Simulation on Demand for Pricing Many Securities

Ming Liu[1], Barry L. Nelson and Jeremy Staum

Department of Industrial Engineering and Management Sciences, Robert R. McCormick School of Engineering and Applied Science, Northwestern University, Evanston, IL 60208-3119, U.S.A.

Abstract

We develop a sequential experiment design procedure to construct multiple metamodels based on a single stochastic simulation model. We apply the procedure to approximate many securities' prices as functions of a financial scenario. We propose a cross-validation method that adds design points and simulation effort at the design points to target all metamodels' relative prediction errors. To improve the expected quality of the metamodels given randomness of the scenario that is an input to the simulation model, we also propose a way to choose design points so that the scenario is likely to fall inside their convex hull.
Keywords: simulation on demand, metamodeling, stochastic kriging, cross-validation, risk management

1. Introduction

"Simulation on demand" is a computing paradigm that delivers real-time answers as accurate as those that would be generated by a time-consuming run of a simulation model. This is achieved by investing computational effort in advance, before decision-makers ask about a specific scenario. Simulation metamodeling is one method that supports simulation on demand. A metamodel is an approximation to the function y of interest, whose value $y(\boldsymbol{x})$ in scenario $\boldsymbol{x}$ is estimated by running the simulation model for scenario $\boldsymbol{x}$. Building a metamodel requires a simulation experiment in which the simulation model is run for several

[1]Corresponding author. Email address: mingliu2010@u.northwestern.edu

scenarios, but after this computational investment has been made, it can be very fast to evaluate the metamodel in any scenario. This article is devoted to experiment design in building multiple metamodels based on the same simulation model.

We consider a financial example: a firm deals in many securities whose prices are functions of the financial scenario. A single simulation model is used to determine the securities' prices. Simulation on demand provides an approximate picture of the way the prices change as the markets move. Our procedure is related to one due to Frye [4], involving a grid-based interpolation technique, which requires a grid design for the simulation experiment. Because of the impracticality of a high-dimensional grid design, Frye used principal component analysis to reduce the dimension of the simulation model. One of our contributions is to extend Frye's work by showing that the latest metamodeling techniques make it computationally feasible to construct highly accurate metamodels of each security's price, not merely one moderately accurate metamodel of the portfolio's value. In particular, by using an experiment design that is practical in higher dimension, we avoid the loss of accuracy entailed by dimension reduction. This requires a metamodeling technique other than grid-based interpolation. We use stochastic kriging [1], but our procedure works with many metamodeling techniques.

Our main contribution is a sequential experiment design procedure that adds design points and simulation effort at the design points to reduce all metamodels' prediction errors to an acceptable level relative to the true values. It is appropriate to focus on relative (not absolute) error in applications involving multiple metamodels which have very different magnitudes, including our financial example, in which some options have much larger prices than others. The key ingredient in our procedure is cross-validation, which is widely used for validating prediction schemes [5]. Kleijnen [6] discusses cross-validation in stochastic simulation metamodeling. The novel aspect of our procedure for stochastic simulation is that it continues until the prediction errors are likely to be small, instead of continuing until the simulation output is consistent with the metamodeling technique's assumptions.

In some applications, the metamodels are functions of a scenario that can be regarded as random: in our financial example, the scenario that will occur tomorrow is random. In such applications, it is meaningful to consider the expected performance of the metamodels at a randomly selected scenario. Another contribution of this article is an initial experiment design that aims to make the scenario fall inside the convex hull of design points with high probability. This improves the metamodels' performance because many metamodeling techniques are much better at interpolation than at extrapolation.

2. Motivating Example

We consider a portfolio of 75 European-style options. The underlying vector stochastic process models six equity indices: S&P500, Nikkei225, Stoxx50, FTSE100, Hang Seng, and KOSPI Composite. To keep the example simple, we use a very basic model and approach to model calibration. The equity indices, denoted by $j = 1, 2, \ldots, 6$, follow geometric Brownian motion where the non-annualized daily volatilities σ_j and correlation matrix $\Sigma_{\boldsymbol{X}}$ are estimated from 1000 historical daily returns. A scenario $\boldsymbol{X}$ is a vector whose components are called risk factors, and it determines the values of the equity indices. The risk factors are six standard normal random variables with correlation matrix $\Sigma_{\boldsymbol{X}}$. If tomorrow's scenario is $\boldsymbol{X}$, then for each $j = 1, 2, \ldots, 6$, tomorrow's value of the jth index is $S_j(1) = S_j(0) \exp(\sigma_j X_j)$, where $S_j(0)$ is today's value of the jth index. Here we have set the drift to zero, because it is negligible over one day. The portfolio contains five classes of options: put options on the average return of all six indices, call options on the average return of the S&P500, Nikkei225, and Stoxx50, call options on the average return of the FTSE100, Hang Seng, and KOSPI Composite, call options on the minimum return of the S&P500, Nikkei225, and Stoxx50, and call options on the minimum return of the FTSE100, Hang Seng, and KOSPI Composite. Within each class, there are 15 options with one of five maturities and one of three strike prices. The maturities are 3, 4, 5, 6, and 7 years. The three strike prices are chosen to make the option in the money, (roughly) at the money, or out of the money. Given that tomorrow's scenario is $\boldsymbol{X}$, the value of the jth index after t days is

