Proceedings of the 2015 Winter Simulation Conference
L. Yilmaz, W. K V. Chan, I. Moon, T. M. K. Roeder, C. Macal, and M. D. Rossetti, eds.

Database Monte Carlo for Simulation on Demand

Imry Rosenbaum
Jeremy Staum

Department of Industrial Engineering and Management Sciences
McCormick School of Engineering
Northwestern University
2145 Sheridan Road
Evanston, IL 60208-3119, USA

ABSTRACT

In the simulation-on-demand paradigm, we invest computational effort by running a simulation experiment before a question is asked, and then we quickly provide an answer by making use of the results of the earlier simulation experiment. This can be done by building a metamodel, but standard metamodeling methods used in stochastic simulation have the disadvantage that they require validation. We show how to use Database Monte Carlo with control variates to provide simulation-on-demand without metamodel validation.

1 INTRODUCTION

In a panel discussion on the grand challenges facing modeling and simulation (Taylor et al. 2012), Ernie Page addressed simulation on demand:

> Many of today's analytical simulations have very long shelf lives—they get applied to dozens, even hundreds of analytical studies. Many of these simulations also take a very long time to run. ...The availability of cheap, elastic computing in the cloud gives us the opportunity to run our long-running simulations continuously as "background" processes. These runs could serve to both conduct detailed exploration of the model's response surface, and to generate metamodels, which could be used to shorten the analysis cycle when a new question is asked of the model.

The response surface is the function that maps the simulation model's input to the mean simulation output. A metamodel is an approximation to this function. When a question is asked of the model, we want to know the value of the response surface for a particular input to which the question refers. The motivation to achieve simulation on demand is that we do not know exactly what questions will be asked of the model in the future, but once a question is asked, the amount of time required to run the model to obtain a sufficiently precise answer may be excessive. In the simulation-on-demand paradigm, we invest computational effort by running a simulation experiment before a question is asked. Based on this simulation experiment, we can create a metamodel, and then use the metamodel to answer a question quickly once it is asked. Applications of simulation on demand range from production planning (Ankenman et al. 2011) to financial engineering (Liu, Nelson, and Staum 2010).

Metamodel validation is a major issue in developing simulation-on-demand procedures based on metamodeling. For metamodeling and metamodel validation in stochastic simulation, see Kleijnen (2007).

The standard metamodeling methods used in stochastic simulation are function approximation methods that involve regression, smoothing, or interpolation. The metamodels resulting from these methods require validation before they can be used. Otherwise, the answer from the metamodel may have excessive bias or variance as an estimator of the true answer, based on the value of the response surface. If the metamodel is invalid, it can be improved by increasing the computational effort of the simulation experiment: running the simulation for more values of its inputs, using more replications, etc. The need to achieve a valid metamodel creates costs and potential problems for simulation-on-demand via function approximation methods:

1. If a limited computational budget is available for the simulation experiment to build a metamodel, it may not be possible to obtain a valid metamodel.
2. The metamodel validation techniques do not actually yield guarantees that the metamodel's error is sufficiently small for every possible input that may be the subject of a question in the future. This creates a risk of inaccuracy in the answers obtained from a metamodel, even if it passed a validation test.
3. Validation costs analyst time for assessing validity and planning a follow-up experiment, unless one possesses automated procedures for running simulation experiments until metamodels are valid (Liu, Nelson, and Staum 2010).
4. The computational cost required to obtain a valid metamodel is often quite high, in our experience. For example, in the simulation experiment reported in Section 5, the method we proposed worked well, but metamodeling did not reliably generate valid metamodels given the same computational budget.

All these costs and problems of metamodel validation motivated us to develop a simulation-on-demand method that does not require metamodel validation.

The importance of validation in metamodeling stems from the bias that is typically created by the function approximation methods that are employed. This bias is difficult to quantify rigorously, whereas in many Monte Carlo methods, it is easy to quantify variance adequately using elementary techniques. We avoid metamodel validation by dispensing with metamodeling, instead using a variance reduction technique so that we can avoid introducing bias. Once a question is asked in the simulation-on-demand paradigm, the question reveals that we want to know the value of the response surface at a particular input called the "prediction point." Our simulation-on-demand method runs a simulation at the prediction point. The difficulty is that, in the simulation-on-demand paradigm, the time available for this simulation is so short that it yields an unacceptably high variance. To reduce this variance to an acceptable level, we use a variance reduction technique that employs the results of the initial simulation experiment, which was performed before the question was asked.

