Monte Carlo variance reduction - a new universal method for multivariate problems

Wolfgang M. Schmidt*
Frankfurt School of Finance & Management
Centre of Practical Quantitative Finance
w.schmidt@frankfurt-school.de

Eurobanking, Maribor, May 2008

*Based on joint work with Natalie Packham. Work in progress.

Abstract

Multivariate valuation problems in finance like portfolio value at risk or the pricing of complex structured products often require Monte Carlo simulation. To reduce the variance of the desired simulation results a variety of variance reduction techniques are available. Some of them, like, e.g., importance sampling or antithetic sampling, rely on specific properties of the problem at hand.

Sampling from a multi-dimensional distribution with arbitrary dependence structure (copula) it is far from trivial how to generate simulations which reproduce the given distribution as closely as possible in sample.

We propose a universal new variance reduction technique for high dimensional problems which can be seen as a variant of Latin hypercube sampling or stratified sampling allowing for dependent random variables.

1 Monte Carlo Simulation in Finance

- Risk assessment for portfolio of financial claims, e.g., value at risk; Valuation of complex financial claims $\rightsquigarrow$ Monte Carlo Simulation
  - Generate random sample paths of the evolution of underlying financial securities and calculate corresponding values of portfolio/claims
  - Estimate portfolio quantile or value of payoff by expectation/ sample mean of portfolio/claim's payoff
  - Result is still an random quantity, which stabilizes as the number of simulations increase
- Require estimator to be unbiased and consistent
- Variance of estimator is key figure for rate of convergence

  $\rightsquigarrow$ Variance reduction techniques

- Here: Portfolios/claims
  - that depend on several underlying financial securities
  - that are sensitive to dependence structure of underlyings

- Examples: Value at risk, valuation of CDO tranches, First-to-default credit baskets

2 Variance Reduction Methods

Variance reduction methods aim to reduce the variance of the estimator. Here are the most popular methods:

- Antithetic variates:
  Base simulation on two sets of random drawings that are negatively correlated; e.g., in case of standard normals use X and $-X$
- Control variates:
  Simultaneously and in addition simulate another claim whose value is already known but that is highly correlated with the variable of interest
- Importance sampling:
  Generate sample paths under a distribution that is different from the intended

one, but emphasizes the “important” values. Correct for the differing distribution by Radon-Nikodym derivative. Application is highly problem dependent.

- Stratified sampling:
  Ensure that the simulations are grouped properly in relatively homogeneous mutually exclusive subgroups (strata).

3 Stratified Sampling

Suppose the goal is to estimate $\mathbf{E}g(U)$ with $U \sim U(0,1)$. Let $A_1, \ldots, A_n$ be a partition of $[0,1]$. Then,

$$\mathbf{E}g(U) = \sum_{i=1}^{n} \mathbf{E}(g(U)|U \in A_i)\mathbf{P}(U \in A_i). \tag{1}$$

In the simplest case, a stratified estimator is obtained by choosing equiprobable strata $A_i = (i - 1/n, i/n]$, $i = 1, \ldots, n$, and drawing one sample from each stratum, e.g., let $U_1, \ldots, U_n$ be independent samples drawn from a $U(0,1)$

distribution and set

$$
\begin{aligned}
V_i &= \frac{i-1}{n} + \frac{U_i}{n}, \quad i = 1, \ldots, n \\
\widehat{\mathbf{E}g(u)} &= 1/n \sum_{i=1}^{n} g(V_i).
\end{aligned}
$$

A detailed description can be found in [1, Chapter 4.3].

Simply extending stratified sampling to d-dimensional random vectors by stratifying each dimension with n samples requires n^d samples to have at least one sample in each stratum, which is unfeasible for large dimensions d or fine strata.

4 Multivariate Stratification – Latin Hypercube Sampling

Latin hybercube sampling (LHS) efficiently extends stratified sampling into multiple dimensions.

4.1 Independent dimensions

Assume that the goal is to sample a d-dimensional random vector $(U^1, \ldots, U^d)$ of independent uniform random variables. Fixing a sample size n, generate n independent samples $(U_i^1, \ldots, U_i^d)$, $i = 1, \ldots, n$, and generate d independent permutations $\pi^1, \ldots, \pi^d$ of $\{1, \ldots, n\}$ drawn from the distribution that makes all permutations equally probable. Denoting by π_i^j the value to which i is mapped by the j-th permutation, a Latin hypercube sample is given by

$$V_i^j := \frac{\pi_i^j - 1}{n} + \frac{U_i^j}{n}, \quad j = 1, \ldots, d, \quad i = 1, \ldots, n.$$

In each dimension j, $(V_1^j, \ldots, V_n^j)$, is a stratified sample. Each point $(V_i^1, \ldots, V_i^d)$, $i \in \{1, \ldots, n\}$, is uniformly distributed on $[0,1]^d$.