$$S_j(t) = S_j(1) \exp\left(\left(\mu_j - \frac{\sigma_j^2}{2}\right)(t-1) + \sigma_j \sqrt{t-1} B_j\right), \tag{1}$$

where $\boldsymbol{B}$ is a vector of six standard normal random variables with correlation matrix $\Sigma_{\boldsymbol{X}}$, and each risk-neutral drift μ_j equals the non-annualized daily yield of a government bond denominated in the relevant currency minus the dividend yield of the index. A sample path, simulated conditional on $\boldsymbol{X}$, includes the values of all six equity indices after 3, 4, 5, 6, and 7 years. The simulation model for option pricing computes discounted payoffs for all 75 options on a single sample path.

3. Simulation Procedure

Our procedure chooses scenarios, called design points, in which to run simulations. It also determines the number of sample paths to simulate at each design point. A sample path is a simulation of the underlying stochastic process conditional on the scenario given by the design point, as in Equation (1). The sample average $\bar{Y}_h(\boldsymbol{x})$ of the discounted payoffs of security h on every sample path simulated at design point $\boldsymbol{x}$ serves as an estimate of the price of security h in the scenario $\boldsymbol{x}$. The procedure produces one metamodel $\hat{\mathsf{Y}}_h$ for the price of each security $h = 1, 2, \ldots, r$. The goal of the procedure's first phase, initial simulation, is to get accurate estimates of all security prices at all design points. This is not enough to ensure that the metamodels will give accurate estimates of all security prices at scenarios that are not design points. The goal of the second phase, metamodel validation, is to improve the accuracy of the metamodels away from the design points. This phase adds design points and sample paths until the metamodels pass a cross-validation test of their ability to estimate security prices at scenarios that are not used at design points. An outline of the procedure is:

Phase I. Initial Simulation

1. Generate k design points $\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_k$.
2. For $i = 1, 2, \ldots, k$, simulate n_i sample paths at design point $\boldsymbol{x}_i$, where n_i is chosen to target the relative accuracy in estimating the value of each security at $\boldsymbol{x}_i$.

Phase II. Metamodel Validation

1. Create metamodels by stochastic kriging.
2. Perform cross-validation on the metamodels. If they all pass the cross-validation test, the procedure terminates.
3. If they do not all pass, simulate more sample paths at existing design points or generate additional design points and simulate sample paths at them. Then return to Step 1 of Phase II.

3.1. Initial Simulation Phase

The initial simulation phase consists of two parts: a method for choosing k design points and a two-stage simulation procedure. Information from the first stage, which simulates n_0 sample paths at each design point, is used to choose the total sample size n_i at each design point $\boldsymbol{x}_i, i = 1, 2, \ldots, k$. The second stage of simulation generates the sample paths required to reach those total sample sizes.

3.1.1. Design Points

In choosing design points, we have two goals. One goal is for tomorrow's scenario to fall inside the convex hull of the design points with high probability. The reason for this is that many metamodeling techniques, including stochastic kriging, are much better at interpolation than at extrapolation. The other goal is, as usual in simulation metamodeling, to fill the space of scenarios evenly with design points.

To address the probability that tomorrow's scenario $\boldsymbol{X}$ falls inside the convex hull of the design points, we need a joint distribution $F_{\boldsymbol{X}}$ for the risk factors $X_1, X_2, \ldots, X_d$. Our procedure requires that we be able to evaluate a function f such that, if $\boldsymbol{U}$ is uniformly distributed on $(0,1)^d$, then $f(\boldsymbol{U})$ has distribution $F_{\boldsymbol{X}}$. We choose design points $\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_k$ in $\mathcal{X}$ by choosing design points $\boldsymbol{u}_1, \boldsymbol{u}_2, \ldots, \boldsymbol{u}_k$ in $\mathcal{U}$ and transforming them to get $\boldsymbol{x}_i = f(\boldsymbol{u}_i)$ for $i = 1, 2, \ldots, k$. One way to get $f : \mathcal{U} \to \mathcal{X}$ is from a simulation algorithm: even if f is not known explicitly, a typical simulation algorithm takes independent uniform random variables as inputs and generates a random vector with distribution $F_{\boldsymbol{X}}$. However, sometimes the analyst may have only marginal distributions $F_{X_1}, F_{X_2}, \ldots, F_{X_d}$ for the risk factors and their correlation matrix $\Sigma_{\boldsymbol{X}}$. A further assumption about dependence among the risk factors is required to get a joint distribution $F_{\boldsymbol{X}}$. An assumption often made in financial engineering models (although it may give an unrealistic picture of extreme events), is that $\boldsymbol{X}$ has a Gaussian copula [8, § 5.1]. The same assumption is used in simulation input modeling in the normal-to-anything (NORTA) transformation [2]. Then the transformation $\boldsymbol{X} = f(\boldsymbol{U})$ is accomplished as follows, where Φ is the standard normal cumulative distribution function:

1. For each $i = 1, 2, \ldots, k$, set $W_i = \Phi^{-1}(U_i)$ to get a standard normal vector $\boldsymbol{W}$.
2. Where $\Sigma_{\boldsymbol{Z}}^{-1/2}$ satisfies $\Sigma_{\boldsymbol{Z}}^{-1/2} \left(\Sigma_{\boldsymbol{Z}}^{-1/2}\right)^{\top} = \Sigma_{\boldsymbol{Z}}$, set $\boldsymbol{Z} = \Sigma_{\boldsymbol{Z}}^{-1/2} \boldsymbol{W}$ to get a vector $\boldsymbol{Z}$ with standard normal marginal distributions and correlation matrix $\Sigma_{\boldsymbol{Z}}$.
3. For each $i = 1, 2, \ldots, k$, set $\tilde{U}_i = \Phi(Z_i)$ to get a vector $\tilde{\boldsymbol{U}}$ whose components are dependent and have marginal distributions that are uniform on $(0, 1)$.
4. For each $i = 1, 2, \ldots, k$, set $X_i = F_{X_i}^{-1}(\tilde{U}_i)$ to get a vector $\boldsymbol{X}$.

Cario and Nelson [2, 3] show how to choose $\Sigma_{\boldsymbol{Z}}$ so that the correlation matrix of $\boldsymbol{X}$ is $\Sigma_{\boldsymbol{X}}$.

Our experiment design contains two kinds of points: corner points and points sampled via quasi-Monte Carlo. Let p be a target probability for tomorrow's scenario to fall into the convex hull of the design points. Each of the 2^d corner points is the image under f of a vertex of the hypercube $\mathcal{U}_p = \{\boldsymbol{u} : 0.5(1 - p^{1/d}) \leq u_j \leq 0.5(1 + p^{1/d})\ \forall j = 1, 2, \ldots d\}$, which has volume p. Including 2^d corner points is feasible when the dimension d is moderate. The target probability p must be less than one if it is impossible to map the vertices of the unit hypercube $[0, 1]^d$ to usable scenarios in $\mathcal{X}$: e.g., in the example of Section 2, if $U_1 = 1$, then the first risk factor X_1 and the value S_1 of the S&P500 index are infinite. The probability that tomorrow's scenario falls inside the convex hull of the design points is not guaranteed to be p, but in many cases, choosing a large target probability p makes tomorrow's scenario fall inside the convex hull of the design points with high probability. The remaining $k - 2^d$ design points are generated from a Sobol' sequence scaled to fit inside $\mathcal{U}_p$: if $\boldsymbol{u}'$ is a point in a Sobol' sequence in $[0, 1)^d$, the corresponding design point is $f(0.5(1 - p^{1/d}) + p^{1/d}\boldsymbol{u}')$. Figure 1, whose panels are in sequence left to right and then top to bottom, shows the process of creating $k = 50$ design points for a version of the example of Section 2 in which there are only two risk factors, corresponding to the S&P500 and Nikkei225 indices. The panels for $\boldsymbol{Z}$ and $\boldsymbol{X}$ are the same because the risk factors in this example are normally distributed, so the step of generating $\tilde{\boldsymbol{U}}$ is redundant.

3.1.2. Sample Sizes in a Two-Stage Simulation Procedure

In the first stage of Phase I, we simulate n_0 sample paths at each design point and compute the sample average $\bar{Y}_h(\boldsymbol{x}_i)$ and sample variance $s_h^2(\boldsymbol{x}_i)$ of the discounted payoffs for each design point $i = 1, 2, \ldots, k$ and each security $h = 1, 2, \ldots, r$. Then we choose a total sample size n_i to attain at each design point after a second stage of sampling, and simulate $n_i - n_0$ additional sample paths at design point $\boldsymbol{x}_i$ for $i = 1, 2, \ldots, k$. We choose the sample size n_i to target a relative precision for the simulation output. Based on an assumption that the discounted payoffs are normally distributed with unknown variance, a fixed sample size of n yields a half-width of the $(1-\alpha)$ confidence interval for $y_h(\boldsymbol{x}_i)$ of $l_h(\boldsymbol{x}_i, n; \alpha) = t_{n-1,1-\alpha/2} s_h(\boldsymbol{x}_i)/\sqrt{n}$, where $t_{n-1,1-\alpha/2}$ is the $1-\alpha/2$ quantile of the t distribution with $n-1$ degrees of freedom. Let $y_h(\boldsymbol{x}_i)$ be the true price of security h in scenario $\boldsymbol{x}_i$. The relative precision of the average of n_i sample paths is $l_h(\boldsymbol{x}_i, n_i; \alpha)/|y_h(\boldsymbol{x}_i)|$. We target a relative precision of γ between 0 and 1. After the first stage, we choose the sample size