The variance reduction technique that we use is Database Monte Carlo with control variates (Borogovac and Vakili 2008). In this technique, simulation output from the initial experiment is used like a control variate for simulation output at the prediction point. The difficulty is not knowing the mean of the simulation output that is being used like a control variate. However, this difficulty is circumvented by treating it as a quasi control variate (Emsermann and Simon 2002) or control variate with estimated mean (Pasupathy et al. 2012). Thus, Database Monte Carlo serves as a way to invest computational effort before the prediction point is revealed and make it pay dividends in the form of reduced variance for the simulation at the prediction point.

2 MATHEMATICAL FRAMEWORK

This section specifies a mathematical framework for the simulation-on-demand problem. We describe simulation output as a function of model inputs using the concept of random fields. We also discuss figures of merit to use for comparing simulation experiment designs and analysis methods.

When a simulation is run with input θ, the simulation output is regarded as a random variable: denote it $Y(\theta)$. The simulation output Y can be described as a random field (Staum 2009). That is, the simulation output is a measurable function $Y : \Theta \times \Omega \to \mathbb{R}$, where Θ is a domain in which the input θ is chosen, and Ω is a probability space with probability measure $\mathbb{P}$ on it. A key feature of the random field framework is that there can be probabilistic dependence between the outputs $Y(\theta)$ and $Y(\theta')$ when the simulation is run at the inputs θ and θ'. This represents the effect of common random numbers in simulating the same model for two different values of its inputs. As is standard in analyzing Monte Carlo simulation, we assume that the second moment of the simulation output $Y(\theta)$ is finite for all $\theta \in \Theta$.

The goal is to estimate the mean function μ of the random field Y. The mean function, also known as the response surface, is the function $\mu : \Theta \to \mathbb{R}$ defined by

$$\mu(\theta) = \mathbb{E}[Y(\theta)] = \int_\Omega Y(\theta, \omega)\, \mathrm{d}\mathbb{P}(\omega).$$

A simulation-on-demand method enables us to produce an estimator $\hat{\mu}(\theta)$ of the value $\mu(\theta)$ of the response surface at a prediction point θ, once that prediction point θ has been chosen. Let $\mathbb{Q}$ denote a probability measure on Θ that specifies the distribution of a random prediction point ϑ. Then we can use the mean squared error (MSE) at a random prediction point,

$$\int_\Theta \mathbb{E}\left[(\hat{\mu}(\theta) - \mu(\theta))^2\right] \mathrm{d}\mathbb{Q}(\theta), \tag{1}$$

as the figure of merit. In other settings, there may be no given probability measure on Θ. Instead, one may treat every input $\theta \in \Theta$ uniformly. For example, one may use Lebesgue measure on Θ when it is a bounded, measurable subset of $\mathbb{R}^d$. This leads to integrated mean squared error (IMSE),

$$\int_\Theta \mathbb{E}\left[(\hat{\mu}(\theta) - \mu(\theta))^2\right] \mathrm{d}\theta, \tag{2}$$

as the figure of merit.

3 DATABASE MONTE CARLO WITH CONTROL VARIATES

This section reviews the method of Database Monte Carlo (DBMC) with control variates (Borogovac and Vakili 2008). It is the key tool used to develop our simulation-on-demand procedure in Section 4. With that application in mind, we will see here how DBMC with control variates (DBMC-CV) can produce an estimator $\hat{\mu}(\theta)$ of $\mu(\theta)$ quickly once a prediction point θ is chosen.

A control variate estimator of $\mu(\theta)$ has the form

$$\tilde{Y}_m(\theta) = \bar{Y}_m(\theta) - \beta\left(\bar{C}_m - \mathbb{E}[C]\right), \tag{3}$$

where:

- The sample average $\bar{Y}_m(\theta) = \sum_{j=1}^m Y^{(j)}(\theta)/m$ where the outputs $Y^{(1)}(\theta), \ldots, Y^{(m)}(\theta)$ are from m independent replications of the simulation run with input θ.
- The control variate C has known mean $\mathbb{E}[C]$. Its sample average over the same m replications is $\bar{C}_m = \sum_{j=1}^m C^{(j)}/m$. For each replication j, $C^{(j)}$ and $Y^{(j)}(\theta)$ are dependent because they are simulated using common random numbers.
- The choice of the coefficient β relates to the simple linear regression of $Y(\theta)$ on C. This is discussed in more detail in Section 4.3.