LHS depends on the independence assumption of the random vector involved.

Applying LHS to a sample of a random vector whose components are dependent will destroy the dependence by application of random and independent permutations in each dimension. Conversely, applying LHS to a sample of a random vector with independent components and then a transform to introduce dependence will, in general, result in a sample whose marginals are not stratified, thereby losing much of the appeal of LHS.

An in-depth treatment of LHS is found in [1, Section 4.4].

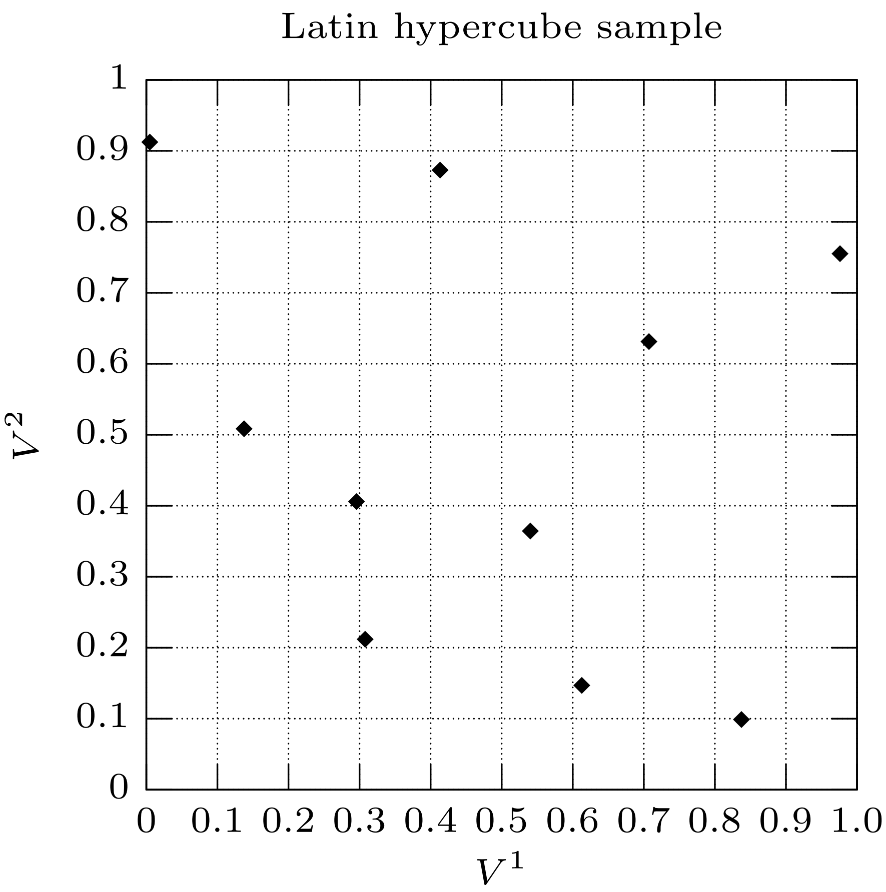


Fig. 1. Latin hypercube sample with 10 samples stratified in each dimension.

4.2 Sampling with dependencies

We are looking for an LHS-like efficient sampling strategy for random vectors $(U^1, \ldots, U^d)$ whose components are no longer independent but admit a certain dependence structure(copula).

The general idea is to generate a Latin hypercube sample, albeit with the following modification: instead of choosing a random permutation in each dimension, we choose a particular permutation that depends on the samples of that dimension. For this we need the notion of a rank statistic.

