An Analysis of Sobol Sequence and the Brownian Bridge

Sébastien Gurrieri *

Risk Management Department, Mizuho International
London

Abstract

We investigate the loss of efficiency of Sobol low-discrepancy sequence at high dimensions and the apparent improvement provided by the use of the Brownian bridge construction of Brownian motion paths. We show numerically that some often cited potential causes for these phenomena such as low quality of Sobol coordinates at high dimensions are not to blame and instead isolate a bias in Sobol sequence which we conjecture to be the main cause of the problem. We motivate this conjecture by an analysis of the equations defining both the Incremental and Brownian bridge constructions in a simplified setting, showing how the identified bias is removed by the use of the Brownian bridge. We further give numerical evidence that randomizing Sobol sequence can remove most of this bias and achieve a good convergence at high dimensions. We explain why this is particularly relevant for efficient inline implementations in massively parallel environments such as GPUs under the programming language CUDA. Tested products and models include vanilla and barrier options as well as TARN PRDCs in Local Volatility. We also provide proofs of the homogeneity properties of the Gaussian deviates derived from Sobol sequence for particular numbers of iterations.

Keywords: Sobol sequence, Brownian bridge, Quasi Monte Carlo, parallel programming, CUDA.

*Sébastien Gurrieri
Bracken House, One Friday Street
London EC4M 9JA
tel: +44-20-7090-6342
email: sebastien.gurrieri@uk.mizuho-sc.com

Contents

1 Introduction

The analysis presented here was motivated by our interest in pricing exotic derivatives on equities or foreign exchange rates (FX) under models requiring many time steps and thus Brownian motions with high dimensions.

Such situations arise for example in the pricing of Power-Reverse-Dual-Coupon (PRDC) swaps where the underlying FX rate is modelled by a Local Volatility, for example following Dupire's framework [1]. In order to recover the market skew/smile accurately, many steps must be included in the time direction as the freezing of the volatility function between two times yields an error that is minimized by small step sizes. Often long-dated, these swaps may require the modelling of the stochasticity of both the domestic and foreign interest rates in order to properly manage the risk on these quantities. This will further multiply the dimension several times depending on how many factors are used to model the interest rates.

An other situation where very high dimensions can be encountered is the pricing of basket options for which some of the underlyings follow a Local Volatility process. Again many time steps must be included, and the dimension will be multiplied by the number of assets in the definition of the trade.

If one considers that a typical configuration for a Local Volatility model is easily of 250 time steps, and for example that we are pricing a PRDC in a 3 factor hybrid setting where the interest rates follow a 1 factor short-rate model, then the dimension of the sequence of Gaussian numbers required to build the correlated Brownian path is already 750. For a basket option, this can go much higher.

Since most of the products and models mentioned here have at least 3 factors, Finite Difference Methods are cumbersome to implement and potentially slow, although still possible in 3 factors [2]. A Monte Carlo simulation is the more commonly chosen method in such circumstances.

However, due to the large number of dimensions, and the slow Monte Carlo convergence in terms of the number of simulations, especially for long-dated products, even Monte-Carlo pricing can become very time-consuming. Variance reduction techniques are therefore welcome. One such technique is the Quasi-Monte Carlo (QMC) framework with for example Sobol low-discrepancy sequence [3, 4], which is recognized to generally achieve faster convergence than typical Pseudo-Random generators [5, 6].

On the other hand Sobol sequence suffers from a well-known loss of efficiency at high dimensions, and this is precisely what we need for the products and models considered here. To circumvent this difficulty, the Brownian bridge path construction method is often introduced in conjunction with Sobol sequence. Many authors have reported the excellent performance of this combination with empirical studies, but the mechanisms by which this is achieved are not totally understood yet. We have come across several possible reasons when reading literature or interacting with practitioners and cannot cite all references here. Interesting studies on the subject of QMC and the Brownian bridge (or dimensional reduction) can be found in [7, 5, 8, 9], among others. Often cited reasons are

- the coordinates at high dimensions in Sobol sequence would have worse uniformity and projections than those at low dimensions. In an Incremental path construction, all the "good" points would be placed at first, and the "bad" points at last, which would result in poor pricing accuracy due to the accumulation of errors in the same areas. The Brownian bridge, by scrambling these coordinates, would mix good and bad points and the errors would become less apparent.

- the Brownian bridge, with its splitting of the interval at each iteration, would attribute the best Sobol dimensions to the "most important" times of the trade
- the Brownian bridge would result in a reduction of the effective dimension and thus would make the problem caused by Sobol sequence at high dimensions less visible.

Although these sound like reasonable explanations, we find some of them not totally convincing or justified. For example, as we will show later in this work, it is not clear to us that the higher coordinates in Sobol sequence have worse uniformity or projections, especially considering the latest results in the search for good direction integers [10]. It is not clear to us either that the splitting of the Brownian bridge by half-intervals really brings the first coordinates of the sequence on the "most important times" of the trade. Furthermore, the notion of reduction of effective dimension is not always easy to apply to predict the convergence performance, although some interesting progress have been made for example in [9].

In spite of all these unknowns, the very good performance of the combination of Sobol sequence in conjunction with the Brownian bridge has been observed by many practitioners in many situations and it is not our goal to question it here. On the contrary, we would like to benefit from it as much as possible in order to reduce the runtime of our pricing applications.

An other way of improving the calculation efficiency is to use parallel programming, especially in the case at hand of Monte-Carlo simulations, whose intrinsic parallel character makes it possible to obtain large speed increases. Various hardware architectures and programming languages are available for this purpose. We are particularly interested in Graphic Processing Units (GPUs), which, containing very large numbers of cores, potentially lead to dramatic speed improvements. The language CUDA [16], developed for this purpose by GPU maker NVIDIA, has been deployed with success recently by several major market participants, and literature on the subject started to appear [11, 12].

Ideally we would like to benefit both from parallelism and from the good variance reduction properties of Sobol sequence and/or the Brownian bridge. Although implementations of Sobol sequence are already available in the standard CUDA library, it is not so for the Brownian bridge. One reason we see for this is the relative difficulty to implement the Brownian bridge in parallel and the special use that it makes of memory storage and accesses.

This brings us to the study presented here. Having for goal to run Monte-Carlo simulations in high dimensions and in parallel, we wanted to understand more precisely how the Brownian bridge acts with the optimistic view to, either modifying it to implement it more easily in parallel, or finding an other way to achieve similar performances with other random number generators. An other motivation for this work is also simply the desire to understand in more depth the mechanism by which the Brownian bridge can, sometimes quite spectacularly, improve the convergence of Sobol sequence, all this applied to very practical problems in exotic pricing.

We start in section 2 by illustrating the loss of convergence efficiency occurring in Sobol sequence at high dimensions, and to which extent the Brownian bridge improves the situation. The goal of this section is to remind the reader of how the problem comes about and set up some notations as well as testing framework.

In section 3 we proceed by elimination of factors in order to isolate more precisely the causes of the phenomenon. We show numerically that the effect of the Brownian bridge is restricted to Sobol sequence (among the several generators we tested) and conclude that it corrects a deficiency in Sobol sequence not related to a supposed poorer quality of the coordinates at high dimensions. We find that the Brownian bridge compensate for this not by scrambling the coordinates but rather thanks to the particular way in which it attributes the Gaussian deviates on the path, avoiding sums of large numbers of deviates.

In order to find what specific defect might exist in Sobol sequence and not in other generators, we perform a statistical analysis of Sobol sequence at high dimensions, observing the first 4 moments and the auto-correlations. We find a bias in some of these quantities that is not present in other generators and conjecture that this bias is removed by the Brownian bridge construction. By using special simulations settings for which Sobol sequence exhibits exceptional regularities, we show in a simple analytical model that the bias identified previously is indeed responsible for an extra error appearing in the Incremental construction but not in the Brownian bridge. We further show numerically that this extra error can be rather significant. Simple proofs of the regularity properties of Sobol sequence are given in the appendix, for both the uniform and normal samples.

In section 4 we describe in more details how we view the difficulty of the implementation of the Brownian bridge in the parallel environment of GPUs programmed with CUDA. We finally show numerically how in a simple randomized version of Sobol sequence, known since as early as 1976 in [13], most of the bias of the original sequence is removed while retaining most of its good convergence properties. We provide examples of its convergence and show that, even in the Incremental path construction, its convergence speed can compete with the Brownian bridge at high (and low) dimensions. This gives further illustration of the conjectured relation between the bias in Sobol sequence and the convergence properties in high dimensions. This algorithm is particularly well suited for implementations on architectures for which memory amounts and accesses can be sources of limitations of performance.

Throughout this document we assume that the reader is familiar with the definitions of Sobol sequence and the Incremental and Brownian bridge path constructions so we will not recall them here. For a good review on these subjects, see [5]. All our tests are based on the direction integers "JoeKuo6" of [10]. In the present analysis we used different products including vanilla as well as exotic derivatives, in Black or Local Volatility models. Many parameter and trade configurations have been tested, but for the sake of shortness and clarity we present only a few here.

2 Description of the problem

In this section we consider Up-And-Out barriers with one monitoring event before maturity, at which a call/put option is paid in case of survival. Under Black model, this product has an exact closed form which provides us with a true value to compare the simulation with. The convergence of the Monte-Carlo result is plot by taking snapshots of the simulated price at several intermediate simulation numbers.

We have tested various configurations of call/puts, strike, barrier, maturities, but due to space limitations, we fix the product and model characteristics, and focus on the convergence according to the random numbers and path. We look at a put option with maturity 20Y and strike $K = 80$, the barrier is at 12Y at the level $B = 110$. The spot is at $S_0 = 100$, the rate and dividends are 0, and the volatility is $\sigma = 40\%$.

2.1 Failure of Sobol sequence at high dimensions

In fig. 1 we draw the convergence of the simulation based on Sobol sequence (denoted "Sobol") in the Incremental Brownian path construction, for several numbers of time steps.

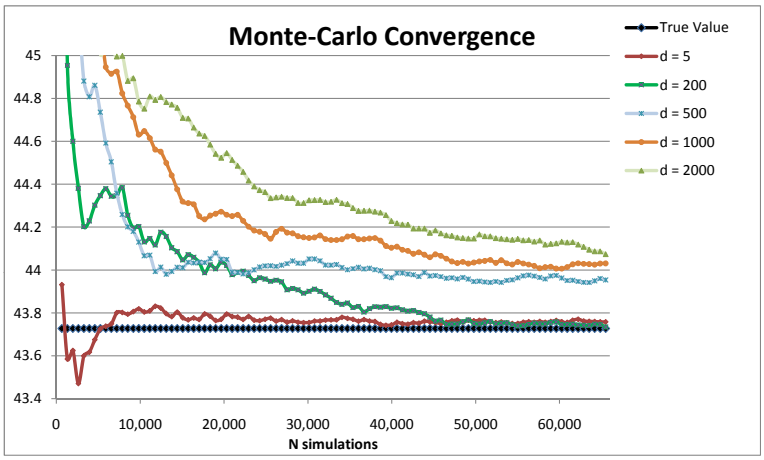

Figure 1: Convergence in Sobol + Incremental

We can clearly see that the more time steps we take, the worse the convergence becomes. This is an illustration of the well-known issue with Sobol sequence at high dimensions.

This problem can be so acute that Sobol sequence can even under-perform a pseudo-random sequence such as Mersenne-Twister (denoted "MT"). In fig. 2 below we can see that at 5 time steps ($d = 5$) Sobol performs better than MT. At 200 time steps in fig. 3 it is no longer obvious which is best, and at 2,000 time steps in fig. 4, MT is more efficient than Sobol.

