Sharpening Comparisons Via Gaussian Copulas and Semidefinite Programming

SHANE G. HENDERSON, Cornell University
SAMUEL M. T. EHRlichMAN, Jane Street Capital

A common problem in operations research involves comparing two system designs through simulation of both systems. The comparison can often be made more accurate through careful control (coupling) of the random numbers that are used in simulating each system, with common random numbers being the standard example. We describe a new approach for coupling the random-number inputs to two systems that involves generating realizations of a Gaussian random vector and then transforming the Gaussian random vector into the desired random-number inputs. We use nonlinear semidefinite programming to select the correlation matrix of the Gaussian random vector, with the goal of sharpening the comparison.

Categories and Subject Descriptors: G.3 [Probability and Statistics]: experimental design; random number generation; I.6.0 [Simulation and Modeling]: General

General Terms: Algorithms, Performance, Theory

Additional Key Words and Phrases: Variance reduction, Selection of the best

ACM Reference Format:
DOI = 10.1145/0000000.0000000 http://doi.acm.org/10.1145/0000000.0000000

1. INTRODUCTION
A ubiquitous problem in simulation analysis is “Which of the random variables \(X\) and \(Y\) has the greater mean?” For example, this problem arises when one is comparing two system designs, with \(X\) and \(Y\) representing some output associated with the two systems, and the selection criterion is based on the means of \(X\) and \(Y\). It also arises in certain search algorithms for simulation optimization, where an incumbent solution is compared to a trial solution. In that setting, \(X\) and \(Y\) denote random variables observed from the incumbent and trial solutions. The problem is difficult because we do not assume the availability of the distribution functions of \(X\) and \(Y\). Rather, we assume that we have simulation models from which instances of \(X\) and \(Y\) can be generated.

To answer the question of which random variable has the greater mean, the standard approach is to generate sample means

\[
\bar{X}_n = \frac{1}{n} \sum_{i=1}^{n} X_i \quad \text{and} \quad \bar{Y}_n = \frac{1}{n} \sum_{i=1}^{n} Y_i
\]
as estimators of $EX$ and $EY$ respectively, and declare, rightly or wrongly, $X$ to have greater mean if $\bar{X}_n > \bar{Y}_n$. Here we are implicitly assuming that $\bar{X}_n$ and $\bar{Y}_n$ can be compared directly which, in the language of probability theory, means that they are constructed on the same probability space.

Assuming finite variances of $X$ and $Y$, a natural measure related to the accuracy of this procedure is

$$\text{var}(\bar{X}_n - \bar{Y}_n) = \frac{\text{var}(X_1 - Y_1)}{n},$$

assuming that the pairs $((X_1,Y_1),(X_2,Y_2),\ldots,(X_n,Y_n))$ are i.i.d. Importantly, we have made no assumption about the joint distribution of $X$ and $Y$.

In this paper, we develop a method for carefully selecting this joint distribution in order to minimize $\text{var}(X - Y)$, thereby sharpening comparisons between two systems. The method involves a preliminary numerical search over a class of joint distributions, followed by a “production run” to perform the actual comparison using the best joint distribution found. The search is over an appropriately dimensioned Gaussian copula under marginal constraints that ensure that $X$ and $Y$ each have the correct marginal distributions. This particular class of copulas is convenient because it is finitely parameterized through the covariance matrix, and it is possible to provide a procedure that seeks a covariance matrix that minimizes (locally) the variance of $X - Y$. We do not claim that this class of joint distributions is “best” in any sense, but it is particularly convenient and tractable.

The idea of seeking a joint distribution that reduces the variance of $X_1 - Y_1$ is certainly not new. In the case where the distribution functions of $X$ and $Y$ are known, the well-known Hoeffding bounds [Whitt 1976, for example] provide the minimum possible variance of $X - Y$ by generating $X$ and $Y$ using inversion from a common uniform random variable on $(0,1)$. In our setting the simulation algorithms for generating instances of $X$ and $Y$ may be very complicated. Therefore, we do not have access to these distribution functions and inversion is not possible. The earliest reference we have seen to the idea of reducing the variance of $X - Y$ through a carefully constructed joint distribution without knowledge of the distribution functions comes from Wright and Ramsay Jr. [1979] who mention a version of the idea as an aside and, we believe incorrectly, ascribe the idea to Hammersley and Handscomb [1964]. The idea of seeking an appropriate joint distribution is closely related, or central, to many subsequent studies. As examples we mention Schmeiser and Kachitvichyanukul [1986] who build dependence without using inversion, Devroye [1990] who builds a joint distribution between two random vectors that attempts to maximize the number of identical components, and Glasserman and Yao [1992; 2004] who attempt to characterize when “common random numbers” (CRN) is the optimal joint distribution amongst a restricted class of joint distributions.

A similar philosophy to that explored here is followed in the study of variance reduction through antithetic sampling; see, e.g., [Fishman 1996, Section 4.4]. In that setting, optimal antithetic sampling plans that are difficult to implement in practice exist, and the theory motivates a number of implementable, yet suboptimal, sampling plans.

In work closely related to ours, Imai and Tan [2007] search for an orthogonal transformation of a set of independent Gaussian random variables that minimizes a quantity related to the effective dimension in quasi-Monte Carlo integration problems from finance. The study of how to design joint distributions of two or more random variables is known, in general, as coupling theory, and excellent reviews of coupling theory can be found in Lindvall [1992] and Thorisson [2000].

This paper is an outgrowth of Ehrlichman and Henderson [2008].
The primary contributions of this paper are (a) to exhibit a class of joint distributions that includes as special cases the standard independent and common-random-number joint distributions, (b) to develop the optimization problem that minimizes the variance of $X_1 - Y_1$ over this class of joint distributions, (c) to clarify some of the properties of that optimization problem, such as when one might expect the objective function to be convex, and (d) to present several examples that clarify the key ideas.

As we will see, the optimization problem is multimodal in general, and so one cannot expect to find globally optimal solutions. Even if one could guarantee a globally optimal solution, the resulting joint distribution is from a restricted class of joint distributions, and since this class is restricted, we cannot guarantee that the variance thus obtained is minimized over all possible joint distributions. Accordingly, we adopt the modest goal of attempting to do at least as well as the better of generating $X$ and $Y$ independently, and using common random numbers.

The computational effort required to solve the optimization problem can be expected to grow with the dimension (number of uniform random variables required to generate both $X$ and $Y$). Accordingly, it is unlikely that the ideas herein can be successfully applied in high-dimensional situations. We believe that the most likely settings where these techniques could be effective are in situations where a simulation model for $Y$ is obtained by making localized changes to a model for generating $X$. We can then apply the ideas herein to couple only the modified parts of the model, and not the full model. Two of the three examples in Section 5 are exactly of this form, where $X$ and $Y$ represent the output of queueing simulations that are identical except in the way in which service times are constructed. We therefore apply our techniques to minimize the variance of the difference in service times in the two queueing models, as a heuristic that seems likely to yield a smaller variance in the difference of the simulation output for the full queueing models.

The paper is organized as follows. Section 2 presents the class of joint distributions over which we optimize. Section 3 frames the optimization problem and explores some special cases. Section 4 describes two approaches to the numerical optimization problem. Section 5 presents three examples that illustrate the approach. Section 6 concludes the paper.

2. A GAUSSIAN COUPLING

Virtually any simulation model can be viewed as a function that maps a set of independent and identically distributed (i.i.d.) uniform random variables $\{U_1, U_2, \ldots\}$ to an output random variable. We suppose that $X = f_U(U_X)$ where $U_X = (U_X(1), U_X(2), \ldots, U_X(d_X))$ consists of $d_X$ i.i.d. uniform random variables on $(0, 1)$, the dimension $d_X$ is nonrandom and finite, and $f_U$ is some real-valued function. For example, many problems in finance, and many spreadsheet simulation models, are of this form. We assume a similar form for $Y$, namely $Y = g_U(U_Y)$, where $U_Y$ is a $d_Y$-dimensional vector of i.i.d. uniform random variables on $(0, 1)$. Without loss of generality we assume, for notational convenience only, that $d_Y \leq d_X$.

In this paper we explore the joint distribution of $(U_X, U_Y)$. To ensure that $X$ and $Y$ have the appropriate distributions we impose the constraint that the marginal distributions of $U_X$ and $U_Y$ be uniform on $(0, 1)^{d_X}$ and $(0, 1)^{d_Y}$, i.e., that of i.i.d. uniform random variables on $(0, 1)$.

A particularly tractable class of joint distributions arises from the Gaussian distribution. The idea is to begin with a $d_X + d_Y$-dimensional normal random vector $Z = (Z(1), Z(2), \ldots, Z(d_X + d_Y))$, where each component $Z(i)$ has mean 0 and variance 1. One then transforms $Z(i)$ into a uniform random variable $U(i)$ via the transformation $U(i) = \Phi(Z(i))$, where $\Phi$ is the cumulative distribution function of a standard normal random variable. We then take $U_X = (U(1), U(2), \ldots, U(d_X))$ to be the first por-
tion of the resulting vector, and \( U_Y = (U(d_X + 1), U(d_X + 2), \ldots, U(d_X + d_Y)) \) to be the second portion, and obtain \( X = f_U(U_X) \) and \( Y = g_U(U_Y) \). The transformations here are exactly those applied in the NORmal To Anything (NORTA) method [Cario and Nelson 1997], albeit in the setting of comparing two systems rather than inducing a target dependence structure for a single random vector.

