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

MOVING LEAST SQUARES REGRESSION FOR HIGH DIMENSIONAL SIMULATION METAMODELING

Peter Salemi
Barry L. Nelson
Jeremy Staum

Department of Industrial Engineering and Management Sciences
Northwestern University
2145 Sheridan Road
Evanston, IL, 60208-3119, U.S.A.

ABSTRACT

Interpolation and smoothing methods form the basis of simulation metamodeling. In high dimensional metamodeling problems, larger numbers of design points are needed to build an accurate metamodel. This paper introduces a procedure to implement a smoothing method called Moving Least Squares regression in high dimensional metamodeling problems with a large number of design points. We test the procedure with two queueing examples: a multi-product M/G/1 queue and a multi-product Jackson network.

1 INTRODUCTION

A simulation metamodel is a model of a simulation experiment. Simulation metamodeling allows the experimenter to obtain more benefit from a simulation because the simulation can be run when time is plentiful, and quick predictions can be made when decision-making time is scarce or expensive. All simulation metamodeling techniques involve running the simulation at a set of design points and using this information to predict the value we would get from running the simulation at other points in the design space, simulated or not. In higher dimensions, more design points are needed to get an accurate metamodel. Metamodeling techniques that use interpolation or smoothing such as kriging and stochastic kriging (Ankenman, Nelson, and Staum 2010) experience difficulties in higher dimensions because this increase in design points causes numerical instabilities and high computation times. Gaussian process techniques that aim to reduce the number of points used for fitting and prediction (Snelson and Ghahramani 2006) use pseudo-inputs which maximize the likelihood that the actual data was drawn. Methods such as these require solving an optimization problem with many decision variables, which can be too large for high-dimensional problems.

Another metamodeling approach that can deal with large numbers of design points in high dimensional problems uses a multi-stage interpolation technique (Haaland and Qian 2011), but does not yet have a practical method for implementation. High-Dimensional Model Representation (HDMR) (see, for example, Shan and Wang (2010)), in an effort to work with lower-dimensional functions, uses an ANOVA-like decomposition to decompose the underlying function into component functions. However, HDMR requires a search to determine the important terms in the decomposition, and can be ill-suited for problems where higher-order terms are important, since this search requires sequential designs to efficiently determine if a term in the decomposition is in fact important. For higher-dimensional problems, the number of terms in the decomposition increases exponentially.

Lafferty and Wasserman (2008) introduced a greedy algorithm to choose the bandwidths for locally weighted least squares linear regression. This greedy algorithm continually shrinks the bandwidth in a

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

particular direction, if the derivative of the estimator with respect to that bandwidth is larger than some threshold value. This threshold value is dependent on the number of design points and the variance of the observations, which are assumed to be equal throughout the design space. This method also assumes a small number of relevant variables, so if the number of relevant variables is not small, the greedy nature of the algorithm may lead to poor bandwidth choices.

The smoothing method that we employ is Moving Least Squares (MLS) regression (Lancaster and Salkauskas 1981; Levin 1998). The main applications of MLS regression occur in partial differential equations and image processing which feature low dimensional problems with a large number of design points. For example, image processing problems have only two or three dimensions. We are concerned with the case where there are a large number of design points in a high dimensional setting. The approximation error for MLS regression is usually a function of the maximum distance from a design point to its nearest neighbor, which can be made arbitrarily small by adding in more design points. Thus, we expect MLS regression to work well for simulation metamodeling in high dimensions when the simulation can be run for a long time so we can have a very large number of design points. This paper introduces a procedure to apply MLS regression for this case.

2 EXPERIMENT DESIGN

We are interested in predicting a certain system performance measure, for example the expected waiting time in a queue. Denote the system performance measure at a design point, $\mathbf{x}$, by $y(\mathbf{x})$. In the case of a queue, $\mathbf{x}$ could include arrival rates, service rates, etc. Denote the design space, the set of all possible values of the design variables, by $\mathbb{X}$. Furthermore, let $\{\{\mathbb{X}_n, \mathbb{R}_n\}; n \geq 0\}$ denote a sequence of experiment designs, where $\mathbb{X}_n = (\mathbf{x}_1^n, \mathbf{x}_2^n, \ldots, \mathbf{x}_n^n)$ is the set containing the first n generated design points, and $\mathbb{R}_n = (R_1^n, R_2^n, \ldots, R_n^n)$ is the set containing the number of replications we allocate to each design point in $\mathbb{X}_n$. In other words, for the nth sequential design we allocate R_i^n replications to $\mathbf{x}_i^n$. This sequential procedure is a way to analyze an asymptotic regime of increasing simulation effort, however we are not designing a sequential procedure. We assume that

$$\lim_{n\to\infty} \frac{1}{n} \sum_{i=1}^{n} \mathbb{I}\{\mathbf{x}_i^n \in \mathrm{A}\} = \int_{\mathrm{A}} g(\mathbf{z}) d\mathbf{z},$$

for all rectangles $A \subseteq \mathbb{X}$, where $g(\mathbf{z})$ is the limiting density of design points at $\mathbf{z} \in \mathbb{X}$. We also assume that

$$\lim_{n\to\infty} \frac{1}{C_n} \sum_{i=1}^{n} \mathbb{I}\{\mathbf{x}_i^n \in \mathrm{A}\} R_i^n = \int_{\mathrm{A}} \tilde{g}(\mathbf{z}) d\mathbf{z},$$

for all rectangles $A \subseteq \mathbb{X}$, where $C_n = \sum_{i=1}^{n} R_i^n$ is the total number of replications allocated in the nth design, and $\tilde{g}(\mathbf{z})$ is the limiting density of effort spent at $\mathbf{z} \in \mathbb{X}$. In our procedure, we assume that g and $\tilde{g}$ are uniform densities on the unit hypercube $[0,1]^d$, i.e., $g(\cdot) = \tilde{g}(\cdot) = 1$ and $\mathbb{X} = [0,1]^d$.

For the nth sequential design, we run the simulation at $\mathbb{X}_n = (\mathbf{x}_1^n, \mathbf{x}_2^n, \ldots, \mathbf{x}_n^n)$. At design point $\mathbf{x}_i^n$ we run R_i^n replications of the simulation, and we denote the system performance measure for the jth replication by $\mathrm{Y}_j^n(\mathbf{x}_i^n)$. The estimate that we obtain at design point $\mathbf{x}_i^n$ is the sample average $\bar{\mathrm{Y}}^n(\mathbf{x}_i^n; R_i^n) = (1/R_i^n) \sum_{j=1}^{R_i^n} \mathrm{Y}_j^n(\mathbf{x}_i^n)$. We will also need an estimate of the variance $\sigma^2(\mathbf{x}_i^n)$ of a replication at $\mathbf{x}_i^n$ which we estimate by the sample variance $S^2(\mathbf{x}_i^n; R_i^n) = (1/(R_i^n - 1)) \sum_{j=1}^{R_i^n} (\mathrm{Y}_j^n(\mathbf{x}_i^n) - \bar{\mathrm{Y}}^n(\mathbf{x}_i^n; R_i^n))^2$.

