SCISPACE
formerly Typeset

Open access • Proceedings Article • DOI:10.1145/76738.76828

Neural Network Models In Simulation: A Comparison With Traditional Modeling Approaches — Source link

Paul A. Fishwick

Institutions: University of Florida

Published on: 01 Oct 1989 - Winter Simulation Conference

Topics: Time delay neural network, Physical neural network, Nervous system network models, Deep learning and Probabilistic neural network

Related papers:

- Neural network as a simulation metamodel in economic analysis of risky projects
- An investigation on Neural Network Capabilities as Simulation Metamodels
- Using a Neural Network to Enhance the Decision Making Quality of a Visual Interactive Simulation Model
- Time series forecasting using neural networks vs. Box- Jenkins methodology
- Metamodels for simulation input-output relations

Share this paper:

View more about this paper here: https://typeset.io/papers/neural-network-models-in-simulation-a-comparison-with-1hjs8ylgxa

Proceedings of the 1989 Winter Simulation Conference
E.A. MacNair, K.J. Musselman, P. Heidelberger (eds.)

NEURAL NETWORK MODELS IN SIMULATION: A COMPARISON WITH TRADITIONAL MODELING APPROACHES

Paul A. Fishwick
Department of Computer and Information Science
University of Florida
Bldg. CSE, Room 301
Gainesville, FL 32611
INTERNET: fishwick@fish.cis.ufl.edu

ABSTRACT

Neural models are enjoying a resurgence in systems research primarily due to a general interest in the connectionist approach to modeling in artificial intelligence and to the availability of faster and cheaper hardware on which neural net simulations can be executed. We have experimented with using a multi-layer neural network model as a simulation model for a basic ballistics model. In an effort to evaluate the efficiency of the neural net implementation for simulation modeling, we have compared its performance with traditional methods for geometric data fitting such as linear regression and surface response methods. Both of the latter approaches are standard features in many statistical software packages. We have found that the neural net model appears to be inadequate in most respects and we hypothesize that accuracy problems arise, primarily, because the neural network model does not capture the system structure characteristic of all physical models. We discuss the experimental procedure, issues and problems, and finally consider possible future research directions.

1 INTRODUCTION

Are neural network models useful as simulation models? That is, is it advantageous to replace a continuous or discrete simulation model with a neural network model? This is a very general question and, in this paper, we will attempt to provide an answer based on the results of our small experiment. Clearly, further experimentation is needed to provide a more conclusive response. Our answer, then, based on our experiments is negative — the neural network model (multi-layered or otherwise) provides the systems analyst with a less powerful modeling tool than he already uses in traditional systems investigations. Tools including capabilities such as general linear regression analysis, time series analysis, and system identification analysis are readily available. Our basic tenet when beginning this research was not based on the question: *Can neural networks be used as simulation models?* Rather, we asked *How efficient are neural network models when compared with other, similar methods for modeling, identification, and parameter estimation?* We feel that this distinction is extremely important if we are to assess the value of neural network modeling. There are multitudinous methods for systems and simulation modeling. A test of neural network modeling should be predicated on comparative testing, and not on modeling with a neural network for its own sake.

Multi-layered (i.e. containing hidden layers) neural network models are algebraic equations with nonlinear coefficients. Their use in computer simulation can take one of two forms: *behavioral* or *structural*. With behavioral models, we are interested in building neural nets that model the behavior, but not necessarily the structure, of the physical system under consideration. With structural modeling, we can use the neural network model as a structural model. In this paper, we will spend most of our time discussing the behavioral aspect. We will now, however, briefly mention two structurally inclined simulation models. In terms of structural models, neural networks have shown to be useful in modeling physiological functions of real neurons. Physiological

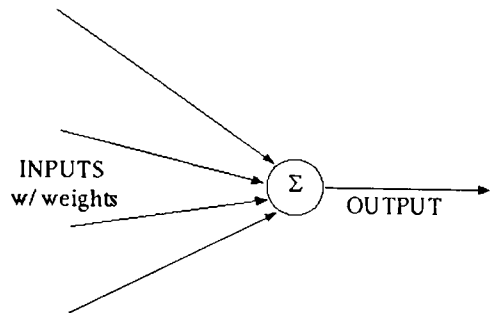


Figure 1: Structural Modeling

modeling is one of the chief reasons for using neural models. Neural models, though, can also be used for non-neurological applications. For instance, we see that it might possibly be used to model a network of entities whose movement is constrained by numerical thresholds. Consider figure 1. This single neuron model can be the basis for modeling an output spooler of a computer operating system — the spooler (neuron) "collects" bytes from many sources (input links) each of which has a different bandwidth (link weight) until it reaches its maximum capacity (threshold), say, at 1024 bytes. When capacity is reached, the spooler spools its output to the printing device (output link). Also, consider the familiar situation where people form in lines waiting to enter a special art exhibit where limited attendance is permitted due to lack of space. In these cases, a guide will wait until she has, say, 20 people in the front of the queue. Only then will she proceed with the people into the exhibit. The threshold is 20 and so the guide waits for the right number of people before she "fires." There are many other structurally related models that can be employed - for instance, some Petri net models can be transformed into the neural network domain by substituting neuron thresholding for the minimum number of input place tokens necessary for transition firing. In behavioral modeling, we can use the neural network to model system behavior. Let us examine this concept. Why might we want to use an algebraic equation to model dynamical system behavior? Certainly it is more natural to use difference or differential equations (or stochastic networks in discrete event modeling). If we have sufficient knowledge about the physical system then we will, indeed, use better modeling methods:

- *Assume Model is Known:* If the physical system and its components lend themselves to experimentation, then we will use models that are familiar to that discipline.
- *Assume Model is Known with Unknown Parameters:* We would use parameter estimation methods based on known models for the system.
- *Assume Model is Unknown:* If a model is completely unknown then we will still hypothesize model structure. For instance, if the system is biological in nature and we are modeling fluid concentrations then exponential models have been shown to be effective. There are many methods for model identification (Sinha 1983, Hsia 1977) based on how much or little is known about the domain and system. One should not necessarily turn to using algebraic equations (as in neural networks) simply because one has a partial or sketchy model.
- *Assume Inputs are Unknown:* Time series methods have been shown to be effective for many predictive and forecasting purposes.

In all of these cases, dynamic discrete system models are based on difference equations of some sort. This seems logical when we consider that dynamic processes involve state variables and dependence on differences in state variables. So we see that using algebraic equations to model system behavior is somewhat odd in itself — solutions will certainly be algebraic but does one "guess" the right algebraic solution for neural net encoding purposes? In many cases, the adding of layers or different transfer functions to a neural network is tantamount to guessing unless one is completely aware of the corresponding equational changes.

Let us consider, though, that we accept the premise of using algebraic equations to model system behavior. To determine the efficiency of such an approach, will must compare the neural network method against other algebraic methods. How do we proceed? In using a neural network to model a system, we train the neural network on input/output pairs obtained via observation and then we use the resulting, trained network to predict new outputs from new inputs. We will consider this approach in the section on experimentation.

Let us briefly discuss terminology in neural network literature. Experience has taught us that

Table 1: Varying Terminology for Similar Functions		
Standard (Numerical)	**Time Series Analysis**	**Neural Networks**
Interpolation	Prediction	Store Information/Memory
Extrapolation	Forecasting	Forecasting
Equation(s)	Equation(s)	Neural Network
Parameter	Coefficient	Link Weight
Geometric Fitting	System Identification	Pattern Recognition
Adaptation	Iteration	Training/Learning

metaphors can be both a boon and a burden when applied to science. In the neural network literature, we are continually confronted with terminology such as "learn," "hidden layer," "training" and so on. From such discussions, one might think that these concepts are entirely novel. We need, though, to look deeper to see that "learning" is adaptation (usually of parameter values), and "hidden layers" are simply arithmetic terms with nonlinear functions of parameters. Using metaphors can help us to create new and creative science but we must be wary of creating new terminology for old concepts. Table 1 displays the relationships in terminology. We must constantly be on guard when we use metaphors so that we don't lose perspective of what calculations and algorithms are actually invoked. With respect to using neural networks, we must study what the metaphors provide us — does "extending a layer" or "creating new feedback loops" provide us with promising equation representations which might otherwise be obscured when viewed from a strict equational perspective (i.e. without the use of the metaphor)? We feel that the neural "metaphor" is not creating novel equations for purposes of system identification – it is simply providing an unstructured way of producing equations that do not reflect physical system structure. Saying that "I'm going to trying adding a layer or a feedback loop" arbitrarily is much like saying "I'm going to arbitrarily raise the powers of some coefficients in my equation." We should strive for more systematic ways of upgrading and refitting our models.

2 RELATED RESEARCH

Within the simulation community, there has been little work done in neural network research; however, most of the current research that does deal with neural networks relates more to using computer simulation to simulate a neural network rather than to study whether or not neural networks provide reasonable alternatives to current methods of continuous and discrete event simulation methodology. Examples of this approach are taken by Bassi and Bekey (1989) and Clarson and Shimp (1989). Wildberger (1989) is actively studying the use of neural networks as one tool in enhancing power plant performance. Other researchers have concentrated on studying various software tools for effective neural network simulation (Rumelhart et al. 1986, D'Autrechy et al. 1988). Within the domain of signal processing we find more evidence of neural net research. For instance, we view the work of Lapedes and Farber (1987) and Farmer and Sidorowich (1988) as being closely related to our work. Lapedes and Farber have found the neural network methodology to work well for identifying system behavior by training the net on time series data; however, we are unsure how their method compares against more integrated modeling methods incorporating the use of correlograms, periodograms, Box-Jenkins and other time series methods.

3 EXPERIMENTS

Our approach to studying the effectiveness of neural networks is based on experimentation with statistical inference. We are not convinced of any other method that will provide the scientific community with some sort of proof as to the effectiveness of neural modeling methods as compared with traditional methodologies. We admit that our experiments do not cover all factors including alternate neural net control strategies and modeling formalisms (BAM, ART, etc.), yet we must start with base cases and then move on slowly.