To ensure that \( X \) and \( Y \) have the correct distributions, we require that \( U_X \) and \( U_Y \) both have i.i.d. components. To this end, we must restrict the set of covariance matrices of the Gaussian vector \( Z \) appropriately. Specifically, we require the covariance matrix of \( Z \) to have the form

\[
\Sigma = \begin{bmatrix} I_{d_X} & A^T \\ A & I_{d_Y} \end{bmatrix}
\]

(1)

where \( I_n \) denotes the \( n \times n \) identity matrix and \( A \) is a \( d_Y \times d_X \) matrix of covariance terms. This form ensures that the first \( d_X \) components of \( Z \) are independent, as are the last \( d_Y \) components of \( Z \), and hence the same is true of the components of \( U_X \) and \( U_Y \) respectively. However, there can be dependence between the vectors \( U_X \) and \( U_Y \) which is derived from the matrix \( A \). The matrix \( A \) cannot be completely arbitrary; it must be chosen so that \( \Sigma \) is positive semidefinite.

The distribution of the \((d_X + d_Y)\)-dimensional vector \((U_X, U_Y)\) has uniform one-dimensional marginals on \((0, 1)\) and is therefore a copula; see Nelsen [1999] for an introduction to the theory of copulas. Two important special cases of copulas are captured within our Gaussian copula construction above. First, the independence copula in which \( U_X \) and \( U_Y \) are independent of one another arises when we take \( A = 0 \). In this case \( Z \) has independent components and, therefore, so does the \((d_X + d_Y)\)-dimensional vector \( U \). Second, CRN arises (assuming \( d_X = d_Y \)) when we take \( A = I_{d_X} \), so that \( Z_X = Z_Y \), where \( Z_X \) (\( Z_Y \)) is the vector consisting of the first \( d_X \) (last \( d_Y \)) components of \( Z \). In this case, \( U_X = U_Y \) as required for CRN.

So our class of joint distributions includes the special cases where \( X \) and \( Y \) are independent, or are obtained from CRN. There are simple examples where a Gaussian copula can yield a smaller value of \( \text{var}(X - Y) \) than these special copulas. Before presenting them, it will be helpful to extend our notation slightly. Recall that \( X = f_U(U_X) \) and \( U_X(i) = \Phi(Z_X(i)), i = 1, 2, \ldots, d_X \). Accordingly, we can view \( X = f(Z_X) \) for some appropriately defined composition-function \( f \). Similarly, we can write \( Y = g(Z_Y) \) for some \( g \).

**Example 2.1.** Take the dimensions \( d_X = 2 \) and \( d_Y = 1 \). Let

\[
f(z) = \frac{z(1) + z(2)}{\sqrt{2}} \quad \text{and} \quad g(z) = z(1).
\]

Here, \( X = f(Z_X) \) and \( Y = g(Z_Y) \) are linear functions of \( Z_X \) and \( Z_Y \) respectively, so are also normally distributed, in fact with mean 0 and variance 1. If we use the independence copula, then \( \text{var}(X - Y) = 2 \). With a version of CRN that sets \( Z_X(1) = Z_Y(1) \) a quick calculation gives \( \text{var}(X - Y) = 2 - \sqrt{2} \), which is an improvement over the independence copula. However, taking

\[
A = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \end{bmatrix}
\]

with the Gaussian copula approach, we obtain zero variance. To see why, notice that the matrix \( A \) corresponds to the transformation

\[
Z_Y(1) = \frac{Z_X(1) + Z_X(2)}{\sqrt{2}}
\]
and then \( X - Y = 0 \).

Hence, in simple examples, impressive variance reductions are possible.

**Example 2.2.** Take the dimensions \( d_X = d_Y = 1 \), \( f_U(U_X) = \mathbb{I}(U_X \in [.5,.6]) \) and \( g_U(U_Y) = \mathbb{I}(U_Y \in [.7,.8]) \), where \( \mathbb{I} \) is the usual indicator function that equals 1 if its argument is true and equals 0 otherwise. Hence, \( EX = EY = 0.1 \). In the case of the independence copula \( \text{var} (X - Y) = 0.18 \), while with CRN \( \text{var} (X - Y) = \text{var} X + \text{var} Y - 2 \text{cov} (X, Y) = 0.2 \). The variance increase due to the use of CRN arises because \( X \) and \( Y \) are never simultaneously equal to 1, so their covariance is negative. The marginal distributions of \( X \) and \( Y \) are fixed, so minimizing the variance of \( X - Y \) is equivalent to maximizing their covariance, or maximizing \( EXY \), since \( EX \) and \( EY \) are fixed. We are working in one dimension, so the covariance matrix is of the form

\[
\Sigma = \begin{bmatrix} 1 & \rho \\ \rho & 1 \end{bmatrix},
\]

and we seek the correlation \( A = \rho \in [-1,1] \) that maximizes \( E[XY] \). Figure 1 shows how \( E[XY] \) depends on \( \rho \). The resulting optimal \( \text{var} (X - Y) \) over Gaussian copulas is 0.15, which occurs at approximately \( \rho = 0.97 \).

The most important observation from Figure 1 is that there are two local maxima. This suggests that, in general, we must expect the optimization problem of minimizing \( \text{var} (X - Y) \) over the class of Gaussian copulas to be a multimodal optimization problem.
3. THE OPTIMIZATION PROBLEM: QUADRATICS

The examples in Section 2 demonstrate that large variance reductions are possible, and that we should expect the optimization problem of identifying the matrix $A$ of covariance terms in (1) to be multimodal. In one dimension it is possible to plot the variance of $X - Y$ as a function of $A$, and thereby graphically determine the optimal value. In higher dimensional problems this is no longer achievable, and we must turn instead to numerical optimization. Ideally one would like to identify the globally optimal choice of $A$, but, as discussed below, that is unlikely to be achievable.

The general optimization problem is to

$$\min_{A} \text{var}(X - Y) \quad (2)$$

s.t. $\begin{bmatrix} I & A^\top \\ A & I \end{bmatrix} \succeq 0$.

Here, “$\succeq 0$” denotes the matrix inequality that requires that the matrix on the left-hand side be positive semidefinite. The theory of semidefinite programming (e.g., see Boyd and Vandenberghe 2004) ensures that the resulting feasible region (in terms of the components of the matrix $A$) is convex. In fact, the constraint is equivalent to requiring $I - AA^\top \succeq 0$, as can be seen through the theory of Schur complements; see, e.g., Boyd and Vandenberghe [2004, p. 651].

The objective function (2) is implicitly a function of $A$, since the joint distribution of $(X,Y)$ depends on $A$, and hence so does $\text{var}(X - Y)$. Minimizing $\text{var}(X - Y)$ is equivalent to maximizing $\text{cov}(X,Y)$ or maximizing $E[XY]$, because the marginal distributions of $X$ and $Y$ are fixed throughout. Accordingly, we can replace the objective function (2) with these other forms when convenient. Even with these changes, the objective function is still nonlinear in $A$, and so our optimization problem is a nonlinear semidefinite program.

Before presenting a general approach to tackling the optimization problem it is worth considering some special cases to develop intuition and enhance understanding.

Example 3.1. Let the dimensions $d_X = d_Y = 1$ and suppose that the functions $f$ and $g$ are both quadratic, i.e.,

$$f(z) = a_1 z + a_2 z^2, \quad \text{and} \quad g(z) = b_1 z + b_2 z^2.$$  

Direct calculation gives $EX = a_2$, $EY = b_2$ and

$$\text{cov}(X,Y) = E[f(Z_X)g(Z_Y)] = 2a_2b_2\rho^2 + a_1 b_1 \rho,$$

where $A = \rho$ is the off-diagonal correlation that is our decision variable. Our goal is to maximize this quadratic function over $\rho \in [-1,1]$. There are two primary cases of interest.

Case 1: $a_2 b_2 > 0$. In this case the covariance is strictly convex in $\rho$, so the optimal correlation $\rho^*$ occurs at an extreme point. In particular, $\rho^* = 1$ if $a_1 b_1 > 0$ and $\rho^* = -1$ if $a_1 b_1 < 0$.

Case 2: $a_2 b_2 < 0$. In this case the covariance function is strictly concave in $\rho$, so there is a unique optimal solution that can be found by local search. Temporarily ignoring the requirement that $\rho \in [-1,1]$, the unrestricted maximum occurs when $\rho = -a_1 b_1 / 4a_2 b_2$, but this solution must be projected onto $[-1,1]$ if it lies outside the interval. We again see that the sign of the optimal correlation agrees with the sign of $a_1 b_1$, since $a_2 b_2 < 0$.  