For ease of notation, we drop the superscript notation for the nth sequential design and let $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n$ denote the design points in the nth sequential design, and $R_1, R_2, \ldots, R_n$ denote the replications allocated to each design point in $\mathbb{X}_n$

3 TWO LOCAL SMOOTHING APPROACHES

In this section we present two smoothing methodologies, the first of which is MLS regression. Although this is the approach that we will use to predict the system performance measure, we will also introduce another methodology called Multivariate Locally Weighted Least Squares Linear Regression (Ruppert and Wand 1994), which we only use to gain insight into how to implement MLS regression.

Both approaches require a weight function of the form $K_{\mathbf{H}}(\mathbf{u}) = |\mathbf{H}|^{-1/2}K(\mathbf{H}^{-1/2}\mathbf{u})$, where K is a compactly supported d-variate kernel such that $\int K(\mathbf{u})d\mathbf{u} = 1$, and $\mathbf{H}$ is a $d \times d$ symmetric positive definite matrix depending on n. The matrix $\mathbf{H}$ is called the bandwidth matrix and its entries are called the bandwidth parameters. The bandwidth matrix determines the shape of the contours of the weight function $K_{\mathbf{H}}$. The number of non-zero entries in the bandwidth matrix represents the number of bandwidth parameters that must be chosen before one can apply either of the two smoothing methodologies. In high dimensional problems, allowing the bandwidth matrix to have non-zero values off the diagonal would result in too many bandwidth parameters that need to be chosen. Therefore we will only consider diagonal bandwidth matrices in our procedure. A diagonal bandwidth matrix will cause the contours of the kernel to be parallel to the main coordinate axes, whereas a full bandwidth matrix would allow the contours of the kernel to be arbitrarily rotated. We do not dwell on this restriction because it has been shown that the improvement gained by allowing off-diagonal entries to be non-zero is not nearly as great as the benefit from allowing the diagonal entries to vary from one another (Wand and Jones 1993). Furthermore, the choice of kernel is not as important as the choice of bandwidth matrix, $\mathbf{H}$ (Wand and Jones 1993). We introduce a kernel which is a function of the maximum norm, given by $||\mathbf{u}||_\infty = \max(|u_1|, |u_2|, \ldots, |u_d|)$ for $\mathbf{u} \in \mathbb{R}^d$. This kernel is

$$K(\mathbf{u}) = [1 - \max(|u_1|, |u_2|, \ldots, |u_d|)]\, \mathbb{I}\{\max(|u_1|, |u_2|, \ldots, |u_d|) \leq 1\},$$

and its support is shown in Figure 1(a). This kernel has a compact rectangular support region with the bandwidth parameters lying on the diagonal of the bandwidth matrix determining the half-length of each edge of the rectangle. This kernel satisfies assumptions (A1) and (A3) of the kernels used in Ruppert and Wand (1994), so it is indeed a bona-fide kernel. The diagonal bandwidth matrix $\mathbf{H} = \text{diag}(h_1^2, \ldots, h_d^2)$ will yield the weight function

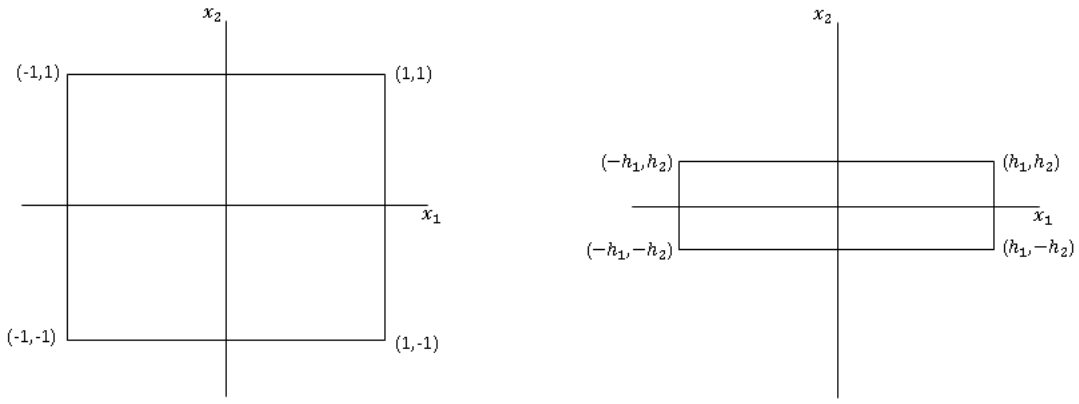
$$K_{\mathbf{H}}(\mathbf{u}) = \frac{1}{h_1 \cdots h_d}\left[1 - \max\left(\left|\frac{u_1}{h_1}\right|, \ldots, \left|\frac{u_d}{h_d}\right|\right)\right]\mathbb{I}\left\{\max\left(\left|\frac{u_1}{h_1}\right|, \ldots, \left|\frac{u_d}{h_d}\right|\right) \leq 1\right\},$$

whose support is shown in Figure 1(b), in relation to the support of the kernel in Figure 1(a).

3.1 Moving Least Squares Regression

MLS regression reinterprets the metamodeling problem as predicting $y(\mathbf{x})$ for any specific $\mathbf{x} \in \mathbb{X}$ instead of building a metamodel to approximate the entire function y. Each design point is assigned a weight, which is similar to weighted least squares regression except that the weight given to a design point depends on the particular prediction point, with the weight being determined by the weight function, $K_{\mathbf{H}}(\cdot)$. Therefore, every time we predict the function value at a prediction point we solve a different weighted least squares problem. In the following, let Π_k^d denote the space of d-variate polynomials of degree k, and let $p_1, p_2, \ldots, p_m$ denote the basis functions of Π_k^d. In this paper, we take the basis functions of Π_k^d to be the standard basis which is the set of $\binom{d+k}{k}$ monomials. The polynomial, $\hat{y}_{\mathbf{x}_0}^{\text{MLS}}$, used for approximating the function value $y(\mathbf{x}_0)$ at the prediction point $\mathbf{x}_0$ is

$$\hat{y}_{\mathbf{x}_0}^{\text{MLS}} = \arg\min_{p \in \Pi_k^d}\left\{\sum_{i=1}^{n}(\bar{\text{Y}}(\mathbf{x}_i; R_i) - p(\mathbf{x}_i))^2 K_{\mathbf{H}}(\mathbf{x}_i - \mathbf{x}_0)\right\}. \tag{1}$$



(a) (b)

Figure 1: (a) In two dimensions, the compact support of the kernel $K(\cdot)$ is the unit hypercube. (b) An example of the compact support of the weight function $K_{\mathbf{H}}(\cdot)$ in two dimensions, with $h_1 > h_2$.

