Proceedings of the 2012 Winter Simulation Conference
S. Jain, R. R. Creasey, J. Himmelspach, K. P. White, and M. Fu, 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 tandem 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 any point 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 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. The only metamodeling approach we are aware of that can deal with large numbers of design points in high dimensional problems uses a multi-stage interpolation technique (Haaland and Qian 2011). However, this multi-stage interpolation technique does not yet have a practical method for implementation.

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. For further reference on MLS regression in image processing, see for example Lipman, Cohen-Or, and Levin (2007). 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 vector of design variables, x, by $y(x)$. In the case of a queue, 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 $\mathbb{X}_n = (x_1^n, x_2^n, \ldots, x_n^n)$ 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 x_i^n. We assume that

$$\lim_{n\to\infty} \frac{1}{n} \sum_{i=1}^{n} \mathbb{I}\{x_i^n \in \mathrm{A}\} = \int_{\mathrm{A}} g(z)dx, \tag{1}$$

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

$$\lim_{n\to\infty} \frac{1}{C_n} \sum_{i=1}^{n} \mathbb{I}\{x_i^n \in \mathrm{A}\} R_i^n = \int_{\mathrm{A}} \tilde{g}(z)dx, \tag{2}$$

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 $\tilde{g}(z)$ is the limiting density of effort spent at $z \in \mathbb{X}$. In our procedure, we assume that g and $\tilde{g}$ are uniform densities on the unit hypercube $[0,1]^d$, i.e. $g(\cdot) = 1, \tilde{g}(\cdot) = 1$ and $\mathbb{X} = [0,1]^d$.

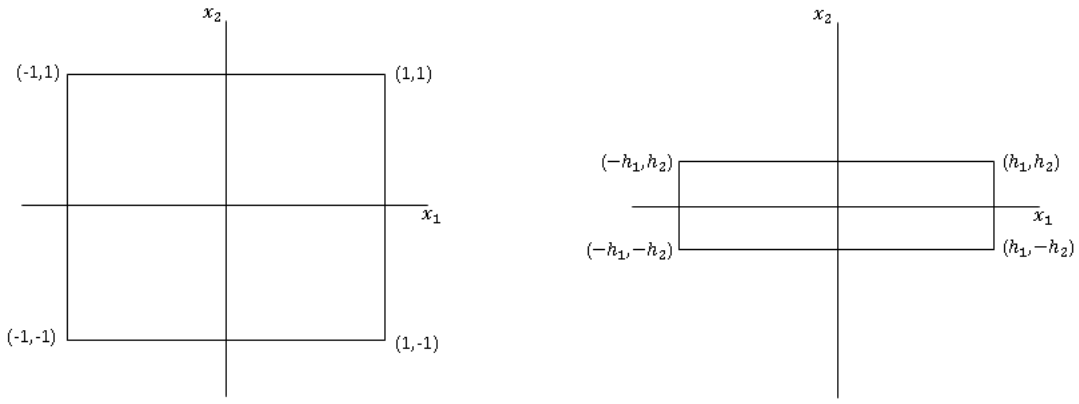
For the nth sequential design, we run the simulation at $\mathbb{X}_n = (x_1^n, x_2^n, \ldots, x_n^n)$. At design point x_i^n we run R_i^n replications of the simulation, and we denote the system performance measure for the jth replication by $\mathrm{Y}_j^n(x_i^n)$. The estimate that we obtain at design point x_i^n is the sample average $\bar{\mathrm{Y}}^n(x_i^n; R_i^n) = (1/R_i^n) \sum_{j=1}^{R_i^n} \mathrm{Y}_j^n(x_i^n)$. We will also need an estimate of the variance $\sigma^2(x_i^n)$ of a replication at x_i^n which we estimate by the sample variance $S^2(x_i^n; R_i^n) = (1/(R_i^n - 1)) \sum_{j=1}^{R_i^n} (\mathrm{Y}_j^n(x_i^n) - \bar{\mathrm{Y}}^n(x_i^n; R_i^n))^2$.