One would use the optimal coefficient $\beta^* = \mathrm{Cov}[Y(\theta), C]/\mathrm{Var}[C]$ if it were available. Using β^* would give the control variate estimator a bias of zero and a variance

$$\mathrm{Var}\left[\bar{Y}_m(\theta) - \beta^*\left(\bar{C}_m - \mathbb{E}[C]\right)\right] = (1-\rho^2)\frac{\mathrm{Var}[Y(\theta)]}{m} = (1-\rho^2)\mathrm{Var}\left[\bar{Y}_m(\theta)\right], \tag{4}$$

where ρ is the correlation between $Y(\theta)$ and C.

The challenge is to find a control variate that has a sufficiently large correlation ρ with the simulation output $Y(\theta)$. If this is done, then one can quickly run a modest number m of replications with input θ and produce a control variate estimator of $\mu(\theta)$ with a sufficiently small variance.

DBMC-CV is the tool that we use to accomplish this goal. The insight of DBMC-CV is that, in many simulation models, $Y(\theta')$ is highly correlated with $Y(\theta)$ if the input θ' is near the prediction point θ. Of course, we cannot expect to know the mean $\mu(\theta')$ of the simulation output $Y(\theta')$, so $Y(\theta')$ is not truly a control variate in the standard sense. However, $Y(\theta')$ can still be used as a quasi control variate (Emsermann and Simon 2002) or control variate with estimated mean (Pasupathy et al. 2012).

In general, in DBMC, a "database" constructed from a simulation run with input θ' is used to decrease the variance of a simulation run with input θ. In DBMC-CV, the database does this by providing an estimate of the mean $\mu(\theta')$ and thus enabling $Y(\theta')$ to be used as a control variate with estimated mean. The database contains M replications of the simulation output $Y(\theta')$ that are independent of the m replications of the simulation output $Y(\theta)$ at the prediction point θ. The DBMC-CV estimator of $\mu(\theta)$ has the form

$$\hat{\mu}(\theta) = \bar{Y}_m(\theta) - \beta\left(\bar{Y}_m(\theta') - \bar{Y}_M(\theta')\right), \tag{5}$$

where:

- The simulation is run for $M+m$ independent replications. For replications $j = 1,\ldots,M$, the simulation is run for input θ' only. For replications $j = M+1,\ldots,M+m$, the simulation is run for inputs θ' and θ, so that $Y^{(j)}(\theta')$ and $Y^{(j)}(\theta)$ are dependent due to common random numbers.
- The sample average $\bar{Y}_m(\theta) = \sum_{j=M+1}^{M+m} Y^{(j)}(\theta)/m$.
- The sample average $\bar{Y}_m(\theta') = \sum_{j=M+1}^{M+m} Y^{(j)}(\theta')/m$.
- The sample average $\bar{Y}_M(\theta') = \sum_{j=1}^{M} Y^{(j)}(\theta')/M$.
- The choice of the coefficient β relates to the simple linear regression of $Y(\theta)$ on $Y(\theta')$. This is discussed in more detail in Section 4.3.

Pasupathy et al. (2012) provide an expression for an optimal coefficient β^* and give an following expression for the variance of the DBMC-CV estimator if β^* were used:

$$\begin{aligned} \mathrm{Var}\left[\bar{Y}_m(\theta) - \beta^*\left(\bar{Y}_m(\theta') - \bar{Y}_M(\theta')\right)\right] &= \left(1-\rho^2\left(\frac{1}{1+m/M}\right)\right)\frac{\mathrm{Var}[Y(\theta)]}{m} \\ &= \left(1-\rho^2\left(\frac{1}{1+m/M}\right)\right)\mathrm{Var}\left[\bar{Y}_m(\theta)\right], \end{aligned} \tag{6}$$

where ρ is the correlation between $Y(\theta)$ and $Y(\theta')$.

The preceding analysis suggests the following sketch of a basic version of a simulation-on-demand procedure using DBMC-CV to produce an estimator $\hat{\mu}(\theta)$ of $\mu(\theta)$ quickly once a prediction point θ is chosen. The number of replications m run at the prediction point is the largest number that can be run within the time constraints imposed by the simulation-on-demand imperative.