This is the standard approach to MLS regression (Bos and Salkauskas 1989). The optimal solution to this problem is obtained from the weighted least squares solution: $\hat{y}^{\text{MLS}}_{\mathbf{x}_0}(\mathbf{x}) = \mathbf{P}(\mathbf{x})^\top \left(\mathbf{P}^\top \mathbf{W}(\mathbf{x}_0)\mathbf{P}\right)^{-1} \mathbf{P}^\top \mathbf{W}(\mathbf{x}_0)\mathbf{Y}$, where $\mathbf{P}(\mathbf{x}) = [p_1(\mathbf{x}), p_2(\mathbf{x}), \ldots, p_m(\mathbf{x})]^\top$, $\mathbf{W}(\mathbf{x}_0) = \text{diag}\{K_{\mathbf{H}}(\mathbf{x}_1 - \mathbf{x}_0), K_{\mathbf{H}}(\mathbf{x}_2 - \mathbf{x}_0) \ldots, K_{\mathbf{H}}(\mathbf{x}_n - \mathbf{x}_0)\}$, $\mathbf{P}$ is the $n \times m$ matrix whose i^{th} row is $[p_1(\mathbf{x}_i), p_2(\mathbf{x}_i), \cdots, p_m(\mathbf{x}_i)]$, and $\mathbf{Y} = \left[\bar{Y}(\mathbf{x}_1; R_1), \bar{Y}(\mathbf{x}_2; R_2), \ldots, \bar{Y}(\mathbf{x}_n; R_n)\right]^\top$. For each prediction point, $\mathbf{x}_0 \in \mathbb{X}$, we get a different approximating polynomial, $\hat{y}^{\text{MLS}}_{\mathbf{x}_0}$.

The minimization in Problem 1 is done over the polynomial space Π^d_k, which gives MLS regression an important property called the polynomial reproduction property. Since d is the dimension of the design space, the only factor that we are able to choose is k. The dimension of Π^d_k is $\binom{d+k}{k}$, so for large d we must be careful to not pick k too large. Otherwise, we must invert a $\binom{d+k}{k} \times \binom{d+k}{k}$ matrix to obtain the prediction, which is infeasible when d and k are large. We will use the space of linear polynomials, Π^d_1. It is clear that design points closer to the prediction point will have more influence on the polynomial used for prediction.

3.2 Locally Weighted Least Squares Linear Regression

The weight function depends on bandwidth parameters that determine the shape and size of its contours. The main problem in MLS regression is obtaining these bandwidth parameters so that they are optimal with respect to some criterion. Locally Weighed Least Squares Linear Regression is a smoothing methodology that is similar to MLS regression when we use the space Π^d_1 in the MLS formulation. This methodology uses a first-order Taylor expansion to approximate the function value at the prediction point, so both the Locally Weighted Least Squares Linear Regression and the MLS regression methodology can reproduce linear polynomials. However, using Locally Weighted Least Squares Linear Regression we can get an approximation to the Asymptotic Mean Squared Error (AMSE) at a prediction point in terms of the weight function, the second derivatives of y at the prediction point, the density of the design points at the prediction point, and the variance in the function observations at the prediction point. We can use this approximation to the AMSE to choose the weight function parameters optimally and then use the obtained weight function in MLS.

We assume that the estimates obtained from the simulation are of the form $\bar{Y}(\mathbf{x}_i; R_i) = y(\mathbf{x}_i) + \frac{\sigma(\mathbf{x}_i)}{\sqrt{R_i}}\varepsilon_i$, where $\sigma(\mathbf{x}_i)$ is the standard deviation of a replication at $\mathbf{x}_i$ and the ε_i are mutually independent and identically distributed random variables with zero mean and unit variance. We assume that the function y is

twice differentiable at the prediction point. The prediction at $\mathbf{x}_0$ is $\hat{y}^{\mathrm{LOC}}(\mathbf{x}_0) := \hat{\beta}_0$, where $\hat{\beta}_0$ is from the solution to the following problem:

$$\min_{\beta_0,\beta_1} \sum_{i=1}^{n} \left\{ \bar{Y}(\mathbf{x}_i; R_i) - \beta_0 - \beta_1^\top (\mathbf{x}_i - \mathbf{x}_0) \right\}^2 K_{\mathbf{H}}(\mathbf{x}_i - \mathbf{x}_0).$$

For the diagonal bandwidth matrix, $\mathbf{H} = \mathrm{diag}(h_1^2, \ldots, h_d^2)$, the AMSE of the estimator $\hat{\beta}_0$ of the function value at $\mathbf{x}_0$ for large n is approximately

$$\begin{aligned} AMSE &\approx \frac{1}{4}\mu_2(K)^2 \mathrm{tr}(\mathbf{H}\nabla_y^2(\mathbf{x}_0))^2 + \frac{R(K)\sigma^2(\mathbf{x}_0)}{C_n|\mathbf{H}|^{1/2}\tilde{g}(\mathbf{x}_0)} \\ &= \frac{1}{4}\mu_2(K)^2 \left(h_1^2 \frac{\partial^2 y(\mathbf{x}_0)}{\partial x_1^2} + \cdots + h_d^2 \frac{\partial^2 y(\mathbf{x}_0)}{\partial x_d^2} \right)^2 + \frac{R(K)\sigma^2(\mathbf{x}_0)}{C_n \tilde{g}(\mathbf{x}_0) h_1 \cdots h_d}, \end{aligned} \tag{2}$$

where $\mu_2(K) = \int_{\mathbb{R}} \mathbf{x}_i K(\mathbf{x})\, d\mathbf{x}$, $R(K) = \int_{\mathbb{R}} K(\mathbf{x})^2\, d\mathbf{x}$, $\nabla_y^2(\mathbf{x}_0)$ is the Hessian of y evaluated at $\mathbf{x}_0$, and $\sigma^2(\mathbf{x}_0)$ is the variance of a replication at the prediction point. We assume that $\sigma^2(\cdot)$ is continuous, and estimate it using the sample variances at the design points via the method discussed in Section 5.1.

