Proceedings of the 2006 Winter Simulation Conference
L. F. Perrone, F. P. Wieland, J. Liu, B. G. Lawson, D. M. Nicol, and R. M. Fujimoto, eds.

GRID ENABLED SEQUENTIAL DESIGN AND ADAPTIVE METAMODELING

Wouter Hendrickx
Dirk Gorissen
Tom Dhaene

Department of Mathematics and Computer Science
University of Antwerp
Antwerp, BELGIUM

ABSTRACT

Metamodeling is emerging as a valuable new tool in simulation: complex computer codes can be approximated by surrogate models (analytic, neural network, SVM, etc.) which can easily be evaluated on-the-fly. Adaptive modeling and sequential design further improve the performance of metamodeling frameworks. Grid computing quickly replaces regular cluster computing when it comes to complex calculations. Several efforts use grid computing to facilitate the exploration of simulator outputs. This contribution combines adaptive modeling and sequential design with distributed, grid-based techniques into one metamodeling framework.

1 INTRODUCTION

This work combines several methodologies for metamodel construction. Each of these deserves a separate section in its own right. Therefore references on these subtopics are deferred to the relevant sections. The introduction will merely give a bird's eye view of the new framework presented here.

The main objective of this work is metamodeling: constructing analytical models for complex computer simulation codes (as will be explained in section 2). In order to obtain accurate and compact metamodels, several techniques are at hand. Adaptive modeling is presented in section 5 and sequential design is discussed in section 6. A combination of these two already gives rise to a robust metamodeling framework.

Distributed computing, and more specifically grid computing, has been used in the past for running simulations in parallel (see section 4). However, often the power of the grid is applied in a brute force manner: As many simulations are run as possible and once the results are in, the data is stored in a database for future analysis or a metamodel is built. Such a scheme leaves much room for improvement if the sequence of simulations is chosen more carefully and if there is a tighter integration with the modeling process.

Combining adaptive modeling, sequential design and grid computing into one global metamodeling framework gives rise to a methodology with great potential. The M3 Toolbox, currently under development in our research group, is used to explore the possibilities of such a framework. The example of section 7 demonstrates what the M3 Toolbox can do for the metamodeling community.

2 METAMODELING

Metamodeling, or *surrogate modeling*, is the process of capturing the features of a complex computer simulator in a simpler, analytical model. This section introduces some metamodeling basics and techniques.

2.1 Concepts

To gain understanding of real-life phenomena, scientists have always constructed approximations and simplifications of reality. The past decades computers provided a valuable tool in constructing such approximations. In many different fields computers are solving partial differential equations in order to predict the behaviour of real-life systems (e.g., fluid dynamics or electromagnetic (EM) field simulators). Others use complex computer models to predict a system's behavior (e.g., traffic simulators or manufacturing process optimisation).

In this text the term *simulator* will be used for such a complex computer model, the term *system* will be reserved for the real-life phenomenon under consideration. For their computations simulators take one or more parameters as *inputs*, e.g., frequencies and length's for an EM-simulator. Based on these input parameters the simulator computes the system's behavior. The term behavior is very general, it can be electrical and/or magnetic fields, current lines for fluid flow, etc. In many cases the end-user is not interested in the full behavior of the system, but rather in some scalar values that can be derived from the simulator's calculations. In electromagnetics, the (scalar) scattering parameters of a

electronic component can be computed based on the EM-fields. In fluid dynamics, pressures and flow rates can be computed. Typically, these scalar values capture the essence of the system's behavior. These scalar values will be called the *outputs* of the simulator.

From hereon, the simulator will be regarded as a black-box with multiple inputs and outputs. The number of inputs will be called d, while the number of outputs is denoted by s. So the simulator can be seen as a function $S : \mathbb{R}^d \rightarrow \mathbb{R}^s$.

In many cases, a simulator is computationally expensive. Therefore, in order to gain a global overview of the output of the simulator as a function of its inputs, one tries to build *metamodels* or *surrogate models*. These are mathematical expressions or models-of-models which map the inputs onto the outputs. Converse to the original simulator, metamodels can be evaluated at almost no cost. Metamodels can be used for optimisation, sensitivity analysis or to "browse" through the input-output behavior of the simulator.

2.2 Overview

A lot of innovative research has been conducted in the field of metamodeling. This section tries to give a very brief survey of such efforts. Some of these focus on optimisation, while others want to gain insight in the global input-output relationship.

The idea of building metamodels for computer simulations is not new. Sacks et al. (1989) propose to use Kriging-like metamodels for predicting simulator outputs. They call their approach *Design and Analysis of Computer Experiments*, or DACE (see, e.g., Santner et al. 2003). Søndergaard (2003) uses the *Space Mapping* technique to align a simple circuit model with complex computer simulator code, in order to find an optimum. Lamecki et al. (2003) use Radial Basis Function (RBF) interpolants to approximate the behavior of electromagnetic components. De Geest et al. (1999) constructs polynomial and rational metamodels of electronic circuitry. Barton (1998) gives an overview of different kinds of modeling techniques used for metamodeling of simulation outputs.

