

Modeling Estuarine Salinity Using Artificial Neural Networks

Christopher Wan
Dreyfoos School of the Arts

I. Personal

My project is a unique combination of computer science and environmental science. Indeed, my research features Artificial Neural Networks (ANNs), an artificial intelligence tool that can model non-linear processes to a high degree of accuracy, applied to modeling estuarine salinity in the Loxahatchee River as a case study.

I completely fell in love with computer science in my Advanced Placement Computer Science A course in my junior year of high school. I was fascinated by the multifaceted and complex nature of the field in its ability to use creative logic for problem solving. From staring at my computer screen for hours on end, I have learned to admire and revel in the extensive process of harnessing the infinite power of computer science right at my fingertips to find the most efficient solution. Thus, with my fervent love for computer science, I sought to combine it with one of the most touching and influential aspects of my life: The Loxahatchee River. Throughout my early development, my parents would take me on canoeing trips down this last “Wild and Scenic River” in South Florida. As these experiences passively culminated in the back of my mind, the passionate ignition to my interest with the preservation efforts in my childhood river came finally in the 7th grade. In my first science fair project, my father and I canoed downstream the river trailing a salinity marker to measure the gradual increase in salinity while I silently remarked at the fascinating vegetation change. My interest in the fascinating change has carried through my high school years. As a junior, I programmed a time series auto-regressive model to evaluate change in salinity with freshwater inflow. However, the model was developed only at one location with one variable. In spite of generally acceptable performance of the model, I was motivated to pursue for other modeling techniques capable of simulating salinity at multiple locations in the river.

My father was the first to introduce me to ANNs when he came home one day at the end of my junior year with a publication from *Ecological Modeling* on ANNs. Although the paper admittedly confused me at first, it quickly pulled me in with the ANN's amazing ability to imitate the neurons in a human brain with a complicated mathematical "learning" process. Because I had just finished taking Multivariable Calculus, the cornerstone of ANN logic, I immediately began scouring the Internet and other journal publications to fully understand how ANNs function. With countless hours of literature search and reading, I was finally ready to tackle modeling estuarine salinity using ANNs. However, due to the complexity of ANNs, it was one thing to understand how they work yet another to use them for modeling. In fact, researchers who use ANNs actually have to purchase commercial software such as MATLAB for upwards of \$5,000. Because I had no such funds, I embarked on the challenging journey of writing ANN code using Java right on my own laptop, opening up a new world of science and advancing my scientific qualities.

With the difficulty of this task, the experience primarily humbled me to realize that in the world of science and academics, I will always be a student. There were times when I ran into logical roadblocks that left me frustrated and encouraged me to seek the aid of experts on the topic. Moreover, this research has taught me the importance of curiosity in driving myself to dive deeper in anything I do. In my research, I quenched my thirst for knowledge by learning more about modeling, fervently exploring various journals, and studying ANN theory. Likewise, I will delve into any new scientific endeavor in the future with child-like curiosity and scholarly zeal. Finally, my research has truly shown me the diverse and endless scope of science in its ability to perform any task. I was fascinated by how computer science and mathematics could simulate the complex biological process of neuronal firing in the human brain to produce an action and learn.

There are many different ways to approach a problem, and it is always interesting finding the most efficient and novel way. With the plethora of problems in the world, engineers and scientists can seek to ameliorate them with the use of computer engineering. No longer is it a question whether we can tackle these problems; the question is when we will tackle them and how quickly we can come up with new ideas to solve them.

With the versatility of computer engineering and of the combination of science and mathematics in general, it is up to the young generation to be the future that everyone dreams of. Upcoming students who endeavor to engage in ground-breaking research will be the ones who change the world. For such students, it is of utmost importance that they pursue what they are passionate about. Passion is what drives motivation and curiosity. It is what keeps researchers patient in the midst of stagnation, optimistic in the face of failure, and diligent in approaching each new day in lab as an opportunity. Without passion, research becomes dull and lacking in drive. Students aspiring to conduct research should also employ the combination of analysis and creativity. For instance, with respect to the former, physicists and mathematicians who use complex equations on a regular basis must reason accurately to avoid errors in calculations and results. Careful analysis is important to all fields, however, in drawing reasonable conclusions and interpreting large trends in data. Creativity, on the other hand, is also important especially when confronted with data that contradict an expected hypothesis and result in failure. Scientists need to be creative in coming up with novel solutions to a problem and actively making unconventional but intuitive leaps to explore a method.