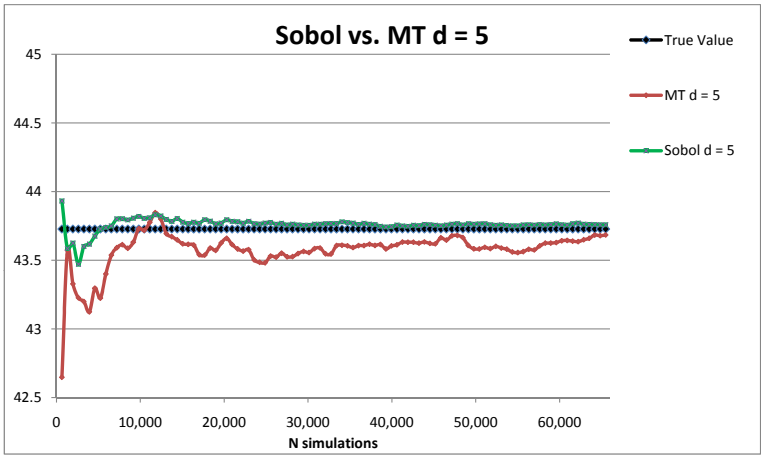

Figure 2: Sobol vs. MT, Incremental

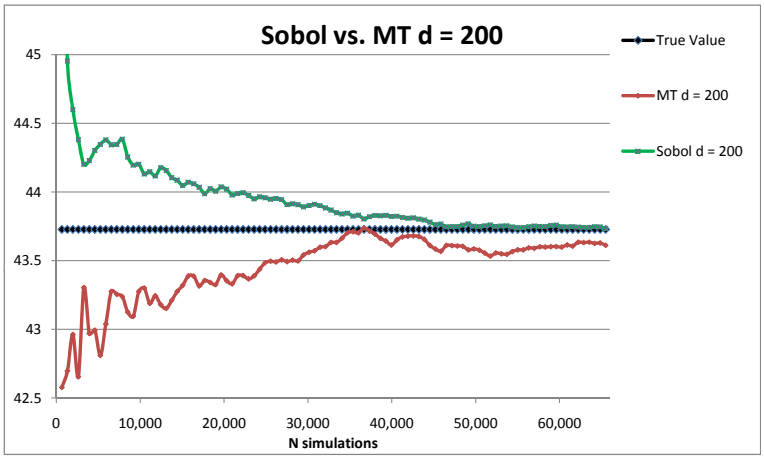

Figure 3: Sobol vs. MT, Incremental

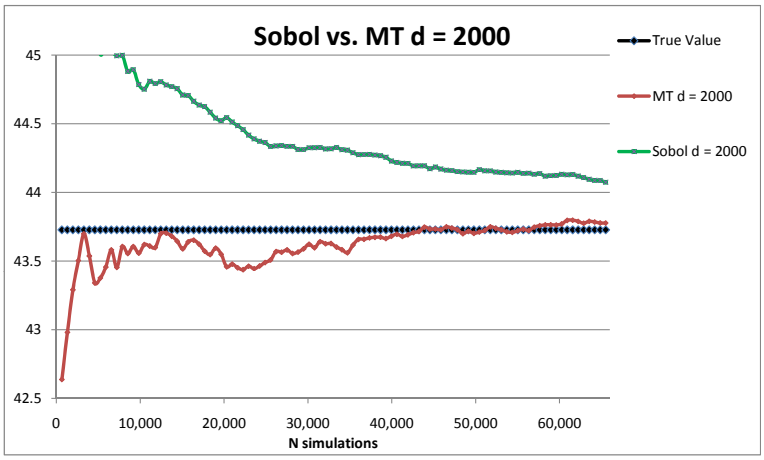

Figure 4: Sobol vs. MT, Incremental

2.2 Improvement with the Brownian bridge

Now we introduce the Brownian bridge (BB) and compare again the convergence with that of Mersenne-Twister Incremental (Inc).

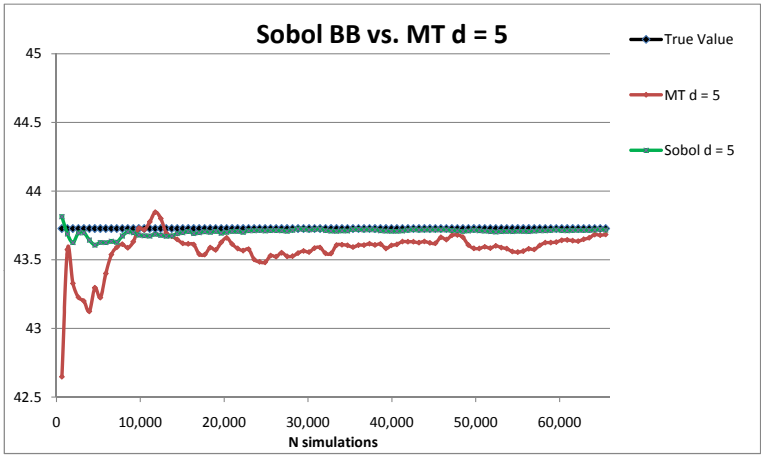

Figure 5: Sobol Brownian bridge vs. MT Incremental

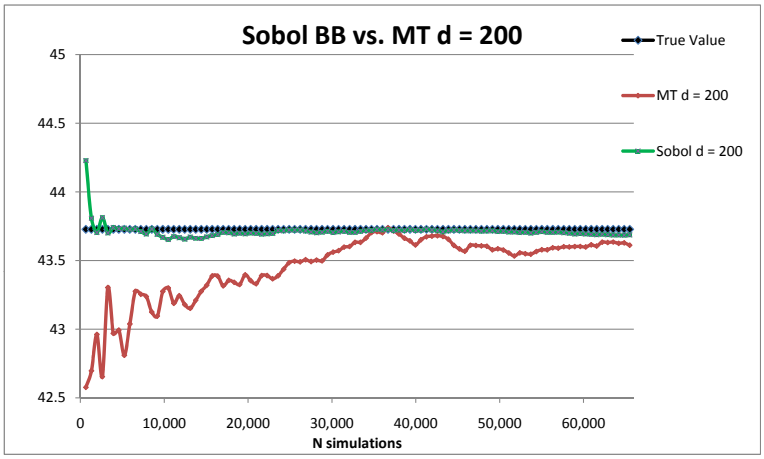

Figure 6: Sobol Brownian bridge vs. MT Incremental

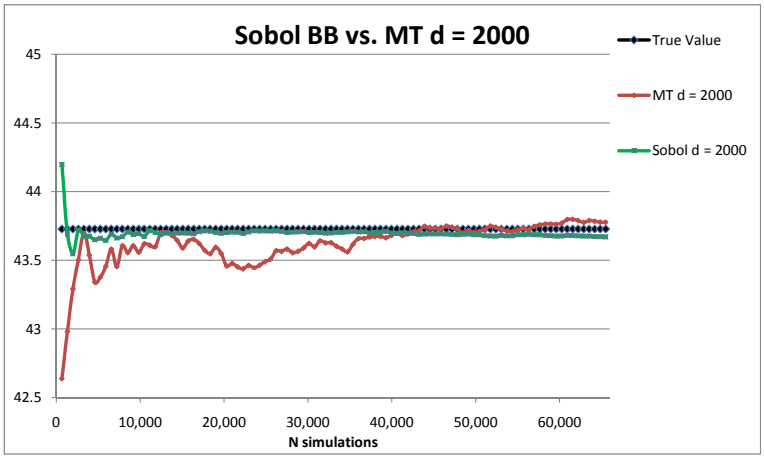

Figure 7: Sobol Brownian bridge vs. MT Incremental

The convergence has improved dramatically on the Sobol side thanks to the use of the Brownian bridge. This improvement is especially visible at high dimensions, as can be observed by comparison of fig. $2 \sim 4$ with fig. $5 \sim 7$.

This raises two questions (at least) that we will attempt to answer in the next section:

1. Why does the convergence under Sobol sequence worsen so much with the dimension?
2. Why does the Brownian bridge solve this problem by compensating so well for the above failure?

3 Investigation of the causes

3.1 Elimination of factors

In order to find hints of answers to the questions in the previous section, we need to separate the influence of several factors in order to isolate the culprit(s).

- **Brownian bridge on other random sequences**

  Here we want to check whether the improvement in the convergence with Sobol sequence thanks to the Brownian bridge can be extended to all sequences. In fig. 8 we compare the convergence under Mersenne-Twister sequence and both the Brownian bridge and Incremental path constructions.

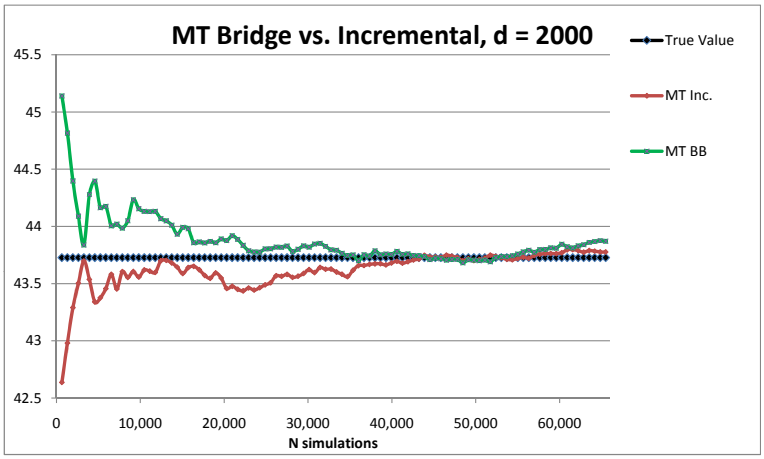

Figure 8: Mersenne-Twister, Brownian bridge vs. Incremental

  A similar pattern can be observed at other dimensions and for other pseudo-random sequences[1]. It appears that the Brownian bridge does not lead to a conclusive effect on other random sequences, although it does bring a significant reduction in variance when used with Sobol sequence. We deduce from this that something in the algorithm of the Brownian bridge is able to remove, or avoid, an imperfection of random generators, exhibited in particular by Sobol sequence, but not by the other sequences we tested.

- **Splitting the Brownian bridge**

  Next we would like to understand specifically which feature of the Brownian bridge algorithm has this effect on Sobol sequence. We view the Brownian bridge mainly as a combination of 2 steps.

  First is a "scrambling" of the random sequence, changing the order in which the Gaussian deviates are attributed to the Brownian motion increments, mixing Sobol coordinates of high dimensions with low ones.

  The second step is the particular order in which the Brownian motion values are calculated, with the last point at first, then the middle point in second, and so on and so forth by halving the interval at every iteration (with matching of the conditional expectations and variances).

[1] We performed the same tests on L'Ecuyer [14] and Multiply-With-Carry [15] and obtained similar results.

Here we want to separate these 2 steps to see which one, if not both, contribute to the variance reduction. One relatively easy way to achieve this is to perform the same scrambling of the sequence as dictated by the Brownian bridge but attribute the resulting Gaussian incrementally. This does lead to a mixing of the high and low dimensions of Sobol sequence, but incrementally.

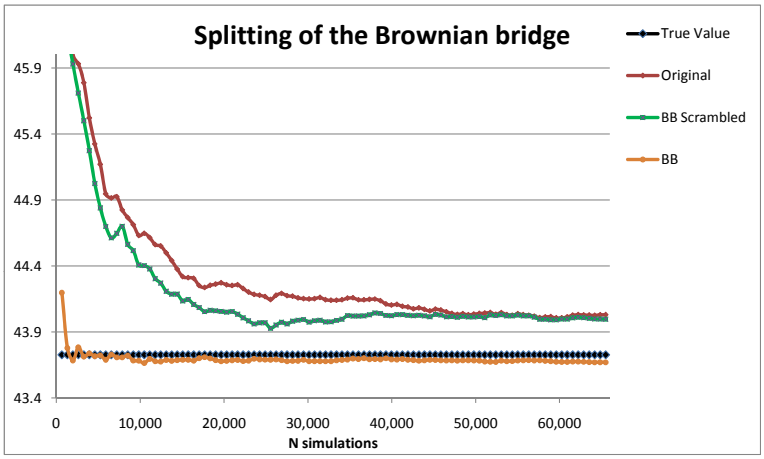

Figure 9: Splitting the effect of scrambling and halving intervals in the Brownian bridge, $d = 1000$

With the convergence of Sobol sequence without bridge in red, we observe that the effect of the bridge scrambling (green) is weak, if not insignificant, compared to the impressive improvement obtained when using the full Brownian bridge algorithm (orange). This leads us to the interpretation that it is not the scrambling of the sequence that makes the Brownian bridge efficient, but rather the attribution of the Gaussian deviates with successive halving of the intervals.