$$n_i = \max\left\{n_0, \left\lceil \max_{h=1,2,\dots,r} \left(\frac{(1+\gamma)t_{n_0-1,1-\alpha/2} s_h(\boldsymbol{x}_i)}{\gamma \bar{Y}_h(\boldsymbol{x}_i)}\right)^2 \right\rceil\right\} \tag{2}$$

because it makes

$$\frac{l_h(\boldsymbol{x}_i, n_i; \alpha)}{\left|\bar{Y}_h(\boldsymbol{x}_i)\right|} \leq \frac{\gamma}{1+\gamma} \tag{3}$$

for every security $h = 1, 2, \dots, r$. The relative precision $l_h(\boldsymbol{x}_i, n_i; \alpha)/|y_h(\boldsymbol{x}_i)| \leq \gamma$ with approximately $1-\alpha$ level confidence if Equation (3) holds [7, p. 502]. This is merely an approximation because the discounted payoffs are not normally distributed, and the sample size n_i is random, not fixed. In particular, it depends on the first-stage sample average $\bar{Y}_h(\boldsymbol{x}_i)$, which spoils the usual arguments for the validity of two-stage fixed-width confidence interval procedures such as Stein's [9]. However, if the sample size is large and the true price is not too close to zero, we expect to attain the desired relative precision with high confidence if α is small.

3.2. Metamodel Validation Phase

After the initial simulation, we construct and validate metamodels. If they fail a test based on leave-one-out cross-validation, we add more design points or generate more sample paths at existing design points until the updated metamodels pass the test. The essential idea of leave-one-out cross-validation, when applied to a metamodel of a deterministic simulation model, is to look at the difference between the true value $y_h(\boldsymbol{x}_i)$, observed by running the simulation at the design point $\boldsymbol{x}_i$, and the leave-one-out prediction $\hat{\mathsf{Y}}_h^{(-i)}(\boldsymbol{x}_i)$ of a metamodel constructed using all the design points except $\boldsymbol{x}_i$. When the simulation is stochastic, the true value $y_h(\boldsymbol{x}_i)$ can not be observed. Our validation method considers the leave-one-out prediction $\hat{\mathsf{Y}}_h^{(-i)}(\boldsymbol{x}_i)$, the simulation output $\bar{Y}_h(\boldsymbol{x}_i)$ which serves as an estimate of the true value $y_h(\boldsymbol{x}_i)$, and the confidence interval half-width $l_h(\boldsymbol{x}_i, n_i; \alpha)$ as a measure of uncertainty in $\bar{Y}_h(\boldsymbol{x}_i)$.

We use cross-validation for a subset $\mathbb{I} \subseteq \{1, 2, \ldots, k\}$ of design points and a subset $\mathbb{H} \subseteq \{1, 2, \ldots, r\}$ of securities. The design points in $\mathbb{I}$ are those that are not on the convex hull of the set of design points. This ensures that for all $i \in \mathbb{I}$, $\hat{\mathsf{Y}}_h^{(-i)}(\boldsymbol{x}_i)$ is an interpolation, not an extrapolation. The subset $\mathbb{H}$ could contain all r securities, but if r is too large, cross-validation will take a very long time. One may choose a smaller subset $\mathbb{H}$ by including only one representative of each class of securities. For example, a class may consist of securities which differ from each other only in maturity and strike price. We suggest choosing the representative of a class to be the security which is most computationally expensive to price with good relative accuracy: for example, after Phase I, one may choose representatives with the highest value of $\max_{i=1,2,\ldots,k} s_h(\boldsymbol{x}_i)/\bar{Y}_h(\boldsymbol{x}_i)$.

