High Dimensional Integration in Option Pricing Models

Naoki Sakai, Max Brodeur

January 2024

1 Introduction

In financial mathematics, there are functions which have "kinks" or "jumps" which hinder our capabilities in estimating the Quasi Monte Carlo method's error. The work done in [3] dealt with a special case of this problem, using a method of preintegration. The studied problem consists an integrand $f : \mathbb{R}^m \to \mathbb{R}$ with smooth function θ and ϕ, written in the following form.

$$f(\mathbf{x}) = \theta(\mathbf{x})\mathbb{I}_{\{\phi(\mathbf{x})\}}$$

Then, if the variables $\mathbf{x}$ are independent, and if there exists x_j such that

- $\frac{\partial \phi}{\partial x_j}(\mathbf{x}) > 0 \, \forall \mathbf{x}$
- $\phi(\mathbf{x}) \to \pm\infty$ as $x_j \to \pm\infty$

we can find a unique value $x_j = \psi(\mathbf{x}_{-j})$ such that $\phi(\mathbf{x}) = 0$, where we can convert the high dimensional integration problem to

$$\int_{\mathbb{R}^m} f(\mathbf{x})p(\mathbf{x})d\mathbf{x} = \int_{\mathbb{R}^{m-1}} \int_{\psi(\mathbf{x}_{-j})}^{\infty} \theta(x_j, \mathbf{x}_{-j})p_j(x_j)dx_j p_{-j}(\mathbf{x}_{-j})d\mathbf{x}_{-j}$$

Since ψ and θ are smooth, the integrand for outer integration might be smooth and we can apply QMC to the outer integral. We will apply this idea to the following option pricing problem and furthermore, use variance reduction techniques to improve our results.

1.1 Option pricing

The integral in question is

$$V_i := e^{-rT}\mathbb{E}\left[\Psi_i(\mathbf{w})\right] = \frac{e^{-rT}}{(2\pi)^{m/2}\sqrt{\det C}} \int_{\mathbb{R}^m} \Psi_i(\mathbf{w}) e^{-\frac{1}{2}\mathbf{w}^\top C^{-1}\mathbf{w}} \, d\mathbf{w}$$

where, in order of appearance we have

- r: the interest rate,
- T: the maturity of the option,
- Ψ_i: the Asian option payoffs (defined below),
- $\mathbf{w} = (w_{t_1}, w_{t_2}, \ldots, w_{t_m})$: the stock prices at times $t_i = i\Delta t$ with $\Delta t = \frac{T}{m}$, and
- C: the covariance matrix of $\mathbf{w}$ defined by $C_{i,j} = \min\{t_i, t_j\}$.

The Asian option payoffs are defined as follows:

$$\Psi_1(\mathbf{w}) := \phi(\mathbf{w})\mathbb{I}_{\{\phi(\mathbf{w})\}} \qquad \text{(Asian call option)}$$
$$\Psi_2(\mathbf{w}) := \mathbb{I}_{\{\phi(\mathbf{w})\}} \qquad \text{(binary digital Asian option)}$$

where $\phi(\mathbf{w}) := \frac{1}{m}\sum_{i=1}^{m} S_{t_i}(w_{t_i}) - K$ and

- S_{t_i} is the stock price at time t_i
- K is the strike price

We model the stock price S with a stochastic differential equation

$$dS = rSdt + \sigma S dw_t, \quad S(0) = S_0$$

and find that it yields the solution

$$S_t = S_0 \exp\left(t\left(r - \frac{\sigma^2}{2}\right) + \sigma w_t\right)$$

2 Transformations to $\mathcal{N}(0,1)$ and recast to unit m-cube

First of all, $\mathbf{w}$ does not have independent components. To be able to separate the integral, we must execute a change of variables making the integrated components independent from each other.

2.1 Cholesky factorization

The first transformation of $\mathbf{w}$ to obtain independent standard normal variables is by Cholesky factorization of the covariance matrix: $C = AA^\top$. Indeed, by applying the change of variables $\mathbf{w} = A\mathbf{x}$ we obtain independent random variables $\mathbf{x} = (x_{t_1}, x_{t_2}, \ldots, x_{t_m})^\top$ with $x_{t_i} \sim \mathcal{N}(0,1)$, resulting in the following integral

$$\begin{aligned} V_i &= \frac{e^{-rT}\det A}{(2\pi)^{m/2}\sqrt{\det C}} \int_{\mathbb{R}^m} \Psi_i(A\mathbf{x}) e^{-\frac{1}{2}\mathbf{x}^\top A^\top C^{-1} A\mathbf{x}}\, d\mathbf{x} \\ &= \frac{e^{-rT}\det A}{(2\pi)^{m/2}\sqrt{\det C}} \int_{\mathbb{R}^m} \Psi_i(A\mathbf{x}) e^{-\frac{1}{2}\mathbf{x}^\top \mathbf{x}}\, d\mathbf{x} \end{aligned}$$

since

$$\begin{aligned} (A\mathbf{x})^\top C^{-1}(A\mathbf{x}) &= \mathbf{x}^\top A^\top C^{-1} A\mathbf{x} \\ &= \mathbf{x}^\top A^\top (AA^\top)^{-1} A\mathbf{x} \\ &= \mathbf{x}^\top A^\top A^{-\top} A^{-1} A\mathbf{x} \\ &= \mathbf{x}^\top \mathbf{x} \end{aligned}$$

Also,

$$\det(C) = \det(AA^\top) = \det(A)\det(A^\top) = (\det A)^2$$

Therefore, $\det A/\sqrt{\det C} = 1$. So we get

$$V_i = \frac{e^{-rT}}{(2\pi)^{m/2}} \int_{\mathbb{R}^m} \Psi_i(A\mathbf{x}) e^{-\frac{1}{2}\mathbf{x}^\top \mathbf{x}}\, d\mathbf{x}$$

