Proceedings of the 2008 Winter Simulation Conference
S. J. Mason, R. R. Hill, L. Mönch, O. Rose, T. Jefferson, J. W. Fowler eds.

RESPONSE SURFACE METHODOLOGY FOR SIMULATING HEDGING AND TRADING STRATEGIES

R. Evren Baysal
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

Suppose that one wishes to evaluate the distribution of profit and loss (P&L) resulting from a dynamic trading strategy. A straightforward method is to simulate thousands of paths (i.e., time series) of relevant financial variables and to track the resulting P&L at every time at which the trading strategy rebalances its portfolio. In many cases, this requires numerical computation of portfolio weights at every rebalancing time on every path, for example, by a nested simulation performed conditional on market conditions at that time on that path. Such a two-level simulation could involve many millions of simulations to compute portfolio weights, and thus be too computationally expensive to attain high accuracy. We show that response surface methodology enables a more efficient simulation procedure: in particular, it is possible to do far fewer simulations by using kriging to model portfolio weights as a function of underlying financial variables.

1 INTRODUCTION

Suppose that one wishes to evaluate the distribution of profit and loss (P&L) resulting from a dynamic trading strategy. For example, one may be interested in a strategy that aims to make a profit by trading stocks and options, or in a hedging strategy that is intended to reduce the risk associated with selling over-the-counter derivative securities. One would like to know the distribution of P&L that results from a strategy so as to decide whether or not to adopt the strategy, or which of several rival strategies to implement.

Monte Carlo simulation is an appealing tool to use for this purpose: given a stochastic model of the market's dynamic behavior, one can simulate multiple scenarios for the market's behavior, determine the strategy's P&L in each scenario, and consider the resulting empirical distribution of P&L over all scenarios. Because of our interest in dynamic trading strategies, each scenario takes the form of a time series, a sequence of snapshots of the market's state at successive moments in time, which we will call a *path*. Given a path, we need to be able to compute the strategy's P&L along the path. This may require computation of the portfolio weights that the strategy chooses at each time and the values of financial securities at those times.

For some problems, those computations are easy, in which case the simulation approach is relatively straightforward (§4.1). The motivation for this article is the case in which substantial computational effort is required to approximate the portfolio weights or security values, and the accuracy of the approximations is proportional to the computational effort expended. This happens frequently in practice. For example, it happens when the portfolio weights are hedge ratios for which no formula exists. We will focus on the use of simulation to compute portfolio weights and security values, but the framework we propose in this article is applicable even if they are computed by other methods such as trees or numerical solution of partial differential equations.

The absence of formulae for portfolio weights and security values leads to a two-level simulation procedure (§4.2) which is used in practice: at the *outer* level, simulate paths; at the *inner* level, use simulation to compute portfolio weights and security values at every time step on every path. The inner-level simulation typically involves simulating many terminal payoffs of relevant securities. We refer to the Monte Carlo samples used to generate these payoffs as *pricing replications*. It may be necessary to use tens of thousands of pricing replications at every time step on every path, and thousands of paths, to attain high accuracy. As there are often dozens to thousands of times at which the portfolio is rebalanced, the total number of pricing replications required can be many millions or even billions. Thus, the disadvantage of this method, which we call "full simulation," is that it can be very slow. Our goal is to create a more efficient simulation procedure.

The central insight is that the full simulation method described above does too much work in pricing replications. For example, suppose that the problem involves hedging

978-1-4244-2708-6/08/$25.00 ©2008 IEEE

an option on a single stock using that stock and a riskless money market account (as in §3). When the full simulation method needs to compute portfolio weights at time step 3 given a stock price of \$100, it does so using a full set of entirely new pricing replications, even if the procedure had previously computed portfolio weights on a different path in which the stock price was \$100.10 at time step 3. For many securities—although not for all securities, such as barrier options, whose value functions can be discontinuous—it seems that we should be able to use information from nearby paths to reduce the computational burden. The same insight underlies the use of regression Monte Carlo methods in pricing American options (see, e.g., Glasserman 2003, §8.6). Those methods use regression to approximate the option's value as a function of underlying state variables, such as the stock price in our example, for each potential exercise time separately. In our context, where there may be many rebalancing times, we expand on this idea by modeling security values and portfolio weights as functions of time and other state variables jointly.

We apply response surface methodology to each security value and portfolio weight that we need to know. Each of them is modeled as a function of time and state variables. We call a combination of time and state variables (such as stock price) a *point*. Response surface methodology in simulation works by performing simulations that estimate the response only at certain *design points*, and then attempting to infer the value of the response at other points. Our simulation procedure based on this idea (§4.3) first performs pricing replications at perhaps a few hundred design points, then constructs response surfaces, and finally uses the response surfaces to approximate portfolio weights and security values at thousands of points.

In this article, we show that response surface methodology can yield a more efficient simulation procedure for evaluating hedging and trading strategies. The response surface methodology we apply to this problem is kriging and the experimental design we use for choosing design points is based on a Latin hypercube, with some modifications due to the structure of this financial problem. These issues are discussed in more detail in §4. First we give a formal statement of the problem and present our notation in §2 and describe a simple example in §3. We present results of computer experiments in §5 and offer some conclusions about future research in §6.