Proposition 1 provides some justification for our method, which aims to control the relative leave-one-out prediction error $|\hat{\mathsf{Y}}_h^{(-i)}(\boldsymbol{x}_i) - y_h(\boldsymbol{x}_i)|/|y_h(\boldsymbol{x}_i)|$ at each design point. As in Section 3.1.2, non-normality of the simulation output and randomness of the sample sizes mean that the method is merely approximate: thus, in interpreting the proposition, we should remember that $\Pr\left\{\left|\bar{Y}_h(\boldsymbol{x}_i) - y_h(\boldsymbol{x}_i)\right| \leq l_h(\boldsymbol{x}_i, n_i; \alpha)\right\}$ may not be exactly $1 - \alpha$, the confidence sought in the construction of Equation (2). For reasons supplied by Proposition 1, our test of validity is based on

$$E_{hi} = \frac{l_h(\boldsymbol{x}_i, n_i; \alpha)}{\left|\bar{Y}_h(\boldsymbol{x}_i)\right| - l_h(\boldsymbol{x}_i, n_i; \alpha)} + \frac{\left|\hat{\mathsf{Y}}_h^{(-i)}(\boldsymbol{x}_i) - \bar{Y}_h(\boldsymbol{x}_i)\right|}{\left|\bar{Y}_h(\boldsymbol{x}_i)\right| - l_h(\boldsymbol{x}_i, n_i; \alpha)}, \tag{4}$$

where the first term measures the relative precision of simulation output and the second term measures the relative discrepancy between metamodel prediction and simulation output. These measurements are relative to $\left|\bar{Y}_h(\boldsymbol{x}_i)\right| - l_h(\boldsymbol{x}_i, n_i; \alpha)$, a lower confidence limit for $y_h(\boldsymbol{x}_i)$.

Proposition 1. *For any $h = 1, 2, \ldots, r$ and $i = 1, 2, \ldots, k$, $\left|\bar{Y}_h(\boldsymbol{x}_i) - y_h(\boldsymbol{x}_i)\right| \leq l_h(\boldsymbol{x}_i, n_i; \alpha)$ implies $\left|\hat{\mathsf{Y}}_h^{(-i)}(\boldsymbol{x}_i) - y_h(\boldsymbol{x}_i)\right| / \left|y_h(\boldsymbol{x}_i)\right| \leq E_{hi}$.*

Proof. If $\bar{Y}_h(\boldsymbol{x}_i) - l_h(\boldsymbol{x}_i, n_i; \alpha) \leq y_h(\boldsymbol{x}_i) \leq \bar{Y}_h(\boldsymbol{x}_i) + l_h(\boldsymbol{x}_i, n_i; \alpha)$, then

$$\left|y_h(\boldsymbol{x}_i)\right| \geq \min\left\{\left|\bar{Y}_h(\boldsymbol{x}_i) - l_h(\boldsymbol{x}_i, n_i; \alpha)\right|, \left|\bar{Y}_h(\boldsymbol{x}_i) + l_h(\boldsymbol{x}_i, n_i; \alpha)\right|\right\} = \left|\bar{Y}_h(\boldsymbol{x}_i)\right| - l_h(\boldsymbol{x}_i, n_i; \alpha),$$

which is positive because Equation (2) implies that $l_h(\boldsymbol{x}_i, n_i; \alpha) < |\bar{Y}_h(\boldsymbol{x}_i)|$. Therefore

$$\begin{aligned}
\frac{\left|\hat{\mathsf{Y}}_h^{(-i)}(\boldsymbol{x}_i) - y_h(\boldsymbol{x}_i)\right|}{\left|y_h(\boldsymbol{x}_i)\right|} &\leq \frac{\left|\bar{Y}_h(\boldsymbol{x}_i) - y_h(\boldsymbol{x}_i)\right| + \left|\hat{\mathsf{Y}}_h^{(-i)}(\boldsymbol{x}_i) - \bar{Y}_h(\boldsymbol{x}_i)\right|}{\left|y_h(\boldsymbol{x}_i)\right|} \\
&\leq \frac{l_h(\boldsymbol{x}_i, n_i; \alpha) + \left|\hat{\mathsf{Y}}_h^{(-i)}(\boldsymbol{x}_i) - \bar{Y}_h(\boldsymbol{x}_i)\right|}{\left|\bar{Y}_h(\boldsymbol{x}_i)\right| - l_h(\boldsymbol{x}_i, n_i; \alpha)} = E_{hi}.
\end{aligned}$$

