*Proceedings of the 2012 Winter Simulation Conference*
*C. Laroque, J. Himmelspach, R. Pasupathy, O. Rose, and A.M. Uhrmacher, eds*

# Metamodeling of Simulations Consisting of Time Series Inputs and Outputs

Scott L. Rosen
Christopher P. Saunders
Samar K. Guharay

The MITRE Corporation
7515 Colshire Drive
McLean, VA 22102, USA

## ABSTRACT

Long run times of a simulation can be a hindrance when an analyst is attempting to use the model for timely system analysis and optimization. In this situation, techniques such as simulation metamodeling should be considered to expedite the end user's intended analysis procedure. A difficult problem arises in the application of metamodeling when the simulation inputs and outputs are not of a single value, but constitute a time series, a phenomenon that is seen repeatedly in the area of financial simulations and many naturally occurring events. This paper provides a method to develop a mapping between multiple time series inputs of a simulation and a single Figure of Merit (FoM) of the system across a given time period of interest. In addition, this paper discusses a means for an end user to define a tailored FoM with respect to their own specific system beliefs and objectives in the case of multiple simulation outputs.

## 1 INTRODUCTION

Simulation is a valuable tool towards the evaluation of different alternatives, designs, or technological additions to a system. It supports the systems engineering analysis process of mapping component level Measures of Performance (MoP) to system-level Measures of Effectiveness (MoE) to a single Figure of Merit (FoM) of the overall system performance. However, the modeling approaches practitioners use are becoming more and more complex resulting in the set-up and run times of these simulations becoming too long to be used in many analysis situations. The MoP-MoE-FoM mapping procedure in Rosen et al. (2012) integrated a metamodeling procedure to advance MoP-MoE-FoM mappings for real-time analysis and amend them for easier distribution to multiple users, remedying file size and licensing issues inherent in simulation.

During the development of this metamodeling approach, the problem of mapping time series system inputs or MoPs to system level MoEs was encountered. This led to the problem of applying simulation metamodeling on simulations with time series inputs and/or outputs. Time series inputs and outputs occur naturally in geophysical events, simulations involving financial markets as well as simulations pertaining to economic stability or risk. More notably, there has been an increasing trend in agent-based simulations of financial markets that attempt to model the effect of trader behaviors on the overall market dynamics. The agent-based models of equity stock first began with the Santa Fe Institute market model (Arthur et al. 1996) and have evolved in agent behavior rules to the widely referenced Ghoulmie Heterogeneous Trading Model (Ghoulmie 2005). The simulation of these agent-based models lends complexities due to several factors. First, the end user of these models must use the entire time series in order to analyze system behavior. Moreover, the financial system simulations can have multiple MoEs or system performance measures, making it quite difficult for the user to analyze the outputs stemming from the model.

**978-1-4673-4781-5/12/$31.00 ©2012 IEEE**

A wealth of metamodeling families is available to initially consider for this time series metamodeling problem. Notable metamodeling families include Response Surface Models (Myers 1976, Box and Draper 1987), Splines (deBoor 1978, Myers et al. 1996), Radial Basis Functions (Dyn et al. 1986, Meghabghab 2001), Spatial Correlation Metamodels / Kriging (Sacks 1989, Kleijnen 2009), and Neural Networks (Lippman 1987, Al-Hindi 2004). However, it is not immediately apparent how these approaches can be applicable to situations where the simulation consists of time series inputs and/or outputs as in the case of financial markets as discussed above.

This paper provides an approach for simulation metamodeling with simulations consisting of time series inputs and outputs. As many financial simulations contain multiple time series outputs, the approach also considers the situation of aggregating multiple simulation outputs to provide the user with a single FoM value for any time period of interest. Section 2 outlines the technical problem and the framework for our approach. Section 3 outlines a Multiattribute Value Function (MAV) approach to be used to define the overall FoM relevant to the analyst's preferences in the case of multiple simulation model outputs. Section 4 then presents a Neural Network metamodeling procedure to capture the behavior of the financial simulations, which coupled with the MAV function, creates a means to rapidly and efficiently quantify the FoM on any time period of interest. Section 5 illustrates these ideas in a case study involving the metamodeling of a variant of the Ghoulmie (2005) model. Conclusions are provided in Section 6.

## 2 PROBLEM AND APPROACH

A two-step procedure is proposed to map multiple simulation input time series to a FoM that is a single numerical value from an omnibus time series for a point of interest *t*. The schematic below (Figure 1) illustrates this two-step approach. There are two points not explicitly depicted by this schematic. First, since the MAV model provides an output for a particular point on the time series *t*, the MAV model must be sampled repeatedly across the time series to generate the omnibus time series for the FoM. In addition, the Neural Network becomes the surrogate model of both the simulations and MAV function in union.