This result makes sense intuitively, since these new $\mathbf{x}$ variables are independent, and so have an identity covariance matrix $C = I$. The above formula echoes that of an m-dimensional standard normal distribution, indicating that this transformation is indeed valid.
Furthermore, under this transformation we find that the stock price model evaluates to the following

$$\begin{aligned} \phi(A\mathbf{x}) &= \frac{1}{m}\sum_{i=1}^{m} S_{t_i}\left(\sum_{j=1}^{m} A_{ij}\mathbf{x}_j\right) - K \\ &= \frac{S_0}{m}\sum_{i=1}^{m} \exp\left(t_i\left(r - \frac{\sigma^2}{2}\right) + \sigma\sum_{j=1}^{m} A_{ij}\mathbf{x}_j\right) - K \end{aligned}$$

2.2 Gaussian increments

From the definition of Wiener process, we can make independent variables in the following way:

$$\xi_i = \frac{w_{t_i} - w_{t_{i-1}}}{\sqrt{t_i - t_{i-1}}}, \quad i = 1, \dots, m$$

where ξ_i are independently drawn from a standard normal distribution. Then, by rearranging the terms we get

$$\begin{aligned} w_{t_i} &= w_{t_{i-1}} + \xi_i \sqrt{t_i - t_{i-1}} \\ &= \sum_{j=1}^{i} \xi_j \sqrt{t_j - t_{j-1}} + w_{t_0} \\ &= \sum_{j=1}^{i} \xi_j \sqrt{\Delta t} \end{aligned}$$

In matrix form, we can express $\mathbf{w} = B\xi$ with

$$B = \begin{bmatrix} 1 & 0 & 0 & \cdots & 0 \\ 1 & 1 & 0 & \cdots & 0 \\ 1 & 1 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & 1 & 1 & \cdots & 1 \end{bmatrix}$$

Now, we get that

$$\begin{aligned} C &= \mathbb{E}[\mathbf{w}\mathbf{w}^\top] \\ &= \mathbb{E}[B\xi\,\xi^\top B^\top] \\ &= B\mathbb{E}[\xi\,\xi^\top]B^\top \\ &= BIB^\top \\ &= BB^\top \end{aligned}$$

Combined with the fact that B is a lower triangular matrix, we can say B is exactly the same matrix as the one obtained by Cholesky, since because the Cholesky decomposition is unique.

2.3 Lévy-Ciesielski construction

The Lévy-Ciesielski construction gives the following approximation for our Wiener process values, using independent standard random variables:

$$w_N(t) = X_0 \eta_0(t) + \sum_{n=1}^{N} \sum_{i=1}^{2^{n-1}} X_{n,i} \eta_{n,i}(t)$$

where

- w_N denotes the approximated Wiener process using 2^N variables, with $N \in \mathbb{N}, N > 0$.
- X_0 and $X_{i,n} \sim \mathcal{N}(0,1)$ are the *i.i.d.* random variables.
- $\eta_0(t) := t$ and $\eta_{n,i}(t)$ are time dependent coefficients given by

$$\eta_{n,i}(t) := \begin{cases} 2^{(n-1)/2} \left(t - \frac{2i-2}{2^n}\right), & t \in \left[\frac{2i-2}{2^n}, \frac{2i-1}{2^n}\right] \\ 2^{(n-1)/2} \left(\frac{2i}{2^n} - t\right), & t \in \left[\frac{2i-1}{2^n}, \frac{2i}{2^n}\right] \\ 0, & \text{o.w.} \end{cases}$$

The random variables $X_{n,i}$ are represented as a flattened vector of independent random variables $\mathbf{x}$ with $X_k := X_{n,i}$ given by the change of indices: $k = 2^n + i$.

$$\mathbf{x} = (X_0, X_{1,1}, X_{2,1}, X_{2,2}, X_{3,1}, X_{3,2}, X_{3,3}, X_{3,4}, \dots, X_{N,1}, \dots, X_{N,2^{N-1}})^\top$$

$$= (X_1, \ldots, X_{2^N})^\top$$

Now, we are approximating the Wiener process $\mathbf{w}$ by $\mathbf{w}_N = \eta\mathbf{x}$, where η is an $m \times 2^N$ matrix of coefficients per each time step value, and $\mathbf{w_N}, \mathbf{x}$ are column vectors of length 2^N. In contrast to a change of variables like Cholesky, we have $\mathbf{w} \neq \mathbf{w}_N$. Therefore we integrate with a new covariance matrix C_N:

$$\begin{aligned} C_N &= \mathbb{E}\left[\mathbf{w}_N \mathbf{w}_N^\top\right] \\ &= \mathbb{E}\left[\eta\mathbf{x}\,\mathbf{x}^\top \eta^\top\right] \\ &= \eta\mathbb{E}\left[\mathbf{x}\,\mathbf{x}^\top\right]\eta^\top \\ &= \eta I \eta^\top \\ &= \eta\,\eta^\top \end{aligned}$$

where we have used the fact that the covariance matrix of an independent normal vector is $\mathbb{E}[\mathbf{x}\,\mathbf{x}^\top] = I$. It follows that $\sqrt{\det C_N} = \det \eta$. Furthermore, by setting $N = log_2(m)$ we keep the dimensionality of our integration and thus preserve the its computational complexity. All of this leads to the following approximation of our integral:

$$\begin{aligned} V_i^{(N)} &= \frac{e^{-rT}}{(2\pi)^{m/2}\sqrt{\det C_N}} \int_{\mathbb{R}^m} \Psi_i(\eta\mathbf{x}) e^{-\frac{1}{2}\mathbf{x}^\top \eta^\top C_N^{-1} \eta\mathbf{x}} \det\eta\, d\mathbf{x} \\ &= \frac{e^{-rT}}{(2\pi)^{m/2}} \int_{\mathbb{R}^m} \Psi_i(\eta\mathbf{x}) e^{-\frac{1}{2}\mathbf{x}^\top\mathbf{x}}\, d\mathbf{x} \end{aligned}$$

As in the Cholesky transformation, the form of our approximated integral is that of an m-dimensional standard normal distribution once again indicating that this transformation is indeed valid. The work in [1] derived an upper bound of the error of $\mathbf{w}_N$ which is $\mathcal{O}(\sqrt{N/2^N})$.

