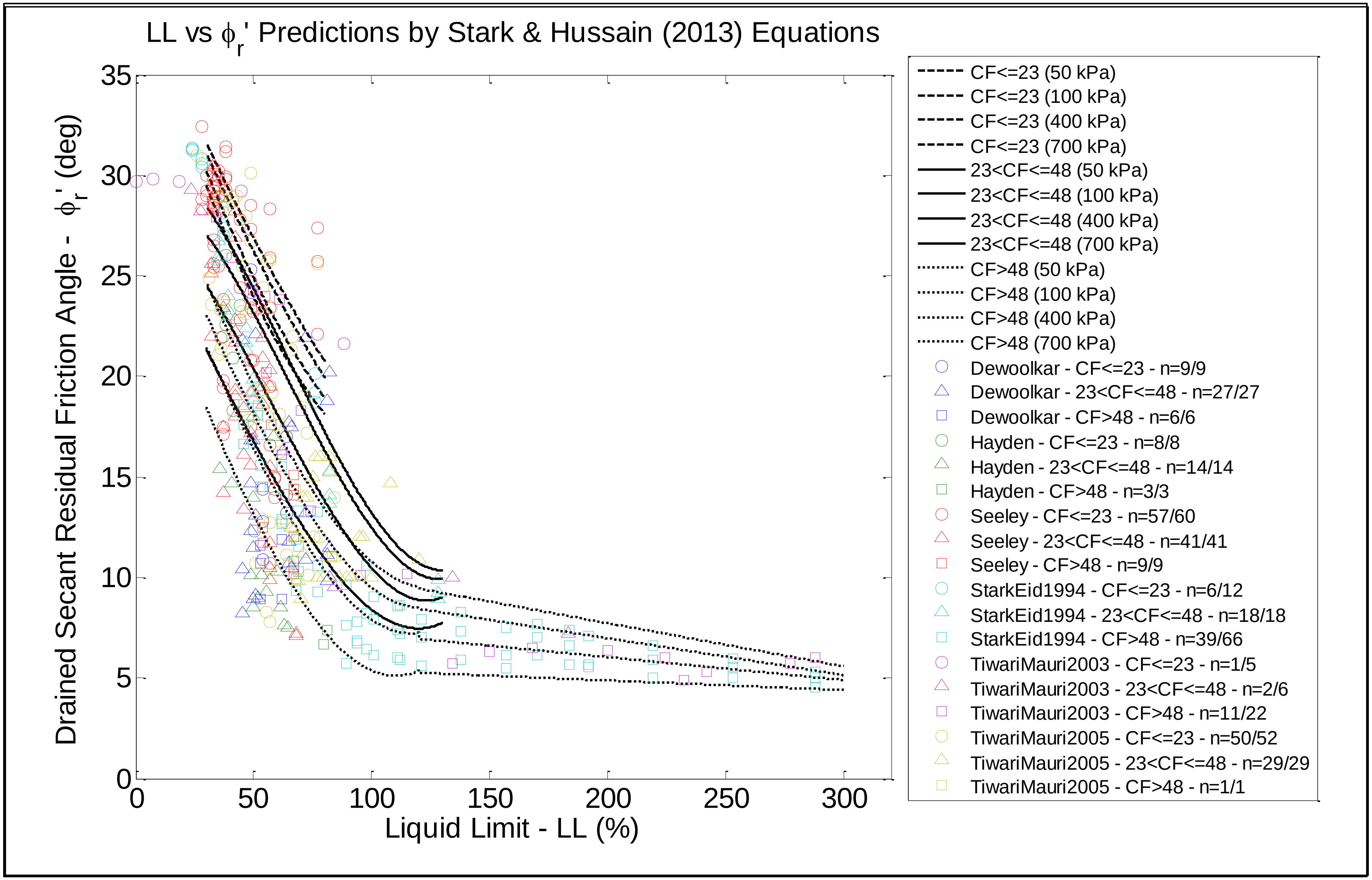


Figure 36: ϕ_r' predictions from Stark & Hussain (2013) correlations shown on a ϕ_r' vs. LL plot.

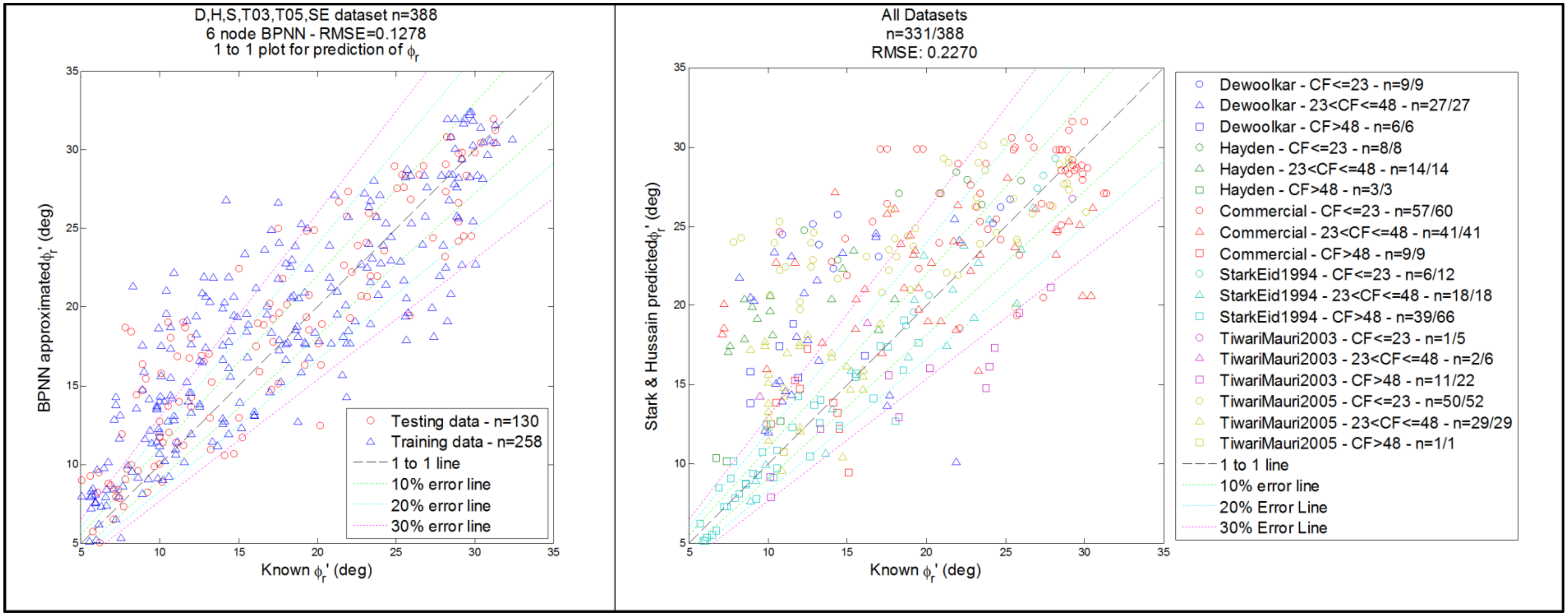


Figure 37: Comparison between ø$_r$’ predictions using a BPNN and Stark and Hussain equations for all data.

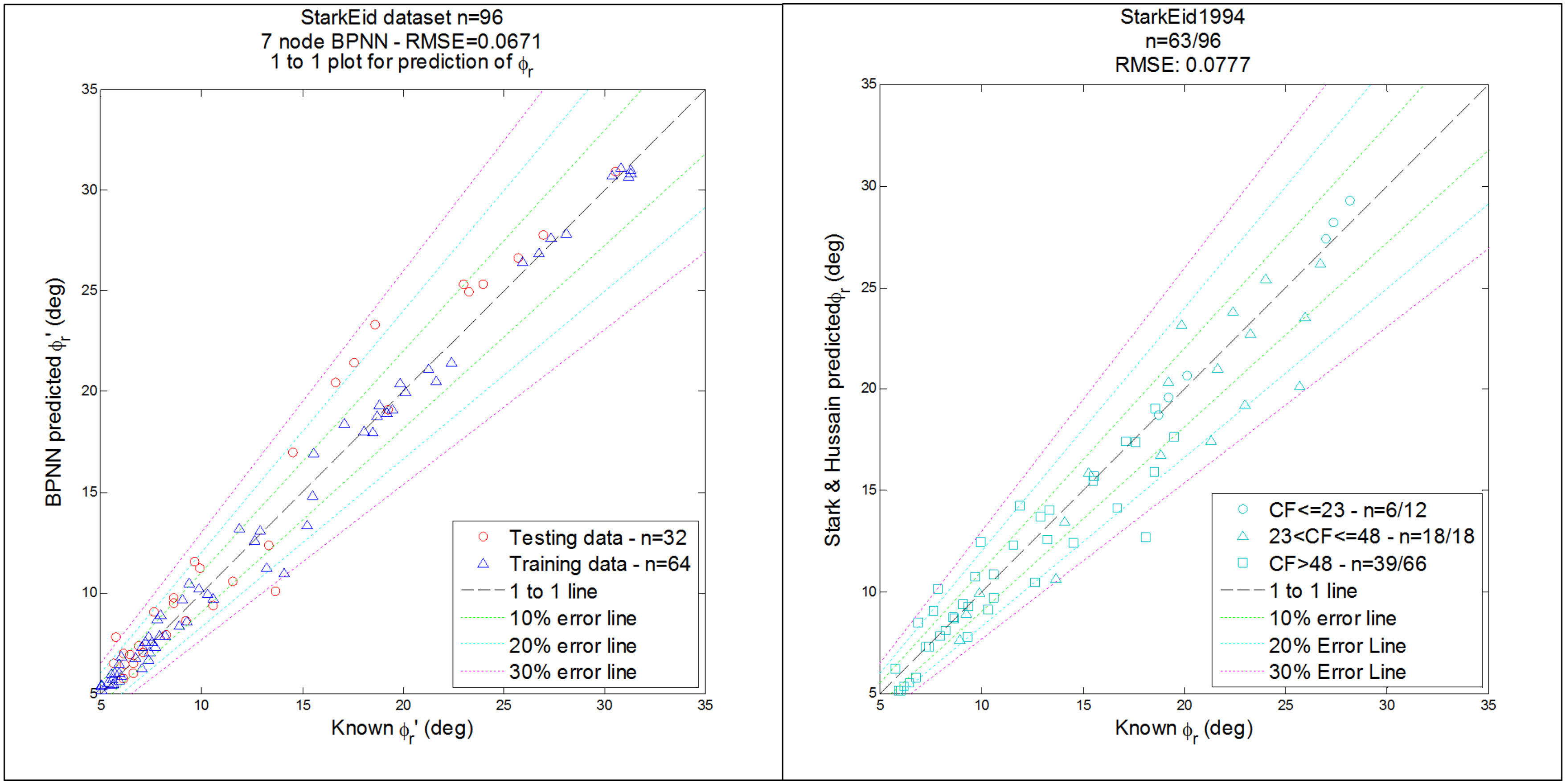


Figure 38: Comparison between ϕ_r' predictions using a BPNN and Stark and Hussain equations for Stark & Eid 1994 data.

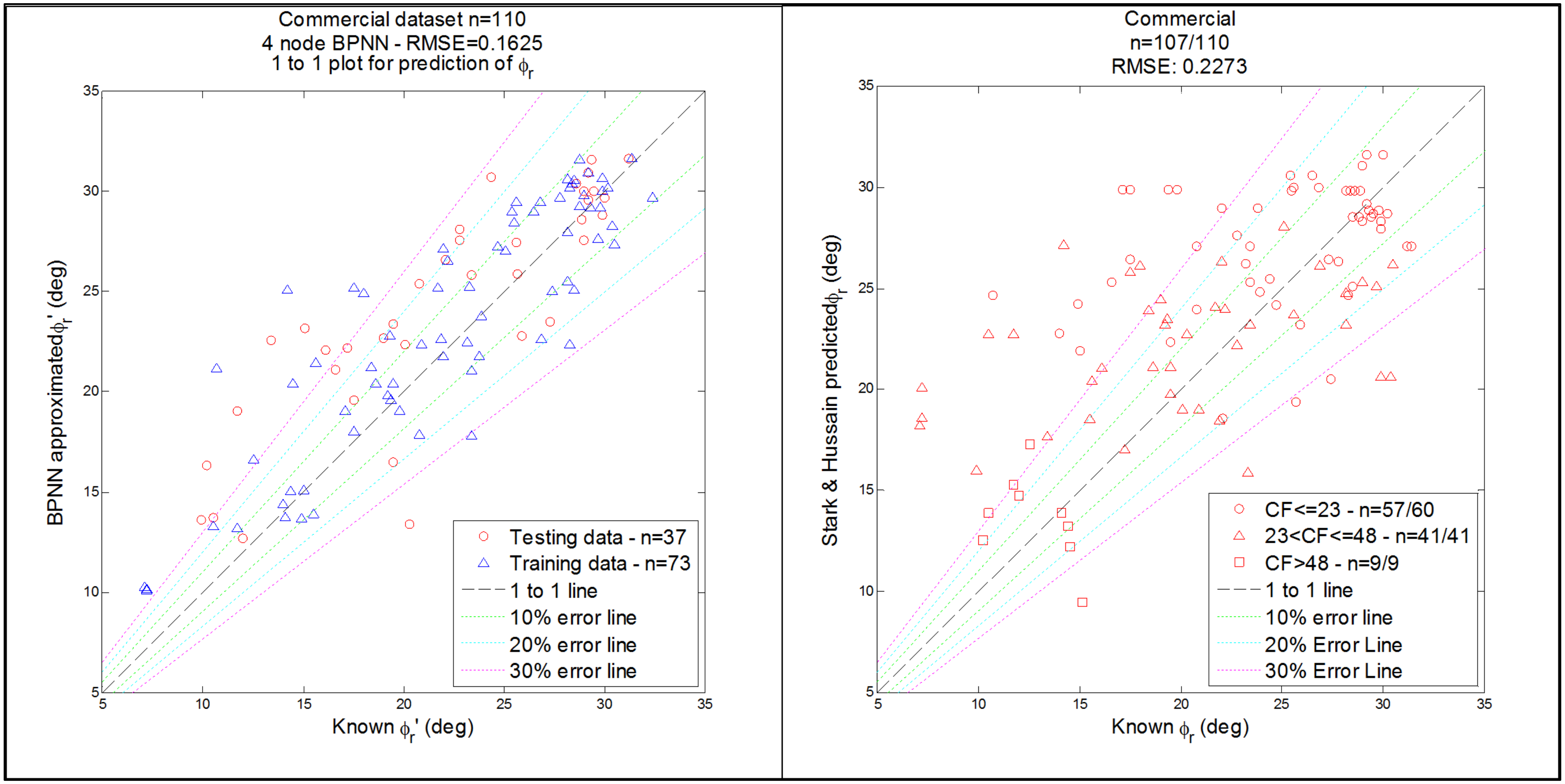


Figure 39: Comparison between ϕ_r' predictions using a BPNN and Stark and Hussain equations for Commercial data.

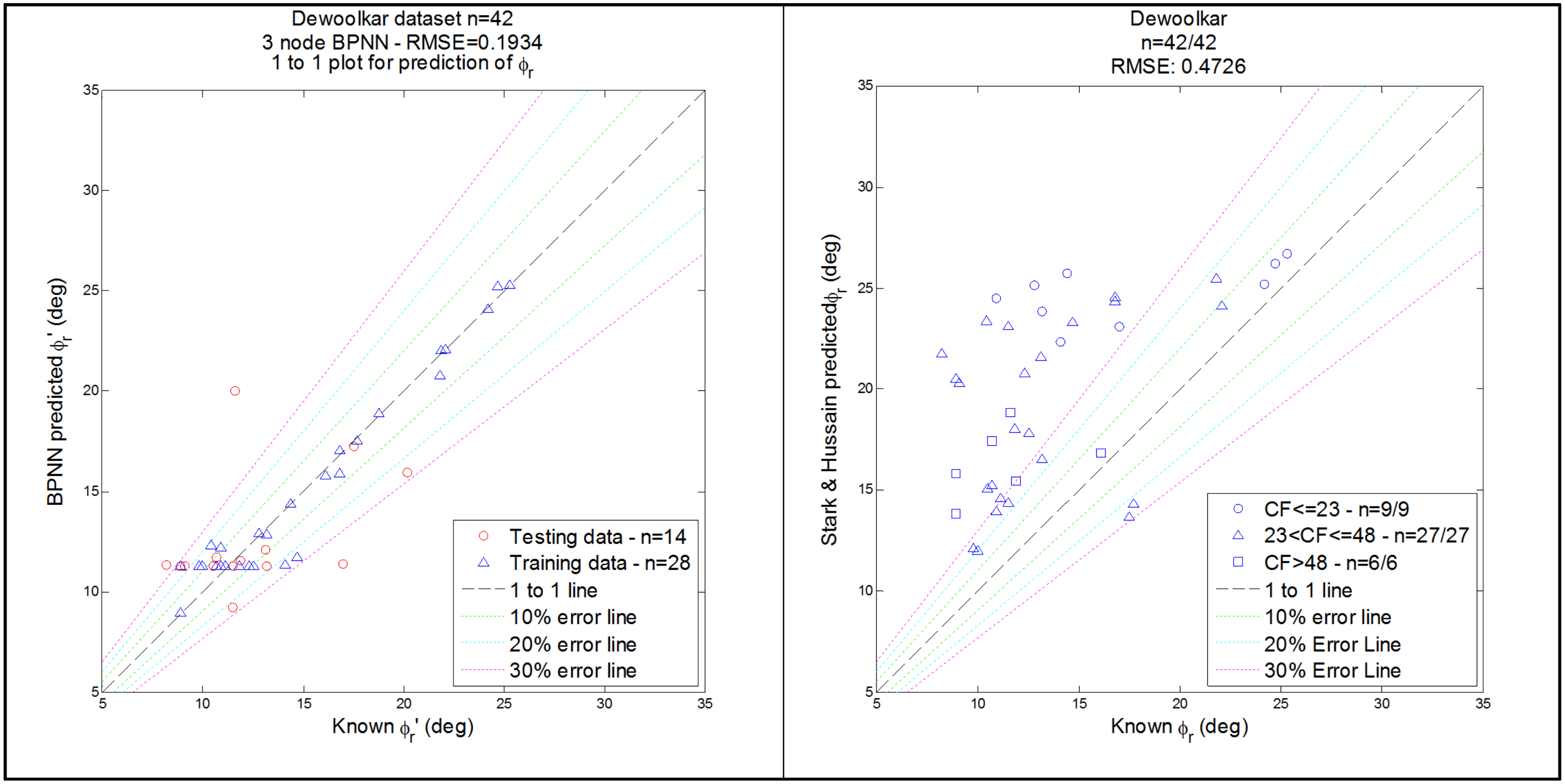


Figure 40: Comparison between $ø_r$’ predictions using a BPNN and Stark and Hussain equations for Dewoolkar data.

Tables

Table 1: Comparison of BPNN and regression model prediction accuracies.

Model	R^2 - BPNN	R^2 – REG
Unconfined Compressive Strength of Carbonate Rocks (Ceryan, Okkan et al. 2013)	0.81	0.74
Shearing Resistance Angle (ø) for Uniform Sands (Sezer 2013)	0.98	0.77
Tunnel Boring Machine Performance (Torabi, Shirazi et al. 2013)	0.99	0.79

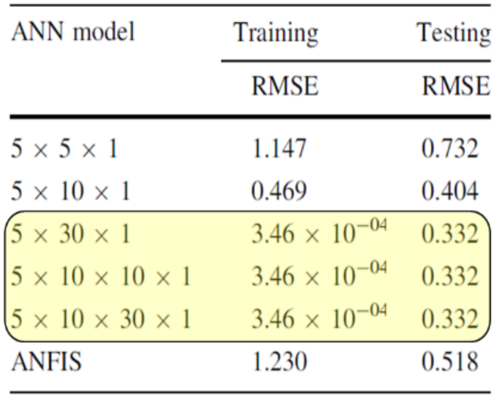
Table 2: List of parameters associated with ø$_r$’.