Equation 2 shows the bias-variance trade-off with respect to the bandwidth parameters, $h_1, \ldots, h_d$. The first term in the sum represents the bias of the estimator, while the second term represents variance. When the bandwidth parameters are small, the bias of the estimator $\hat{\beta}_0$ is small, but fewer design points are used in the prediction, making the variance of the estimator high. For large bandwidth parameters, the opposite happens. Thus, we can see how the bias-variance trade-off would affect the bandwidth parameters if we were to choose them by minimizing the AMSE equation. In the bias term, given by the first part of Equation 2, directions corresponding to larger changes in the underlying function (i.e., larger second partial derivatives) result in smaller bandwidth parameters corresponding to those directions. This regulates the bias because weight decays more rapidly in directions where there are larger changes in the underlying function. In the variance term, given by the second part of Equation 2, a higher variance at the prediction point, $\sigma^2(\mathbf{x}_0)$, with all other parameters fixed, will increase the bandwidth parameters, incorporating more design points in the approximation and therefore filtering out the larger noise. The limiting density of effort spent at the prediction point, $\tilde{g}(\mathbf{x}_0)$, with all other parameters fixed, will give smaller bandwidth parameters to prediction points in regions of higher density. Intuitively, this is because in regions where we have spent the most simulation effort, we would like the prediction to be based on design points closer to the prediction point, making the bandwidth parameters smaller, and hence decreasing the bias.

4 MOVING LEAST SQUARES PROCEDURE

We are now ready to discuss the procedure for implementing MLS regression in high dimensional problems. For our procedure we will use the weight function $K_{\mathbf{H}}(\cdot)$, given by

$$K_{\mathbf{H}}(u) = \frac{1}{(h_1^l \vee h_1^r) \cdots (h_d^l \vee h_d^r)} \left[1 - \max\left(\left| \frac{u_1}{(h_1^l \vee h_1^r)} \right|, \ldots, \left| \frac{u_d}{(h_d^l \vee h_d^r)} \right| \right) \right]^+,$$

with the associated 'prediction window' defined by the region $\Omega := \left\{ \mathbf{x} \in \mathbb{X} : |x_i - x_{0,i}| \leq h_i^l \vee h_i^r, \forall i = 1, \ldots, d \right\}$. The variable h_i^l denotes the distance from the left edge of the prediction window to the prediction point in the ith coordinate, and the variable h_i^r denotes the distance from the right edge to the prediction point. The bandwidth parameters, $h_1^l \vee h_1^r, \ldots, h_d^l \vee h_d^r$, determine the bandwidth in the corresponding coordinate direction. For example, $h_1^l \vee h_1^r$ determines how fast the weight decays in the direction along the first basis vector of $\mathbb{R}^d$. The region Ω is the intersection of the compact support of the kernel $K_{\mathbf{H}}$ and the design

space $\mathbb{X}$, so the design points that fall in the region will be the design points used for prediction, hence the name 'prediction window'.

To find the optimal bandwidth parameters $({h_1^l}^* \vee {h_1^r}^*), \ldots, ({h_d^l}^* \vee {h_d^r}^*)$ we solve Problem 3, whose objective function is a modification of the AMSE equation. Estimation of $\sigma^2(\mathbf{x}_0)$ and $\frac{\partial^2 y(\mathbf{x}_0)}{\partial x_1^2}, \ldots, \frac{\partial^2 y(\mathbf{x}_0)}{\partial x_d^2}$ is discussed in Section 5.

$$\begin{aligned}
\min_{\{h_1^l, h_1^r, \ldots, h_d^l, h_d^r\}} \quad & \frac{1}{4}\mu_2(K)^2 \left(\left(\frac{h_1^l + h_1^r}{2} \right)^2 \left| \frac{\partial^2 y(\mathbf{x}_0)}{\partial x_1^2} \right| + \cdots + \left(\frac{h_d^l + h_d^r}{2} \right)^2 \left| \frac{\partial^2 y(\mathbf{x}_0)}{\partial x_d^2} \right| \right)^2 \\
& \qquad + \frac{R(K)\sigma^2(\mathbf{x}_0)}{C_n \tilde{g}(\mathbf{x}_0) \left(\frac{h_1^l + h_1^r}{2} \right) \cdots \left(\frac{h_d^l + h_d^r}{2} \right)} \qquad (3) \\
\text{s.t.} \quad & \dim(\Pi_1^d) + \delta \leq ng(\mathbf{x}_0)(h_1^l + h_1^r) \cdots (h_d^l + h_d^r) \leq \text{Mass}_{\text{UB}} \\
& 0 \leq h_i^l \leq x_{0,i}, \qquad \forall i = 1, 2, \ldots, d \\
& 0 \leq h_i^r \leq 1 - x_{0,i}, \qquad \forall i = 1, 2, \ldots, d.
\end{aligned}$$

The motivation for the first constraint is the following. Without an upper bound on the number of design points that are included in the prediction window, the optimal prediction window might contain too many points for the computing time to be acceptable. The upper bound is denoted by Mass_{UB}, and we use $\text{Mass}_{\text{UB}} = 2000$ in this paper (of course, this could be much higher depending on computing power). We also want to ensure that the number of design points is at least the dimension of Π_1^d and to protect against having linearly dependent columns in the matrix $\mathbf{P}$ of the solution to Problem 1, which we do by adding the constant δ to the dimension of Π_k^d in the lower bound. We use $\delta = 3d$. An approximation to the number of design points that lie within the prediction window is $ng(\mathbf{x}_0)(h_1^l + h_1^r)(h_2^l + h_2^r) \cdots (h_d^l + h_d^r)$. This can be interpreted as the density of design points at the prediction point $ng(x_0)$ times the volume of the prediction window which gives us the total number of design points included in the prediction window. The limiting density of design points that makes $ng(\mathbf{x}_0)(h_1^l + h_1^r)(h_2^l + h_2^r) \cdots (h_d^l + h_d^r)$ the best approximation is the uniform density, which is the density we use in this procedure.

The second derivatives in the AMSE equations have been replaced by the absolute values of the second derivatives to ensure that the bandwidth parameters behave well when some second derivatives are positive and some are negative. To see the motivation for this change, consider the case where f has both positive and negative second partial derivatives. By setting the bandwidth parameters in the proper proportion to each other, we can kill the asymptotic bias. Then we can reduce the variance by increasing the size of the window. However this increase in window size reduces the validity of the bias approximation, so for a fixed value of n Equation 2 may cease to be a good approximation to the AMSE when a large window is used. Thus, we take a conservative approach to the window size and use an upper bound on the approximate AMSE.