ACM Transactions on Modeling and Computer Simulation, Vol. 9, No. 4, Article 39, Publication date: March 2010.
The key observation in Example 3.1 is that the optimization problem may possess multiple local optima in some cases, whereas in others there will be a unique global maximum that can be found by a local-search algorithm. A second observation is that there are situations where the optimal solution corresponds to a “change of variables,” whereby the Gaussian random variable $Z_Y$ is a deterministic function of $Z_X$. This happens in Example 3.1 when the covariance is a convex function of the correlation, since $Z_Y = Z_X$ (CRN) when $\rho = 1$, and $Z_Y = -Z_X$ when $\rho = -1$.

Can similar observations persist when the dimension is greater than one? Before answering this question we need some more notation. Let $L$ be a matrix such that $LL^\top = \Sigma$, where $\Sigma$ is the matrix in (1). We can then generate the Gaussian random vector $Z$ with covariance matrix $\Sigma$ using $Z = LN$, where $N$ is a Gaussian random vector with i.i.d. zero-mean, unit-variance components. We can take

$$L = \begin{bmatrix} I & 0 \\ A & B \end{bmatrix},$$

(3)

where $A$ is the submatrix appearing in $\Sigma$ and $B$ is a square matrix with the property that $BB^\top = I - AA^\top$. So for any given matrix $A$, let $B = B(A)$ and $L = L(A)$ be defined in this way.

**Example 3.2.** Now consider quadratics in some dimension $d_X = d_Y = d > 1$. We take

$$f(z) = q^\top z + z^\top Qz,$$

and

$$g(z) = r^\top z + z^\top Rz$$

for appropriate (column) vectors $q$ and $r$ and symmetric matrices $Q$ and $R$. Let $\text{tr} (V)$ be the trace of the square matrix $V$, i.e., the sum of its diagonal elements. To develop expressions for the means of $X$ and $Y$ and the covariance between $X$ and $Y$, we use the relation $\text{tr} (V^\top W) = \sum_{i,j=1}^d V_{ij} W_{ij}$ for square matrices $V$ and $W$, along with values for the moments of normal random vectors. Direct calculation then gives, after some effort,

$$EX = \text{tr} (Q),$$

$$EY = \text{tr} (A^\top QA) + \text{tr} (B^\top RB),$$

and

$$\text{cov} (X,Y) = 2\text{tr} (QA^\top RA) + r^\top Aq.$$

Here, $A$ and $B$ are the submatrices of the matrix (3). Our goal is to maximize the covariance over matrices $A$ that satisfy $I - AA^\top \succeq 0$, where $I$ is the $d \times d$ identity matrix. As in Example 3.1 there are two primary cases of interest.

**Case 1:** $Q \succeq 0, R \succeq 0$, or $-Q \succeq 0, -R \succeq 0$. Here both quadratic functions are convex, or both are concave. Consider just the convex case ($Q \succeq 0, R \succeq 0$), as the concave case is essentially identical. In that setting, $\text{cov} (X,Y)$ is a convex function of the elements of $A$. To see why, write $Q = L_Q L_Q^\top$ and $R = L_R L_R^\top$ for matrices $L_Q$ and $L_R$. This is possible because we assumed that $Q, R \succeq 0$. Using the additional fact that $\text{tr} (AB) = \text{tr} (BA)$ for square matrices $A$ and $B$, we see that

$$\text{cov} (X,Y) = 2\text{tr} (L_Q L_Q^\top A^\top L_R L_R^\top A) + r^\top Aq$$

$$= 2\text{tr} (L_Q^\top A^\top L_R L_R^\top A L_Q) + r^\top Aq$$

$$= \|L_R^\top A L_Q\|^2_F + r^\top Aq,$$
where \( \|A\|_F^2 \) is the square of the Frobenius norm of the matrix \( A \), namely \( \sum_{i,j} A_{i,j}^2 \). So the covariance is a sum of a convex quadratic function (in \( A \)) and a term that is linear in \( A \), and hence is convex.

Hence, we are attempting to maximize a convex function over a convex set, and in searching for local or global optima we may restrict our attention to extreme points. We are optimizing over matrices that satisfy the matrix inequality \( I - AA^\top \succeq 0 \), and so it follows that an optimal matrix, \( \bar{A} \) say, is such that \( I - \bar{A}\bar{A}^\top \) is singular (and positive semidefinite). Perhaps surprisingly, we can assert something stronger, namely that \( I - \bar{A}\bar{A}^\top = 0 \), i.e., \( A \) is orthogonal. The proof of the following result can be found in Appendix A.

**Proposition 3.3.** When \( Q \succeq 0 \) and \( R \succeq 0 \), or when \( -Q \succeq 0 \) and \( -R \succeq 0 \), there exists an optimal solution \( \bar{A} \) that is orthogonal.

**Case 2:** \( Q \succeq 0, -R \succeq 0, \) or \( -Q \succeq 0, R \succeq 0 \). Here one of the functions \( f \) and \( g \) is convex and the other is concave. Using similar reasoning to that for the previous case we discover that \( \text{cov} (X, Y) \) is concave in \( A \). Hence we are maximizing a concave function over a convex set, and a global optimum can be identified by a local-search algorithm.

We conclude from Example 3.2 that indeed, there are again situations where one can expect multiple local optima, and others where the objective function is strictly concave so that a unique optimum exists. We are unlikely to be able to determine, for a given problem, which setting we are in. Therefore, rather than attempt to identify a global optimum, we instead adopt the less ambitious goal of identifying a Gaussian copula that does at least as well (in terms of variance), and hopefully better than, the better of the independence and CRN copulas. This can be achieved by performing two local searches, starting from the two solutions corresponding to the independence and CRN copulas. We then select the better of the two Gaussian copulas obtained. Furthermore, with our modest goal there is no need to obtain an extremely accurate solution to the optimization problem. This should save in computation time searching for good copulae that could otherwise be spent simulating.

We can also conclude from Example 3.2 that it is possible that an optimal matrix \( \bar{A} \) is orthogonal. In this case, \( B = 0 \), and \( Z_Y \) is an orthogonal transformation of \( Z_X \). Orthogonal transformations can be viewed as changes of variables, so in that sense, the orthogonal transformations represent a generalization of CRN. (CRN can be viewed as the particular orthogonal transformation \( Z_Y = Z_X \).

### 4. THE OPTIMIZATION PROBLEM: ALGORITHMS

The optimization problem (2) is difficult to solve, partly because the objective function is, in general, a nonlinear function of \( A \), but also because we cannot compute the objective function exactly.

An approach to the latter problem can be based on the general principle of sample-average approximation (SAA) (see Shapiro 2003) as follows. In our setting we first select a sample size \( n \) and generate and fix an i.i.d. sample of \((d_X + d_Y)\)-dimensional standard normal random vectors \( N^{(1)}, N^{(2)}, \ldots, N^{(n)} \), so that, in particular, each vector \( N^{(m)} \) has zero mean and covariance matrix given by the identity matrix in dimension \( d_X + d_Y \). Let \( N^{(m)}_X \) (\( N^{(m)}_Y \)) be the first \( d_X \) (last \( d_Y \)) components of \( N^{(m)} \). Let \( L \) be the matrix given by (3). Then we can take

\[
\begin{bmatrix}
Z^{(m)}_X \\
Z^{(m)}_Y \\
\end{bmatrix} = L \begin{bmatrix}
N^{(m)}_X \\
N^{(m)}_Y \\
\end{bmatrix}
\]
for each \( m = 1, 2, \ldots, n \) and then the pairs \((Z_X^{(m)}, Z_Y^{(m)})\) have the desired joint distribution. We then search for a \( d_Y \times d_X\) matrix \( A \) that minimizes the sample variance of \( X - Y \),

\[
\frac{1}{n-1} \sum_{m=1}^{n} \left[ f(Z_X^{(m)}) - g(Z_Y^{(m)}) - M_n \right]^2,
\]

where

\[
M_n = \frac{1}{n} \sum_{m=1}^{n} \left[ f(Z_X^{(m)}) - g(Z_Y^{(m)}) \right]
\]

is the sample mean of \( X - Y \). (Recall that given such a matrix \( A \), we can find a matrix \( B = B(A) \) such that \( BB^\top = I - AA^\top \), and together these matrices determine \( L \).) Conditional on the original sample of \((d_X + d_Y)\)-dimensional normal random vectors, the SAA problem is a deterministic optimization problem and we can appeal to methods for deterministic nonlinear programming to solve it.

We provide two formulations of the SAA problem. One is based on nonlinear semidefinite programming while the other is based on nonlinear optimization over a manifold. In both cases the appropriate optimization algorithms are more effective when the objective function is differentiable and derivatives can be provided. We discuss derivatives for both formulations.