2.4 Recast

Recasting the integral to the unit m-cube is done by inverting the CDF of the standard normal distribution: $\Phi^{-1}(\mathbf{y}) : [0,1]^m \to \mathbb{R}^n$. The change of variables yields

$$\begin{aligned} \mathbf{x} &= \Phi^{-1}(\mathbf{y}) \\ \Phi(\mathbf{x}) &= \mathbf{y} \\ \mathrm{d}\Phi(\mathbf{x}) &= \mathrm{d}\mathbf{y} \\ \Phi'(\mathbf{x})\mathrm{d}\mathbf{x} &= \mathrm{d}\mathbf{y} \\ \implies \mathrm{d}\mathbf{y} &= \frac{1}{(2\pi)^m} e^{-\frac{1}{2}\mathbf{x}^\top\mathbf{x}} \mathrm{d}\mathbf{x} \end{aligned}$$

where $\Phi'(\mathbf{x})$ gives the multivariate standard normal distribution density. Finally, since the resulting integrals of the Cholesky and Lévy-Ciesielski transformations only differ by their matrices, we can perform the recast as follows, using the variable M as a placeholder for the transformation matrices A and η:

$$V_i = e^{-rT} \int_{[0,1]^m} \Psi_i(M\Phi^{-1}(\mathbf{y}))\, d\mathbf{y}$$

$$V_1 = e^{-rT} \int_{[0,1]^m} \phi(M\Phi^{-1}(\mathbf{y})) \mathbb{I}_{\{\phi(M\Phi^{-1}(\mathbf{y}))\}}\, d\mathbf{y}$$

$$V_2 = e^{-rT} \int_{[0,1]^m} \mathbb{I}_{\{\phi(M\Phi^{-1}(\mathbf{y}))\}} d\mathbf{y}$$

Which is precisely the form we will use for the Monte Carlo approximations.

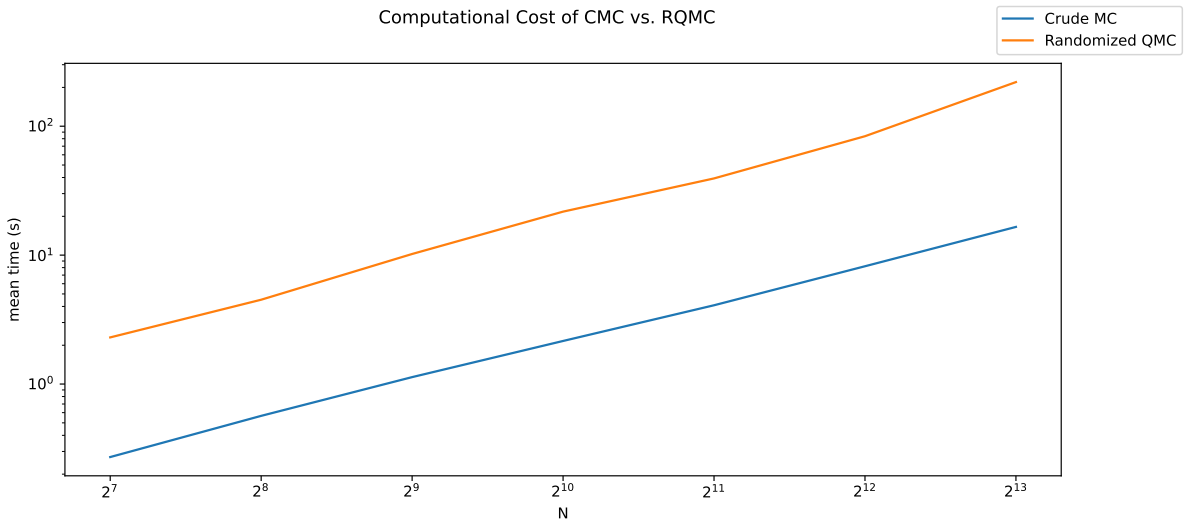

Fig. 1: The computational cost, in seconds, of the Crude Monte Carlo method compared to the Randomized QMC method.

3 Implementation

3.1 Randomized QMC

We will use $K = 10$ point sets of N points for Randomized QMC (RQMC). In fact, as shown in Figure 1, RQMC consistently takes K times longer to simulate.

3.2 Preintegration

The Monte Carlo approximation of the transformed, recast and preintegrated equation yields the following approximated value:

$$V_i \approx e^{-rT} \frac{1}{N} \sum_{n=1}^{N} \int_{y^*_{j(n)}}^{1} \Psi_i(M\Phi^{-1}(y_j, \mathbf{y}^{(n)}_{-j}))\, dy_j$$

where $\mathbf{y}^{(n)}_{-j} \in [0,1]^{m-1}$ are the N random normal vector samples with the j-th dimension removed, M is the transformation matrix and y^*_j is the root of ϕ, *i.e.*

$$\phi(y^*_{j(n)}, \mathbf{y}^{(n)}_{-j}) := \phi(M\Phi^{-1}(\mathbf{y}^{(n)})) = 0$$

where $\mathbf{y}^{(n)}$ is the vector consisting of $\mathbf{y}^{(n)}_{-j}$ and $y^*_{j(n)}$ in their corresponding dimensions. Now, the implementation of this pre-integration trick introduces two difficulties: for each sampled vector $\mathbf{y}^{(n)}_{-j}$, we must compute a root and an integral.

For a fixed sample $\mathbf{y}_{-j}$, computing the root of $\phi(y_j, \mathbf{y}_{-j})$ isn't as straightforward as it may seem. The problem is that the value $y^*_j \in [0,1]$ is arbitrarily close to 0, and often well passed machine precision, as shown in Figure 2. Therefore we can't use, nor find the value of y^*_j. To circumvent this issue, we opted to manipulate the root before the unit m-cube recast, since we only need the integrated variables to be independent (and recast) to separate the j-th integral. In other words, we find the root of the following function instead.

$$\phi(x_j, \mathbf{x}^{(n)}_{-j}) := \phi(M\mathbf{x}^{(n)})$$

For a fixed sample $\mathbf{x}^{(n)}_{-j}$, where $x_j, \mathbf{x}_{-j}, \mathbf{x}^{(n)}$ have analogous definitions to their $\mathbf{y}$ counterparts (recall that $\mathbf{y_{-j}} = \Phi(\mathbf{x_{-j}})$).