II. Research

1.0 Introduction

To predict the response of estuarine ecosystems to anthropogenic and natural changes, process-based physical computer models serve as an important tool for simulation of estuarine salinity. Among the school of data-driven parametric models as alternative tools for process-based physical models to simulate environmental variables, artificial neural networks (ANNs) have become an increasingly popular modeling technique over the past two decades (Maren et al., 1990; Schalkoff, 1997; Dawson and Wilby, 2001; Maier and Dandy, 2001; Dawson et al., 2005; Pao, 2008). ANNs is a programming logic model using multivariable calculus and an algorithmic “learning” process to simulate various functions related with information processing, including pattern recognition, forecasting, and data compression. The logic of ANNs aims to imitate the workings of individual neurons in the human brain, making it able to dynamically model non-linear functions with very high accuracy. In this way, a modeler using ANNs has no need to explore the intermediate processes that occur in the relationship between an input variable and the final output. Instead, the ANN implicitly takes them into account during its learning process. Transport of salt in estuaries is influenced by multiple factors such as freshwater inflows and tide, and their relationship with salinity is highly complex and non-linear, making it ideal cases for the application of ANNs. The objective of this study is to develop ANNs to predict estuarine salinity using the Loxahatchee River as a case study. The Loxahatchee River is selected because of concerns about saltwater intrusion into the river (SFWMD, 2002; 2006; Kaplan et al., 2010; Liu et al., 2011). The hypothesis is that salinity in the Loxahatchee River can be effectively simulated with ANNs, through properly training and testing, using freshwater inflow, rainfall, and tide as inputs.

2.0 Materials and Methods

2.1 Data sources

The United States Geological Service (USGS) has been monitoring flow over the Lainhart Dam, the headwater structure of the River, since April 1971. The data are downloaded from the SFWMD Database at www.sfwmd.gov. The SFWMD has been monitoring rainfall over the watershed (Figure 1). Daily rainfall data were downloaded from the SFWMD Database at www.sfwmd.gov.

The USGS, under contract with the SFWMD, has been monitoring conductivity, temperature, and tidal water levels at several locations along the river since 2002 (Figure 1). The sites selected for this study include RM9, RM8 and RM6 (RM=River Miles from the Ocean). A pressure sensor is used to measure the water level. Salinity is calculated with conductivity and temperature using standard equations based on the practical salinity unit (psu). These data were downloaded from <http://waterdata.usgs.gov/nwis>.

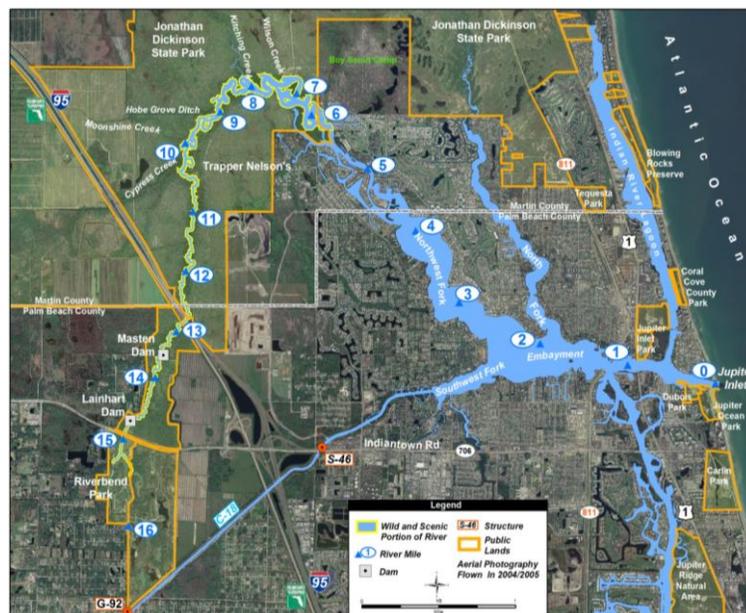


Figure 1. The Loxahatchee River watershed and the river miles within the Loxahatchee River (Courtesy of SFWMD).

2.2 Theory and techniques of ANNs

ANNs are a form of artificial intelligence that can learn a behavior or pattern. The most common learning type, supervised learning, involves presenting the network with a set of expected outputs, and each predicted output is compared with the respective expected output to obtain an error. Using the error, each weight is iteratively updated using gradient descent to minimize the error. Tasks that fall within the paradigm of supervised learning are pattern recognition and regression.