For ease of notation, we drop the superscript notation for the nth sequential design and let $x_1, x_2, \ldots, x_n$ 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 $K_H(u) = |H|^{-1/2} K(H^{-1/2}u)$, where K is a compactly supported d-variate kernel such that $\int K(u)du = 1$, and H is a $d \times d$ symmetric positive definite matrix depending on n. The matrix 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_H. The number of non-zero entries in the bandwidth matrix represents the number of bandwidth parameters that



(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 $K_H(\cdot)$ in two dimensions, with $h_1 > h_2$.

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, H (Wand and Jones 1993). We introduce a kernel which is a function of the maximum norm, given by $||u||_\infty = \max(|u_1|, |u_2|, \ldots, |u_d|)$ for $u \in \mathbb{R}^d$. This kernel is

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

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 $H = \mathrm{diag}(h_1^2, \ldots, h_d^2)$ will yield the weight function

$$K_H(u) = \frac{1}{h_1 \cdots h_d} \left[1 - \max\left(\left|\frac{u_1}{h_1}\right|, \ldots, \left|\frac{u_d}{h_d}\right|\right)\right] \mathbb{I}\left\{\max\left(\left|\frac{u_1}{h_1}\right|, \ldots, \left|\frac{u_d}{h_d}\right|\right) \leq 1\right\}, \tag{3}$$

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 $y(x)$ for any specific $x \in \mathbb{X}$ 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, $K_H(\cdot)$. 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

of Π_k^d which is the set of $\binom{d+k}{k}$ monomials. The polynomial, $\hat{y}_{x_0}^{\text{MLS}}$, used for approximating the function value $y(x_0)$ at the prediction point x_0 is

$$\hat{y}_{x_0}^{\text{MLS}} = \arg\min_{p \in \Pi_k^d} \left\{ \sum_{i=1}^{n} (\bar{\Upsilon}(x_i;R_i) - p(x_i))^2 K_H(x_i - x_0) \right\}. \tag{4}$$

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: $\hat{y}_{x_0}^{\text{MLS}}(x) = \mathbf{P}(x)^\top \left(\mathbf{P}^\top \mathbf{W}(x_0)\mathbf{P}\right)^{-1} \mathbf{P}^\top \mathbf{W}(x_0)\mathbf{Y}$, where $\mathbf{P}(x) = [p_1(x), p_2(x), \ldots, p_m(x)]^\top$, $\mathbf{W}(x_0) = \text{diag}\{K_H(x_1 - x_0), K_H(x_2 - x_0) \ldots, K_H(x_n - x_0)\}$, $\mathbf{P}$ is the $n \times m$ matrix whose i^{th} row is $[p_1(x_i), p_2(x_i), \cdots p_m(x_i)]$, and $\mathbf{Y} = \left(\bar{\Upsilon}(x_1;R_1), \bar{\Upsilon}(x_2;R_2), \ldots, \bar{\Upsilon}(x_n;R_n)\right)^\top$. For each prediction point, $x_0 \in \mathbb{X}$, we get a different approximating polynomial, $\hat{y}_{x_0}^{\text{MLS}}$.

The minimization in Problem (4) is done over the polynomial space Π_k^d, 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_k^d is $\binom{d+k}{k}$, so for large d we must be careful to not pick k too large. Otherwise, we must solve $\binom{d+k}{k}$ equations in n unknowns, which is infeasible when d and k are large. We will use the space of linear polynomials, Π_1^d. 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_1^d 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 $\bar{\Upsilon}(x_i;R_i) = y(x_i) + \frac{\sigma(x_i)}{\sqrt{R_i}}\varepsilon_i$, where $\sigma(x_i)$ is the standard deviation of a replication at 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 x_0 is $\hat{y}^{\text{LOC}}(x_0) := \hat{\beta}_0$, where $\hat{\beta}_0$ is from the solution to the following problem:

$$\min_{\beta_0, \beta_1} \sum_{i=1}^{n} \left\{ \bar{\Upsilon}(x_i;R_i) - \beta_0 - \beta_1^\top (x_i - x_0) \right\}^2 K_H(x_i - x_0). \tag{5}$$