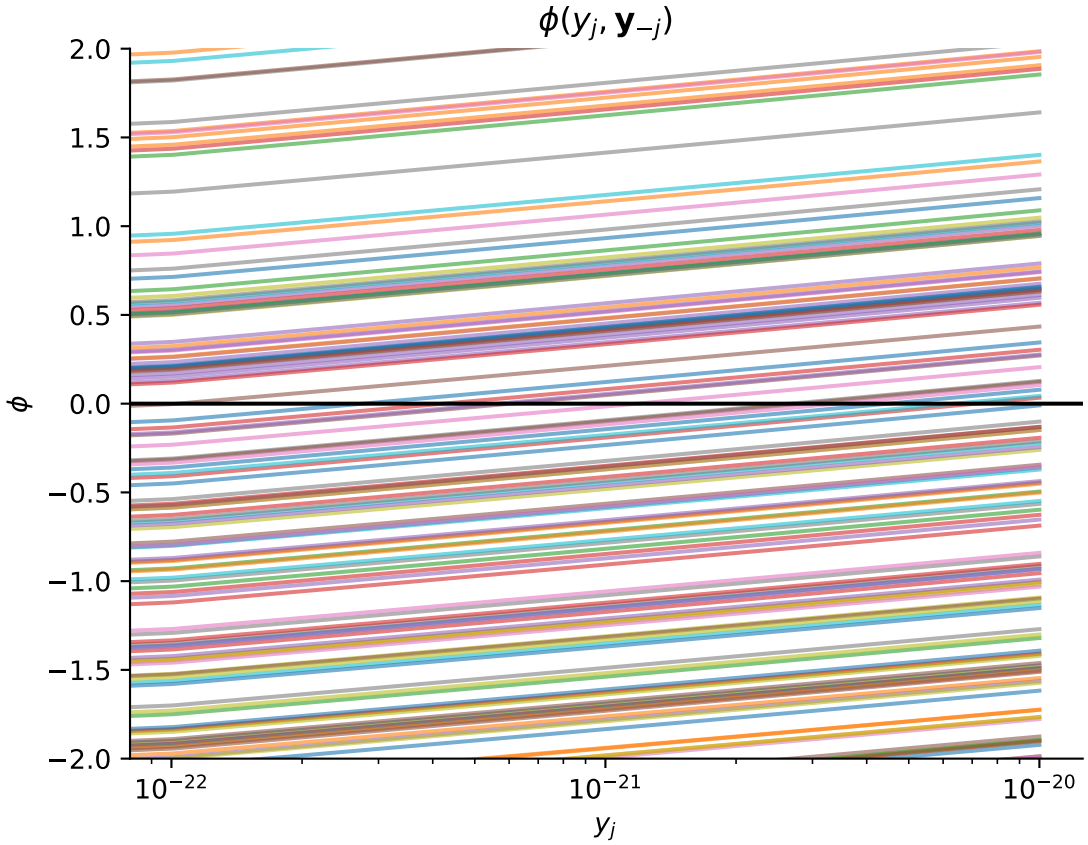

Fig. 2: The function ϕ for multiple $\mathbf{y}_{-j}$ samples (colored lines) plotted against small values of y_j. Many of the samples yield values of ϕ with roots much smaller than the lowest plotted y_j value.

Then, we can approximate the root using a standard Newton method (we used scipy's Fortran API) by providing the first derivative given by

$$
\begin{aligned}
\frac{\partial \phi}{\partial x_k}(x_k, \mathbf{x_{-k}}) &= \frac{\partial \phi}{\partial x_k}(M\mathbf{x}) \\
&= \frac{\partial}{\partial x_k}\left(\frac{1}{m}\sum_{i=1}^{m} S_{t_i}\left(\sum_{j=1}^{m} A_{ij}\mathbf{x}_j\right) - K\right) \\
&= \sum_{i=1}^{m} \frac{\partial}{\partial x_k}\left(S_{t_i}\left(\sum_{j=1}^{m} A_{ij}\mathbf{x}_j\right)\right) \\
&= \frac{1}{m}\sum_{i=1}^{m} S_{t_i}\left(\sum_{j=1}^{m} M_{ij}\mathbf{x}_j\right) \cdot \sigma M_{ik} \\
&= \frac{\sigma}{m}\begin{bmatrix} S_{t_1}\left(\sum_{j=1}^{m} M_{1j}\mathbf{x}_j\right) & \dots & S_{t_m}\left(\sum_{j=1}^{m} A_{mj}\mathbf{x}_j\right)\end{bmatrix}\begin{bmatrix} M_{1k} \\ \vdots \\ M_{mk}\end{bmatrix} \\
&= \frac{\sigma}{m} S(A\mathbf{x})^{\top}\begin{bmatrix} M_{1k} \\ \vdots \\ M_{mk}\end{bmatrix}
\end{aligned}
$$

From this, it is possible to compute the root x_j^*, giving a lower bound to the integral

$$\int_{x_j^*}^{\infty} \Psi_i(M\mathbf{x}^{(n)}) \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}x_j^2}\, dx_j$$

The problem now is that of approximating a semi-definite integral. To do so we can recast back to the m-unit cube. However, inverting the gaussian CDF gives an approximation of the arbitrarily low value of y_j^*. Since its arbitrary, this approximated value will most probably lead to the integration over the kink/jump and in turn badly affecting our results. Instead, the following nifty change of variables trick let's us recast to the m-unit cube while still manipulating our well behaved x_j^* value: $x_j \mapsto x_j^* + \frac{t}{1-t}$.

$$= \int_0^1 \Psi_i \left(A \begin{bmatrix} x_1^{(n)} \\ \vdots \\ x_j^* + \frac{t}{1-t} \\ \vdots \\ x_m^{(n)} \end{bmatrix} \right) \frac{\exp(-\frac{1}{2}(x_j^* + \frac{t}{1-t})^2)}{\sqrt{2\pi}(1-t)^2}\, dt$$

