



A subsidiary of Sumitomo Mitsui Banking Corporation

## Pricing complex options using a simple Monte Carlo Simulation

**Peter Fink**

Among the different numerical procedures for valuing options, the Monte Carlo simulation is well suited to the construction of powerful yet simple pricing models. It is especially useful for single variable European options where, as a result of a non-standard pay-out, a closed-form pricing formula either does not exist or is difficult to derive. In addition, the price of complex options is sometimes difficult to explain intuitively and a simulation can often provide some insight into the factors that determine the pricing.

Once a basic pricing model has been constructed, it can be applied to different options simply by changing the pay-out function of the underlying variable to suit the particular case. There is no further need for complicated algebraic calculations.

This article examines the performance of an extremely simple Monte Carlo simulation spreadsheet which estimates the price of options using the same assumptions and basic methodology as the pricing models used by many banks. Although the presentation of detailed calculations and results is limited to LIBOR based non-standard options, the formalism is easily extended to other underlying variables such as other floating-rate indices, term swap rates and foreign exchange rates.

At the core of the simulation is an algebraic expression containing a stochastic term. This formula generates paths for the underlying variable (e.g. LIBOR, swap rate or foreign exchange rate) which are consistent with a given set of forward rates, volatilities and an assumed statistical distribution. The derivation of the generating formula is addressed in a number of texts but the following outline emphasizes the key points. The model assumes that an infinitesimal process for the underlying variable  $S$  is given by

$$dS = \mu S dt + \sigma S dz$$

$\mu$  is the expected proportional change in  $S$  per unit time,  $\sigma$  is the volatility of  $S$  and  $dz$  is the Wiener process  $dz = \varepsilon \sqrt{dt}$  (where  $\varepsilon \sim N(0,1)$ , a normal distribution of unit standard deviation and zero mean). Using Itô's lemma it is possible to derive the infinitesimal process for  $\ln(S)$ . The resulting algebraic expression is then in a form that can be readily extended to finite time intervals.

For a general time interval starting at time  $t$  and ending at time  $T$ , the generating formula for the simulated value of  $S$  at time  $T$  ( $S_T$ ) is

$$S_T = S_t e^{\left(m - \frac{1}{2}\sigma^2\right)(T-t) + \sigma \varepsilon \sqrt{(T-t)}}$$

At time T this formula generates a new value for S which depends, *inter alia*, on the value generated at time t.

The formula needs little alteration to bring in LIBOR as the specific underlying variable. The expected proportional change in LIBOR per unit time ( $\mu$ ) is given by the logarithm of the ratio of successive values of forward LIBOR. Then the formula becomes:

$$L_T = L_t \times \left( \frac{\text{Forward LIBOR}_T}{\text{Forward LIBOR}_t} \right) \times e^{\left( -\frac{1}{2} \sigma^2 \right) (T-t) + \sigma \epsilon \sqrt{(T-t)}}$$

(It assumes that values of implied forward LIBOR have been calculated.)

The pricing spreadsheet can now be assembled. A simple macro or procedure should be written so that the spreadsheet can automatically recalculate itself for any desired number of runs. In a simulation of LIBOR rates, the forward LIBOR curve and resulting discount factors must first be calculated. For example, if the term is 2 years and the reset frequency is 6-monthly, 4 discount factors and 4 values of 6 month LIBOR will be relevant to the pricing of transactions (For spot starting transactions there are only 3 relevant discount factors and values of LIBOR).

While it is recognized that this specification of the LIBOR process, using static discount factors, is not arbitrage-free, the results have proved accurate for a wide range of product types.

If the particular spreadsheet (e.g. Lotus 1-2-3 or Excel) does not provide a function for calculating random normal deviates but does provide a function which generates numbers from a uniform distribution over [0,1], the following approximation for a normal deviate  $\epsilon$  can be used

$$\epsilon \sim N(0,1) \approx \sum_{i=1}^{i=12} R_i - 6$$

To standardize the model so that a minimum number of changes is needed to price different options, it is useful to represent the option cash flows as though they were part of an interest rate swap. Another reason for this is that many of the options that we price at SBCM are typically embedded in structured notes and we price the swap which enables the issuer of the note to swap into a conventional floating rate liability. So each option is split into a LIBOR leg that is received and a constrained LIBOR leg that is paid, where the constraints reflect the desired optionality. The received LIBOR is the simulated LIBOR generated by the formula and the constrained LIBOR leg is the simulated LIBOR leg modified by the inclusion of IF, MAX or MIN statements as appropriate.