1. Before the prediction point θ is chosen, choose a "database point" θ', and run $M+m$ replications of the simulation with input θ'.
2. Once the prediction point θ is chosen, run m replications of the simulation with input θ and compute the DBMC-CV estimator $\hat{\mu}(\theta)$ in Equation (5).

In the next section, we supply more details and enhancements in developing a practical simulation-on-demand procedure.

4 A PRACTICAL PROCEDURE

The previous section reviewed the DBMC-CV method of Borogovac and Vakili (2008) and showed how it can be used in our in our setting, ending with a sketch of a basic simulation-on-demand procedure. This section develops a practical, effective simulation-on-demand procedure (Section 4.4) by improving upon the sketch in two ways. One way is to supply the discussion, which was absent from the sketch and was deferred to Section 4.3, of how to choose the coefficient β in Equation (5). The second way is to address experiment design (Section 4.1) and multiple control variates (Section 4.2) so as to allow the simulation-on-demand procedure to produce a DBMC-CV estimator with sufficiently low variance.

To motivate this discussion of experiment design and multiple control variates, we explain why the basic procedure sketched in the previous section may yield an unacceptably large variance of the DBMC-CV estimator in Equation (5). In the basic procedure, there is only a single database point θ'. As the database size $M \to \infty$, the variance in Equation (6) converges to the variance in Equation (4), which applies to the case in which the mean of the control variate $Y(\theta')$ is known. This variance depends on the number m of replications that can be run at the prediction point θ in the time allowed by the simulation-on-demand imperative, and on the correlation ρ between $Y(\theta)$ and $Y(\theta')$. If m were so large that $\mathrm{Var}[Y(\theta)]/m$ were acceptably small, then simulation on demand could be achieved without DBMC-CV. Suppose, then, that $\mathrm{Var}[Y(\theta)]/m$ is unacceptably large. If so, then making the variance in Equation (4) acceptably small requires making ρ^2 sufficiently large. This may be impossible to achieve with a single database built by running the simulation at a single input θ'. The figures of merit, MSE at a random prediction point in Equation (1) or IMSE in Equation (2), contain more than just a single prediction point θ. Suppose that the correlation $\rho = \rho(\theta, \theta')$ decays to zero as the distance $\|\theta - \theta'\|$ grows to infinity. If so, then DBMC-CV with a single database point θ' provides little variance reduction for a prediction point θ that is sufficiently far from the database point. Then IMSE in Equation (2) could be too large for any choice of the database point θ' if the domain Θ is too large. Likewise, the distribution of the random prediction point ϑ could make the expected distance from ϑ to the database point θ' too large for any choice of θ', thus making MSE in Equation (1) too large.

4.1 Experiment Design

The preceding discussion, which motivated the need for more than one database point, suggests making DBMC-CV achieve good variance reduction for every possible prediction point $\theta \in \Theta$ by designing an experiment such that the average distance from any $\theta \in \Theta$ to the nearest database point is small. For the figure of merit IMSE in Equation (2), "average" relates to an integral interpreted as an unweighted average over points $\theta \in \Theta$. For the figure of merit MSE at a random prediction point ϑ in Equation (1), "average" means a weighted average over points $\theta \in \Theta$, where the weight on a point θ is specified by the probability measure $\mathbb{Q}$, or equivalently, by the distribution of ϑ.

Consider the unweighted average related to IMSE. Choosing the database points according to a space-filling design makes the average distance from a prediction point θ to the nearest database point small. As the basis of our space-filling design, we have chosen a scrambled Sobol' point set (Glasserman 2003). This is a point set in the unit hypercube $[0,1]^d$. It is easily transformed to a space-filling design for a rectangular set $\Theta \subset \mathbb{R}^d$.

Consider the weighted average related to MSE at a random prediction point ϑ. Let F be the distribution of ϑ on $\Theta \subset \mathbb{R}^d$. When we aim for low MSE at a random prediction point, our experiment design is the point set $\{F^{-1}(u) : u \in U\}$, where U is our space-filling design for the unit hypercube $[0,1]^d$.