3 MODEL FLAVORS

This section gives an overview of the three different kinds of models implemented in the M3 Toolbox. Examples are given in the two-dimensional case, but can be extended to more dimensions in a straightforward manner. Other modelling techniques, such as DACE and Support Vector Machines will be added to the Toolbox' codebase shortly.

3.1 Artificial Neural Networks

Since their initial conception as crude models of biological neurons Artificial Neural Networks (ANN) have proven very useful, flexible and powerful tools for solving a wide range of complex problems: classification, pattern recognition, system control, time series prediction, function approximation, regression, optimisation, reasoning, etc.

In the context of metamodeling they constitute a particularly attractive method considering their universal, black-box nature. ANN have been proven, under reasonable conditions, to be able to approximate any computable function with arbitrary accuracy through learning (White 1990, Valiant 1988) without requiring any a priori knowledge of the underlying system. However, this universality often proves to be a double edged sword, as will be described below.

Nevertheless ANN have successfully been applied to many metamodeling problems (Xiao et al. 2003, Hillbrand and Karagiannis 2002, Dahm and Ziegler 2002, Panayiotou et al. 2000) ranging from economic validation of capital projects (Chaveesuk and Smith 2003), through process optimisation (Chambers and Mount-Campbell 2002), to microwave circuit modeling (Devabhaktuni et al. 2001, Zhang et al. 2003).

An ANN consists of a number of, usually adaptive, artificial neurons that are interconnected to form a network. This is illustrated in figure 1.

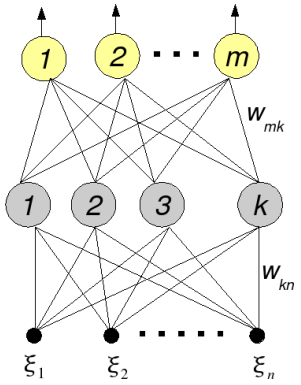

Figure 1: Structure of a Typical Multi Layer Perceptron with One Hidden Layer with k Units

The most popular network type is the feed-forward Multi-Layer Perceptron (MLP) but many others exist: CSOM, Cellular Networks, ART, Polynomial Neural Networks, RBF Networks, Hopfield, Neural-GAS, Fuzzy Nets, etc. Each neuron in the network is intended to respond to stimuli in a manner not unlike their biological counterparts. A neuron typically takes a weighted sum of its inputs ξ_i and bias θ and computes an output value $y = g(h)$ where h is defined as (see figure 2):

$$h = \left(\sum_{i=1}^{n} \xi_i w_i \right) + \theta \tag{1}$$

The weights and biases w_i, θ constitute the free parameters of the network. These are initialised randomly and gradually adapted through training the network with a number of examples.

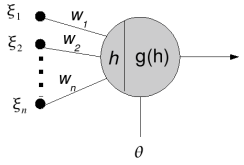

Figure 2: A Typical Neuron

The training algorithms used are usually gradient based algorithms such as Levenberg-Marquardt or Resilient Backpropagation. The objective is to adapt the weights in such a way that they minimise the mean squared error between the prediction of the network and the true values. The function $g(.)$ is referred to as the *activation* function and is typically a logistic function like the sigmoid:

$$g(h) = \frac{1}{1 + e^{-h}} \tag{2}$$

Together equations (1) and (2) from the *transfer* function of the neuron. The high nonlinear nature of such a transfer function means that a neural network is able to model exceedingly complex systems. However, care must be taken to ensure the network is not too complex for else the nonlinearities become uncontrollable.

This brings us to an important point. Though attractive theoretically due to their universal approximation properties, the design and application of ANN in practical settings is not all that straightforward. The problem is twofold. First there is the problem of choosing an appropriate network model (RBF, MLP, GRNN, etc.) and then the classic problem of choosing the best network parameters. Due to their versatility ANN have a great number of parameters that may be tuned: number of hidden layers, number of hidden units, training function, learning rate, update method (batch/incremental), number of epochs, stopping criteria, etc. Concerning the network topology there are many anecdotal and published "rules of thumb". However for all but the most specific cases such guidelines are very vague and fail to capture the complex interaction between the numbers of input and output units, the number of training cases, the amount of noise in the targets, the complexity of the function or classification to be learned and many other factors (Neural Networks).

To tackle this researchers have come up with three types of solutions: constructive algorithms that grow or prune the network as training proceeds (e.g., Cascade Correlation, Tiling, Optimal Brain Damage), population based methods (e.g., committees (ensembles), genetic algorithms, genetic programming) and combinations of both. By smartly exploring the search space of networks such methods have proven to perform much better than their one-shot counterparts.

3.2 Radial Basis Functions

Radial Basis Function (RBF) interpolants are a very simple yet powerful tool in multivariable approximation. The book (Wendland 2005) can be used as a reference on the subject. RBF interpolants have been used in a wide range of applications, going form solving partial differential equations (Kansa 1990) to surface smoothing in computer graphics (Carr et al. 2001). Their simple formulation makes RBF interpolants easy to use and understand, although sometimes a more complex meta-model would produce more satisfactory results.