The multilayer perceptron (MLP) shown in Figure 2 is the most common type of ANNs. In MLP, there are different layers containing different numbers of neurons: an input layer, hidden layers, and an output layer. In the input layer, input neurons are fed input values, and these values are sent to hidden neurons in the hidden layer through connections with values known as weights which are multiplied to the input values. The hidden neurons then sum the inputs and apply the sum to an “activation function” whose role is to control the amplitude of the hidden neurons’ output. The output from the activation function is then sent to the output neurons via connections with other weights. The output neurons then finally compute the output with the same activation function used by the hidden neurons. All numbers must be normalized to a value between 0 and 1 (in this study) in order to prevent possible overestimates or underestimates of output that may be caused by large numbers.

In the input and output layers, the number of input neurons and output neurons will correspond to the number of inputs required to obtain an output. But determining the number of hidden neurons is much more of a challenge. If there are too many hidden neurons, the network will not be able to recognize chance or random deviations from normal data. If there are too few hidden neurons, then the network may not be able to learn properly. A common practice is to

determine the number of hidden neurons through trial and error (Maier and Dandy, 1996). Biased neurons are also an important facet of ANNs. The biased neuron will only send a raw output of 1 or -1, and it is never able to receive any input because its output will be same regardless of its input. Without a biased neuron, the modeler is simultaneously adjusting the weight and the output value of the neuron. But adding a bias neuron without a changing value allows the modeler to control the behavior of the layer.

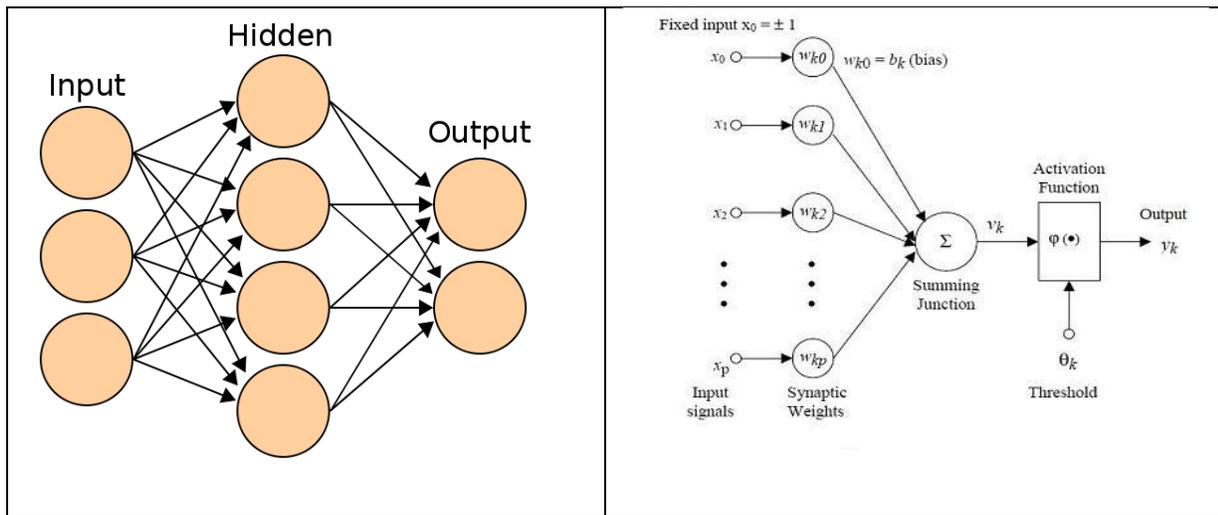


Figure 2. The structure (left) and data flows (right) of the forward feed of ANNs.

The MLP also uses what are known as activation functions. These mathematical functions are used to keep the processed data as it progresses from one node to the next at a normalized value. Many MLPs typically use the logistic sigmoid function or the hyperbolic tangent function as activation functions (Dawson and Wilby, 2001). These two functions effectively normalize the data to values usable by the ANN. The logistic sigmoid function has an infinite domain, and the range is strictly $0 < y < 1$, and the hyperbolic tangent curve similarly has an infinite domain, and its range is $-1 < y < 1$. Therefore, a modeler would usually choose

between the two depending on the nature of the data sets and whether negative values would apply to it.