Our first formulation views the matrix \( A \) as the primary decision variable, and takes the matrix \( B = B(A) \) to be the Cholesky factor of \( I - AA^\top \) for any \( A \). We then have, as above, that \( Z_Y^{(m)} = AN_Y^{(m)} + B(A)N_Y^{(m)} \) for each \( m = 1, 2, \ldots, n \), and \( M_n \) is the sample mean (4). Both \( Z_Y^{(m)} \) and \( M_n \) depend on \( A \), but we suppress this for notational simplicity. The optimization problem is then

\[
\min h(A) = \frac{1}{n-1} \sum_{m=1}^{n} \left[ f(Z_X^{(m)}) - g(Z_Y^{(m)}) - M_n \right]^2
\]

\[
\text{s.t. } I - AA^\top \succeq 0.
\]

Let us call this formulation the “direct formulation.” The direct formulation is a nonlinear semidefinite programming problem with a convex feasible region. Unfortunately, gradients (with respect to the components of the matrix \( A \)) are difficult to obtain, owing to the fact that the matrix \( B \) is a function of \( A \). Expressions for these gradients are obtained in Appendix B under the assumption that \( g \) is differentiable. To obtain a sense of the computational effort involved in computing the derivatives, suppose that \( d_X = d_Y = d \). Then the derivatives can be obtained in order \( O(d^3) \) work, because one has to follow the steps of a Cholesky decomposition, which is \( O(d^3) \) work, to get the partial derivative of \( h \) with respect to a particular component of \( A \), and there are \( d^2 \) components of \( A \). The same amount of work results if one uses finite differences to obtain the gradients, because for each component of \( A \), one has to perform a Cholesky decomposition. One might attempt to use efficient updating rules for the Cholesky factors that reduce this work to \( O(d^4) \), but in any case the direct formulation is likely to be practical only for modest dimensions \( d \), say \( d \leq 10 \).

For our second formulation, define the \( d_Y \times (d_X + d_Y) \) matrix \( C = [A \ B] \), where \( A \) is the lower-left submatrix of \( \Sigma \) and \( B \) is any matrix that yields \( AA^\top + BB^\top = I \). We now
take \( Z_Y^{(m)} = CN^{(m)} \) and the optimization problem becomes
\[
\min h(C) = \frac{1}{n-1} \sum_{m=1}^{n} [f(N_X^{(m)}) - g(CN^{(m)}) - M_n]^2
\]
\[\text{s.t. } CC^\top = I,\]
where the sample mean \( M_n \) is again defined as in (4). The feasible region is not convex but is, instead, an example of a Stiefel manifold. Software is available for optimizing over this manifold [Edelman and Lippert 2000]. Gradients are readily obtained in this formulation. In particular, assuming that \( g \) is differentiable with (column-vector) gradient \( \nabla g \), the matrix \( \mathcal{G} \) with \((i,j)\)th coordinate giving \( \partial h / \partial C_{ij} \) is given by
\[
\mathcal{G} = -\frac{2}{n-1} \sum_{m=1}^{n} (X_m - Y_m - M_n)\nabla g(CN^{(m)})(N^{(m)})^\top,
\]
where \( \nabla g(CN^{(m)})(N^{(m)})^\top \) is an outer product. Let us call this formulation the “Stiefel formulation.”

5. EXAMPLES
5.1. Comparing Portfolios
Consider a simplified risk-management problem as follows. Suppose that the true \( (\begin{array}{c}
\text{as opposed to the equivalent martingale measure) prices of} \end{array} \) \( d \) stocks evolve according to a \( d \)-dimensional geometric Brownian motion. More precisely, assume that the vector \( S(t) = (S_1(t), S_2(t), \ldots, S_d(t)) \) of true prices evolves according to the system of stochastic differential equations
\[
dS_i(t) = S_i(t)(r_i \, dt + \sigma_i \, dB_i(t)), \quad i = 1, 2, \ldots, d,
\]
where \( B_1(\cdot), B_2(\cdot), \ldots, B_d(\cdot) \) are potentially correlated standard Brownian motions. A portfolio of stocks consists of a vector, \( \mathbf{p} \) say, giving the holdings in each stock. Suppose we wish to compare two portfolios, \( \mathbf{p} \) and \( \mathbf{q} \), with respect to their risk, perhaps as part of a risk mitigation process where a search is conducted for a low-risk portfolio that is “close” to a current portfolio. To measure risk we compute the expected loss over some loss threshold \( \ell \) over a time interval \([0, t]\). (This measure is related to, but not the same as, conditional value at risk.) For a portfolio \( \mathbf{p} \), this expected loss can be written as
\[
E(X) = E[-\ell - \mathbf{p}^\top (S(t) - S(0))]^+,
\]
where \( \lfloor x \rfloor^+ = \max(x, 0) \) is the positive part of \( x \). Similarly, for the portfolio \( \mathbf{q} \), the expected loss is \( E(Y) = E[-\ell - \mathbf{q}^\top (S(t) - S(0))]^+ \). (Our simulations replicate the true dynamics of the stock prices rather than the equivalent martingale measure dynamics because our application area is not pricing instruments, but rather understanding risk.)

To generate an instance of \( X \) or \( Y \), we need to generate \( S(t) \), the \( i \)th component of which is
\[
S_i(t) = S_i(0) \exp[(r_i - \sigma_i^2/2) t + \sigma_i B_i(t)].
\]
Hence we can generate \( X \) or \( Y \) by first generating the \( d \) dimensional multivariate normal random vector \( B(t) \). If \( B(t) \) has covariance matrix \( \Sigma \) then we can write \( B(t) = \sqrt{tL}Z \), where \( L \) is a lower-triangular matrix such that \( LL^\top = \Sigma \) and \( Z \) is a \( d \) dimensional Gaussian random vector with mean vector \( 0 \) and covariance matrix \( I \). We can therefore employ the methods of this paper by optimizing over the joint distribution of \( [Z_X, Z_Y] \) where \( Z_X \) and \( Z_Y \) are each \( d \) dimensional standard normal random vectors.
To obtain a particularly transparent instance, we take \( t = 1/12 \) (one month), \( d = 3 \) (three stocks), \( r_i = r = 0.05, \sigma_i = \sigma = 0.3, S_i(0) = 100 \) for each \( i \), and \( \ell = 4 \). We also take

\[
L = \begin{pmatrix}
1 & 0 & 0 \\
\rho_{12} \sqrt{1 - \rho_{12}^2} & 0 & 0 \\
\rho_{13} & 0 & \sqrt{1 - \rho_{13}^2}
\end{pmatrix},
\]

so that the prices of Stocks 2 and 3 are correlated with Stock 1, but conditional on the price of Stock 1 are uncorrelated. We took \( \rho_{12} = \rho_{13} = 0.3 \). We wish to compare the portfolios \( p = [0.5, 0.5, 0] \) and \( q = [0.5, 0, 0.5] \), which actually have the same expected loss because under the above parameter choices \( X \) and \( Y \) have the same distribution.

This example is tailored to ensure that a coupling with zero variance exists. This coupling arises when \( Z_Y(1) = Z_X(1) \) and \( Z_Y(3) = Z_X(2) \), because then the prices of Stocks 2 and 3 are interchanged and \( X = Y \) everywhere. So in this example we should be able to dramatically outperform both independent sampling and CRN.

Recall that we first solve, or approximately solve, an optimization problem that yields a matrix \( A \) (or \( C = [A \ B] \) in the Stiefel formulation), and then perform a “production run” where we simulate using the Gaussian copula based on \( A \).

We first tested the direct formulation, solving the optimization problems using a sequential semi-definite programming algorithm [Jarre 2009]. The code calls the package “SeDuMi” to solve the linearized subproblems [SeDuMi 1.3 2011] and runs in Matlab 2008a for compatibility with SeDuMi. We then tested the Stiefel formulation using the Matlab package sgmin [Edelman and Lippert 2000] with tolerances of 5% in the tests of convergence based on both gradient and function information. We used a Macintosh Powerbook Pro with a 2.66GHz Intel Core i7 processor and 4GB of RAM.

We varied the number of samples used in the optimization phase (the term “\( n \)” in (5)). We began the search for an optimal matrix \( A \) from both the independence copula (\( A = 0 \)), and from the CRN copula (\( A = I \)). We followed up the optimization runs with production simulation runs with a sample size of 250,000.

The times required to perform the simulation runs themselves varied very little so we report averages rather than detailed results. Simulation using CRN and simulation using the copula arising from the optimization starting at \( A = 0 \) both averaged 0.21s over the experiments reported in Table I. Simulation using the independence copula took an average of 0.23s, while simulation using the copula arising from the optimization starting at \( A = I \) took an average of 0.24s. From these times we conclude that there is little to compare the methods based on the speed of the simulation runs themselves.

The estimated variance from standard Monte Carlo (the independence copula) was 46, while the estimated variance using CRN was 22.

In Table I we report, to two significant figures,

(1) the time \( t_{\text{opt}} \) to perform the specified optimization and the estimated variance \( v \) from the production run,

(2) the estimated efficiency improvements obtained from using the Gaussian copula approach relative to standard Monte Carlo \( \text{EI}_{MC} = t_{MC} v_{MC} / (t_{cop} v_{cop}) \), and the analogous quantity for CRN. An increase in efficiency is indicated by a number greater than 1.