The main ingredient of an RBF interpolant, is a function $\phi : [0, \infty[\rightarrow \mathbb{R}$, for example the bell-shaped Gaussian function $x \mapsto \mathrm{e}^{-\alpha x^2}$ for some α. Given a set of points $(\zeta_k, \xi_k) \in \mathbb{R}^2$, called *centers*, one searches for a function

$$\begin{aligned} H(x,y) &= \sum_{k=1}^{K} \beta_k \phi\left(\|(x,y) - (\zeta_k, \xi_k)\|_2\right) \\ &= \sum_{k=1}^{K} \beta_k \mathrm{e}^{-\alpha\left((x-\zeta_k)^2 + (y-\xi_k)^2\right)} \end{aligned} \tag{3}$$

which is linear in the unknowns β_k. This can be seen as an overlay of shifted ϕ's scaled by the different β_k's. An approximant of the form (3) to a set of data points and values (x_l, y_l, f_l) can be found by solving a least-squares linear system.

3.3 Rational Approximants

Linear regression models have always been a key tool for modeling a wide variety of systems. Specifically, using polynomials for interpolation and approximation of scattered data has been popular for many years. In the context of metamodeling, De Geest et al. (1999) and Lamecki et al. (2004) have modeled simulation outputs with polynomials and ratios of polynomials:

$$H(x,y) = \frac{\alpha_1 + \alpha_2 x + \alpha_3 y + \alpha_4 xy + \ldots}{1 + \beta_2 x + \beta_3 y + \beta_4 xy + \ldots} \tag{4}$$

Ratios of polynomials are known as *rational functions*.

A main drawback is that they are not linear in their unknowns. To overcome this inconvenience, one usually solves the linearized system of equations

$$\begin{aligned} &(\alpha_1 + \alpha_2 x_k + \alpha_3 y_k + \alpha_4 x_k y_k + \ldots) \\ &- f(x_k, y_k)\,(1 + \beta_2 x_k + \beta_3 y_k + \beta_4 x_k y_k + \ldots) = 0 \end{aligned} \tag{5}$$

When the number of design points matches the number of unknowns exactly, an interpolant can be found most of the times. Otherwise, the least-squares solution to this system differs from the true nonlinear least-squares approximant, which would minimise $\sum_k |f(x_k, y_k) - H(x_k, y_k)|^2$. In many cases, it is acceptable to neglect the error caused by the linearisation and just use the solution of the linearized equations.

To create a diversity of different rational metamodels through the same set of sample points, the degrees of x and y present in numerator and denominator can be varied. Our current implementation assigns weights to each variable, and uses these weights to select which degrees to use. Also, the number of degrees of freedom can be varied between different rational models.

4 GRID COMPUTING

This section briefly introduces some ideas and concepts behind distributed computing using grids. It discusses a number of middlewares that enable grid computing and their application to metamodeling problems.

Grid computing is a decade old research field within distributed computing which can conceptually be described as follows: the interconnecting of heterogeneous, non-dedicated, geographically distributed, computing resources (clusters, desktop machines, disk arrays, etc.) into a large, aggregate pool of resources that may be shared among users. An often used and intuitively appealing analogy is that the computing power of a grid should be as ubiquitous and easily accessible to researchers as electricity from the power grid.

As such, grid computing differs markedly from cluster computing which only deals with limited, dedicated, homogenous resources over which the user typically has complete control.

4.1 Grid Systems

Multiple software layers have been developed to make grid computing possible. These are referred to as *middlewares.* Popular middlewares include Globus (Foster et al. 2001), Unicore (Erwin 2002), Legion Natrajan et al. 2002, JGrid Pota et al. 2003, VgrADS (Berman et al. 2005), ProActive (Huet et al. 2004), Gridbus (Buyya and Venugopal 2004), and Triana (Taylor et al. 2005), of which Globus is the most well known. Together these have been used to help solve a wide variety of computationally expensive or data intensive problems such as airflow simulation, virtual reality environments, image rendering and weather prediction.

4.2 Grid-based Metamodeling

Grid computing and metamodeling form a potentially perfect match. A problem typically encountered in sequential metamodeling is that many evaluations of the original simulator are required to build a metamodel. If the simulator can be evaluated cheaply this is no problem. Typically one simulation run may require many minutes or hours of computing time. Requiring many simulation points to build an accurate metamodel thus quickly becomes intractable, especially if done sequentially. This is where grid computing can provide a solution. By delegating the evaluation of the expensive simulator to the grid, computing time may be greatly reduced.

Research efforts and tools that integrate metamodeling and grid computing techniques can be divided into two categories: those catered towards design optimisation and those geared towards the building of standalone scalable metamodels. The first is by far the most populous with projects as GEODISE (Eres et al. 2005), DAKOTA Giunta and Eldred 2000, Nimrod/O (Abramson et al. 2001) and the work in Ong et al. 2003, Ong et al. 2004. The latter take an evolutionary approach to metamodel based design optimisation while simultaneously harnessing the power of the grid (Ng et al. 2005, Ng et al. 2005).

While all projects mentioned above are tailored towards optimisation, they are not concerned with creating a global, scalable metamodel. Research efforts that do build replacement metamodels exist (Lehmensiek and Meyer 2001, De Geest et al. 1999, Martin and Simpson 2002, Hendrickx and Dhaene 2005a), but fail to tackle the computational burden by distributing and parellizing sample evaluation. To the authors knowledge there are no other real projects that solve this problem. Perhaps the project that comes closest to what is achieved in this work is described in (Parmee et al. 2005), though it is also biased towards optimisation.