Hence, for each simulation run, values are assigned to both legs of the swap in every period. By calculating the net present value (NPV) of the difference between the swap legs using the per-period discount factors and repeating the exercise for several thousand simulation runs, a distribution of NPVs

can be constructed. The mean of this distribution is identified as the up-front premium (or the net premium when more than one embedded option is implied).

To illustrate this method, it is shown below how to estimate the price of two products:

**One-way (“sticky”) floating rate note.**

This is a floating rate note (FRN) whose coupon cannot fall below its previous setting or rise by more than a pre-defined amount between consecutive periods. A buyer of such a note has bought a floor and sold a cap. The floor rate is equal to the immediately preceding coupon setting. The cap rate is equal to the sum of the immediately preceding coupon setting and the maximum per-period increment. Clearly, the cap and floor rates will vary with the actual coupon settings that occur during the life of the note. Hence the options are said to be path dependent.

The example chosen is a 2 year DEM 6 month LIBOR one-way FRN with a maximum per-period increment of 0.25%. The coupon is 6 month DEM LIBOR flat and each successive coupon setting is never less than the previous setting. However, each successive coupon setting cannot exceed the previous coupon by more than 0.25%. The caps and floors implicit in this structure can be priced using the simulation to give an expected up-front premium.

This type of structure often requires that the price be given in terms of the coupon’s margin over LIBOR. This is easily calculated from the up-front premium by dividing it by the sum of the discount factors [(see table)] and multiplying the result by the coupon frequency (i.e. for semi-annual coupons, multiply by 2).

Semi-annual Period	Discount Factor	Spot and Forward 6 mth			Receive	Pay	PV of cash flow
		DEM LIBOR	Volatility	N(0,1)	Unconstrained LIBOR L1	Constrained LIBOR L2	
0	1.0000	4.490%	20.00%	(0.433)			
1	0.9773	4.615%	20.00%	(1.041)	4.49%	4.49%	0.00%
2	0.9550	5.254%	20.00%	1.462	3.94%	4.49%	-0.26%
3	0.9303	5.696%	20.00%	0.696	5.47%	4.74%	0.34%
4	0.9044	6.271%	20.00%	0.478	6.47%	4.99%	0.68%
						NPV	0.76%

$$L2_0 = L1_0$$

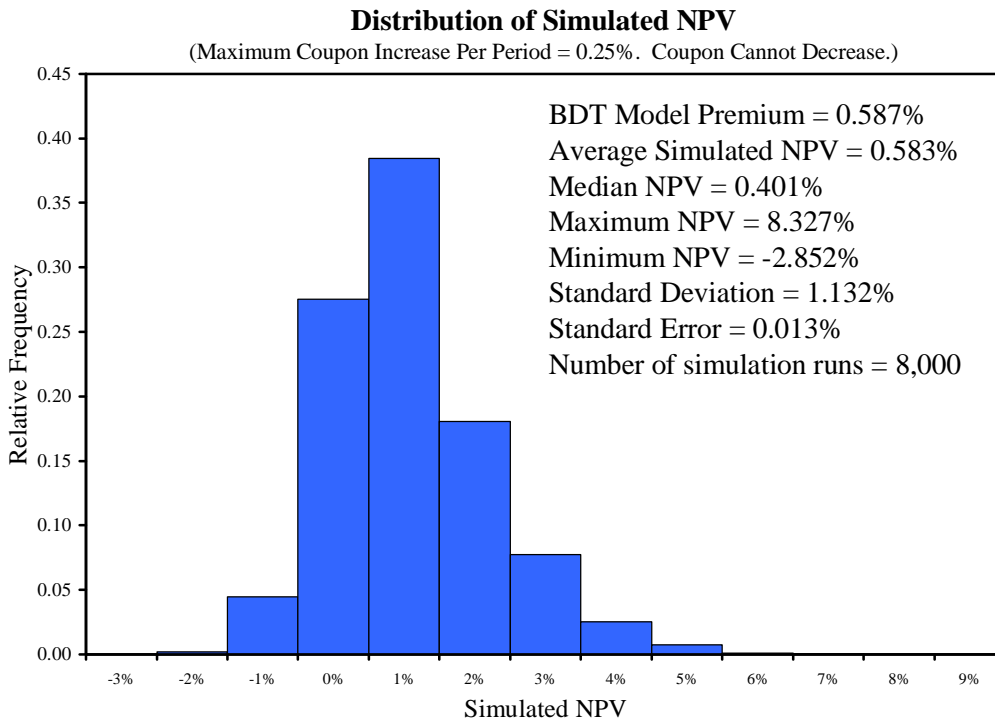
$$L2_j = \text{MIN} \left( \text{MAX} (L1_j, L2_{j-1}), L2_{j-1} + 0.25\% \right) \text{ for } j = 1, 2 \text{ or } 3$$