We see from these results that

— the optimization times for the direct formulation are significantly larger than those for the Stiefel formulation,

— the performance of the optimization routines in terms of the estimated variance of the final estimator is quite robust to the sample size \( n \) used,
Table I. Numerical results for the portfolio example (two significant figures). All times are in seconds.

<table>
<thead>
<tr>
<th>n</th>
<th>Optimization from $A = 0$</th>
<th>Optimization from $A = I$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$t_{opt}$</td>
<td>$\nu$</td>
</tr>
<tr>
<td>Direct</td>
<td></td>
<td></td>
</tr>
<tr>
<td>100</td>
<td>1.4</td>
<td>0.24</td>
</tr>
<tr>
<td>200</td>
<td>2</td>
<td>0.25</td>
</tr>
<tr>
<td>500</td>
<td>3.2</td>
<td>0.24</td>
</tr>
<tr>
<td>Stiefel</td>
<td></td>
<td></td>
</tr>
<tr>
<td>100</td>
<td>0.4</td>
<td>0.17</td>
</tr>
<tr>
<td>200</td>
<td>0.42</td>
<td>0.16</td>
</tr>
<tr>
<td>500</td>
<td>0.64</td>
<td>0.17</td>
</tr>
</tbody>
</table>

— the optimized copula starting from $A = I$ using the direct formulation is quite poor for $n = 100, 500$, suggesting the presence of a local minimum that is avoided through randomness when $n = 200$,

— the efficiency improvements are all extremely large, except in the cases mentioned above where the direct formulation does not perform as well, and

— the optimal $A$ matrix is, except in the two cases highlighted above, close to the matrix that yields zero variance. The (numerical) optimization routines do not return the truly optimal matrix due to the use of the error tolerances mentioned above that allow a tradeoff between obtaining low estimated variance and not optimizing over estimation noise.

Our estimated efficiency improvement metric does not factor in the time for the optimization because the optimization is performed once for any length of simulation, and so by varying the simulation length one would see larger or smaller apparent efficiencies. The results indicate that the direct formulation can require a nontrivial amount of time for optimization relative to the time spent on simulation, but that the time required for optimization with the Stiefel formulation is modest, at least relative to the efficiency improvements obtained.

This example demonstrates that extremely large variance reductions can be obtained through numerical optimization when a zero-variance estimator exists. To see what kind of results one might expect on more realistic problems, we took $d_X = 20$, the rate vector $r$ to be linearly spaced from 0.02 to 0.08 (inclusive), the standard deviation vector $\sigma$ to be linearly spaced from 0.2 to 0.4, $p$ to be linearly spaced in reverse order from 1 to 0 and normalized to have length 1, and $q$ to be linearly spaced from 0 to 1 and normalized to have length 1. We generated the lower triangular matrix $L$ so that the $i$th row consisted of the absolute value of iid normal random variables, normalized to have length 1 so that $LL^\top$ is a correlation matrix with all positive correlations. (This seemed more appropriate than to allow large negative correlations.) The particular $L$ that was generated is given in an appendix. All other parameters remained the same as in the 3-dimensional example. We used a sample size of 500 in the optimization phase and 250000 in the production runs, and optimized using the Stiefel formulation.

The optimization steps took 3.7s (2.4s) starting from the independence (CRN) copula. The production runs all took 1.2 seconds. The estimated variance using the independence (CRN) copula was 820 (190). The estimated variance using our optimized copulas starting the search from the independence (CRN) copula was 48 (51). These results represent efficiency improvements of a factor of approximately 16 over the independence copula and a factor of almost 4 over CRN.

5.2. Comparing Two Queues I

In both of our remaining examples, jobs arrive to a service facility according to a Poisson process with constant rate $\lambda$. Jobs are served in first-in-first-out order, and the service times are i.i.d. and independent of the arrival process. The two examples differ in the way service times are constructed. In each example we are interested in se-
lecting one of two configurations that has the smaller expected steady-state time that jobs spend in the system. In both cases we could appeal to the Pollaczek-Khintchine formula for $M/G/1$ queues, but our purpose is to demonstrate comparisons via simulation. To compare these systems using simulation, we simulate the processing of a large number of jobs, $k$ say, in both systems, compute the mean time in system of the jobs in both configurations, and finally take the difference of those means. In order to assess the uncertainty in this difference of means, we use the batch means method. Each batch consists of $m$ jobs, and we select the number of batches $b$ so that $k = mb$. For each batch we compute the difference in mean system times of the jobs in the batch, giving $b$ approximately i.i.d. observations of this difference. According to batch means theory, e.g., [Fishman 2001, p. 246], for sufficiently large batch sizes $m$, each batch mean has variance approximately equal to $\sigma^2/m$, where $\sigma^2$ is the so-called time-average variance constant that appears in a central limit theorem for the difference of sample means. An estimate of $\sigma^2$ is then given by $ms^2$, where $s^2$ is the sample variance of the batch means.

For each example it is natural to take the arrival processes to the two system configurations to be identical, i.e., to couple the arrival processes using CRN. However, it is not clear how to couple the service times, and for that step we apply the methods of this paper.

We turn now to our first example of comparing two queueing systems.

In the first system configuration (System 1), jobs are processed on a sequence of $d_X$ machines by an operator. The operator completes all processing on a single job, taking the job through all $d_X$ machines before retrieving the next job. The second system configuration (System 2) consists of a single machine that can complete all processing in a single step.

For each job, there are $d_X + 1$ service times involved, where the $(d_X + 1)$th service time is that of the job on the single machine in System 2. The simulation comparison we want to perform is essentially infinite dimensional, in the sense that it involves $k(d_X + 1)$ random variables, and there are no natural bounds on the runlength $k$. One could select a value for $k$ and attempt to solve an optimization problem in $k(d_X + 1)$ dimensions but this is impractical. Alternatively, one could assume that the service times for every job have the same joint distribution and select this joint distribution to minimize the variance associated with the difference in waiting times. This is certainly possible, but we adopt an even simpler approach. We construct the joint distribution of the $d_X + 1$ separate processing times associated with a single job with the goal of minimizing the variance of $X - Y$, where $X$ is the total processing time of a single job on all $d_X$ machines in System 1, and $Y$ is the processing time of a single job on the single machine in System 2. So when we construct the joint distribution we use the variance of $X - Y$, but when we evaluate a joint distribution, we will do so on the basis of the batch-means estimated variance of the estimator of the difference in average waiting times.

The processing time on each of the existing machines is Weibull distributed with shape parameter 2 and scale parameter $2/\sqrt{\pi}$, i.e., the processing time has distribution function $F(t) = 1 - \exp(-\pi t^2/4)$ for each $j = 1, 2, \ldots, d_X$. All processing times are independent of one another and of the arrival process. Thus, the mean and variance of the processing time on a single machine are $1$ and $1 - \pi/4$ respectively. (See, e.g., Law [2007, p. 286] for properties of Weibull random variables.) The total processing time for each job then has mean $d_X$ and variance $d_X(1 - \pi/4)$.

The processing time on the single machine in System 2 is also Weibull distributed with shape parameter 2 and scale parameter $\beta = 2d_X/\Gamma'(1/2)$, where $\Gamma(\cdot)$ is the gamma function. This choice of parameters ensures that the processing times in both systems
Table II. Numerical results for the first comparison of queues using the direct formulation (two significant figures). All times are in seconds.

<table>
<thead>
<tr>
<th>$d_X$</th>
<th>$n$</th>
<th>$t_{opt}$</th>
<th>$t_{cop}$</th>
<th>$v_{cop}$</th>
<th>$t_{MC}$</th>
<th>$v_{MC}$</th>
<th>$t_{CRN}$</th>
<th>$v_{CRN}$</th>
<th>$E_{IMC}$</th>
<th>$E_{ICRN}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>100</td>
<td>0.62</td>
<td>4.3</td>
<td>1.7</td>
<td>3.4</td>
<td>16</td>
<td>3.5</td>
<td>3.4</td>
<td>5.6</td>
<td>5.0</td>
</tr>
<tr>
<td>2</td>
<td>200</td>
<td>0.88</td>
<td>4</td>
<td>1.7</td>
<td>3.4</td>
<td>17</td>
<td>3.5</td>
<td>6.1</td>
<td>8.5</td>
<td>2.9</td>
</tr>
<tr>
<td>2</td>
<td>500</td>
<td>0.65</td>
<td>4.7</td>
<td>1.7</td>
<td>3.5</td>
<td>18</td>
<td>3.5</td>
<td>6.2</td>
<td>7.9</td>
<td>2.9</td>
</tr>
<tr>
<td>5</td>
<td>100</td>
<td>2.2</td>
<td>5.9</td>
<td>30</td>
<td>4.5</td>
<td>82</td>
<td>4.1</td>
<td>63</td>
<td>2.1</td>
<td>1.3</td>
</tr>
<tr>
<td>5</td>
<td>200</td>
<td>1.1</td>
<td>6.2</td>
<td>31</td>
<td>4.4</td>
<td>83</td>
<td>4.2</td>
<td>56</td>
<td>1.9</td>
<td>1.4</td>
</tr>
<tr>
<td>5</td>
<td>500</td>
<td>0.99</td>
<td>6.4</td>
<td>30</td>
<td>4.2</td>
<td>88</td>
<td>4.1</td>
<td>61</td>
<td>1.9</td>
<td>1.4</td>
</tr>
<tr>
<td>10</td>
<td>100</td>
<td>2.8</td>
<td>9.2</td>
<td>190</td>
<td>5.8</td>
<td>290</td>
<td>5.7</td>
<td>250</td>
<td>1.0</td>
<td>0.7</td>
</tr>
<tr>
<td>10</td>
<td>200</td>
<td>2.7</td>
<td>9.2</td>
<td>190</td>
<td>5.8</td>
<td>300</td>
<td>5.7</td>
<td>260</td>
<td>1.2</td>
<td>0.8</td>
</tr>
<tr>
<td>10</td>
<td>500</td>
<td>2.8</td>
<td>9.2</td>
<td>180</td>
<td>5.9</td>
<td>320</td>
<td>5.9</td>
<td>260</td>
<td>1.2</td>
<td>0.8</td>
</tr>
</tbody>
</table>