Beautiful! Now, our assumption that $\frac{\partial \phi}{\partial x_j} > 0$ makes it such that we can remove the indicator function $\mathbb{I}_{\{\phi>0\}}$ from our integrands of both Ψ_i, and we approximate the following integrals via Gaussian quadrature:

$$V_1 \approx e^{-rT} \sum_{n=1}^{N} \int_0^1 \phi \left(M \begin{bmatrix} \Phi^{-1}(y_1^{(n)}) \\ \vdots \\ x_j^{*(n)} + \frac{t}{1-t} \\ \vdots \\ \Phi^{-1}(y_m^{(n)}) \end{bmatrix} \right) \frac{\exp(-\frac{1}{2}(x_j^{*(n)} + \frac{t}{1-t})^2)}{\sqrt{2\pi}(1-t)^2} dx_j$$

$$V_2 \approx e^{-rT} \sum_{n=1}^{N} \int_0^1 \frac{\exp(-\frac{1}{2}(x_j^{*(n)} + \frac{t}{1-t})^2)}{\sqrt{2\pi}(1-t)^2} dx_j$$

As seen in figure 4, the computational cost of this implementation is heavy. Since it is a non-vectorized python code, computing a quadrature and root for each point takes a toll on the performance.
Now, by looking at the partial derivative $\frac{\partial \phi}{\partial x_k}$ from earlier, we see that it satisfies the preintegration condition $\frac{\partial \phi}{\partial x_k} > 0$ for any dimension $1 \leq j \leq m$ because the transformation matrix M has no zero-columns, and the price function S is always strictly positive. Furthermore, it is easy to see that $\phi \to \infty$ as any $x_j \to \infty$.

4 Variance Reduction

4.1 Antithetic variables

To reduce variance of our variables $Z = f(y_1, \ldots, y_m) := \Psi(\Phi^{-1}(y_1, \ldots, y_m))$, one option is to use what is called an *Antithetic Variable*. Since $\mathbf{y} \sim \mathcal{U}[0,1]^m$, we can conveniently construct our variable like so: $Z_a = f(1-y_1, \ldots, 1-y_m)$. We can then directly apply this to our simulation, by replacing the last $N/2$ samples by our antithetic variable.

$$V_i \approx \frac{1}{N} \sum_{j=1}^{N/2} \Psi_i(A\Phi^{-1}(\mathbf{y}^{(j)})) + \Psi_i(A\Phi^{-1}(\mathbf{1} - \mathbf{y}^{(j)}))$$

This strategy reduces variance because $\text{cov}(Z, Z_a) \leq 0$. To see this, we first note that y_i and $1 - y_i$ are also negatively correlated. Since $\Psi(M\Phi^{-1}(\cdot))$ is monotonic in each of its arguments, then it follows that $\text{cov}(\Psi_i(M\Phi^{-1}(\mathbf{y})), \Psi_i(M\Phi^{-1}(\mathbf{1} - \mathbf{y}))) \leq 0$. We can use this antithetic variable for preintegration as well:

$$V_1 \approx e^{-rT} \frac{1}{N} \sum_{i=1}^{N/2} \left(\int_{\psi_{-j}(\mathbf{y}_{-j}^{(i)})}^{1} \phi(A\Phi^{-1}(y_j, \mathbf{y}_{-j}^{(i)}))dy_j + \int_{\psi_{-j}(\mathbf{1}-\mathbf{y}_{-j}^{(i)})}^{1} \phi(M\Phi^{-1}(y_j, \mathbf{1} - \mathbf{y}_{-j}^{(i)}))dy_j \right)$$

$$V_2 \approx e^{-rT} \frac{1}{N} \sum_{i=1}^{N/2} (1 - \psi_{-j}(\mathbf{y}_{-j}^{(i)}) + 1 - \psi_{-j}(\mathbf{1} - \mathbf{y}_{-j}^{(i)}))$$

Variance reduction still holds since both the functions

$$p(\mathbf{y}_{-j}) := \int_{\psi_{-j}^{(i)}(\mathbf{y}_{-j})}^{1} \phi(A\Phi^{-1}(y_j, \mathbf{y}_{-j})dy_j$$
$$p(1-\mathbf{y}_{-j}) = \int_{\psi_{-j}^{(i)}(1-\mathbf{y}_{-j})}^{1} \phi(A\Phi^{-1}(y_j, 1-\mathbf{y}_{-j})dy_j$$

are monotonic and hence, negatively correlated. Their monotonicity intuitively comes from the fact that an increase in their arguments leads to an increase in the value of the integrand (by monotonicity of ϕ), and a decrease in the lower bound. The latter is because $\psi(\cdot)$ outputs the root of $\phi(\cdot)$, and so an increase in $\mathbf{y}\_j$, increases ϕ since it is monotonous aswell, and therefore the value of its root must decrease.

4.2 Control Variables

A second option to reduce variance is to, as opposed to antithetic variables, use a variable Y that is *highly correlated* with our variable of interest $Z = \Psi(\mathbf{y}(\mathbf{w}))$. The obvious choice is to take $Y = \phi(\mathbf{w})$. And so, our control variable is defined as $Z_\alpha = Z - \alpha(Y - \mathbb{E}[Y])$ and so we have that $\mathbb{E}[Z] = \mathbb{E}[Z_\alpha]$ and we can approximate our integral by

$$V_i \approx \frac{1}{N}\sum_{j=1}^{N} Z^{(i)} - \alpha(Y^{(i)} - \mathbb{E}[Y^{(i)}])$$

We find that

$$\begin{aligned}
\mathbb{E}[\phi(w)] &= \mathbb{E}[\frac{1}{m}\sum_{i=1}^{m} S_{t_i}(w_{t_i}) - K] \\
&= \frac{S_0}{m}\sum_{i=1}^{m} \mathbb{E}[\exp\left(t_i\left(r - \frac{\sigma^2}{2}\right) + \sigma w_{t_i}\right)] - K \\
&= \frac{S_0}{m}\sum_{i=1}^{m} \exp(t_i r) - K \\
&= \frac{S_0}{m}\frac{\exp(\Delta t r)}{1-\exp(\Delta t r)}(1 - exp(Tr)) - K
\end{aligned}$$

In our implementation, we estimate $\sigma_{\Psi_i,\phi}$ and σ_ϕ and set $\alpha = \frac{\sigma^2_{\Psi_i,\phi}}{\sigma^2_\phi}$. And so, we can estimate V_i in the following way

$$V_i \approx \frac{1}{N}\sum_{j=1}^{N} \Psi_i(A\Phi^{-1}(\mathbf{y}^{(j)})) - \alpha\phi(A\Phi^{-1}(\mathbf{y}^{(j)})) + \alpha(\frac{S_0}{m}\frac{\exp(\Delta t r)}{1-\exp(\Delta t r)}(1 - exp(Tr)) - K)$$

This will reduce variance since $Z = \Psi_i(A\Phi^{-1}(\mathbf{y}^{(j)}))$ and $Y = \phi(A\Phi^{-1}(\mathbf{y}^{(j)}))$ are positively correlated.