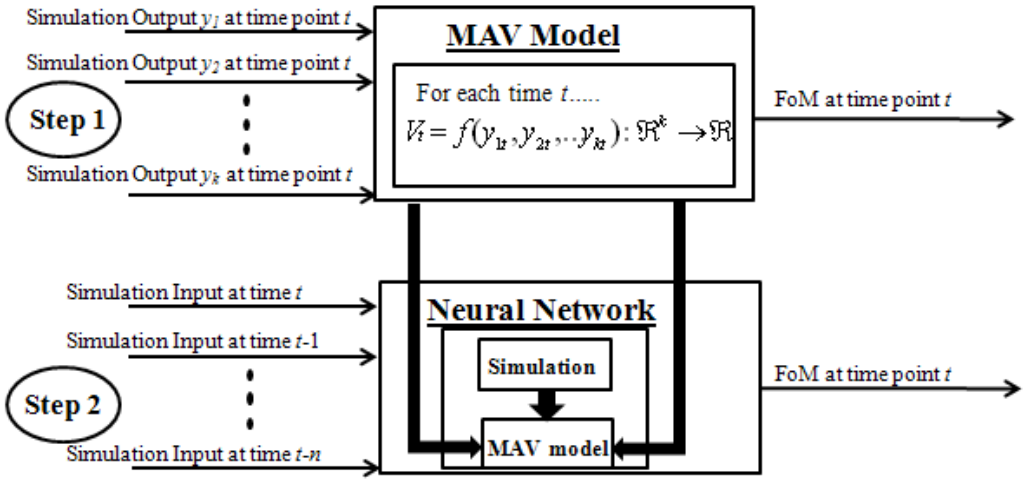


Figure 1: FoM Estimation via Preference Modeling and Metamodeling of Time Series Simulation Data

The first step involves the assessment of a closed-form Multiattribute Value (MAV) function (Dyer and Sarin 1979) through interaction with the model end user to quantify relative importance between the simulation model outputs with regard to the FoM or overall system effectiveness. The MAV function $V_t$ enables the aggregation of multiple simulation outputs $y_1, y_2, ..., y_k$ into a single aggregated time series. The second step consists of a Neural Network metamodel to model the relationship between the time se-

ries inputs from the last *n* points in time, $t, t-1, ..., t-n$ to the simulations and the MAV single-valued response (FoM) from the corresponding simulation outputs at time $t$. The Neural Network is then sampled at the time period of interest to project the FoM for that time period. A whole time series involving the FoM can then be computed by resampling the Neural Network at each point along the time series.

## 3 MULTI-ATTRIBUTE VALUE MODEL FOR OUTPUT AGGREGATION

In our approach for metamodeling, the Figure of Merit (FoM) can be interpreted as the overall level of preference a decision maker has towards a set of MoE values, which in our paradigm are analogous to outputs from a simulation. A Multiattribute Value (MAV) function (Dyer and Sarin 1979) approach is selected for use over Multiattribute Utility (MAU) theory (Von Neumann and Morgenstern 1947) due to its more simplified assessment procedure. This was earlier applied by Rosen et al. (2006), and it was shown to provide more accurate decision models than those generated through a traditional MAU function.

A MAV function in multi-linear form, as shown in Equation (1), is used to map levels of *n* MoE expected values and corresponding standard deviations to a single-valued FoM on [0,1].

$$V(y) = \sum_{i=1}^{2n} w_i v_i(y_i) + \sum_{i=1}^{2n}\sum_{j>1}^{2n} w_{ij} v_i(y_i) v_j(y_j) + \sum_{i=1}^{2n}\sum_{j>i}^{2n}\sum_{l>j}^{2n} w_{ijl} v_i(y_i) v_j(y_j) v_l(y_l) + ..... \quad (1)$$
$$..... + w_{123...2n} v_1(y_1) v_2(y_2) ... v_{2n}(y_{2n}) + \varepsilon$$

The variables $y_i$ can represent levels of either expected value or standard deviation for a particular performance measure or output from the simulation or MoEs. The basis functions $v_i(y_i)$ represent value functions for an expected value or standard deviation corresponding to a specific simulation output. The coefficients $w_i$ can be interpreted as weights representing the expected change in the decision maker's FoM value to a change in $y_i$. The error term $\varepsilon$ represents the error in assessing the decision maker's preferences.

The basis value functions pertaining to standard deviation do not always need to be calibrated and can be omitted when the level of uncertainty in the simulation's output MoE values are not fully understood or when uncertainty in an MoE value has insignificant effect on the overall FoM. There is an assessment procedure for the proposed MAV Function that is applicable to this domain. For further details on this assessment procedure and how it supports a generic MoP-MoE-FoM process, see Rosen et al. (2012). The MAV model sets up a generalized framework for this approach, however, the focus of this paper is on the metamodeling of time series simulations via Neural Networks, which is further discussed in the upcoming section.