2 PROBLEM FORMULATION

Let Π be a random variable representing the strategy's terminal P&L. We focus on learning the univariate distribution F_Π of terminal P&L, although our method is also applicable to studying the joint distribution of P&L at various times. We do not know F_Π or how to sample from it directly; we only know how to sample from the distribution of paths: the outer level of simulation. P&L is some function of path, but we do not know this function either. P&L is given by Equation (1) below, but to use it, we need to compute security values and portfolio weights, which are also not known as functions of the path. We do know how to estimate these quantities by an inner level of simulation. One way to think of the situation is that we can approximately sample from F_Π, by sampling a path and then doing high-precision inner-level simulation to approximate the P&L on this path. Lan, Nelson, and Staum (2007) have discussed a rigorous framework for understanding two-level simulation and shown how to get confidence intervals from it for functionals of F_Π. Another recent contribution to two-level simulation in financial risk analysis is Gordy and Juneja (2008). Here we will focus on point estimation of the mean and variance of F_Π, corresponding to evaluation of the strategy's average profitability and its risk.

We will use the following notation:

- T: the time horizon over which the strategy is to be evaluated.
- s: the number of times at which the portfolio can change.
- $t_0, t_1, \ldots, t_s$: times at which the portfolio is analyzed. A portfolio is set up at $t_0 = 0$, final P&L is measured at $t_s = T$, and $t_1, t_2, \ldots, t_{s-1}$ are the times at which the portfolio is rebalanced.
- $\mathbf{S}(t)$: the state vector of market risk factors at time t.
- $\mathbf{S}_i = \mathbf{S}(t_i)$: condensed notation for the state vector of market risk factors at time t_i.
- $\mathbf{V}_i$: the vector of values of the securities at time t_i.
- $\mathbf{f}$: the vector function of security values, so that the value of security l at time i is $V_{il} = f_l(t_i, \mathbf{S}_i)$.
- θ_i: the vector of portfolio weights at time t_i.
- $\mathbf{g}$: the vector function of portfolio weights, so that the number of shares of security l in the portfolio at time i is $\theta_{il} = g_l(t_i, \mathbf{S}_i)$.
- Π_i: the cumulative P&L up to time t_i.
- k: the number of paths simulated.
- m: the number of replications used in inner-level simulation at any point.
- A superscript indicates the realization of a random variable on a particular path: for example, $\mathbf{S}_i^j$ is the vector of market risk factors at time t_i on path j, and $\mathbf{S}^j$ is the jth sample path of the discrete-time vector Markov process $\mathbf{S}$.

The P&L that occurs over step i, that is, between times t_{i-1} and t_i, is the sum of the changes in value of the portfolio's holdings in each security. The number of shares of each security remains constant over this time step, so step i's contribution to P&L is $\theta_{i-1}^\top(\mathbf{V}_i - \mathbf{V}_{i-1})$. The cumulative

P&L up to time t_i is thus

$$\Pi_i = \sum_{i'=1}^{i} \theta_{i'-1}^\top (\mathbf{V}_{i'} - \mathbf{V}_{i'-1}). \tag{1}$$

In mathematical finance, it is standard to consider *self-financing* strategies, which have the property that the portfolio's value does not change when the portfolio is rebalanced:

$$\theta_{i-1}^\top \mathbf{V}_i = \theta_i^\top \mathbf{V}_i \tag{2}$$

for all $i = 1, 2, \ldots, s$. For self-financing strategies, P&L also equals the change in the portfolio's value across all steps to date, which is

$$\Pi_i = \theta_i^\top \mathbf{V}_i - \theta_0^\top \mathbf{V}_0. \tag{3}$$

To estimate the P&L of a strategy at times $t_0 = 0, t_1, \ldots, t_s = T$, we simulate a path of market risk factors $\mathbf{S}_0, \mathbf{S}_1, \ldots, \mathbf{S}_s$ at each of these times. We need to estimate security values $\mathbf{V}_i$ and portfolio weights θ_i at each time $i = 0, 1, \ldots, s$ to use Equation (1) to find P&L at each time. Section 3 describes some simplifications that are possible in a typical application of hedging a derivative security. We simulate k paths to get a sample $\Pi^1, \Pi^2, \ldots, \Pi^k$. Its sample average is an estimate of the mean terminal P&L, while its sample standard deviation is an estimate of the standard deviation of terminal P&L.

Estimation of security values and portfolio weights is founded on inner-level simulation. We assume that we are in the typical complete-markets, Markov-process setting in which each security's value at any time t is the conditional expectation of its payoff under the risk-neutral probability measure given the value of $\mathbf{S}(t)$ (see, e.g., Nielsen 1999, §5.8). For simplicity, we also assume that all securities have maturity $T = t_s$ and are path-independent. In this case, the inner-level simulation need only sample values of $\mathbf{S}(T) = \mathbf{S}_s$. Otherwise, the inner-level simulation must sample state vectors at the maturities of all securities. If securities are path-dependent, then the inner-level simulation must sample paths of the state vector observed at all relevant times.