The MLP learns through the error back-propagation algorithm. This algorithm utilizes gradient descent and multivariable calculus. Let i denote an anterior node and j denote a posterior node for any given neuron or weight. The error signal for unit j (δ_j) is the derivative of the error (δE) with respect to the weighted output (δnet_j), so

$$\delta_j = -\frac{\delta E}{\delta net_j} \quad [1]$$

The gradient, or change, for weight ij (Δw_{ij}) is the derivative of the error (δE) with respect to the weights (δw_{ij}), so

$$\Delta w_{ij} = -\frac{\delta E}{\delta w_{ij}} \quad [2]$$

Using the Chain Rule:

$$\Delta w_{ij} = -\frac{\delta E}{\delta net_i} * \frac{\delta net_i}{\delta w_{ij}}$$

$$\frac{\delta net_i}{\delta w_{ij}} = \frac{\delta}{\delta w_{ij}} \sum w_{ij} y_j = y_j$$

$$\Delta w_{ij} = \delta_i y_j \quad [3]$$

Most MLPs use the sum-squared loss to calculate the error (E) for an output unit

$$E = \frac{1}{2} \sum (t_o - y_o)^2$$

$$\delta_o = t_o - y_o$$

where t_o = expected output and y_o = actual output. Calculating the error for hidden units is more difficult because the error must be propagated back from the output nodes. Using the chain rule, Eq. [1] becomes:

$$\delta_j = - \sum \frac{\delta E}{\delta net_i} * \frac{\delta net_i}{\delta y_i} * \frac{\delta y_i}{\delta net_j}$$

Examining the summation:

$$\frac{\delta E}{\delta net_i} = \delta_i$$

$$\frac{\delta net_i}{\delta y_i} = \frac{\delta}{\delta y_i} \sum w_{ij} y_j = w_{ij}$$

$$\frac{\delta y_i}{\delta net_j} = \frac{\delta f_j(net_j)}{\delta net_j} = f'_j(net_j)$$

So

$$\delta_j = f'_j(net_j) \sum \delta_i w_{ij} \quad [4]$$

In order to prevent the network from reaching a localized error rather than an absolute error in the error function, many researchers use a learning rate (μ) and a momentum (β) in the error backpropagation algorithm. The learning rate is multiplied to the weight change, and then the weight change at time $t-1$ is multiplied to the momentum to essentially “push” the error from the local minimum to the absolute. It is given by:

$$\Delta w = \mu \Delta w + \beta \Delta w_{t-1} \quad [5]$$

The network will stop learning once a specified minimum error or a specified number of iterations (epochs) has been reached. Prior to learning, the data is divided into two parts, one for calibration of the network (training), and one for validation to check the network (testing). These splits will often vary depending on the nature of the data set and the choice of the researcher.

2.3 Programming of ANNs with Java

Many studies (e.g., Maier and Dandy, 1996; Patel et al., 2002) use NeuralWare, pre-written ANNs modeling technology (NeuralWare Inc., 1991). In this study, a novel ANNs code is independently written and tested using Java without purchase of external codes. The MLP as

defined in Eqs. [1]-[5] is coded in the ANNs. Four classes are written in the code. A *Connection* class is created to properly represent every connection within the neural network. An instance of *Connection* will hold the weight between any two given neurons with the capability to update it. A *Neuron* class is created to represent any type of neuron. It holds the activation function for any neuron and is able to function as a biased neuron. A *NeuralNetwork* class is created that uses *Connections* and *Neurons* arranged in the various node configurations. This class implements the error backpropagation algorithm and trains the weights using the specified learning rate, momentum, and number of epochs. The final weights are printed onto a text file to be used by a *Validation* class. The *Validation* class is created using the final weights in the *NeuralNetwork* class stored in a text file, and the weights are initialized in the constructor of the *Validation* class. It is capable of running the validation data set without re-learning the network.

2.4 Training and testing of ANNs

The structure of ANNs developed in this study consists of an input layer, an output layer, and a hidden layer in between. One hidden layer was selected because many studies indicated that ANNs with one hidden layer is capable of modeling any nonlinear function (Schalkoff, 1997; Maier and Dandy, 2002). The input variables are flow, rainfall, and tide (water level), and the output is salinity. The choice of these input variables is based on the understanding of transport processes of saltwater in estuaries (Ji, 2008) in conjunction with inspections of time series plots of these inputs and output (not shown). There was one biased neuron in the input layer and one biased neuron in the hidden layer. The input values were normalized between 0 and 1 by subtracting the minimum from each data and divided by the range of the data set. For the neurons in both the hidden and output layers, the logistic sigmoid function is used as the activation function:

$$f(x) = \frac{1}{1+e^{-x}} \quad [6]$$

where x = the weighted sum of inputs to the neurons and $f(x)$ = the neuron's output. This function is common in ANNs applications because it is continuous and relatively easy to compute (Dawson and Wilby, 2001).