5 ADAPTIVE MODELING

While building stand-alone metamodels can be useful, usually one does not know in advance which model to build for a given problem. The correct model complexity is unknown and a lot of questions have to be answered, like which degrees should be selected for rational models, which basis function and shape parameter α should be used when creating RBF metamodels, or how many hidden nodes and which transfer functions give rise to a useful ANN metamodel?

5.1 Model Parameter Selection

An *Adaptive Modeling* tool solves this by shifting the responsibility for selecting suitable metamodel parameters from the end-user to the modeling software. Adaptive modeling is the process of iteratively generating and selecting model

parameters in a manner akin to natural selection. Each iteration the usefulness of a population of model parameters is assessed by verifying the accuracy of the corresponding models. New parameters are generated to replace the least fit individuals (those producing the worst models). Figure 3 shows a flow chart illustrating the adaptive modeling process.

Depending on the number and type of model parameters, several techniques for adaptive modeling are at hand. In the case of rational metamodels, a simple scheme for selecting new model parameters already produces promising results: select new variable weightings in the neighb!ourhood of those that generated the best models. For RBF models, the variable parameter is the shape parameter (α in the case of the Gaussian basis function). For ANN models adaptive modeling gets more interesting. Due to the large number of free parameters (network topology, training algorithm, etc.) the search space of possible ANN models is much larger than that of the polynomial or RBF models. Thus automatic tuning of parameters through a parallel evolutionary search makes sense.

Implementing adaptive modeling can be done in roughly three ways (from simple to complex): Evolution Strategies (ES), Genetic Algorithms (GA) and Genetic Programming (GP). ES are popular in numerical optimisation and represent an individual metamodel as a vector of real-valued features (variables). They typically utilise uniform random selection, a discrete or intermediary recombination operator and a Gaussian, self-adaptive mutation operator. GAs are more complex in that they use more involved selection (non-uniform), recombination and mutation operators and generally use larger populations. GP is similar to GAs except that it uses an indirect encoding instead of a direct one. In GP a population of programs that generate solutions is evolved instead of the solutions themselves. However, these are just simple guidelines, in practise many variations exist and the distinctions are less clear cut.

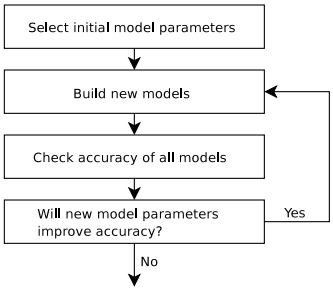

Figure 3: Adaptive Modeling Flow-Chart

5.2 Example

To demonstrate the usefulness of adaptive modeling, experiments were conducted on the test function

$$K : [0,4]^5 \to \mathbb{R} : \\ (x,y,a,b,c) \mapsto \frac{\mathrm{e}^{-(x-1)^2}}{1.2+(y-2.5)^2} + \text{ noise} \quad (6)$$

The output value of this function only depends on the first two inputs. Some noise is added to the outputs in order to obscure the irrelevance of the last three variables (a uniformly distributed random error in $[-0.001, 0.001]$ was used for this purpose). As a basic test case, 100 samples where randomly taken in the domain, and rational metamodels where constructed approximating the data. Each iteration, the best models were kept and the worst were replaced by new ones (resembling the best models). Figure 4 shows the evolution of the weights the rational modeler assigns to each variable. A lower weight implies increased importance. The code clearly identifies the important factors, while marginalizing the others. These tests were run on the adaptive modelling code as it appears in the M3 Toolbox' implementation. Therefore it is reasonable to assume the adaptive modelling technology is working and provides a valuable contribution to the entire meta-modelling process.

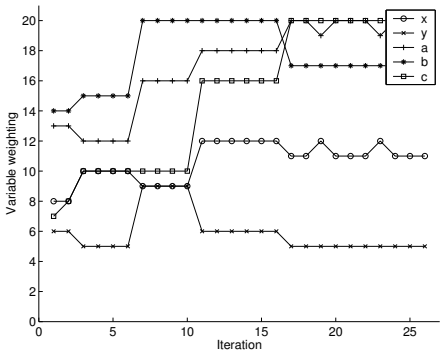

Figure 4: Evolution of the Weights for Each Factor, Lower Weight Means More Importance

A similar result is observed when adaptively modeling using ANN metamodels. ANN have the attractive property that they are able to automatically detect complex relationships in the data. In this case an MLP with 5 inputs, one output and 2 hidden layers with 3 nodes was trained with a similar set of 100 points. That it detected the irrelevance of the last three input variables can be seen from the connection weights connecting the inputs to the first hidden layer: the weights leaving the inputs a, b and c tend to zero, while the ones for x and y remain significant (see table 1). Note

that this is a property of the ANN itself and not attributed to the adaptive modeling process. An example of adaptive modeling using ANN is illustrated in figure 5. The modeled system is is the EM simulator described in section 7 and depicted in figure 7.