Consider an inner-level simulation at some point $x = (t^*, \mathbf{S}^*)$ which is a combination of time t^* and state $\mathbf{S}^*$. The inner-level simulation samples *inner-level replications* $\mathbf{S}_s^1(x), \mathbf{S}_s^2(x), \ldots, \mathbf{S}_s^m(x)$ from the risk-neutral conditional distribution of $\mathbf{S}_s = \mathbf{S}(T)$ given $\mathbf{S}(t^*) = \mathbf{S}^*$. The terminal payoff function $g(T, \cdot)$ of all securities at maturity is known. This allows us to estimate the security value $g_l(t^*, \mathbf{S}^*)$ at time t^* by $\sum_{j=1}^m g_l(T, \mathbf{S}_s^j(x))/m$ for each security l (Glasserman 2003, §1.2).

Portfolio weights in hedging strategies often arise as sensitivities of security values to risk factors. See Glasserman (2003, Ch. 7) on Monte Carlo methods for estimating sensitivities such as delta. In general, the method we describe can be applied if the portfolio weights can be computed by some method founded on simulation: for example, if the portfolio weights arise from a portfolio optimization problem at each point, then they can be estimated by optimization-via-simulation, and the method applies.

We give a simple, concrete example of a hedging strategy in this framework in §3. In §4, we describe three different methods for estimating security values $\mathbf{V}_i$ and portfolio weights θ_i at each time $i = 0, 1, \ldots, s$.

3 DELTA-HEDGING A EUROPEAN PUT OPTION

Our computational experiments feature the example of delta-hedging a European put option on a stock under the Black-Scholes model (see, e.g., Nielsen 1999, Ch. 6 for further background). Under the Black-Scholes model, the stock price S is geometric Brownian motion. The securities in the hedging portfolio include a riskless money-market account and the underlying stock. Their values at time t_i are respectively $V_{i1} = e^{rt_i}$ where r is the interest rate and $V_{i2} = S_i$. The put option (denoted as security 0) is hedged from the time it is sold until its maturity, when it pays off $V_{s0} = f_0(T, S_s) = \max\{K - S_s, 0\}$, where K is the strike price.

For $i = 0, 1, \ldots, s-1$, the number θ_{i2} of shares of stock to hold at time t_i is set equal to the negative of the first-order sensitivity $\partial V_{i0}/\partial S_i$ of the put option value with respect to the stock price at that point in time, which is called the *delta* of the option. At time $t_s = T$, the option matures and the hedge is unwound, so $\theta_{s2} = 0$. The number of shares in the money-market account is set so that the hedging strategy is self-financing:

$$\theta_{i1} = \theta_{i-1,1} + (\theta_{i-1,2} - \theta_{i2}) V_{i2}/V_{i1} \tag{4}$$

for $i = 1, 2, \ldots, s$, based on Equation (2). The initial number of shares in the money-market account is set so that the value of the initial portfolio is zero:

$$\theta_{01} = -(V_{00} + \theta_{02} V_{02})/V_{01}. \tag{5}$$

At any time t_i, $i = 0, 1, \ldots, s$, there is $\theta_{i0} = 1$ share of the put option. Because the total initial portfolio value is zero, the final P&L according to Equation (3) is

$$\Pi_s = V_{s0} + \theta_{s1} V_{s1}. \tag{6}$$

Some simplifications of the general framework for P&L presented in §2 apply in many examples, including this one. Suppose that we have a self-financing strategy for hedging a derivative security until maturity, the derivative security's initial value is known, and we are only interested in terminal P&L. Then we can use Equation (3) and we

need only compute portfolio weights, not security values. The reason is that underlying security values (e.g., of the stock and money market account) are known functions of the path, and the value of the derivative security (e.g., put option) is known at the times that we need it. We do not need to know the derivative security value at intermediate times, its initial value is given, and its terminal value is always a known function of the path, such as $\max\{K-S_s,0\}$ in the example of the European put option. Even though only the initial and terminal portfolio weight vectors θ_0 and θ_s appear in Equation (3), we do need to know the portfolio weight vectors at intermediate times too. They are required to enforce the self-financing condition. For our put option example, we can use Equation (4) recursively with Equation (5) as its initial condition, use $V_{i1} = e^{rt_i}$ and $V_{i2} = S_i$, plug into Equation (6), and get

$$\Pi_s = V_{s0} + \sum_{i=1}^{s} (\theta_{i-1,2} - \theta_{i2}) S_i e^{r(T-t_i)} - (V_{00} + \theta_{02} S_0) e^{rT}.$$

This shows that we need to know delta at all steps $i = 0,1,\ldots,s-1$, to get θ_{i2}, but the only option values we need are V_{00}, the given initial value, and V_{s0}, which is the payoff $\max\{K-S_s,0\}$. We use pathwise derivative estimation (Glasserman 2003, §7.2) for delta:

$$\hat{\theta}_{i2} = e^{-r(T-t_i)} \frac{1}{m} \sum_{j=1}^{m} \mathbf{1}\{S_s^j < K\} \frac{S_s^j}{S_i^j}.$$

In the particular example we use in our computational experiments, the stock price follows a geometric Brownian motion with initial value $S_0 = \$100$, drift $\mu = 8\%$, and volatility $\sigma = 15\%$. The put option has maturity of $T = 1$ years and strike price $K = \$110$. The interest rate on the money-market account is $r = 5\%$. There are $s = 60$ rebalancing times, and $t_i = iT/s$ for $i = 1,2,\ldots,s$.