Symbol	Parameter	Units	Description
LL	**Liquid Limit**	%	Minimum water content at which a cohesive soil continues to exhibit plastic properties.
PL	**Plastic Limit**	%	Maximum water content at which a cohesive soil continues to exhibit plastic soil properties
CF	**Clay Fraction**	%	Mass fraction of clay (soil solids smaller than 0.002 mm)
SF	**Sand Fraction**	%	Mass fraction of sand (soil solids between 2-0.075 mm in diameter)
σ’	**Normal Stress**	kPa	Normal stress loading applied to a soil sample during shear strength test

Table 3: 6 soil datasets used in this study.

Dataset Name	Abbreviation	n - Sample Size	Internally Consistent	SF	Source
Dewoolkar	D	43	X	Yes	Dewoolkar & Huzjak 2005
Hayden	H	25	Yes	Yes	UVM - Connor Hayden - Honors Thesis Manuscript
Tiwari 2003	TM03	33	Yes	Yes	Tiwari & Mauri 2003
Tiwari 2005	TM05	82	Yes	X	Tiwari & Mauri 2005
Stark & Eid	SE	96	Yes	X	Stark & Eid 1994
Commercial	S	97	X	Yes	Commercial Dataset

Table 4: BPNN performance for varied structures.



ANN model	Training RMSE	Testing RMSE
5 × 5 × 1	1.147	0.732
5 × 10 × 1	0.469	0.404
5 × 30 × 1	3.46×10^{-04}	0.332
5 × 10 × 10 × 1	3.46×10^{-04}	0.332
5 × 10 × 30 × 1	3.46×10^{-04}	0.332
ANFIS	1.230	0.518

Table 5: CF ranges as considered by Stark & Hussain 2013 and this study.

	Low CF Range	Middle CF Range	High CF Range
Stark & Hussain 2013	CF ≤ 20	25 $<$ CF ≤ 45	CF > 50
This Study	**CF ≤ 23**	**23 $<$ CF ≤ 48**	**CF > 48**

Equations

$$\boldsymbol{\tau = \sigma' tan(\text{ø}'_r)} \tag{10}$$

Where: σ' = normal stress loading

$$\boldsymbol{x_{norm} = \frac{x - x_{min}}{x_{max} - x_{min}}} \tag{11}$$

Where:
x_{norm} = *normalized data value*; x = *parameter data value*
x_{max} = *parameter minimum*; x_{min} = *parameter maximum*

$$\boldsymbol{f(x) = \frac{1}{1 + e^{-x}}} \tag{12}$$

Where:
x = *node data*; e − *exponential function*

$$\boldsymbol{x = x_{norm}(x_{max} - x_{min}) + \ x_{min}} \tag{13}$$

Where:
x_{norm} = *normalized data value*; x = *parameter data value*
x_{max} = *parameter minimum*; x_{min} = *parameter maximum*

$$\boldsymbol{\delta_{out} = f'(x)\left(o_k - o_p\right)} \tag{14}$$

Where:
δ_{out} = *output layer error*; $f'(x)$ = *derivative of logistic squashing function*
o_k = *known output vector*; o_p = *predicted output vector*

$$\boldsymbol{\delta_{hid} = f'(x)[\delta_{out} w_{out}]} \tag{15}$$

Where:
δ_{hid} = *hidden layer error*; $f'(x)$ = *derivative of logistic squashing function*
δ_{hid} = *output layer error*; w_{out} = *output weights*

$$\boldsymbol{\Delta w_{ij} = \eta \delta_j x_i} \tag{16}$$

Where:
Δw_{ij} = *change in weight matrix between layers i and j*
η = *updating coefficient (generally* 0.5); δ_j = *error for forward layer j*
x_i = *current values for nodes at backward layer i*

17

$$\boldsymbol{w_{ij}^{new}} = \boldsymbol{w_{ij}} + \Delta \boldsymbol{w_{ij}}$$

Where:

$w_{ij}^{new} = \ new\ weight\ matrix\ between\ layers\ i\ and\ j$
$w_{ij} = \ current\ weight\ matrix\ between\ layers\ i\ and\ j$
$\Delta w_{ij} = change\ in\ weight\ \ matrix\ between\ layers\ i\ and\ j$

18

$$\Delta \boldsymbol{w_{ij}} = \boldsymbol{\eta \delta_j x_i} + \boldsymbol{\alpha} \Delta \boldsymbol{w_{ij}^{prev}}$$

Where:

$\Delta w_{ij} = change\ in\ weight\ \ matrix\ between\ layers\ i\ and\ j;\ \eta = updating\ coefficient\ (generally\ 0.5)$
$\delta_j = error\ for\ forward\ layer\ j;\ x_i = current\ values\ for\ nodes\ at\ backward\ layer\ i$
$\alpha = momentum\ coefficient\ (generally\ 0.9);\ \Delta w_{ij}^{prev} = change\ in\ weight\ matrix\ from\ previous\ training\ step$

Using Backpropagation Neural Networks for the Prediction of Residual Shear Strength of Cohesive Soils

Luke Detwiler

APPENDIX

Appendix Contents

APPENDIX A – BPNN Code

Functions:

normalize()

```
function [normIn,minIn,maxIn,normOut,minOut,maxOut]=normalize(in,out)
% Luke Detwiler - Aug 2013

% Normalizes input and output data vectors for BPNN training. Provided
% input and output data matrices, normalize() determines linearly
% normalized values (0 to 1) based on minimum and maximum values. This
% process is applied on a column, by column basis, as data is parametrized
% by column. Minimum and maximum data values associated with each of these
% data ranges are provided for post-training data un-normalization using
% the function unNormalize().

% in - input data matrix (data point x parameter)
% out - output data matrix (data point x parameter)

% normIn - normalized input data matrix (data point x parameter)
% minIn - minimum input data value vector (1 x parameter)
% maxIn - maximum input data value vector (1 x parameter)
% normOut - normalized output data matrix (data point x parameter)
% minOut - minimum output data value vector (1 x parameter)
% maxOut - maximum output data value vector (1 x parameter)

%% Normalize Input values
% Preallocate
normIn=zeros(size(in));
minIn=zeros(1,size(in,2));
maxIn=zeros(1,size(in,2));
% Repeat for each data parameter (iterate by column)
for col=1:size(in,2)
%     Store minimum and maximums
    minIn(1,col)=min(in(:,col));
    maxIn(1,col)=max(in(:,col));
%     Calculate normalized data values
    normIn(:,col)=(in(:,col)-minIn(1,col))./(maxIn(1,col)-minIn(1,col));
    if isnan(normIn(:,col))
        normIn(:,col)=1;
    end
end

%% Normalize Output Values (same as above)
normOut=zeros(size(out));
minOut=zeros(1,size(out,2));
maxOut=zeros(1,size(out,2));
for col=1:size(out,2)
    minOut(1,col)=min(out(:,col));
    maxOut(1,col)=max(out(:,col));
    normOut(:,col)=(out(:,col)-minOut(1,col))./(maxOut(1,col)-minOut(1,col));
    if isnan(normOut(:,col))
        normOut(:,col)=1;
    end
end
```

backprop()

```
function Network=backprop(in,out,eta,alph,maxit,tol,hid,r)
% Luke Detwiler - 1/18/2014

% This function trains a one hidden layer BPNN (sized by hid) to the the
% dataset (in,out), given training parameters eta, alph, maxit, tol, & r.

% in - input dataset (# data subsets,# input parameters)
% out - output dataset (# data subsets, # output parameters)
% eta - learning coefficient - defines rate at which weights change
% alph - momentum coefficient - defines rate at which momentum updates
% maxit - maximum number of training iterations
% tol - RMSE error tolerance threshold
% hid - vector of number of nodes in hidden layer(s) - length(hid)
%       determines how many
% r - random number seed (for testing network) - IF r=-1: NO SEEDING

% Network - MATLAB structure holding relevant data for a trained BPNN.
%           Includes network weights (W & V) and training & testing RMS
%           (RMS & validRMS).

%% PARAMETERIZATION (Allow network to run for various input parameterizations)
switch nargin
%     Run for general case using 3 in and out data values. Used to prove
%     network is working.
    case 0
in=[-.2 .1 .3 -.4;.6 -.1 .7 -.5;.8 .1 -.6  0];  % training input data
out=[.4 .6 .5 .7;.1 .3 .2 .9;.7 .1 .2 .1];      % training output data

eta=.5;     % learning coef
alph=.9;    % momentum coef
maxit=1000; % maximum # iterations
tol=.01;    % RMSE tolerance (breaks iterative updating once reached)
hid=3;      % # hidden nodes
r=3;        % rng seed
%     Run for general case where no training parameters are provided
    case 2
eta=.5;     % learning coef
alph=.9;    % momentum coef
maxit=1000; % maximum # iterations
tol=.01;    % RMSE tolerance (breaks iterative updating once reached)
hid=3;      % # hidden nodes
r=3;        % rng seed
%     Run for case when no random number is specified.
    case 7
r=3;        % rng seed
%     Run when all parameters are specified. (This is how it should run normally)
    case 8
            % do nothing
    otherwise
        errordlg({'runloopBPNNvalidate() requires:';'     -0';'     -2 (in,out)';'     -7
(in,out,eta,alph,maxit,tol,hid)';'parameters to run correctly'})
end

bestW=0;
%% NODE SIZING
% input node size
i=size(in,2);
% hidden node size
j=hid;
% output node size
k=size(out,2);

%% Randomly select training and validation sets from data

% NOTE: Random seed is used to ensure that training is done correctly, to
% identify a best set of weights, vary the random generator.
if r~=-1
rng(r)                                          % Ensures same random idx
end
%   Creates a randomized list of index values corresponding to the # of
%   data points within a dataset. This is divided up into a training and
%   validation section in order to train and validate on randomized
%   samples.
randIdx=randperm(size(out,1));                  % Creates random idx

trainIdx=randIdx(1:floor(length(out).*2./3));   % Uses first 2/3 for train
validIdx=randIdx(length(trainIdx)+1:end);       % Uses last 1/3 for validation
```

```
% Stores traning input and output sets
trainIn=in(trainIdx,:);
trainOut=out(trainIdx,:);

% Stores validationg input and output sets
validIn=in(validIdx,:);
validOut=out(validIdx,:);

%% GENERATE RANDOM WEIGHTS
% % rng(r) seeds random number generator to generate the same numbers
if r~=-1
rng(r)
end
W=2*(rand(i,j)-.5);

if r~=-1
rng(r)
end
V=2*(rand(j,k)-.5);
%% PREALLOCATION

% RMS vector updating BPNN until RMS reaches a specified RMS threshold
RMS=ones(1,maxit);
        e=1./(1+exp(-trainIn*W));
        % hidden to output layer using squashing function:
        f=1./(1+exp(-e*V));
        % a=s*V;
        res=f;
RMS(1)=sqrt(sum(sum((trainOut-res).^2))./(size(res,1).*size(res,2)));

% RMS vector for cross checking for generalization instead of memorization
validRMS=size(RMS);
        c=1./(1+exp(-validIn*W));
        % hidden to output layer using squashing function:
        d=1./(1+exp(-c*V));
        % a=s*V;
        calcValOut=d;
validRMS(1)=sqrt(sum(sum((validOut-calcValOut).^2))./(size(calcValOut,1).*size(calcValOut,2)));

% count used in updating RMS size before breaking from training loop
count=0;

% instantiates deltaWeight vectors (1 space bigger than the number of
% iterations it can run in order to count the first space as a 0 for 1st
% iteration)
vjk=zeros(size(V,1),size(V,2),maxit+1);
wij=zeros(size(W,1),size(W,2),maxit+1);

% holds weight updates from previous iteration for each input set (row of
% in vector). Used in 'momentum' influenced weight updating:
%   W=W+wij+alph*wold
wold=zeros(size(W,1),size(W,2));
vold=zeros(size(V,1),size(V,2));
%% Start BPNN Training (runs for maxIt iterations or until validRMS=tol)
for x=1:length(RMS)
    count=count+1;
    for y=1:size(trainIn,1)
%% FORWARD PROPOGATION
        % input to hidden layer using squashing function:
        s=1./(1+exp(-trainIn(y,:)*W));
        % hidden to output layer using squashing function:
        a=1./(1+exp(-s*V));
        % a=s*V;
%% BACK ERROR
        % Error between calculated and real output:
        %   Difference of (actual-calculated)*(derivative squashing function)
        dk=(out(y,:)-a).*a.*(1-a);

        % Weight updates (w calculated as an avg of error between all training
        % sets with momentum)
        %   Momentum: The current change in weight is given 'momentum' to
        %   remove error by adding some fraction of the previous iteration's
        %   change in weight:
        %       (momentum weight change)=(current wt chg)+(#<1)*(prev wt chg)
        %   for hidden to output layer:
        vjk=eta.*s'*dk+alph.*vold;

        % Apply weight change to weight matrices
        V=V+vjk;

        % store weight changes for momentum update at next iteration for
        % current input dataset defined by y
        vold=vjk;
%% Repeat for input to hidden layer matrix
        dj=(dk*V').*s.*(1-s);
        wij=eta.*in(y,:)'*dj+alph.*wold;
        W=W+wij;
        wold=wij;

    end
%% RMS ERROR
        % calculate current network output:
        g=1./(1+exp(-trainIn*W));
        % hidden to output layer using squashing function:
        h=1./(1+exp(-g*V));
        % a=s*V;
        res=h;

        % calculate validation RMS:
        c=1./(1+exp(-validIn*W));
        % hidden to output layer using squashing function:
        d=1./(1+exp(-c*V));
        % a=s*V;
        calcValOut=d;

    % Root Mean Square Error:
    RMS(x+1)=sqrt(sum(sum((trainOut-res).^2))./(size(res,1).*size(res,2)));
    validRMS(x+1)=sqrt(sum(sum((validOut-
calcValOut).^2))./(size(calcValOut,1).*size(calcValOut,2)));

    % if validRMS is at its lowest, weights are stored
    if validRMS(x+1)==min(validRMS)
        bestW=W;
        bestV=V;
        bestIt=x+1;
    end

    % if error is within predefined RMS tolerance, tol, break loop & store
    % RMS values only for effective part of preallocated RMS vector
    if validRMS(x+1)<=tol
        RMS=RMS(1:count+1);
        validRMS=validRMS(1:count+1);
        break
    end

end
%% Relevant data storage

% Store best weights as end weights if last iteration holds best training
% weights (Most cases this is true)
if bestW==0
    bestW=W;
    bestV=V;
    bestIt=length(RMS);
end
% Store relevant BPNN data in Network
Network.W=W;
Network.V=V;
Network.bestW=bestW;
Network.bestV=bestV;
Network.bestIt=bestIt;
Network.RMS=RMS;
Network.validRMS=validRMS;
Network.trainIdx=trainIdx;
Network.validIdx=validIdx;
```

unNormalize()

```
function [in,out,netOut]=unNormalize(normIn,minIn,maxIn,normOut,normNetOut,minOut,maxOut)
% Luke Detwiler - Aug 2013

% Un-normalizes data matrices normIn, normOut, and normNetOut using
% corresponding minimum and maximum data values stored from data
% normalization using the normalize() function. The primary focus of this
% function is to normalize the predictions from BPNN training, though the
% original input and output data vectors are also re-normalized.

% normIn - normalized input data matrix (data point x parameter)
% minIn - minimum input data value vector (1 x parameter)
% maxIn - maximum input data value vector (1 x parameter)
% normOut - normalized output data matrix (data point x parameter)
% normNetOut - normalized BPNN predicted out values (size(out))
% minOut - minimum output data value vector (1 x parameter)
% maxOut - maximum output data value vector (1 x parameter)

% in - un-normalized input data matrix (data point x parameter)
% out - un-normalized output data matrix (data point x parameter)
% netOut - un-normalized BPNN predicted out values (size(out))

%% Un-normalize input data
for a=1:size(normIn,2)
%     Use input minimum and maximums to perform linear un-normalization
    normIn(:,a)=normIn(:,a)*(maxIn(1,a)-minIn(1,a))+minIn(1,a);
end

%% Un-normalize output & predicted data (same as above)
for b=1:size(normOut,2)
    normOut(:,b)=normOut(:,b)*(maxOut(1,b)-minOut(1,b))+minOut(1,b);
    normNetOut(:,b)=normNetOut(:,b)*(maxOut(1,b)-minOut(1,b))+minOut(1,b);
end
```

Driver:

Directory1to1withErrorLines.m

```
% Directory1to1withErrorLines.m - 1/24/2014

% Runs BPNN training for all datasets in a given directory. Each dataset is
% trained for the hidden layer sizings specified by the n vector. For each
% network training, this script creates 1 to 1 plots of data with 10/20/30%
% error lines and RMS training plots, and stores all necessary data
clear all, close all, clc

%% Prepare for Training
% Define hidden node sizings to train for
n=3:20;     % train each BPNN at 3 and 20 nodes

% Ask user for necessary directory information
Dir=uigetdir([],'Where is your code located?');
Dir2=uigetdir([],'Where are your soil datasets?');
cd(Dir2)
datasets=dir;
plotsave=uigetdir([],'Where is the master folder where you would like images & data stored?');


%% Begin Training
% Iterate through each dataset in the Dir2 directory.
for dataset=3:length(datasets)
    %% Normalize data

    % normalize data between 0 and 1, storing mins and maxs for future
    % 'unnormalization'
    %% Train BPNN w/ Validation
    % trains a version of the BPNN whose error convergence is entirely based on
    % the convergence of a validation error set. runloopBPNNvalidate stores
    % weight matrices W,V the index of the best validation error, RMS values
    % for both training and validation RMS, and the indices of the sub-datasets
    % (training and validation) for the location of these values within the
    % total data vector (in).
    for node=n
        %%
        cd(Dir2)
        load(datasets(dataset).name)
        cd(Dir)
%           normalize data:
        [TrainingPatterns,minIn,maxIn,Targets,minOut,maxOut]=normalize(in,out);
%           train BPNN:
        Network = backprop(TrainingPatterns,Targets,.5,.9,10000,.01,node,1);
%           un-normalize data:

[TrainingPatterns,Targets,Predictions]=unNormalize(TrainingPatterns,minIn,maxIn,Targets,Network.p
redictions,minOut,maxOut);

        %% Compare Predicted Outputs to Known Outputs on 1 to 1 plot
        figure(1)
%         plot 1 to 1 comparisons
        plot(Targets(Network.validIdx),Predictions(Network.validIdx),'ro')
        hold on
%         plot training data
        plot(Targets(Network.trainIdx),Predictions(Network.trainIdx),'b^')
%         specify square axes, apply error lines, & add labels and titles:
        ax=[5 35];
        set(gca,'Xlim',ax,'Ylim',ax)
        plot(get(gca,'Xlim'),get(gca,'Ylim'),'k--')


plot(1.1*get(gca,'Xlim'),get(gca,'Ylim'),'g:',1.2*get(gca,'Xlim'),get(gca,'Ylim'),'c:',1.3*get(gc
a,'Xlim'),get(gca,'Ylim'),'m:',get(gca,'Xlim'),1.1*get(gca,'Ylim'),'g:',get(gca,'Xlim'),1.2*get(g
ca,'Ylim'),'c:',get(gca,'Xlim'),1.3*get(gca,'Ylim'),'m:')
        axis square
        xlabel('Known \phi_r'' (deg)')
        ylabel('BPNN approximated \phi_r'' (deg)')

        ttlStr2=sprintf('%d node BPNN - RMSE: %.3',node,Network.validRMS(end));
        title({strcat(datasets(dataset).name(1:end-4),' dataset
n=',num2str(length(Targets))),ttlStr2,'1 to 1 plot for validation values of \phi_r'})
        valid=strcat('Validation data - n=',num2str(length(Network.validIdx)));
        train=strcat('Training data - n=',num2str(length(Network.trainIdx)));
        legend(valid,train,'1 to 1 line','10% error line','20% error line','30% error
line','Location','SouthEast')
        %% Plot training and testing RMS
        figure(2)
        p=plot(Network.RMS,'b');
        set(p,'LineWidth',2)
        hold on
        q=plot(Network.validRMS,'r--');
        set(q,'LineWidth',2)
        legend('RMS of training data','RMS of validation data')
        xlabel('# of iterations')
        ylabel('Root Mean Square Error')
        title({strcat(datasets(dataset).name(1:end-4),' dataset
n=',num2str(length(Network.validIdx))),strcat(num2str(node),' node BPNN'),'Prediction Error
plot'})

        %% Plot Results by CF type

%         repeat 1 to 1 plot for specified CF ranges
        CF=TrainingPatterns(:,4);
        CFlow=find(CF<=23);
        CFhig=find(CF>48);
        a=zeros(size(CF));
        a(CFlow)=1;
        a(CFhig)=1;
        CFmid=find(a==0);

        figure(3)
        plot(Targets(CFlow),Predictions(CFlow),'ro')
        hold on
        plot(Targets(CFmid),Predictions(CFmid),'b^')
        plot(Targets(CFhig),Predictions(CFhig),'ks')
        ax=[5 35];
        set(gca,'Xlim',ax,'Ylim',ax)
        plot(get(gca,'Xlim'),get(gca,'Ylim'),'k--')

plot(1.1*get(gca,'Xlim'),get(gca,'Ylim'),'g:',1.2*get(gca,'Xlim'),get(gca,'Ylim'),'c:',1.3*get(gc
a,'Xlim'),get(gca,'Ylim'),'m:',get(gca,'Xlim'),1.1*get(gca,'Ylim'),'g:',get(gca,'Xlim'),1.2*get(g
ca,'Ylim'),'c:',get(gca,'Xlim'),1.3*get(gca,'Ylim'),'m:')
        axis square
        xlabel('Known \phi_r'' (deg)')
        ylabel('BPNN approximated \phi_r'' (deg)')

        ttlStr2=sprintf('%d node BPNN - RMSE: %.3f',node,Network.validRMS(end));
        title({strcat(datasets(dataset).name(1:end-4),' dataset
n=',num2str(length(Targets))),ttlStr2,'1 to 1 plot for validation values of \phi_r'})
        CFlowStr=strcat('CF<=23 - n=',num2str(length(Network.validIdx)));
        CFmidStr=strcat('23<CF<=48 - n=',num2str(length(Network.trainIdx)));
        CFhigStr=strcat('CF>48 - n=',num2str(length(Network.trainIdx)));
        legend(CFlowStr,CFmidStr,CFhigStr,'1 to 1 line','10% error line','20% error line','30%
error line','Location','SouthEast')
%% Save data and figures
%       Save plots
        plotfile1to1=strcat(plotsave,'\',datasets(dataset).name(1:end-
4),'\',datasets(dataset).name(1:end-4),num2str(node),'node1to1');
        plotfileRMS=strcat(plotsave,'\',datasets(dataset).name(1:end-
4),'\',datasets(dataset).name(1:end-4),num2str(node),'nodeRMS');
        plotfileCF=strcat(plotsave,'\',datasets(dataset).name(1:end-
4),'\',datasets(dataset).name(1:end-4),num2str(node),'CFcomparisons');
        saveas(figure(1),plotfile1to1,'fig')
        saveas(figure(2),plotfileRMS,'fig')
        saveas(figure(3),plotfileCF,'fig')
%       Save data
        savefile=strcat(plotsave,'\',datasets(dataset).name(1:end-
4),'\',datasets(dataset).name(1:end-4),num2str(node),'data');
        save(savefile,'TrainingPatterns','Targets','Predictions','Network')
%       Clear workspace for next BPNN training
        close all
        clearvars -except Dir Dir2 dataset datasets plotsave
    end

end
```