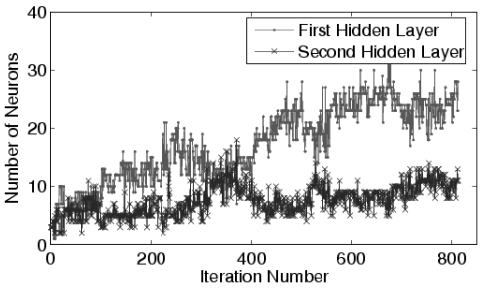

Figure 5: Evolution of the Number of Hidden Nodes

As can be seen from the figure the adaptive modeling process will automatically select the optimal number of hidden units given the data. The evolution mechanism used here was a simple $(\mu + \lambda)$-Evolution Strategy (μ=20, λ=10) with a Gaussian mutation operator and no recombination operator. Evolution was allowed to proceed for 10 generations before moving on to the next sequential design iteration. Training was done using 4-fold crossvalidation and regularisation for an evolved average of 800 epochs. To further reduce the risk of overfitting a constraint was placed on the network complexity. The number of free parameters should never be more than k times the number of available samples. It seems obvious to choose $k < 1$, however, Sarle (1995), Caruana et al. (2001), and others have shown that allowing more weights than datapoints is not always a bad thing and can actually provide better results, especially when using early stopping. Therefore we have set $k = 2$ in this case.

While this already gives good results, 10 generations is really to small to truly benefit from natural selection. Also more powerfull mutation and recombination operators are needed that not only act on the topology but also on training-related parameters. Valuable work in this respect has already been done (Yao 1999).

Altough the current imlementation performs fine in our two testcases, there is still room for improvement. More general Genetic Algorithm (GA) inspired techniques can be used to select optimal model parameters. Other optimization techniques might also be used to select the optimal model.

6 SEQUENTIAL DESIGN

Usually, in real-life experiments, a design in the input space is constructed and all experiments are conducted. Only after all data has been collected are models constructed to approximate the input-output relation. Quite often it is impossible or too expensive to conduct more real-life experiments.

Table 1: Connection Weights Between the Input Layer and the First Hidden Layer, Redundant Inputs Receive Lower Weights

		Input Neuron				
		1	2	3	4	5
Hidden Neuron	1	0.126	0.957	-0.003	-0.005	-0.012
	2	-0.652	0.522	0.003	0.005	-0.009
	3	0.189	-1.134	-0.001	-0.002	0.013

6.1 Adaptive Sample Selection

When performing computer simulations, it is still feasible to conduct more experiments *after* initial metamodels have been built. *Sequential design* or *adaptive sampling* is the process of iterative metamodel construction, while running new simulations each iteration.

One-shot design based metamodeling has some major drawbacks. In some regions the design will be too dense, the complexity of the input-output relation could be found with far less sample points. In other regions, the simulator output is far more complex than expected and more sample points are needed. Likewise, for optimisation it might be beneficial to select more sample points in the regions where extrema are located.

These drawbacks are overcome by sequential design: initially, a small amount of sample points is selected in the input space and simulations are run for these points. Using this data, metamodels are built in an adaptive loop, as described in the previous section. The models that are the most accurate are then used to select new inputs. For these new data points simulation outputs are computed and added to the set of samples. These steps are iterated until the estimated accuracy reaches a threshold, or when no further progress seems possible. Figure 6 shows a flow-chart of the sequential design process combined with an adaptive modeling loop.

Sequential design has been used for metamodeling purposes on several occasions (Kleijnen and Van Beers 2003, De Geest et al. 1999, Hendrickx and Dhaene 2005b).

6.2 Grid computing for adaptive sampling

Distributed computing can further improve the performance (run time) of the sequential design and adaptive modeling scheme. There are several levels of integration when com-

bining grid computing with a sequential design automaton as shown in figure 6.

In a first phase, each iteration of the sequential design loop (the outer loop) can be sped up by evaluating the new samples in parallel on the grid. Assuming the network transfer times are negligible (which is the case for expensive simulations), a speed-up of *number of samples to be evaluated* over *number of machines* can be achieved (assuming more samples than hosts). The main advantage of this approach is that it yields a major performance improvement, without interfering with the standard sequential design work flow.

The adaptive modeling loop can repeatedly build new models and suggest new inputs to simulate by maintaining a priority queue. A grid computing interface then selects inputs from the queue proportional to their priority, runs simulations and feeds the results back into the adaptive modeling loop. In this process, it is important to keep the queue balanced: one has to ensure that the queue remains filled without flooding it with useless sets of design points.

Currently, our code features a back-end for distributed computing using the AppleS framework, using the ProActive framework and using the Globus middleware. Through AppleS tests can be run by just grouping several desktop computers into a simple computer cluster. The ProActive framework provides us with access to the CalcUA computer cluster of our department. The Globus middleware enables us to run distributed tests on larger scale grid infrastructures.