4 METHODS

In this section, we describe three simulation procedures used in our experimental results. The procedures differ only in how they estimate security values and portfolio weights at times after t_0: by using a formula, by using a nested simulation at every time step on every path, or by using response surface modeling. We assume that security values and portfolio weights are known at t_0 even in procedures that do not rely on formulae for them. In practice, even if security values and portfolio weights at t_0 need to be estimated by simulation, they have usually been estimated to very high precision; it is only security values and portfolio weights in hypothetical scenarios at future times that we can not afford to estimate to high precision.

For similar reasons, we regard the P&L distribution produced by the formula-based procedure (§ 4.2) as the true P&L distribution. Even if formulae are not available, the decision-maker will estimate $\mathbf{f}(t_i,\mathbf{S}_i)$ and $\mathbf{g}(t_i,\mathbf{S}_i)$ to high precision when in state $\mathbf{S}_i$ at time t_i. The actual P&L arising on a path $\mathbf{S}_1,\mathbf{S}_2,\ldots,\mathbf{S}_s$ will be nearly the P&L assigned to it by the formula-based procedure, which is the limit of the P&L assigned to it by the full nested simulation procedure (§ 4.1) as the number m of inner-level replications goes to infinity.

4.1 Formula-Based

The example described in §3 is so simple that we actually do have formulae for the put option's value and delta as functions of time and stock price (see, e.g., Nielsen 1999, §6.6). This means that the security values and portfolio weights are known as functions of the path. By using these functions, we can avoid any inner-level simulation and eliminate all associated statistical error. In general, these functions are unknown, but using them in this simple example allows us to study the statistical error associated with inner-level simulation in one of the following two methods. The formula-based procedure for sampling P&L Π_i^j for every step $i = 1,2,\ldots,s$ and on paths $j = 1,2,\ldots,k$ is as follows.

- for $j = 1,2,\ldots,k$,
  - $\Pi_0^j = 0$
  - for $i = 1,2,\ldots,s$,
    - sample $\mathbf{S}_i^j$ from the conditional distribution of $\mathbf{S}_i$ given $\mathbf{S}_{i-1} = \mathbf{S}_{i-1}^j$
    - $\theta_i^j = \mathbf{g}(t_i,\mathbf{S}_i^j)$ and $\mathbf{V}_i^j = \mathbf{f}(t_i,\mathbf{S}_i^j)$
    - $\Pi_i^j = \Pi_{i-1}^j + (\theta_{i-1}^j)^\top (\mathbf{V}_i^j - \mathbf{V}_{i-1}^j)$
- next j

4.2 Full Nested Simulation

In the absence of formulae for these functions, we require estimates of security values and portfolio weights at each of s time steps on each of k paths. One way to get them is from fully nested simulation: at each of ks points, do an inner-level simulation to estimate security values and portfolio weights there. For simplicity, we suppose there are the same number m of inner-level replications at all points. The full nested simulation procedure is as follows.

- for $j = 1,2,\ldots,k$,
  - $\Pi_0^j = 0$
  - for $i = 1,2,\ldots,s$,
    - sample $\mathbf{S}_i^j$ from the conditional distribution of $\mathbf{S}_i$ given $\mathbf{S}_{i-1} = \mathbf{S}_{i-1}^j$

  - for $h = 1,2,\ldots,m$,
    - sample $\mathbf{S}_s^h(t_i,\mathbf{S}_i^j)$ from the conditional risk-neutral distribution of $\mathbf{S}_s$ given $\mathbf{S}_i = \mathbf{S}_i^j$
  - using this inner-level simulation, estimate θ_i^j by $\hat{\theta}_i^j$ and $\mathbf{V}_i^j$ by $\hat{\mathbf{V}}_i^j$
  - $\Pi_i^j = \Pi_{i-1}^j + (\hat{\theta}_{i-1}^j)^\top(\hat{\mathbf{V}}_i^j - \hat{\mathbf{V}}_{i-1}^j)$
- next j

4.3 Response Surface Modeling

Instead of performing inner-level simulation at each of the ks points at which we want to know security values and portfolio weights, this method reads estimates at each of those points from functions called *response surfaces*. These response surfaces are in turn computed as the result of inner-level simulation, which is performed only at certain *design points*. Frye (1998) and Shaw (1998) have also applied response surface modeling in financial risk analysis. In our experiments, we use a response surface methodology called *kriging*, which chooses the value of a response surface at any point that is not among the design points by interpolating among the values observed at design points. Figure 1 shows a response surface for the value of the put option constructed by kriging based on inner-level simulation, and the error due to kriging in estimating the response surface given by the Black-Scholes formula in the example of §3. On the kriging method and its use in the design and analysis of computer experiments see, e.g., Fang, Li, and Sudjianto (2006), Santner, Williams, and Notz (2003), and Stein (1999).

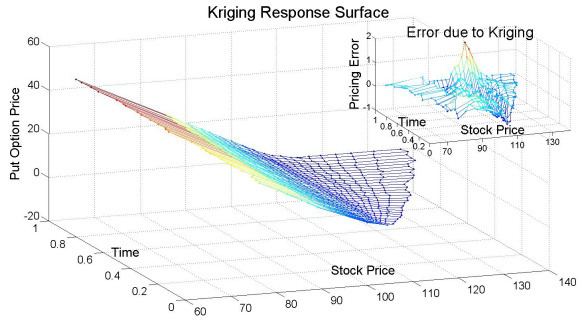

Figure 1: A response surface estimated by kriging, and its difference from the Black-Scholes formula, for the price of the put option in §3.