Data from the period of November 2003 through October 2011 were selected for training and testing the ANNs at each location. The data were divided into two sets: November 2003 to May 2007 as the calibration set, and June 2007 to October 2011 as the validation set. This long-term data set encompasses a sufficient range of hydrologic events and conditions including the average, wet, and dry years to activate all salinity response processes during both the calibration and the validation periods.

To determine the optimum model structure, the number of hidden neurons in the hidden layer was evaluated following Maier and Dandy (2001). In the evaluation, the number of hidden neurons varied from 3 – 8. The error backpropagation algorithm was run until a maximum number of epochs or a minimum error was reached. The maximum number of epochs was set to 50,000, and the minimum error was set to 1×10^{-9} . Preliminary trials indicated that further increase in the number of epochs did not reduce the error significantly. The root mean squared error (RMSE) between the network output (modeled) and the target output (measured) was calculated for the entire data set (including both training and testing) (see Eq. [7]). The optimum number of hidden neurons was selected if it gave the smallest RMSE value. The learning process then continues with determination of the optimal learning rate and momentum, respectively. The learning rate was changed from 0.1 – 1. The momentum varied from 0.05 – 0.3. The RMSE values calculated with both the entire data sets were again used to determine the optimal learning rate and momentum.

In addition to the aforementioned RMSE, two other statistical parameters are used to evaluate the performance of ANNs: the coefficient of determination (r^2) and the Relative Root Mean Square Error (RRE). The mathematical equations of RMSE, r^2 , and RRE are given below:

$$RMSE = \sqrt{\frac{\sum_{i=1}^n (O_i - P_i)^2}{n}} \quad [7]$$

$$r^2 = \left[\frac{\sum_{i=1}^n (O_i - \bar{O})(P_i - \bar{P})}{\sqrt{\sum_{i=1}^n (O_i - \bar{O})^2 \sum_{i=1}^n (P_i - \bar{P})^2}} \right]^2 \quad [8]$$

$$RRE = \frac{\sqrt{\frac{\sum_{i=1}^n (O_i - P_i)^2}{n}}}{O_{max} - O_{min}} \times 100 \quad [9]$$

where n = the number of days during the period of evaluation, O_i = the observed daily salinity, \bar{O} = the mean of the observed daily salinity, O_{max} = the maximum value of observed salinity, O_{min} = the minimum value of observed salinity, P_i = the simulated daily salinity, and \bar{P} = the mean of the simulated daily salinity.

3.0 Results and Discussion

3.1 Model structure

Figure 3 shows the trial-and-error process in finding the number of hidden nodes, learning rate, and momentum. This is an important step in training the network because the lower the error, the more able the ANNs predict salinity. The optimum number of hidden neurons was found to be 4 for RM6, 5 for RM8, and 6 for RM9 (Table 1). Only 3 input neurons were used in this study to reduce the size of the network and consequently reduce the training time and increase the generalization ability of the network for any given data set.

Table 1. Optimum structure and internal parameters of ANNs.

Location	Number of hidden neurons	Learning rate	Momentum
RM9	6	0.8	0.1
RM8	5	0.5	0.1
RM6	4	0.3	0.1