Problem 3 can be solved using the **Bandwidth Procedure** in the appendix. Denote the optimal solution by $\mathbf{h}^* = \{{h_1^l}^*, {h_1^r}^*, \ldots, {h_d^l}^*, {h_d^r}^*\}$. The weight function used for prediction is given by

$$K_{\mathbf{H}}^*(u) = \frac{1}{({h_1^l}^* \vee {h_1^r}^*) \cdots ({h_d^l}^* \vee {h_d^r}^*)} \left[1 - \max \left(\left| \frac{u_1}{({h_1^l}^* \vee {h_1^r}^*)} \right|, \ldots, \left| \frac{u_d}{({h_d^l}^* \vee {h_d^r}^*)} \right| \right) \right]^+.$$

The approximate AMSE Equation 2 is the result of using locally weighted least squares linear regression for prediction, which results in second derivatives in the bias term. The second derivatives arise because we use a linear approximation, and therefore cannot account for higher-order derivatives. The bias term in

Equation 2 is only an approximation to the asymptotic bias at the prediction point, and will underestimate the amount of bias in the prediction window since the approximation only considers the main second partial derivatives, and assumes the prediction window is symmetric about the prediction point. In an effort to further reduce the bias, we use a stepwise regression method to determine if there are necessary interaction and second-order terms that should be included in the model. Denote the prediction window of $K_{\mathbf{H}}^*$ by Ω^*, and let $\mathbf{x}_1^*, \ldots, \mathbf{x}_{|\Omega^*|}^*$ denote the $|\Omega^*|$ design points that fall into the prediction window. The stepwise procedure is as follows:

1. Initialize the $|\Omega^*| \times (d+1)$ regression matrix $\mathbf{X}_{\Omega^*}$, with ith row $(1, x_{1,1}^*, \ldots, x_{1,d}^*)$, and let $\mathbf{Y}_{\Omega^*}$ denote the vector of observations at the design points in the prediction window. Also, let $\mathbf{X}$ denote the regression matrix consisting of all possible second-order terms.
2. Normalize and center the columns of $\mathbf{X}$.
3. Calculate $c = \mathbf{X}^\top \left(\mathbf{Y}_{\Omega^*} - \mathbf{X}_{\Omega^*} \left(\mathbf{X}_{\Omega^*}^\top \mathbf{X}_{\Omega^*} \right)^{-1} \mathbf{X}_{\Omega^*}^\top \mathbf{Y}_{\Omega^*} \right)$. Choose the ith term, say $x_j x_k$, such that $c_i = \min\{c\}$.
4. Add a column to $\mathbf{X}_{\Omega^*}$ corresponding to $x_j x_k$, and remove the corresponding column from $\mathbf{X}$. If $c_i \leq 0.2$ or maximum number of iterations is reached, STOP. Otherwise, go to Step 3.

5 PARAMETER ESTIMATION

As mentioned in Section 4, estimation of $\sigma^2(\mathbf{x}_0)$, and $\frac{\partial^2 y(\mathbf{x}_0)}{\partial x_1^2}, \ldots, \frac{\partial^2 y(\mathbf{x}_0)}{\partial x_d^2}$ is required to solve Problem 3. As is often done in Locally Weighted Least Squares Linear Regression, we use plug-in estimators for $\sigma^2(\mathbf{x}_0)$ and $\frac{\partial^2 y(\mathbf{x}_0)}{\partial x_1^2}, \ldots, \frac{\partial^2 y(\mathbf{x}_0)}{\partial x_d^2}$ (see, for example, Ruppert, Sheather, and Wand (1995)).

5.1 Variance Estimation

Having access to replications from the simulation makes it easy for us to get an estimate of the variance at each design point. We are interested in an estimate of the variance as it pertains to determining the size of the prediction window. We use a k-nearest neighbor estimate to get an estimate of the variance of a replication at $\mathbf{x}_0$ from the neighboring design points. The estimate of $\sigma^2(\mathbf{x}_0)$ is $S^2(\mathbf{x}_0) := \frac{1}{k} \sum_{\mathbf{x}_i \in \mathrm{I}_k(\mathbf{x}_0)} S^2(\mathbf{x}_i; R_i)$, where $\mathrm{I}_k(\mathbf{x}_0)$ is the set of the k nearest design points to $\mathbf{x}_0$. The choice of k is not critical and we use $k = 15$.

5.2 Second Derivative Estimation

To estimate the second partial derivatives we fit a third-order polynomial in a neighborhood of the prediction point and use the coefficients of the second-order terms as estimates of the second partial derivatives. In general, Ruppert and Wand (1994) suggest using an r-order polynomial to estimate partial derivatives of order m, where $r-m$ is an odd integer, with the most popular choice for r being $r = (m+1)$. A third-order polynomial with all interaction terms has $\binom{d}{3} + 1$ terms, which makes the regression problem infeasible in high dimensions. Thus, we do not include any interaction terms in the third-order polynomial and solve

$$\min_{\beta_0, \beta_1, \beta_2, \beta_3} \sum_{\mathbf{x}_i \in \mathrm{I}_{k^*}(\mathbf{x}_0)} \left(\bar{\mathrm{Y}}(\mathbf{x}_i; R_i) - \beta_0 - \beta_1^\top (\mathbf{x}_i - \mathbf{x}_0) - \beta_2^\top (\mathbf{x}_i - \mathbf{x}_0)^2 - \beta_3^\top (\mathbf{x}_i - \mathbf{x}_0)^3 \right)^2, \tag{4}$$

where $(\mathbf{x}_i - \mathbf{x}_0)^m := [(x_{i,1} - x_{0,1})^m, \ldots, (x_{i,d} - x_{0,d})^m]^\top$. We use $\hat{\beta}_2$ from the solution of Problem 4 as our estimate of the second partial derivatives. Specifically, $\hat{\beta}_{2,i}$ is our estimate of $\left(\partial^2 y(\mathbf{x}_0) / \partial x_i^2 \right)$. To find k^*, the optimal number of neighbors to be used in the estimation of the second partial derivatives, we use the **Nearest − Neighbor Procedure** in the appendix. This procedure searches for the optimal number of neighbors to use to fit the cubic polynomial by maximizing the goodness-of-fit criterion $R^2(k)$, which denotes the R^2 statistic using the k nearest neighbors, over k. This procedure is a variation of the procedure used in Ruppert, Sheather, and Wand (1995), which divides the design space into disjoint blocks and finds

the number of blocks that gives the best polynomial fit in each block. In high dimensions, dividing the design space into blocks is infeasible, which is why we search over the number of nearest neighbors instead.

6 EXPERIMENTS

We are mainly concerned with how the differentiability of the function, number of design points, variance of the function observations, and dimension affect the procedure. We use two queueing simulations, a multi-product M/G/1 queue and a multi-product Jackson network, whose simulation response surfaces are the expected number of products in the queue and expected cycle time of a product, respectively. The response surface for the multi-product M/G/1 queue is differentiable everywhere, while the response surface for the multi-product Jackson network is non-differentiable in some places.