have common mean $d_X$. Let $G$ be the distribution function of the processing time in System 2, so that $G(t) = 1 - \exp(-t/\beta)^2$.

We can now write

$$f(z(1), z(2), \ldots, z(d_X)) = \sum_{i=1}^{d_X} F^{-1}((\Phi(z(i))))$$

and

$$g(z(d_X + 1)) = G^{-1}((\Phi(z(d_X + 1))))$$

where $\Phi(\cdot)$ is the distribution function of a standard normal random variable.

The arrival rate $\lambda$ is chosen to yield a server utilization of 0.5.

We used the same computational setup as for the portfolio example described above. We varied the number of samples used in the optimization phase (the term “$n$” in (5)) and the dimension $d_X$. We began the search for an optimal matrix $A$ from both the independence copula ($A = 0$), and from $A = [1, 0, 0, \ldots, 0]$ (CRN does not apply directly to this example since the dimensions of the two systems are different.) Both starting points gave very similar results, so we report the results only for the starting point $A = 0$. For the production simulation run we used $b = 1000$ batches, each consisting of $m = 3000$ successive waiting times.

In Table II we report, to two significant figures,

(1) the time $t_{opt}$ required to perform the optimization (starting from $A = 0$),
(2) the time $t_{cop}$ required to perform the copula-based simulation and the resulting estimated variance $v_{cop}$,
(3) the time $t_{MC}$ required to perform standard Monte Carlo (no Gaussian copula) using independent instances of $X$ and $Y$, and the resulting estimated variance $v_{MC}$,
(4) the time $t_{CRN}$ required to perform CRN, and the resulting estimated variance $v_{CRN}$,
(5) the estimated efficiency improvements obtained from using the Gaussian copula approach relative to standard Monte Carlo $E_{IMC} = t_{MC}v_{MC}/(t_{cop}v_{cop})$, and the analogous quantity for CRN.

Testing the Stiefel formulation using the Matlab package sgmin [Edelman and Lippert 2000] using tolerances of 5% in the tests of convergence based on both gradient and function information again gave very similar results whether initiating from the independence copula or the CRN copula, so we only present results for optimization starting from the independence copula in Table III.

For $d = 2$, the $A$ matrix found using the direct formulation with a sample size $n = 100$ is $A = [0.6878, 0.7245]$, suggesting that near-optimal variance reductions might be obtained from the symmetrical solution $A = [1, 1, \ldots, 1]/\sqrt{d_X}$. The (numerically) optimal $A$ matrix in higher dimensions $d_X$ did not appear to take this form, but produced very similar variance reductions to those produced by this specific choice of $A$. 

ACM Transactions on Modeling and Computer Simulation, Vol. 9, No. 4, Article 39, Publication date: March 2010.
Table III. Numerical results for the first comparison of queues using the Stiefel formulation (two significant figures). All times are in seconds.

<table>
<thead>
<tr>
<th>(d_X)</th>
<th>(n)</th>
<th>(t_{opt})</th>
<th>(t_{cop})</th>
<th>(v_{cop})</th>
<th>(v_{MC})</th>
<th>(t_{CRN})</th>
<th>(v_{CRN})</th>
<th>(E_{IMC})</th>
<th>(E_{ICRN})</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>100</td>
<td>0.14</td>
<td>4.6</td>
<td>1.7</td>
<td>3.7</td>
<td>17</td>
<td>3.5</td>
<td>5.9</td>
<td>2.6</td>
</tr>
<tr>
<td>2</td>
<td>500</td>
<td>0.12</td>
<td>4.3</td>
<td>1.7</td>
<td>3.6</td>
<td>16</td>
<td>3.7</td>
<td>5.8</td>
<td>2.7</td>
</tr>
<tr>
<td>5</td>
<td>100</td>
<td>0.13</td>
<td>6.2</td>
<td>4.5</td>
<td>83</td>
<td>4.6</td>
<td>57</td>
<td>1.8</td>
<td>1.3</td>
</tr>
<tr>
<td>5</td>
<td>200</td>
<td>0.18</td>
<td>7.1</td>
<td>4.6</td>
<td>88</td>
<td>4.6</td>
<td>61</td>
<td>1.7</td>
<td>1.2</td>
</tr>
<tr>
<td>5</td>
<td>500</td>
<td>0.24</td>
<td>8.5</td>
<td>4.8</td>
<td>83</td>
<td>4.4</td>
<td>55</td>
<td>1.5</td>
<td>0.9</td>
</tr>
<tr>
<td>10</td>
<td>100</td>
<td>0.82</td>
<td>10</td>
<td>170</td>
<td>7</td>
<td>300</td>
<td>6.8</td>
<td>260</td>
<td>1.2</td>
</tr>
<tr>
<td>10</td>
<td>200</td>
<td>0.73</td>
<td>11</td>
<td>190</td>
<td>6.8</td>
<td>330</td>
<td>6.1</td>
<td>270</td>
<td>1.1</td>
</tr>
<tr>
<td>10</td>
<td>500</td>
<td>0.85</td>
<td>9.9</td>
<td>200</td>
<td>6.2</td>
<td>280</td>
<td>5.8</td>
<td>270</td>
<td>0.9</td>
</tr>
</tbody>
</table>

The results indicate that both formulations lead to similar efficiency gains or losses, but that the Stiefel formulation yields substantially faster optimization times. Large efficiency gains are possible in low dimension with efficiency improvement factors of approximately 3 over CRN, and factors of approximately 7 over standard Monte Carlo. The efficiency gains decrease as the dimension increases, and for \(d_X = 10\) one would marginally prefer CRN. The simulation times for standard Monte Carlo are faster than those for our method because our method requires the generation of Gaussian random variables, and their transformation to a uniform and then Weibull distribution, whereas standard Monte Carlo only requires the latter transformation. CRN simulations are faster again because fewer variates need to be generated.

We believe that the decrease in efficiency improvement as the dimension increases is due to the fact that as dimension increases, the service time distribution in the system with \(d_X\) machines becomes well approximated by a normal distribution, whereas the service time with 1 machine remains Weibull. Hence, the service times become harder to correlate, and so the efficiency improvement deteriorates.

### 5.3. Comparing Two Queues II

Now suppose that the service times are instead obtained as the maximum-length path through a stochastic-activity network. There are two activity networks corresponding to the two systems, and these are depicted in Figure 2. Within each activity network all arc durations are independent.

Let \(T_i\) denote a generic (random) duration corresponding to Arc \(i, i = 1, 2, \ldots, 12\). We take all task durations \(T_i\) to be lognormal with mean \(\mu_i\) and variance \(\sigma_i^2\). In System 1 we take \(\mu_1 = \mu_7 = 11, \sigma_1^2 = \sigma_7^2 = 4, \mu_2 = \mu_3 = \cdots = \mu_6 = 4,\) and \(\sigma_2^2 = \sigma_3^2 = \cdots = \sigma_6^2 = 4\).

In System 2 we take \(\mu_8 = \mu_12 = 10, \sigma_8^2 = \sigma_12^2 = 4, \mu_9 = \mu_{10} = \mu_{11} = 8\) and \(\sigma_9^2 = \sigma_{10}^2 = \sigma_{11}^2 = 16\). The arrival rate is chosen to yield a traffic intensity of approximately 0.75.

We used the same computational framework as above. CRN does not apply directly to this problem because \(d_X\) and \(d_Y\) are different. By “CRN” then, we mean that we used inversion from a common uniform random variable to generate...
Table IV. Numerical results for Example 2 (two significant figures). All times are in seconds.