APPENDIX B – Stark & Hussain 2013 Interpolation Code

Function:

```
function data=StarkEidPredictions(data)
% function data=StarkEidPredictions(data) 1/30/2014 - Luke Detwiler
%   This function predicts secant residual friction angles for a soil
%   dataset that contains data in the form of a structure (data) that
%   includes correlation values: 1-3 (CorrVal) that correspond to CF ranges
%   specified in Stark and Eid's relations for the data provided (2013).
%   For each CF range, 4 empirical correlations related to the Norm
%   loadings applied in a ring shear test are given for a soil's secant
%   residual friction angle as a function of the soil's LL value. For Norm
%   loadings determined to be betweeen ranges, predicted values are
%   linearly interpolated between the two nearest equations.

%   IT SHOULD BE NOTED that these range specifications have been modified
%   from their original ranges (CF<=20,25<=CF<=45,CF>=50) to fit a
%   continuous range (CF<=23,23<CF<=48,CF>48).

% Loop through length of data structure (number of datasets).
for i=1:length(data)
    % Loop through number of subsets making up each dataset.
    for j=1:size(data(i).set,1)
        % Extract and store necessary soil characteristics for calculation.
        CFtype=data(i).CorrVal(j);
        LL=data(i).set(j,2);
        Norm=data(i).set(j,1);
        % Calculate secant residual friction angle considering the CF type,
        % Norm value, and LL value (in that order).
        switch CFtype
            case 1
                % Ensure that soil data is within the LL range specified by
                % Stark & Eid.
                if LL>=30&&LL<80
                    % If Norm lower than lowest specified loading, use that
                    % correlation.
                    if Norm<=50
                        data(i).pred(j)=39.71-0.29*LL+.000663*LL^2;
                    % If between ranges, interpolate.
                    elseif Norm>50&&Norm<=100
                        Low=39.71-0.29*LL+.000663*LL^2;
                        High=39.41-0.298*LL+.000681*LL^2;
                        data(i).pred(j)=Low+(Norm-50)/(100-50)*(High-Low);
                    elseif Norm>100&&Norm<=400
                        Low=39.41-0.298*LL+.000681*LL^2;
                        High=40.24-0.375*LL+.00136*LL^2;
                        data(i).pred(j)=Low+(Norm-100)/(400-100)*(High-Low);
                    elseif Norm>400&&Norm<=700
                        Low=40.24-0.375*LL+.00136*LL^2;
                        High=40.34-0.412*LL+.001683*LL^2;
                        data(i).pred(j)=Low+(Norm-400)/(700-400)*(High-Low);
                    % If Norm higher than highest specified loading, use
                    % that correlation.
                    else
                        data(i).pred(j)=40.34-0.412*LL+.001683*LL^2;
                    end
                % If not within the LL range specified for a given CF
                % range, store prediction as a nan to suppress output.
                else
                    data(i).pred(j)=nan;
                end
            case 2
                % See case 1 (lines 28-58) for comments
                if LL>=30&&LL<130
                    if Norm<=50
                        data(i).pred(j)=31.4-.00679*LL-.003616*LL^2+.00001864*LL^3;
                    elseif Norm>50&&Norm<=100
                        Low=31.4-.00679*LL-.003616*LL^2-.00001864*LL^3;
                        High=29.8-.0003627*LL-.003584*LL^2+.00001854*LL^3;
                        data(i).pred(j)=Low+(Norm-50)/(100-50)*(High-Low);
                    elseif Norm>100&&Norm<=400
                        Low=29.8-.0003627*LL-.003584*LL^2+.00001854*LL^3;
                        High=28.4-.05622*LL-.002952*LL^2+.00001721*LL^3;
                        data(i).pred(j)=Low+(Norm-100)/(400-100)*(High-Low);
                    elseif Norm>400&&Norm<=700
                        Low=28.4-.05622*LL-.002952*LL^2+.00001721*LL^3;
                        High=28.05-.2083*LL-.0008183*LL^2+.000009372*LL^3;
                        data(i).pred(j)=Low+(Norm-400)/(700-400)*(High-Low);
                    else
                        data(i).pred(j)=28.05-.2083*LL-.0008183*LL^2+.000009372*LL^3;
                    end
                else
                    data(i).pred(j)=nan;
                end
            case 3
                if LL>=30&&LL<120
                    if Norm<=50
                        data(i).pred(j)=33.5-.31*LL+.00039*LL^2+.0000044*LL^3;
                    elseif Norm>50&&Norm<=100
                        Low=33.5-.31*LL+.00039*LL^2+.0000044*LL^3;
                        High=30.7-.2504*LL-.00042053*LL^2+.0000080479*LL^3;
                        data(i).pred(j)=Low+(Norm-50)/(100-50)*(High-Low);
                    elseif Norm>100&&Norm<=400
                        Low=30.7-.2504*LL-.00042053*LL^2+.0000080479*LL^3;
                        High=29.42-.2621*LL-.0004011*LL^2+.000008718*LL^3;
                        data(i).pred(j)=Low+(Norm-100)/(400-100)*(High-Low);
                    elseif Norm>400&&Norm<=700
                        Low=29.42-.2621*LL-.0004011*LL^2+.000008718*LL^3;
                        High=27.7-.3233*LL+.0002896*LL^2+.0000071131*LL^3;
                        data(i).pred(j)=Low+(Norm-400)/(700-400)*(High-Low);
                    else
                        data(i).pred(j)=27.7-.3233*LL+.0002896*LL^2+.0000071131*LL^3;
                    end
                elseif LL<=120&&LL<=300
                    if Norm<=50
                        data(i).pred(j)=12.03-.0215*LL;
                    elseif Norm>50&&Norm<=100
                        Low=12.03-.0215*LL;
                        High=10.64-.0183*LL;
                        data(i).pred(j)=Low+(Norm-50)/(100-50)*(High-Low);
                    elseif Norm>100&&Norm<=400
                        Low=10.64-.0183*LL;
                        High=8.32-.0114*LL;
                        data(i).pred(j)=Low+(Norm-100)/(400-100)*(High-Low);
                    elseif Norm>400&&Norm<=700
                        Low=8.32-.0114*LL;
                        High=5.84-.0048*LL;
                        data(i).pred(j)=Low+(Norm-400)/(700-400)*(High-Low);
                    else
                        data(i).pred(j)=5.84-.0048*LL;
                    end
                else
                    data(i).pred(j)=nan;
                end
        end
    end
end
```

Driver:

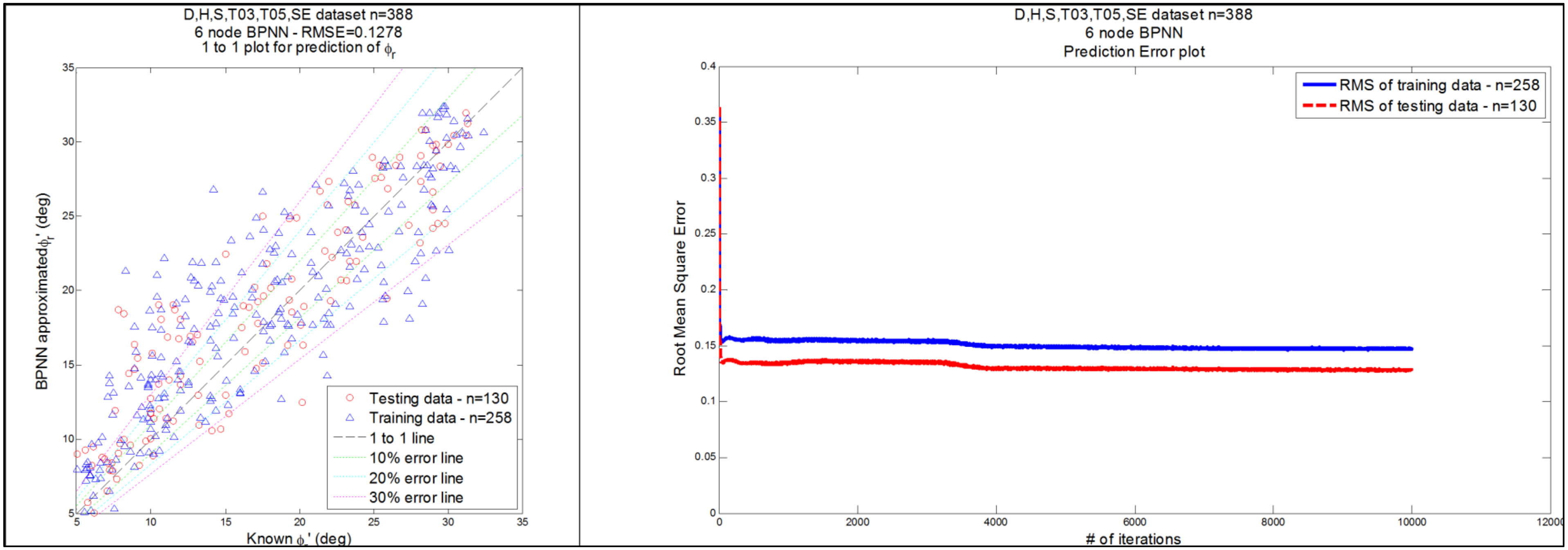
starkHussainAnalysis1to1Plots.m

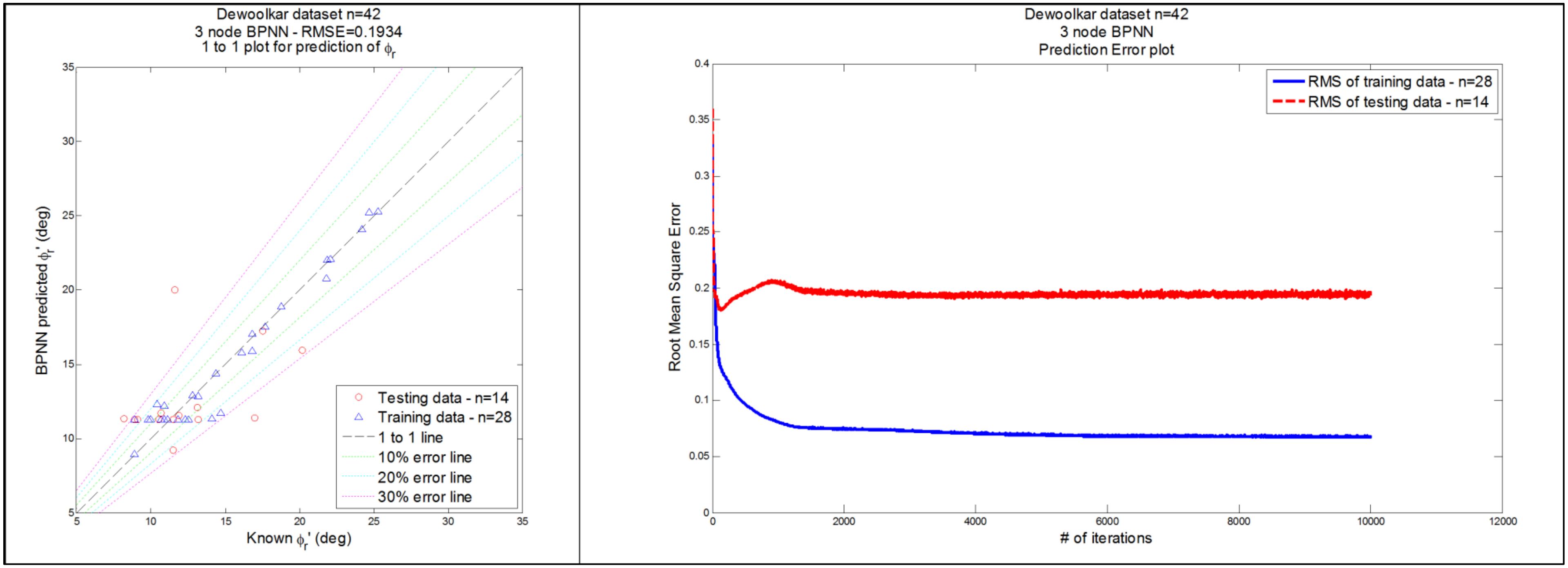
```
% starkHussainAnalysis1to1Plots.m  - 1/26/2014 - Luke Detwiler
%
% This code manipulates soil data for the prediction of secant residual
% friction angle using equations provided in Stark and Hussain 2013.
% Instead of being plotted with phi on the y axis and LL on the x axis,
% the data is plotted to show the predicted values of phi vs the known
% values of phi in a 1 to 1 plot. This will allow for the easy comparison
% of this prediction method's performance to that of the BPNN's.

clear all
close all
load ALLDATAStarkHussainCorrelation.mat

% Create CF data indices for future predcition of friction angle.
% CorrVal=1: low CF range
% CorrVal=2: middle CF range
% CorrVal=3: high CF range
names=who();
data=struct;
for i=1:length(names)
    data(i).name=names(i);
    data(i).set=eval(names{i});
    for j=1:size(data(i).set,1)
        if round(data(i).set(j,4))<=23
            data(i).CorrVal(j)=1;
        elseif round(data(i).set(j,4))>48
            data(i).CorrVal(j)=3;
        else
            data(i).CorrVal(j)=2;
        end
    end
end

% Use StarkHussainPredictions to calculate interpolated Phi_r from LL polynomial
% equations from (Stark and Hussain, 2013):
data=starkHussainPredictions(data);

%% Plot prediction results for each dataset:
cmap=colormap(lines);
for i=1:length(data)
    corrInd1=find(data(i).CorrVal==1);
    corrInd2=find(data(i).CorrVal==2);
    corrInd3=find(data(i).CorrVal==3);
    known=eval(data(i).name{1});
    known=known(:,6);
    figure(i)
    plot(known(corrInd1),data(i).pred(corrInd1),'o','MarkerEdgeColor',cmap(i,:))
    hold on
    plot(known(corrInd2),data(i).pred(corrInd2),'^','MarkerEdgeColor',cmap(i,:))
    plot(known(corrInd3),data(i).pred(corrInd3),'s','MarkerEdgeColor',cmap(i,:))
    ax=[5 35];
    set(gca,'Xlim',ax,'Ylim',ax)
    plot(get(gca,'Xlim'),get(gca,'Ylim'),'k--')

plot(1.1*get(gca,'Xlim'),get(gca,'Ylim'),'g:',1.2*get(gca,'Xlim'),get(gca,'Ylim'),'c:',1.3*get(gc
a,'Xlim'),get(gca,'Ylim'),'m:',get(gca,'Xlim'),1.1*get(gca,'Ylim'),'g:',get(gca,'Xlim'),1.2*get(g
ca,'Ylim'),'c:',get(gca,'Xlim'),1.3*get(gca,'Ylim'),'m:')
    axis square
    xlabel('Known \phi_r (deg)')
    ylabel('Stark & Hussain predicted \phi_r (deg)')
    ntot=size(data(i).pred,2)-sum(isnan(data(i).pred));
    nall=size(data(i).pred,2);
    nlow=size(data(i).pred(corrInd1),2)-sum(isnan(data(i).pred(corrInd1)));
    nlowall=size(data(i).pred(corrInd1),2);
    nmid=size(data(i).pred(corrInd2),2)-sum(isnan(data(i).pred(corrInd2)));
    nmidall=size(data(i).pred(corrInd2),2);
    nhig=size(data(i).pred(corrInd3),2)-sum(isnan(data(i).pred(corrInd3)));
    nhigall=size(data(i).pred(corrInd3),2);
    legLow=strcat('CF<=23 - n=',num2str(nlow),'/',num2str(nlowall));
    legMid=strcat('23<CF<=48 - n=',num2str(nmid),'/',num2str(nmidall));
    legHig=strcat('CF>48 - n=',num2str(nhig),'/',num2str(nhigall));
    totaln=strcat('n=',num2str(ntot),'/',num2str(nall));
%   RMSE calculation:
    a=~isnan(data(i).pred);
    [normOut,normPred,minOut,maxOut]=normalizeStarkEid(data(i).set(a,end),data(i).pred(a)');
        sqDif=(normOut-normPred).^2;
    RMSE=sqrt(sum(sum(sqDif))/sum(a));
    RMSEstring=sprintf('RMSE: %.4f',RMSE);

    legend(legLow,legMid,legHig,'1 to 1 line','10% error line', '20% Error Line','30% Error
Line','Location','EastOutside')
    title({data(i).name{1};totaln;RMSEstring})
    data(i).CFlow=corrInd1;
    data(i).CFmid=corrInd2;
    data(i).CFhig=corrInd3;
    data(i).RMSE=RMSE;
end
% Establish maximum length for legend cell array (each smaller string needs
% to be padded to the same size by sprintf):
maxlength=60;
spacing=['%-',num2str(maxlength),'s'];
legStrs=sprintf(spacing,[]);
allOuts=[];
allPred=[];
% Plot each dataset on the same figure
for i=1:length(data)
    % Establish indices for each CF data subset
    corrInd1=find(data(i).CorrVal==1);
    corrInd2=find(data(i).CorrVal==2);
    corrInd3=find(data(i).CorrVal==3);
    known=eval(data(i).name{1});
    known=known(:,6);
    % Plot on new figure
    figure(length(data)+1)
    % Plot each data subset following the same colormap
    plot(known(corrInd1),data(i).pred(corrInd1),'o','MarkerEdgeColor',cmap(i,:))
    hold on
    plot(known(corrInd2),data(i).pred(corrInd2),'^','MarkerEdgeColor',cmap(i,:))
    plot(known(corrInd3),data(i).pred(corrInd3),'s','MarkerEdgeColor',cmap(i,:))
    % Count # of data points being used in each CF group:
    ntot(i)=size(data(i).pred,2)-sum(isnan(data(i).pred));
    nall(i)=size(data(i).pred,2);
        nlow=size(data(i).pred(corrInd1),2)-sum(isnan(data(i).pred(corrInd1)));
    nlowall=size(data(i).pred(corrInd1),2);
    nmid=size(data(i).pred(corrInd2),2)-sum(isnan(data(i).pred(corrInd2)));
    nmidall=size(data(i).pred(corrInd2),2);
    nhig=size(data(i).pred(corrInd3),2)-sum(isnan(data(i).pred(corrInd3)));
    nhigall=size(data(i).pred(corrInd3),2);
    % Store legend strings for each data subset (had a lot of trouble with
    % cell arrays of strings being interpreted by the legend() function)
    legCell=strcat(data(i).name,' - CF<=23 - n=',num2str(nlow),'/',num2str(nlowall));
    legStr=legCell{1};
    legStr=sprintf(spacing,legStr);
    if i==1
        legStrs=legStr;
    else
        legStrs=[legStrs;legStr];
    end
    legCell=strcat(data(i).name,' - 23<CF<=48 - n=',num2str(nmid),'/',num2str(nmidall));
    legStr=legCell{1};
    legStr=sprintf(spacing,legStr);
    legStrs=[legStrs;legStr];
    legCell=strcat(data(i).name,' - CF>48 - n=',num2str(nhig),'/',num2str(nhigall));
    legStr=legCell{1};
    legStr=sprintf(spacing,legStr);
    legStrs=[legStrs;legStr];
```