## 4 NEURAL NETWORK TIME SERIES METAMODEL

Previous research proposed the use of Neural Networks for time series forecasting; a couple of examples include Faraway and Chatfield (1996) and Gheyas and Smith (2009). Neural Networks' ability to incorporate large input sets makes them malleable for time series based modeling. Coupling this with their known capability of capturing the behavior inside complex simulations (Barton and Meckesheimer 2006) makes them an intriguing selection for metamodeling of simulation outputs that consist of a system of time series. Hill et al. (1996) also argue that Neural Networks are potentially less subjective to the limitations of traditional statistical time series methods, such as, having to expertly specify the functional form relating to input and output variables, making data transformations, and other evaluations requiring expert user interaction.

The basic premise for time series metamodeling with Neural Networks is to consider each index of past observations of the FoM time series to be an individual input to the Neural Network along with the

time series simulation inputs $x_t, x_{t-1}, .., x_{t-n}$ as well as single-valued simulation inputs $s$. In the Heterogeneous Trading Model, the time series inputs $x_t, x_{t-1}, .., x_{t-n}$ can represent the news signal level over time and the single-valued inputs $s$ can represent the number of traders participating in the market, for example. We also utilize the past FoM observations $y_{t-1}, y_{t-2}, .., y_{t-n}$ as inputs to the Neural Network. The basic function mapping for a given time $t$, with error term $\varepsilon_t$, becomes:

$$y_t = f(y_{t-1}, y_{t-2}, .., y_{t-n}, x_t, x_{t-1}, .., x_{t-n}, s) + \varepsilon_t \tag{2}$$

By following this structure, one can also integrate multiple time series into the input layer of the Neural Network. The Neural Network output is the value of the time series at point *t*, which can be the next period in time or a period further in the future. To model a complete time series for multiple points in time, the Neural Networks is resampled for each of the indices of the time series.

The remainder of this section provides additional detail on the proposed single hidden layer structure of Neural Networks for time series metamodeling. Section 4.2 provides a discussion on an appropriate space-covering experimental design alternative.

### 4.1 Structure and Training of the Neural Network Mapping

A single hidden layer, feed-forward structure has been the most widely attempted Neural Network structure for time series forecasting (Zhang et. al 1998). Therefore, we have implemented this structure for the time series metamodeling problem. Under a single layer, the mathematical relationship between the inputs of the network $y_{t-1}, y_{t-2}, .., y_{t-n}$ and $x_{t-1}, x_{t-2}, .., x_{t-n}$ and output $y_t$ can be simplified as follows:

$$y_t = \alpha_0 + \sum_{j=1}^{h} \alpha_j \theta\left( \beta_{0j} + \sum_{i=1}^{n} \beta_{ij} y_{t-i} + \sum_{i=0}^{n} \beta_{ij} x_{t-i} \right) + \varepsilon_t \tag{3}$$

In this expression, $h$ is the number of hidden nodes and $\alpha$ and $\beta$ represent the connection weights of the network. The transfer function of the hidden layer neurons of the network pertains to $\theta(\bullet)$, which can be comprised of more complicated forms. Figure 2 presents a graph of the single hidden layer, feed-forward Neural Network structure.

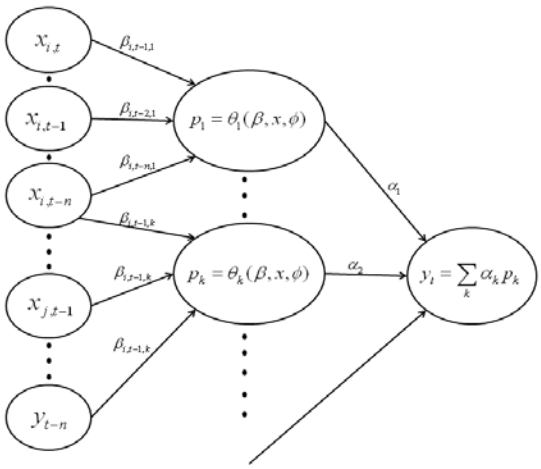


Figure 2: Single Layer Neural Network for Time Series Metamodeling

The input layer is the column of nodes on the left-hand side of the network. Each node pertains to a single value on an individual time series *i* for *n* periods before the current time *t*, which is noted by $x_{i,t-n}$. Note that the nodes in the input layer can represent a simulation input $x_{i,t-n}$ or a past observation of the FoM time series $y_{t-n}$. The hidden layer is contained in the middle column of nodes, which constitute a transformation from some subset of the input nodes in the network through weight terms $\beta_{i,t-n,k}$ with subscript *i* referring to the time series of simulation input *i*, *t-n* referring to the time index within the time series and *k* referring to the hidden layer node the input node is mapping to.