By choosing a set of $n \ll ks$ design points, the response surface modeling method can be faster than full nested simulation with the same numbers k of paths, s of time steps, and m of inner-level replications per point. This speed can come at the price of accuracy in assessing the distribution of P&L, because the response surface models of security values and portfolio weights at a point $(t_i,\mathbf{S}^*)$ tend not to be as accurate as estimates provided by inner-level simulation performed conditional on $\mathbf{S}_i = \mathbf{S}^*$; that is, interpolating estimates from other points is not as accurate as simulating at this point. The goal is to trade off a little bit of accuracy for a lot of speed to get a more efficient procedure.

Our response surface modeling procedure has three parts. First, choose a set $\{x_1,x_2,\ldots,x_n\}$ of design points and perform inner-level simulation at each of them to estimate the security values and portfolio weights there. Second, use kriging to build response surfaces for each component, separately, of the security value function $\mathbf{f}$ and the portfolio weight function $\mathbf{g}$. Third, compute P&L on each path by using the response surfaces to provide estimates of security values and portfolio weights.

The choice of experimental design (the set of design points) can have a very large impact on the fidelity of the response surface generated by kriging. In most applications of kriging, one knows in advance a finite region within which one wishes to estimate a response. However, in the financial examples we consider here, this is not so: for example, asset prices are typically unbounded. We solve this problem by first simulating all the paths we need, and then modeling responses over a finite region that contains all the simulated data. In the example of §3, this region is contained in the Cartesian product $[t_0,t_{s-1}] \times [S_{\min},S_{\max}]$ where $S_{\min} = \min\{S_i^j | j = 1,2,\ldots,k,\ i = 1,2,\ldots,s\}$ and $S_{\max} = \max\{S_i^j | j = 1,2,\ldots,k,\ i = 1,2,\ldots,s\}$ are the smallest and largest observed stock prices. The reason that $t_s = T$ is excluded from response surface modeling is twofold. We do not need a response surface model at maturity because security values then are payoffs, which are known functions, and likewise portfolio weights then do not need to be estimated. There is also a discontinuity at maturity in the put option's delta, and hence in g_2, the number of shares of stock to hold: $\lim_{S\uparrow K} g_2(T,S) = 1$ while $\lim_{S\downarrow K} g_2(T,S) = 0$. In general, response surface modeling can encounter severe problems when the true function is discontinuous.

Latin hypercube designs (see, e.g., Glasserman 2003, §4.4) have been reported to be more effective in kriging than uniform grid designs; our experiments confirmed this. Our experimental design for the example of §3, based on a Latin hypercube, is constructed as follows.

- Partition $[t_0,t_{s-1}]$ and $[S_{\min},S_{\max}]$ into d intervals of equal width.
- Sample one point uniformly within each interval.
- Randomly pair the time and stock price values to get d design points.
- Add the four points $(t_0,S_{\min})$, $(t_0,S_{\max})$, $(t_{s-1},S_{\min})$ and $(t_{s-1},S_{\max})$ to the design.

The last step is important: without the addition of the four corner points there can be a point (t_i, S_i^j) on some path that falls outside the convex hull of the Latin hypercube design. This causes problems because kriging, as an interpolation method, performs quite badly when it is used to extrapolate outside the convex hull of design points. An illustration of this experimental design appears in Figure 2.

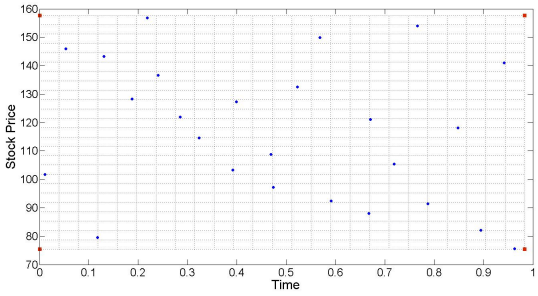

Figure 2: An experimental design for the example of §3 with 29 design points. The 4 red squares are corner points and the 25 blue dots form a Latin hypercube design.

We feed estimated security values and portfolio weights at the design points to kriging, which constructs response surfaces by interpolating among them. The interpolated value at a point x is a weighted average of the values observed at all design points. The weight a design point x_j receives depends on its distance to x; the basic idea behind kriging is that nearer design points are more highly correlated (in a loose sense) with the point of interest, and so get greater weight. The kriging model requires a choice of a correlation function that governs how weights diminish with distance. We used the exponential correlation function.

The response surface modeling procedure is as follows.