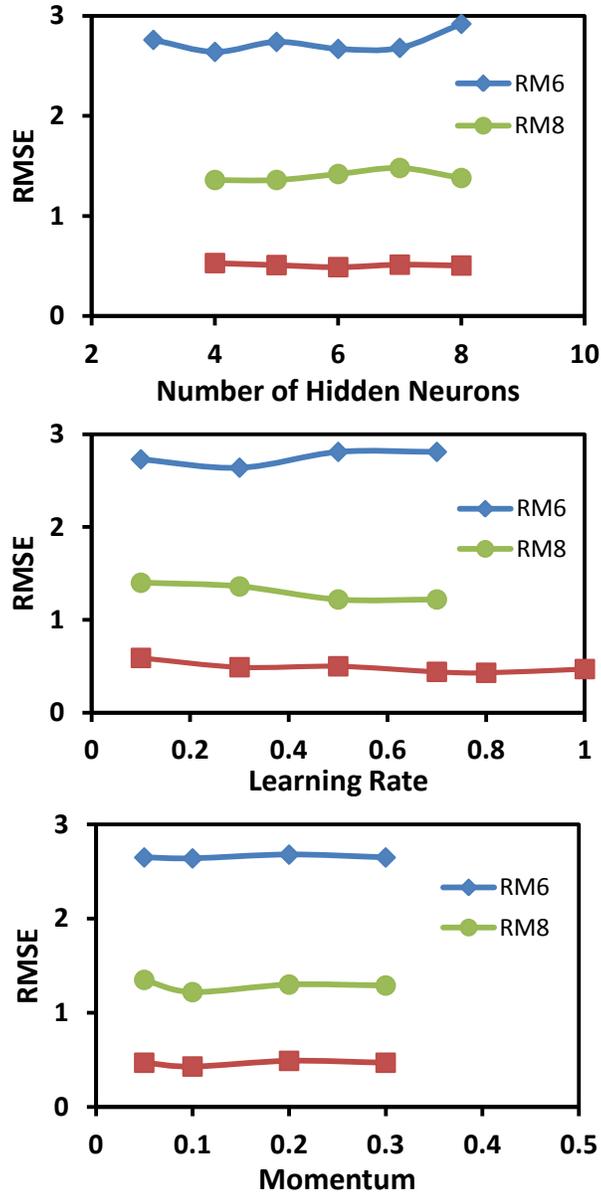


Figure 4. Responses of changing the number of hidden neurons (top), learning rate (middle), and momentum (bottom) represented by the root mean squared error (RMSE).

The optimal learning rate was found to be 0.3 for RM6, 0.5 for RM8, and 0.8 for RM9, again varying with locations (Table 1). Larger learning rates (2-4) were tested and the results indicated that they tended to create over-learning problems with poor model performance for testing data in spite of good correlation with training data. With the optimum number of hidden neurons and learning rate, the momentum of 0.1 provided the best results for all the locations (Figure 3). The learning rate and momentum values obtained in this study were similar to these of Dawson et al. (2005) and Maier and Dandy (1996).

3.2 Model performance

The statistical parameters for training and testing of ANNs are summarized in Table 1. Graphic comparisons between observed and simulated salinities are presented in Figure 4 for visual inspection of the goodness-of-fit. Overall, the performance of ANNs is considered good according to calibration and validation criteria proposed by Dawson and Wilby (2001) for ANNs and Ji (2008) for estuarine hydrodynamic models. For example, the r^2 values ranged from 0.83 at RM9 and RM9 to 0.90 at RM8 during the training period. The RMSE values ranged from 0.41 at RM9 to 2.38 at RM6, with corresponding REE values ranging from 3.96 to 9.17%. During the testing period, r^2 values were smaller (0.57 for RM9, 0.77 for RM8, and 0.7 for RM6) and RMSE and REE values were higher, but they were still in the acceptable range compared with performance of physical hydrodynamic models (Ji, 2008). There were areas where the ANNs did not properly model the correct salinity values. For example, Figure 4a shows that the ANNs over-predicted salinity around November 2009 and then in November 2010 for RM9. Performance at this location during the testing period was not as good as for other locations, possibly due to the fact that the river is mostly fresh in this upstream location and the ANNs did

not capture all the processes triggering salinity spikes properly. Tests of sensitivities as proposed by Maier and Dandy (2001) would be helpful to further elucidate the relationship between flow, rainfall, and tide with salinity.

Table 2. Summary statistics of model performance for training and testing of ANNs.

Location	n	Mean (psu)		Range (psu)		r^2	RMSE (psu)	RRE (%)
		Measured	Modeled	Measured	Modeled			
RM9								
Training	1302	0.64	0.64	0.10- 10.46	0.03- 9.48	0.83	0.41	3.96
Testing	1613	0.41	0.52	0.01- 6.56	0.09- 5.54	0.57	0.44	6.83
RM8								
Training	1304	3.16	3.15	0.01- 16.54	0.04- 14.72	0.90	1.03	6.25
Testing	1555	1.90	1.85	0.05- 11.60	0.05- 10.70	0.77	1.33	11.51
RM6								
Training	1303	10.83	10.83	0.10- 26.10	0.44- 22.60	0.83	2.38	9.17
Testing	1460	9.74	9.74	0.01- 22.35	0.80- 21.68	0.70	2.87	12.85