4.3 Scrambling and Digital Shift

A third way to reduce variance is to use better sample points and randomization for our QMC simulations. Using a normal shift which should be used for lattice changes the t-values of the (t-m-s) net, whereas using a digital net would preserve them. Furthermore, since our integrand can be view as the sum of low dimensional functions, its effective dimension is low and QMC can perform an effective estimation. In our case, with the Asian option prices, the effective dimension is 2[4]. Even more, the Sobol points made by Joe-Kuo that we use in our implementation have a very uniform projection to 2 dimensions and therefore preserving this property after randomization is preferable.

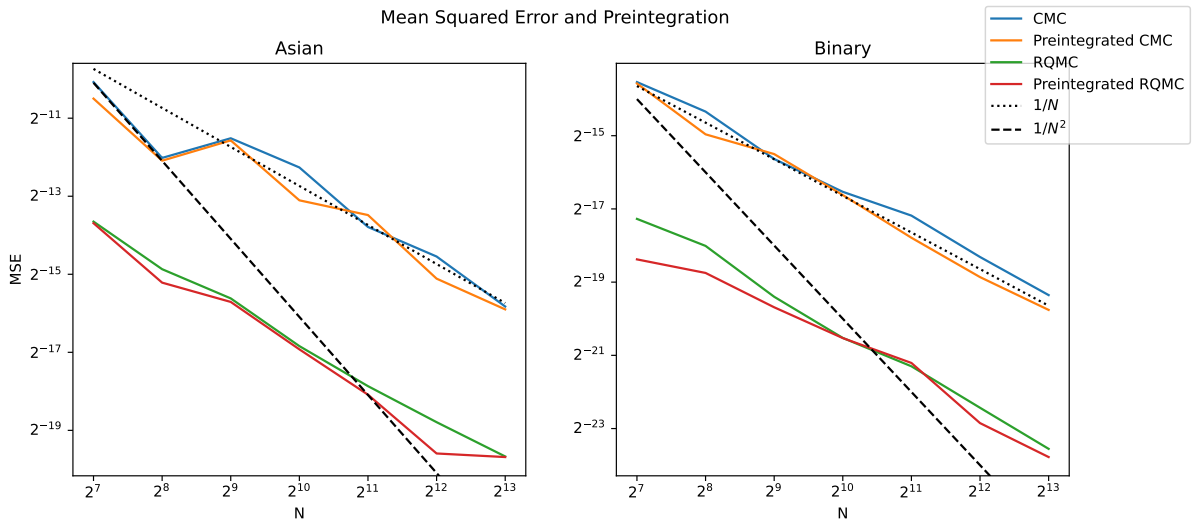

Fig. 3: The Mean Squared Error for both Randomized Quasi Monte Carlo and Crude Monte Carlo with and without preintegration.

4.4 Truncated Mean

From the work done in [2], to reduce the variance, instead of taking the mean of K sets of randomized QMC points using scrambling and digital net points, we can use median or truncated mean (*i.e.* removing the smallest and greatest values). This is because most QMC points give good approximations, but a small number of them aren't, and so throwing them out increases our precision.

5 Results

5.1 Transformation

When comparing the Cholesky transformation versus the Lévy-Ciesielski construction, we were expecting the latter method to be more efficient, since we chose the N such that both matrices would have the same dimensions, and because building η does not require an expensive Cholesky factorization. However, we noticed little difference when comparing both of them in terms of computing time. We conclude that the reason for this result is that the bulk of our simulations lie elsewhere down the line (e.g. quadrature). On the other hand, we expected the Lévy-Ciesielski construction to induce a higher error estimation given that it is based on the approximation of a Wiener Process. Again, not much difference in estimated error was observed, possibly because approximating a Wiener process with m variables is enough to yield an estimation with negligible error. In light of these observations, the results shown in this paper were generated using the Cholesky transformation, for no particular reason.

5.2 Crude MC and Randomized QMC

In terms of magnitude, the estimated Mean Squared Error of RQMC, with or without preintegration, is much lower than that of CMC, as depicted in Figure 3. Theoretically, the absolute error of QMC is $(\log n)^s/N$ and will should be worse than CMC at around $s = 20$ dimensions. However, as we mentioned in the variance reduction section 4.3, since the effective dimension of the Asian option price is low, QMC outperformed MC. As for the convergence rate of CMC, it matches exactly the theoretical convergence rate of the MSE of $1/N$.

5.3 Preintegration

The results regarding the estimated error and convergence rate of the preintegration trick are inconclusive. Preintegration was expected to reduce the error, since Gauss-Quadrature is much more precise than a 1d version of MC or QMC. But as seen in Figure 3, the decrease is too low to draw any conclusions. Even more, the results varied greatly between runs, so some runs looked like it decreased the error sufficiently, while others not. As for the MSE convergence rate, the Crude Monte Carlo is, as expected, almost unaffected when preintegrated. For the QMC method, we couldn't observe the improvement of the convergence rate which was seen in [3].