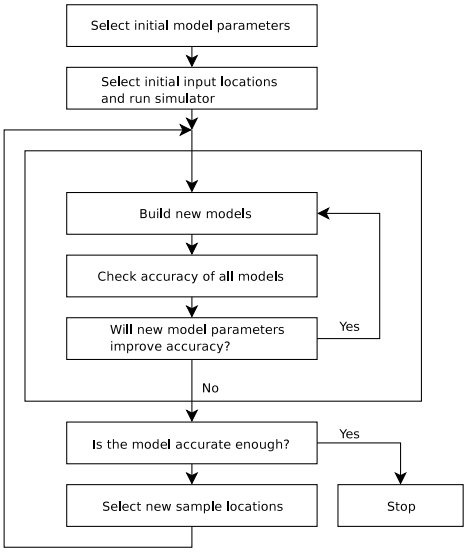

Figure 6: Sequential Design with Adaptive Modeling

7 COMPLETE EXAMPLE

To demonstrate the usefulness of distributed metamodeling, an electromagnetic simulator is used. In this particular case, the simulator computes the scattering parameters for a step discontinuity in a rectangular waveguide. The inputs consists of input frequency, the gap height and the gap length. The outputs are the scattering parameters of this 2-port system. Figure 7 shows an approximate plot of the input-output relation at three different discrete frequencies.

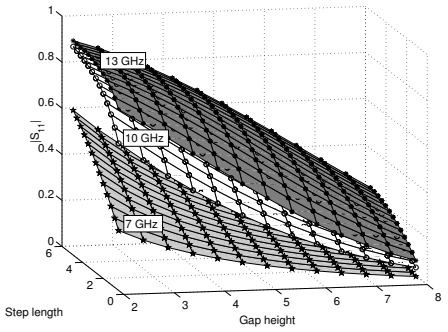

Figure 7: A Plot of $|S_{11}|$, the Modulus of the First Scattering Parameter

Adaptive modeling and *sequential design* techniques were used on this dataset, first using rational metamodels and then using ANN metamodels. Figures 8 and 9 show the evolution of the errors as a function of the number of samples. The horizontal axis contains the number of samples, while the vertical axis gives percentages. Each shaded region corresponds to a specific error range given in the legend. Each of the metamodels was compared to reference data on a 50^3 full factorial design for verification. The height of each region depicts the percentage of these verification points that have an error within the error range for that region. Note the difference in scale on the horizontal axis between the two figures. Rational metamodels tend to be more suitable for relatively simple physical systems, as they can capture the system's global behavior more easily. The ANN metamodels are expected to do much better when the system's outputs are more complex.

As EM-simulators are computationally expensive, the use of grid computing is crucial for fast and efficient metamodeling. Although the previous tests were run on a single machine, we also conducted some distributed tests. In order to emulate a real life situation, the step discontinuity simulator code was first slowed down by adding some sleeps. The APST framework was used to distribute the tests over a small testbed of 6 desktop computers. It would have been preferable to run these tests on "real" grid resources, but this is not really an issue since (1) we are currently

only interested in a proof-of-concept, and (2) the speedup is expected to increase linearly as the number of machines increases. The total time spent on running simulations when using one machine averages to 2170 seconds, while in the distributed setting the simulation time averages to 561 seconds. The reason an ideal speedup of 6 was not reaches, is that the adaptive sampling loop selects 3 to 6 samples each iteration. Some machines are running idle part of the time.

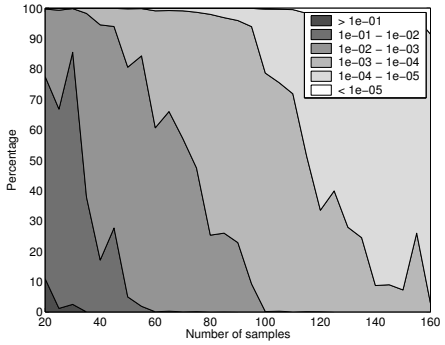

Figure 8: Error Percentage Plots Using Rational Metamodels

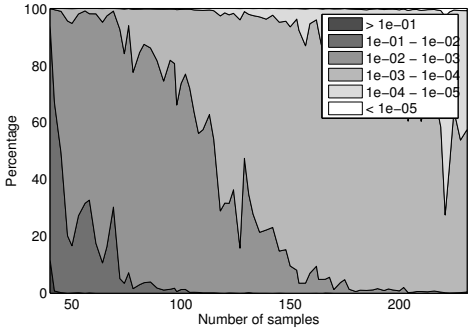

Figure 9: Error Percentage Plots Using ANN Metamodels

8 CONCLUSIONS

Grid computing, adaptive modeling and sequential design have proven useful for many applications. This work integrates all into a single metamodeling framework (that we have implemented as the publically available M3 Toolbox) that benefits from the advantages of all three of these key concepts. Future research will improve and fine-tune the concepts laid out in this paper.

9 ACKNOWLEDGMENTS

The authors would like to thank Robert Lehmensiek and Petrie Meyer for providing the MATLAB code for the example of section 7.

This research was supported by the FWO - Flemish Fund for Scientific Research.

REFERENCES

Abramson, D., A. Lewis, T. Peachey, and C. Fletcher. 2001. An automatic design optimization tool and its application to computational fluid dynamics. In *Proceedings of the 2001 ACM/IEEE conference on Supercomputing (CDROM)*, 25–25.

Barton, R. R. 1998. Simulation meta-models. *Proceedings of the 1998 Winter Simulation Conference*:167–174.