Within each node in the hidden layer is a threshold function $p_k = \theta_k(\beta, x, \phi)$. We apply a sigmoid form here controlled by a threshold $\phi$. The network then maps each of the function outputs $p_k$ to a single-valued numerical output $y_t$, which is targeting the value of the FoM time series at time $t$. Training or calibrating the network involves the use of sampled simulation runs and solving for the coefficients $\beta_{i,t-n,k}$ and $\alpha_k$ that minimize the error between the MAV function outputs and $y_t$. Multiple methods have been proposed to solve for these weights and they remain applicable in this time series case. The customized backpropagation method (Man-Chung et al. 2000) works effectively for time series Neural Networks. This is what is applied in the case study discussed in section 5.

### 4.2 Experimental Design for Training Neural Network

We employed a generalized experimental design for training Neural Networks of time series data. The design is based on a Latin Hypercube Design (LHD) (Kleijnen et al. 2005) and consists of random yet relatively even sampling of points on the interior of the design space via a LHD structure. Van Dam et al. (2009) discussed the merits of LHD for approximating computational simulation models.

In the Latin Hypercube design, for *r* runs and *m* simulation inputs, we will be dealing with an *r* x *m* design. In this type of design, each factor appears only once with each of the *r* equally spaced levels. LHD permutation within the space defined by each level is selected randomly as it favors Neural Network calibration. Due to the random generation of points, significant gaps in the design space are possible. Therefore we suggest a resampling procedure for the LHD component of this hybrid design, which is a modification of a method introduced within Alam et al. (2002).

The basic procedure here is to create *r* layers through assigning *r* equally spaced levels over the design range for each index $x_{i,t-n}$ of simulation input time series and to randomly sample once from each of the *r* layers. The distances $d_{ij,t-n} = \left| x_{i,t-n} - x_{j,t-n} \right| \forall i \neq j, t-n$ between all design point vectors $x$ in the design space with the same time lag *t* are computed along with $\bar{d}_{t-n}$, which is the average distance between these points at time $t-n$. Multiple random samples or multiple candidate designs can be executed and the design that minimizes $S_d = \sqrt{\dfrac{\sum (d_{ij,t-n} - \bar{d}_{t-n})^2}{nm(m-1)/2}}$ is the one that should be selected. Here $nm(m-1)/2$ represents the total number of distances that need to be computed.

The motivation behind the standard deviation calculation is to provide an automated way to select a randomized design that is not overly clustered in particular areas of the design space. A design with a high $S_d$ would be due to having design points clustered together along with large gaps in the design space.

## 5 CASE STUDY

This section presents a case study involving the metamodeling of a simulation with financial time series to illustrate how the proposed approach can be applied and to examine the level of fidelity that can potentially be obtained with this metamodeling approach. The simulation model exercised for this experiment is a MITRE-developed extension of the Heterogeneous Trading Model (Ghoulmie et al. 2005). The Heterogeneous Trading Model is an agent-based simulation that involves modeling of a market, consisting of a single asset and traded by $N$ agents. Trading occurs at discrete dates, $t = 0,1,2....m,$ and at each date, every agent receives public news about the asset's performance. Through what is referred to as an intrinsic subjective criterion, each agent assesses whether the news is positively or negatively significant and subsequently decides whether to place a buy or sell order based on trading rules inherent in the model.

The MITRE-extension (Tivnan et al. 2011) to the model includes an exogenous news signal for equities rather than a random draw of news signals. The main inputs to the model are a time series representing the level of the news signals and a single-valued variable for the number of traders participating in the market. There are two outputs to the model: an equity price time series and a trade volume time series. In order to concentrate on the metamodeling performance in this case study, we are focusing strictly on the equity price time series output and we assume that to be the FoM here. In practice however, multiple simulation outputs could be aggregated via Equation (1) before doing the Neural Network calibration.

The metamodeling experiment consists of trying to develop a surrogate model mapping the time series inputs of the Heterogeneous Trading Model to its equity price output. Moreover, the experiment entails evaluating how the Neural Network metamodel can replicate this large-scale agent-based simulation consisting of time series inputs and outputs. In other words, the experiment is attempting to determine how close is the metamodel output with respect to the simulation output, given that the metamodel has the same input information, which is current and past equity price and previous news levels. As noted in Section 4, we are also employing past observations of the equity price time series to aid the prediction for an Equity price at the following date.

The applied Neural Network metamodel has a single hidden layer feed forward structure as discussed in Section 4 with sigmoid transfer functions. Table 1 below summarizes the inputs used into the Neural Network. The last ten observations of the equity price time series are included along with the last two periods of the news signal level: current news signal level and previous news signal level. Only using the news signal levels from the last two periods is pertinent for the logic of this model and one can of course choose a different number of past observations of the input time series to apply in different situations. Held constant during this study in the model are the number of traders and the threshold probabilities for trade decisions. The LHD experimental design component is implemented only on single-valued variables so it was not utilized for this case study.