<table>
<thead>
<tr>
<th></th>
<th>$t_{opt}$</th>
<th>$t_{cop}$</th>
<th>$v_{cop}$</th>
<th>EI</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Direct Formulation</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Standard Monte Carlo</td>
<td>-</td>
<td>6.3</td>
<td>7400</td>
<td>-</td>
</tr>
<tr>
<td>CRN</td>
<td>-</td>
<td>5.4</td>
<td>2100</td>
<td>4.1</td>
</tr>
<tr>
<td>Independence Copula</td>
<td>19</td>
<td>6.4</td>
<td>1400</td>
<td>5.1</td>
</tr>
<tr>
<td>CRN Copula</td>
<td>4.4</td>
<td>6.8</td>
<td>2200</td>
<td>3.2</td>
</tr>
<tr>
<td><strong>Stiefel Formulation</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Standard Monte Carlo</td>
<td>-</td>
<td>5.1</td>
<td>7200</td>
<td>-</td>
</tr>
<tr>
<td>CRN</td>
<td>-</td>
<td>4.4</td>
<td>2300</td>
<td>3.6</td>
</tr>
<tr>
<td>Independence Copula</td>
<td>0.6</td>
<td>5.1</td>
<td>1200</td>
<td>5.9</td>
</tr>
<tr>
<td>CRN Copula</td>
<td>0.7</td>
<td>5.4</td>
<td>1200</td>
<td>5.8</td>
</tr>
</tbody>
</table>

pairs of task times, with the pairs consisting of Tasks (1, 8), (7, 12), (2, 9), (4, 10), (5, 11). Each pair was generated independently of all other pairs, and of the task times for Tasks 3 and 6. The results appear in Table IV. The column “EI” gives the estimated efficiency improvement relative to standard Monte Carlo.

The direct formulation requires substantial optimization times, and yields effective variance reductions over standard Monte Carlo. The solution obtained by starting from the Independence Copula yields another 25% in variance reduction over CRN. The Stiefel formulation performs even better, requiring dramatically smaller optimization time and delivering substantial efficiency improvements, including an approximate 60% improvement over CRN.

The $A$ matrix found using the Stiefel formulation with a sample size $n = 500$ starting from the independence copula is

$$
\begin{pmatrix}
0.2179 & 0.4559 & -0.1291 & 0.1304 & 0.3913 & 0.0156 & 0.2991 \\
-0.3944 & 0.1173 & 0.2228 & -0.0336 & 0.1581 & 0.7087 & 0.3531 \\
0.5065 & -0.3948 & 0.0049 & 0.145 & 0.4965 & 0.1434 & 0.0527 \\
0.2368 & 0.3837 & 0.7657 & -0.0929 & -0.2133 & -0.1675 & 0.0828 \\
0.5058 & 0.1265 & -0.1018 & -0.095 & -0.1071 & 0.0892 & 0.4435
\end{pmatrix}
$$

We do not have an intuitive explanation for the form of this solution.

6. CONCLUSION

We have presented a method for constructing a joint distribution for two random variables, $X$ and $Y$, that are each constructed from a bounded number of random variables, with the goal of minimizing the variance of $X - Y$. The method uses a Gaussian copula, constrained so that the marginal distributions of $X$ and $Y$ are correct. We analyzed the case where $X$ and $Y$ are quadratic functions of the underlying Gaussian random variables, showing that in certain cases the optimization problem has a unique global optimum that will be identified by local search procedures. However, in general one must expect the objective function to be multimodal. In the case of quadratic functions, it will often be the case that an optimal copula corresponds to a change of variables.

We gave two algorithms for searching for an effective copula, and tested them on three examples. Two of the examples involved comparing two discrete-event simulation models, where the models differed in terms of how certain service times were constructed in queueing systems. Estimated efficiency-improvement factors relative to standard Monte Carlo and to common random numbers were substantial in some cases, but smaller in others. In a small number of cases, it is more efficient to use common random numbers instead of our techniques even though the variance is larger, because less computation is required. The direct formulation required much longer optimization times than the Stiefel formulation, but both formulations yielded similar estimators except in a few cases where the Stiefel formulation yielded stronger esti-
mators. On that basis, it appears that the Stiefel formulation is superior, at least with the numerical optimization algorithms we employed.

When simulation is used to evaluate incremental system improvements through modest changes to a system, the models to be compared will typically be quite similar. So the examples given here may be representative of many situations in practice, and may therefore represent a fruitful area of application of the techniques proposed.

The optimization steps required to find an effective Gaussian copula in the Stiefel formulation represented a small portion of the overall computational effort in our queueing examples. That may not always be true, in which case one might wonder whether the extra computation is worthwhile. To provide some insight on how much effort should be devoted to optimization we offer the following heuristic argument, which is similar to one given in Kim and Henderson [2007].

Given a computational budget $c$ as measured in units of simulation replications for standard Monte Carlo, the variance of standard Monte Carlo is of the order $\sigma^2/c$, for some variance constant $\sigma^2$. Assuming the computational effort in the optimization phase of our algorithm is linear in $n$ (since each function evaluation involves averaging $n$ samples), the computational effort in the optimization phase is $\gamma n$, where it is reasonable to expect that $\gamma \gg 1$. The work in the simulation stage is also linear in the sample size $k$, and taken to be $\beta k$ with $\beta > 1$ since we must generate Gaussian random variables and typically convert them to uniform random variables. (If we are comparing complex simulation models and the portion of the models constructed jointly using the methods of this paper is small, then $\beta$ may be close to 1.) So the total work is $\gamma n + \beta k$, and then the number of simulation replications is $k = \beta^{-1}(c - \gamma n)$. The variance associated with the coupling obtained at the end of the optimization phase depends on the accuracy of the optimized solution, which in turn depends on $n$. Heuristic arguments [Kim and Henderson 2007] suggest that the resulting variance is of the form $(v^2 + w/n)$, where $v^2 < \sigma^2$ is the minimal variance, and $w > 0$ gives the additional error associated with minimizing the sampled variance rather than the true variance, and is related to the curvature of the variance function near the optimal point. The overall variance is then

$$\frac{\beta(v^2 + w/n)}{c - \gamma n}.$$  (6)

Minimizing (6) with respect to $n$, we find that for large $c$, the optimal $n$ is asymptotically of the form

$$n^* \sim \left(\frac{w}{\gamma v^2}\right)^{1/2} \sqrt{c},$$

i.e., of order $\sqrt{c}$, which is asymptotically negligible compared to the computational budget $c$. As the optimization becomes computationally more expensive ($\gamma$ increases), $n^*$ decreases. For example, we would expect that $\gamma$ increases with the dimension of the problem, so that for problems in higher dimensions, the expense associated with optimization would lead us to sacrifice some potential variance reduction (by choosing a smaller $n^*$ than otherwise) in favor of more simulation replications. Also, as the optimal variance $v^2$ decreases, $n^*$ increases because it is especially worthwhile identifying a good solution. Finally, $n^*$ also increases with $w$, because if the (true) variance function has high curvature near the optimal point, then an accurate solution is required. The variance of the coupling procedure is, for large $c$, asymptotically $\beta v^2/c$, which beats standard Monte Carlo provided that $\beta v^2 < \sigma^2$, and this is essentially the metric we used to evaluate the efficiency improvement.
Our task is to develop an expression for the derivative of (5) in Section 4.

B. OBTAINING THE GRADIENT FOR THE DIRECT FORMULATION

A. PROOFS

APPENDIX

PROOF OF PROPOSITION 3.3. We prove the result when \( Q \succeq 0 \) and \( R \succeq 0 \). The other case is essentially identical.

Using the singular-value decomposition, we write \( A = UDV^T \), where \( U \) and \( V \) are orthogonal \( d \times d \) matrices and \( D \) is diagonal. Fixing \( U \) and \( V \) for now, the constraint \( I - AA^T \succeq 0 \) can be written

\[
I - AA^T = I - UDV^T VDU^T = UU^T - UD^2U^T = U(I - D^2)U^T \succeq 0,
\]

which is equivalent to \( I - D^2 \succeq 0 \). So for fixed \( U \) and \( V \), the set of feasible diagonal matrices \( D \) are such that \( -1 \leq D_{ii} \leq 1 \) for all \( i \). We will show that for any fixed \( U \) and \( V \), an optimal choice of \( D \) has \( D_{ii} = \pm 1 \) for all \( i \). This is true for arbitrary \( U \) and \( V \), and so it is also true of optimal choices of \( U \) and \( V \). Hence, there exists an optimal solution, \( \bar{A} = \bar{U} \bar{D} V^T \) say, such that \( D_{ii} = \pm 1 \) for all \( i \), so that \( I - D^2 = 0 \) and \( \bar{A}\bar{A}^T = I \), which is what we want to prove.

So we now fix the orthogonal matrices \( U \) and \( V \) and show that an optimal choice of \( D \) has \( D_{ii} = \pm 1 \) for all \( i \).