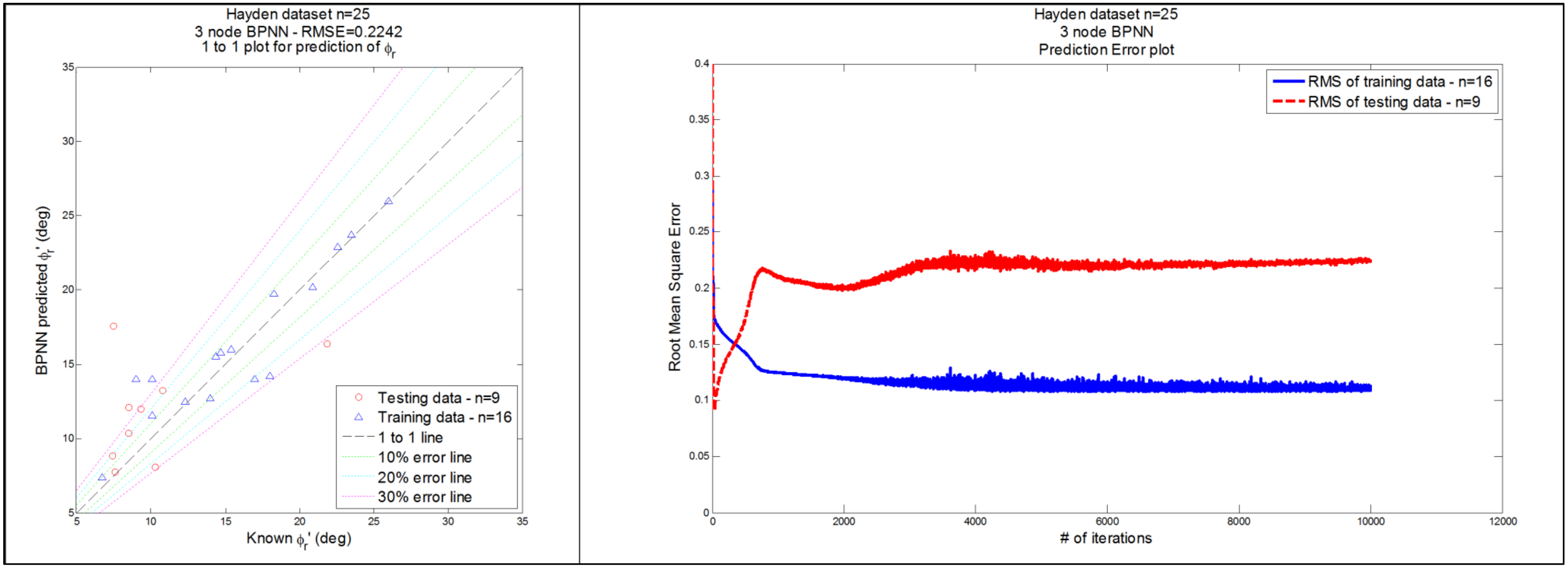
```
%   Combine all useful data for SSE calculation preparation:
    a=~isnan(data(i).pred);
        allOuts=[allOuts; data(i).set(a,end)];
        allPred=[allPred; data(i).pred(a)'];
end

%% Finish All Data Plot
ax=[5 35];
set(gca,'Xlim',ax,'Ylim',ax)
plot(get(gca,'Xlim'),get(gca,'Ylim'),'k--')
plot(1.1*get(gca,'Xlim'),get(gca,'Ylim'),'g:',1.2*get(gca,'Xlim'),get(gca,'Ylim'),'c:',1.3*get(gc
a,'Xlim'),get(gca,'Ylim'),'m:',get(gca,'Xlim'),1.1*get(gca,'Ylim'),'g:',get(gca,'Xlim'),1.2*get(g
ca,'Ylim'),'c:',get(gca,'Xlim'),1.3*get(gca,'Ylim'),'m:')
axis square
xlabel('Known \phi_r'' (deg)')
ylabel('Stark & Hussain predicted \phi_r'' (deg)')
totaln=strcat('n=',num2str(sum(ntot)),'/',num2str(sum(nall)));
legStr=sprintf(spacing,'1 to 1 line');
legStrs=[legStrs;legStr];
legStr=sprintf(spacing,'10% error line');
legStrs=[legStrs;legStr];
legStr=sprintf(spacing,'20% Error Line');
legStrs=[legStrs;legStr];
legStr=sprintf(spacing,'30% Error Line');
legStrs=[legStrs;legStr];
legend(legStrs,'Location','EastOutside')

%% RMSE calculations:
SSE=0;
n=0;
[normOut,normPred,minOut,maxOut]=normalizeStarkEid(allOuts,allPred);
RMSE=sqrt(sum((normOut-normPred).^2)/numel(allPred));
RMSEstring=sprintf('RMSE: %.4f',RMSE);
title({'All Datasets';totaln;RMSEstring})

%% Save figures and data for future use:
figHandles=findobj('Type','figure');
for x=1:length(figHandles)
    filename=get(get(get(figure(figHandles(x)),'CurrentAxes'),'Title'),'String');
    saveas(figHandles(x),filename{1},'fig')
end

save('ConnorData_2_1_14','data')
```

APPENDIX C – BPNN Optimization

No SF

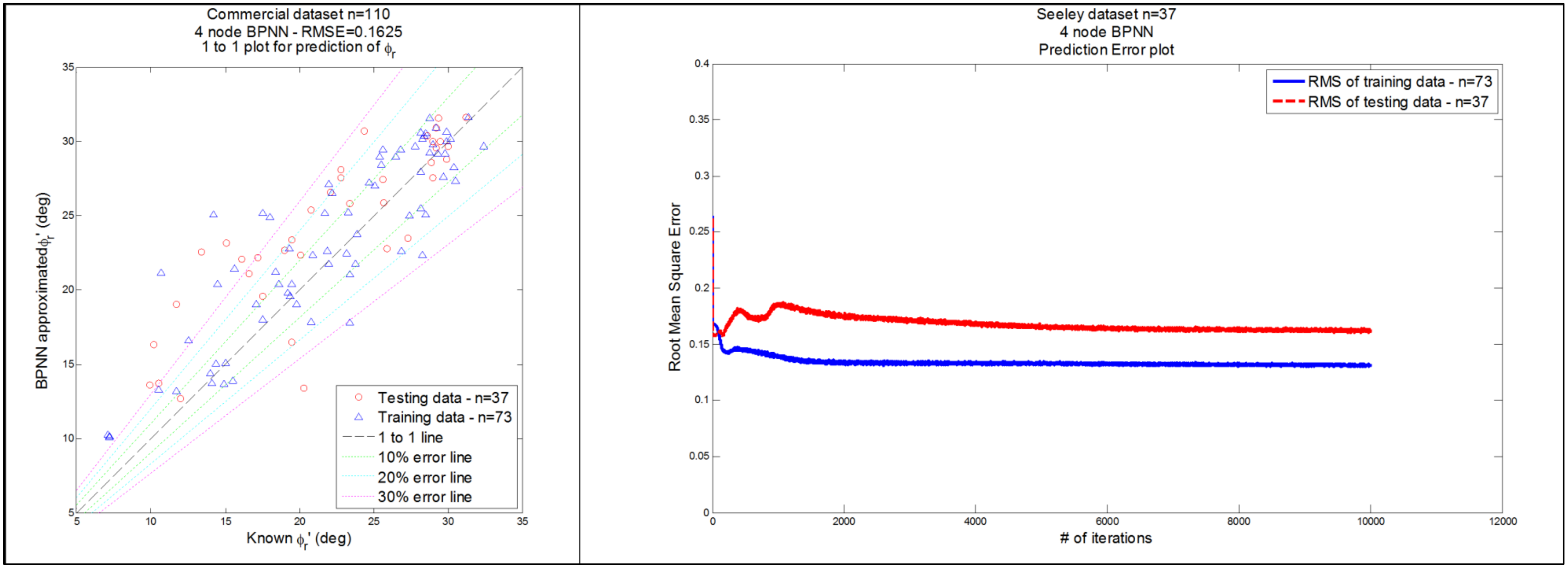


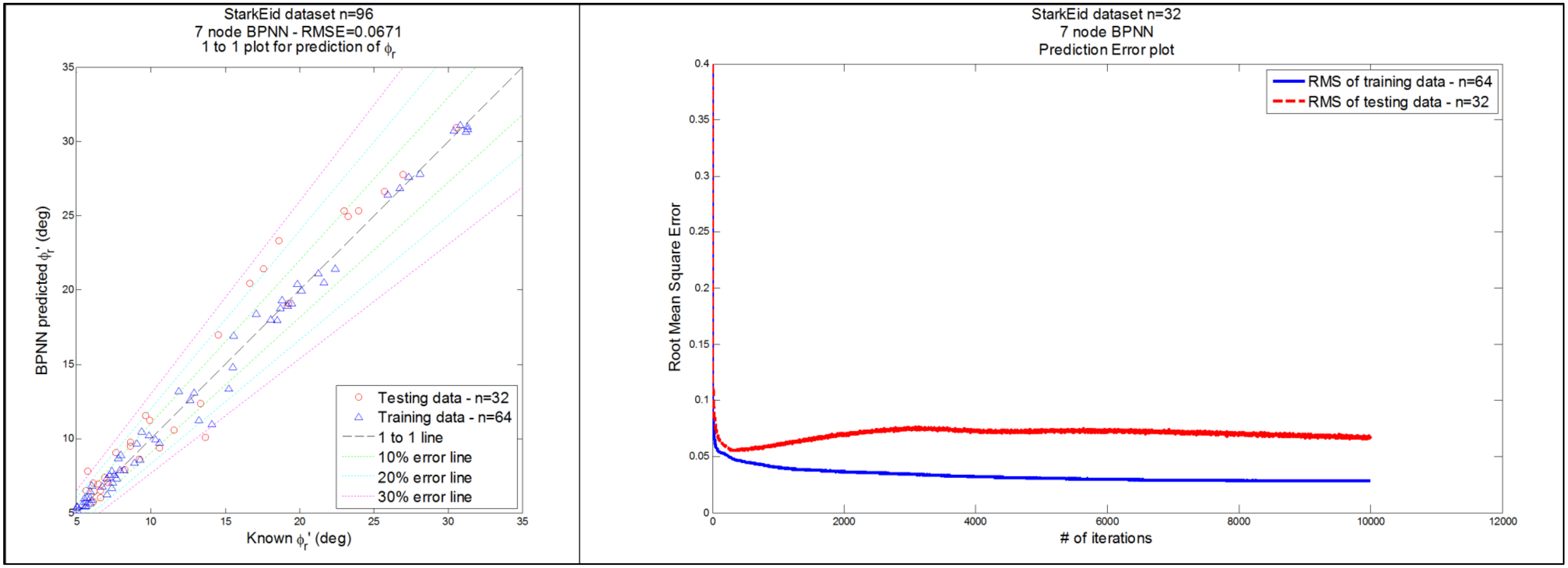

Dewoolkar dataset n=42
3 node BPNN - RMSE=0.1934
1 to 1 plot for prediction of ϕ_r
BPNN predicted ϕ_r' (deg)
Known ϕ_r' (deg)
5
10
15
20
25
30
35
Testing data - n=14
Training data - n=28
1 to 1 line
10% error line
20% error line
30% error line
Dewoolkar dataset n=42
3 node BPNN
Prediction Error plot
Root Mean Square Error
# of iterations
0
0.05
0.1
0.15
0.2
0.25
0.3
0.35
0.4
0
2000
4000
6000
8000
10000
12000
RMS of training data - n=28
RMS of testing data - n=14

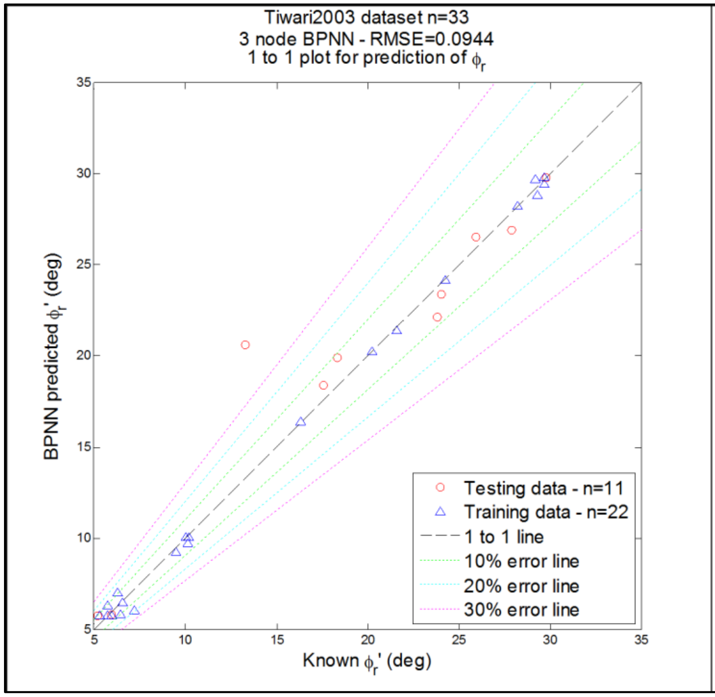


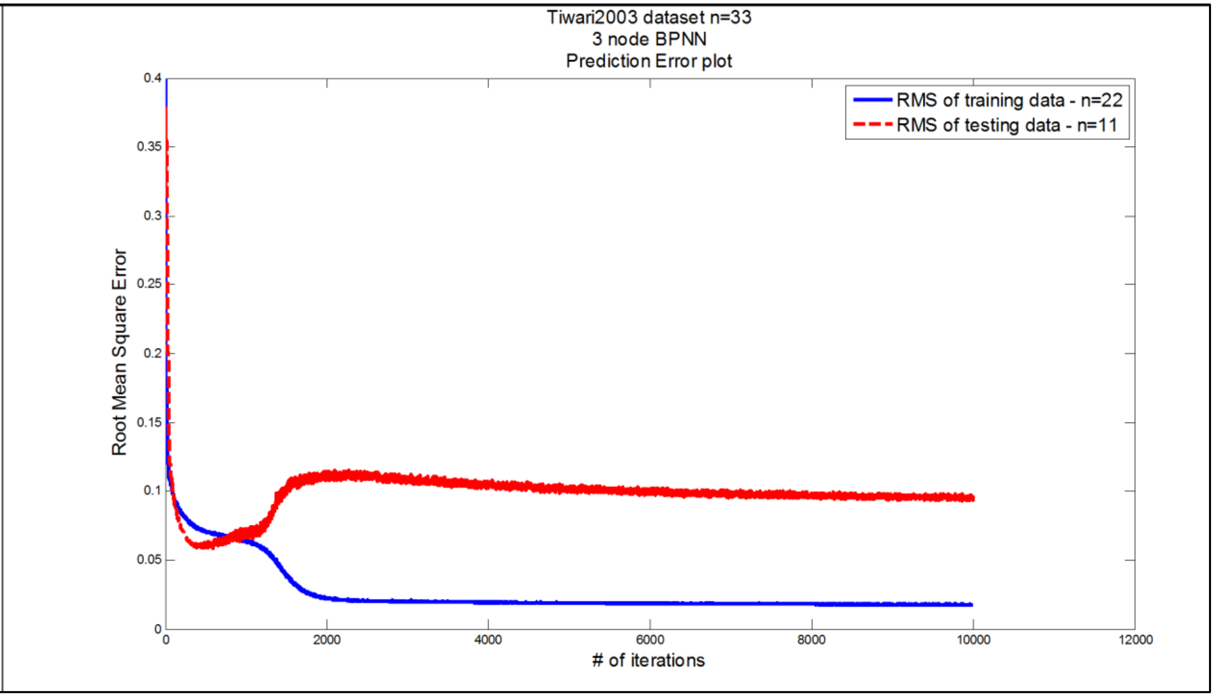
Hayden dataset n=25
3 node BPNN - RMSE=0.2242
1 to 1 plot for prediction of ϕ_r

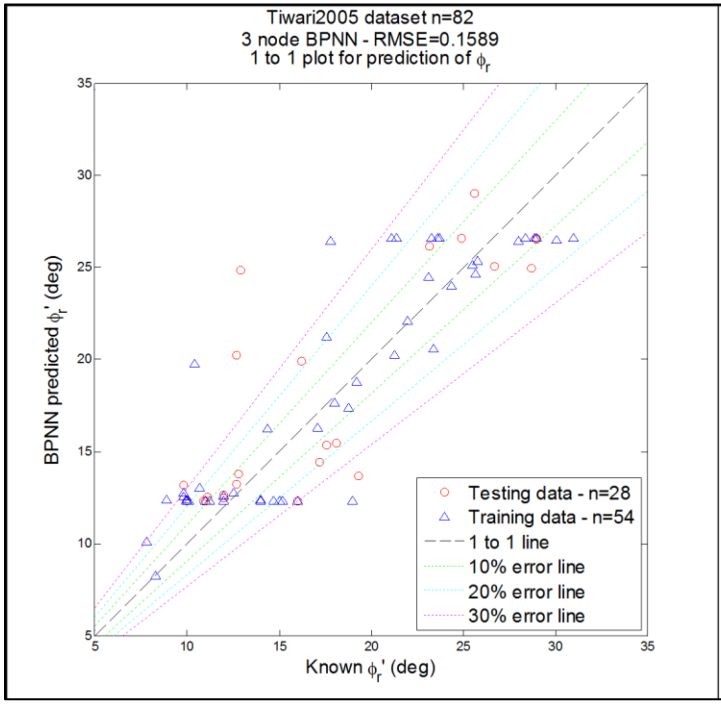
Hayden dataset n=25
3 node BPNN
Prediction Error plot

