# Monte Carlo Methodology (for CQF Projects)

## Improvement for Credit Spread Pricing

Please see the section on Averaging PL and DL. The number of simulations required for convergence (with naive Excel random numbers) can reach into $100,000$s, particularly for the higher kth-to-default. Check sensibility of your pricing result (is it as expected, is its variance reducing) even if it *appears* to converge after $10,000$s simulations.

The key reference for simulation design and sampling strategies, particularly when sampling from copula required, is a textbook on *Monte-Carlo Methods in Finance* by Peter Jaekel.

## Improvement for HJM Simulations

With $k = 3 \ldots 5$ factors (volatility functions) used to simulate the curve, the dimensionality of Monte Carlo simulation is sufficiently low. Potentially, fractal patterns in random numbers across the depth of multiple dimensions can lead to pricing patterns that can be arbitraged.

There is no requirement to simulate the curve for up to 25 years into the 'future time' (with $\Delta t = 0.01$ that would be 2500 rows on the HJM Spreadsheet). You can start with a 5-year future period – that will give enough data for most caplet pricing illustrations.

However, for each simulated 'table' of forward rates one can observe that the same kind of simulated curve propagates into the future time. That raises an issue about importance of re-sampling when pricing by Monte Carlo.

[Insert 3D plots here]

## Random Numbers Generation (for faster convergence)

There are following RN generation methods that you can adopt from ready implementations (eg, NAG Library) to give professional quality to the Monte Carlo within your project. Low discrepancy generators provide statistically dependent numbers with improved evenness in the multidimensional space.

1. Mersenne Twister (Pseudo RN)
2. Halton Numbers (low discrepancy, Quasi RN)
3. Sobol Numbers (low discrepancy, Quasi RN)

Generally, the convergence provided by low latency RN generators is the order of $c(d)\frac{(\ln N)^d}{N}$ vs. $\frac{1}{\sqrt{N}}$ for quasi RN generators such as Excel's *RAND()* and Mersenne Twister. $N$ refers to the number of simulations, $d$ is number of dimensions (eg, $d = 3$ factors in HJM SDE and $d = 5$ reference names in Basket CDS), and $c(d)$ is some scaling function.

*Example for HJM implementation:* while Excel-generated random numbers might not give convergence even after $2,000$ simulations, Monte Carlo with proper low discrepancy RNs gives a good (low variance) estimate right after 200 simulations and satisfactory convergence after about 600 simulations. The second problem brought by this example is that **the use of Excel's pseudo random numbers over-estimated the price of a derivative.**

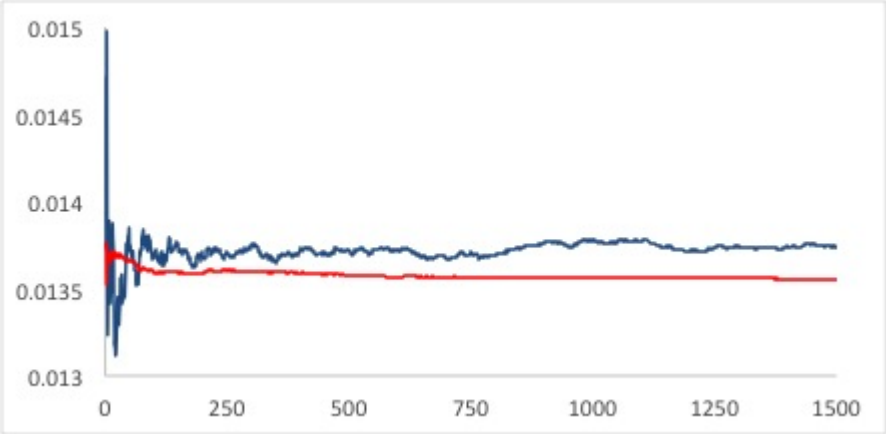


Figure 4: Dark blue line is a derivative price using *RAND()*, while red line result is for the same derivative obtained using low-latency RNs.

### Variance Reduction

Monte-Carlo simulation is evaluated by the speed (computational criterion) as well as variance of the estimator. That is, if you are estimating bond price or caplet price by Monte Carlo, you might would like to plot running standard deviation (after each 100 of simulations added).

The most important trick to reduce variance is to identify which simulated inputs increase the standard deviation of the estimate. For example in Basket CDS projects very small $u$ imply default times $\tau_i$ which are less than one quarter. Because each default time is a stand-alone output (inter-arrival times are conditionally independent) it possible to remove those early defaults without breaking the continuity of Monte-Carlo. On the other hand, you will not be able to do the same thing when simulating a log-normal asset price (i.e., removing price levels at will).

**Between Excel RAND() and Quasi RN, the variance of the estimator can vary threefold!**

**The note is an excerpt from the Q&A document on *Model Implementation and Robust Estimation* by Dr Richard Diamond, CQF.**