- **Bad quality at high dimensions?**

  We interpret the above results as meaning that the Brownian bridge corrects a deficiency in Sobol sequence, that is not present in some other random sequences. The nearly unchanged convergence with just a "bridge-like scrambling" of the sequence, without "halving interval attribution", seems to point to the fact that coordinates at high dimensions in the sequence are not to blame. Let us do just one more test to confirm this, since the supposed bad quality of the high-dimensional coordinates is often cited as one cause of the problem.

  In order to test this explanation, say at dimension d, we draw a sequence of size $2d$ and price with the first d coordinates, and then the last d coordinates. If the coordinates at high dimensions indeed are worse than at low dimensions, we should see a significant worsening of the convergence when using the last d dimensions. In fig. 10 and 11 below we show the result of this permutation

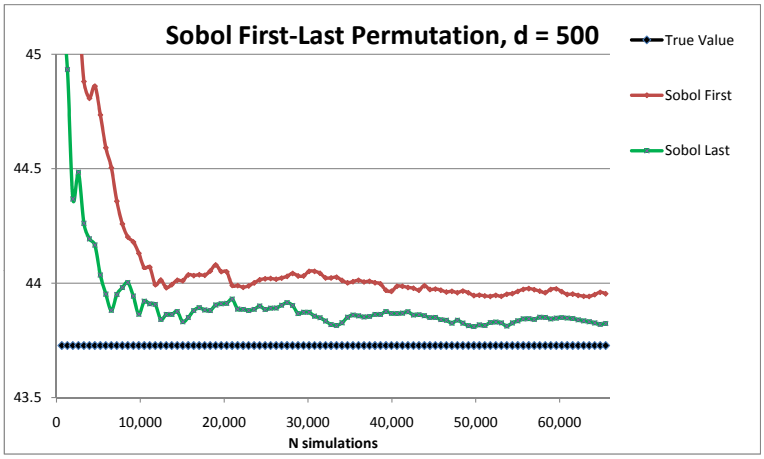

Figure 10: Permutation of Sobol sequence by first-last inversion

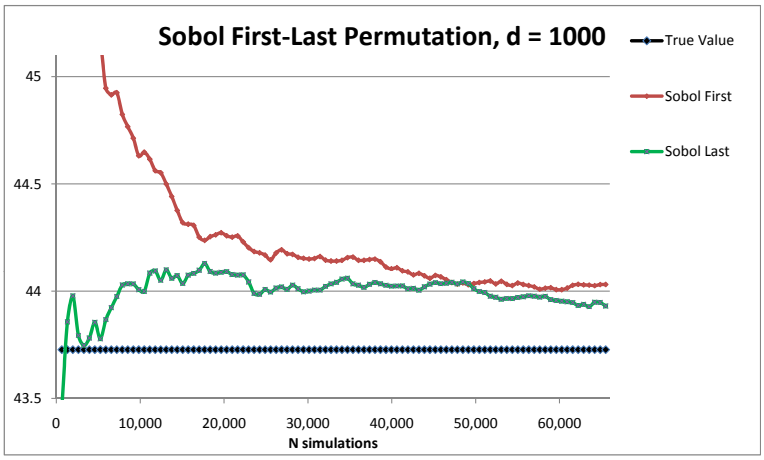

Figure 11: Permutation of Sobol sequence by first-last inversion

We do not observe a conclusive effect of the permutation. This is a confirmation that the problem in using Sobol sequence with many dimensions does not seem to be related to a lower quality of the coordinates at high dimensions.

3.2 A bias in Sobol sequence

With the tests in the previous section, we conclude that the attribution of the Gaussian deviates by the halving of the intervals compensates or avoids a deficiency in Sobol sequence, and that this deficiency is not a lower quality of the Sobol coordinates at high dimensions. Here we try to understand what this deficiency could be by calculating various statistics on Sobol sequence. More precisely we look at the first 4 moments of the distribution of the Gaussian deviates, calculated by applying the inverse normal cumulative distribution function to the uniform sequence, as well as the "auto-correlations" between dimension i and $i+1$, for the first 1000 dimensions. We observe the distribution of the 1000 values for each of these moments (999 auto-correlations) and compare with Mersenne-Twister sequence.

We first do this at the special number of simulations $2^n - 1$ with n integer, as these have particular homogeneity properties for Sobol sequence. These have long been known for the uniform sequence in the framework of digital nets [3], but the properties we describe below at the level of the Gaussian deviates are less commonly used, as far as we know. We give simple proofs of some of them in appendix A. The following patterns can be observed:

1. The means and skews at all dimensions are exactly 0
2. The variances (resp. kurtosis) are the same at all dimensions, and equal to a value slightly lower than 1 (resp. 3)
3. The auto-correlations are small but biased on the negative side.

In table 1 below we give an illustration of points 1) and 2) with the values of the 4 moments at different numbers of simulations.

Table 1: 4 moments of Sobol sequence

n	Simulations	Means	Variances	Skews	Kurtosis
10	1,023	$\sim 10^{-16}$	0.9883	$\sim 10^{-16}$	2.8353
12	4,095	$\sim 10^{-16}$	0.9964	$\sim 10^{-16}$	2.9406
14	16,383	$\sim 10^{-16}$	0.9989	$\sim 10^{-16}$	2.9797
16	65,535	$\sim 10^{-16}$	0.9997	$\sim 10^{-16}$	2.9933

In fig. 12 we display the distribution of the 999 auto-correlations, with $n = 14$, i.e. 16,383 simulations. The x-axis represents the ranges of the correlations, and the y-axis the percentage of the total number of points (999) falling into this range. The y-axis cuts the x-axis at the 0% correlation in order to make the asymmetry clearer.

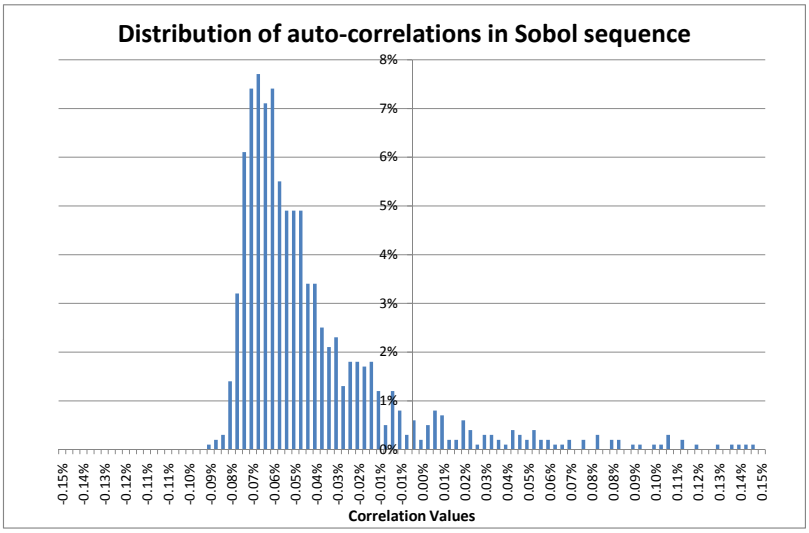

Figure 12: Distribution of the auto-correlations in Sobol sequence at $N = 16,383$ simulations

The bias is very visible in this figure. Most of the distribution lies on the negative side, while some much larger but rare values can be observed on the positive side and could not be included in this graph for scaling reasons (maximum observed correlation at 1.02%). This bias persists at lower or higher numbers of simulations such as those represented in table 1.

When going away from the special numbers of simulations $2^n - 1$, the moments are no longer equal at all dimensions, and the odd-moments are no longer exactly equal to 0. This means that their values can no longer be represented in such simple tables as 1. Instead we draw their distributions. For the mean, we find no particular problems, see fig. 13.

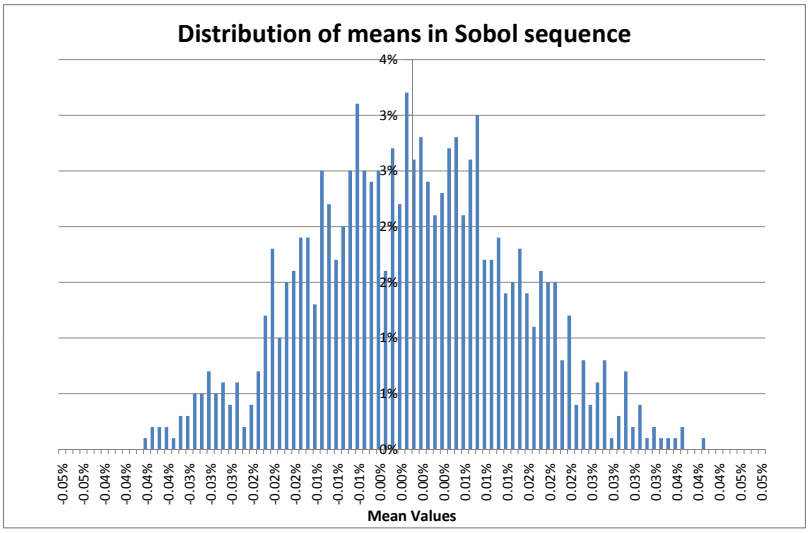

Figure 13: Distribution of the means in Sobol sequence at $N = 24,000$ simulations

When comparing with the distribution of the means in Mersenne-Twister sequence in fig. 14, we find a similar pattern of good symmetry around 0, although for Mersenne-Twister the range of attained means is much larger, showing the superiority of Sobol sequence as far as the individual means are concerned.

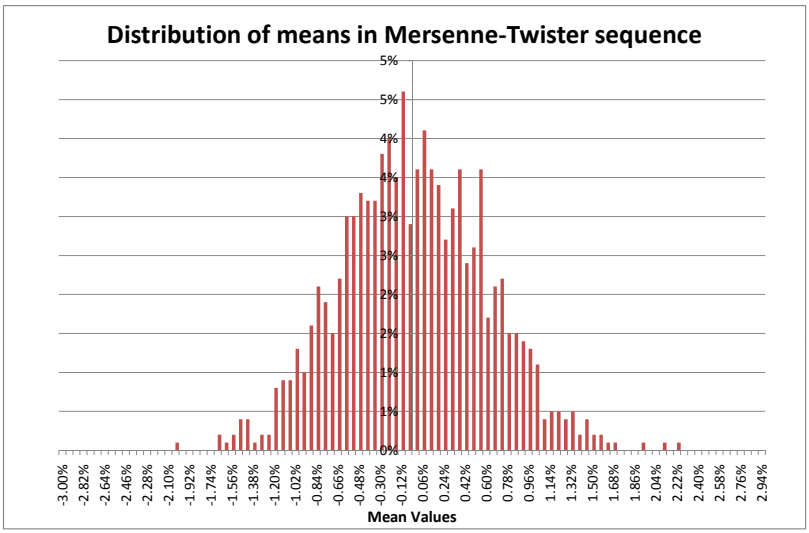

Figure 14: Distribution of the means in Mersenne-Twister sequence at $N = 24,000$ simulations

The variances are no longer equal at all dimensions and also spread near their theoretical value 1, but there is a strong bias on the lower side as can be seen in fig. 15. The variances in Mersenne-

Twister sequence on the other hand do not show such a bias, see fig. 16, although, similarly to the means, they span a much wider range around 1.