The n design points we use in each experiment are the first n points from the Sobol Sequence (Sobol 1967). We fix the number of replications at each design point to 64 replications. For each replication, the simulation run-length is chosen to obtain constant relative standard deviation over the design space using a heavy-traffic approximation to the asymptotic variance presented in Whitt (1989). The relative standard deviation we use here is $\left(\sigma(\mathbf{x}_i)/\sqrt{N_i}\right)/|y(\mathbf{x}_i)|$, so, for example, a relative standard deviation of 0.25 means $\sigma(\mathbf{x}_i)/|y(\mathbf{x}_i)| = 2 = 0.25\sqrt{64}$. Using designs generated by the Sobol sequence and fixing the number of replications at each design point satisfies our assumption of a uniform limiting density of design points and simulation effort.

The prediction points $\mathbf{t}_1, \mathbf{t}_2, \ldots, \mathbf{t}_{150}$ are 150 points uniformly sampled from the unit hypercube, $[0,1]^d$, rescaled to fit inside the hypercube $[0.1, 0.9]^d$. We repeat the experiment 50 times to get 50 predictions at each prediction point. We evaluate the predictions using Root Average Relative Mean Squared Error

$$\text{RARMSE} = \sqrt{\frac{1}{7500}\sum_{j=1}^{50}\sum_{i=1}^{150}\left(\frac{\hat{y}_j(\mathbf{t}_i)}{y(\mathbf{t}_i)} - 1\right)^2},$$

where $\hat{y}_j(\mathbf{t}_i)$ is the estimated value of $y(\mathbf{t}_i)$ on the jth experiment at the ith prediction point.

6.1 Multi-Product M/G/1 Queue

In the multi-product M/G/1 queue, $d-1$ types of products arrive to a queue according to a Poisson Process. Let the service rate of product i be μ_i. The vector of design variables is $\mathbf{x} = (x_1, x_2, \ldots, x_{d-1}, \rho)$, where ρ is the traffic intensity and the x_i determine the arrival rates for the $d-1$ types of products. For $\mathbf{x} = (x_1, x_2, \ldots, x_{d-1}, \rho)$ the arrival rate for product i is $\lambda_i = cx_i$ where $c = \rho / \sum_{i=1}^{d-1} \frac{x_i}{\mu_i}$. The design space is $[5,10]^{d-1} \times [0.8, 0.95]$, which after rescaling is $[0,1]^d$. The system performance measure that we estimate with the simulation is the steady-state expected waiting time in the queue. The closed form solution for the steady-state expected waiting time used for evaluating the predictions is

$$y(\mathbf{x}) = \frac{\rho \sum_{i=1}^{d-1} \frac{\lambda_i}{\mu_i^2}}{(1-\rho)\sum_{i=1}^{d-1} \frac{\lambda_i}{\mu_i}}.$$

6.2 Multi-Product Jackson Network

In the multi-product Jackson Network, $d-1$ products arrive to the first station of a system of 3 single-server stations according to a Poisson Process. The service rate at station j is μ_j, which is independent of the product type. The vector of design variables is $\mathbf{x} = (x_1, x_2, \ldots, x_{d-1}, \rho)$, where ρ is the traffic intensity and the x_i determine the arrival rates for the $d-1$ types of products to the first station. For $\mathbf{x} = (x_1, x_2, \ldots, x_{d-1}, \rho)$ the arrival rate for product i is $\lambda_i = cx_i$ where $c = \max_j \rho / \sum_{i=1}^{d-1} \frac{x_i}{\mu_j}$. The design space is $[5,10]^{d-1} \times [0.8, 0.95]$,

Table 1: Relative difference (rel. diff.) of RARMSE and relative standard deviation (RSD) using estimated values of the second derivatives, and the corresponding runtime (in seconds).

M/G/1 Queue				
d	n	runtime	RSD	rel. diff.
5	500	1.4	0.05	-60%
			0.1	-62%
			0.25	-69%
20	5000	9.8	0.05	-48%
			0.1	-53%
			0.25	-61%
75	50000	45.2	0.05	-40%
			0.1	-41%
			0.25	-51%

Jackson Network				
d	n	runtime	RSD	rel. diff.
5	500	1.5	0.05	-67%
			0.1	-68%
			0.25	-70%
20	5000	9.8	0.05	-49%
			0.1	-58%
			0.25	-60%
75	50000	46.7	0.05	-42%
			0.1	-43%
			0.25	-47%

which after rescaling is $[0,1]^d$. We denote the expected number of visits to station j by product i by δ_{ij}. The system performance measure that we estimate with the simulation is the expected cycle time of product 1, which has the closed form solution,

$$y(\mathbf{x}) = \sum_{j=1}^{3} \frac{\delta_{1j}}{\mu_j - \sum_{k=1}^{d-1} \lambda_k \delta_{kj}}.$$

6.3 Experiment Results

Table 1 gives the relative difference of RARMSE and relative standard deviation. These values are calculated by subtracting the relative standard deviation used to choose the run length in the experiment from the RARMSE and standardizing by dividing the difference with the relative standard deviation. For example, if we used a relative standard deviation of 0.25, and obtained an RARMSE of 0.1 for that experiment, the value in the table would be $100\% \times (0.1 - 0.25)/0.25 = -60\%$.

From these tables, it is clear that our procedure is successful in filtering out the noise obtained from using noisy observations at the design points. Although the procedure can handle many more design points than the amount used to calculate the values in the tables, there was not much decrease in the RARMSE when more design points were used. This is a result of overestimating the second derivatives and is discussed in more detail in the next subsection.

The experiments were run in R using a 64-bit, quad-core processor with Windows 7. The average runtime for each dimension and corresponding number of design points is shown in Table 1. The majority of the time was spent on estimation of the second partial derivatives and sorting the data matrix in high dimensions.

6.3.1 Procedure using Actual Second Derivative Values

In each experiment, the estimated second partial derivatives were compared with the actual values. The estimates we obtained using the method in Section 5.2 were larger, which may explain the limited improvement in RARMSE as the number of design points increases. These larger estimates make our procedure choose smaller prediction windows than is actually optimal, hence limiting the smoothing capability of the procedure.

