Algorithmica
Research AB

Using a Sobol sequence for Monte Carlo simulation
using Quantlab 3.0

Using a Sobol sequence for Monte Carlo simulation

This document will give some guidance on how to use the Sobol sequences in Quantlab 3.0. In particular we will create an example that will price a simple arithmetic average option using a Sobol sequence instead of ordinary random Monte Carlo. The example is very similar to the example "Running a Monte Carlo on Quantlab cluster" found in the same document library.

Background

There are many ways to address performance problems in mathematical finance. The best ways usually require an in-depth knowledge of the specific calculation or program that has performance problems. By knowing the problem, it is easy (sometimes) to understand if it lends itself to parallel computations, re-coding in a faster language, use of convergence speed-up techniques, or by so called quasi random sequences.

Many problems require a simulative solution, especially financial risk problems, where the tail of the return distribution is the interesting output. Also, indeed, in pure pricing problems speed is an issue and will benefit from a faster Monte Carlo simulation.

A Sobol[1] sequence is a quasi random number series that evenly fills the probability space. There are other sequences that are similar, but for financial applications the Sobol is the more widely used.

Are there any limitations? Well yes, since many problems require paths or have many assets to simulate, more than one dimension in the probability hypercube is needed. A well designed normal random generator can have, as the Mersenne Twister[2], 624 dimensions. Also the Sobol sequence has a dimensional limitation. In the early days, this was limited to double digits, but now run into the thousands of dimensions. Quantlab's Sobol generator currently provides 1111 dimensions.
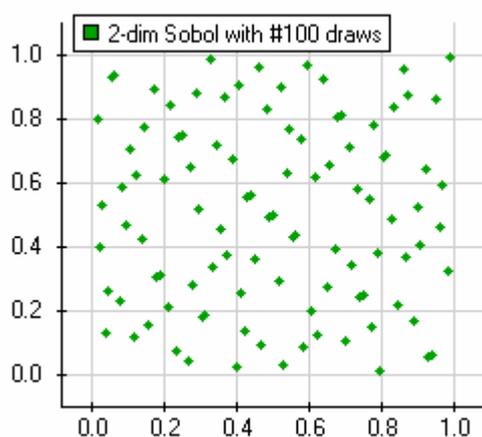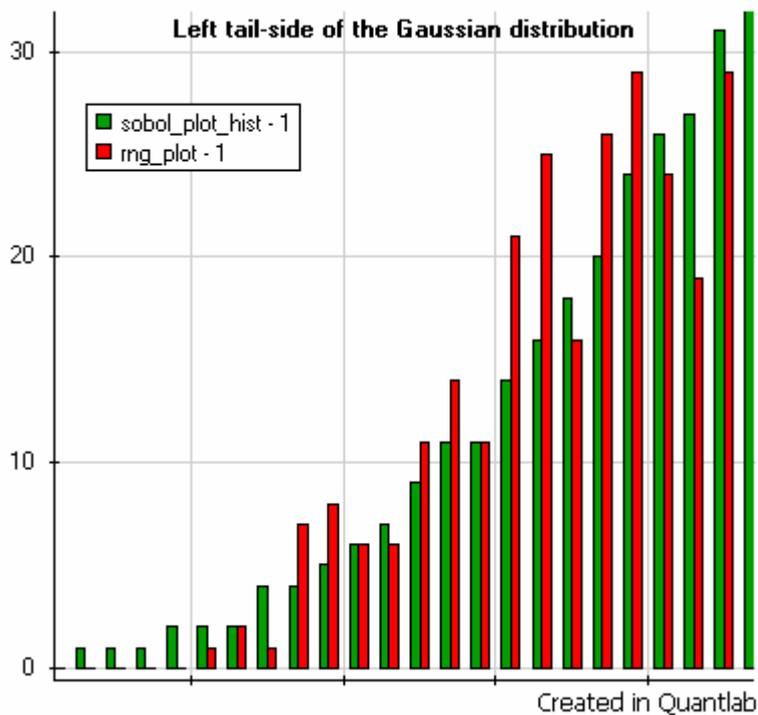

Created in Quantlab
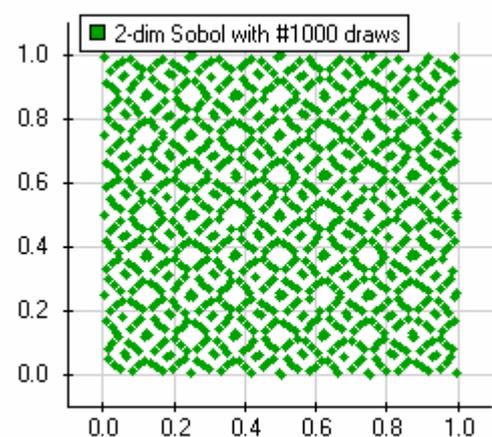
Created in Quantlab

---

[1] Named after Prof. I. M. Sobol', a Russian mathematician.
[2] Mersenne Twister MT19937, by Matsumoto and Nishimura 1997

Left tail-side of the Gaussian distribution

Created in Quantlab

As mentioned before, the left tail of the return distribution is especially useful in financial risk analytics. In order to quickly get a smooth high percentile estimate, the Sobol sequence generates a smooth tail already using 1000 draws. The standard Gaussian random numbers have a high degree of variability leading to a need for more draws.