Further research questions remain for experiment design for our simulation-on-demand procedure with DBMC-CV. In our simulation experiments reported in Section 5, we have used experiment designs with $M+m$ replications at each of k database points. Given a target for IMSE, how large should k be? If the total number of replications $k(M+m)$ is fixed, how should k and M be chosen? Should the number of

replications vary across database points, and if so, how? In the context of stochastic kriging, Ankenman, Nelson, and Staum (2010) formulated these questions in more detail and explored some of them.

4.2 Multiple Control Variates

Given multiple database points $\theta_1, \ldots, \theta_k$ as described in Section 4.1, it is attractive to use more than one among the simulation outputs $Y(\theta_1), \ldots, Y(\theta_k)$ as a control variate with estimated mean. The possibility of multiple control variates with estimated means was addressed already by Borogovac and Vakili (2008) in developing the DBMC-CV method. As is standard in the theory of control variates, one simply considers multiple linear regression in place of simple linear regression, which is used in the case of a single control variate.

The difficulty in our setting is that the number k of potential control variates may be large. Indeed, it could easily happen in practice that k exceeds m, the number of replications run at the prediction point θ. In that case, there would not be enough data to choose the coefficient vector β by multiple linear regression. Even if it were possible, because $m > k$, nonetheless using all of the potential control variates might not be the way to minimize the variance of the DBMC-CV estimator $\hat{\mu}(\theta)$. Stepwise regression has been suggested as a way to choose a good set of control variates to use from among a large set of potential control variates (Nelson 1987). At each step, forward stepwise regression selects the unused potential control variate that yields the greatest increase in R^2, the fraction of the variance explained by the linear model. However, it only adds the selected control variate if this leads to an increased in adjusted R^2, which equals $1-(1-R^2)(m-1)/(m-k'-1)$, where k' is the number of control variates in the linear model. When no further increase in adjusted R^2 is possible, forward stepwise regression stops.

4.3 Coefficients

As is standard in the application of control variates, we have chosen the coefficient vector β to be the least-squares-optimal coefficient $\hat{\beta}$ provided by linear regression. Let $\{i_1, \ldots, i_{k'}\}$ be the set of indices of database points that were selected to be used in the DBMC-CV estimator $\hat{\mu}(\theta)$. Let

$$C^{(j)} = \left[Y^{(j)}(\theta_{i_1}), \ldots, Y^{(j)}(\theta_{i_{k'}}) \right]. \tag{7}$$

Then $\hat{\beta}$ is provided by linear regression of $Y^{(M+1)}(\theta), \ldots, Y^{(M+m)}(\theta)$ on $C^{(M+1)}, \ldots, C^{(M+m)}$. We did not take account of the variance due to estimated means, as did Pasupathy et al. (2012) in providing an expression for the optimal β for a single control variate with estimated mean. This remains for future research.

Using $\hat{\beta}$ gives the DBMC-CV estimator $\hat{\mu}(\theta)$ a bias $\mathbb{E}[\hat{\mu}(\theta)] - \mu(\theta)$ which may be non-zero. This bias can be eliminated, if desired, by a splitting technique (Avramidis and Wilson 1993).

4.4 The Procedure

Before the prediction point θ is revealed, the procedure simulates $M+m$ replications at each of k design points using common random numbers:

1. Choose database points $\theta_1, \ldots, \theta_k$ based on a scrambled Sobol' point set (Section 4.1).
2. For each database $i = 1, \ldots, k$,
   (a) For each replication $j = 1, \ldots, M+m$, generate the simulation output $Y^{(j)}(\theta_i)$. (For any i and i', $Y^{(j)}(\theta_i)$ and $Y^{(j)}(\theta_{i'})$ are dependent due to common random numbers.)
   (b) Compute the sample averages $\bar{Y}_M(\theta_i) = \sum_{j=1}^{M} Y^{(j)}(\theta_i)/M$ and $\bar{Y}_m(\theta_i) = \sum_{j=M+1}^{M+m} Y^{(j)}(\theta_i)/m$.