3.1 Tools for Experimentation

As a basis for experimentation, we used a variety of software packages for cross-checking of results and to extend our capabilities. These packages are depicted in figure 2. In lieu of actual ballistics experimentation, we used a standard differential equation

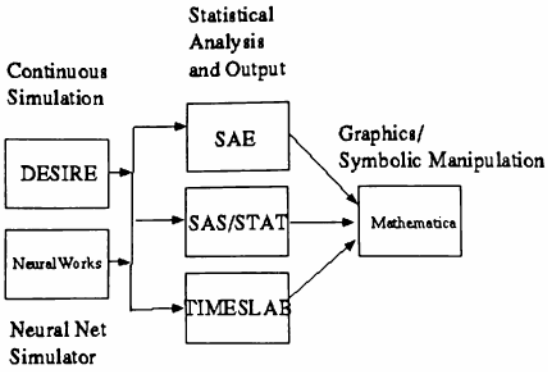


Figure 2: Experiment Setup

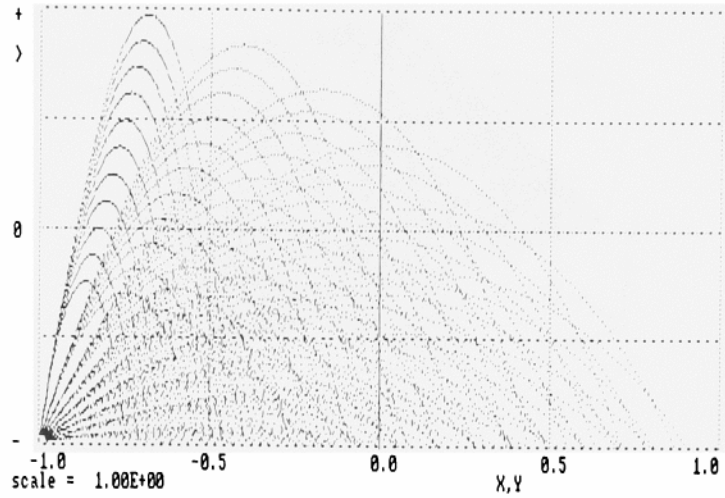


Figure 3: Trajectories Used for Gathering Data

that is known to be an accurate model of simple ballistics incorporating air resistance. We used the DESIRE program (Korn 1989) to model the differential equations and obtain varying results based on several initial conditions. Results from DESIRE were used to train a neural network which was implemented in Neural Wares' product called NeuralWorks (Klimasauskas 1988). Results from DESIRE and NeuralWorks were also translated into several statistical packages for analysis: SAS (SAS Institute 1988), SAE (Barnes 1988), and TIMESLAB (Newton 1988). We found it useful to cross-check results using different packages. Most graphical output was created using the Mathematica package (Wolfram 1988) and some output was dumped directly from the computer screen. We encountered some difficulties in translating from one package to another but these were relatively minor. For instance, one package could not accept data in scientific notation (i.e. using exponentiation) whereas another package could generate data only in scientific notation.

3.2 The Ballistics Model

We chose a ballistics model since this model is easy enough to understand but difficult to analyze using non-numerical methods. The equations use Cartesian space and are listed as (1) through (5). Equations (1) and (2) are initial conditions for component velocity while (3) through (5) comprise the equations of motion.

$$\dot{x}_0 = v_0 cos(\theta) \quad (1)$$

$$\dot{y}_0 = v_0 sin(\theta) \quad (2)$$

$$\ddot{x} = -kv\dot{x} \quad (3)$$

$$\ddot{y} = -kv\dot{y} - g \quad (4)$$

$$v = (\dot{x}^2 + \dot{y}^2)^{\frac{1}{2}} \quad (5)$$

Note that $\dot{x} = vcos(\theta)$ and $\dot{y} = vsin(\theta)$ represent the components of the velocity vector. This equation set represents a model of a cannon ball (or missile) fired from height $y = 0$. The gravitational constant is $g = 32.2$. The ball encounters air resistance (specified by the constant $k = 7.5 \times 10^{-5}$) that is proportional to velocity squared. The model without air resistance has a fairly straightforward solution; however, with air resistance proportional to velocity squared, the equations are not-amenable to closed form solution (Hart et al. 1988).

3.3 Experimental Procedure and Results

The first step was to solve the differential equations of motion to obtain a table of values for the variables θ (initial angle) , v_0 (initial velocity), and h (horizontal distance that the projectile travels until it hits the ground). h was determined by coding DESIRE to mark the value of x when y crossed zero. θ and v_0 are inputs while h is the single output. Angle θ was provided in increments of 9 degrees from 0 to 81 degrees. Velocity v_0 was provided in increments of 5 from 0 to 100 feet/sec. This resulted in 210 total observations. The output h ranged from 0 to 300.14 feet. Figure 3 depicts the trajectories that were formulated for purposes of gathering data on θ, v_0 and h.

The data appears in graphical form in figure 4. Figure 4(a) shows the isometric view while figure 4(b) shows a front view. Notice that, as expected, h increases as we increase v_0 while keeping the angle roughly between 30 and 60 degrees. 'ODE' refers to the Ordinary Differential Equation model used to obtain the data.