Berman, F., H. Casanova, A. Chien, K. Cooper, H. Dail, A. Dasgupta, W. Deng, J. Dongarra, and Co.. 2005. New grid scheduling and rescheduling methods in the GrADS project. *International Journal of Parallel Programming (IJPP)* Volume 33 (2-3): 209–229.

Buyya, R., and S. Venugopal. 2004. The gridbus toolkit for service oriented grid and utility computing: An overview and status report. In *Proceedings of the First IEEE International Workshop on Grid Economics and Business Models (GECON 2004)*, 19–36.

Carr, J., R. K. Beatson, J. Cherrie, T. Mitchell, W. Fright, B. McCallum, and T. Evans. 2001. Reconstruction and representation of 3D objects with Radial Basis Functions. ACM SIGGRAPH 2001, Los Angeles, CA.

Caruana, R., S. Lawrence, and C. L. Giles. 2001. Overfitting in neural networks: Backpropagation, conjugate gradient, and early stopping. In *Advances in Neural Information Processing Systems*. Denver, Colorado.

Chambers, M., and C. A. Mount-Campbell. 2002, September. Process optimization via neural network metamodeling. *International Journal of Production Economics* 79 (2): 93–100.

Chaveesuk, R., and A. Smith. 2003. Economic valuation of capital projects using neural network metamodels. *The Engineering Economist* 48 (1): 1–30.

Dahm, I., and J. Ziegler. 2002. Using artificial neural networks to construct a meta-model for the evolution of gait patterns of four-legged walking robots. In *Proc. of the 5th Conf. on Climbing and Walking Robots and the Support Technologies for Mobile Machines (CLAWAR2002)*, ed. P. Bidaud and F. B. Amar, 825–832. Bury St. Edmunds, UK: Professional Engineering Publ.

De Geest, J., T. Dhaene, N. Faché, and D. De Zutter. 1999. Adaptive CAD-model building algorithm for general planar microwave structures. *IEEE Transactions on Microwave Theory and Techniques* 47 (9): 1801–1809.

Devabhaktuni, V., M. Yagoub, Y. Fang, J.Xu, and Q. Zhang. 2001. Neural networks for microwave modeling: model development issues and nonlinear modeling techniques. *International Journal of RF and Microwave CAE* 11:4–21.

Eres, M. H., G. E. Pound, Z. Jiao, J. L. Wason, F. Xu, A. J. Keane, and S. J. Cox. 2005. Implementation and utilisation of a grid-enabled problem solving environment in matlab. *Future Generation Comp. Syst.* 21 (6): 920–929.

Erwin, D. 2002. Unicore - a grid computing environment. *Concurrency-Practice and Experience* 14:1395–1410.

Foster, I., C. Kesselman, and S. Tuecke. 2001. The Anatomy of the Grid: Enabling Scalable Virtual Organizations. *Lecture Notes in Computer Science* 2150.

Giunta, A., and M. Eldred. 2000. Implementation of a trust region model management strategy in the DAKOTA optimization toolkit. In *Proceedings of the 8th AIAA/USAF/NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimization, Long Beach, CA*.

Hendrickx, W., and T. Dhaene. 2005a. Multivariate modelling of complex simulation-based systems. *Proceedings of the IEEE NDS 2005 conference*:212–216.

Hendrickx, W., and T. Dhaene. 2005b. Sequential design and rational metamodelling. In *Proceedings of the 2005 Winter Simulation Conference*, ed. M. Kuhl, S. N. M., F. B. Armstrong, and J. A. Joines, 290–298.

Hillbrand, C., and D. Karagiannis. 2002. Using artificial neural networks to prove hypothetic cause-and-effect relations: A metamodel-based approach to support strategic decisions. In *ICEIS*, 367–373.

Huet, F., D. Caromel, and H. E. Bal. 2004. A high performance java middleware with a real application. In *SC*, 2.

Kansa, E. J. 1990. Multiquadrics, a scattered data approximation scheme with applications to computational fluid-dynamics I. *Computers and Mathematics with Applications* 19:127–146.

Kleijnen, J. P. C., and W. C. M. Van Beers. 2003. Application driven sequential designs for simulation experiments: Kriging metamodels. Technical report. CentER discussion paper No. 2003-33.

Lamecki, A., P. Kozakowski, and M. Mrozowski. 2003. Efficient implementation of the Cauchy method for automated CAD-model construction. *IEEE Microwave and wireless components letters* (7): 268–270.

Lamecki, A., P. Kozakowski, and M. Mrozowski. 2004. CAD-model construction based on adaptive radial basis functions interpolation technique. 3:799–802. Microwaves, Radar and Wireless Communications, 2004. MIKON-2004.

Lehmensiek, R., and P. Meyer. 2001, 8. Creating accurate multivariate rational interpolation models for microwave circuits by using efficient adaptive sampling to minimize the number of computational electromagnetic analyses. *IEEE Trans. Microwave Theory Tech.* 49 (8): 1419–.

Martin, J. D., and T. W. Simpson. 2002. Use of adaptive metamodeling for design optimization. In *Proceedings of the 9th AIAA/ISSMO Symposium on Multidisciplinary Analysis and Optimization, Atlanta, GA*.