- for $j = 1,2,\ldots,k$,
  - for $i = 1,2,\ldots,s$,
    - sample $\mathbf{S}_i^j$ from the conditional distribution of $\mathbf{S}_i$ given $\mathbf{S}_{i-1} = \mathbf{S}_{i-1}^j$
  - next i
- lay down a design of points $x_1 = (t_1^*, \mathbf{S}_1^*)$, $x_2 = (t_2^*, \mathbf{S}_2^*)$, $\ldots$, $x_n = (t_n^*, \mathbf{S}_n^*)$
- for $j = 1,2,\ldots,n$,
  - for $h = 1,2,\ldots,m$,
    - sample $\mathbf{S}_s^h(x_j)$ from the conditonal risk-neutral distribution of $\mathbf{S}_s$ given $\mathbf{S}(t_j^*) = \mathbf{S}_j^*$
  - using this inner-level simulation, estimate $\theta(x_j)$ by $\hat{\theta}(x_j)$ and $\mathbf{V}(x_j)$ by $\hat{\mathbf{V}}(x_j)$
- next j
- for each security l,
  - build response surfaces $\hat{f}_l$ for f_l using $\hat{V}_l(x_1), \hat{V}_l(x_2), \ldots, \hat{V}_l(x_n)$ and $\hat{g}_l$ for g_l using $\hat{\theta}_l(x_1), \hat{\theta}_l(x_2), \ldots, \hat{\theta}_l(x_n)$
- next l
- for $j = 1,2,\ldots,k$,
  - $\Pi_0^j = 0$
  - for $i = 1,2,\ldots,s$,
    - $\Pi_i^j = \Pi_{i-1}^j + (\hat{\mathbf{g}}(t_i, \mathbf{S}_{i-1}^j)^\top (\hat{\mathbf{f}}(t_i, \mathbf{S}_i^j) - \hat{\mathbf{f}}(t_i, \mathbf{S}_{i-1}^j))$
- next j

5 EXPERIMENTS

We performed experiments in MATLAB, implementing kriging with the `DACE` toolbox and generating Latin hypercube designs with the `lhsdesign` function. We used $k = 1,000$ paths and $m = 1,000$ inner-level replications in the example described in §3. Our experimental designs had $n = 104$ or $n = 404$ design points. This means that the full nested simulation procedure required 60 million inner-level replications while the response surface modeling procedure required only 104,000 or 404,000. Of course, performing kriging takes time too. However, the relative effort required for inner-level replications, sampling paths, and performing kriging varies with the problem and the computing platform. Because we expect that the time spent on inner-level replications will dominate unless m is small and the stochastic model of the market is simple, we ignore the other computational costs.

To assess the accuracy of these simulation procedures, we ran 100 macro-replications, each an independent run of the entire procedure, involving independently generated paths and design points. However, within a single macro-replication, all procedures used the same paths. This produced 100 estimates of the mean P&L and the standard deviation of P&L at each time step, for each procedure: that is, each of 100 runs produces an estimate of $\mathbf{E}[\Pi_i]$ and an estimate of $\sqrt{\mathbf{Var}[\Pi_i]}$ for each time step $i = 0,1,\ldots,s$. We depict the average and the sample root mean squared error (RMSE) of these 100 estimates in Figures 3–6. The error used in computing RMSE is the difference between one macro-replication's estimate of $\mathbf{E}[\Pi_i]$ or $\sqrt{\mathbf{Var}[\Pi_i]}$ and our best estimate of $\mathbf{E}[\Pi_i]$ or $\sqrt{\mathbf{Var}[\Pi_i]}$, which we get by combining all 100,000 replications generated by the formula-based procedure.

It is a feature of the financial example, not a problem with the simulation procedure, that the mean P&L $\mathbf{E}[\Pi_i]$ is decreasing in the time step i for the formula-based procedure in Figure 3. Because of the option's convexity, the discrete-time hedging strategy that is delta-neutral at the beginning of each time step holds too little stock, on average, between portfolio rebalancing times. Because the stock's expected return exceeds the interest rate, holding too little stock gives the strategy a negative expected return.

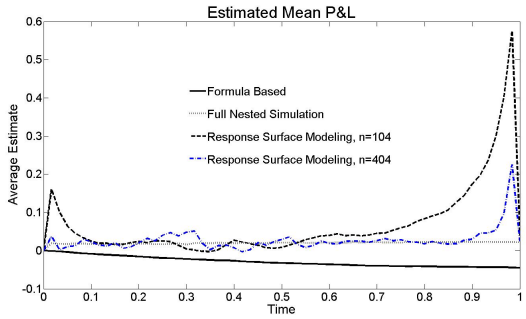

Figure 3: The average, over 100 macro-replications, of four simulation methods' estimates of mean P&L at each time step.

In Figure 4, even the formula-based procedure yields an estimate with positive RMSE because of outer-level sampling error: each macro-replication has a different set of $k = 1,000$ paths.

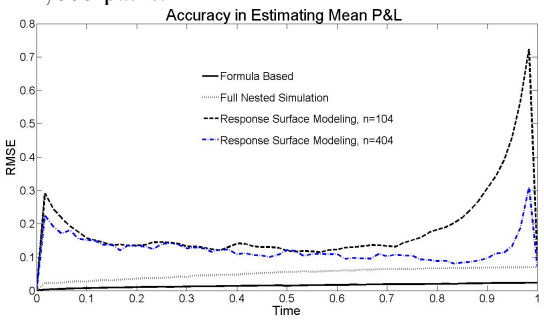

Figure 4: The sample root mean squared error, over 100 macro-replications, of four simulation methods' estimates of mean P&L at each time step.