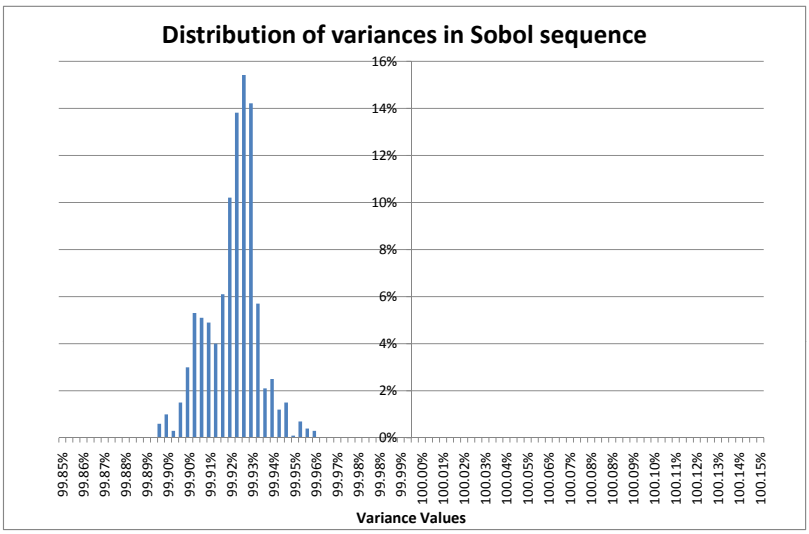

Figure 15: Distribution of the variances in Sobol sequence at $N = 24,000$ simulations

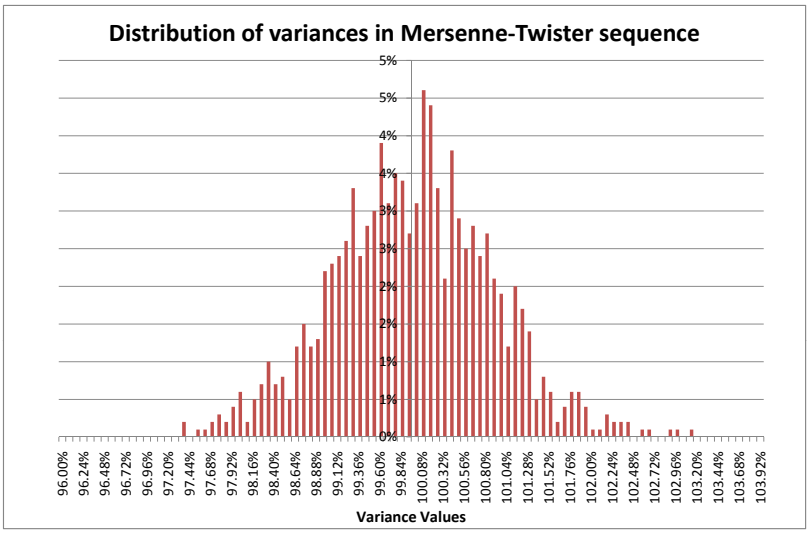

Figure 16: Distribution of the variances in Mersenne-Twister sequence at $N = 24,000$ simulations

Similar conclusions hold for the other moments: the distribution of the skew is nearly symmetric around 0 for both sequences, with a much shorter range on Sobol side. For the kurtosis, there is a very visible bias in the Sobol sequence at values lower than 3, while Mersenne-Twister sequence is nearly symmetric around 3.

Finally, let us show the evolution of the auto-correlations. In Sobol sequence, in a manner analogous to the moments, their distribution has spread compared to the special number of simulations with $n = 14$, but it is still visibly biased, see fig. 17. No such bias is exhibited in Mersenne-Twister sequence in fig. 18, with on the other hand a larger range of values.

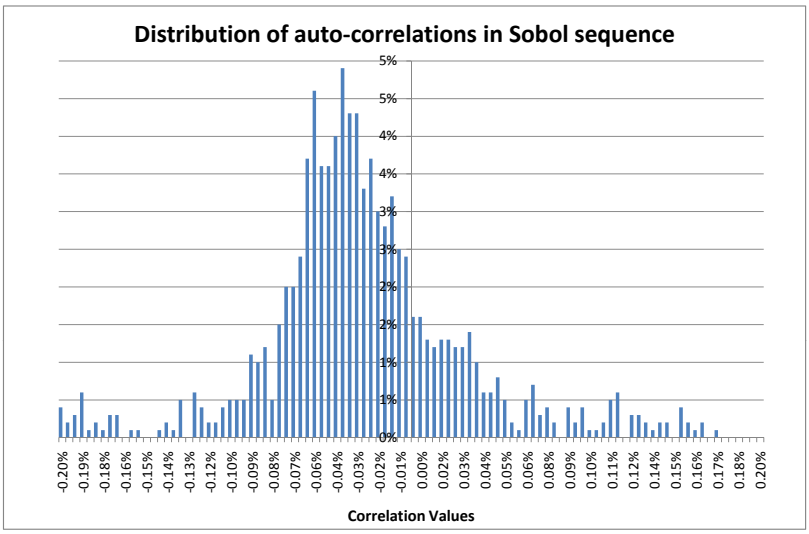

Figure 17: Distribution of the auto-correlations in Sobol sequence at $N = 24,000$ simulations

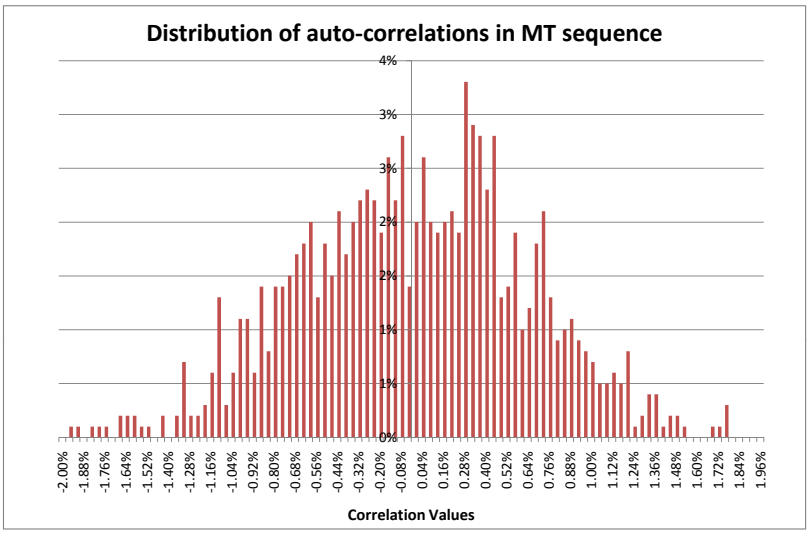

Figure 18: Distribution of the auto-correlations in Mersenne-Twister sequence at $N = 24,000$ simulations

Conclusion

Sobol sequence appears to have its moments and auto-correlations in general much closer to their ideal values, but the even-order moments suffer from a bias on the lower side. Mersenne-Twister sequence on the other hand does not show such biases. At the special number of simulations $2^n - 1$, these phenomena are magnified to the point where the 4 moment distributions reduce to a single value for all dimensions.

3.3 Bias and path construction

The fact that the Brownian bridge does not lead to significant improvements on Mersenne-Twister sequence but does on Sobol sequence is an indication that something particular happens in this sequence and that the Brownian bridge corrects it. In the previous section we went through an analysis of these two sequences in order to find a particular failure on Sobol side, and we have found one, a bias in the even moments as well as the auto-correlations. This does not mean however that the Brownian bridge repairs this particular defect in Sobol sequence. One could imagine that Sobol sequence would have an other deficiency and that the Brownian bridge would fix that one.

Let us try to see if we can make this clearer by a more detailed analysis of the pricing procedure. First of all, the poor convergence of the pricing can be observed on pure vanilla options as well. We chose barrier options in order to avoid a too special situation where the payoff would depend on only one time, which might result in special effects hiding the main problem. It can be easily checked that in fact the biases observed previously are also seen on pure vanillas, maybe even more pronounced. Let us also assume for simplicity that all time steps are equal and that we use $2^n - 1$ simulations such that all expectations are 0 and all variances are equal.

The price of a vanilla option depends only on the value of the spot at the last time T, and in particular depends only on the value of the Brownian motion at this time, W_T. Let us build the time grid by N equal time steps $dt = T/N$, and let us draw N Gaussian deviates $g_i, i = 1..N$ from Sobol uniform sequence. Since we are in the special simulation settings, we know that

- $\forall i, E(g_i) = 0$, where $E(.)$ is the expectation
- $\forall i, V(g_i) = v < 1$, where $V(.)$ is the variance
- the auto-correlations defined as $\rho_{ij} = E(g_i g_j) - E(g_i)E(g_j), i \neq j$, are small but biased on the negative side.

In the Brownian bridge construction, the value of the Brownian motion at T is easy to calculate as it is the one by which the algorithm starts:

$$W_T^{BB} = \sqrt{T} g_1. \tag{1}$$

In the incremental construction, it is obtained last after summing all the increments from 0, i.e.

$$W_T^I = \sum_{i=1}^{N} \sqrt{dt} g_i. \tag{2}$$

Using some simple algebra and the fact that we are in the special simulation configuration, we can calculate that

$$E(W_T^{BB}) = \sqrt{T}E(g_1) = 0 \tag{3}$$

$$E(W_T^I) = \sum_{i=1}^{N} \sqrt{dt}E(g_i) = 0 \tag{4}$$

$$V(W_T^{BB}) = Tv \tag{5}$$

$$V(W_T^I) = \sum_{i=1}^{N} dtV(g_i) + 2\sum_{i<j} dt\rho_{ij} \tag{6}$$

$$= T\Big(v + \frac{2}{N}\sum_{i<j}\rho_{ij}\Big). \tag{7}$$

Since the expectations are 0 in both constructions, the errors observed on the vanilla options are due to the errors on the variances (at leading order). The Brownian bridge will not see any impact of an increase in dimension in this simplified situation and the error is due mainly to the fact that the variance is not exactly equal to 1, i.e. $v < 1$. In the special number of simulations, equation (7) shows that the contributions from the variances of all the individual Gaussian deviates do not lead to a higher error than in the Brownian bridge, since they all add to the same amount. The difference comes from the fact that the individual Gaussian deviates are not exactly independent and that the total variance involves their correlations. Since in Sobol sequence the correlations have a bias on the negative side, the extra error in the Incremental construction is

$$C = \frac{2}{N}\sum_{i<j}\rho_{ij} \tag{8}$$

which we expect to be larger and negative for Sobol sequence, while it should be closer to 0 with an undetermined sign for Mersenne-Twister. In table 2 below we give the evaluation of this coefficient for $n = 14$ and several dimensions.

Table 2: Variance and Incremental construction extra error C

	Sobol			Mersenne-Twister		
d	v	C	v+C	v	C	v+C
10	0.9989	-0.0050	0.9939	0.9936	-0.0169	0.9767
100	0.9989	-0.0385	0.9604	1.0002	0.0017	1.0018
500	0.9989	-0.1121	0.8868	0.9999	-0.0045	0.9954
1000	0.9989	-0.1781	0.8208	1.0001	0.0089	1.0090
2000	0.9989	-0.2075	0.7914	1.0000	-0.0052	0.9947

We also show the result obtained with Mersenne-Twister (where v is the average variance over all dimensions). This shows that even though the "pure" variance part v is the same whatever the dimension is in Sobol sequence, the extra error due to the auto-correlations is more and more negative with the dimension such that the total variance $v + C$ goes farther and farther away from 1. This fact seems to us to strongly point towards the conclusion that it is the bias in the auto-correlations of Sobol sequence that causes (at least part of) the loss of efficiency at high dimensions, and that the Brownian bridge acts by reducing the number of Gaussian deviates required to reach a certain time that matters most for the product we are calculating. By having less Sobol Gaussian deviates to sum, we avoid more biased auto-correlations and reach a better accuracy.

4 Relevance to Parallel Programming

In this section we assume that the reader is familiar not only with the main notions in the implementation of Monte-Carlo simulations in parallel, but also with the architecture of GPUs and the programming language CUDA. We expect that most issues described here should be common to other architectures and languages. For more information about CUDA language and the GPU architecture, see for example [16, 17].

4.1 Inline and global memory path generation

When generating random numbers on a parallel architecture for use in a Monte-Carlo simulation, we are considering 2 types of strategies, both having their advantages and drawbacks.

- **Storage in memory** In a first step, we generate all the sequences, for all dimensions and paths, and store them in the memory. In a separate second step, we launch the Monte-Carlo simulation and calculate the underlying and product paths by loading the pre-calculated random numbers (or Brownian paths) from the memory to the function calculating the product in parallel.

  One of the advantages is to decouple the Monte-Carlo engine from the Brownian path creation, with the obvious gain in flexibility to change the type of path creation without touching the Monte-Carlo calculator, cleanly separating the model/products from the random number generation. One can even imagine storing the Gaussian deviates in text files and simply reading/loading them at each simulation, without even recalculating in the first step. An obvious drawback is that depending on the model, product, engine configuration, this may require a very large amount of computer memory.