After the prediction point θ is revealed, the procedure simulates m replications at the prediction point and performs stepwise regression to compute the DBMC-CV estimator:

1. For each replication $j = M+1, \ldots, M+m$, generate the simulation output $Y^{(j)}(\theta)$. (For any i, $Y^{(j)}(\theta)$ and $Y^{(j)}(\theta_i)$ are dependent due to common random numbers.) Compute the sample average $\bar{Y}_m(\theta) = \sum_{j=M+1}^{M+m} Y^{(j)}(\theta)/m$.
2. Use forward stepwise regression to select which databases to include in a linear model, and compute the corresponding coefficient vector $\hat{\beta}$ (Sections 4.2 and 4.3).
3. The DBMC-CV estimator of $\mu(\theta)$ is

$$\hat{\mu}(\theta) = \bar{Y}_m(\theta) - \hat{\beta}\left(\bar{C}_m - \bar{C}_M\right),$$

where $\bar{C}_m$ and $\bar{C}_M$ represent vectors of sample averages for the selected databases, similar to the selection in Equation (7).

5 SIMULATION EXPERIMENT

In this section we report a simulation experiment that illustrates the effectiveness of the proposed method. We compare the accuracy of DBMC-CV and stochastic kriging (Ankenman, Nelson, and Staum 2010), a metamodeling technique based on function approximation.

The experiments are performed on a marketing simulation based on Aydin and Porteus (2008). The input to the model is $\theta = [\theta_1, \theta_2]$, where θ_1 and θ_2 are the prices chosen for two products. The demand follows a stochastic logit model, based on the prices and the coefficients $\alpha_1 = 10, \alpha_2 = 25, c_1 = 6, c_2 = 20$, and incorporating random variables uniformly distributed between 100 and 500. The inventory levels are set equal to their optimal levels, which are known in closed form. The output of the simulation is profit. The response surface in this example, expected profit, is known in closed form: it is given by

$$\mu(\theta) = \frac{200\,(\theta_1 - 6)^2 \exp(10 - \theta_1)}{\theta_1\,(1 + \exp(10 - \theta_1) + \exp(25 - \theta_2))} + \frac{200\,(\theta_2 - 20)^2 \exp(25 - \theta_2)}{\theta_1\,(1 + \exp(10 - \theta_1) + \exp(25 - \theta_2))}.$$

Knowing the response surface in this example makes it easier to assess IMSE over the domain $\Theta = [5.5, 21.5] \times [13.2, 44.6]$ and compare the accuracy of DBMC-CV and SK. The response surface is shown in Figure 1.

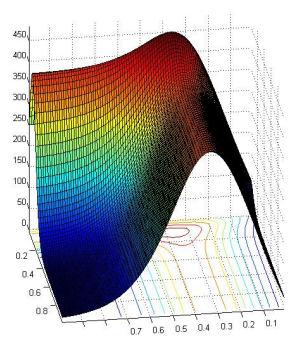

Figure 1: The response surface, expected profit as a function of two prices, in the marketing simulation. The horizontal axes have been re-scaled.

We assessed IMSE as follows. In each macro-replication of the simulation experiment, we simulated $M+m$ replications at each of $k = 49$ database points. Then we randomly sampled 1000 prediction points uniformly in Θ. For each prediction point θ, we simulated m replications at θ and computed the DBMC-CV estimator $\hat{\mu}(\theta)$ using only these m replications at this prediction point θ and the $M+m$ replications at the single nearest database point. We did not use stepwise regression in this experiment. The estimate of IMSE for this macro-replication was the average of $(\hat{\mu}(\theta) - \mu(\theta))^2$ over 1000 prediction points. We performed independent macro-replications of the simulation experiment, and computed the average of the

estimates of IMSE, and its standard error. For SK, the experiment was similar but independent. We did not use the same simulation output for SK as for DBMC-CV. DBMC-CV uses common random numbers. Chen, Ankenman, and Nelson (2012) showed that common random numbers increase the variance of the SK estimator of $\mu(\theta)$. For purposes of a fair comparison, the experiment for SK had the same structure as the experiment for DBMC-CV, but used independent sampling instead of common random numbers. No metamodel validation was performed.