Figures 3 and 4 show that response surface modeling can introduce substantial bias and variance into estimating mean P&L, but the problem is ameliorated by using a moderate number of design points. With $n = 104$ design points, kriging tends to generate a response surface with poor fidelity. Particularly noticeable is the bias just before maturity, where the put option's price and delta are badly behaved functions, as explained in §3. Increasing the number of design points to $n = 404$ greatly reduces this bias. With 404 design points, response surface modeling estimates mean P&L with a fair degree of accuracy, compared to the standard deviation of P&L portrayed in Figure 5. The bias and variance introduced by response surface modeling are much greater at time t_{s-1} than at time $t_s = T$ and at time t_1 than at time $t_0 = 0$ (where P&L is zero by construction). This happens because the simulation procedure relies on the response surface for the put option price to produce P&L at times t_{s-1} and t_1 but not at times t_s and t_0.

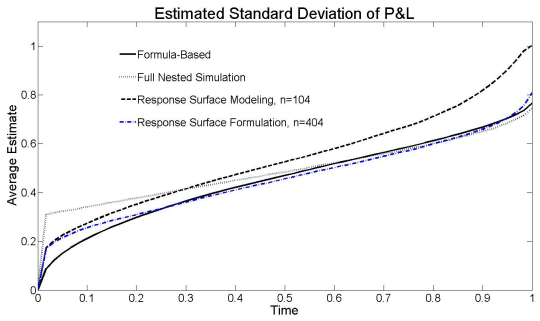

Figure 5: The average, over 100 macro-replications, of four simulation methods' estimates of the standard deviation of P&L at each time step.

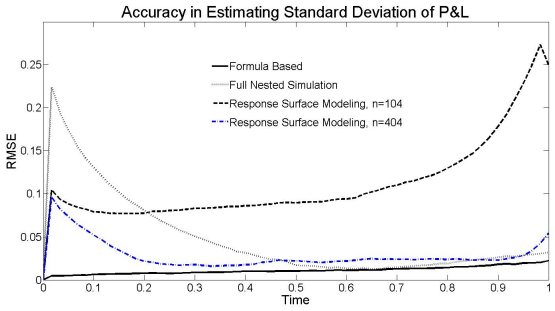

Figure 6: The sample root mean squared error, over 100 macro-replications, of four simulation methods' estimates of the standard deviation of P&L at each time step.

Figures 5 and 6 show that response surface modeling with $n = 404$ design points performs similarly to the other procedures in estimating the standard deviation of P&L, except at early times, when it is not as accurate as the formula-based procedure, but more accurate than full nested simulation. Again, the jump from time t_0 to time t_1 in RMSE of estimates from methods other than the formula-based method is explained by inner-level sampling error in estimating the put option value function $f_0(t_1, \cdot)$.

Next we focus on the distribution of terminal P&L generated by each of the simulation methods. We pool the terminal P&L values on $k = 1,000$ paths in each of 100 macro-replications to produce a picture of the probability density of terminal P&L in Figure 7. Table 1 summarizes the results in Figures 3–6 for P&L at time $T = 1$ only. It also shows the standard errors in estimating these quantities with 100 macro-replications. The standard error in estimating RMSE is calculated by the delta method (see, e.g., Asmussen and Glynn 2007, §III.3). From the table we see that the response surface modeling procedure with $n = 404$ design points is quite accurate in estimating the mean and standard deviation of terminal P&L, compared to

full nested simulation, but it is on the order of 100 times faster.

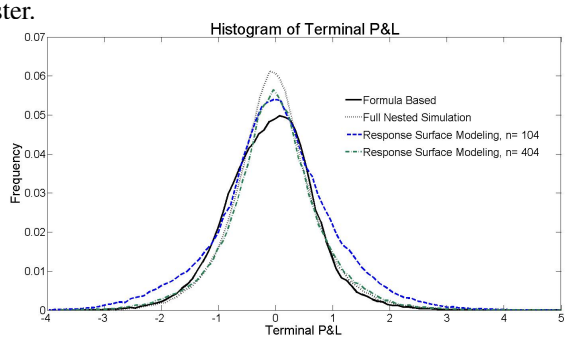

Figure 7: Approximate probability density functions of terminal P&L generated by four simulation methods, based on $100,000$ samples of terminal P&L.

Table 1: The average, standard error (SE) and the root mean square error (RMSE) statistics of the mean estimates of terminal P&L.

	Mean P&L		Std. Dev. of P&L	
Method	Average	RMSE	Average	RMSE
Formula	-0.045	0.023	0.767	0.022
Based	*(0.002)*	*(0.002)*	*(0.002)*	*(0.001)*
Full Nested	0.022	0.070	0.745	0.032
Simulation	*(0.002)*	*(0.002)*	*(0.002)*	*(0.002)*
RSM, n=104	0.040	0.096	1.002	0.249
	(0.004)	*(0.004)*	*(0.008)*	*(0.009)*
RSM, n=404	0.022	0.072	0.807	0.054
	(0.003)	*(0.003)*	*(0.004)*	*(0.003)*

6 FUTURE RESEARCH

We have found, in exploring one example, that using kriging to create response surfaces for the price and delta of a put option enables fast, accurate estimation of the mean and standard deviation of the terminal P&L of a hedging strategy for the put option. This suggests that kriging can be an effective tool for reducing the computational cost of nested simulations of hedging and trading strategies. However, the method's performance needs to be investigated in other examples, especially examples in which response surfaces are built over a higher-dimensional space. Kriging in higher dimensions is more challenging, and the example we used here has only two dimensions, time and stock price.