(Where  $L2_j$  is the constrained LIBOR coupon set at time  $j$  and  $L1_j$  is the unconstrained LIBOR also set at the beginning of period  $j$ )

$$\text{Equivalent semi - annual margin over LIBOR} = \frac{2 \times \text{NPV} \times \left( \frac{360}{365} \right)}{\sum_{t=1}^{t=4} \text{Discount Factor at Period } t}$$

A Black, Derman and Toy (BDT) model using a single volatility of 20% gives an up-front premium value

of 0.587%. Figure 1 shows the distribution of NPVs given by the Monte Carlo simulation after 8,000 runs



### Single date knock-out cap.

The knock-out cap is designed to reduce the up-front premium payable by the option buyer. The knock-out rate is higher than the cap rate and whenever LIBOR sets below the knock-out rate, a knock-out cap is indistinguishable from a conventional cap. When LIBOR sets above the knock-out rate, the cap ceases to operate.

Two features of a knock-out structure must be distinguished. First, the knock-out condition can apply on one or several dates. Second, if a knock-out occurs, it can either be for that period only or it can apply to all of the remaining caplets. From a pricing point of view, the simplest case is a per-period knock-out where the condition is applied in each period but only to the caplet for that period. In this case the buyer of such a cap has effectively bought a regular cap at the cap rate and has also sold a regular cap and a binary cap, both at the knock-out rate. The binary pay-out is equal to the difference between the knock-out rate and the underlying cap rate.

The example chosen here is more difficult to price. It is a 3 year GBP 6 month LIBOR 8% cap with a knock-out rate of 10% applicable after 1 year. While the knock-out condition is applied on only one date, if the condition is satisfied, all of the remaining caplets vanish. This type of knock-out cap is not equivalent to a combination of conventional and binary caps. As far as the simulation goes, it is not difficult to adapt the model. In each run a flag is set up to monitor the level of LIBOR on the knock-out date. Whenever the knock-out rate is breached, the constrained LIBOR leg of the swap in all subsequent periods is set equal to the unconstrained LIBOR leg, i.e. all remaining caplets vanish. The simulation is run as before. Using the same notation as in the previous example, the constraint equation is given by:

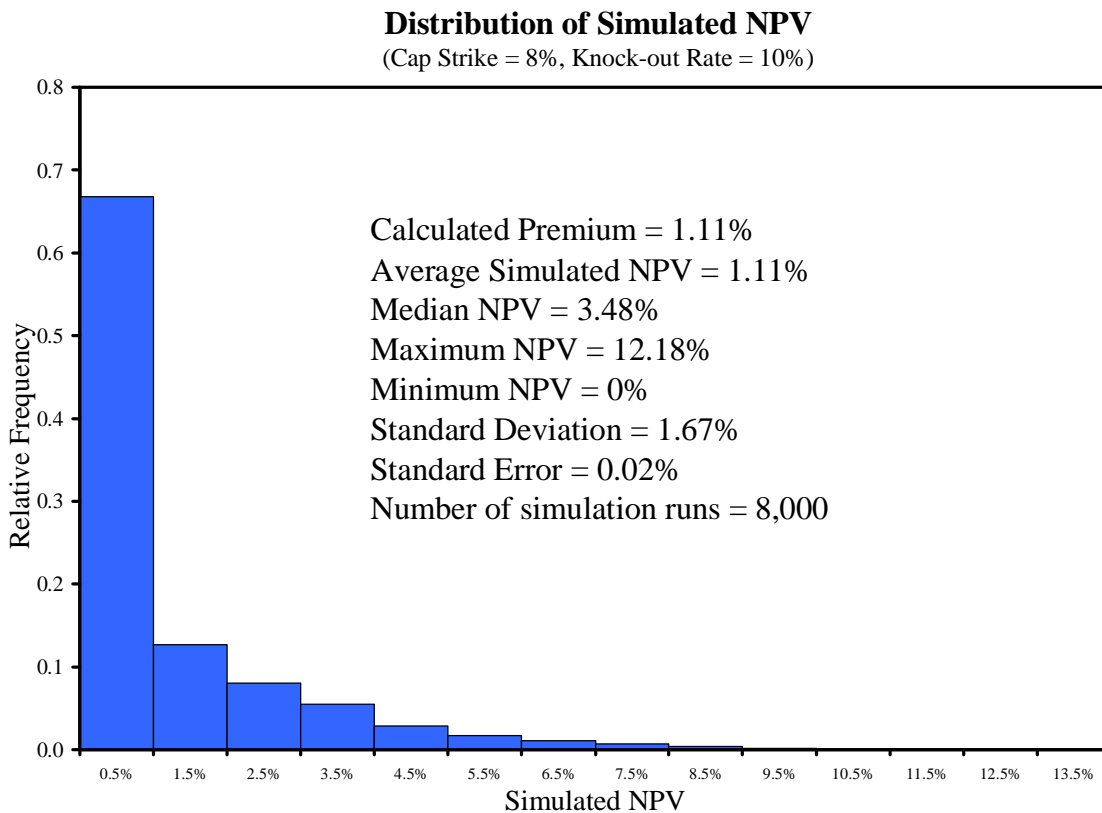
$$L2_0 = L1_0$$

$$L2_1 = \text{MIN}(L1_1, 0.08)$$

$$\text{FLAG: IF}(L1_2 > 0.1, 0, 1)$$

$$L2_j = \text{IF}(\text{FLAG} = 0, L1_j, \text{MIN}(L1_j, 0.08)) \text{ for } j = 2, 3, 4 \text{ or } 5$$

A volatility of 20% is assumed. A closed-form solution pricing model using a single volatility of 20% gives an up-front premium of 1.11%. The resulting distribution of the NPV from the simulation is given in figure 2.



The simulation model is easily changed to price both a multi-date knock-out where all of the remaining caplets disappear if the knock-out level is breached and a per-period knock-out. The premia are 0.43% and 0.49% respectively.

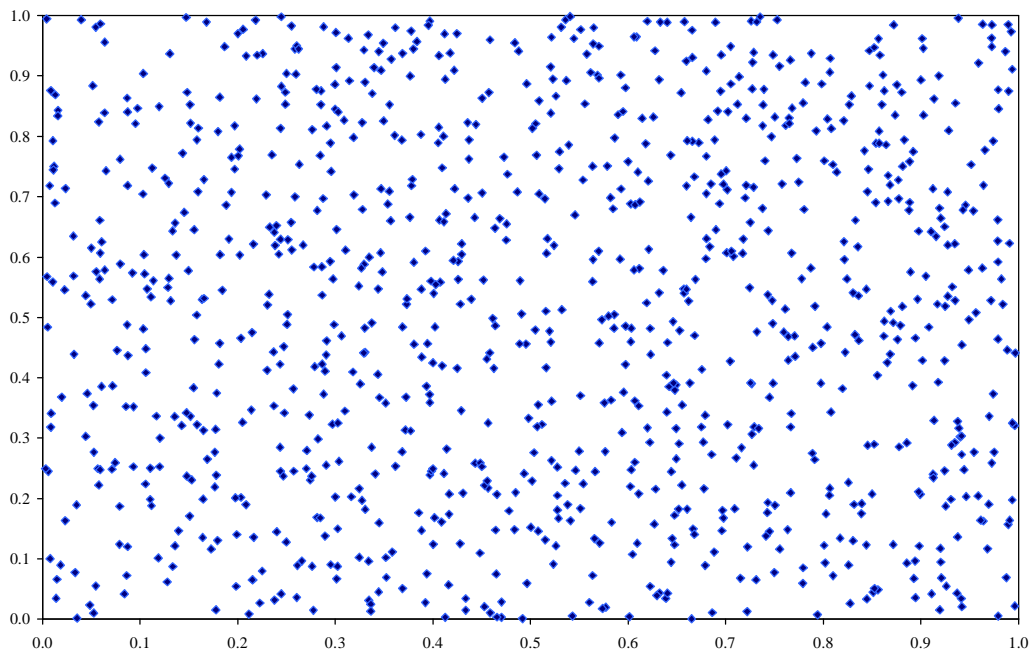
The sensitivity of the option price to changes in the level of interest rates and volatilities can be calculated from the simple model. Essentially 2 parallel simulations are run. For example, the vega of an option is estimated by running a simulation at the original volatility and simultaneously running a simulation at a volatility 1% higher but using the same set of normal deviates. Similarly, the delta would be estimated by adjusting the forward LIBOR rates upward by 1 basis point and running 2 parallel simulations for the original and shifted forward curves. Using the simulation for a conventional 3 year GBP cap at 10% gives a delta of 0.41 and a vega of 5.3 (a Black model calculation gives 0.41 and 5.6). For very small values of vega, of the order of 1 basis point, the simple model used here may be inaccurate owing to the sampling error.