Table 1 presents the results of the comparison between DBMC-CV, SK, and standard Monte Carlo. In the standard Monte Carlo method, the estimator of $\mu(\theta)$ is simply $\bar{Y}_m(\theta)$, the sample average of the m replications at the prediction point θ. When DBMC-CV was used with the database size M an order of magnitude larger than the number m of replications at the prediction point, it yielded IMSE about an order of magnitude smaller than the IMSE of standard Monte Carlo. Larger database sizes resulted in further significant reductions in IMSE. However, if M and m were the same size, IMSE was not significantly reduced by DBMC-CV compared to standard Monte Carlo. For this reason, we did not even run experiments in which the database size M was smaller than m, which caused the blank entries in Table 1. SK resulted in a much larger IMSE than standard Monte Carlo. That is, it was better to use only m replications at the prediction point than to put them into a SK metamodel along with the simulation output at the database points. The reason is the bias caused by SK. This bias arises because the SK estimator of $\mu(\theta)$ puts a substantial amount of weight on database points which, in this experiment design, are often too far from the prediction point θ. This bias gets larger for larger database size M. Large M makes the standard error associated with the database points small compared to the standard error associated with the prediction point, which decreases the weight put on the simulation output at the prediction point. This effect on weights caused by large heterogeneity in standard errors is a known source of trouble with bias in SK (Staum 2009). We conclude that, in this example, DBMC-CV is highly effective in reducing IMSE, whereas SK metamodels were not reliably valid, given a similar experiment design.

Table 1: Estimates of integrated mean squared error (IMSE) in the marketing simulation, for three simulation-on-demand methods: standard Monte Carlo, Database Monte Carlo with control variates ("DBMC-CV"), and stochastic kriging. There are M replications at each of $k = 49$ database points and m replications at a prediction point. Standard errors of the IMSE estimates are in parentheses beneath the estimates.

	Standard	DBMC-CV			Stochastic Kriging		
m	Monte Carlo	$M = 10^3$	$M = 10^4$	$M = 10^5$	$M = 10^3$	$M = 10^4$	$M = 10^5$
10^2	2.6×10^2	2.6×10^1	3.0×10^0	8×10^{-1}	1.3×10^3	1.4×10^3	1.6×10^3
		(5×10^{-1})	(5×10^{-2})	(1×10^{-2})	(4×10^0)	(6×10^0)	(7×10^0)
10^3	2.6×10^1	2.3×10^1	2.9×10^0	3×10^{-1}	6.0×10^2	6.2×10^2	7.1×10^2
		(4×10^{-1})	(6×10^{-2})	$(< 10^{-2})$	(3×10^{-1})	(3×10^{-1})	(3×10^{-1})
10^4	2.6×10^0		2.8×10^0	2×10^{-1}		3.6×10^0	3.5×10^0
			(6×10^{-2})	$(< 10^{-2})$		(1×10^{-2})	(1×10^{-2})

6 CONCLUSIONS AND FUTURE RESEARCH

We proposed a simulation-on-demand method that does not require metamodel validation, based on Database Monte Carlo with control variates (DBMC-CV). In a simulation experiment (Section 5), we saw that the proposed DBMC-CV method was successful, given sufficiently large databases: depending on database size, DBMC-CV reduced integrated mean squared error by a factor of 10-300 compared to standard Monte Carlo.

In ongoing research, we are experimenting with stepwise regression in DBMC-CV (Section 4.2) and investigating computational complexity. Other future research topics include experiment design (see Section 4.1) and selecting the coefficient vector $\hat{\beta}$ for the control variates (see Section 4.3). Furthermore,

other DBMC methods (Zhao, Zhou, and Vakili 2006, Zhao, Borogovac, and Vakili 2007, Zhao and Vakili 2008) may be applied to simulation on demand.

It also remains to apply DBMC-CV to steady-state simulations in which only a single replication or only a few replications would be run at the prediction point. One issue is that having a small number of replications creates a challenge for selecting the coefficient vector $\hat{\beta}$ for the control variates (Section 4.3). This can be addressed by batching (Nelson 1989). There is also the opportunity to use a longer run length in simulation at database points than in the simulation at the prediction point, which could help to mitigate initial-transient bias from a short run at the prediction point.

ACKNOWLEDGMENTS

The authors are grateful to Barry Nelson and Fred Hickernell for discussions.

REFERENCES

Ankenman, B. E., J. M. Bekki, J. Fowler, G. T. Mackulak, B. L. Nelson, and F. Yang. 2011. "Simulation in Production Planning: An Overview with Emphasis on Recent Developments in Cycle Time Estimation". In *Planning Production and Inventories in the Extended Enterprise*, edited by K. G. Kempf, P. Keskinocak, and R. Uzsoy, Chapter 19, 565–591. New York: Springer Science+Business Media.