$\square$

The proposition suggests an iterative procedure that adds simulation effort until $E_{hi} \leq \beta$ for all $h \in \mathbb{H}$ and $i \in \mathbb{I}$, where β is a target error. A key question is whether to add more design points or more sample paths at existing design points. This is a difficult question and here we provide a very simple response to it; future research should lead to better answers. The difference $|\hat{\mathsf{Y}}_h^{(-i)}(\boldsymbol{x}_i) - \bar{Y}_h(\boldsymbol{x}_i)|$ can be large because the simulation output is far from the true value (at $\boldsymbol{x}_i$ or at other design points) or because of a large difference between the true value $y_h(\boldsymbol{x}_i)$ and the leave-one-out prediction at $\boldsymbol{x}_i$ that would arise if the true values were known at the other design points. In the former case, we want to add more sample paths; in the latter, we want to add more design points. The problem is that we do not know the true values, so we do not know the cause. However, if the sample size at $\boldsymbol{x}_i$ is already large enough to make the half-width of the confidence interval for $y_h(\boldsymbol{x}_i)$ very small, then it is unlikely that the cause is that $\bar{Y}_h(\boldsymbol{x}_i)$ is far from $y_h(\boldsymbol{x}_i)$, so it seems attractive to add a new design point. We add a new design point if the first term $l_h(\boldsymbol{x}_i, n_i; \alpha) / \left(\left|\bar{Y}_h(\boldsymbol{x}_i)\right| - l_h(\boldsymbol{x}_i, n_i; \alpha)\right)$ of E_{hi} is less than $\lambda\beta$, where $\lambda \in (0,1)$ is a parameter of the simulation procedure whose purpose is to control the effect of Monte Carlo variability during cross-validation. In our experiments, we found that $\lambda = 1/4$ worked well. An outline of Phase II of our procedure is:

1. For $i = 1, 2, \ldots, k$, initialize the sample size $N_i \leftarrow n_i$.
2. For $i = 1, 2, \ldots, k$ and $h \in \mathbb{H}$, compute the sample average $\bar{Y}_h(\boldsymbol{x}_i)$ and sample standard deviation $s_h(\boldsymbol{x}_i)$ of the N_i discounted payoffs of security h on each sample path at $\boldsymbol{x}_i$.
3. For all $i \in \mathbb{I}$ and $h \in \mathbb{H}$, compute E_{hi} as in Equation (4) but with sample size N_i.
4. Set $(h^*, i^*) \leftarrow \arg\max_{h \in \mathbb{H}, i \in \mathbb{I}} E_{hi}$. If $E_{h^* i^*} \leq \beta$, terminate.
5. If $l_{h^*}(\boldsymbol{x}_{i^*}, N_{i^*}; \alpha) / \left(\bar{Y}_{h^*}(\boldsymbol{x}_{i^*}) - |l_{h^*}(\boldsymbol{x}_{i^*}, N_{i^*}; \alpha)|\right) \geq \lambda\beta$, then:
   (a) Simulate N_{i^*} additional sample paths at $\boldsymbol{x}_{i^*}$ and set $N_{i^*} \leftarrow 2N_{i^*}$.
   (b) Update $\bar{Y}_h(\boldsymbol{x}_{i^*})$ and $s_h(\boldsymbol{x}_{i^*})$ for all $h \in \mathbb{H}$.
   (c) Return to Step 3.
6. Otherwise,
   (a) Add design point $\boldsymbol{x}_{k+1}$ midway between $\boldsymbol{x}_{i^*}$ and the nearest design point, and set $\mathbb{I} \leftarrow \mathbb{I} \cup \{k+1\}$.
   (b) Perform two-stage simulation at $\boldsymbol{x}_{k+1}$ as described in Section 3.1, initialize $N_{k+1} \leftarrow n_{k+1}$, and compute $\bar{Y}_h(\boldsymbol{x}_{k+1})$ and $s_h(\boldsymbol{x}_{k+1})$ for all $h \in \mathbb{H}$.
   (c) Set $k \leftarrow k + 1$ and return to Step 3.

4. Numerical Experiment Results

We tested our procedure on the example of Section 2. There are $2^6 = 64$ corner points coming from a hypercube $\mathcal{U}_p$ of volume $p = 0.99$ and $k = 74$ total design points in Phase I. The first-stage sample size $n_0 = 5000$. The confidence level $1 - \alpha = 0.9$ and the target levels β and γ for relative error are both 0.05.

Our figure of merit is root average relative mean squared error (RARMSE): for security h, $\mathrm{RARMSE}(h) = \sqrt{\int_{\mathcal{X}} \mathrm{E}[(\hat{\mathsf{Y}}_h(\boldsymbol{x})/y_h(\boldsymbol{x}) - 1)^2]\, dF_{\boldsymbol{X}}(\boldsymbol{x})}$. To analyze the performance of our procedure, we sample $\boldsymbol{X}_1, \boldsymbol{X}_2, \ldots \boldsymbol{X}_K$ independently from the distribution of tomorrow's scenario, and use this sample to approximate the integral. In these $K = 1000$ scenarios, we compare the metamodels' predictions to very accurate estimates of the true security prices obtained from another simulation experiment. We approximate the expectation by running $m = 30$ macro-replications of our procedure and estimate RARMSE(h) by

$$\frac{1}{mK} \sum_{i=1}^{K} \sum_{j=1}^{m} \left(\frac{\hat{\mathsf{Y}}_h^{(j)}(\boldsymbol{X}_i)}{y_h(\boldsymbol{X}_i)} - 1 \right)^2,$$

where $\hat{\mathsf{Y}}_h^{(j)}$ is the metamodel of the price of security h in the jth macro-replication.