Table 1: Inputs to Neural Network

| | **Time Periods** | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| **News Level** | $t$ | $t$-1 | | | | | | | | |
| **Equity Price** | $t$-1 | $t$-2 | $t$-3 | $t$-4 | $t$-5 | $t$-6 | $t$-7 | $t$-8 | $t$-9 | $t$-10 |

Each of the outputs from the model replications is provided in Figure 3 below. Thirty replications are performed in total of this simulation.

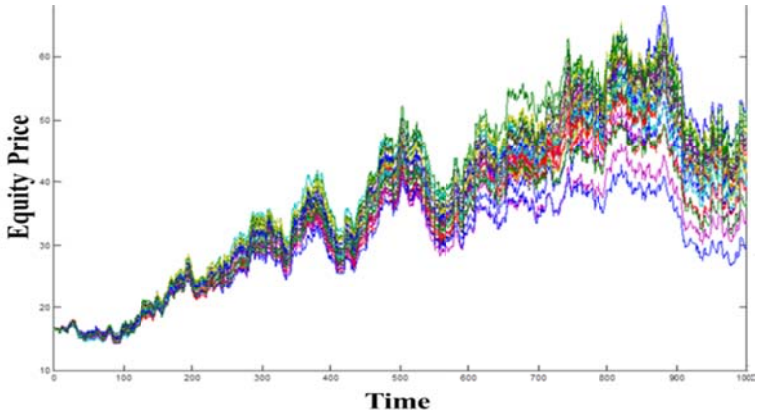


Figure 3: Outputs of Heterogeneous Trading Model

There are twenty-one replications (70%) that are used for training, while five are used for validation and four are used for testing. The training during this case study was achieved with backpropgation using an Ordinary Least Squares (OLS) estimator as given in Equation 4:

$$\varepsilon_{LS} = \frac{1}{2N}\sum_{p=1}^{N}\left(s_p - m_p\right)^2 \tag{4}$$

where $N$ is the total number of observations in the sample, $s_p$ is the simulation response at period $p$ and $m_p$ is the metamodel response at period $p$. Figure 4 below shows the goodness of fit results for the simulation output sample used to train the Neural Network via a regression plot between simulation and metamodel responses. It demonstrates a strong goodness of fit between the metamodeling outputs and outputs from the simulation model within the sample used to train the Neural Network. An r value of 0.99857 was achieved indicating a significant correlation between the simulation responses and metamodel responses under similar time periods.

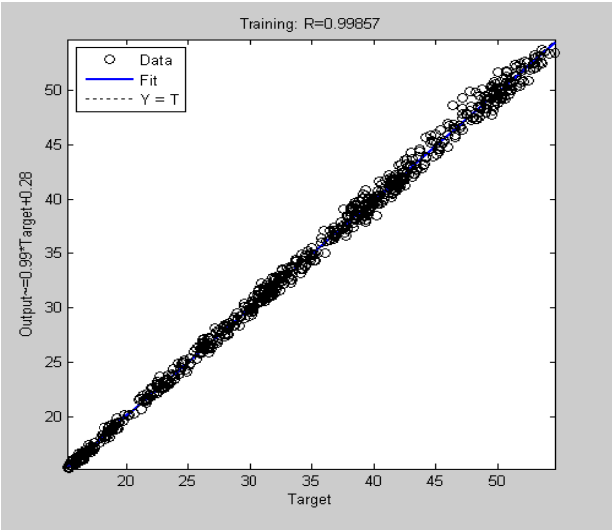


Figure 4: Goodness of Fit of Metamodeling with Training Data

The validation partition of the data is used for model selection and is used during the training phase of the Neural Network to ensure that the Network is not overfit. During validation, the Neural Network weights are not being configured any more with the data, but the data is now utilized to check that any increase in accuracy over the training data set actually yields an increase in accuracy over a data set not previously shown to the Network. If the accuracy over the validation data stays the same or decreases, then it can be concluded that overfitting is beginning to occur and training should stop. In our case, the r value decreases ever so slightly from 0.99857 to 0.99807 implying that it is an appropriate time for training to stop. The regression plot between the simulation data and metamodel data under the validation partition is shown below in Figure 5.