Table 2 shows the relative difference of RARMSE and relative standard deviation when we switch from using the method in Section 5.2 to estimate the second derivatives, to using the actual values of the second

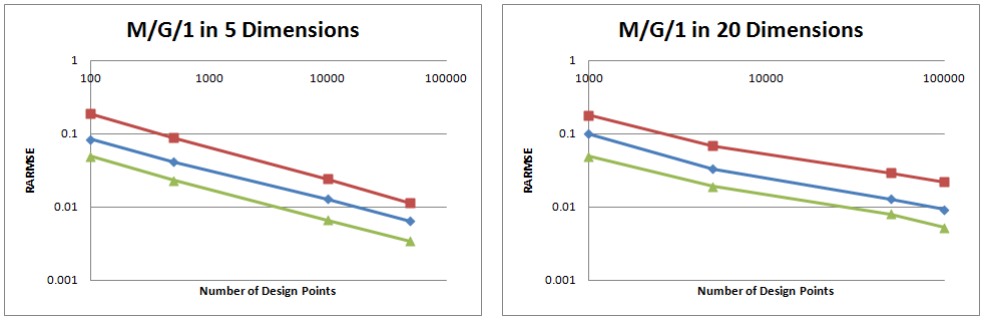
Table 2: Relative difference (rel. diff.) of RARMSE and relative standard deviation (RSD) when the actual second derivative values are used, and the relative difference for weighted least squares regression (WLS).

M/G/1 Queue, $d = 5$			
n	RSD	rel. diff.	WLS
500	0.05	-54%	76%
	0.1	-59%	-11%
	0.25	-65%	-62%
10000	0.05	-87%	78%
	0.1	-87%	-11%
	0.25	-90%	-64%
50000	0.05	-93%	78%
	0.1	-94%	-10%
	0.25	-96%	-64%

M/G/1 Queue, $d = 20$			
n	RSD	rel. diff.	WLS
5000	0.05	-62%	78%
	0.1	-67%	-11%
	0.25	-73%	-64%
50000	0.05	-84%	77%
	0.1	-87%	-11%
	0.25	-88%	-65%
100000	0.05	-90%	75%
	0.1	-91%	-12%
	0.25	-91%	-66%

derivatives. From these tables, we can see that our procedure works substantially better if we have more accurate estimates of the second partial derivatives. When the number of design points becomes large enough, the AMSE approximation becomes more valid, hence the larger decreases in the RARMSE in the tables. When the number of design points is too small, our procedure might not choose the optimal parameters for the weight function which explains the increase in RARMSE for fewer design points. The steep slopes of approximately $-1/2$ in Figure 2 exhibit the benefit obtained from adding more design points, and verify that the procedure is taking advantage of the large number of design points.

The last column in Table 2 displays the relative difference of RARMSE and relative standard deviation when weighted least squares regression (WLS) is used. From the table, we can see that a significant improvement over WLS is obtained when we localize the prediction using MLS. Although a large number of design points is needed in order for the bandwidths to remain local in higher dimensions, our method still produces results that are superior to WLS because MLS can assign different weight to each design point. Therefore, even though design points that fall in the prediction window may be 'far' away, they can still be assigned a very small weight. Since the performance of our procedure is improved significantly when



(a) (b)

Figure 2: In each log-log plot, the triangle, diamond, and square marked lines correspond to experiments where the simulation run-length was chosen to achieve relative standard deviations of 0.05, 0.1, and 0.25, respectively, and actual second derivative values were used.

we have better estimates of the second partial derivatives, future research needs to be done in this area. There are several other methods for estimating the second derivatives in Locally Weighted Least Squares Linear Regression for the one dimensional case, such as estimating the bias empirically, or using Principal

Component Analysis. However, these methods would have to be adapted to the higher dimensional cases we examine here.

ACKNOWLEDGEMENTS

This article is based upon work supported by the National Science Foundation under Grant No. CMMI-0900354.

APPENDIX

Bandwidth Procedure (Input: $\delta, \text{Mass}_{\text{UB}}$. Output: $h_1^{l\,*}, h_1^{r\,*}, \ldots, h_d^{l\,*}, h_d^{r\,*}$)
Step 1: Perform a line search over the interval $[\dim(\Pi_1^d)+\delta, \text{Mass}_{\text{UB}}]$, using the Golden Search Method (Bazaraa, Sherali, and Shetty 2006). For each $i \in [\dim(\Pi_1^d)+\delta, \text{Mass}_{\text{UB}}]$, the value $q(i)$ used in the line search is the optimal value of the optimization problem,

$$
\begin{aligned}
q(i) := \min_{\{h_1^l, h_1^r, \ldots, h_d^l, h_d^r\}} \quad & \frac{1}{4}\mu_2(K)^2 \left(\left(\frac{h_1^l + h_1^r}{2}\right)^2 \left|\frac{\partial^2 y(\mathbf{x}_0)}{\partial x_1^2}\right| + \cdots + \left(\frac{h_d^l + h_d^r}{2}\right)^2 \left|\frac{\partial^2 y(\mathbf{x}_0)}{\partial x_d^2}\right| \right)^2 \\
& + \frac{R(K)\sigma^2(\mathbf{x}_0)}{C_n \tilde{g}(\mathbf{x}_0) \left(\frac{h_1^l + h_1^r}{2}\right) \cdots \left(\frac{h_d^l + h_d^r}{2}\right)} \qquad (5) \\
\text{s.t.} \quad & n g(\mathbf{x}_0)(h_1^l + h_1^r) \cdots (h_d^l + h_d^r) = i \\
& 0 \le h_i^l \le x_{0,i}, \qquad \forall i = 1, 2, \ldots, d \\
& 0 \le h_i^r \le 1 - x_{0,i}, \qquad \forall i = 1, 2, \ldots, d.
\end{aligned}
$$

Optimization problem 5 can be solved using the **Inner Procedure** below, with $\Phi = i$. This procedure is based on a variation of the variable pegging procedure presented in Bitran and Hax (1981). Denote the optimal solution to the line search by i^* and let the corresponding optimal solution to the associated optimization problem be denoted by $h_1^{l\,*}, h_1^{r\,*}, \ldots, h_d^{l\,*}, h_d^{r\,*}$. This solution is optimal for optimization problem 3.