  Indeed, the first step will lead to the storage of N paths of dimension d, so of $N \times d$ real numbers. Assuming these are encoded in a single precision format, taking 4 bytes of memory, a total of $4N \times d$ bytes will be required to store the paths necessary for the simulation of a single stochastic process. Let us illustrate this in the example where the architecture is a GPU programmed with the language CUDA. Since the two functions generating the random paths and performing the simulation are independent, these numbers have to be stored in the global memory of the GPU. The amount of such memory can vary a lot depending on the device, but at the time of writing, October 2011, it ranged from 256Megabytes (MB) in low-level general use devices to a few Gigabytes (GB) for the high-end professional-oriented ones.

  The simulation of a 30Y maturity product can easily require $N = 100,000$ simulations in order to reach a sufficient accuracy on the price as well as its sensitivities. In some models, such as Local Volatility, or for some products, such as continuous barriers, a large number of dimensions may be required, say for example $d = 500$ time steps. On top of this, the amount of required storage will be multiplied by the number of "factors" in the model, i.e. the number of sources of randomness. Such a multi-factor model can be a hybrid of Foreign Exchange Rate/Interest Rate hybrids, or Libor Market Models with several factors, or Stochastic Volatility models such as Heston. Or the product may be a basket option involving the generation of paths for many different assets.

  The configuration above already requires $4N \times d = 200$MB of memory per factor, such that the amount of memory can quickly be a limitation when considering multi-factor models or products. Adding to this is the problem of access to this memory, which may be slower than the few basic arithmetic operations required to directly calculate the random numbers.

- **Inline**

  The other possibility is to generate the random numbers right at the point of the implementation where they are used, i.e. just before calculating the spot value of the underlying(s) on each thread. The immediate advantage is that there is no more limitation of memory so a priori any kind of model or product, with any number of simulations, may be considered. The number of accesses to global memory are also greatly reduced, giving often a better performance than for the storage strategy.

  One drawback is that the calculation of the random numbers (or Brownian increments) has to be incorporated inside the Monte-Carlo engine, leading to a less elegant, less easily maintainable code.

  An other drawback is that the implementation of the Brownian bridge becomes more involved. Indeed, at each time step, we will need to generate a Brownian motion increment to calculate the spot, but in the Brownian bridge construction, the calculation of one increment involves the combination of 2 Brownian motion values pre-calculated recursively, such that the whole Brownian path must be calculated by the algorithm and stored in a memory location available to the thread, before it is able to pick the right increment to calculate the next value of the spot in time. The limitation of memory re-appears here, as register or shared memory are certainly insufficient to hold all the paths of the threads within a block on the grid, with a number of threads sufficiently large to benefit from latency hiding.

This motivates our study of the Brownian bridge and the mechanisms by which it reduces the variance in Sobol sequence. The previous sections have shown that the Brownian bridge enables us to remain on the winning side against the Pseudo-Random sequences at high dimensions. Unfortunately, we do not see how to implement it in parallel and inline in order to keep enough flexibility to run a wide range of models and products.

In the next section, we introduce a different algorithm, based on the shifting of Sobol sequence, which we show numerically can remove most of the bias discovered in Sobol sequence and lead to a convergence that can compete against the Brownian bridge while at the same time being easy to implement in parallel and inline.

4.2 Randomizing Sobol sequence

We consider a simple randomization of Sobol sequence along the lines of [13]. There are many ways to perform this kind of operations. Since [13] several improvements have been considered, for example digital randomization or scrambling according to [18]. A very detailed review of this subject can be found in [19]. It is not our purpose here to analyse many of these algorithms. Instead we want to choose a simple one that can easily be implemented in parallel, and see how the bias identified in the previous section is affected by the randomization process. Given the dimension d, we choose a vector $\Delta_i, i = 1..d$ such that $0 \leq \Delta_i < 1$. We denote by r_i^p the ith coordinate of the pth Sobol draw. The randomized Sobol vector $\tilde{r}_i^p$ is

$$\tilde{r}_i^p = r_i^p + \Delta_i, \quad r_i^p + \Delta_i < 1 \tag{9}$$

$$\tilde{r}_i^p = r_i^p + \Delta_i - 1, \quad r_i^p + \Delta_i \geq 1. \tag{10}$$

This algorithm is particularly simple and well suited for our purpose, implementation in parallel and without large data storage. Indeed, compared to the standard Sobol algorithm, it requires only

one additional real number per dimension, Δ_i. This number can be easily loaded in the shared memory at each time step. The rest is made of an *if..else* statement and at most 2 sums.

Let us now observe the statistical behaviour of this sequence following the same tests as in the previous sections. For simplicity we take $\Delta_i = (i-1)/d$ but many other choices can be made. First of all, in the special numbers of simulations $2^n - 1$ this randomized sequence no longer has the homogeneity properties of the original sequence, i.e. the means and skews are no longer exactly 0, and the variances and kurtosis are not equal at all dimensions. Let us look at the distributions of the first moments at 24,000 simulations only, for shortness of the expose. As opposed to Mersenne-Twister sequence, this sequence can compete with the original Sobol sequence as to the ranges of values the moments take, such that we can draw the 2 distributions on the same graph. First let us look at the means in fig. 19

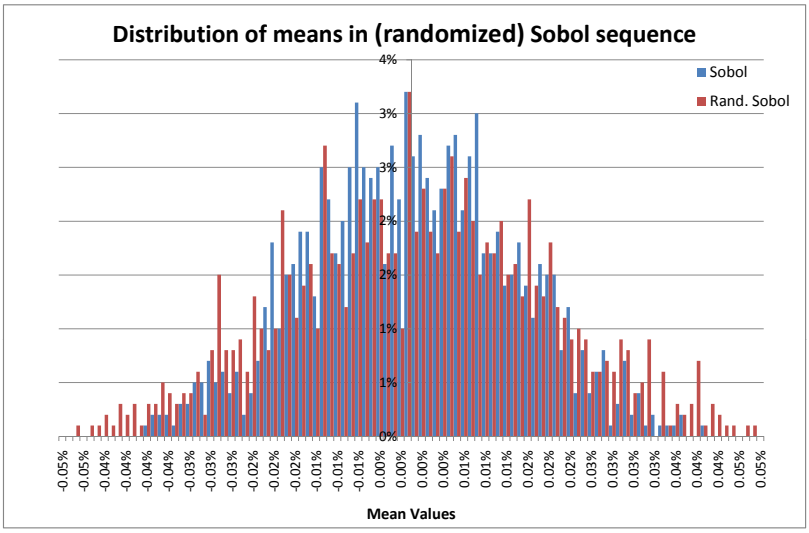

Figure 19: Distribution of the means in (randomized) Sobol sequence at $N = 24,000$ simulations

Although it may be possible to see a small loss on the randomized side, this is not obvious at all and we can see that the randomized Sobol sequences performs very well on the means.

Now for the variances,

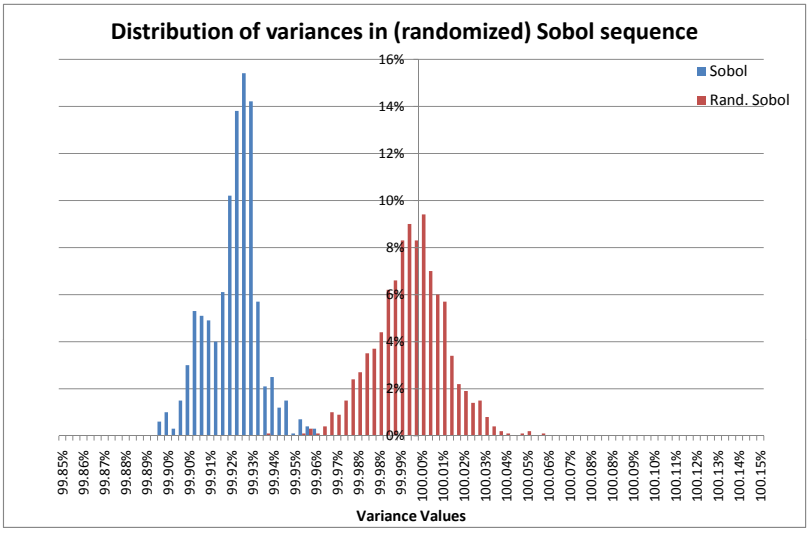

Figure 20: Distribution of the variances in (randomized) Sobol sequence at $N = 24,000$ simulations

we observe that the bias of the original sequence on the low variances has disappeared and the randomized sequence is well distributed around the theoretical value 1. Similar conclusions hold for the skew and kurtosis with a nearly equally good behaviour of the randomized sequence on the skew and a bias removed on the kurtosis. Finally for the auto-correlations,

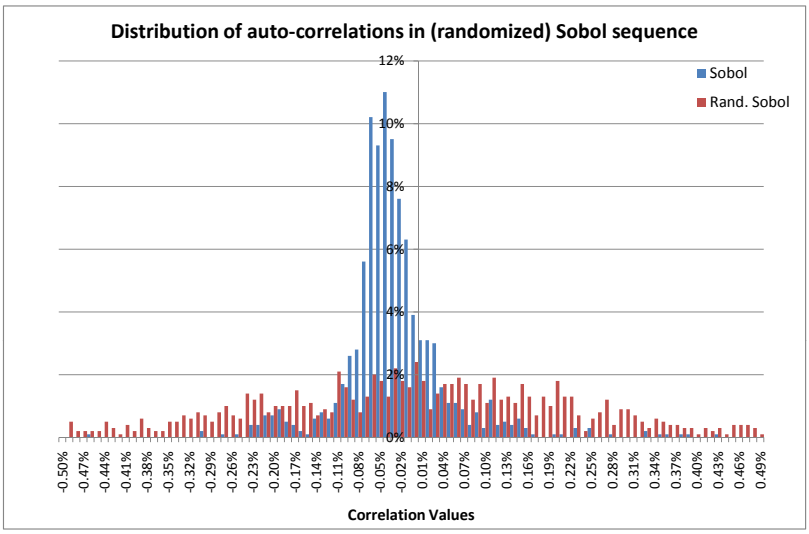

Figure 21: Distribution of the auto-correlations in (randomized) Sobol sequence at $N = 24,000$ simulations

we find that the bias also seems to be largely removed thanks to the randomization. This translate into a smaller extra error coefficient C in the incremental construction as in table 3 below.

Table 3: Variance and Incremental construction extra error C

	Sobol			Randomized Sobol		
d	v	C	v+C	v	C	v+C
10	0.9989	-0.0050	0.9939	1.0000	-0.0029	0.9972
100	0.9989	-0.0385	0.9604	1.0000	-0.0036	0.9964
500	0.9989	-0.1121	0.8868	1.0000	0.0019	1.0019
1000	0.9989	-0.1781	0.8208	1.0000	-0.0146	0.9854
2000	0.9989	-0.2075	0.7914	1.0000	-0.0067	0.9936

We can see that both the Brownian bridge variance v and the Incremental variance $v+C$ are good, now that the extra error C is greatly reduced in the randomized sequence due to the elimination of the bias in the auto-correlations.