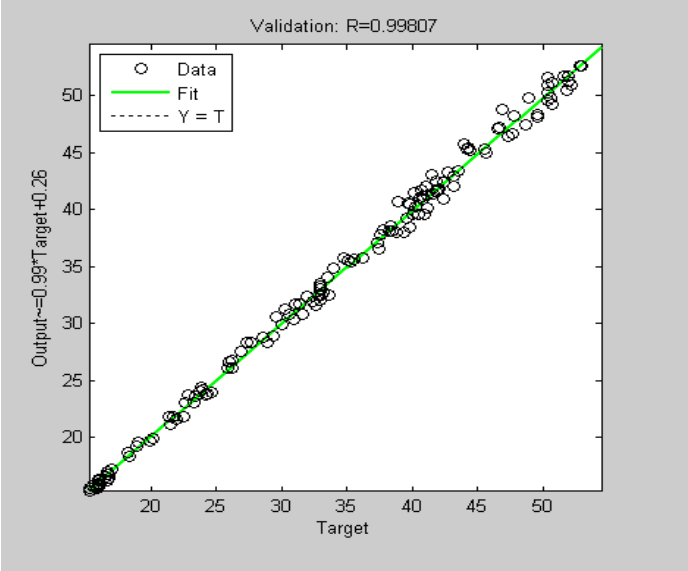


Figure 5: Goodness of Fit of Metamodeling with Validation Data

The testing partition of the data is used to evaluate how well the Neural Network can extrapolate outside of the data space it was trained. The reader must be careful in this case as to not misinterpret this as the Neural Network capability to predict futures values. This testing is used to see how well the Neural Network can represent the simulation under the simulation inputs that were not used to fit the model. However, this is the most appropriate measure in the adequacy of the metamodel here. The r value between the metamodel and simulation model for the testing data is 0.99643, which is only a slight decrease that what was achieved in the training data. This is a very encouraging result showing that the metamodel is quite capable of replicating the simulation model outside of the training space. The regression plot between the simulation data and metamodel data under the testing partition is shown below in Figure 6.

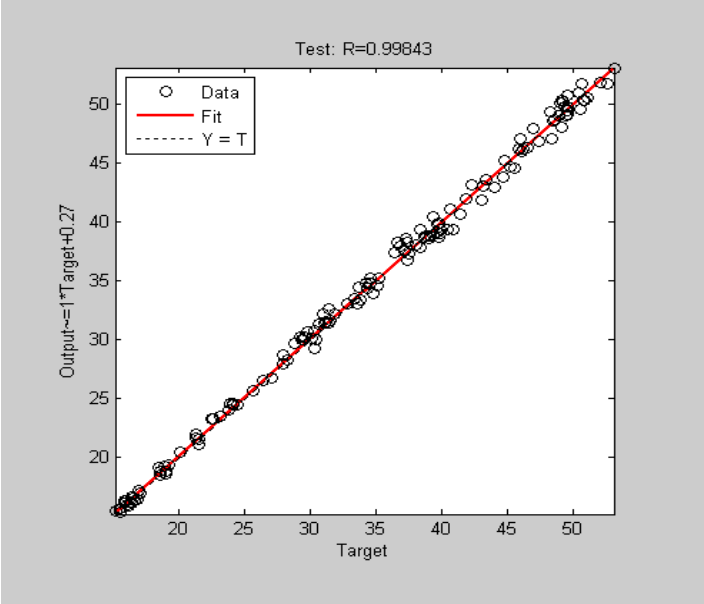


Figure 6: Goodness of Fit of Metamodel with Testing Data

A mean squared error of 0.4119 (error between metamodel output an simulation output of Equity Price) was achieved for the training data signifying a sound goodness of fit for the Neural Network metamodel. To further depict the fit of the Neural Network metamodel, the metamodel time series is plotted (blue) overtop the simulation output time series (red) in Figure 7 below. The metamodel appears to be able to capture most of the trends of the simulation time series.