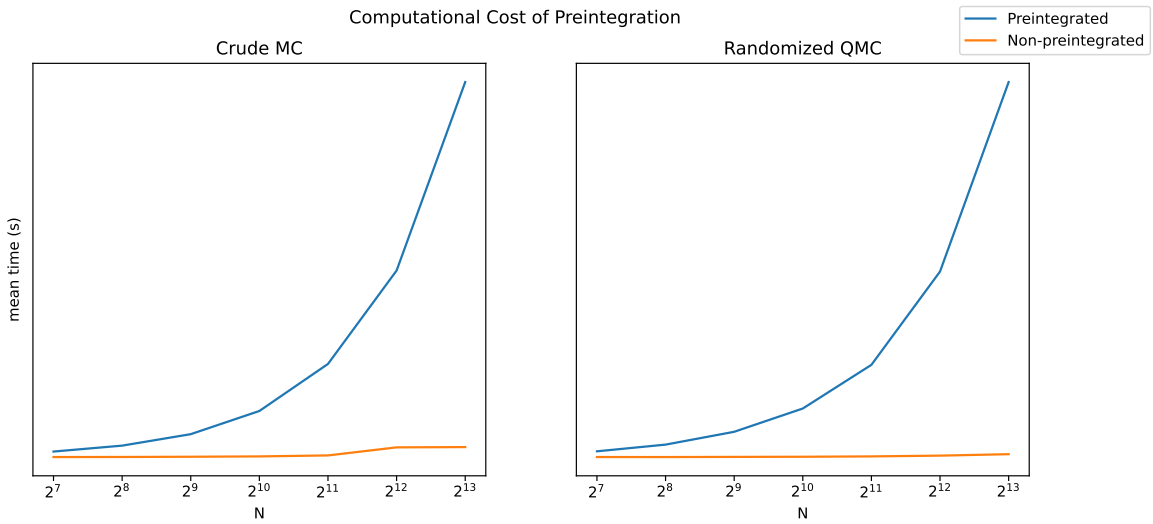

Fig. 4: The mean computation time over dimension values m against the number of samples N.

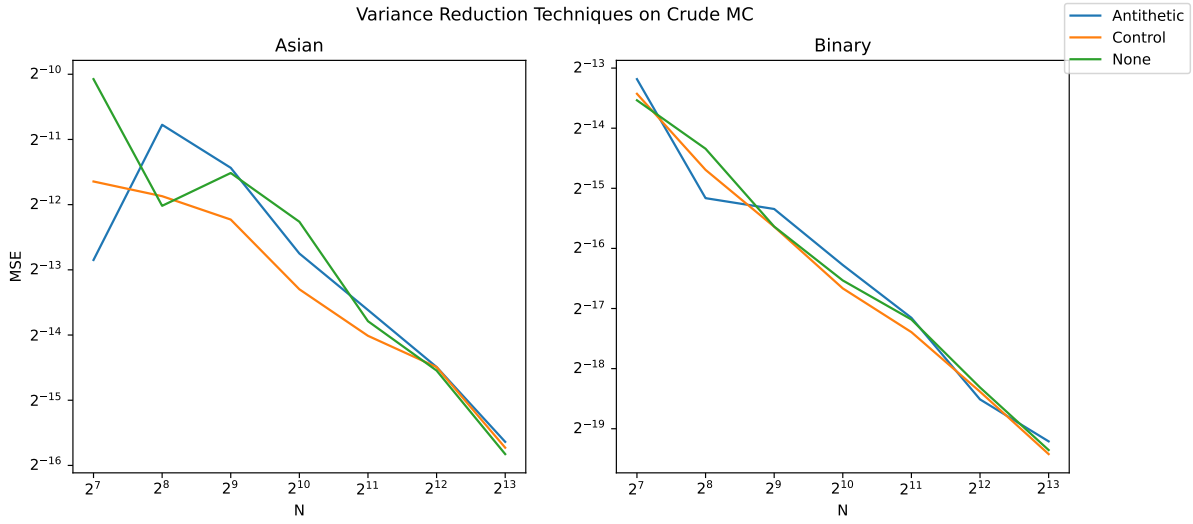

Fig. 5: Antithetic and Control variables applied to the Crude MC method to reduce variance.

5.4 Variance reduction

For all applicable methods, we applied Antithetic and Control variables to reduce variance (results shown in Figures 5, 6 and 7). Sadly, we couldn't see any significant change when applied. One possible reason for this might be the value of the covariance of our chosen variables. For example, for the Antithetic variable, $\text{cov}(\Psi_i(A\Phi^{-1}(\mathbf{y})), \Psi_i(A\Phi^{-1}(\mathbf{1}-\mathbf{y})))$ is indeed negative but might not have a big enough magnitude to make a difference.

In addition to these two methods, we used truncated mean of digital shifted and scrambled points to Randomized QMC (results shown in Figure 6). This method made a significant variance reduction, indicating that this method is a better choice than randomly shifted Sobol points.

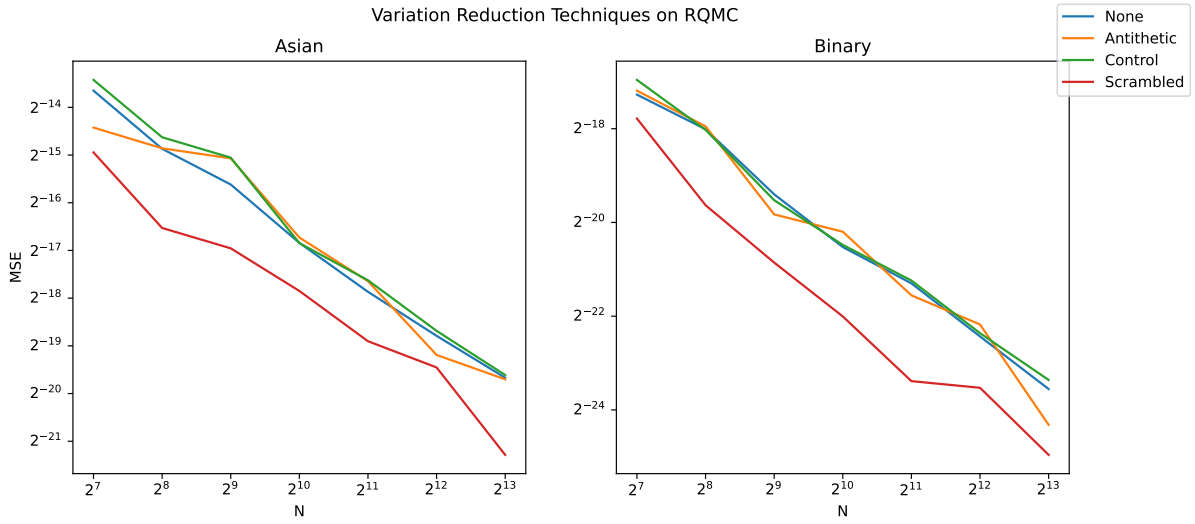

Fig. 6: Antithetic and Control variables, and truncated mean of scrambled points applied to the Randomized QMC method to reduce variance.