Example in Quantlab 3.0

Below is an implementation using the Sobol sequence instead of the normal random number generator - replicating the Asian Option presented in the example for using cluster Monte Carlo simulation.

```
out number myAsianOptionPayoffSobol(logical iscall, number strike, number S0, number time,
                        number rate, number carry, number vol, number steps, number sim)
{
        // Generate a sobol sequence having steps number of dimensions
        // There is an optional second argument that can skip to a different starting value
        sobol_gen s_rand = sobol_gen(steps);
        vector(number) sobol_path[steps];

        // Implement an ordinary monte carlo simulation of a path dept option
        number dt = time / steps;
        number drift = (carry - vol*vol / 2) * dt;
        number dt_vol_sqr = vol * sqrt(dt);
        number z;

        if(iscall)
                z=1;
        else
                z=-1;

        number Payoffsum = 0;
        for(number s = 1; s <=sim; s++){
```

```
        number St = S0;
        number Ssum = 0;

        // Draw s-th complete first vector containing n-steps quasi random numbers
        // Since we are pricing using a log-normal model we create a gaussian number
        sobol_path = s_rand.gauss();

        for(number j = 0; j < steps; j++){
                St = St * exp(drift + dt_vol_sqr * sobol_path[j]); //get the j-th step
                Ssum = Ssum + St;
        }

        number Savg = Ssum / steps;
        Payoffsum = Payoffsum + max(z * (Savg - strike), 0);
    }
    //Normalise the payoff sum with the number of simulations and discount to get price
    return exp(-rate * time)*Payoffsum/sim;
}
```
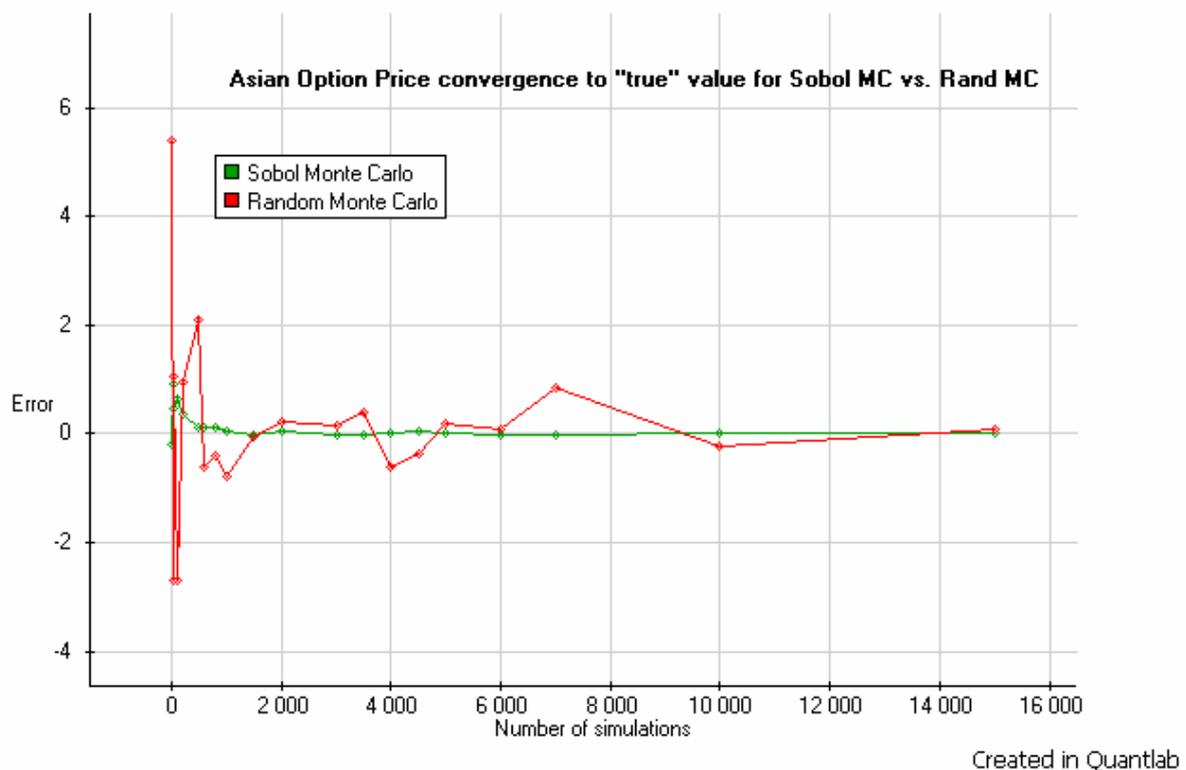
In order to compare the results and the convergence, we create an identical function using the normal rng() function and present the results as deviations from the "true" result.

It is easy to see that the Sobol Monte Carlo converges to the "true" value and stays within an error that is less than 0.25% already from around 1500-2000 simulations. As the normal Monte Carlo simulation need a factor 10 more simulations to reach the same stability.

From a time-saving perspective, the Sobol and Random Monte Carlo take about the same time to perform given the same number of draws and dimension. Having about a factor 1:10 in number of draws for the same convergence leads to a 10 times faster result, all things equal.