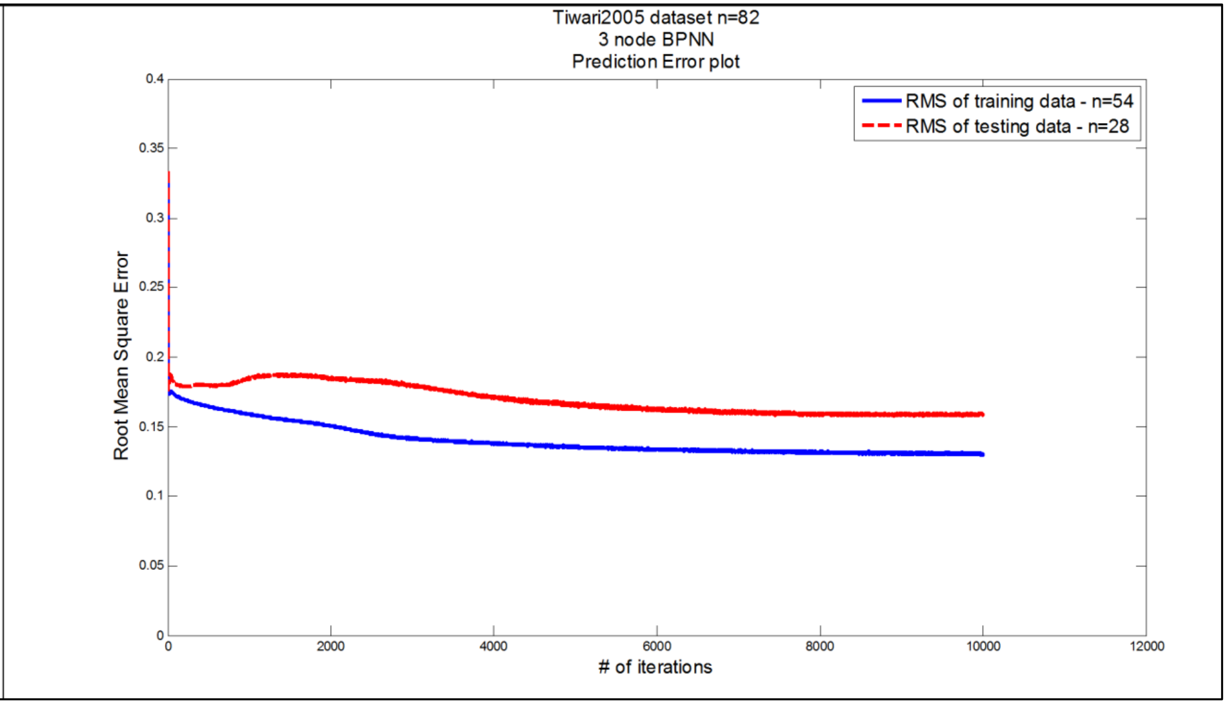

Commercial dataset n=110
4 node BPNN - RMSE=0.1625
1 to 1 plot for prediction of ϕ_r
BPNN approximatedϕ_r' (deg)
Known ϕ_r' (deg)
5
10
15
20
25
30
35
Testing data - n=37
Training data - n=73
1 to 1 line
10% error line
20% error line
30% error line
Seeley dataset n=37
4 node BPNN
Prediction Error plot
Root Mean Square Error
# of iterations
0
0.05
0.1
0.15
0.2
0.25
0.3
0.35
0.4
2000
4000
6000
8000
10000
12000
RMS of training data - n=73
RMS of testing data - n=37


StarkEid dataset n=96
7 node BPNN - RMSE=0.0671
1 to 1 plot for prediction of ϕ_r
BPNN predicted ϕ_r' (deg)
Known ϕ_r' (deg)
35
30
25
20
15
10
5
5
10
15
20
25
30
35
○ Testing data - n=32
△ Training data - n=64
1 to 1 line
10% error line
20% error line
30% error line
StarkEid dataset n=32
7 node BPNN
Prediction Error plot
Root Mean Square Error
# of iterations
0.4
0.35
0.3
0.25
0.2
0.15
0.1
0.05
0
0
2000
4000
6000
8000
10000
12000
RMS of training data - n=64
RMS of testing data - n=32

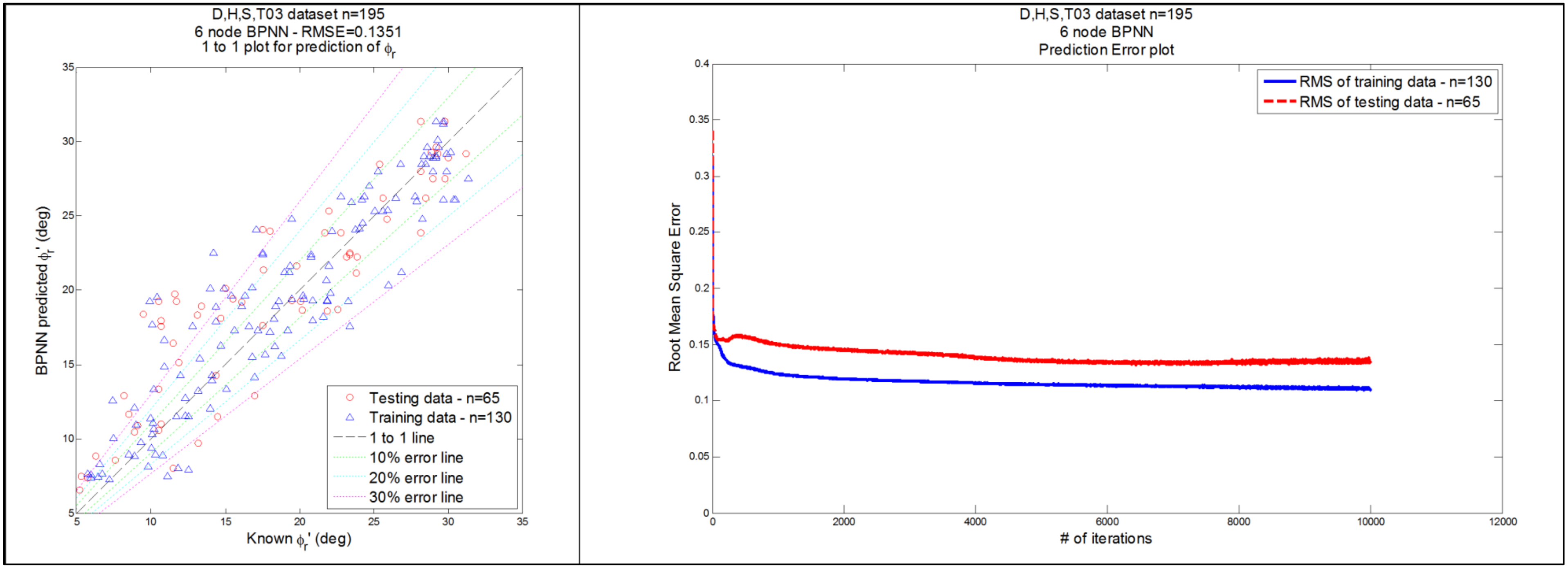

Tiwari2003 dataset n=33
3 node BPNN - RMSE=0.0944
1 to 1 plot for prediction of ϕ_r
BPNN predicted ϕ_r' (deg)
Known ϕ_r' (deg)
5
10
15
20
25
30
35
Testing data - n=11
Training data - n=22
1 to 1 line
10% error line
20% error line
30% error line

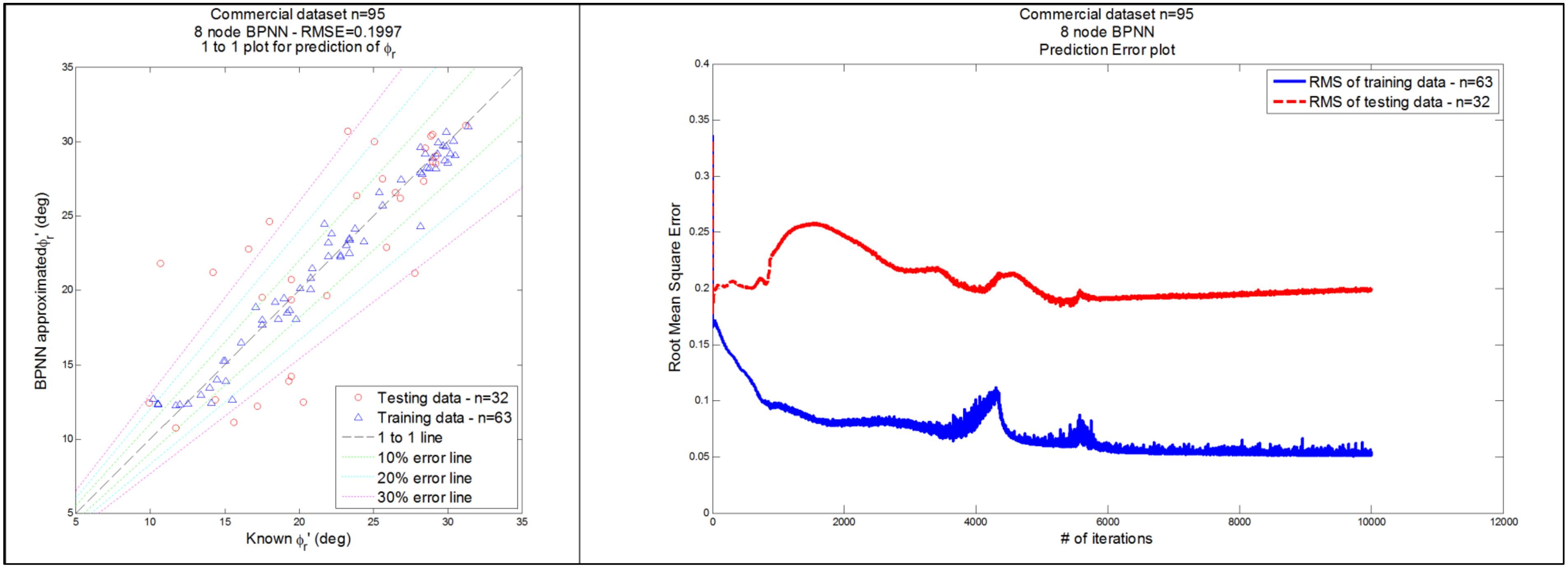

Tiwari2003 dataset n=33
3 node BPNN
Prediction Error plot
Root Mean Square Error
# of iterations
0
0.05
0.1
0.15
0.2
0.25
0.3
0.35
0.4
2000
4000
6000
8000
10000
12000
RMS of training data - n=22
RMS of testing data - n=11


Tiwari2005 dataset n=82
3 node BPNN - RMSE=0.1589
1 to 1 plot for prediction of ϕ_r
BPNN predicted ϕ_r' (deg)
Known ϕ_r' (deg)
35
30
25
20
15
10
5
5
10
15
20
25
30
35
○ Testing data - n=28
△ Training data - n=54
1 to 1 line
10% error line
20% error line
30% error line

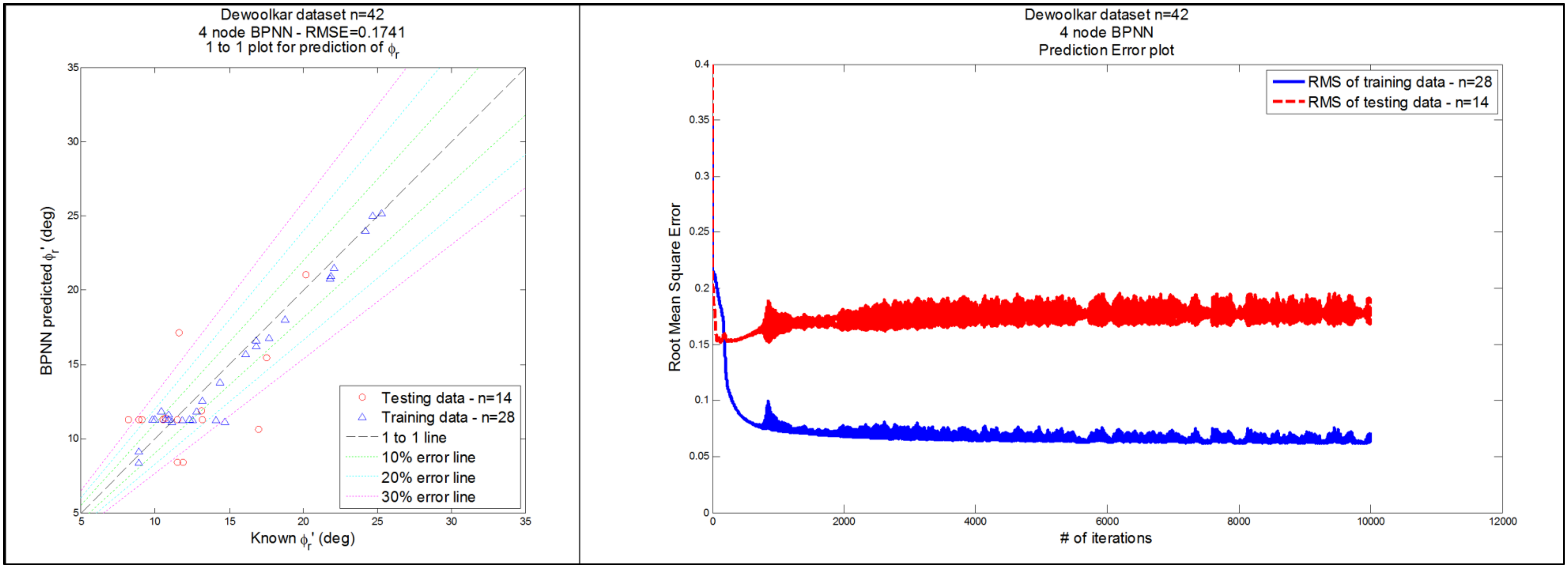

Tiwari2005 dataset n=82
3 node BPNN
Prediction Error plot
Root Mean Square Error
# of iterations
0.4
0.35
0.3
0.25
0.2
0.15
0.1
0.05
0
0
2000
4000
6000
8000
10000
12000
RMS of training data - n=54
RMS of testing data - n=28

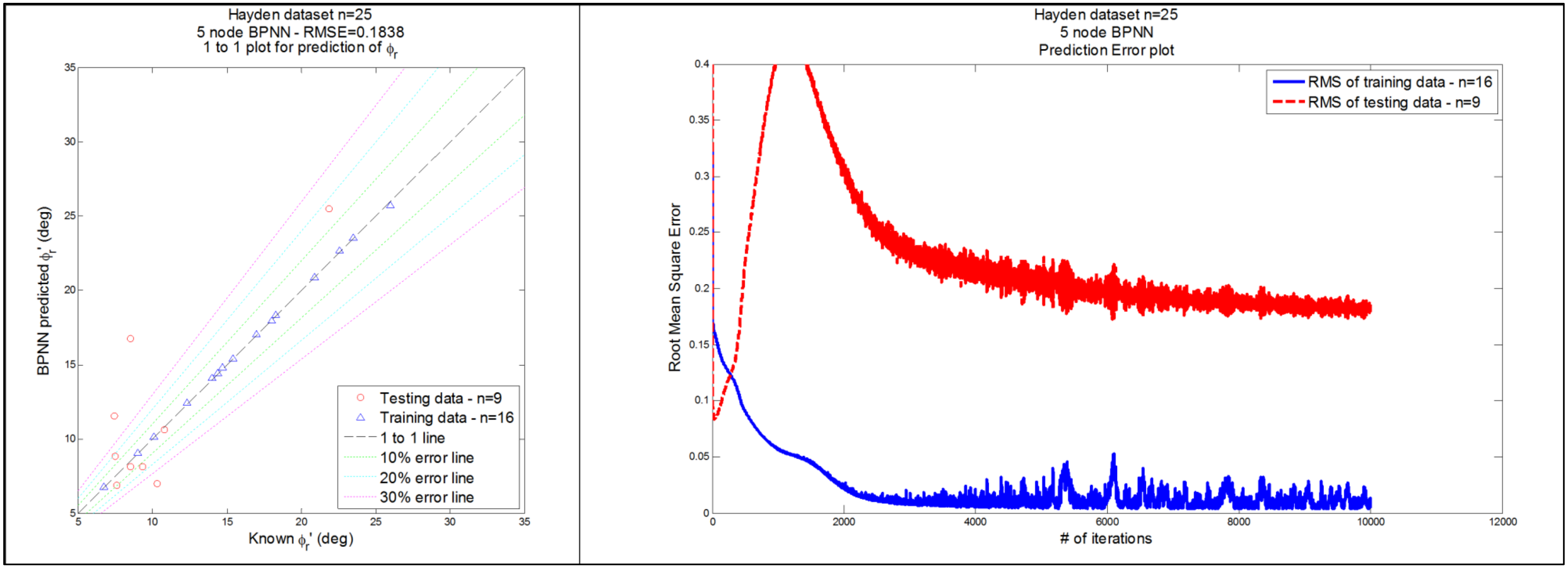
With SF

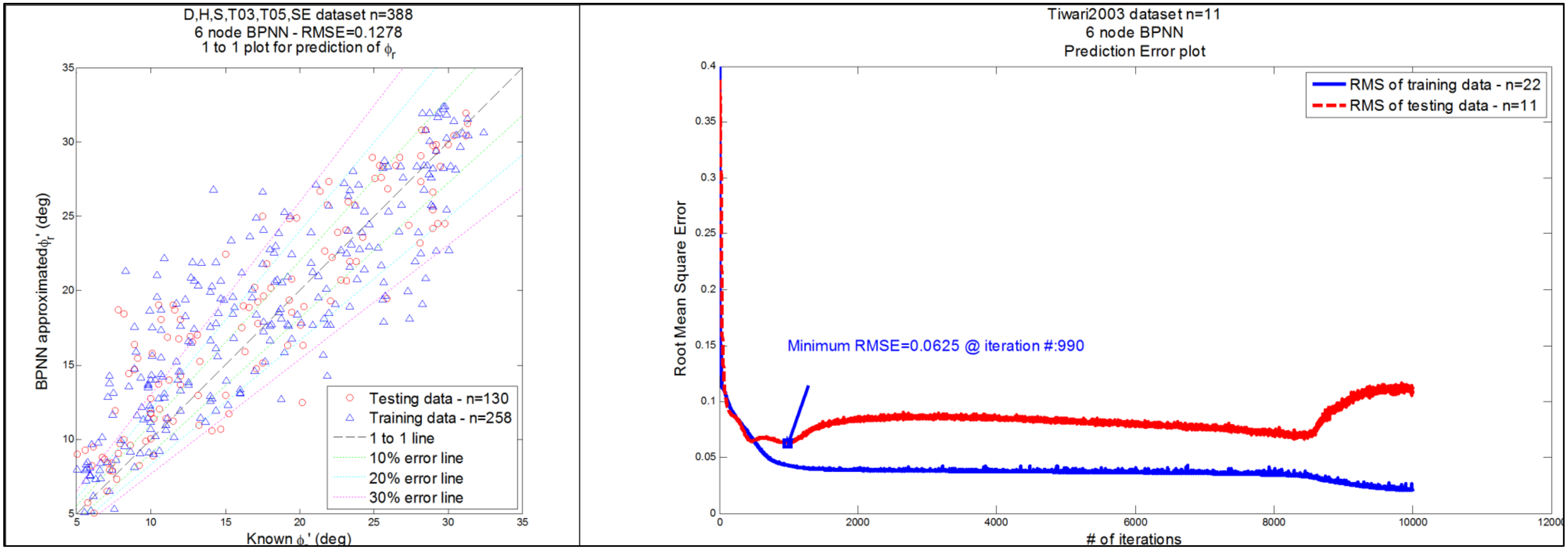

D,H,S,T03 dataset n=195
6 node BPNN - RMSE=0.1351
1 to 1 plot for prediction of ϕ_r
BPNN predicted ϕ_r' (deg)
Known ϕ_r' (deg)
5
10
15
20
25
30
35
Testing data - n=65
Training data - n=130
1 to 1 line
10% error line
20% error line
30% error line
D,H,S,T03 dataset n=195
6 node BPNN
Prediction Error plot
Root Mean Square Error
# of iterations
0
0.05
0.1
0.15
0.2
0.25
0.3
0.35
0.4
2000
4000
6000
8000
10000
12000
RMS of training data - n=130
RMS of testing data - n=65