For the diagonal bandwidth matrix, $H = \text{diag}(h_1^2, \ldots, h_d^2)$, the AMSE of the estimator $\hat{\beta}_0$ of the function value at x_0 for large n is approximately

$$\begin{aligned} AMSE &\approx \frac{1}{4}\mu_2(K)^2 \text{tr}(H\nabla_y^2(x_0))^2 + \frac{R(K)\sigma^2(x_0)}{C_n|H|^{1/2}\tilde{g}(x_0)} \\ &= \frac{1}{4}\mu_2(K)^2 \left(h_1^2 \frac{\partial^2 y(x_0)}{\partial x_1^2} + \cdots + h_d^2 \frac{\partial^2 y(x_0)}{\partial x_d^2} \right)^2 + \frac{R(K)\sigma^2(x_0)}{C_n \tilde{g}(x_0) h_1 \cdots h_d}, \end{aligned} \tag{6}$$

where $\mu_2(K) = \int_{\mathbb{R}} x_i K(x)\,dx$, $R(K) = \int_{\mathbb{R}} K(x)^2\,dx$ and $\sigma^2(x_0)$ is the variance of a replication at the prediction point. We do not assume anything about the smoothness of $\sigma^2(\cdot)$, and estimate it using the sample variances at the design points using the method discussed in Section 5.2.

Equation (6) shows the bias-variance trade-off with respect to the bandwidth parameters, $h_1,\ldots,h_d$. 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 (6), 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 (6), a higher variance at the prediction point, $\sigma^2(x_0)$, 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, $\tilde{g}(x_0)$, 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 for High Dimensional Problems

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 $\hat{K}_H(\cdot)$, which is a slight modification of the weight function $K_H(\cdot)$, given by

$$\hat{K}_H(u) = \frac{1}{h_1\cdots h_d}\left[1-\max\left(\left|\frac{u_1}{h_1}\right|,\ldots,\left|\frac{u_d}{h_d}\right|\right)\right]\mathbb{I}\left\{\max\left(\left|\frac{u_1}{\hat{h}_1}\right|,\ldots,\left|\frac{u_d}{\hat{h}_d}\right|\right)\leq 1\right\}, \tag{7}$$

where we distinguish between the two different vectors of parameters. The entries of the parameter vector $\mathbf{h} = (h_1,h_2,\ldots,h_d)^\top$ are called the bandwidth parameters, and the entries of the parameter vector $\hat{\mathbf{h}} = (\hat{h}_1,\hat{h}_2,\ldots,\hat{h}_d)^\top$ are called the window parameters. The bandwidth parameters determine the bandwidth in the corresponding coordinate direction. For example, h_1 determines how fast the weight decays in the direction along the first basis vector of $\mathbb{R}^d$. The window parameters determine the compact support of the kernel $\hat{K}_H(\cdot)$. The compact support of the kernel $\hat{K}_H(\cdot)$ is called the prediction window since the design points that lie in the compact support of the kernel will be the design points used for prediction. We denote the prediction window by $\hat{\Omega} := \left\{x\in\mathbb{X} : |x_1-x_0|\leq \hat{h}_1,\ldots,|x_d-x_0|\leq \hat{h}_d\right\}$. The set $\Omega := \{x\in\mathbb{X} : |x_1-x_0|\leq h_1,\ldots,|x_d-x_0|\leq h_d\}$ would be the prediction window if we used the original weight function K_H defined in Equation (3).