Definition 1. [Rank statistic] *Let $X_1, \ldots, X_n$ be i.i.d. random variables with continuous distribution function. Reorder them such that $X^{(1)} < \cdots < X^{(n)}$. The index of X_i within $X^{(1)}, \ldots, X^{(n)}$ is the i-*th rank statistic*, given by*

$$r_{i,n}(X_1, \ldots, X_n) := \sum_{k=1}^{n} \mathbf{1}_{\{X_k \leq X_i\}}. \tag{2}$$

Let $(U_i^1, \ldots, U_i^d)$, $i = 1, \ldots, n$, be n independent samples of the random vector $(U^1, \ldots, U^d)$ with $U(0,1)$ marginals. For any $i \in \{1, \ldots n\}$ and any dimension j denote by $r_{i,n}^j$ the i-th rank statistic of $(U_1^j, \ldots, U_n^j)$. A Latin hypercube sample is given by

$$V_i^j := \frac{r_{i,n}^j - 1}{n} + \frac{U_i^j}{n}, \quad i = 1, \ldots, n, \quad j = 1, \ldots, d.$$

Just as in regular LHS, $(V_1^j, \ldots, V_n^j)$ is a stratified sample in each dimension j. However, through use of the rank statistic we lose the property that V^j is uniformly distributed within each stratum. Conditional on $\{r_{i,n}^j = k\}$, U_i^j follows a beta distribution with parameters k and n, $\mathbf{P}(U_i^j \leq x | r_{i,n}^j = k) = B_k^n(x)$ (this is the distribution of the k-th order statistic of n independent uniform random variables.

Either one of the following two transformations provides a LHS with uniform marginals in each stratum:

$$V_i^j := \frac{r_{i,n}^j - 1}{n} + \frac{B^n_{r_{i,n}^j}(U_i^j)}{n}, \tag{3}$$

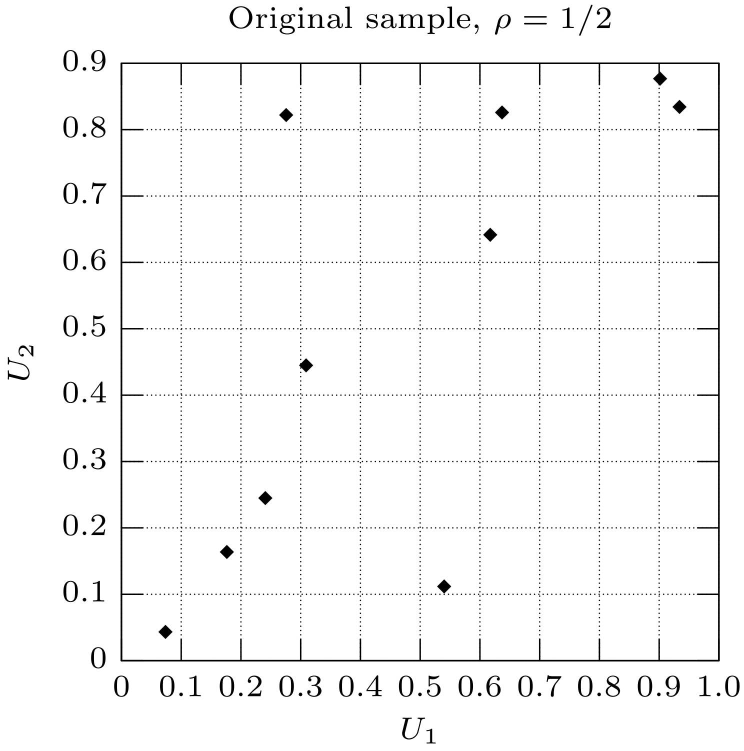
$$V_i^j := \frac{r_{i,n}^j - 1}{n} + \frac{W_i^j}{n}, \tag{4}$$

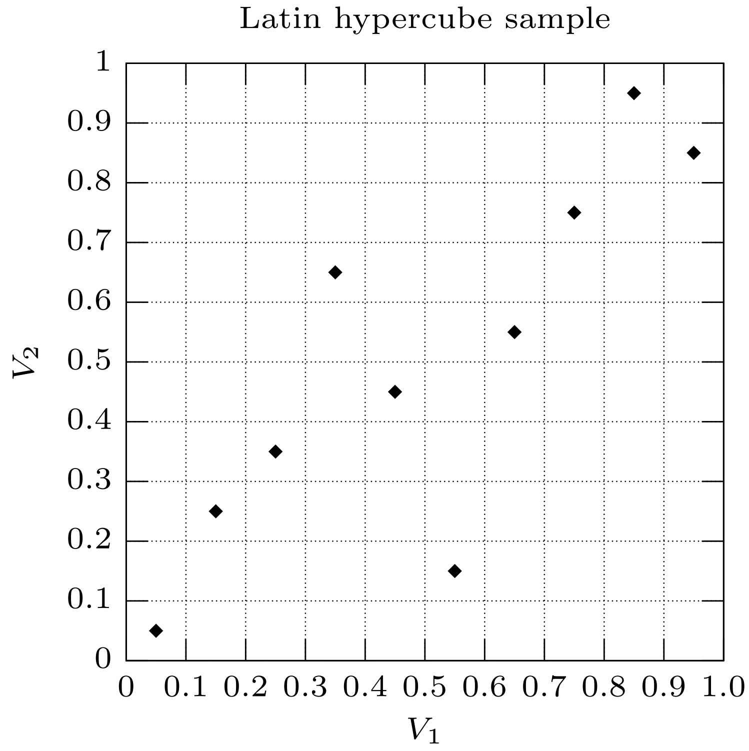
$i = 1, \ldots, d, \quad j = 1, \ldots, n,$ with $(W_i^j)_{i=1,\ldots,d; j=1,\ldots n}$ independent $U(0,1)$ samples independent of $(U_i^j)_{i=1,\ldots,d; j=1,\ldots n}$.