The objective function is

\[
\text{cov} (X, Y) = 2\text{tr} (QA^T RA) + r^T Aq = 2\text{tr} (QV DU^T RU VD^T) + r^T Aq = 2\text{tr} (DU^T RU V^T QV) + r^T UDV^T q.
\]

(7)

Letting \( \bar{D} \) be the \( d \times 1 \) vector with \( i \)th component \( D_{ii} \) we see from (7) that

\[
\text{cov} (X, Y) = \sum_{i,j=1}^{d} D_{ii} D_{jj} (U^T RU)_{ij} (V^T QV)_{ij} + \sum_i (U^T r)_i (V^T q)_i D_{ii} = \bar{D}^T \Gamma \bar{D} + p^T \bar{D},
\]

(8)

where \( \Gamma \) is the Hadamard product of \( U^T RU \) and \( V^T QV \), i.e., the elementwise product of these two matrices, and \( p \) is the Hadamard product of \( U^T r \) and \( V^T q \). The Schur product theorem, e.g., [Horn and Johnson 1985, Theorem 7.5.3], allows us to conclude that \( \Gamma \) is positive semidefinite, since the same is true of \( V^T QV \) and \( U^T RU \). Hence, (8) is a convex quadratic in \( \bar{D} \), with feasible region given by the cube \( \{ -1 \leq \bar{D}_i \leq 1 : i = 1, 2, \ldots, d \} \). An optimal solution then exists that lies at an extreme point, so an optimal \( D \) exists with \( D_{ii} = \pm 1 \) for all \( i \), and this completes the proof. \( \square \)

B. OBTAINING THE GRADIENT FOR THE DIRECT FORMULATION

Our task is to develop an expression for the derivative of (5) in Section 4. Let \( \nabla g_k \) denote the \( k \)th term of the gradient of \( g \), \( k = 1, 2, \ldots, d_Y \). Then we can write

\[
\frac{\partial h}{\partial A_{ij}} = \frac{-2}{n-1} \sum_{m=1}^{n} (X_m - Y_m - M_n) \sum_{k=1}^{d_Y} \nabla g_k (Z_Y^{(m)}(k)) \frac{\partial Z_Y^{(m)}(k)}{\partial A_{ij}},
\]

where \( Z_Y^{(m)} = AN_X^{(m)} + B(A)N_Y^{(m)} \). We can derive \( \partial Z_Y^{(m)}(k)/\partial A_{ij} \) as follows.

First,

\[
\frac{\partial (AN_X^{(m)})_k}{\partial A_{ij}} = N_X^{(m)}(j) \mathbb{1}(i = k).
\]

(9)
The more challenging term to differentiate is the $k$th element of $B(A)N_Y^{(m)}$. That derivative is given by

$$\sum_{l=1}^{d_Y} \frac{\partial B_{kl}(A)}{\partial A_{ij}} N_Y^{(m)}(l),$$

so the problem reduces to determining the derivatives of the Cholesky factor $B$ with respect to each $A_{ij}$. In general this can be performed using automatic differentiation; see Smith [1995]. We briefly present an equivalent explicit approach specialized to our context. The idea is quite straightforward, and essentially involves following the Cholesky algorithm, and accumulating derivatives. For notational convenience, let $\Gamma = I - AA^\top$. The Cholesky algorithm can be written as

for $k = 1$ to $d_Y$

$$B_{kk} = \left( \Gamma_{kk} - \sum_{m=1}^{k-1} B^2_{km} \right)^{1/2}$$

for $l = k + 1$ to $d_Y$

$$B_{lk} = \frac{1}{B_{kk}} \left( \Gamma_{kl} - \sum_{m=1}^{k-1} B_{km}B_{lm} \right)$$

next $l$

next $k$

It follows that we can obtain the matrix $\partial B = \partial B(i, j)$ of derivatives of the entries of $B$ with respect to $A_{ij}$ with the following algorithm.

for $k = 1$ to $d_Y$

$$\partial B_{kk} = \frac{1}{2B_{kk}} \left( \partial \Gamma_{kk} - \sum_{m=1}^{k-1} 2B_{km}\partial B_{km} \right)$$

for $l = k + 1$ to $d_Y$

$$\partial B_{lk} = \frac{1}{B_{kk}} \left( -B_{lk}\partial B_{kk} + \frac{\partial \Gamma_{kl}}{\partial A_{ij}} - \sum_{m=1}^{k-1} [B_{km}\partial B_{lm} + B_{lm}\partial B_{km}] \right)$$

next $l$

next $k$

Finally, it remains to specify how to compute the derivatives $\partial \Gamma_{kl}/\partial A_{ij}$. Let $\partial \Gamma = \partial \Gamma(i, j)$ denote the matrix of such derivatives. Recall that $\Gamma = I - AA^\top$, so by direct calculation we find that

$$\partial \Gamma_{ii} = -2A_{ij},$$
$$\partial \Gamma_{ki} = -A_{kj}, k \neq i,$$
$$\partial \Gamma_{il} = -A_{lj}, l \neq i,$$
$$\partial \Gamma_{kl} = 0, k \neq i, l \neq i.$$  

The quantity $\partial Z_Y^{(m)}(k)/\partial A_{ij}$ is then given by the sum of (9) and (10).
C. THE $L$ MATRIX IN THE LARGER PORTFOLIO INSTANCE

The first 17 columns of $L$ are

\[
\begin{bmatrix}
1.000 \\
0.841 & 0.542 \\
0.118 & 0.666 & 0.736 \\
0.106 & 0.521 & 0.683 & 0.501 \\
0.256 & 0.537 & 0.534 & 0.411 & 0.438 \\
0.373 & 0.544 & 0.329 & 0.045 & 0.231 & 0.634 \\
0.319 & 0.393 & 0.063 & 0.169 & 0.423 & 0.532 & 0.502 \\
0.444 & 0.183 & 0.154 & 0.637 & 0.030 & 0.165 & 0.389 & 0.401 \\
0.023 & 0.386 & 0.058 & 0.278 & 0.516 & 0.426 & 0.228 & 0.456 & 0.250 \\
0.720 & 0.033 & 0.311 & 0.060 & 0.105 & 0.403 & 0.026 & 0.098 & 0.329 & 0.297 \\
0.180 & 0.027 & 0.576 & 0.014 & 0.214 & 0.212 & 0.117 & 0.117 & 0.627 & 0.312 & 0.162 \\
0.149 & 0.027 & 0.013 & 0.206 & 0.148 & 0.396 & 0.473 & 0.272 & 0.320 & 0.020 & 0.171 & 0.571 \\
0.047 & 0.201 & 0.067 & 0.131 & 0.207 & 0.248 & 0.337 & 0.222 & 0.201 & 0.010 & 0.387 & 0.684 & 0.107 \\
0.165 & 0.251 & 0.338 & 0.345 & 0.011 & 0.549 & 0.164 & 0.168 & 0.349 & 0.309 & 0.157 & 0.244 & 0.058 & 0.125 \\
0.003 & 0.254 & 0.207 & 0.303 & 0.432 & 0.047 & 0.571 & 0.061 & 0.129 & 0.005 & 0.080 & 0.034 & 0.142 & 0.308 & 0.381 \\
0.153 & 0.164 & 0.354 & 0.097 & 0.130 & 0.014 & 0.020 & 0.298 & 0.518 & 0.193 & 0.110 & 0.412 & 0.148 & 0.396 & 0.015 & 0.205 \\
0.182 & 0.393 & 0.235 & 0.423 & 0.031 & 0.288 & 0.236 & 0.165 & 0.051 & 0.019 & 0.198 & 0.014 & 0.055 & 0.318 & 0.363 & 0.309 & 0.195 \\
0.273 & 0.294 & 0.091 & 0.423 & 0.188 & 0.018 & 0.335 & 0.096 & 0.132 & 0.541 & 0.266 & 0.174 & 0.055 & 0.152 & 0.051 & 0.007 & 0.121 \\
0.142 & 0.002 & 0.127 & 0.193 & 0.085 & 0.129 & 0.288 & 0.422 & 0.014 & 0.127 & 0.302 & 0.222 & 0.117 & 0.272 & 0.189 & 0.537 & 0.039 \\
0.085 & 0.101 & 0.016 & 0.101 & 0.099 & 0.282 & 0.105 & 0.133 & 0.593 & 0.387 & 0.085 & 0.200 & 0.024 & 0.146 & 0.207 & 0.049 & 0.168
\end{bmatrix}
\]

The last 3 rows and 3 columns of $L$ are

\[
\begin{bmatrix}
0.202 \\
0.131 & 0.231 \\
0.448 & 0.080 & 0.056
\end{bmatrix}
\]

ACKNOWLEDGMENTS

We thank Florian Jarre for supplying optimization code for the direct formulation and helpful exchanges, Adrian Lewis and Mike Todd for discussions about matrix inequalities, and the referees and editors for helpful suggestions that improved the paper. Portions of this work were completed while Shane Henderson was visiting the Mathematical Sciences Institute at the Australian National University, and he thanks them for their hospitality and support.

REFERENCES


Received February 2007; revised March 2009; accepted June 2009