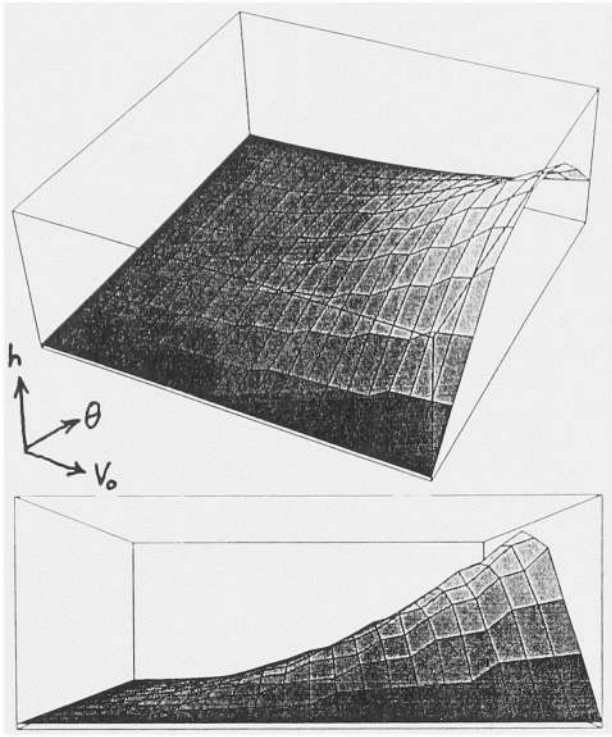


Figure 4: (a) Original Ballistic Data: ODE, and (b) Front View

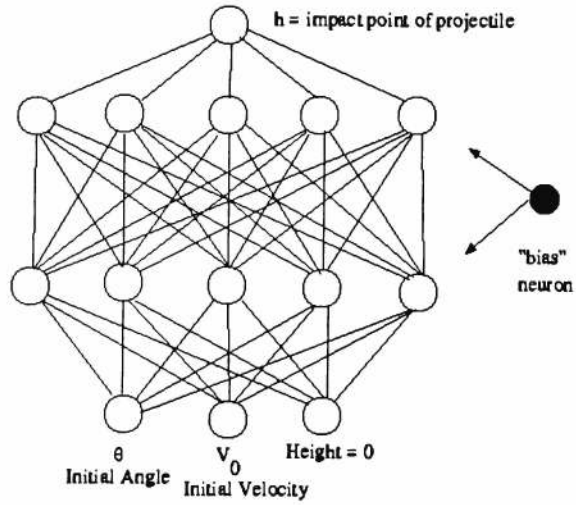


Figure 5: Neural Network for Ballistics Simulation

3.4 Neural Network Model

We based our neural network on a ballistics simulation included in the NeuralWorks package which was, in turn, inspired from networks used by Lapedes and Farber (1987) in their paper on modeling nonlinear time series using neural networks. The network, shown in figure 5, is defined as a feed-forward network with an input and output layer, and two hidden layers. The input and output layers have linear activation functions whereas the hidden layers use a sigmoidal activation function to pre-process their output.

This network is equivalent to a set of nonlinear equations (due to the sigmoidal activation function *tanh*) as discussed by Farmer and Sidorowich (1988). The two inputs to our neural net are θ and v_0. The third input is inactive, for simplicity, since it represents initial height which is set to zero. The single output is horizontal distance h traversed by the projectile. The neural network equations are listed as (6) through (10). Equation (6) represents the output layer with single output h. Equations (7) and (8) represent the hidden layers, and (9) and (10) represent the two inputs to the neural network.

$$h = \sum_k w_k z_k - a_0 \quad (6)$$

$$z_k = tanh(\sum_j w_j y_j - a_k) \quad (7)$$

$$y_j = tanh(\sum_i w_i x_i - a_j) \quad (8)$$

$$x_1 = \theta \quad (9)$$

$$x_2 = v_0 \quad (10)$$

When viewed in an equational manner, it is easier to see what is meant by terms such as "layer" and "activation function." Without the sigmoidal curve (i.e *tanh*), the equations could be expressed simply as a single linear equation with nonlinear weights. The addition of the sigmoidal activation (or transfer) function makes a single equation representation more complex.

Neural Ware has multiple options for control and learning strategies — we chose to use the same options that they did: backpropagation. We retrained the network to learn for 180,000 iterations. Then, once the patterns were "stored," we used the network to predict values for h based on the original values for θ and v_0. Figure 6 displays the original ODE surface (which is re-displayed for comparison) and the approximating surface created by executing the neural network (NN). Figure 7(a) shows a different view of the same neural network surface (from the front) and figure 7(b) shows the residual error plot for each of the 210 observations.