This translates into a correction of the Monte-Carlo convergence problems encountered in Sobol sequence. For example, at 2000 dimensions, the convergence goes as

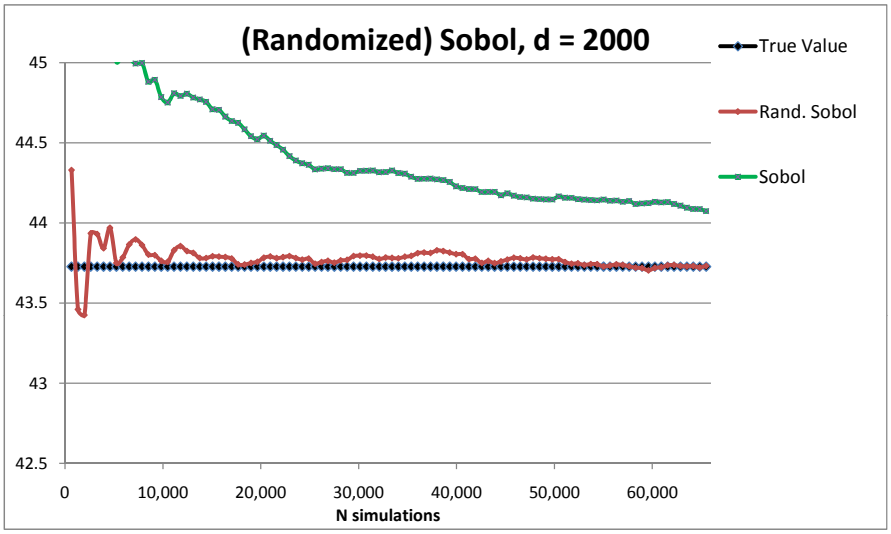

Figure 22: Sobol vs. Randomized Sobol sequences, d = 2000

which shows a great improvement thanks to the randomization. We can also compare the randomized sequence with the Incremental construction to the original Sobol sequence with the Brownian bridge, and we find

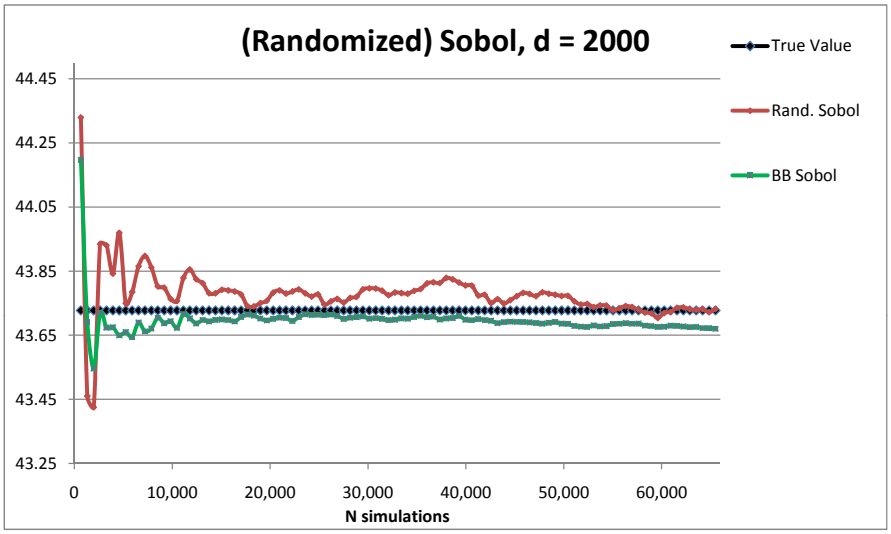

Figure 23: Brownian bridge Sobol vs. Incremental Randomized Sobol sequences, d = 2000

which shows that the randomized sequence in the Incremental construction can even compete against the Brownian bridge. These performances of the randomized sequence together with the disappearance of the bias in the auto-correlation are an other hint at the fact that the loss of efficiency of Sobol sequence at high dimensions is indeed connected to the bias. The randomized sequence appears as a very good and simple alternative to the combination of Sobol and Brownian bridge, especially in situations where the Brownian bridge is difficult to implement.

5 Other products and models

The products and models we have used until now, i.e. vanilla options or one-time barrier options with closed forms, and under Black model with constant volatility, are useful in order to simplify the problem and be able to perform an analysis and reach conclusions. However, a lot of models other than Black are used in practice, and Monte-Carlo simulations are not, in general, used to calculate the prices of vanilla options. In this section we look at a wider variety of products and models. We briefly come back to the issue of vanilla and barrier options to refine our conclusions, and then we turn to more complicated models/products such as PRDC swaps under a Local Volatility diffusion.

5.1 Vanilla options in Black model

Vanilla options have the particularity of depending on only the value of the spot at maturity. If the simulation is used to calculate only one option at a given maturity, then naturally the Brownian path will be constructed up to this maturity. This means in particular that in the Brownian bridge construction, the value of the Brownian motion at maturity is obtained using only one Gaussian deviate. According to our interpretation of the error in Sobol sequence with Incremental construction, we expect the Brownian bridge to be particularly efficient compared to Incremental method when pricing vanillas. This effect should be particularly strong in Black model with constant volatility and equal time steps, for which the increments of the Brownian motion multiplied by the standard deviation other each interval exactly sum to the total standard deviation times the final Brownian motion. We give an example of this below, with a put option struck at 80, maturity 20Y, and constant volatility 40%.

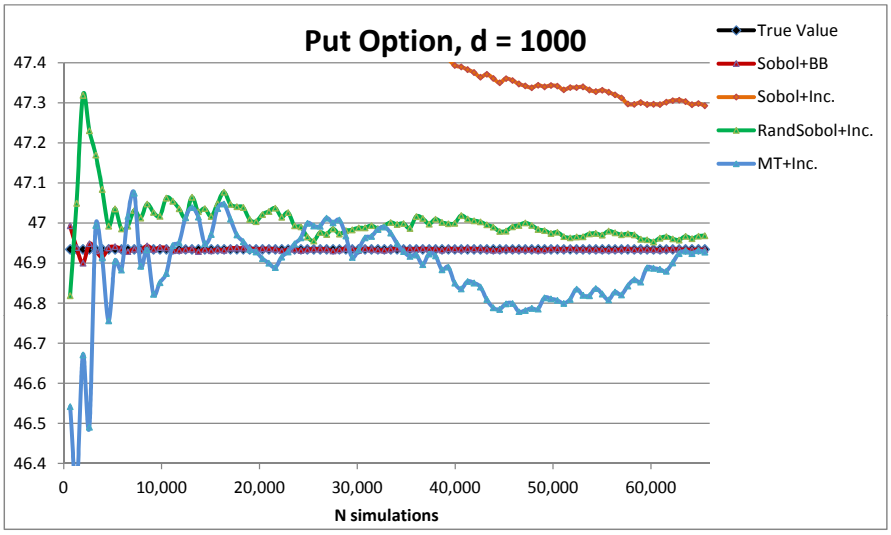

Figure 24: Put Option with several sequences, d = 1000

The improvement due to the Brownian bridge is spectacular, and the poor performance of Sobol sequence with Incremental construction is obvious. The randomized Sobol sequence performs quite well, with a similar convergence as Mersenne-Twister sequence, but is not as accurate as the Brownian bridge.

As one more illustration of the performance on vanilla options, we choose a digital option with the same maturity and volatility, at strike 60 and with low coupon at 2% and high coupon at 4%. We find the convergence

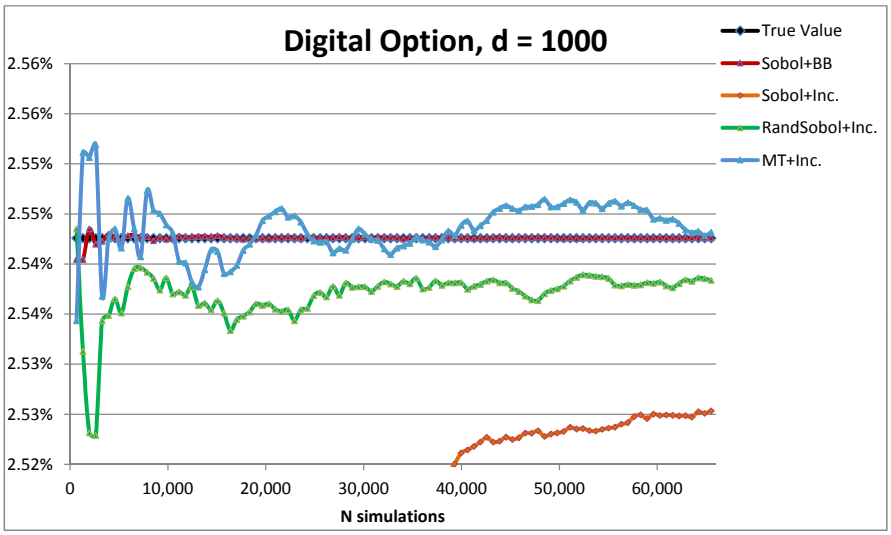

Figure 25: Digital Option with several sequence types, d = 1000

which shows a similar pattern as for the put options. Finally for the vanilla options we want to stress the fact that the randomization of Sobol sequence has not completely fixed the deficiency of Sobol sequence. This can be seen by plotting the convergence of the randomized sequence with both the Incremental and Brownian bridge constructions.

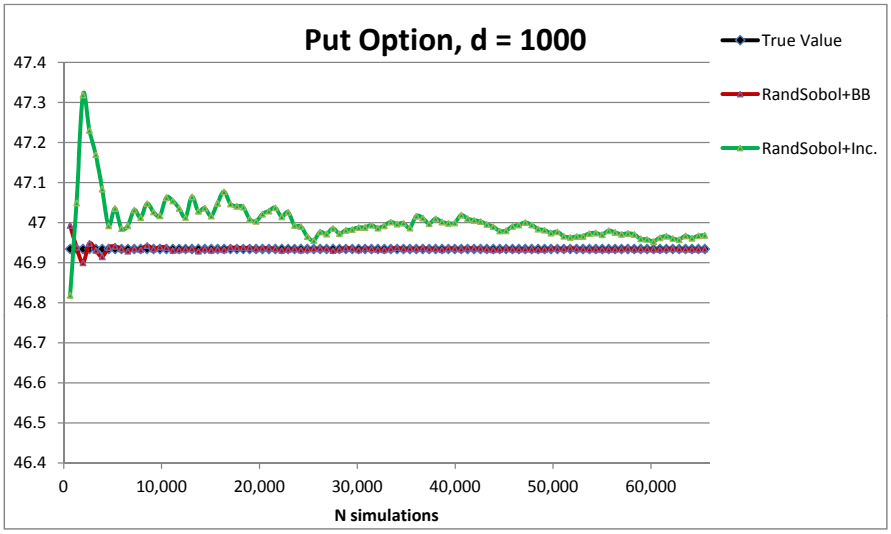

Figure 26: Put Option with Randomized Sobol sequence, d = 1000

Although the randomized Sobol sequence used here represents a good progress over the original sequence, it can still be improved by the use of the Brownian bridge, which means that there remains some deficiency in the original sequence that could not be fixed by this randomization.

5.2 Barrier options in Black model

We have already tested some barrier options in this document. For them the conclusion is more difficult to draw. Fig. 23 shows for example that contrary to the vanilla options, the superiority of the Brownian bridge over the randomization is no longer obvious. We tried all kinds of configurations by changing the values of the barrier, of the strike, of the maturity, of the barrier monitoring time. We do not show them here because they do not bring more to the discussion. In some instances the Brownian bridge can perform better than the Incremental randomized Sobol sequence, but in other instances it is the opposite. The randomized sequence is often more accurate than Mersenne-Twister but this cannot be taken for granted as one can find several examples of the contrary. Which sequence, among Sobol with Brownian bridge, randomized Sobol and Mersenne-Twister with Incremental construction, performs best depends on the product. What is clear is that Sobol sequence with Incremental construction performs poorly compared to the other three possibilities, in nearly all configurations with high dimensions.

5.3 Vanilla PRDCs in Local Volatility model

Here we use the Local Volatility model as calibrated thanks to Dupire's formula [1]. In this model the volatility becomes not only time-dependent but also spot-dependent, and therefore, the last spot at maturity cannot be written exactly in terms of the last Brownian motion. As products, we calculate Power-Reverse-Dual-Coupon swaps for which the structured leg is a sum of coupons of the form

$$C_i = \max(g_f \frac{FX}{FX_0} - g_d, 0) \tag{11}$$

where g_f, g_d are called the foreign and domestic coupons, FX_0 is a scaling factor fixed at the beginning of the trade, and FX is the spot Foreign Exchange rate, modelled with the Local Volatility calibrated from the FX surface. The swap exchanges a series of these coupons C_i quarterly against funding coupons based on Libor. The maturity is 30Y and the notional 10,000. In this section, we do not consider an exotic exercise, such that the Present Value (PV) of the structuring leg is essentially a sum of call options at different maturities and strikes.