If the primary goal is to capture the joint distribution, we may use the following strategy, which just places each sample in the middle of its stratum and is least CPU intensive:

$$V_i^j := \frac{r_{i,n}^j - 1/2}{n}, \quad i = 1, \ldots, d, \quad j = 1, \ldots, n. \tag{5}$$

The figure shows an example with 10 samples sampled from a bivariate distribution that are linked through a Gaussian copula with a correlation of 0.5 and the corresponding samples after applying the transformation from (5).





We now turn to analysing the asymptotic behaviour of the estimator that is based on the modified simulations. Given n independent samples $(U_i^1, \ldots, U_i^d)$, $i = 1, \ldots, n$, of $(U^1, \ldots, U^d)$, the corresponding LHS estimator for $\mathbf{E}g(U^1, \ldots, U^d)$ with dependence is

$$\frac{1}{n} \sum_{k=1}^{n} g(V_{i,n}^1, \ldots, V_{i,n}^d), \tag{6}$$

with $V_{i,n}^j$, $i = 1, \ldots, n$, $j = 1, \ldots, d$, obtained from one of the transformations (3), (4) or (5). We emphasise that the usual laws of large numbers do not apply:

- By application of the rank statistic, the samples in each dimension are not independent anymore.
- For any i, j, $V_{i,n}^j \neq V_{i,n+1}^j$, hence, when progressing from n to $n + 1$, we are

not just adding an $(n+1)$-th term to the existing sum, but all terms of the sum change.

Proposition 1. *Let $g : (0,1)^d \to \mathbb{R}$ be bounded and continuous at $(U^1, \ldots, U^d)$. Then, the estimator given by* (6) *is consistent, i.e.,*

$$\frac{1}{n}\sum_{i=1}^{n} g(V_{i,n}^1, \ldots, V_{i,n}^d) \xrightarrow{\mathbf{P}} \mathbf{E}g(U^1, \ldots, U^d), \qquad \text{as } n \to \infty. \tag{7}$$

It follows immediately by Dominated Convergence that the estimator is asymptotically unbiased:
Corollary 1. *Let $g : (0,1)^d \to \mathbb{R}$ be as in Proposition 1. Then, the estimator*

given by (6) is asymptotically unbiased, i.e.,

$$\mathbf{E}\left(\frac{1}{n}\sum_{i=1}^{n} g(V_{i,n}^1,\ldots,V_{i,n}^d)\right) \longrightarrow \mathbf{E}g(U^1,\ldots,U^d), \qquad \textit{as } n \to \infty.$$

Proposition 2. *Let $g : (0,1)^d \to \mathbb{R}$ be uniformly continuous. Then, the estimator given by (6) is strongly consistent, i.e.,*

$$\frac{1}{n}\sum_{i=1}^{n} g(V_{i,n}^1,\ldots,V_{i,n}^d) \longrightarrow \mathbf{E}g(U^1,\ldots,U^d), \qquad \textit{as } n \to \infty.$$

Intuitively, the results state that in the limit the proposed LHS technique does not destroy the joint distribution.

5 Sample Application

We demonstrate the effectiveness of LHS with dependence by valuing a first-to-default credit basket (FTD). An FTD is a contract between two counterparties, a protection buyer and a protection seller, that insures the protection buyer against the loss incurred by the first default event in a portfolio of some underlying risky entities over a fixed time horizon. The protection buyer regularly pays a constant premium, called the *spread*. In turn, the protection buyer compensates the protection buyer for the loss incurred by the first default event at the time of default.