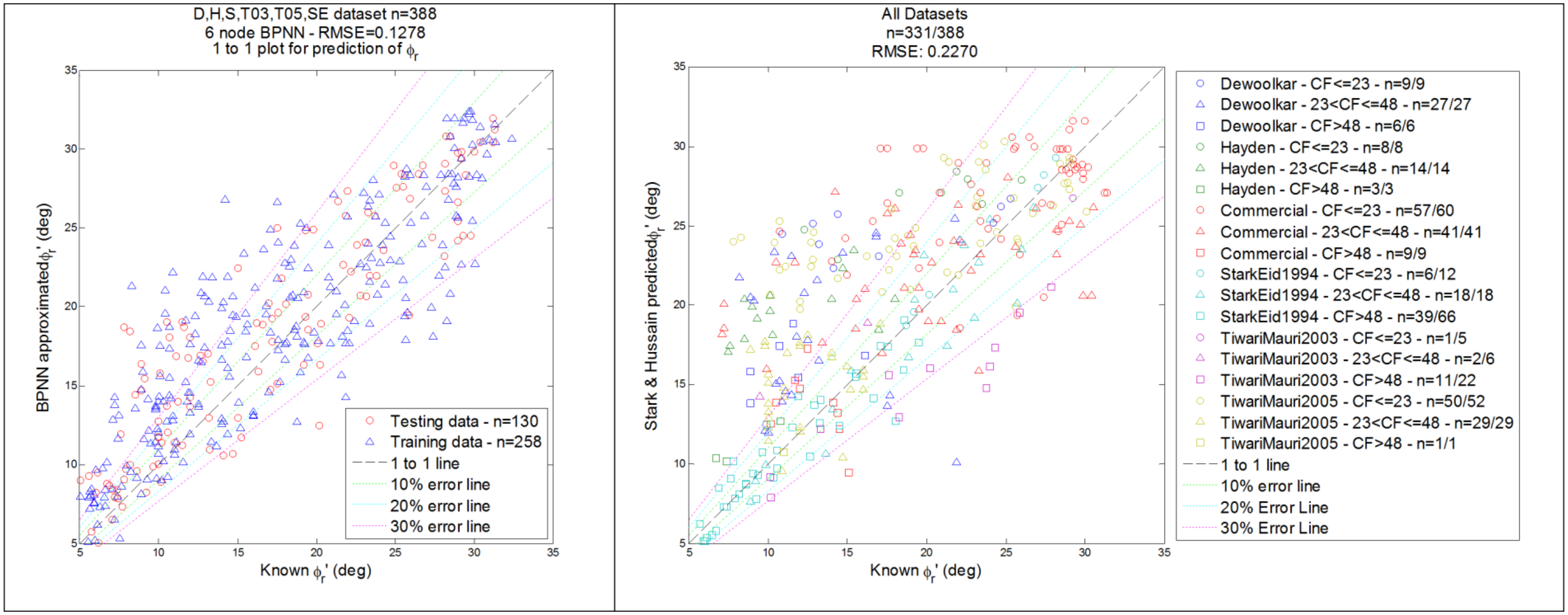
Commercial dataset n=95
8 node BPNN - RMSE=0.1997
1 to 1 plot for prediction of ϕ_r
BPNN approximatedϕ_r' (deg)
Known ϕ_r' (deg)
35
30
25
20
15
10
5
5
10
15
20
25
30
35
Testing data - n=32
Training data - n=63
1 to 1 line
10% error line
20% error line
30% error line
Commercial dataset n=95
8 node BPNN
Prediction Error plot
Root Mean Square Error
# of iterations
0.4
0.35
0.3
0.25
0.2
0.15
0.1
0.05
0
0
2000
4000
6000
8000
10000
12000
RMS of training data - n=63
RMS of testing data - n=32


Dewoolkar dataset n=42
4 node BPNN - RMSE=0.1741
1 to 1 plot for prediction of ϕ_r
BPNN predicted ϕ_r' (deg)
Known ϕ_r' (deg)
5
10
15
20
25
30
35
Testing data - n=14
Training data - n=28
1 to 1 line
10% error line
20% error line
30% error line
Dewoolkar dataset n=42
4 node BPNN
Prediction Error plot
Root Mean Square Error
# of iterations
0
0.05
0.1
0.15
0.2
0.25
0.3
0.35
0.4
2000
4000
6000
8000
10000
12000
RMS of training data - n=28
RMS of testing data - n=14


Hayden dataset n=25
5 node BPNN - RMSE=0.1838
1 to 1 plot for prediction of ϕ_r
BPNN predicted ϕ_r' (deg)
Known ϕ_r' (deg)
5
10
15
20
25
30
35
Testing data - n=9
Training data - n=16
1 to 1 line
10% error line
20% error line
30% error line
Hayden dataset n=25
5 node BPNN
Prediction Error plot
Root Mean Square Error
# of iterations
0
0.05
0.1
0.15
0.2
0.25
0.3
0.35
0.4
2000
4000
6000
8000
10000
12000
RMS of training data - n=16
RMS of testing data - n=9

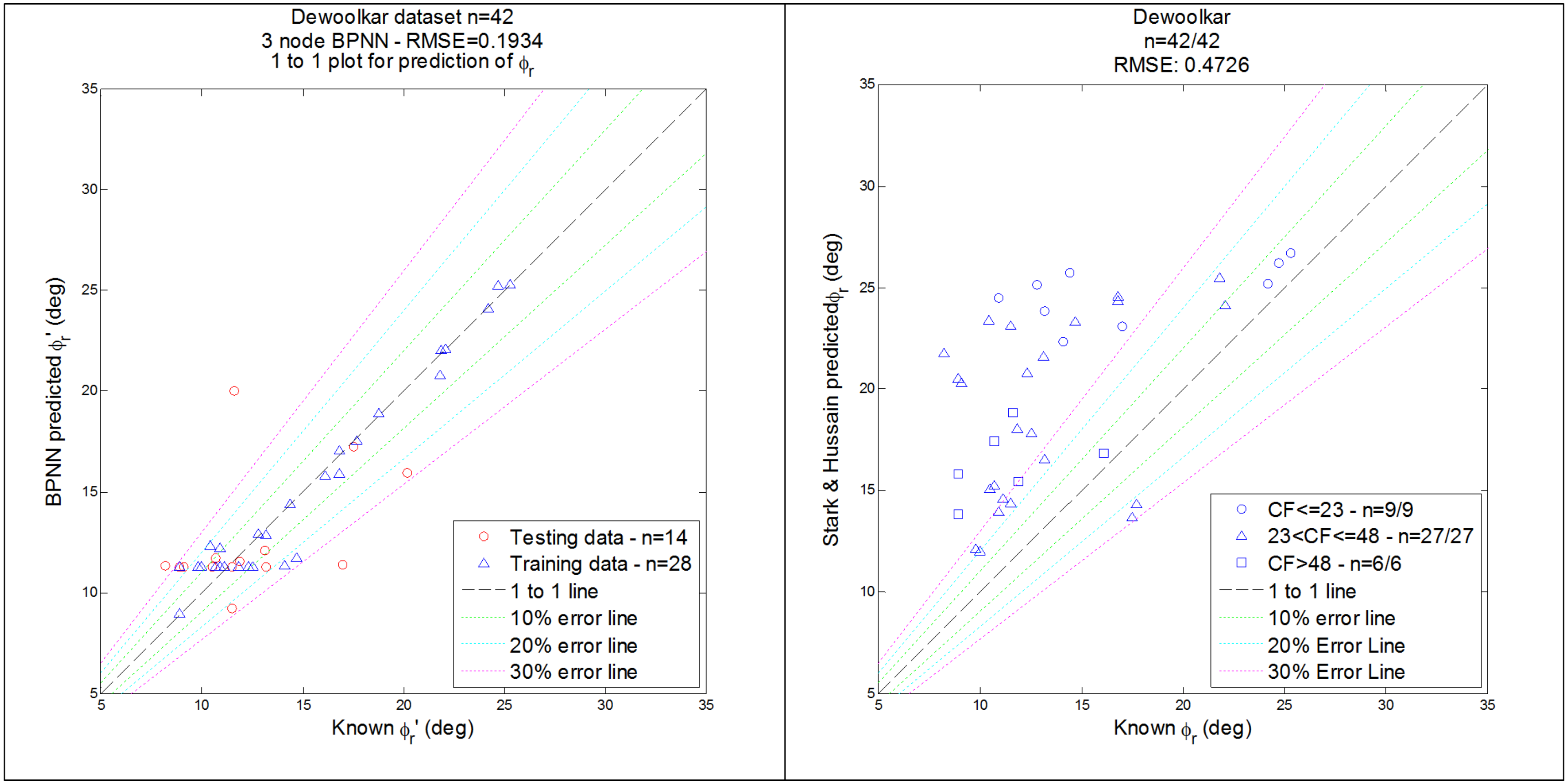

D,H,S,T03,T05,SE dataset n=388
6 node BPNN - RMSE=0.1278
1 to 1 plot for prediction of ϕ_r
BPNN approximatedϕ_r' (deg)
Known ϕ_r' (deg)
35
30
25
20
15
10
5
5
10
15
20
25
30
35
Testing data - n=130
Training data - n=258
1 to 1 line
10% error line
20% error line
30% error line
Tiwari2003 dataset n=11
6 node BPNN
Prediction Error plot
Root Mean Square Error
# of iterations
0.4
0.35
0.3
0.25
0.2
0.15
0.1
0.05
0
0
2000
4000
6000
8000
10000
12000
RMS of training data - n=22
RMS of testing data - n=11
Minimum RMSE=0.0625 @ iteration #:990

APPENDIX D – BPNN vs. Stark & Hussain (2013) Equations



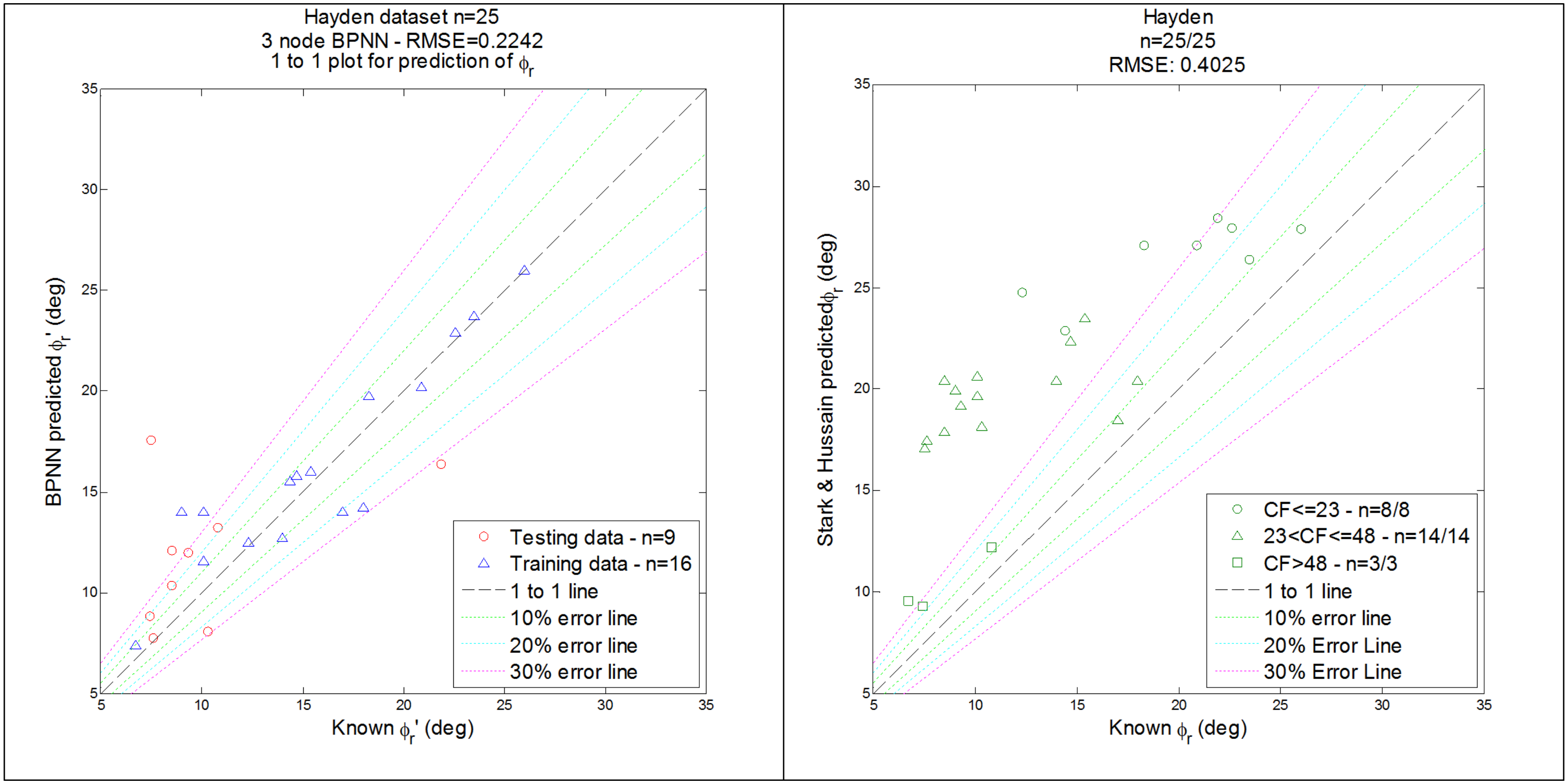
D,H,S,T03,T05,SE dataset n=388
6 node BPNN - RMSE=0.1278
1 to 1 plot for prediction of ϕ_r

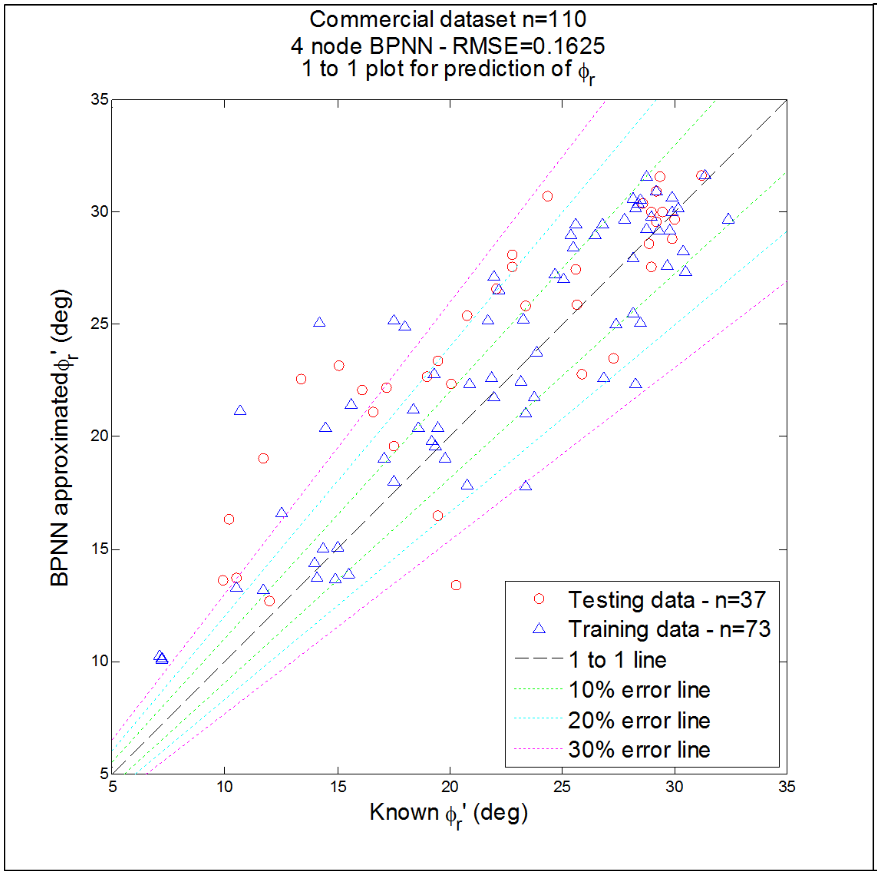
All Datasets
n=331/388
RMSE: 0.2270

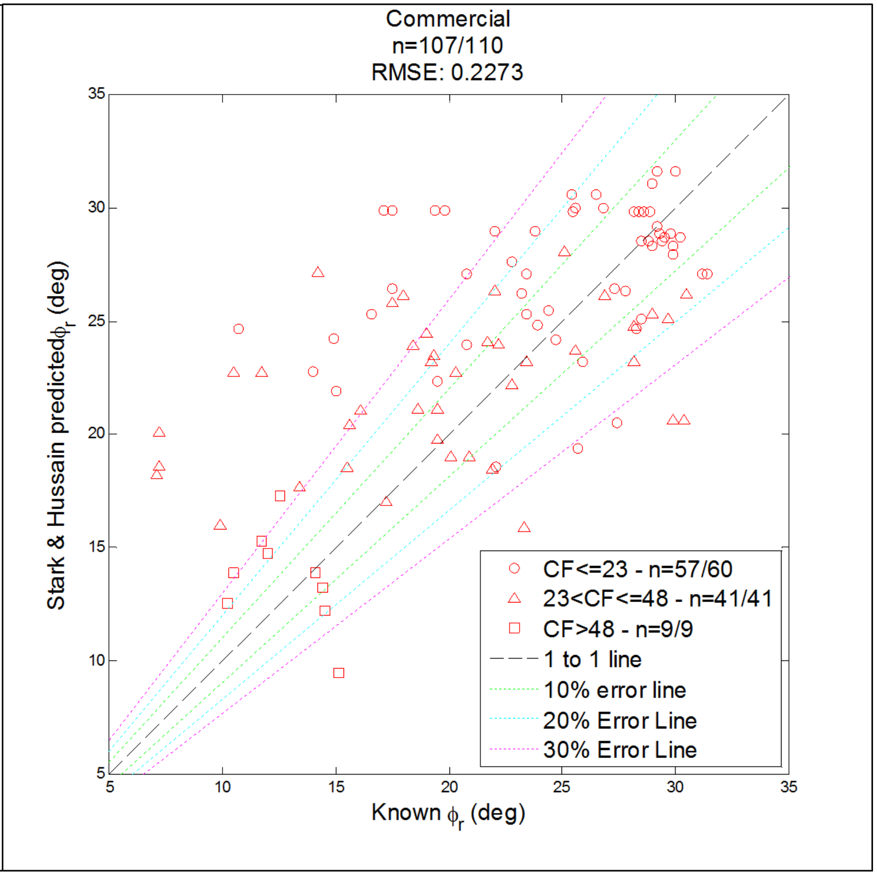


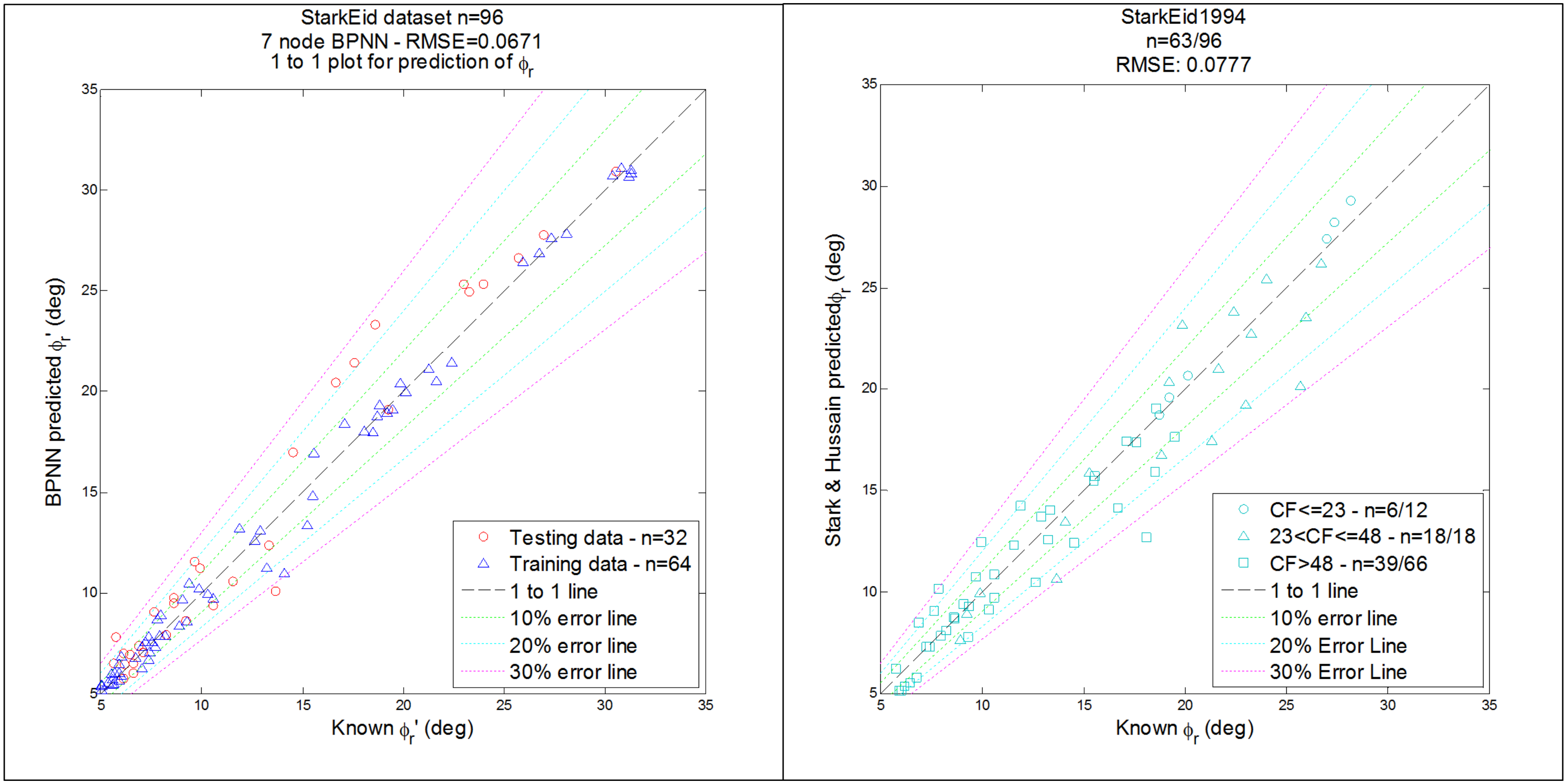
Dewoolkar dataset n=42
3 node BPNN - RMSE=0.1934
1 to 1 plot for prediction of ϕ_r