To find the optimal bandwidth parameters in $\mathbf{h}$ we solve Problem (8), whose objective function is a slight modification of the AMSE equation. Estimation of $\sigma^2(x_0)$, and $\frac{\partial^2 y(x_0)}{\partial x_1^2},\ldots,\frac{\partial^2 y(x_0)}{\partial x_d^2}$ is discussed in Section 5.

$$\min_{\{h_1,\ldots,h_d\}} \quad \frac{1}{4}\mu_2(K)^2\left(h_1^2\left|\frac{\partial^2 y(x_0)}{\partial x_1^2}\right|+\cdots+h_d^2\left|\frac{\partial^2 y(x_0)}{\partial x_d^2}\right|\right)^2+\frac{R(K)\sigma^2(x_0)}{C_n\tilde{g}(x_0)h_1h_2\cdots h_d} \tag{8}$$

$$\text{s.t.} \quad \dim(\Pi_1^d)+\delta \leq ng(x_0)2^d h_1\cdots h_d \leq \text{Mass}_{\text{UB}} \tag{8.a}$$

$$0\leq h_i\leq \frac{1}{2}, \qquad \forall i=1,2,\ldots,d. \tag{8.b}$$

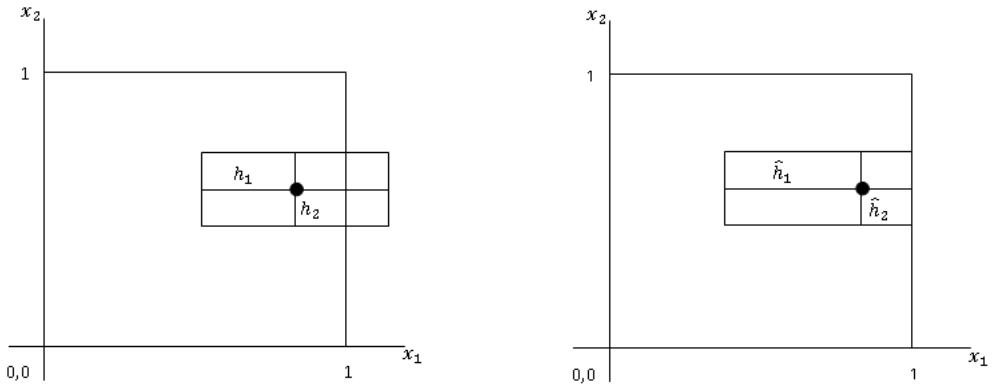


Figure 2: When a prediction point is near the boundary of the unit hypercube, part of the prediction window may lie outside of the unit hypercube, as shown in (a). The result of the Window Procedure is the prediction window shown in (b). The shifted prediction window now falls within the unit hypercube.

The motivation for constraint (8.a) is the following. Without an upper bound on the amount of design points that is 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 $\text{Mass}_{\text{UB}} = 1000$ 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 (4), which we do by adding the constant δ to the dimension of Π_k^d in the lower bound. We use $\delta = d$. An approximation to the number of design points that lie within the prediction window is $ng(x_0)2^d\hat{h}_1\hat{h}_2\cdots\hat{h}_d$. 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 amount of design points included in the prediction window. From the way that we transform the optimal bandwidth parameters to get the window parameters, the volume of the prediction window $2^d\hat{h}_1\hat{h}_2\cdots\hat{h}_d$ is equal to $2^d h_1 h_2\cdots h_d$, so $ng(x_0)2^d\hat{h}_1\hat{h}_2\cdots\hat{h}_d = ng(x_0)2^d h_1 h_2\cdots h_d$. The limiting density of design points that makes $ng(x_0)2^d h_1 h_2\cdots h_d$ 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 (6) 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 (8) can be solved using the **Bandwidth Procedure** in the appendix. Denote the optimal solution by $\mathbf{h}^* = \{h_1^*, \ldots, h_d^*\}$.