With each credit $j = 1, \ldots, d$ of the underlying portfolio we associate the random default time τ_j and the known recovery rate R_j. Furthermore, we assume given the default distributions $F_j(t) = \mathbf{P}(\tau_j \leq t)$, $t \geq 0$, $j = 1, \ldots, d$, which can be derived from the quotes in the credit default swap (CDS) market.

Denote the time of the FTD's default event by $\tau := \min(\tau_1, \ldots, \tau_d)$. For $t \geq 0$ let B_t denote today's default-free zero bond price with maturity t. The fair spread s of the FTD is then obtained by equating the expected value of the premium and the protection leg,

$$s \sum_{k=1}^{K} \Delta_{t_k} B_{t_k} \mathbf{P}(\tau > t_k) = \sum_{j=1}^{d} (1 - R_j) \int_0^T B_u \mathbf{P}(\tau \in \mathrm{d}u, \tau = \tau_j). \tag{8}$$

From this equation and from $\mathbf{P}(\tau \leq t) = 1 - \mathbf{P}(\tau_1 > t, \tau_2 > t, \ldots, \tau_d > t)$ it is clear that value of the FTD depends on the joint distribution of $\tau_1, \ldots, \tau_d$. In our example we assume that the joint distribution of the default times $\tau_1, \ldots, \tau_d$ is driven by a Normal copula (Gaussian copula),

$$\mathbf{P}(\tau_1 \leq t_1, \ldots, \tau_d \leq t_d) = \mathrm{N}_\Sigma \left(\mathrm{N}^{(-1)}(F_1(t_1)), \ldots, \mathrm{N}^{(-1)}(F_d(t_d)) \right),$$

with N_Σ the multivariate standard normal distribution function with correlation matrix Σ and $\mathrm{N}^{(-1)}$ the inverse of the univariate standard normal distribution function.
The input parameters for an example involving 5 homogeneous credits are

Parameter	Value
Maturity	$T = 5$ (years)
spread payment dates (frequency)	$(t_k)_{k=1,\ldots,K}$ (quarterly)
Default-free zero bond prices	$B_t = \mathbf{e}^{-.05t}$, $t \geq 0$
Number of underlying credits	$d = 5$
5yr.-CDS spread of each credit	$s_j = 1\%$, $j = 1, \ldots, d$
Recovery rate of each credit	$R_j = 0.3$, $j = 1, \ldots, d$
Correlation between any two credits	$\rho = 30\%$

The fair spread of the FTD is 417.99 bps The FTD spread was computed from simulations using random numbers and using low discrepancy sequences, both "as is" and adding a LHS step. This leads to the following four simulation scenarios:

1. Standard Monte Carlo simulation
2. LHS with dependence based on random numbers
3. Simulation with low discrepancy sequence
4. LHS with dependence based on low discrepancy sequence

Mean square error estimates were obtained by simulating each estimator 100 times. The MSE estimates and MSE ratios for various samples sizes are given in the table below. The ratios of CPU time consumed for generating samples with and without LHS is also shown for various sample sizes. It should be noted that the CPU time ratios do not include the CPU time required for computing the

FTD payoff. Adding computation steps that are equal for all methods reduces these ratios. The LHS step requires that the sequence be sorted, hence the computational overhead of the LHS step is of the order of the sorting algorithm. On the other hand, computing the rank statistics can be done from samples of any distribution, whereas in a typical simulation the e.g. normally distributed samples must additionally be transformed to uniforms.

No. of simulations ($\times 10^3$)	200	400	600	800	1000
MC	4.06421	2.16733	1.21682	0.80025	0.65415
LHS	1.00806	0.37867	0.28097	0.20744	0.15365
Sobol	0.09865	0.04820	0.03352	0.02753	0.01761
Sobol + LHS	0.05204	0.02194	0.02074	0.01537	0.01368
MC/LHS	4.0317	5.7236	4.3308	3.8578	4.2573
Sobol/(Sobol + LHS)	1.8957	2.1966	1.6161	1.7917	1.2873
CPU time LHS/MC	1.6561	1.7083	1.7535	1.7817	1.8185
CPU time (Sobol + LHS)/Sobol	1.4659	1.5205	1.5360	1.5495	1.5594

References

[1] GLASSERMAN, P.: *Monte Carlo Methods in Financial Engineering*, Springer, 2004 7, 9

[2] PACKHAM, N. AND SCHMIDT, W.: Latin hypercube sampling with dependence, Working paper in progress, 2008