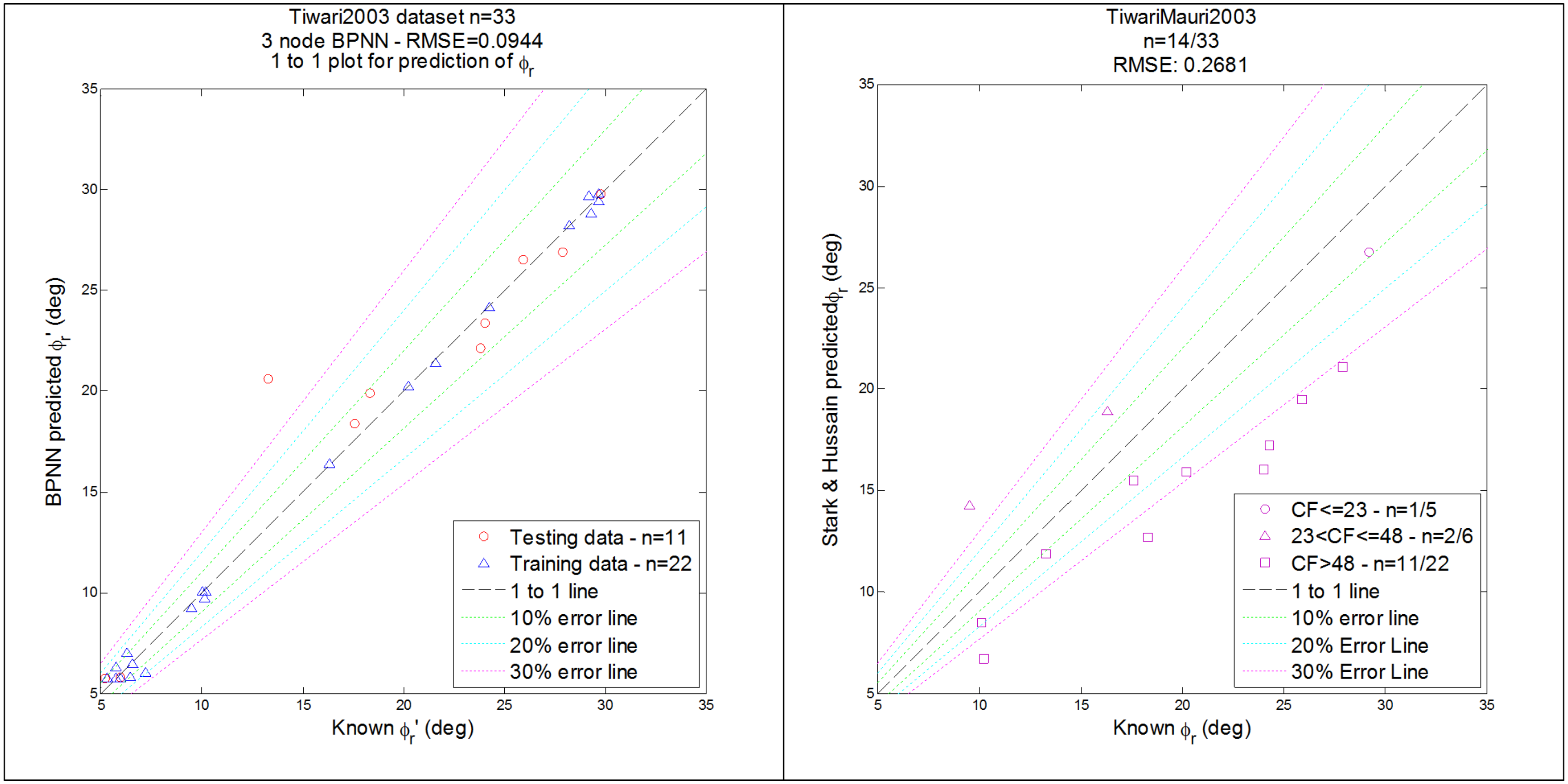
Dewoolkar
n=42/42
RMSE: 0.4726

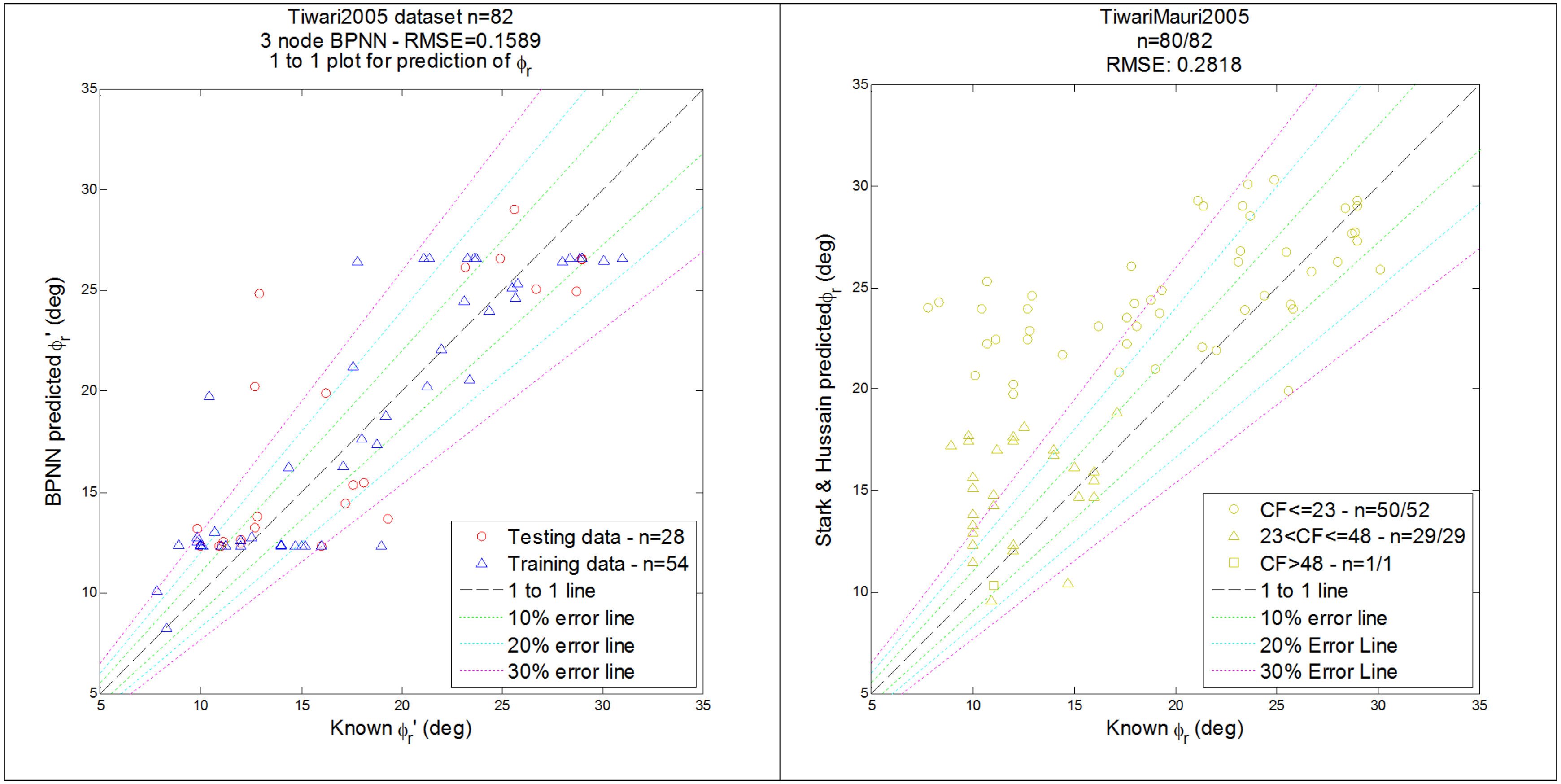

Hayden dataset n=25
3 node BPNN - RMSE=0.2242
1 to 1 plot for prediction of ϕ_r
BPNN predicted ϕ_r' (deg)
Known ϕ_r' (deg)
5
10
15
20
25
30
35
○ Testing data - n=9
△ Training data - n=16
1 to 1 line
10% error line
20% error line
30% error line
Hayden
n=25/25
RMSE: 0.4025
Stark & Hussain predictedϕ_r (deg)
Known ϕ_r (deg)
○ CF<=23 - n=8/8
△ 23<CF<=48 - n=14/14
□ CF>48 - n=3/3
1 to 1 line
10% error line
20% Error Line
30% Error Line


Commercial dataset n=110
4 node BPNN - RMSE=0.1625
1 to 1 plot for prediction of ϕ_r
BPNN approximatedϕ_r' (deg)
Known ϕ_r' (deg)
5
10
15
20
25
30
35
Testing data - n=37
Training data - n=73
1 to 1 line
10% error line
20% error line
30% error line


Commercial
n=107/110
RMSE: 0.2273
Stark & Hussain predictedϕ_r (deg)
Known ϕ_r (deg)
5
10
15
20
25
30
35
CF<=23 - n=57/60
23<CF<=48 - n=41/41
CF>48 - n=9/9
1 to 1 line
10% error line
20% Error Line
30% Error Line


StarkEid dataset n=96
7 node BPNN - RMSE=0.0671
1 to 1 plot for prediction of ϕ_r
BPNN predicted ϕ_r' (deg)
Known ϕ_r' (deg)
5
10
15
20
25
30
35
○ Testing data - n=32
△ Training data - n=64
1 to 1 line
10% error line
20% error line
30% error line
StarkEid1994
n=63/96
RMSE: 0.0777
Stark & Hussain predictedϕ_r (deg)
Known ϕ_r (deg)
○ CF<=23 - n=6/12
△ 23<CF<=48 - n=18/18
□ CF>48 - n=39/66
1 to 1 line
10% error line
20% Error Line
30% Error Line


Tiwari2003 dataset n=33
3 node BPNN - RMSE=0.0944
1 to 1 plot for prediction of ϕ_r
BPNN predicted ϕ_r' (deg)
Known ϕ_r' (deg)
5
10
15
20
25
30
35
Testing data - n=11
Training data - n=22
1 to 1 line
10% error line
20% error line
30% error line
TiwariMauri2003
n=14/33
RMSE: 0.2681
Stark & Hussain predictedϕ_r (deg)
Known ϕ_r (deg)
CF<=23 - n=1/5
23<CF<=48 - n=2/6
CF>48 - n=11/22
1 to 1 line
10% error line
20% Error Line
30% Error Line


Tiwari2005 dataset n=82
3 node BPNN - RMSE=0.1589
1 to 1 plot for prediction of ϕ_r
BPNN predicted ϕ_r' (deg)
Known ϕ_r' (deg)
5
10
15
20
25
30
35
○ Testing data - n=28
△ Training data - n=54
1 to 1 line
10% error line
20% error line
30% error line
TiwariMauri2005
n=80/82
RMSE: 0.2818
Stark & Hussain predictedϕ_r (deg)
Known ϕ_r (deg)
○ CF<=23 - n=50/52
△ 23<CF<=48 - n=29/29
□ CF>48 - n=1/1
1 to 1 line
10% error line
20% Error Line
30% Error Line

APPENDIX E – Datasets

Dewoolkar

Sample ID #	Soil ID #	Effective Normal Stress (kPa)	PI (%)	LL (%)	PL (%)	CF (%)	SF (%)	Secant Residual Friction Angle (deg)
1	B-109, S-3 (2m)	27.8	40	64	24	0	4	13.2
2	B-109, S-3 (2m)	109.6	40	64	24	0	4	17
3	B-109, S-3 (2m)	275.8	40	64	24	0	4	14.1
4	96-B102 (7m)	67	33	54	21	0	12	14.4
5	96-B102 (7m)	139.3	33	54	21	0	12	12.8
6	96-B102 (7m)	281.5	33	54	21	0	12	10.9
7	TP-410 (3m)	79	28	49	21	0	14	25.3
8	TP-410 (3m)	154.2	28	49	21	0	14	24.7
9	TP-410 (3m)	379.2	28	49	21	0	14	24.2
10	B-303 (16m)	70.4	41	65	24	36	6	17.7
11	B-303 (16m)	286.3	41	65	24	36	6	11.8
12	B-303 (16m)	557.3	41	65	24	36	6	10.7
13	B-203 (40m)	69	50	72	22	36	22	21.9
14	B-203 (40m)	281.5	50	72	22	36	22	13.2
15	B-203 (40m)	547.3	50	72	22	36	22	10.9
16	B-201 (34m)	70.4	56	81	25	41	3	18.8
17	B-201 (34m)	287.7	56	81	25	41	3	11.1
18	B-201 (34m)	559.3	56	81	25	41	3	9.8
19	Sample-5	16.8	17	45	28	42	5	21.8
20	Sample-5	80.9	17	45	28	42	5	10.4
21	Sample-5	369.2	17	45	28	42	5	8.2
22	B-302 (14m)	69.9	40	66	26	43	4	17.5

Sample ID #	Soil ID #	Effective Normal Stress (kPa)	PI (%)	LL (%)	PL (%)	CF (%)	SF (%)	Secant Residual Friction Angle (deg)
23	B-302 (14m)	283.9	40	66	26	43	4	12.5
24	B-302 (14m)	552.1	40	66	26	43	4	10.5
25	B-301 (10m)	70.4	55	82	27	43	18	20.2
26	B-301 (10m)	286.3	55	82	27	43	18	11.5
27	B-301 (10m)	556.4	55	82	27	43	18	10
28	Sample-2	17.7	21	51	30	45	3	22.1
29	Sample-4	35	22	49	27	45	3	16.8
30	Sample-2	82.4	21	51	30	45	3	13.1
31	Sample-4	102.5	22	49	27	45	3	14.7
32	Sample-4	379.2	22	49	27	45	3	12.3
33	Sample-2	381.1	21	51	30	45	3	9.1
34	Sample-6	35	32	50	18	47	2	16.8
35	Sample-6	104.4	32	50	18	47	2	11.5
36	Sample-6	380.7	32	50	18	47	2	8.9
37	Sample-3	35	38	62	24	49	1	16.1
38	Sample-3	104.9	38	62	24	49	1	11.9
39	Sample-3	382.6	38	62	24	49	1	8.9
40	Sample-1	35	22	53	31	53	1	11.6
41	Sample-1	104.4	22	53	31	53	1	10.7
42	Sample-1	380.2	22	53	31	53	1	8.9

Hayden

Sample ID #	Soil ID #	Effective Normal Stress (kPa)	PI (%)	LL (%)	PL (%)	CF (%)	SF (%)	Secant Residual Friction Angle (deg)
1	1	400	17.5	44	26.5	12.5	65	23.5
2	2	400	6.1	36.3	30.2	14.1	45.9	21.9
3	3	400	16.9	38.2	21.3	16.6	59.6	26
4	4	400	18	50.5	32.5	18.5	14.2	12.3
5	5	400	12	38	26	18.5	47.3	22.6
6	6	400	8.3	41.2	32.9	19.1	43.8	18.3
7	7	400	7.8	41.3	33.5	22.5	36.9	20.9
8	8	400	24.6	58.8	34.2	22.8	52.6	14.4
9	9	400	19.9	48.9	29	24.1	32.8	10.1
10	10	400	7.8	40.9	33.1	24.3	43	14.7
11	11	400	21.8	49.7	27.9	26.8	9.1	14
12	12	400	27	60	33	27.8	3.1	10.3
13	13	400	9.3	35.5	26.2	28.9	58	15.4
14	14	400	24.2	64.7	40.5	31.4	14	7.5
15	15	400	25.9	52	26.1	32.2	0	9
16	16	400	26.8	55.3	28.5	32.9	4.1	9.3
17	17	400	23.9	49.9	26	34.7	23.9	18
18	18	400	30	63.1	33.1	34.8	2	7.6
19	19	400	27	53.2	26.2	36.1	0	10.1
20	20	400	32.8	58.5	25.7	37.3	17.9	17
21	21	400	18.5	49.8	31.3	38.3	5.1	8.5
22	22	400	32.2	61.2	29	45.9	1.5	8.5
23	23	400	46.3	79.8	33.5	48.8	1.9	6.7
24	24	400	39	67	28	56.5	3.1	10.8
25	25	400	49.5	81.3	31.8	61.4	24.7	7.4

Commercal

Sample ID #	Soil ID #	Effective Normal Stress (kPa)	PI (%)	LL (%)	PL (%)	CF (%)	SF (%)	Secant Residual Friction Angle (deg)
1	1	12.3	8.0	37.0	29.0	5.6	2.5	17.1
2	1	49.0	8.0	37.0	29.0	5.6	2.5	17.5
3	1	195.9	8.0	37.0	29.0	5.6	2.5	23.8
4	2	12.3	8.0	37.0	29.0	5.6	3.3	19.8
5	2	49.0	8.0	37.0	29.0	5.6	3.3	19.4
6	2	195.9	8.0	37.0	29.0	5.6	3.3	22.0
7	3	24.5	25.0	49.0	24.0	7.5	1.6	23.4
8	3	49.0	25.0	49.0	24.0	7.5	1.6	20.8
9	3	97.9	25.0	49.0	24.0	7.5	1.6	17.5
10	4	146.9	9.0	38.0	29.0	8.9	0.2	29.3
11	4	293.8	9.0	38.0	29.0	8.9	0.2	29.9
12	4	1150.7	9.0	38.0	29.0	8.9	0.2	31.2
13	5	146.9	9.0	38.0	29.0	8.9	0.9	29.8
14	5	293.8	9.0	38.0	29.0	8.9	0.9	29.0
15	5	1150.7	9.0	38.0	29.0	8.9	0.9	31.4
16	6	97.9	17.0	35.0	18.0	9.2	65.5	28.2
17	7	97.9	28.0	59.0	31.0	10.2	26.6	14.9
18	7	416.2	28.0	59.0	31.0	10.2	26.6	14.0
19	7	710.0	28.0	59.0	31.0	10.2	26.6	15.0
20	8	97.9	27.0	57.0	30.0	13.0	38.0	28.3
21	8	416.2	27.0	57.0	30.0	13.0	38.0	25.9
22	8	734.5	27.0	57.0	30.0	13.0	38.0	19.5
23	9	97.9	55.0	77.0	22.0	13.0		27.4
24	9	416.2	55.0	77.0	22.0	13.0		25.7
25	9	710.0	55.0	77.0	22.0	13.0		22.1