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

Avramidis, A. N., and J. R. Wilson. 1993. "A Splitting Scheme for Control Variates". *Operations Research Letters* 14:187–198.

Aydin, G., and E. L. Porteus. 2008. "Joint Inventory and Pricing Decisions for an Assortment". *Operations Research* 56 (5): 1247–1255.

Borogovac, T., and P. Vakili. 2008. "Control Variate Technique: A Constructive Approach". In *Proceedings of the 2008 Winter Simulation Conference*, edited by S. J. Mason, R. R. Hill, L. Mönch, O. Rose, T. Jefferson, and J. W. Fowler, 320–327. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

Chen, X., B. E. Ankenman, and B. L. Nelson. 2012. "The Effects of Common Random Numbers on Stochastic Kriging Metamodels". *ACM Transactions on Modeling and Computer Simulation* 22 (2).

Emsermann, M., and B. Simon. 2002. "Improving Simulation Efficiency with Quasi Control Variates". *Stochastic Models* 18 (3): 425–448.

Glasserman, P. 2003. *Monte Carlo Methods in Financial Engineering*. Springer-Verlag.

Kleijnen, J. 2007. *Design and Analysis of Simulation Experiments*. International Series in Operations Research & Management Science. Springer.

Liu, M., B. L. Nelson, and J. Staum. 2010. "Simulation on Demand for Pricing Many Securities". In *Proceedings of the 2010 Winter Simulation Conference*, edited by B. Johansson, S. Jain, J. Montoya-Torres, J. Hugan, and E. Yücesan, 2782–2789. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

Nelson, B. L. 1987. "Variance Reduction for Simulation Practitioners". In *Proceedings of the 1987 Winter Simulation Conference*, edited by A. Thesen, H. Grant, and W. D. Kelton, 43–51. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

Nelson, B. L. 1989. "Batch Size Effects on the Efficiency of Control Variates in Simulation". *European Journal of Operational Research* 43:184–196.

Pasupathy, R., B. W. Schmeiser, M. R. Taaffe, and J. Wang. 2012. "Control-Variate Estimation Using Estimated Control Means". *IIE Transactions* 44 (5): 381–385.

Staum, J. 2009. "Better Simulation Metamodeling: The Why, What, and How of Stochastic Kriging". In *Proceedings of the 2009 Winter Simulation Conference*, edited by A. Dunkin, R. G. Ingalls, E. Yücesan,

M. D. Rossetti, R. Hill, and B. Johansson, 119–133. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

Taylor, S. J. E., R. Fujimoto, E. H. Page, P. A. Fishwick, A. M. Uhrmacher, and G. Wainer. 2012. "Panel on Grand Challenges for Modeling and Simulation". In *Proceedings of the 2012 Winter Simulation Conference*, edited by C. Laroque, J. Himmelspach, R. Pasupathy, O. Rose, and A. M. Uhrmacher, 2614–2628. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

Zhao, G., T. Borogovac, and P. Vakili. 2007. "Efficient estimation of option price and price sensitivities via structured database Monte Carlo (SDMC)". In *Proceedings of the 2007 Winter Simulation Conference*, edited by S. G. Henderson, B. Biller, M.-H. Hsieh, J. Shortle, J. D. Tew, and R. R. Barton, 984–990. Piscataway, N. J.: IEEE Press.

Zhao, G., and P. Vakili. 2008. "Monotonicity and stratification". In *Proceedings of the 2008 Winter Simulation Conference*, edited by S. J. Mason, R. R. Hill, L. Mönch, O. Rose, T. Jefferson, and J. W. Fowler, 313–319. Piscataway, N. J.: IEEE Press.

Zhao, G., Y. Zhou, and P. Vakili. 2006. "A new efficient simulation strategy for pricing path-dependent options". In *Proceedings of the 2006 Winter Simulation Conference*, edited by L. F. Perrone, F. P. Wieland, J. Liu, B. G. Lawson, D. M. Nicol, and R. M. Fujimoto, 703–710. Piscataway, N. J.: IEEE Press.

AUTHOR BIOGRAPHIES

IMRY ROSENBAUM is a Ph. D. candidate in the Department of Industrial Engineering and Management Sciences at Northwestern University. His research interest is efficient computer simulation methods, with application to financial models. His e-mail address is imryrosenbaum2016@u.northwestern.edu.

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.