The window parameters in $\hat{\mathbf{h}}$ are obtained from a transformation of the optimal values of the bandwidth parameters in $\mathbf{h}$ using the **Window Procedure** in the appendix. To see why this transformation is necessary, and hence why we need two different sets of parameters for our modified weight function, consider the following. Based on the original weight function K_H defined in Equation (3), the bandwidth

parameters would also determine the prediction window, Ω. For prediction points near the boundary of the unit hypercube, this may cause part of the prediction window to lie outside of the unit hypercube. Since the limiting density of effort is the uniform density and the design space is $\mathbb{X} = [0,1]^d$, we need $2^d h_1 h_2 \cdots h_d$ to be the volume of the intersection of the prediction window and the unit hypercube. The transformation will transform the prediction window, Ω, for the original weight function into the prediction window, $\hat{\Omega} \subseteq [0,1]^d$, for the modified weight function keeping the volume of the two windows the same, in the manner presented in Figure 2. The modified weight function, $\hat{K}_H(\cdot)$, converges to the original weight function, $K_H(\cdot)$, as $n \to \infty$ because this increase in design points makes the variance term in Equation (6) negligible and hence shrinks the prediction window, making the transformation unnecessary. Thus, the use of our modified weight function is justified asymptotically.

Let $\hat{\mathbf{h}}^*$ denote the values of the window parameters obtained from the **Window Procedure**. Once we have the optimal values for the bandwidth parameters $\mathbf{h}^*$ and the values for the window parameters $\hat{\mathbf{h}}^*$, we obtain the weight function used for prediction

$$K_{H^*}(u) = \frac{1}{h_1^* \cdots h_d^*} \left[1 - \max\left(\left|\frac{u_1}{h_1^*}\right|, \ldots, \left|\frac{u_d}{h_d^*}\right|\right)\right] \mathbb{I}\left\{\max\left(\left|\frac{u_1}{\hat{h}_1^*}\right|, \ldots, \left|\frac{u_d}{\hat{h}_d^*}\right|\right) \leq 1\right\}.$$

5 Parameter Estimation

As mentioned in Section 4, estimation of $\sigma^2(x_0)$, and $\frac{\partial^2 y(x_0)}{\partial x_1^2}, \ldots, \frac{\partial^2 y(x_0)}{\partial x_d^2}$ is required to solve Problem (8). As is often done in Locally Weighted Least Squares Linear Regression, we use plug-in estimators for $\sigma^2(x_0)$ and $\frac{\partial^2 y(x_0)}{\partial x_1^2}, \ldots, \frac{\partial^2 y(x_0)}{\partial x_d^2}$ (see, for example, Ruppert, Sheather, and Wand (1995)).

5.1 Variance Estimation

Having access to replications from the simulation makes it easier 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 x_0 from the neighboring design points. The estimate of $\sigma^2(x_0)$ is

$$S^2(x_0) := \frac{1}{k} \sum_{x_i \in \mathrm{I}_k(x_0)} S^2(x_i; R_i).$$

where $\mathrm{I}_k(x_0)$ is the set of the k nearest neighbors to x_0. The choice of k is not critical and we use $k = 50$.

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

$$\min_{\beta_0, \boldsymbol{\beta_1}, \boldsymbol{\beta_2}, \boldsymbol{\beta_3}} \sum_{x_i \in B(x_0, k^*)} \left[\bar{\mathrm{Y}}(x_i; R_i) - \beta_0 - \boldsymbol{\beta_1}^\top (x_i - x_0) - \boldsymbol{\beta_2}^\top (x_i - x_0)^2 - \boldsymbol{\beta_3}^\top (x_i - x_0)^3\right]^2. \tag{9}$$

We use $\hat{\boldsymbol{\beta}}_{\mathbf{2}}$ from the solution of (9) as our estimate of the second partial derivatives. Specifically, $\hat{\beta}_{2,i}$ is our estimate of $\left(\partial^2 y(x_0)/\partial x_i^2\right)$. 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 based on a signal-to-noise ratio criterion. This procedure searches for the optimal number of neighbors to use to fit the cubic polynomial by maximizing the ratio of $\sum_{i=1}^{d} \hat{\beta}_{\mathbf{2},\mathbf{i}}^{k}$ to its corresponding estimated standard deviation. This procedure is a variation of the procedure used in Ruppert, Sheather, and Wand (1995), which divides the design space into blocks and finds the number of blocks that gives the best polynomial fit. In high dimensions, dividing the design space into blocks is infeasible, which is why we search over the number of nearest neighbors instead. Intuitively, we are searching for the number of neighbors that gives the second-order with the least standard deviation relative to their magnitude.