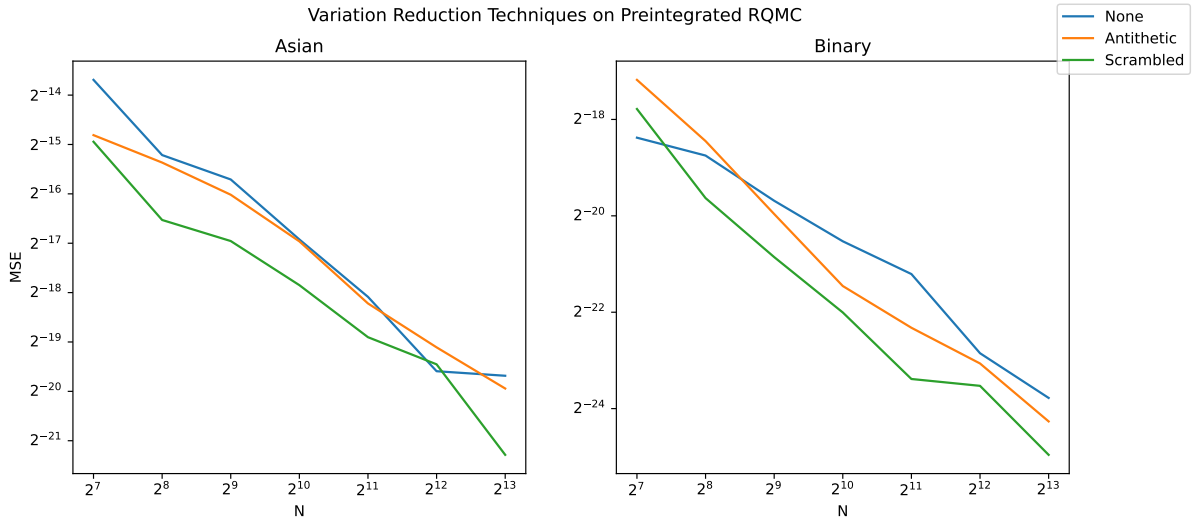

Fig. 7: Antithetic variables and truncated mean of scrambled points applied to the preintegrateed Randomized QMC method to reduce variance.

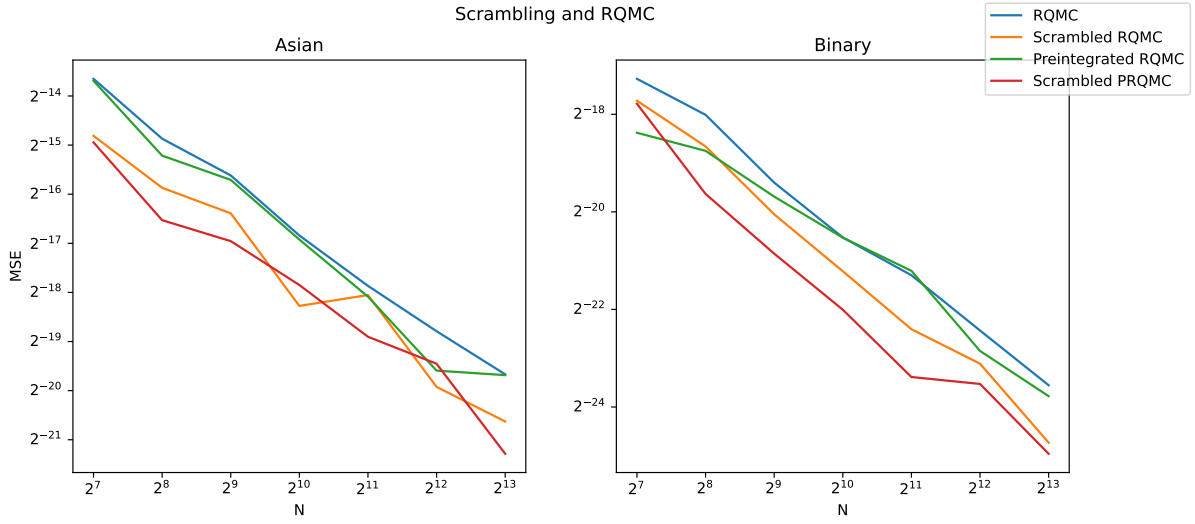

Fig. 8: Truncated mean of scrambled points and RQMC

6 Conclusion

First of all, we couldn't observe a significant advantage of pre-integration of QMC shown in [3]. The possible reason is that the reference paper used 2^{12} to 2^{16} points, while we used 2^7 to 2^{13} points, so our sample size may not have been sufficient. It would be interesting to test our implementation on a more powerful machine. Another possibility to yield better results would be to look into a more precise method of quadrature and root finding, which could possibly reduce the variance. Furthermore, a difference in implementations with the reference paper is the calculation of inverse of standard normal distribution. Also, we also didn't observe the effect of antithetic variables and control variables in variance reduction. What worked however, was using the truncated mean of the estimated values gained by randomly digital shifted and scrambled Sobol points.

References

[1] Bruce Brown, Michael Griebel, Frances Y Kuo, and Ian H Sloan. On the expected uniform error of geometric brownian motion approximated by the l\'evy-ciesielski construction. *arXiv preprint arXiv:1706.00915*, 2017.

[2] Takashi Goda and Pierre L'ecuyer. Construction-free median quasi-monte carlo rules for function spaces with unspecified smoothness and general weights. *SIAM Journal on Scientific Computing*, 44(4):A2765–A2788, 2022.

[3] Andreas Griewank, Frances Y Kuo, Hernan Leövey, and Ian H Sloan. High dimensional integration of kinks and jumps—smoothing by preintegration. *Journal of Computational and Applied Mathematics*, 344:259–274, 2018.

[4] Christiane Lemieux. Quasi–monte carlo constructions. *Monte Carlo and Quasi-Monte Carlo Sampling*, pages 1–61, 2009.