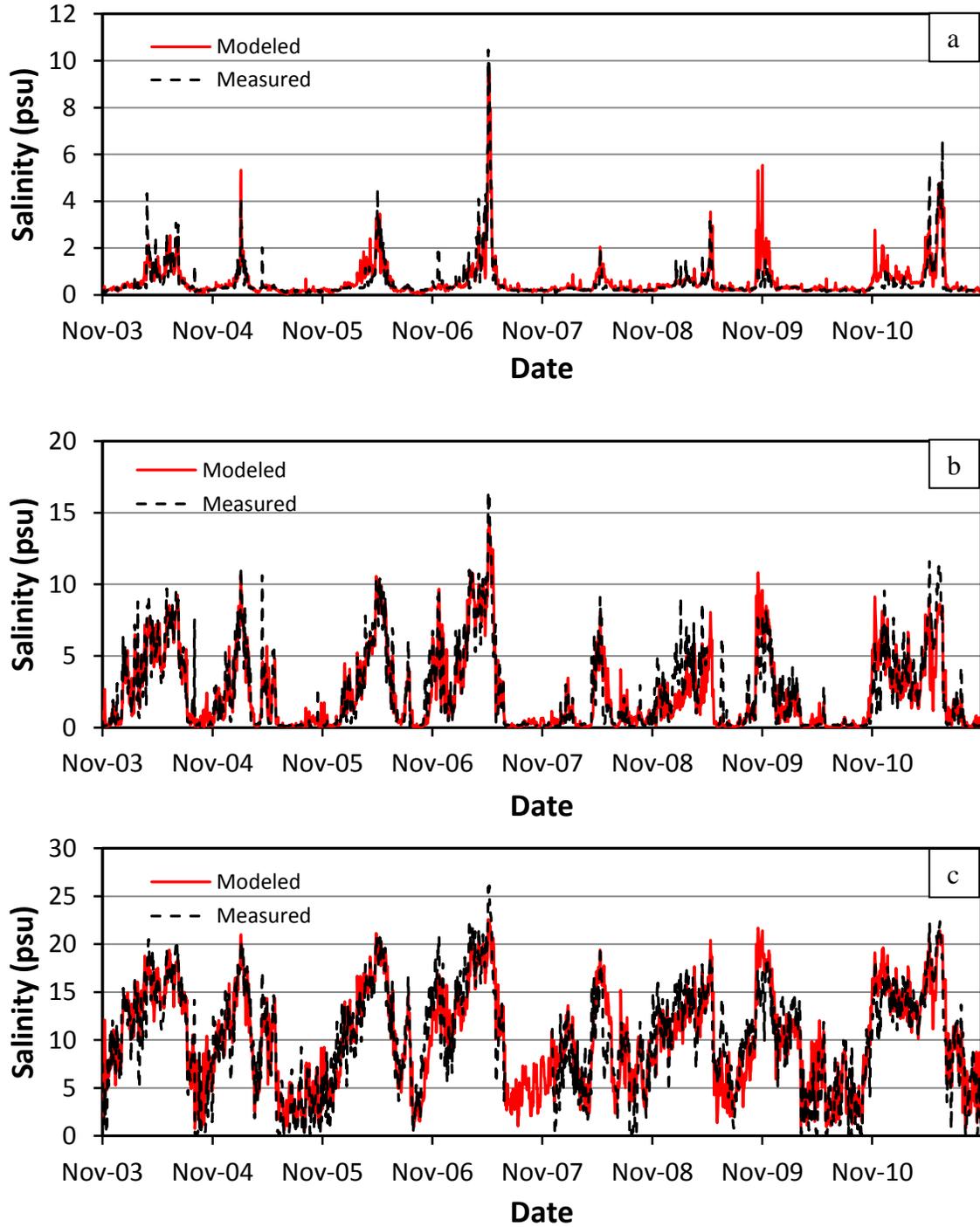


Figure 4. Measured salinity vs. modeled salinity using ANNs at (a) RM9, (b) RM8, and (c) RM6 of the Loxahatchee River during the training period (11/2003~5/2007) and the testing period (6/2007~10/2011).

4.0 Conclusions

Although estuarine salinity is traditionally simulated with multi-dimensional, physically-based hydrodynamic models (Ji, 2008), this study demonstrated for the first time that ANNs can serve as an alternative modeling tool using the multilayer perceptron algorithms newly programmed in Java. The code was written in an object-oriented manner, allowing for flexibility to easily change the structure of ANNs and to carry out multiple tests in a personal computer. Using eight years of data collected at three locations in the Loxahatchee River, the ANNs were successfully trained and tested to simulate salinity using freshwater inflow, rainfall, and tide as inputs. The r^2 values ranged from 0.83 to 0.90 during the training period and 0.57 to 0.77 in the testing period.