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, both of which have closed form solutions for the system performance measures. 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.

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 Monte Carlo standard deviation over the design space using a heavy-traffic approximation to the asymptotic variance presented in Whitt (1989). The relative Monte Carlo standard deviation we use here is $\left(\sigma(x_i)/\sqrt{N_i}\right)/y(x_i)$, so, for example, a relative Monte Carlo standard deviation of 0.25 means $\sigma(x_i)/y(x_i) = 2 = 0.25\sqrt{64}$. 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 $t_1, t_2, \ldots, t_{150}$ are the first 150 Halton points (Halton 1964) rescaled to fit inside the hypercube $[0.1, 0.9]^d$. At each prediction point, we repeat the experiment 50 times to get 50 predictions. We evaluate the predictions using Root Average Relative Mean Squared Error

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

where $\hat{y}_j(t_i)$ is the estimated value of $y(t_i)$ 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 $x = (x_1, x_2, \ldots, x_{d-1}, \rho)$, where ρ is the traffic intensity and the x_i determine the arrival rates for the $d-1$ types of products. For $x = (x_1, x_2, \ldots, x_{d-1}, \rho)$ 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 we can assume the design space 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

$$\mathrm{E}[W_Q] = \frac{\rho \sum_{i=1}^{d-1} \frac{\lambda_i}{\mu_i^2}}{(1-\rho)\sum_{i=1}^{d-1} \frac{\lambda_i}{\mu_i}}.$$

To obtain the heavy-traffic approximation to the asymptotic variance of the steady-state expected cycle time for the M/G/1 queue, see Section 3 of Whitt (1989).

6.2 Multi-Product Jackson Network

In the multi-product Jackson Network, $d-1$ products arrive to the first station of a tandem 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 $x = (x_1, x_2, \ldots, x_{d-1}, \rho)$, 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 $x = (x_1, x_2, \ldots, x_{d-1}, \rho)$ 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 $[5,10]^{d-1} \times [0.8, 0.95]$, which after rescaling we can assume the design space 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 (Yang, Liu, Tongarlak, Ankenman, and Nelson 2007),

$$\mathrm{E}[CT_1] = \sum_{j=1}^{3} \frac{\delta_{1j}}{\mu_j - \sum_{k=1}^{d-1} \lambda_k \delta_{kj}}.$$

To obtain the heavy-traffic approximation to the asymptotic variance of the steady-state expected cycle time for the Jackson Network, see Section 5.5 of Whitt (1989).

6.3 Experiment Results

The tables in Figure 3 give the relative difference of RARMSE and relative Monte Carlo standard deviation. These values are calculated by subtracting the relative Monte Carlo standard deviation used to choose the run length in the experiment from the RARMSE and standardizing by dividing the difference with the relative Monte Carlo standard deviation. For example, if we used a relative Monte Carlo 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 a significant decrease in the RARMSE when more design points were used. This 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 time to get a prediction was 2.6 seconds, 14.9 seconds, and 1.06 minutes for 5, 20, and 75 dimensions, respectively. The majority of the time was spent on estimation of the second partial derivatives and sorting the data matrix in high dimensions.