There are also possibilities for improving the performance of kriging in this setting. It is well-known in kriging that experimental design has a major impact on performance, and we found the same in our example, having tried other designs in computational experiments not reported here. One could look for designs better than the one illustrated in Figure 2. Kriging is based on an assumption that the response surface has a location-invariant correlation structure: roughly speaking, the values of the response surface at locations separated by the same distance have the same degree of similarity regardless of where you look. Figure 1 suggests that this assumption is untrue in our example of a put option: for example, the put option's values at$(0.8, 120)$ and $(0.8, 130)$ are very similar (nearly zero), but the option values at $(0.8, 80)$ and $(0.8, 90)$ are quite different (by about \$10). A transformation of the coordinate system from time and stock price to some other coordinate system might make kriging's spatial correlation modeling assumptions more nearly true in the transformed space and thus improve the performance of kriging. We used a basic version of kriging, in which the response surface is formed simply by interpolating among observed values. There is also a theory of kriging that incorporates the modeling of trends in the response surface, as in regression. For example, one might model the response surface as a linear function of the spatial variables plus local deviations from this trend: then the spatial correlation model applies only to the local deviations. In the literature on design and analysis of computer experiments, trend modeling is frequently reported not to work well in practice. However, because we have many more design points in our examples than are typical in that literature, it is possible that trend modeling may improve the performance of kriging here.

Kriging was developed for analyzing the results of deterministic experiments, for example, physical experiments in geology or deterministic computer experiments such as finite-element codes: if these experiments were to be repeated, they would yield the same results, at least roughly. Kriging interpolates between the values observed at design points because it assumes that these values are the truth. This is not so in our framework, in which the values observed at design points include inner-level sampling error. Ankenman, Nelson, and Staum (2008) describe a stochastic kriging procedure which takes account of this uncertainty when constructing the response surface. The resulting response surface may work better in our application, and the associated analysis may help in constructing better experimental designs.

ACKNOWLEDGMENTS

This material is based upon work supported by the National Science Foundation under Grant No. DMI-0555485. We are grateful for discussions with Tom Santner and Michael Sotiropoulos. The views expressed are those of the authors, who are responsible for any errors.

REFERENCES

Ankenman, B., Nelson, B. L., and J. Staum. 2008. Stochastic kriging for simulation metamodeling. Working Paper 08-01, Dept. of IEMS, Northwestern University.

Asmussen, S., and P. W. Glynn. 2007. *Stochastic Simulation*. New York: Springer-Verlag.

Fang, K. T., R. Li, and A. Sudjianto. 2006. *Design and Modeling for Computer Experiments*. Boca Raton, Florida: Chapman & Hall/CRC.

Frye, J. 1998. Monte Carlo by day. *Risk Magazine*, Nov. 1998, 66–71.

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

Gordy, M. B., and S. Juneja. 2008. Nested simulation in portfolio risk measurement. Working paper 2008-21, Finance and Economics Discussion Series, Federal Reserve Board.

Lan, H., B. L. Nelson, and J. Staum. 2007. Two-level simulations for risk management. In *Proceedings of the 2007 INFORMS Simulation Society Research Workshop*, ed. S. Chick, C.-H. Chen, S. Henderson, and E. Yücesan, 102–107. Fontainebleau, France: INSEAD.

Nielsen, L. T. 1999. *Pricing and Hedging of Derivative Securities*. New York: Oxford University Press.

Santner, T. J., B. J. Williams, and W. I. Notz. 2003. *The Design and Analysis of Computer Experiments*. New York: Springer-Verlag.

Shaw, J. 1998. Beyond VAR and stress testing. In *Monte Carlo: Methodologies and Applications for Pricing and Risk Management*, ed. B. Dupire, 231-244. London: Risk Books.

Stein, M. L. 1999. *Interpolation of Spatial Data: Some Theory for Kriging*. New York: Springer-Verlag.

AUTHOR BIOGRAPHIES

R. EVREN BAYSAL is a Ph. D. student in the Department of Industrial Engineering and Management Sciences at Northwestern University. His e-mail and web addresses are <e-baysal@northwestern.edu> and <users.iems.northwestern.edu/~rebaysal/>.

BARRY L. NELSON is the Charles Deering McCormick Professor of Industrial Engineering and Management Sciences at Northwestern University. His research centers on the design and analysis of computer simulation experiments on models of stochastic systems, and he is Editor in Chief of *Naval Research Logistics*. Nelson has held many positions for the Winter Simulation Conference, including Program Chair and member of the Board of Directors. His e-mail and web addresses are <nelsonb@northwestern.edu> and <users.iems.northwestern.edu/~nelsonb/>.

JEREMY STAUM is Associate Professor of Industrial Engineering and Management Sciences at Northwestern University. His research interests include risk management and simulation in financial engineering. Staum is Associate Editor of *ACM Transactions on Modeling and Computer Simulation*, *Naval Research Logistics*, and *Operations Research*, and was Risk Analysis track coordinator at the 2007 Winter Simulation Conference. His e-mail and web addresses are <j-staum@northwestern.edu> and <users.iems.northwestern.edu/~staum/>.