Natrajan, A., A. Nguyen-Tuong, M. Humphrey, M. Herrick, B. P. Clarke, and A. S. Grimshaw. 2002. The legion grid portal. *Concurrency and Computation: Practice and Experience* 14 (13-15): 1365–1394.

Neural Networks. The neural network FAQ: ftp://ftp.sas.com/pub/neural/faq.html.

Ng, H.-K., D. Lim, Y.-S. Ong, B.-S. Lee, L. Freund, S. Parvez, and B. Sendhoff. 2005. A multi-cluster grid enabled evolution framework for aerodynamic airfoil design optimization. In *ICNC (2)*, 1112–1121.

Ng, H.-K., Y.-S. Ong, T. Hung, and B.-S. Lee. 2005. Grid enabled optimization. In *EGC*, 296–304.

Ong, Y. S., P. B. Nair, and A. J. Keane. 2003. Evolutionary optimization of computationally expensive problems via surrogate modeling. *American Institute of Aeronautics and Astronautics Journal* 41 (4): 687–696.

Ong, Y. S., P. B. Nair, A. J. Keane, and K. W. Wong. 2004. *Studies in fuzziness and soft computing series*, Chapter Surrogate-Assisted Evolutionary Optimization Frameworks for High-Fidelity Engineering Design Problems, Knowledge Incorporation in Evolutionary Computation, 307 – 331. Springer Verlag.

Panayiotou, C. G., C. G. Cassandras, and W.-B. Gong. 2000. Model abstraction for discrete event systems using neural networks and sensitivity information. In *Proceedings of the 2000 Winter Simulation Conference*, 335–341: Society for Computer Simulation International.

Parmee, I., J. Abraham, M. Shackelford, O. F. Rana, and A. Shaikhali. 2005. Towards autonomous evolutionary design systems via grid-based technologies. In *Proceedings of ASCE Computing in Civil Engineering, Cancun, Mexico*.

Pota, S., K. Kuntner, and Z. Juhasz. 2003, May. Jini network technology and grid systems. In *MIPRO 2003, Hypermedia and Grid Systems, Opatija, Croatia*, 144–147.

Sacks, J., W. J. Welch, T. Mitchell, and H. P. Wynn. 1989. Design and analysis of computer experiments. *Statistical science* 4 (4): 409–435.

Santner, T. J., B. J. Williams, and W. I. Notz. 2003. *The design and analysis of computer experiments*. Springer series in statistics. Springer.

Sarle, W. 1995. Stopped training and other remedies for overfitting. In *Proceedings of the 27th Symposium on the Interface of Computing Science and Statistics*, 352–360.

Søndergaard, J. 2003. *Optimization using surrogate models by the space mapping technique*. Ph. D. thesis.

Taylor, I., I. Wang, M. Shields, and S. Majithia. 2005. Distributed computing with Triana on the Grid. *Concurrency and Computation:Practice and Experience* 17 (1–18).

Valiant, L. G. 1988. Functionality in neural nets. In *Computational Learning Theory (COLT)*, 28–39.

Wendland, H. 2005. *Scattered data approximation*. Cambridge University Press.

White, H. 1990. Connectionist nonparametric regression: multilayer feedforward networks can learn arbitrary mappings. *Neural Networks* 3 (5): 535–549.

Xiao, S., B. Z. Wang, X. Zhong, and G. Wang. 2003, July. Wideband mobile antenna design based on artificial neural network models. *International Journal of RF and Microwave Computer-Aided Engineering* 13 (4): 316–320.

Yao, X. 1999, September. Evolving artificial neural networks. *Proceedings of the IEEE* 87 (9): 1423–1447.

Zhang, Q., K. Gupta, and V. Devabhaktuni. 2003, March. Artificial neural networks for RF and microwave design: from theory to practice. *IEEE Trans. Microwave Theory Tech* 51:1339–1350.

AUTHOR BIOGRAPHIES

WOUTER HENDRICKX is a PhD student with the research group Computer Simulation and Modeling of complex systems (COMS) of the University of Antwerp, Belgium. His e-mail address is <wouter.hendrickx@ua.ac.be> and his web page can be found on the site of the research group <www.coms.ua.ac.be>.

DIRK GORISSEN is a PhD student with the COMS research group of the University of Antwerp, Belgium. He is currently supported by the FWO - Flemish Fund for Scientific Research. His e-mail address is <dirk.gorissen@ua.ac.be> and his web page can be found on <www.coms.ua.ac.be>.

TOM DHAENE received his PhD degree in electrotechnical engineering from the University of Gent, Belgium, in 1993. His PhD research focused on different aspects of full-wave electro-magnetic circuit modeling, transient simulation, and time-domain characterisation of high-speed interconnections. Since September 2000, he is a Professor at the University of Antwerp, Belgium, in the Department of Mathematics and Computer Science. He is head of the COMS research group (Computer Modeling and Simulation). His research interests are in the field of metamodeling, circuit and electromagnetic modeling, and adaptive system identification. His e-mail address is <tom.dhaene@ua.ac.be> and his web page can be found on the site of the research group <www.coms.ua.ac.be>.