We do not want to go into more details as to how the volatility surface is calibrated, since this would not make the discussion clearer. Our interest here is rather in the fact that the path of the spot is built only once up to the maturity of the last coupon, such that the intermediate coupons are no longer calculated from the last point of the path, as was the case for simple vanilla options. This means that the Brownian motions involved in the calculation of the intermediate options are derived from several Gaussian deviates even when using the Brownian bridge. Furthermore, the locality of the volatility also implies that in order to calculate the spot at a given time, the increments of the Brownian motion, weighted by the local volatility, no longer sum to the original value of the Brownian motion at this time. All this combined makes the calculation of this product more complicated and the advantage of the Brownian bridge less obvious, at least from a theoretical point of view.

On the other hand, since each cash flow is still a vanilla option, we can calculate the value of this product exactly, provided we know the implied volatility surface[2]. In fig. 27 we show the convergence of the PV of the structuring leg with different sequences.

[2]Here we use SVI [20]. Note that the knowledge of the implied volatility surface is required to calibrate the model anyway.

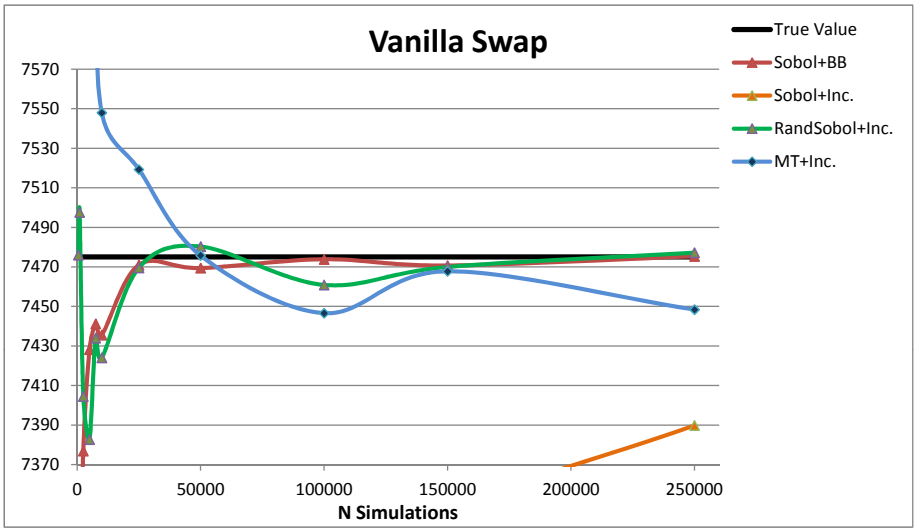

Figure 27: PRDC vanilla swap with several sequences, d = 1000

We can see that the Brownian bridge is again very efficient, and that the randomized Sobol sequence performs nearly as well. MT sequence is still oscillating at 250K simulations, and Sobol sequence with Incremental construction is off. This shows that the switch to the Local Volatility model and the accumulation of cash flows did not disturb the patterns observed on simpler models and products.

5.4 Exotic PRDCs in Local Volatility model

Here we add an exotic exercise to the above PRDC structuring leg. We consider a Targeted Accrual Redemption Note (TARN) feature, for which the structuring coupon is accumulated up to a limit after which the remaining cash flows are cancelled. The monitoring of the sum of cash flows is done after every cash flow is paid, and since the present PRDC coupon is paid quarterly up to 30Y, this amounts to 120 monitoring dates. The TARN limit is set at 30% of the notional. This time there is no true value to compare the Monte-Carlo results with.

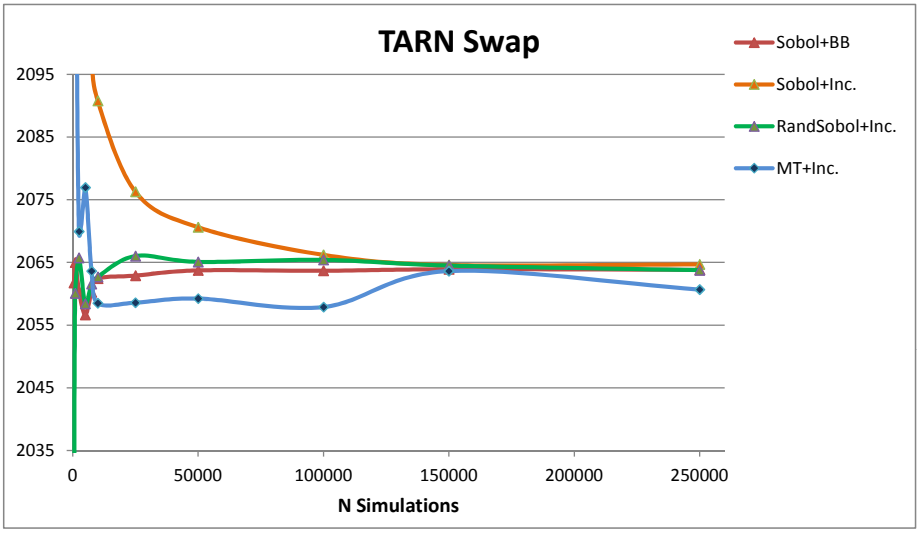

Figure 28: TARN PRDC with several sequences, d = 1000

The TARN exotic feature reduces the PV of the leg greatly due to the cancellation of a number of paths hitting the limit. The convergence is slightly faster than for the vanilla swap[3], such that even though there is no known true value here, we can still conjecture that the excellent agreement between the Brownian bridge and the randomized Sobol sequence, from as early as 50K simulations, points to a true value around 2064. MT sequence shows the same pattern of oscillation as for the vanilla swap. The noticeable difference here is that Sobol sequence with the Incremental construction is no longer so off, but still has a slower convergence than the Brownian bridge and the randomized sequence.

[3]Our understanding of this phenomenon is that the PRDC coupon, as a call option, suffers from a slow convergence due to the fact that it is not limited from above, and Monte-Carlo outlier paths tend to give it very high values in some rare instances, leading to more noise than for example on a put option-like coupon. The TARN feature, by cutting off late cash flows, tends to discard these outliers and thus leads to a faster convergence.

6 Conclusion

With the goal to find replacements to the Brownian bridge construction for efficient inline implementation in parallel, we investigated the mechanisms underlying the well-known loss of convergence speed of Sobol sequence at high dimensions and the improvements coming from applying the Brownian bridge construction to this sequence. By isolating the impact of the main factors, we reached the conclusion that it is a deficiency specific to Sobol sequence, at least absent in the Pseudo-Random sequences we tested, that the Brownian bridge seems to compensate for or avoid.

Our results also point to the fact that, although commonly cited as a possible cause of these phenomena, the supposed loss of uniformity or quality of projections in Sobol sequence at the coordinates at high dimensions is not to blame for the observed failure. In particular, having in mind the special properties of the Gaussian deviates derived from Sobol sequence proved in the appendix, it appears to us that the existence itself of the loss of uniformity is far from obvious. Moreover, we think that recent progresses in the search for direction integers also render the assumption of worse projections less likely [10]. Our tests using the high dimensional coordinates only confirms this interpretation.

With a subsequent statistical analysis of Sobol sequence, we showed numerically that the distribution of its variances, but more importantly of its auto-correlations, is biased on the lower side, a phenomenon that does not occur for other sequences such as Mersenne-Twister. By using a simple model of the Brownian motion constructions, and the particular homogeneity properties of the Gaussian deviates derived from Sobol sequence at $2^n - 1$ iterations, we showed that the bias present in the auto-correlations will impact the Incremental construction much more than the Brownian bridge, especially for vanilla options at high dimensions. We calculated the contribution of the auto-correlations to the total variance and showed numerically that it can be rather significant, motivating our conjecture that it is (at least mostly) by removing the impact of these auto-correlations that the Brownian bridge can recover the excellent convergence of Sobol sequence even at high dimensions. With the iterative halving of intervals, the Brownian bridge makes it possible to reach given points in time by summing less Gaussian deviates and thus involving less auto-correlations. This can explain why the Brownian bridge is efficient on Sobol sequence but not on Mersenne-Twister sequence, which does not exhibit such a bias in the auto-correlations.

Taking the example of GPUs programmed with CUDA, we explained the difficulty of efficient implementation of the Brownian bridge in parallel at high dimensions, and showed how even a very simple randomization of Sobol sequence following [13] can remove most of the auto-correlation bias and then achieve an excellent speed of convergence at high dimensions, competing against the Brownian bridge. This shifting algorithm can easily be implemented in parallel and inline with at most 2 additions and an *if..else* statement on top of the original Sobol algorithm.

A further analysis of the convergence of this randomized Sobol sequence showed that although yielding a very good improvement over the original sequence, it is not as efficient as the Brownian bridge on simple vanilla options. We do not have a pessimistic view on this though, since Monte-Carlo simulations are used more for the calculation of complicated products, in particular exotics, than for vanilla options. For PRDC swaps with TARN features priced under Dupire's Local Volatility model [1] at high dimensions, we find that the randomized Sobol sequence performs very well, at a comparable level with the Brownian bridge. These two sequences are more accurate than Mersenne-Twister, while the original Sobol sequence in Incremental construction performs very poorly especially on the vanilla swaps.

We think that the analysis in this work has brought some light on the behaviour of Sobol sequence and its interaction with the Brownian bridge at high dimensions. We also showed that randomizing Sobol sequence appears as a very promising strategy when unable to implement the

Brownian bridge in an efficient manner at high dimensions. However, a few tests have shown that, although representing a very significant improvement over the un-shifted Sobol sequence, this is not as efficient as the Brownian bridge, at least on vanilla options. Fortunately, some more advanced randomization techniques exist, such as digital randomization or the scrambling algorithms of [18]. We think it would be very valuable to extend this study to incorporate these more advanced algorithms, possibly including different products in order to increase the confidence as to the range of validity of these results in Finance.

Acknowledgements

We would like to thank J. Zhu and S. Joe for their guidance on the existing literature. We thank T. Wong and G. Poggiaspalla for their comments on the manuscript and their suggestions during testing and writing.

References

[1] B. Dupire, "Pricing with a smile," *Risk*, vol. 7, pp. 18–20, 1994.

[2] D. M. Dang, C. C. Christara, K. R. Jackson, and A. Lakhany, "A PDE Pricing Framework for Cross-Currency Interest Rate Derivatives," *Working Paper*, 2010.

[3] I. M. Sobol, "Distribution of points in a cube and approximate evaluation of integrals," *USSR Computational Mathematics and Mathematical Physics*, vol. 7, pp. 86–112, 1967.

[4] I. A. Antonov and V. M. Saleev, "An economic method of computing lp-sequences," *USSR Computational Mathematics and Mathematical Physics*, vol. 19(1), pp. 252–256, 1979.

[5] P. Jaeckel, *Monte Carlo Methods in Finance*. Wiley Finance, 2002.

[6] S. H. Paskov and J. F. Traub, "Faster valuation of financial derivatives," *J. Portfolio Management*, vol. 22, pp. 113–120, 1995.

[7] R. E. Caflisch, W. Morokoff, and A. B. Owen, "Valuation of mortgage backed securities using brownian bridges to reduce effective dimension," *J. Comp. Finance*, vol. 1, pp. 27–46, 1997.

[8] X. Wang and I. H. Sloan, "Why are high-dimensional finance problems often of low effective dimension?," *SIAM J. Sci. Comput.*, vol. 27, pp. 159–183, 2005.

[9] X. Wang and I. H. Sloan, "Brownian Bridge and Principal Component Analysis: Towards Removing the Curse of Dimensionality," *Working Paper*, March 2006.

[10] S. Joe and F. Y. Kuo, "Constructing Sobol sequences with better two-dimensional projections," *SIAM J. Sci. Comput.*, vol. 30, pp. 2635–2654, 2008. http://web.maths.unsw.edu.au/ fkuo/sobol/index.html.

[11] A. Bernemann, R. Schreyer, and K. Spanderen, "Accelerating Exotic Option Pricing and Model Calibration Using GPUs," *Working Paper*, 2011.

[12] M. S. Joshi, "Graphical Asian Options," *Working Paper*, 2009.

[13] R. Cranley and T. N. L. Patterson, "Randomization of number theoretic methods for multiple integration," *SIAM Journal on Numerical Analysis*, vol. 13(6), pp. 904–914, 1976.

[14] P. L'Ecuyer, F. Blouin, and R. Couture, "A Search for Good Multiple Recursive Random Generators," *ACM TOMACS*, vol. 3, pp. 87–98, 1993.