While the aim of this article has been to present as simple a model as possible, the accuracy of the model can be improved without any major increase in complexity. Since the estimate of the option value is a sample mean from an underlying distribution, some variation will be observed between sets of simulations. The standard deviation of the sample means (the standard error) scales inversely with the square root of the number of runs, i.e. quadrupling the number of runs will only double the accuracy but will proportionately increase the running time. A better method for reducing the standard error is to use antithetic variables. This relies on the fact that any finite normal deviate  $\epsilon$  has the same probability of being drawn from a  $N(0,1)$  distribution as  $-\epsilon$ .

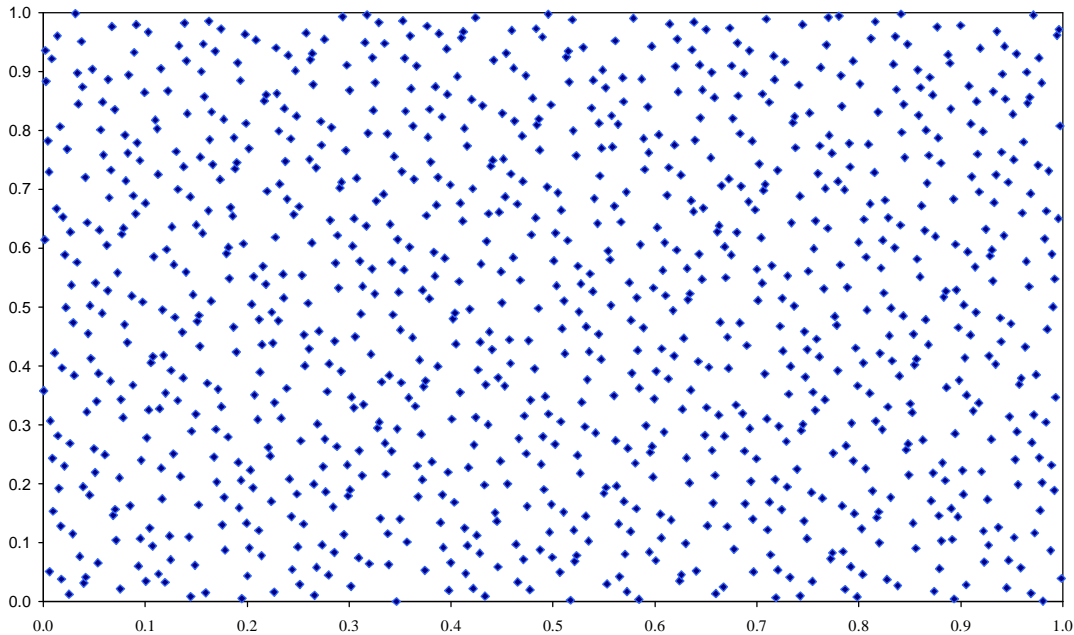
The method then requires that two parallel simulations are run for normal deviates  $\pm\epsilon$ . This produces 2 distributions of NPVs and the arithmetic mean of the means of the distributions should then be a more reliable estimate of the option value.

Another source of inaccuracy is the sampling error arising from the type of random number generator used in the simulation. The simple model presented here uses a crude approximation to a normal distribution derived from a uniform random number generator. If this particular generator (the @RAND function in Lotus 1-2-3 and the RAND function in Excel) is used to generate a series of (x,y) points on a graph (figure 3) the resulting non-uniformity is clearly evident when compared with a similar plot produced using a Faure sequence (figure 4).

**Figure 3 - Plot of Excel RAND function number pairs (x,y)**



**Figure 4 - Plot of Faure sequence number pairs (x,y)**



A number of articles in financial publications demonstrated the greater accuracy of Faure and Sobol sequences over the more conventional methods used for low dimension simulations, (e.g. RISK June 1996, pages 63-65). However, such methods would increase the complexity of the model here, against the aim of this article to present the simplest possible model which reproduces the results of larger and more complicated calculations. **n**

*Peter Fink is Senior Vice President and Head of Marketing at SMBC Capital Markets, Inc., New York  
e-mail: [fink@smbc-cm.com](mailto:fink@smbc-cm.com)*