Inner Procedure (Input: Φ. Output: $h_1^{l\,*}, h_1^{r\,*}, \ldots, h_d^{l\,*}, h_d^{r\,*}$)
Step 0: Initialize $\mathbb{J}^1 = \{1, \ldots, d\}$, $\mathbf{P}^1 = \ln\left(\frac{\Phi}{n g(\mathbf{x}_0) 2^d}\right)$, and Iteration $\beta = 1$.
Step 1: For all $j \in \mathbb{J}^\beta$, set $h_j^\beta = \frac{1}{|\mathbb{J}^\beta|}\mathbf{P}^\beta - \frac{1}{2}\ln\left(\left|\frac{\partial^2 y(\mathbf{x}_0)}{\partial x_j^2}\right|\right) + \frac{1}{2|\mathbb{J}^\beta|}\sum_{k \in \mathbb{J}^\beta} \ln\left(\left|\frac{\partial^2 y(\mathbf{x}_0)}{\partial x_k^2}\right|\right)$.
If $h_j^\beta \le \ln\left(\frac{1}{2}\right)$ for all $j \in \mathbb{J}^\beta$, set $h_j^* = h_j^\beta$ and go to Step 3. Otherwise go to step 2.
Step 2: Let $\mathbb{J}_+^\beta = \left\{ j \in \mathbb{J}^\beta : h_j^\beta \ge \ln\left(\frac{1}{2}\right) \right\}$
Define $h_j^* = \ln\left(\frac{1}{2}\right), \forall j \in \mathbb{J}_+^\beta$ and let $\mathbb{J}^{\beta+1} = \mathbb{J}^\beta \setminus \mathbb{J}_+^\beta$, $\mathbf{P}^{\beta+1} = \mathbf{P}^\beta - |\mathbb{J}_+^\beta| \ln\left(\frac{1}{2}\right)$
If $\mathbb{J}^{\beta+1} = \varnothing$ go to Step 3. Else, $\beta \leftarrow \beta + 1$ and go to Step 1.
Step 3: Set $h_j^* \leftarrow e^{h_j^*}$. For each $i \in \{1, \ldots, d\}$, if $h_i^* \le \min(x_{0,i}, 1 - x_{0,i})$, set $h_i^{l\,*} = h_i^{r\,*} = h_i^*$. Else, if $x_{0,i} \le 1 - x_{0,i}$ set $h_i^{l\,*} = x_{0,i}$ and $h_i^{r\,*} = 2h_i^* - x_{0,i}$. Else, set $h_i^{r\,*} = 1 - x_{0,i}$ and $h_i^{l\,*} = 2h_i^* - 1 + x_{0,i}$.

Nearest – Neighbor Procedure
Step 1 Search over the grid $\left[7d, 8d, \ldots, \min\left\{25d, \left\lfloor \frac{n}{d} \right\rfloor\right\}\right]$. For each $k \in \left[7d, 8d, \ldots, \min\left\{25d, \left\lfloor \frac{n}{d} \right\rfloor\right\}\right]$, the value that is used in the line search is $R^2(k) := 1 - \frac{\mathbf{Y}_k^\top \left(\mathbf{I}_{k\times k} - \mathbf{X}_k (\mathbf{X}_k^\top \mathbf{X}_k)^{-1} \mathbf{X}_k^\top\right) \mathbf{Y}_k}{\mathbf{Y}_k^\top \left(\mathbf{I}_{k\times k} - \frac{1}{k}\mathbf{J}_{k\times k}\right) \mathbf{Y}_k}$, where $\mathbf{Y}_k$ and $\mathbf{X}_k$ is the vector of observations and the regression matrix of the k nearest neighbors to the prediction point, respectively, and $\mathbf{I}_{k\times k}$ is the $k \times k$ identity matrix, and $\mathbf{J}_{k\times k}$ is the $k \times k$ matrix of ones. Choose the k that maximizes $v(k)$.

REFERENCES

Ankenman, B., B. L. Nelson, and J. Staum. 2010, March. "Stochastic Kriging for Simulation Metamodeling". *Operations Research* 58 (2): 371–382.

Bazaraa, M., H. Sherali, and C. Shetty. 2006. *Nonlinear Programming: Theory and Algorithms*. Wiley Interscience.

Bitran, G. R., and A. C. Hax. 1981, April. "Disaggregation and Resource Allocation Using Convex Knapsack Problems with Bounded Variables". *Management Science* 27 (4): 431–441.

Bos, L., and K. Salkauskas. 1989. "Moving Least-Squares are Backus-Gilbert Optimal". *Journal of Approximation Theory* 59:267–275.

Haaland, B., and P. Z. G. Qian. 2011. "Accurate Emulators for Large-Scale Computer Experiments". *The Annals of Statistics* 39 (6): 2974–3002.

Lafferty, J., and L. Wasserman. 2008. "Rodeo: Sparse, Greedy Nonparametric Regression". *The Annals of Statistics* 36 (1): 28–63.

Lancaster, P., and K. Salkauskas. 1981, July. "Surfaces Generated by Moving Least Squares Methods". *Mathematics of Computation* 37 (155): 141–158.

Levin, D. 1998, October. "The Approximation Power of Moving Least Squares". *Mathematics of Computation* 67 (224): 1517–1531.

Ruppert, D., S. Sheather, and M. Wand. 1995, December. "An Effective Bandwidth Selector for Local Least Squares Regression". *Journal of the American Statistical Association* 90 (432): 1257–1270.

Ruppert, D., and M. Wand. 1994, September. "Multivariate Locally Weighted Least Squares Regression". *The Annals of Statistics* 22 (3): 1346–1370.

Shan, S., and G. Wang. 2010. "Metamodeling for High Dimensional Simulation-Based Design Problems". *Journal of Mechanical Design* 132 (5): 1–11.

Snelson, E., and Z. Ghahramani. 2006. "Sparse Gaussian Processes using Pseudo-inputs". In *Advances in Neural Information Processing Systems 18*: MIT Press.

Sobol, I. 1967. "The Distribution of Points in a Cube and the Accurate Evaluation of Integrals". *USSR Computational Mathematics and Mathematical Physics* 7:784–802.

Wand, M., and M. Jones. 1993. "Comparison of Smoothing Parameterizations in Bivariate Kernel Density Estimation". *Journal of the American Statistical Association* 88:520–528.

Whitt, W. 1989. "Planning Queueing Simulations". *Management Science* 35 (11): 1341–1366.

AUTHOR BIOGRAPHIES

PETER SALEMI is a Ph.D. candidate in the Department of Industrial Engineering and Management Sciences at Northwestern University. His research interests are in simulation metamodeling and simulation for financial engineering. His email address is petersalemi2014@u.northwestern.edu.

BARRY L. NELSON is the Walter P. Murphy Professor and Chair of the Department of Industrial Engineering and Management Sciences at Northwestern University, and a Fellow of INFORMS. His research centers on the design and analysis of computer simulation experiments on models of stochastic systems. His email and web addresses are nelsonb@northwestern.edu and www.iems.northwestern.edu/ ∼ nelsonb.

JEREMY STAUM is an Associate Professor of Industrial Engineering and Management Sciences at Northwestern University. He coordinated the Risk Analysis track of the 2007 and 2011 Winter Simulation Conferences and serves as department editor for financial engineering at *IIE Transactions* and as an associate editor at *Management Science*. His website is users.iems.northwestern.edu/∼ staum.