3.5 Linear Regressive Model

For comparison with the neural network approximation, we chose a linear model:

$$\hat{h} = \beta_0 + \beta_1\theta + \beta_2 v_0$$

We used SAS, SAE and TIMESLAB for this purpose and obtained the following parameter estimates:

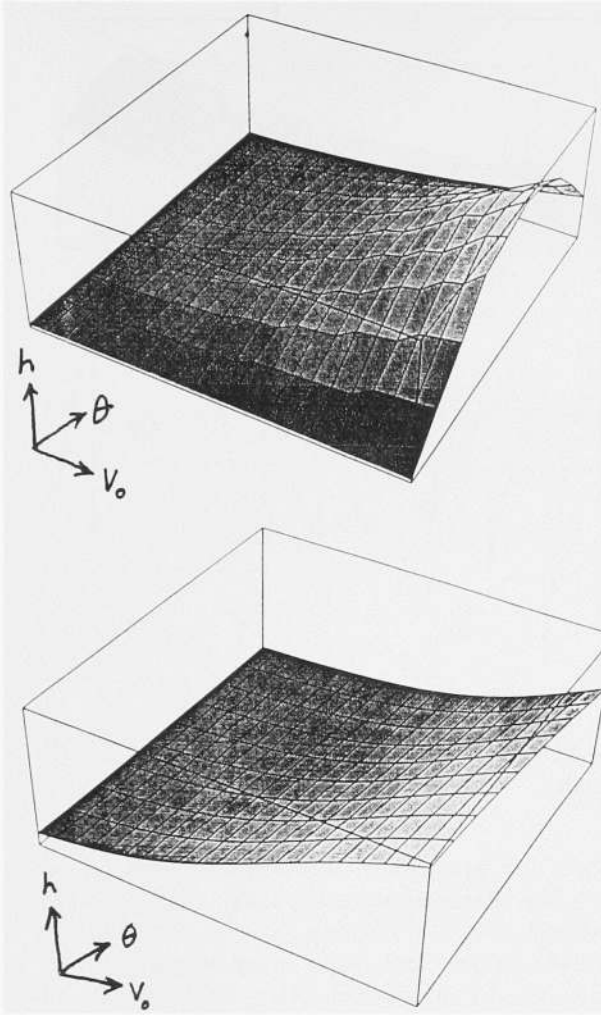


Figure 6: (a) Original ODE Surface, and (b) NN Surface: Isometric View

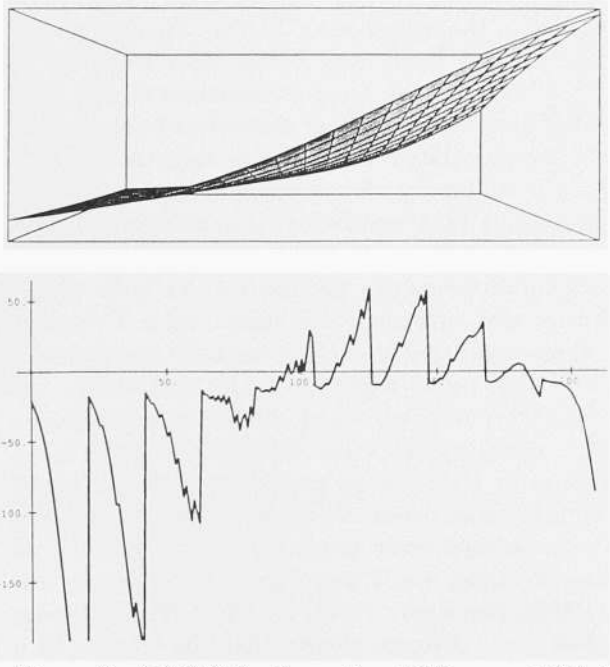


Figure 7: (a) NN Surface: Front View, and (b) NN Residuals

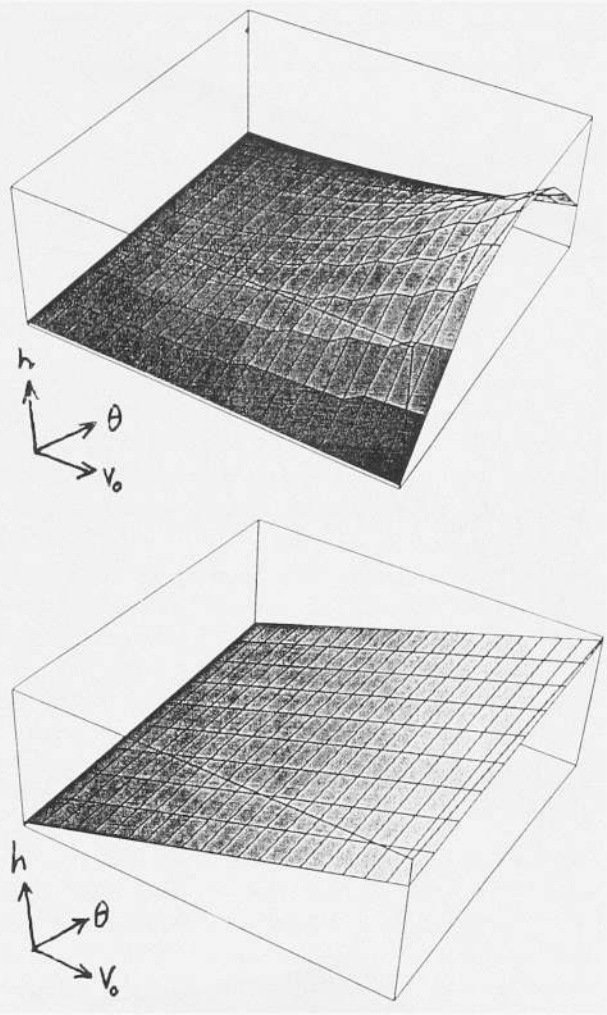


Figure 8: (a) Original ODE Surface, and (b) LR Surface: Isometric View

$\beta_0 = -48.972$, $\beta_1 = 0.45177$, and $\beta_2 = 1.9264$. Linear Regression (LR) resulted in an approximation surface shown in figures 8 and 9.