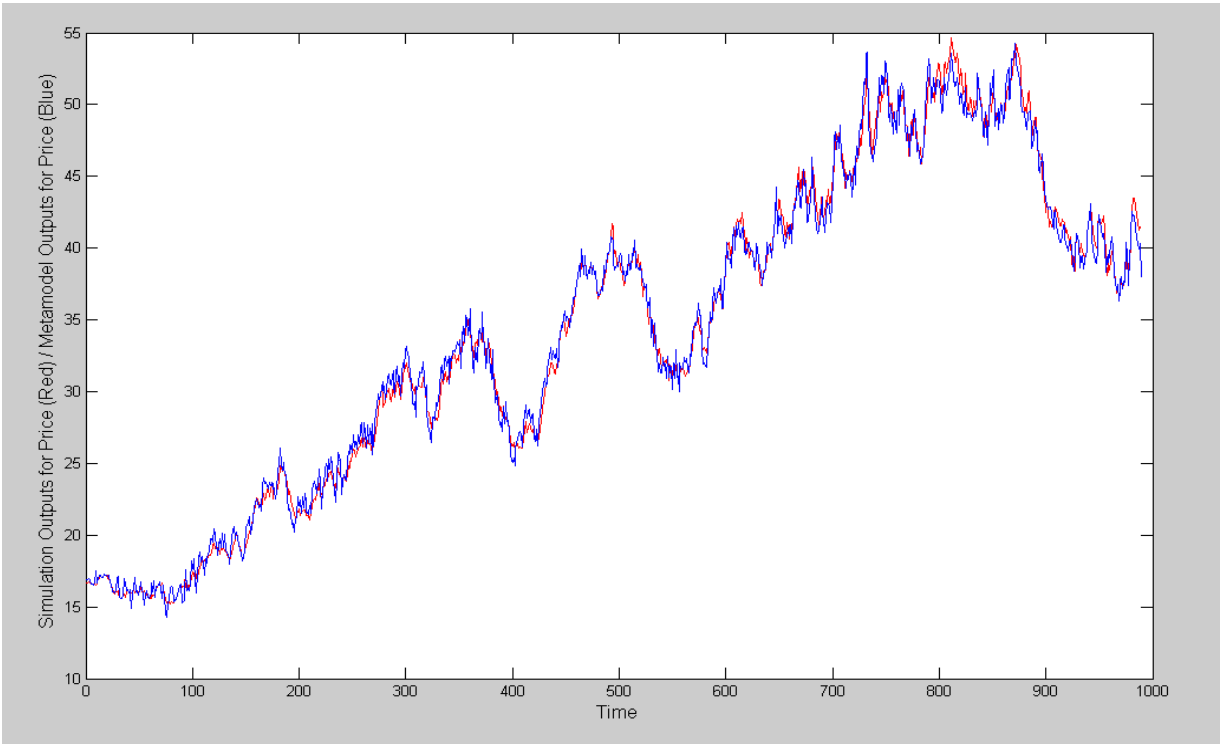


Figure 7: Comparison of Metamodel (Red) and Simulation (Blue) Time Series

This case study has demonstrated the effectiveness of Neural Networks in metamodeling large-scale simulations that contain time series inputs and outputs. A single hidden layer Neural Network was applied, but there are of course many other Neural Network structures that can be applied in this case. Other structures can lead to additional complications with non-linearity, especially when multiple hidden layers are present. Under multiple hidden layers, the additivity assumption of a Latin Hypercube Design will not necessarily fit so other experimental designs need to be substituted.

## 6 CONCLUSIONS

This paper addresses the problem of metamodeling of time series data that naturally occur in financial simulations. A general approach is presented, which is also suitable to other system domains yielding time series outputs. We provide a procedure generating a single-valued Figure of Merit for time series outputs that employs metamodeling and an MAV model that can be recalibrated under different users of the simulation(s). This mapping enables real-time analysis and analysis capabilities that can be easily distributed amongst multiple stakeholders at play.

The success of this study lays down the framework for further research. It will be important to further investigate and identify the time series simulation characteristics where this approach is applicable. Moreover, the next step in this research is to see if single hidden layer can be successful across a wider range of simulations and under simulations consisting of both singled-valued and time series input parameters. In addition, further studies into applying heuristic search procedures, such as Evolutionary Algorithms and Genetic Algorithms to locate effective Neural Network structures will be valuable.

## ACKNOWLEDGMENTS

The authors thank Mr. Mathew McMahon and Ms. Jenny McFarland for providing the time series data (pertaining to Fig. 3) used in the case study. Additionally, the authors thankfully acknowledge many valuable discussions with Dr. Charles Worrell, Ms. Marie Francesca, Dr. Kevin Cabana and Dr. David Colella. The authors have thankfully noted and addressed several insightful comments and suggestions made by the reviewers.

## REFERENCES

Alam, F., K. McNaught and T. Ringrose. 2002. "A Comparison of Experimental Designs in the Development of a Neural Network Simulation Metamodel." *Simulation Modeling Practice and Theory*. 12:559-578.

Arthur, W., J. Holland, B LeBaron, R. Palmer, and P. Tayler. 1996. "Asset Pricing under Endogenous Expectations in an Artificial Stock Market." Santa Fe Institute, Santa Fe, NM 96-12-093.

Barton, R. and M. Meckesheimer. 2006. "Metamodel-Based Simulation Optimization." *Handbook in OR & MS, Vol*. 13. Edited by S.G. Henderson and B.L. Nelson.

Box, G. and N. Draper. 1987. *Empirical Model-Building and Response Surfaces*. New York: John Wiley and Sons.

Deboor, C. 1978. *A Practical Guide to Splines*. New York: Springer-Verlag.

Dyer, J. and Sarin, R. 1979. "Measurable Multiattribute Value Functions." *Operations Research*. 27(4): 810-822.

Dyn, N., Levin, D., and S. Rippa. 1986. Numerical procedures for surface fitting of scattered data by radial functions. *SIAM Journal of Scientific and Statistical Computing*, 7: 639-659.

Faraway, J. and C. Chatfield. 1996. "Time Series Forecasting with Neural Networks: A Comparative Study Using the Airline Data." *Journal of the Royal Statistics Society: Series C (Applied Statistics)*. 47(2): 231-250.

Gheyas, I and L. Smith. 2009. "A Neural Network Approach to Time Series Forecasting." *Proceedings of the World Congress on Engineering Vol II*.