Figure 2 contains histograms of the RARMSE of the 75 metamodels produced by our procedure. One histogram is obtained when the set $\mathbb{H}$ of securities used in cross-validation contains all 75 securities, and the other when $\mathbb{H}$ contains only five securities. In the latter case, the representative in $\mathbb{H}$ of each of the five classes of options described in Section 2 is the option that has the longest maturity and is deepest out of the money, which therefore has the highest ratio of payoff variance to price. The figure shows that using only five securities in cross-validation increases RARMSE only slightly; the metamodels are still quite accurate, with the biggest RARMSE around 0.72%.

The advantage of using a small set $\mathbb{H}$ for cross-validation is reduced computational cost. To study this, we simulated sample paths with a time step of two days. This is not necessary when the equity indices follow geometric Brownian motion, but some other models do require simulation with small time steps. Implemented in MATLAB 7.6 and run on a computer with a 2.4GHz CPU and 3.4GB memory under 32-bit Windows XP, the procedure took 4.2 hours when including all 75 securities in cross-validation, and 2.2 hours when including only five securities. In practice, portfolios may consist of hundreds of securities, in which case more computing power is required to run our procedure in one night. Our procedure is easy to parallelize: simulation of different sample paths and building metamodels for different securities can be allocated to separate processors.

Acknowledgements

This article is based upon work supported by the National Science Foundation under Grant No. CMMI-0900354. The authors are grateful for discussions with Michael Sotiropoulos.

References

[1] Ankenman, B., Nelson, B. L., Staum, J., 2008. Stochastic kriging for simulation metamodeling. Operations Research, forthcoming.

[2] Cario, M. C., Nelson, B. L., 1997. Modeling and generating random vectors with arbitrary marginal distributions and correlation matrix. Technical report, Northwestern University.

[3] Cario, M. C., Nelson, B. L., 1998. Numerical methods for fitting and simulating autoregressive-to-anything processes. INFORMS Journal on Computing 10, 72–81.

[4] Frye, J., November 1998. Monte Carlo by day. Risk 11, 66–71.

[5] Geisser, S., 1993. Predictive Inference: An Introduction. Chapman and Hall, New York.

[6] Kleijnen, J. P. C., 2008. Design and Analysis of Simulation Experiments. Springer-Verlag, New York.

[7] Law, A. M., 2007. Simulation Modeling and Analysis, 4th Edition. McGraw-Hill, Boston, pp. 500–503.

[8] McNeil, A. J., Frey, R., Embrechts, P., 2005. Quantitative Risk Management: Concepts, Techniques, Tools. Princeton University Press, Princeton, New Jersey.

[9] Stein, C., 1945. A two-sample test for a linear hypothesis whose power is independent of the variance. Annals of Mathematical Statistics 16, 243–258.