M/G/1 Queue

Relative Monte Carlo Standard Deviation

	0.05	0.1	0.25
d=5, n=500	-59%	-63%	-67%
d=20, n=5000	-47%	-53%	-59%
d=75, n=50000	-41%	-41%	-49%

(a)

Jackson Network

Relative Monte Carlo Standard Deviation

	0.05	0.1	0.25
d=5, n=500	-67%	-66%	-69%
d=20, n=5000	-50%	-59%	-59%
d=75, n=50000	-43%	-44%	-46%

(b)

Figure 3: Relative difference of RARMSE and relative Monte Carlo standard deviation.

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, giving an explanation for 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.

The tables in Figure 4 show the decrease in RARMSE when we switch from using the method in Section 5.2 to estimate the second derivatives, to using the actual values of the second 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 less design points. The steep slopes of approximately $-1/2$ in Figure 5 exhibit the benefit obtained from adding more design points, and verify that the procedure is taking advantage of the large amounts of design points. Since the performance of

M/G/1 in 5 Dimensions

Relative Monte Carlo Standard Deviation

	0.05	0.1	0.25
100	18%	32%	74%
500	-12%	-18%	-23%
10000	-59%	-59%	-63%
50000	-74%	-76%	-79%

(a)

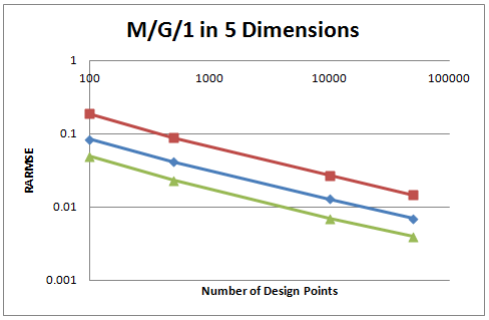
M/G/1 in 20 Dimensions

Relative Monte Carlo Standard Deviation

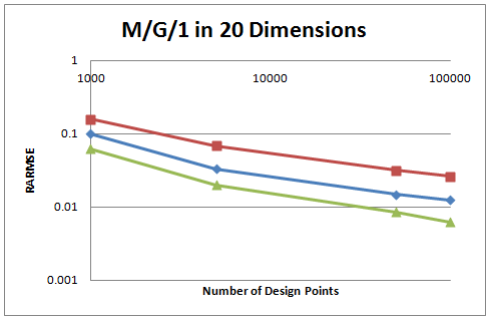
	0.05	0.1	0.25
1000	73%	66%	31%
5000	-25%	-30%	-34%
50000	-51%	-57%	-62%
100000	-61%	-62%	-66%

(b)

Figure 4: The tables in (a) and (b) show the relative change in the RARMSE when the actual second derivative values are used in the procedure for predicting the expected steady-state waiting time in the M/G/1 queue.



(a)



(b)

Figure 5: In each plot, the triangle, diamond, and square marked lines correspond to experiments where the simulation run-length was chosen to achieve relative Monte Carlo standard deviations of 0.05, 0.1, and 0.25, respectively, and actual second derivative values were used.

our procedure is increased significantly when 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.

APPENDIX

Bandwidth Procedure (Input: $\delta, \text{Mass}_{\text{UB}}, \sigma^2(x_0), g(x_0), \tilde{g}(x_0), \frac{\partial^2 y(x_0)}{\partial x_1^2}, \ldots, \frac{\partial^2 y(x_0)}{\partial x_d^2}$. Output: $h_1^*, \ldots, h_d^*$)

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,

$$q(i) := \min_{\{h_1,\ldots,h_d\}} \quad \frac{1}{4}\mu_2(K)^2 \left(h_1^2 \left| \frac{\partial^2 y(x_0)}{\partial x_1^2} \right| + \cdots + h_d^2 \left| \frac{\partial^2 y(x_0)}{\partial x_d^2} \right| \right)^2 + \frac{R(K)\sigma^2(x_0)}{C_n \tilde{g}(x_0) h_1 \cdots h_d} \tag{10}$$

$$\text{s.t.} \quad n g(x_0) 2^d h_1 \cdots h_d = i \tag{10.a}$$

$$0 \le h_i \le \frac{1}{2}, \qquad \forall i = 1, 2, \ldots, d. \tag{10.b}$$

Optimization problem (10) 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 $h_1^*, h_2^*, \ldots, h_d^*$. The solution $h_1^*, h_2^*, \ldots, h_d^*$ is optimal for optimization problem (8).