[15] G. Marsaglia, "The Mother of All Random Generators," 1994. ftp.taygeta.com.

[16] NVIDIA, "NVIDIA CUDA C Programming Guide," *User Manual*, 2011. http://developer.nvidia.com/category/zone/cuda-zone.

[17] D. B. Kirk and W. W. Hwu, *Programming Massively Parallel Processors*. Morgan Kaufmann, 2010.

[18] A. B. Owen, "Local antithetic sampling with scrambled nets," *ArXiv*, 2008.

[19] P. L'Ecuyer and C. Lemieux, "Recent Advances in Randomized Quasi-Monte Carlo Methods," *Modeling Uncertainty: An Examination of Stochastic Theory, Methods, and Applications*, pp. 419–474, 2002. Kluwer Academic Publishers.

[20] J. Gatheral, "A parsimonious arbitrage-free implied volatility parameterization with application to the valuation of volatility derivatives," Global Derivatives & Risk Management 2004, Madrid, May 2004.

A Proofs

By calculating the first few draws of any Sobol sequence, it is easy to convince oneself that these uniform numbers are drawn in a very specific pattern of successive "layers" where each layer covers the interval (0,1) perfectly regularly and with twice more "precision" as the previous layer. Let us make this more precise by looking at the first few draws

$$\begin{array}{ll} \text{Layer 1:} & \frac{1}{2} \\ \text{Layer 2:} & \frac{1}{4}, \frac{3}{4} \\ \text{Layer 3:} & \frac{1}{8}, \frac{3}{8}, \frac{5}{8}, \frac{7}{8} \\ & . \\ & . \\ \text{Layer p:} & \frac{2k+1}{2^p}, \; k = 0..2^{p-1} - 1 \end{array} \tag{12}$$

One remark is that at the end of each layer we have the most homogeneous distribution of points possible on the interval (0,1) for the number of points drawn. Second, each layer is symmetric around $\frac{1}{2}$, which as we will see later in this section, has important consequences for the Gaussian deviates. Note that this pattern appears to be the same in all dimensions and independently of the direction integer, the difference residing only in the order in which the points are drawn within a layer.

These properties have been known for long and are due to the fact that Sobol sequence is a particular case of digital net [3]. In section A.1, we recall simplified proofs in the special case of Sobol sequence. These properties have important consequences on the Gaussian deviates drawn with the cumulative normal distribution function, which we prove in section A.2[4]. Our notations follow [5] closely so the reader is invited to read this reference in case some expression or notation is unclear here.

A.1 Equi-distribution of the uniforms

Definition 1. *For $p \geq 1$, we denote by p-th layer the integers $2^{p-1} + k$, $k = 0..2^{p-1} - 1$.*

Proposition 1. *For all integers of the p-th layer, the leftmost non-zero bit is the p-th bit. Conversely, any integer whose leftmost non-zero bit is the p-th bit is in the p-th layer.*

Proof. This follows directly from the decomposition in base 2 of $n = 2^{p-1} + k$, $k = 0..2^{p-1} - 1$. □

Proposition 2. *For any integer n in the p-th layer, the Gray code of n, denoted $G(n)$, is also in the p-th layer.*

Proof. The particular type of Gray code we are talking about here is defined as

$$G(n) = n \oplus_2 [n/2].$$

[4]We have not been able to find in the literature where the original proofs of several properties mentioned here first appeared. For the properties on the Gaussian deviates, we are not aware of their existence. If however they did exist, we apologize in advance for not being able to cite their authors and will do so in a future revision.

From proposition 1, the leftmost non-zero bit of n is the p-th, and since division by 2 shifts all bits to the right, the p-th bit of $G(n)$ is obtained by the XOR operation

$$1 \oplus_2 0 = 1.$$

All bits on the left of the p-th bit of n and $n/2$ are zero so the XOR operation will result in 0, and thus the leftmost non-zero bit of $G(n)$ is the p-th.

□

Definition 2. *At dimension k, we denote the l-th direction integer by v_{kl}. Recall that it must satisfy the following 2 conditions*

1. *only the l leftmost bits can be non-zero*
2. *the l-th leftmost bit is set.*

The n-th draw at dimension k is defined as

$$x_{nk} = \sum_{i=1}^{b} v_{ki} \mathbf{1}_{\{i\text{-th bit (from the right) of } G(n) \text{ is set}\}}$$

where b is the number of bits in an unsigned integer and $\sum$ represents XOR sums.

Proposition 3. *At dimension k, the 2^{p-1} Sobol integer draws in the p-th layer are*

$$2^{b-p}(1+2m), \quad m = 0..2^{p-1}-1.$$

Proof. Let us consider the n-th draw x_{nk} at dimension k with n in the p-th layer. From proposition 2, the Gray code of n is in the p-th layer so x_{nk} can be written as

$$x_{nk} = v_{kp} \oplus_2 \sum_{i=1}^{p-1} v_{ki} \mathbf{1}_{\{i\text{-th bit (from the right) of G(n) is set}\}} \tag{13}$$

since we know that the p-th bit of $G(n)$ is set while no higher bit is. Constraint 1. in Definition 2. and eq. (13) imply that only the p leftmost bits of x_{nk} can be set and that the p-th is, i.e.

$$x_{nk} = \sum_{l=b-p}^{b-1} a_l 2^l \tag{14}$$

with $a_{b-p} = 1$ and $a_l = 0$ or $a_l = 1$ for $l > b-p$. Consequently x_{nk} can be rewritten as

$$\begin{aligned} x_{nk} &= 2^{b-p}\Big(1 + 2\sum_{l=b-p+1}^{b-1} a_l 2^{l+p-b-1}\Big) \\ &= 2^{b-p}\Big(1+2m\Big) \end{aligned} \tag{15}$$

with $m = \sum_{l=b-p+1}^{b-1} a_l 2^{l+p-b-1}$. The smallest possible value for m is when all $a_l = 0$, i.e. $m = 0$, and its largest possible value is when all $a_l = 1$, i.e. $m = 2^{p-1}-1$.

What remains to be shown is that m takes *all* the values between 0 and $2^{p-1}-1$. Here we use the fact that no integer can be generated twice by Sobol sequence, a property of digital nets [3]. Since there are 2^{p-1} integers n in the p-th layer, and since all of them lead to a different x_{nk}, there must be 2^{p-1} x_{nk} which implies that m takes 2^{p-1} distinct values and therefore all values in $[0, 2^{p-1}-1]$.

□

Definition 3. *The uniform draw y_{nk} corresponding to x_{nk} is defined as*

$$y_{nk} = \frac{1}{2^b} x_{nk}.$$

Proposition 4. *For any dimension k*

1. *the 1st layer contains $\frac{1}{2}$ as its only draw*
2. *in the set of 2^{p-1} uniform draws obtained from the p-th layer of integers with $p > 1$, for each uniform draw equal to $\frac{1}{2} + \delta$, $0 < \delta < \frac{1}{2}$, $\frac{1}{2} - \delta$ is also a draw in the p-th layer.*

Proof. Point 1. follows directly from proposition 3. with $p = 1$.

In the p-th layer with $p > 1$, take a uniform draw with value

$$y_{n_0 k} = \frac{1}{2^p}(1 + 2m_0)$$

for $m_0 \in [2^{p-2}, 2^{p-1}-1]$. Then $y_{n_0 k} = \frac{1}{2} + \delta$ with $\delta = \frac{1}{2^p}(1+2m_0) - \frac{1}{2} > 0$. Then

$$\begin{aligned} \frac{1}{2} - \delta &= 1 - \frac{1}{2^p}(1 + 2m_0) \\ &= \frac{1}{2^p}(2^p - 1 - 2m_0) \\ &= \frac{1}{2^p}(1 + 2k_0) \end{aligned} \tag{16}$$

where $k_0 = 2^{p-1} - 1 - m_0$ is an integer and $0 \leq k_0 \leq 2^{p-1} - 1$. Proposition 3. implies that $\frac{1}{2} - \delta$ is a draw in the p-th layer.

□

Definition 4. *For $p > 1$, proposition 3. implies that we can define a series of distinct real numbers δ_i^p, $i = 1..2^{p-2}$, such that $0 < \delta_i^p < \frac{1}{2}$ and the p-th layer of uniform draws is composed of the 2 subsets $u_i^{p\pm} = \frac{1}{2} \pm \delta_i^p$. The 1st layer is composed of $u^1 = \frac{1}{2}$.*

Proposition 5. *The moments of the distribution estimated by the first $2^n - 1$ uniform draws of Sobol sequence are equal at all dimensions. The first moment is equal to $\frac{1}{2}$.*

Proof. First of all note that the first $2^n - 1$ iterations are made of the draws generated from the layers up to order n. Indeed, there are 2^{p-1} integers in the $p-$th layer, and consequently the layers up to order n contain all together

$$\sum_{k=1}^{n} 2^{k-1} = 1 + 2 + \cdots 2^{n-1} = 2^n - 1$$

integers.

Next let us emphasize one particular implication of proposition 3., which is that the set of draws within a layer is the same at all dimensions, only the order in which the draws appear varies. The sums of any powers of draws over a layer are therefore independent of the dimension, and so are the total sums up to iteration $2^n - 1$, since these are the sums of all partial sums in each layer up to order n. Consequently, the moments are the same at all dimensions.

In particular for the first moment m_1, we obtain

$$\begin{aligned} m_1 &= \frac{1}{2^n - 1} \sum_{i=1}^{2^n - 1} y_{ik} \\ &= \frac{1}{2^n - 1} \Big(u^1 + \sum_{p=2}^{n} \sum_{j=1}^{2^{p-2}} (u_j^{p+} + u_j^{p-}) \Big) \\ &= \frac{1}{2^n - 1} \Big(\frac{1}{2} + \sum_{p=2}^{n} 2^{p-2} \Big) \\ &= \frac{1}{2} \end{aligned}$$

where the step from the second to the third line is allowed by the cancellations due to proposition 4.2.

□

A.2 Equi-distribution of the Gaussian deviates

Proposition 6. *The moments of the normal distribution estimated by applying the inverse cumulative normal distribution function $\mathcal{N}^{-1}(.)$ to the first $2^n - 1$ uniform draws of Sobol sequence are equal at all dimensions. Moreover, the odd moments are equal to 0.*

Proof. The fact that the moments are equal at all dimensions follows from a similar argument as for the estimation of the uniform distribution. Let us call g_l the l-th Gaussian deviate calculated from the l-th uniform draw u_l by $g_l = \mathcal{N}^{-1}(u_l)$. The sum of any power of the first $2^n - 1$ g_l can be written as the sum on layers up to order n and the sums on layers are independent of the dimension.

With obvious notations, let us define the a-th moment m_a as

$$\begin{aligned} m_a &= \frac{1}{2^n - 1} \sum_{l=1}^{2^n - 1} (g_l)^a \\ &= \frac{1}{2^n - 1} \sum_{p=1}^{n} \sum_{i=0}^{2^{p-1} - 1} (g_i^p)^a \\ &= \frac{1}{2^n - 1} \Big(g^{1a} + \sum_{p=2}^{n} \sum_{j=1}^{2^{p-2}} [(g_j^{p+})^a + (g_j^{p-})^a] \Big) \end{aligned} \tag{17}$$

where we have decomposed the total sum into partial sums on layers and further into $p = 1$ and $p > 1$ layers with $\pm$ parts. Now on each layer (except for the 1st) the sums decompose on sums other Gaussian deviates generated from uniforms larger/smaller than $\frac{1}{2}$, the $g_j^{p\pm} = \mathcal{N}^{-1}(u_j^{p\pm})$. The asymmetry of $\mathcal{N}^{-1}$ around $\frac{1}{2}$, i.e.

$$\begin{aligned} \mathcal{N}^{-1}(\frac{1}{2}-x) &= -\mathcal{N}^{-1}(\frac{1}{2}+x) \\ \mathcal{N}^{-1}(\frac{1}{2}) &= 0 \end{aligned} \tag{18}$$

implies that

$$\begin{aligned} g_i^{p-} &= -g_i^{p+} \\ g^1 &= 0. \end{aligned}$$

It is then straightforward to see that the r.h.s of eq. (17) reduces to 0 when the moment order a is odd.

□