3.6 Surface Response Model

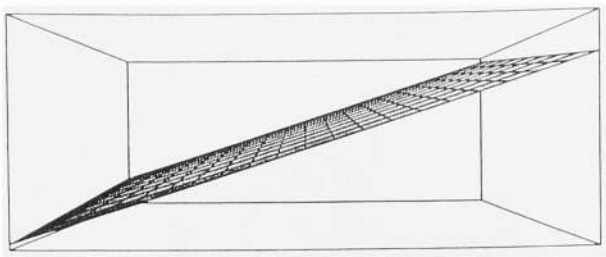
The surface response model was chosen to determine if certain factors played a part in the function $\theta \times v_0 \rightarrow h$. The response model is defined as:

$$\hat{h} = \beta_0 + \beta_1\theta + \beta_2 v_2 + \beta_3\theta^2 + \beta_4 v_0^2 + \beta_5\theta v_0$$

We obtained the following parameter estimates: $\beta_0 = -43.2773$, $\beta_1 = 4.0509$, $\beta_2 = -0.5436$, $\beta_3 = -0.05239$, $\beta_4 = 0.01289$, and $\beta_5 = 0.01927$. Figures 10 and 11 display the surface response (SR) approximation to the original data.

3.7 Analysis of Variance: What We Learned

We learned that the neural network was a relatively poor approximation method given the con-



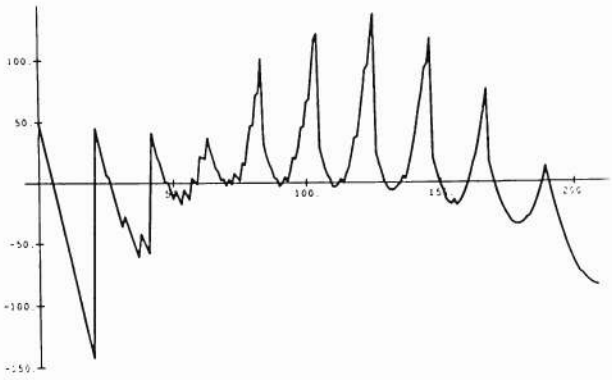
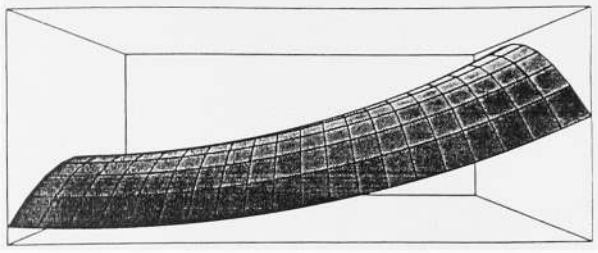