Sample ID #	Soil ID #	Effective Normal Stress (kPa)	PI (%)	LL (%)	PL (%)	CF (%)	SF (%)	Secant Residual Friction Angle (deg)
26	10	97.9	20.0	49.0	29.0	15.0		27.3
27	10	416.2	20.0	49.0	29.0	15.0		28.5
28	10	734.5	20.0	49.0	29.0	15.0		24.7
29	11	24.5	22.0	57.0	35.0	15.5	6.4	23.4
30	11	49.0	22.0	57.0	35.0	15.5	6.4	16.6
31	11	97.9	22.0	57.0	35.0	15.5	6.4	10.7
32	12	244.8	16.0	33.0	17.0	16.4	29.2	28.4
33	12	465.2	16.0	33.0	17.0	16.4	29.2	29.2
34	12	1199.6	16.0	33.0	17.0	16.4	29.2	28.8
35	13	244.8	16.0	33.0	17.0	16.4	44.8	28.6
36	13	465.2	16.0	33.0	17.0	16.4	44.8	29.2
37	13	1199.6	16.0	33.0	17.0	16.4	44.8	29.4
38	14	97.9	15.0	44.0	29.0	16.8	31.9	22.8
39	14	416.2	15.0	44.0	29.0	16.8	31.9	27.8
40	14	710.0	15.0	44.0	29.0	16.8	31.9	24.4
41	15	24.5	14.0	30.0	16.0	18.0	23.2	29.2
42	15	49.0	14.0	30.0	16.0	18.0	23.2	30.0
43	15	97.9	14.0	30.0	16.0	18.0	23.2	29.0
44	16	97.9	18.0	35.0	17.0	18.0	39.0	28.9
45	16	416.2	18.0	35.0	17.0	18.0	39.0	30.2
46	17	97.9	18.0	35.0	17.0	18.5		25.5
47	17	416.2	18.0	35.0	17.0	18.5		29.5
48	17	710.0	18.0	35.0	17.0	18.5		29.9
49	18	73.4	16.0	33.0	17.0	18.8	16.2	26.5
50	18	195.9	16.0	33.0	17.0	18.8	16.2	25.6
51	18	734.5	16.0	33.0	17.0	18.8	16.2	28.5

Sample ID #	Soil ID #	Effective Normal Stress (kPa)	PI (%)	LL (%)	PL (%)	CF (%)	SF (%)	Secant Residual Friction Angle (deg)
52	19	73.4	16.0	33.0	17.0	18.8	25.0	25.4
53	19	195.9	16.0	33.0	17.0	18.8	25.0	26.8
54	19	734.5	16.0	33.0	17.0	18.8	25.0	28.5
55	20	97.9	11.0	28.0	17.0	19.0		28.8
56	20	416.2	11.0	28.0	17.0	19.0		32.4
57	20	979.3	11.0	28.0	17.0	19.0		28.3
58	21	97.9	30.0	50.0	20.0	19.3	21.8	23.2
59	21	416.2	30.0	50.0	20.0	19.3	21.8	23.9
60	21	710.0	30.0	50.0	20.0	19.3	21.8	20.8
61	22	73.7	21.0	42.0	21.0	23.7	18.6	21.7
62	22	97.9	21.0	42.0	21.0	23.7	18.6	28.2
63	22	391.7	21.0	42.0	21.0	23.7	18.6	22.8
64	23	49.0	21.0	42.0	21.0	23.7	18.9	18.0
65	23	195.9	21.0	42.0	21.0	23.7	18.9	22.2
66	24	144.0	15.0	33.0	18.0	24.0	14.6	30.5
67	24	265.5	15.0	33.0	18.0	24.0	14.6	29.7
68	24	704.7	15.0	33.0	18.0	24.0	14.6	30.4
69	25	244.8	15.0	33.0	18.0	24.0	21.5	29.0
70	25	465.2	15.0	33.0	18.0	24.0	21.5	28.2
71	25	1199.6	15.0	33.0	18.0	24.0	21.5	29.9
72	26	24.5	11.0	32.0	21.0	27.8	11.9	25.1
73	26	146.9	11.0	32.0	21.0	27.8	11.9	22.0
74	26	440.7	11.0	32.0	21.0	27.8	11.9	25.6
75	27	49.0	18.0	37.0	19.0	29.8	3.0	14.2
76	27	97.9	18.0	37.0	19.0	29.8	3.0	17.5
77	27	391.7	18.0	37.0	19.0	29.8	3.0	23.4

Sample ID #	Soil ID #	Effective Normal Stress (kPa)	PI (%)	LL (%)	PL (%)	CF (%)	SF (%)	Secant Residual Friction Angle (deg)
78	28	97.9	27.0	46.0	19.0	42.0	3.0	18.4
79	28	416.2	27.0	46.0	19.0	42.0	3.0	16.1
80	28	710.0	27.0	46.0	19.0	42.0	3.0	13.4
81	29	24.5	35.0	57.0	22.0	42.6	1.0	20.3
82	29	73.4	35.0	57.0	22.0	42.6	1.0	15.5
83	29	293.8	35.0	57.0	22.0	42.6	1.0	19.5
84	30	5.1	35.0	57.0	22.0	42.6	3.4	11.7
85	30	13.6	35.0	57.0	22.0	42.6	3.4	10.5
86	30	51.2	35.0	57.0	22.0	42.6	3.4	9.9
87	31	49.8	21.0	42.0	21.0	43.1	3.5	26.9
88	31	143.6	21.0	42.0	21.0	43.1	3.5	19.0
89	31	248.1	21.0	42.0	21.0	43.1	3.5	19.3
90	32	97.9	30.0	49.0	19.0	43.3	46.6	19.2
91	32	416.2	30.0	49.0	19.0	43.3	46.6	15.6
92	32	710.0	30.0	49.0	19.0	43.3	46.6	17.2
93	33	220.3	32.0	54.0	22.0	44.1	4.3	19.5
94	33	440.7	32.0	54.0	22.0	44.1	4.3	20.9
95	33	483.0	32.0	54.0	22.0	44.1	4.3	21.9
96	34	220.3	32.0	54.0	22.0	44.1	6.2	18.6
97	34	440.7	32.0	54.0	22.0	44.1	6.2	20.1
98	34	1199.6	32.0	54.0	22.0	44.1	6.2	23.3
99	35	49.0	46.0	68.0	22.0	46.2		7.2
100	35	97.9	46.0	68.0	22.0	46.2		7.2
101	35	195.9	46.0	68.0	22.0	46.2		7.1
102	36	97.9	36.0	54.0	18.0	48.7	30.2	12.5
103	36	416.2	36.0	54.0	18.0	48.7	30.2	11.7

Sample ID #	Soil ID #	Effective Normal Stress (kPa)	PI (%)	LL (%)	PL (%)	CF (%)	SF (%)	Secant Residual Friction Angle (deg)
104	36	710.0	36.0	54.0	18.0	48.7	30.2	14.5
105	37	88.9	43.0	67.0	24.0	57.1	4.7	12.0
106	37	195.9	43.0	67.0	24.0	57.1	4.7	14.1
107	37	308.4	43.0	67.0	24.0	57.1	4.7	14.4
108	38	195.9	43.0	67.0	24.0	57.1	7.0	10.5
109	38	416.2	43.0	67.0	24.0	57.1	7.0	10.2
110	38	1199.6	43.0	67.0	24.0	57.1	7.0	15.1
111	39	97.9	0.0				2.8	18.9
112	39	416.2	0.0				2.8	20.4
113	39	710.0	0.0				2.8	20.1
114	40	49.0	23.5	31.0	7.5		5.7	19.8
115	41	195.9	23.5	31.0	7.5		5.7	22.2
116	41	440.7	23.5	31.0	7.5		5.7	23.4
117	42	97.9	23.0	41.0	18.0		5.9	13.4
118	43	24.5	0.0				6.8	12.7
119	43	97.9	0.0				6.8	12.7
120	43	195.9	0.0				6.8	10.5
121	44	49.0	0.0				11.3	30.4
122	44	195.9	0.0				11.3	30.4
123	44	440.7	0.0				11.3	30.2
124	45	171.4	42.0	62.0	20.0		11.4	10.1
125	45	342.8	42.0	62.0	20.0		11.4	10.0
126	47	49.0	0.0				14.7	24.7
127	47	195.9	0.0				14.7	22.7
128	47	440.7	0.0				14.7	24.1
129	48	49.0	0.0				21.1	26.0

Sample ID #	Soil ID #	Effective Normal Stress (kPa)	PI (%)	LL (%)	PL (%)	CF (%)	SF (%)	Secant Residual Friction Angle (deg)
130	48	195.9	0.0				21.1	28.1
131	48	440.7	0.0				21.1	29.3
132	49	171.4	27.0	52.0	25.0		24.1	13.8
133	49	342.8	27.0	52.0	25.0		24.1	10.7
134	50	24.5	0.0				32.5	33.7
135	50	97.9	0.0				32.5	28.7
136	50	195.9	0.0				32.5	29.2
137	51	24.5	0.0					13.2
138	51	49.0	0.0					13.7
139	51	97.9	0.0					16.3

Stark & Eid

Sample ID #	Soil ID #	Effective Normal Stress (kPa)	PI (%)	LL (%)	PL (%)	CF (%)	SF (%)	Secant Residual Friction Angle (deg)
1	2	100	10	28	18	10		30.84
2	2	400	10	28	18	10		30.57
3	2	700	10	28	18	10		30.4
4	14	100	35	76	41	16		20.14
5	14	400	35	76	41	16		19.21
6	14	700	35	76	41	16		18.71
7	1	100	8	24	16	18		31.34
8	1	400	8	24	16	18		31.32
9	1	700	8	24	16	18		31.22
10	4	100	12	37	25	19		28.15
11	4	400	12	37	25	19		27.37
12	4	700	12	37	25	19		27
13	8	100	26	50	24	33		19.85
14	8	400	26	50	24	33		19.24
15	8	700	26	50	24	33		18.85
16	7	100	20	47	27	41		22.4
17	7	400	20	47	27	41		21.67
18	7	700	20	47	27	41		21.3
19	16	100	52	82	30	42		15.27
20	16	400	52	82	30	42		14.09
21	16	700	52	82	30	42		13.65
22	5	100	19	39	20	43		24
23	24	100	101	128	27	43		9.86
24	5	400	19	39	20	43		23.28
25	24	400	101	128	27	43		9.21

Sample ID #	Soil ID #	Effective Normal Stress (kPa)	PI (%)	LL (%)	PL (%)	CF (%)	SF (%)	Secant Residual Friction Angle (deg)
26	5	700	19	39	20	43		23
27	24	700	101	128	27	43		8.9
28	3	100	17	35	18	44		26.74
29	3	400	17	35	18	44		25.97
30	3	700	17	35	18	44		25.71
31	10	100	24	53	29	50		17.11
32	10	400	24	53	29	50		15.56
33	10	700	24	53	29	50		14.52
34	12	100	44	68	24	51		11.88
35	12	400	44	68	24	51		9.92
36	12	700	44	68	24	51		9.36
37	13	100	47	69	22	56		13.32
38	13	400	47	69	22	56		11.54
39	13	700	47	69	22	56		10.32
40	17	100	45	89	44	57		9.65
41	17	400	45	89	44	57		7.62
42	17	700	45	89	44	57		5.72
43	15	100	52	77	25	59		13.23
44	15	400	52	77	25	59		10.58
45	15	700	52	77	25	59		9.28
46	9	100	32	52	20	63		19.5
47	21	100	69	111	42	63		8.59
48	9	400	32	52	20	63		18.53
49	21	400	69	111	42	63		7.39
50	9	700	32	52	20	63		18.07
51	21	700	69	111	42	63		6.01

Sample ID #	Soil ID #	Effective Normal Stress (kPa)	PI (%)	LL (%)	PL (%)	CF (%)	SF (%)	Secant Residual Friction Angle (deg)
52	29	100	145	192	47	65		7.1
53	31	100	205	253	48	65		5.98
54	29	400	145	192	47	65		5.67
55	31	400	205	253	48	65		5.57
56	29	700	145	192	47	65		5.57
57	31	700	205	253	48	65		5.04
58	20	100	66	101	35	66		9.05
59	20	400	66	101	35	66		7.93
60	20	700	66	101	35	66		6.14
61	23	100	84	121	37	67		7.94
62	23	400	84	121	37	67		7.02
63	23	700	84	121	37	67		5.62
64	11	100	30	62	32	68		15.52
65	19	100	61	98	37	68		10.57
66	11	400	30	62	32	68		12.91
67	19	400	61	98	37	68		8.19
68	11	700	30	62	32	68		12.64
69	19	700	61	98	37	68		6.45
70	26	100	126	157	31	71		7.51
71	26	400	126	157	31	71		6.15
72	26	700	126	157	31	71		5.51
73	27	100	131	170	39	72		7.71
74	30	100	163	219	56	72		6.59
75	27	400	131	170	39	72		7.03
76	30	400	163	219	56	72		5.92
77	27	700	131	170	39	72		6.13

Sample ID #	Soil ID #	Effective Normal Stress (kPa)	PI (%)	LL (%)	PL (%)	CF (%)	SF (%)	Secant Residual Friction Angle (deg)
78	30	700	163	219	56	72		5.01
79	6	100	21	46	25	73		18.59
80	22	100	59	112	53	73		8.62
81	6	400	21	46	25	73		17.58
82	22	400	59	112	53	73		7.22
83	6	700	21	46	25	73		16.66
84	22	700	59	112	53	73		5.91
85	18	100	68	94	26	77		7.81
86	18	400	68	94	26	77		6.87
87	18	700	68	94	26	77		6.73
88	25	100	97	138	41	78		8.26
89	25	400	97	138	41	78		7.35
90	25	700	97	138	41	78		5.92
91	28	100	129	184	55	84		7.41
92	28	400	129	184	55	84		6.6
93	28	700	129	184	55	84		5.64
94	32	100	244	288	44	88		5.33
95	32	400	244	288	44	88		5.03
96	32	700	244	288	44	88		4.56

Tiwari & Mauri 2003

Sample ID #	Soil ID #	Effective Normal Stress (kPa)	PI (%)	LL (%)	PL (%)	CF (%)	SF (%)	Secant Residual Friction Angle (deg)
1	1	250	0	0	0	0	100	29.7
2	2	250	4	7.5	3.5	10	90	29.8
3	3	250	13.7	44.9	31.2	10	90	29.2
4	4	250	6	18.7	12.7	20	80	29.7
5	5	250	59.3	88	28.7	20	80	21.6
6	6	250	40.7	62.6	21.9	25	75	16.3
7	7	250	6.7	23.5	16.8	30	70	29.3
8	8	250	98.9	134.4	35.5	30	70	10.0
9	9	250	6.9	27.6	20.7	40	60	28.2
10	10	250	58.6	84.1	25.5	40	60	9.5
11	11	250	157.6	183.3	25.7	40	60	7.2
12	12	250	8.9	34.2	25.3	50	50	27.9
13	13	250	34.2	55.3	21.1	50	50	20.2
14	14	250	33.7	57.2	23.5	50	50	17.6
15	15	250	201.7	232.5	30.8	50	50	4.9
16	16	250	13.2	40.5	27.3	60	40	25.9
17	17	250	105.8	134.4	28.6	60	40	5.7
18	18	250	18.1	49.7	31.6	70	30	24.3
19	19	250	64.3	95.1	30.8	70	30	10.1
20	20	250	165.9	199.7	33.8	70	30	6.4
21	21	250	312.4	343.2	30.8	70	30	5.2
22	22	250	19.5	54.8	35.3	80	20	24.0
23	23	250	245.6	288	42.4	80	20	6.0
24	24	250	21.5	61	39.5	90	10	23.8
25	25	250	25	70	45	100	0	18.3
26	26	250	35.7	74.2	38.5	100	0	13.3
27	27	250	73.6	115.2	41.6	100	0	10.2
28	28	250	105.1	150.1	45	100	0	6.3

Sample ID #	Soil ID #	Effective Normal Stress (kPa)	PI (%)	LL (%)	PL (%)	CF (%)	SF (%)	Secant Residual Friction Angle (deg)
29	29	250	119.3	167.9	48.6	100	0	6.5
30	30	250	175.9	224.3	48.4	100	0	6.0
31	31	250	193.7	241.8	48.1	100	0	5.3
32	32	250	228.1	277.5	49.4	100	0	5.7
33	33	250	451.5	485.7	34.2	100	0	4.4

Tuiwariu and Mauri 2005

Sample ID #	Soil ID #	Effective Normal Stress (kPa)	PI (%)	LL (%)	PL (%)	CF (%)	SF (%)	Secant Residual Friction Angle (deg)
1	1	250	9.6	36.4	26.8	0.4		28.4
2	2	250	11.1	31.1	20	1		24.9
3	3	250	4.5	32	27.5	1.8		23.6
4	4	250	6	35	29	2.3		21.1
5	5	250	23.5	54	30.5	2.4		24.4
6	6	250	23	57	34	2.8		10.4
7	7	250	6	41	35	2.9		28.9
8	8	250	5.5	36	30.5	3.9		21.4
9	9	250	16.1	42.7	26.6	4.2		29
10	10	250	16.6	48.6	32	4.3		30.1
12	12	250	8	26	18	5		31
11	11	250	9	49	40	5		26.7
13	13	250	28	56	28	5.4		25.7
14	14	250	8	35	27	6		29
16	16	250	9	47	38	6.5		28
15	15	250	27	65	38	6.5		17.6
17	17	250	42.6	65.8	23.2	6.7		21.3
18	18	250	15.4	37.9	22.5	7		23.7
19	19	250	21.9	47.8	25.9	8.9		17.8
20	20	250	5	36	31	9		23.3
25	25	250	10	36	26	10		29
23	23	250	29.9	41.2	11.3	10		28.7
24	24	250	30.3	46.8	16.5	10		23.1
22	22	250	19	54	35	10		12.9
21	21	250	17	57	40	10		12.7
26	26	250	20.7	44.6	23.9	10.9		23.2
27	27	250	12.7	57.3	44.6	11.2		23.4
28	28	250	38.6	55.6	17	12		8.3

Sample ID #	Soil ID #	Effective Normal Stress (kPa)	PI (%)	LL (%)	PL (%)	CF (%)	SF (%)	Secant Residual Friction Angle (deg)
29	29	250	19	45	26	13		25.5
30	30	250	26	53	27	14		19.3
31	31	250	17.3	57	39.7	15.2		25.8
32	32	250	34.3	56.8	22.5	16		7.8
33	33	250	20.7	77.2	56.5	16.2		25.6
34	34	250	27.5	67.7	40.2	17		14.4
35	35	250	29.9	72.3	42.4	17		17.2
36	36	250	21.9	66.7	44.8	17.5		22
37	37	250	24	61	37	18		18.1
38	38	250	34.1	64	29.9	19.2		12.7
39	39	250	18	55	37	19.5		18.8
40	40	250	29	62	33	19.9		12.8
42	42	250	19	59	40	20		17.6
41	41	250	15	61	46	20		16.2
44	44	250	32.6	55.8	23.2	20.5		18
43	43	250	25.8	78.2	52.4	20.5		12
45	45	250	36.4	71.5	35.1	21.2		19
46	46	250	29.2	58	28.8	21.3		19.2
47	47	250	25.6	75.5	49.9	21.5		12
48	48	250	19	51	32	21.8		10.7
49	49	250	34	65	31	22		10.7
50	50	250	31	84	53	22		14
51	51	250	35.3	73	37.7	22.1		10.1
52	52	250	21	64	43	23		11.1
53	53	250	38	70	32	24		8.9
54	54	250	32	63	31	24.8		17.1
55	55	250	38	71	33	26.2		11.2
56	56	250	31	69.2	38.2	26.3		12
57	57	250	40	69	29	27.2		9.8

Sample ID #	Soil ID #	Effective Normal Stress (kPa)	PI (%)	LL (%)	PL (%)	CF (%)	SF (%)	Secant Residual Friction Angle (deg)
58	58	250	38	69	31	27.5		9.8
59	59	250	31	66	35	27.7		12.5
61	61	250	38.2	68	29.8	28.2		9.8
60	60	250	47.1	86.3	39.2	28.2		10
62	62	250	34.6	68.1	33.5	28.8		12
63	63	250	38	78	40	30.1		16
65	65	250	35	71	36	30.2		14
64	64	250	38	75	37	30.2		15
66	66	250	36	76	40	30.8		16
67	67	250	51	82	31	31		15.2
68	68	250	37.1	72.3	35.2	31.1		14
69	69	250	39.1	82	42.9	32.2		16
70	70	250	48	96.2	48.2	32.2		12
71	71	250	50.5	91.3	40.8	32.8		10
72	72	250	62.4	94.6	32.2	33.2		10
73	73	250	59.2	94.7	35.5	33.5		12
74	74	250	43.9	81.4	37.5	37.5		11
75	75	250	41.9	89	47.1	37.5		10
76	76	250	49.5	77.2	27.7	38.2		10
77	77	250	36.3	79.8	43.5	40.2		10
78	78	250	66.2	108	41.8	41.2		14.7
79	79	250	65.5	120	54.5	42		10.9
80	80	250	68.3	100	31.7	42.8		10
81	81	250	52.9	84	31.1	45.2		11
82	82	250	54	83	29	51		11