Ghoulmie, F., R. Cont and J. Nadal. 2005. "Heterogeneity and Feedback in an Agent-based Market Model." *Journal of Physics: Condensed Matter*, 17(14): S1259-S1268.

Hill, T., M. O'Connor and W. Remus. 1996. "Neural Network Models for Time Series Forecasts." *Management Science*, 42(7): 1082-1091.

Kleijnen, J. 1975. *Statistical Techniques in Simulation, part II*. Dekker, New York.

Kleijnen, J. 1987. *Statistical Tools for Simulation Practitioners.* New York: Marcel Dekker.

Kleijnen, J., S. Sanchez, T. Lucas, and T. Cioppa. 2005. A User's Guide to the Brave New World of Designing Simulation Experiments. *INFORMS Journal on Computing*, 17(3): 263-289.

Kleijnen, J. and R. Sargent. 2000. A Methodology for Fitting and Validating Metamodels in Simulation. *European Journal of Operational Research*, 120: 14-29.

Kleijnen, J. 2009. Kriging Metamodeling in Simulation: A Review. *European Journal of Operational Research*, 192, 707-716.

Man-Chung, C., C. Wong and L. Chi-Chung. 2000. "Financial Time Series Forecasting by Neural Network Using Conjugate Gradient Learning Algorithm and Multiple Linear Regression Weight Initialization." *Computing in Economics and Finance*, 61.

Meghabghab, M. 2001. Iterative radial basis functions neural networks as metamodels of stochastic simulations of the quality of search engines in the World Wide Web. *Information Processing and Management,* 37*:* 571-591.

Myers, R., 1976. *Response Surface Methodology*. Boston: Allyn and Bacon.

Myers, R. V. Ugru, X. Luo, and R. Grandhi. 1996. "MIDAS for pre- and post-processing of ASTROS unsteady aerodynamic flutter models." *6th AIAA/NASA/ISSMO Symposium of Multi-disciplinary Analysis and Optimization*.

Rosen, S. C. Harmonosky, M. Traband. 2006. "A Simulation Optimization Method That Considers Uncertainty and Multiple Performance Measures." *European Journal of Operational Research* 181: 315-330.

Rosen, S. C. Saunders, and S. Guharay. 2012. "A Structured Approach for Rapidly Mapping Multi-Level System Measures via Simulation Metamodeling." To be *Submitted to IEEE Transactions on Systems, Man, and Cybernetics, Part A: Systems and Humans.*

Sacks, J., Welch, W., Mitchell, T., and H. Wynn. 1989. Design and analysis of computer experiments. *Statistical Science*, 4: 409-435.

Tivnan, B., M. Koehler, M. McMahon, M. Olson, N. Rothleder, and R. Shenoy. 2011. "Adding to the Regulator's Toolbox: Integration and Extension of Two Leading Market Models." *2011 Annual Conference of the Eastern Economic Association*.

Torgerson, W. 1958. Theory and Methods of Scaling, Wiley, New York.

Van Dam, E., Gijs, R., and Husslage, B. 2009. Bounds for Maximin Latin Hypercube Designs. Operations Research, 57: 595-608.

Von Neumann, J. Morgenstern, O. 1947. *Theory of Games and Economic Behavior*. Princeton: Princeton University Press.

Zhang, G. E. Patuwo, and M. Hu. 1998. Forecasting with Artificial Neural Networks: the state of the art. *International Journal of Forecasting*. 14: 35-62.

## AUTHOR BIOGRAPHIES

**SCOTT L. ROSEN** is a Group Leader for the Probability and Statistics Group within the MITRE Corporation's Operations Research Department. He has spent the last seven years at MTIRE applying Operations Research to some of the United States' most critical and challenging Systems Engineering problems. His research interests include Simulation Optimization, Simulation Metamodeling, Decision Analysis, and Quantitative Systems Engineering. He received his B.S. from Lehigh University in Industrial and Systems Engineering in 1998 and a M.S. and Ph.D. in Industrial Engineering and Operations Research from Penn State University in 2000 and 2003, respectively.

**CHRISTOPHER P. SAUNDERS** is a mathematical statistician with the MITRE Corporation; he also holds an assistant professorship of statistics at South Dakota State University. He received his Ph.D. in statistics from the University of Kentucky before accepting an Intelligence Community postdoctoral fellowship. His current research focus is related to high dimensional pattern recognition and approximate Bayesian inference with applications to time series analysis and forensic identification.

**SAMAR K. GUHARAY** has been leading research in the area of interdisciplinary problems covering novel sensing technology, algorithms, and rapid systems engineering analysis. He received his Ph.D. in Physics in 1980 from the University of Calcutta, India. Dr. Guharay participated in many national and in-

ternational research programs over more than thirty years of his professional career. He has been affiliated with universities in different capacities. In addition to leading research programs at MITRE he has a long background as PI/Co-PI of projects sponsored by multiple agencies including NSF, NIH, DoE, DoD, industries and abroad.