Figure 9: (a) LR Surface: Front View, and (b) LR Residuals



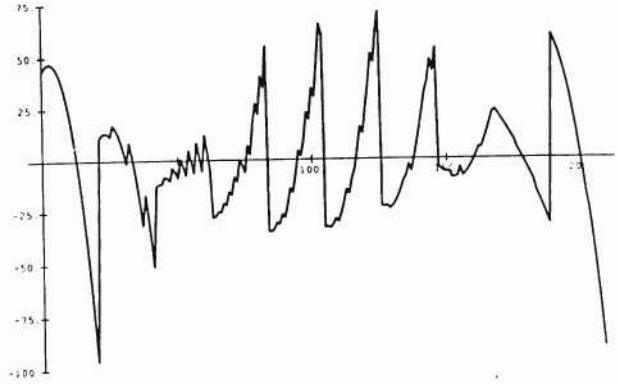


Figure 11: (a) SR Surface: Front View, and (b) SR Residuals

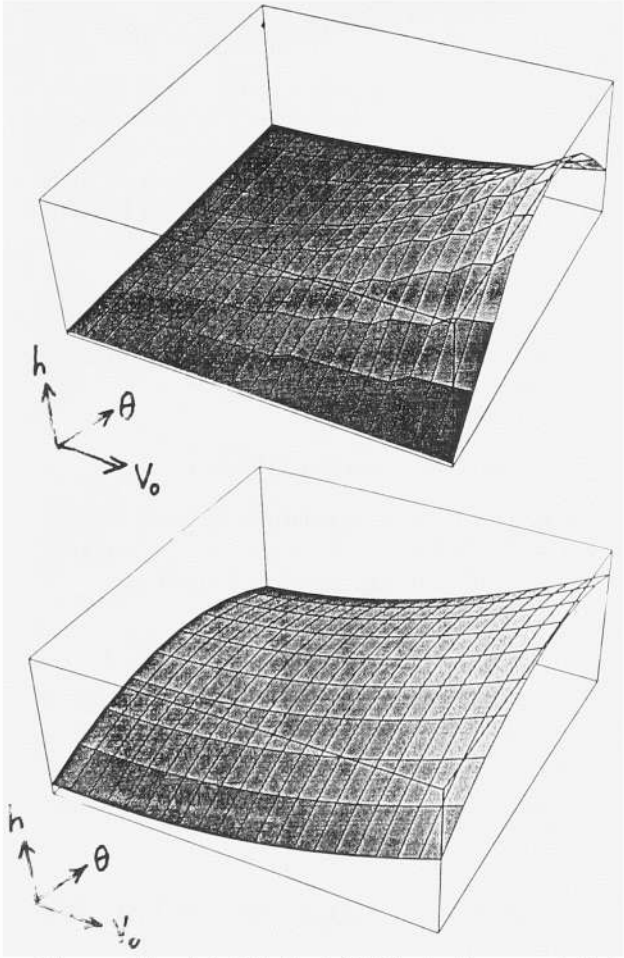


Figure 10: (a) Original ODE Surface, and (b) SR Surface: Isometric View

straints of our experiment. We have not; however, explored other alternatives (at the time of this writing) such as adjusting the learning rate, adjusting initial link weight values, and using other learning methods and neural model types. Also, we would like to run further experiments which involve increased total learning times (beyond the 180,000 iteration count).

Simple linear regression outperformed a neural network (with root mean square error (RMSE) of 46.47 versus the neural net's 70.85). The surface response method fared even better with a RMSE of 29.19. We stress that using geometric (i.e. non time specific) methods for system identification seem generally inappropriate if at least some structural knowledge is available for the physical system. However we are saying that, even when this severe limitation is accepted, traditional geometric methods were still shown to have superior performance in both execution time and approximation value. Table 2 displays the three compared modeling methods (neural networks (NN), linear regression (LR) and surface response (SR)) on the basis of residual sum of squares (RSS), mean square error (MSE), and root mean square error (RMSE). A ranking is provided based on RMSE comparisons. To evaluate the relationships between residual error and independent variable, we plotted θ versus RMSE (see figure 12(a)) and v_0 versus RMSE (see figure 12(b)) for the surface response method. We see clearly that the error is at a minimum for $40 \leq v_0 \leq 80$ and for approximately $10 \leq \theta \leq 30$ and $50 \leq \theta \leq 70$. If our simulation were to use the surface response model as an approxima-

tion to the ODE model, we would obtain reasonable results for h. We postulate that the NN and LR models would produce similar intervals of minimal error.

Table 2: Statistical Results				
Model	RSS	MSE	RMSE	RANK
NN	1,039,188.6	5020.2	70.85	3
LR	447,019.2	2159.5	46.47	2
SR	173,884.2	852.4	29.19	1

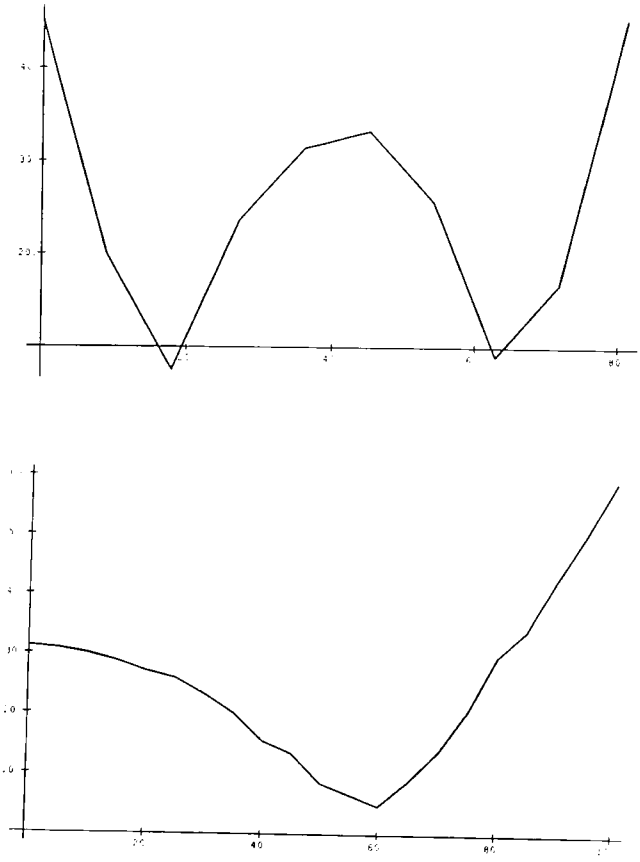


Figure 12: (a) Residuals for values of θ, and (b) Residuals for values of v_0

4 CONCLUSIONS

When a system is being modeled we must use all available knowledge about the domain and potential variables. Even when we appear to have little knowledge about a system, we usually have enough to assume a basic system structure (often a canonical form) and then we can estimate the parameters. If we have outputs then, at the very least, we can use the vast array of time series methods to analyze the system and perform prediction and forecasting. Are neural networks good at forecasting? We do not address this in this paper; however, we have little reason to believe that an algebraic equation (modeled as a neural network) will outperform a model based on autoregressive (i.e. ARIMA) modeling which explicitly models time dependent variables.