While this study demonstrates the ability of using ANNs to model salinity, there are a number of areas for further improvement. First, to make the model more accurate, inputs and output relationship can be further explored using sensitivity analysis (Maier and Dandy 2001). Different lag time of input (inflows and rainfall) can also be tested to determine their influence on model accuracy. Second, different network structures such as a recurrent neural network, radial basis function network, or Kohonen self-organizing network can be programmed in this code to compare with the feed forward network. Third, the model can be applied to determine how meeting the MFL flow target and restoration flow target defined by SFWMD (2002; 2006) would change salinity conditions in the critical locations studied in this paper. Last, with the versatility of the code, the application of the model can be used in other areas of research such as cancer prediction and stock-market evaluation to compare model accuracy.

References

- APHA (2006), Standard Methods for the Examination of Water and Wastewater.
- Dawson, C. W., and R. L. Wilby (2001), Hydrological modelling using Artificial Neural Networks. *Progress in Physical Geography* 25: 80-108.
- Dawson, C.W., R.j. Abrahart, A.Y. Shamseldin, and R.I. Wilby (2006), Flood estimation at ungauged sites using Artificial Neural Networks. *Journal of Hydrology* 319: 391-409.
- Doering, P. H., R. H. Chamberlain, and D.E. Haunert (2002), Using submerged aquatic vegetation to establish minimum and maximum freshwater inflows to the Caloosahatchee Estuary, Florida. *Estuaries*, 25: 1343-1354.
- Lindeboom H. (2002), The coastal zone: an ecosystem under pressure. In: Field J.G., Hempel G. and Summerhayes C.P. (eds), *Oceans 2020: Science, Trends, and the Challenge of Sustainability*, Island Press, Washington, pp. 49–84.
- Liu, G.D., Y.C. Li, R. Munoz-Carpena, M. Hedgepeth, Y.Wan, and R. Robert (2011), Growth of bold cypress (*Taxodium distichum*) seedlings in the tidal floodplain of the Loxahatchee River. *Florida Scientist* 74: 82-99.
- Ji, Z.-G. (2008), *Hydrodynamics and Water Quality: Modeling Rivers, Lakes, and Estuaries*. John Wiley & Sons, Inc., Hoboken, New Jersey.
- Kaplan D., R. Muñoz-Carpena, Y. Wan, M. Hedgepeth, and F. Zheng, R. Roberts, and R. Rossmannith (2010), Linking river, floodplain, and vadose zone hydrology to improve restoration of a coastal river affected by saltwater intrusion. *J. Environ. Qual.* 39:1570–1584
- Maier, H.R. and G.C. Dandy (1996), The use of artificial neural networks for the prediction of water quality parameters. *Water Resources Research* 32: 1013-1022.

- Maier, H.R. and G.C. Dandy (2001), Neural network based modeling of environmental variables: A systematic approach. *Mathematical and Computer Modeling* 33: 669-682.
- Maren, A., C. Harston, and R. Pap (1990). *Handbook of Neural Computing Applications*, 448pp. Academic, San Diego, CA.
- NeuralWare, Inc. (1991), *Neural Computing, NeuralWorks Professional II/Plus and NeuralWorks Explorer*, 360 pp. Pittsburgh, PA.
- Pao, H.-T. (2008), A comparison of neural network and multiple regression analysis in modeling capital structure. *Expert Systems with Applications* 35: 720-27.
- Patel, R. M., S. O. Prasher, P. K. God, and R. Bassi (2002), Soil salinity prediction using artificial neural networks. *Journal of the American Water Resources Association* 38: 91-100.
- Schalkoff, R.J. (1997), *Artificial Neural Networks*. McGraw Hill, New York, pp 146-188.
- SFWMD (2002), *Technical Criteria to Support Development of Minimum Flow and Levels for the Loxahatchee River and Estuary*. SFWMD, Water Supply Department, Water Resources Management, West Palm Beach, FL.
- SFWMD (2006), *Restoration Plan for the Northwest Fork of the Loxahatchee River*. SFWMD, Coastal Ecosystems Division, West Palm Beach, FL.
- Sklar, F.H. and J. A. Browder (1998), Coastal environmental impacts brought about by alterations to freshwater flow in the Gulf of Mexico. *Environmental Management* 22: 547-562.