Inner Procedure (Input: Φ. Output: $h_1^*, \ldots, h_d^*$)

Step 0: Initialize $\mathbb{J}^1 = \{1, \ldots, d\}$, $\mathbf{P}^1 = \ln\left(\frac{\Phi}{n g(x_0) 2^d}\right)$, and Iteration $\beta = 1$.

Step 1: For all $j \in \mathbb{J}^\beta$, set $h_j^\beta = \frac{1}{|\mathbb{J}^\beta|}\mathbf{P}^\beta - \frac{1}{2}\ln\left(\left|\frac{\partial^2 y(x_0)}{\partial x_j^2}\right|\right) + \frac{1}{2|\mathbb{J}^\beta|}\sum_{k\in\mathbb{J}^\beta} \ln\left(\left|\frac{\partial^2 y(x_0)}{\partial x_k^2}\right|\right)$.

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 $h_j^* \leftarrow e^{h_j^*}$. The solution $h_j^*, \forall j \in \{1, \ldots, d\}$ is optimal. Return $h_1^*, \ldots, h_d^*$

Window Procedure (Input: $h_1^*, \ldots, h_d^*$. Output: $\hat{h}_1^*, \ldots, \hat{h}_d^*$)

Step 0: For each $i \in \{1, \ldots, d\}$, if $h_i^* \le \min(x_{0,i}, 1 - x_{0,i})$, set $\hat{h}_i^* = h_i^*$. Else, set $\hat{h}_i^* = (2h_i^* - \min(x_{0,i}, 1 - x_{0,i}))$.

Nearest – Neighbor Procedure

Step 1 Search over the grid $\left[5d, 6d, \ldots, \min\left\{250d, \left\lfloor \frac{n}{d} \right\rfloor\right\}\right]$. For each $k \in \left[5d, 6d, \ldots, \min\left\{250d, \left\lfloor \frac{n}{d} \right\rfloor\right\}\right]$, the value $v(k)$ used in the search is

$$v(k) = \frac{\sum_{i=1}^d \hat{\beta}_{2,i}^k}{\widehat{\text{var}}(\sum_{i=1}^d \hat{\beta}_{2,i}^k)} = \frac{\sum_{i=1}^d \hat{\beta}_{2,i}^k}{\sum_{i=1}^d \widehat{\text{var}}(\hat{\beta}_{2,i}^k) + 2\sum_{i<j} \widehat{\text{cov}}(\hat{\beta}_{2,i}^k, \hat{\beta}_{2,j}^k)},$$

where $\hat{\beta}_2^k$ is from the solution of Problem (9) using the k nearest neighbors, and each of the $\widehat{\text{var}}(\hat{\beta}_{2,i}^k)$ and $\widehat{\text{cov}}(\hat{\beta}_{2,i}^k, \hat{\beta}_{2,j}^k)$ are estimated using the standard estimates of these values from the regression. 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". Preprint submitted to the *Annals of Statistics*. Revised May 23, 2011.

Halton, J. 1964. "Algorithm 247: Radical-inverse Quasi-random Point Sequence". *Communications of the Association for Computing Machinery*.

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

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

Lipman, Y., D. Cohen-Or, and D. Levin. 2007. "Data-Dependent MLS for Faithful Surface Approximation". *Eurographics Symposium on Geometry Processing*.

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.

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).

Yang, F., J. Liu, M. Tongarlak, B. E. Ankenman, and B. L. Nelson. 2007. "Metamodeling for Cycle Time-throughput-product Mix Surfaces using Progressive Model Fitting". In *Proceedings of the 2007 Winter Simulation Conference*.

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 ⟨*petersalemi*2014@*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*⟩.