Given that we have little knowledge about the system other than the available data, we can use a neural network model to store the input/output pairs for later retrieval. However; as we have seen in this paper, the neural net method for our example does not compare favorably with standard methods such as linear regression and surface response methods. We strongly suggest that other experiments similar to ours take place so that we can learn more about the strengths and weaknesses of neural nets as simulation models. Due to space limitations, we have not explicitly discussed a discrete event model, although, we speculate that there will be similar problems with neural nets; discrete event models may also be partitioned into inputs (interarrival and service times) and outputs (queue departure time). One small experiment does not allow us to induce that all neural networks are poor for all simulation models, but it provides us with a yardstick for initial comparative purposes.

ACKNOWLEDGMENTS

We would like to thank the Florida High Technology Council for partial funding during this research period.

REFERENCES

J. Wesley Barnes. (1988). *Statistical Analysis for Engineers: A Computer-Based Approach.* Prentice

Hall.

Danilo F. Bassi and George A. Bekey. (1989). Decomposition of Neural Network Models of Robot Dynamics: A Feasability Study. In Wade Webster, editor, *Simulation and AI*, volume 20, pages 8 - 13. Society for Computer Simulation (SCS).

Virginia H. Clarson and James E. Shimp. (1989). Simulation of a Neural Network Solution to the Media Selection Problem. In Wade Webster, editor, *Simulation and AI*, volume 20, pages 14 - 19. Society for Computer Simulation (SCS).

C. Lynne D'Autrechy, James A. Reggia, Granger G. Sutton, and Sharon M. Goodall. (1988). A General-Purpose Simulation Environment for Developing Connectionist Models. *Simulation Journal*, 51(1), July.

J. Doyne Farmer and John J. Sidorowich. (1988). Exploiting Chaos to Predict the Future and Reduce Noise. In Y. C. Lee, editor, *Evolution, Learning, and Cognition*. World Scientific Press.

Derek Hart and Tony Croft. (1988). *Modelling with Projectiles*. Halsted Press (a division of John Wiley), New York.

T. C. Hsia. (1977). *System Identification*. Lexington Books, Lexington, MA.

SAS Institute. (1988). *SAS/STAT User's Guide*. SAS Institute. Release 6.03 Edition.

Casimir C. Klimasauskas. (1988). *NeuralWorks User's Manual*. Neural Ware Inc., Pittsburgh, PA.

Granino A. Korn. (1989). *Interactive Dynamic System Simulation*. McGraw Hill.

Alan Lapedes and Robert Farber. (1987). Nonlinear Signal Processing using Neural Networks: Prediction and System Modeling. Technical Report LA-UR-87-2662, Los Alamos Research Laboratories. (submitted to Proceedings of the IEEE).

H. Joseph Newton. (1988). *TIMESLAB: A Time Series Analysis Laboratory*. Wadsworth and Brooks, Pacific Grove, CA.

David E. Rumelhart and James L. McClelland. (1986). *Parallel Distributed Processing: Volume III (Explorations)*. MIT Press.

N. K. Sinha and B. Kuszta. (1983). *Modeling and Identification of Dynamic Systems*. Van Nostrand Reinhold.

A. Martin Wildberger. (1989). Application of Expert Systems, Simulation and Neural Networks Combined to Enhance Power Plant Performance. In *Proceedings of the AI and Simulation Workshop*. AAAI, August.

Stephen Wolfram. (1988). *Mathematica: A System for Doing Mathematics by Computer*. Addison Wesley.

AUTHOR'S BIOGRAPHY

PAUL A. FISHWICK is an Assistant Professor in the Department of Computer and Information Sciences at the University of Florida. He received the BS in Mathematics from the Pennsylvania State University, MS in Applied Science from the College of William and Mary, and PhD in Computer and Information Science from the University of Pennsylvania in 1986. He also has six years of industrial/government production and research experience working at Newport News Shipbuilding and Dry Dock Co. (doing CAD/CAM parts definition research) and at NASA Langley Research Center (studying engineering data base models for structural engineering). His research interests are in computer simulation modeling and analysis, systems science and artificial intelligence. He has published a number of journal articles in the topics of process abstraction in modeling, the use of natural language as a simulation modeling medium, and qualitative simulation. He is a member of IEEE, IEEE Society for Systems, Man and Cybernetics, IEEE Computer Society, The Society for Computer Simulation, ACM, AAAI, and IMACS. He is chairman of the IEEE Computer Society Technical Committee on Simulation (TCSIM) which has one thousand members worldwide and publishes SIMULATION DIGEST in conjunction with ACM SIGSIM. He is also an associate editor for SIMULATION DIGEST.