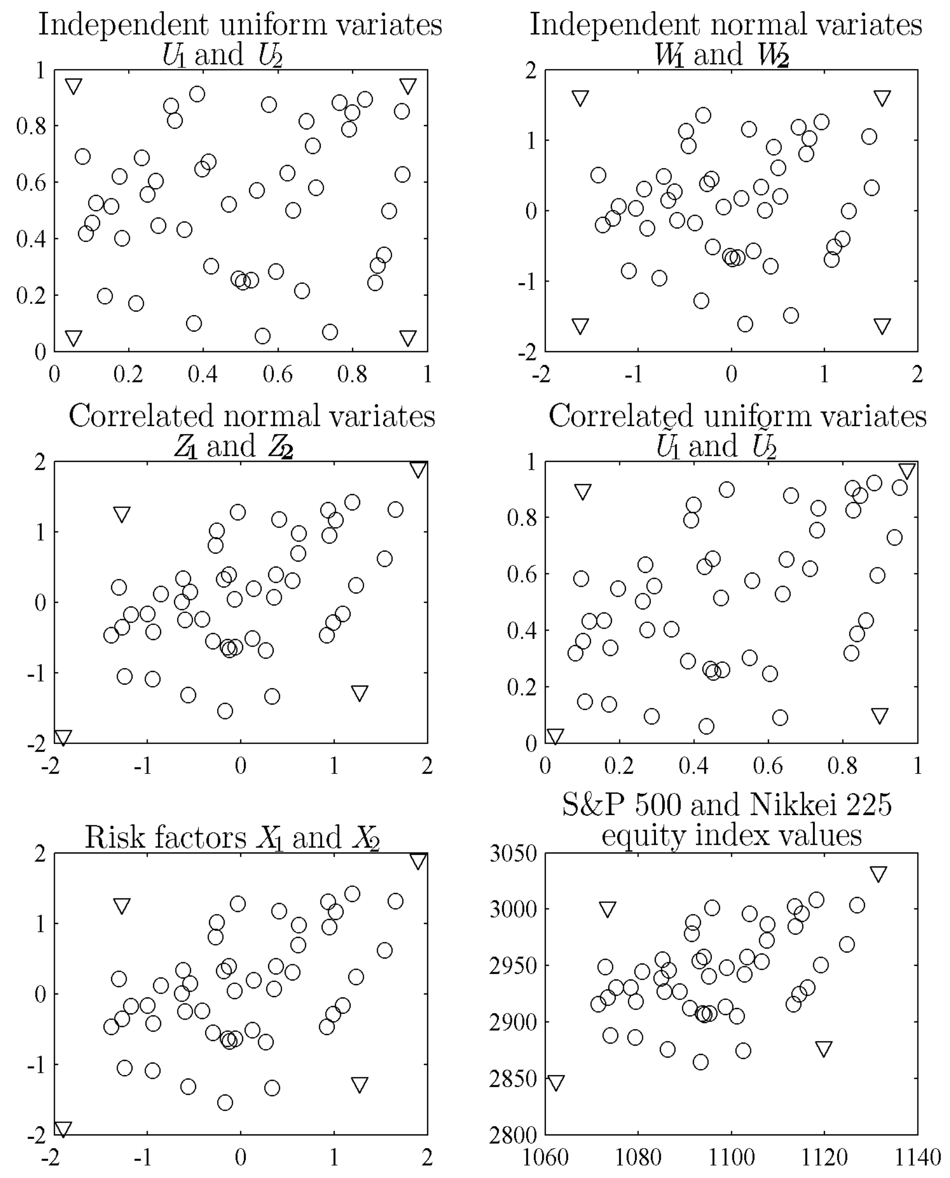

Figure 1: Construction of Phase I design points in a two-dimensional example: $\triangledown$ indicates corner points, and $\circ$ indicates points generated by quasi-Monte Carlo.

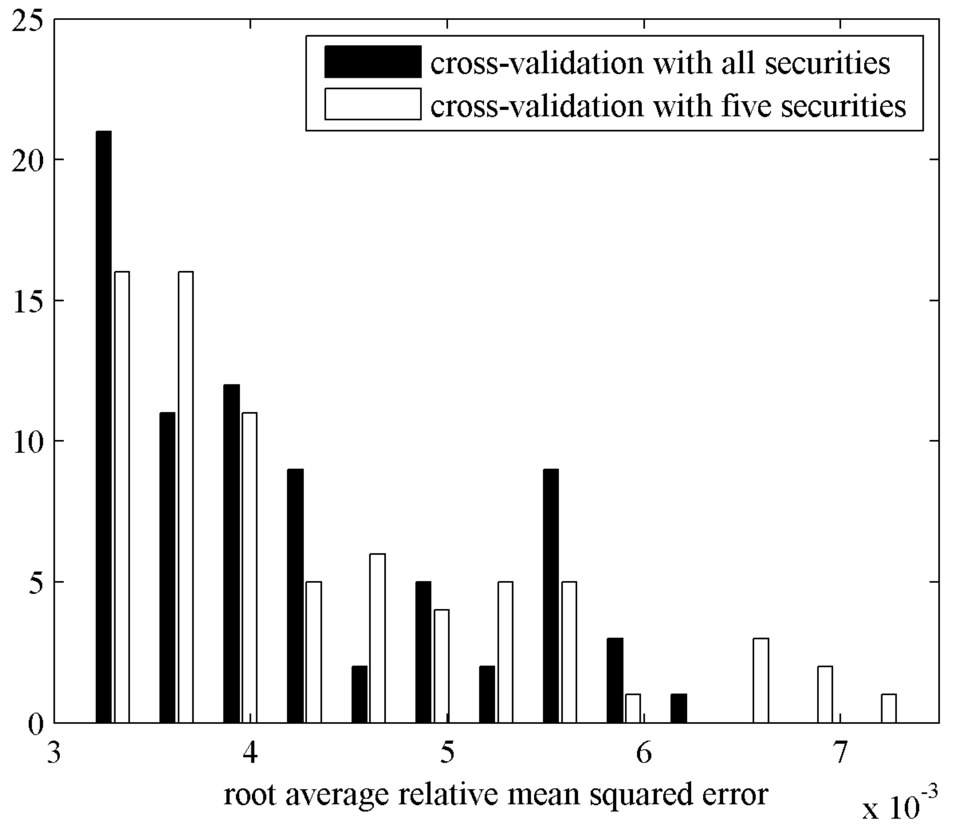

Figure 2: Histogram of estimated root average relative mean squared error of metamodels.