CHAPTER 10
Monte Carlo Simulation

10.1 INTRODUCTION

Monte Carlo simulation is among the most important numerical algorithms of the 20th century (cf. Cipra (2000)) and obviously will remain so in the 21st century as well. Its importance for financial applications stems from the fact that it is most flexible in terms of financial products that can be valued. First applied to European option pricing in 1977 by Phelim Boyle (cf. Boyle (1977)), it took until the 21st century for the problem of valuing American options by Monte Carlo simulation to be satisfactorily solved by Francis Longstaff and Eduardo Schwartz (cf. Longstaff and Schwartz (2001)) and others (cf. Chapter 7). Glasserman (2004) provides a comprehensive introduction to Monte Carlo methods for financial engineering and is a standard reference. Kohler (2009) is a survey article of regression-based valuation approaches for American options.

Although quite flexible, Monte Carlo simulation is generally not very fast (relative to alternative approaches) since millions of computations are necessary to value a single option. Consider a simulation run with 100 time intervals (=100 exercise dates) and 100,000 paths for an American put option on a single stock with constant volatility and constant short rate. You need 10 million random numbers, several arrays (i.e. matrices) of size 101 times 100,000 and you have to estimate 100 least-squares regressions over 100,000 pairs of numbers as well as discounting 100 times 100,000 numbers. If you enrich the financial model to include, for example, stochastic volatility and stochastic interest rates the number of necessary calculations further increases substantially.

For practical purposes, it is important to have available efficient, i.e. accurate and fast, algorithms to value options and other derivatives by Monte Carlo simulation. This chapter therefore analyzes in detail the simulation of financial models of type (9.1)–(9.3) as presented in the previous chapter. The simulation of equation (9.1) turns out be straightforward since an exact discretization is easily found. However, this is not the case for the two square-root diffusions (9.2) and (9.3).

The chapter proceeds as follows. Section 10.2 values zero-coupon bonds in the CIR85 model by Monte Carlo simulation. Here, we only need to consider a single square-root diffusion. Section 10.3 values European options by Monte Carlo simulation in the H93 stochastic volatility model with constant short rate and without jumps. Section 10.4 then adds stochastic short rates of CIR85 type to the H93 setting to value American put options by Monte Carlo simulation. Section 10.5 summarizes the major findings.

10.2 VALUATION OF ZERO-COUPON BONDS

In this section, we consider the stochastic short rate model $\mathcal{M}^{CIR85}$ of Cox-Ingersoll-Ross (cf. Cox et al. (1985)) which is given by the SDE (9.3). We repeat the SDE for convenience:

$$dr_t = \kappa_r(\theta_r - r_t)dt + \sigma_r \sqrt{r_t} dZ_t$$

To simulate the short rate model, it has to be discretized. To this end, we again divide the given time interval $[0, T]$ in equidistant sub-intervals of length Δt such that now $t \in \{0, \Delta t, 2\Delta t, ..., T\}$, i.e. there are $M + 1$ points in time with $M \equiv T/\Delta t$.

The exact transition law of the square-root diffusion is known. The article by Broadie and Kaya (2006) provides an in-depth analysis of this topic. Consider the general square-root diffusion process

$$dx_t = \kappa(\theta - x_t)dt + \sigma \sqrt{x_t} dZ_t \tag{10.1}$$

In Broadie and Kaya (2006) it is shown that x_t, given x_s with $s = t - \Delta t$, is distributed according to

$$x_t = \frac{\sigma^2 \left(1 - e^{-\kappa \Delta t}\right)}{4\kappa} \chi_d'^2 \left(\frac{4\kappa e^{-\kappa \Delta t}}{\sigma^2 (1 - e^{-\kappa \Delta t})} x_s \right)$$

where $\chi_d'^2$ denotes a non-central chi-squared distributed random variable with

$$d = \frac{4\theta\kappa}{\sigma^2}$$

degrees of freedom and non-centrality parameter

$$l = \frac{4\kappa e^{-\kappa \Delta t}}{\sigma^2 (1 - e^{-\kappa \Delta t})} x_s$$

For implementation purposes, it may be convenient to sample a chi-squared distributed random variable χ_d^2 instead of a non-central chi-squared one, $\chi_d'^2$. If $d > 1$, the following relationship holds true

$$\chi_d'^2(l) = \left(z + \sqrt{l}\right)^2 + \chi_{d-1}^2$$

where z is an independent standard normally distributed random variable. Similarly, if $d \leq 1$, one has

$$\chi_d'^2(l) = \chi_{d+2N}^2$$

where N is now a Poisson-distributed random variable with intensity $l/2$. For an algorithmic representation of this simulation scheme refer to Glasserman (2004), p. 124.

The exactness comes with a relatively high computational burden which may, however, be justified by higher accuracy due to faster convergence. In other words, although the computational burden per simulated value of x_t may be relatively high with the exact scheme, the possible reduction in necessary time steps and simulation paths may more than compensate for this. However, Andersen, Jäckel and Kahl argue in Andersen et al. (2010)—referring to the exact simulation approach of Broadie and Kaya (2006)—that

> "One might think that the existence of an exact simulation-scheme … would settle once and for all the question of how to generate paths of the square-root process…., it seems [nevertheless] reasonable to also investigate the application of simpler simulation algorithms. These will typically exhibit a bias for finite values of [M], but convenience and speed may more than compensate for this, …"

We therefore also consider a Euler discretization of the square-root diffusion (10.1). A possible discretization is given by

$$\tilde{x}_t = \tilde{x}_s + \kappa(\theta - \max[0, \tilde{x}_s])\Delta t + \sigma\sqrt{\max[0, \tilde{x}_s]}\sqrt{\Delta t} z_t \tag{10.2}$$

$$x_t = \max[0, \tilde{x}_s] \tag{10.3}$$

with z_t standard normal (this scheme is usually called *Full Truncation*, see below). While x_t cannot reach zero with the exact scheme if the Feller condition $2\kappa\theta > \sigma^2$ is met, this is not the case with the Euler scheme. Therefore, the maximum function is applied several times.[1]

The plan now is as follows. We simulate the CIR85 model and derive Monte Carlo simulation estimates for Zero-Coupon Bond (ZCB) values at different points in time. Since we know these values in closed form in the CIR85 model, we have a natural benchmark to check the accuracy of the Monte Carlo simulation implementation. Chapter 9 presents the respective formula for the present value of the ZCB, i.e. for $t = 0$. Sub-section 10.6.1 contains a Python implementation which allows us to freely choose $0 < t \leq T$. Two adjustments are made:

1. The final date T is replaced by time-to-maturity $T - t$
2. The initial rate r_0 is replaced by the expectation

$$\mathbf{E}(r_t) = \theta_r + e^{-\kappa_r t}(r_0 - \theta_r)$$

Figure 10.1 shows 20 simulated paths for the short rate process of CIR85 and for the example parameters of the Python script of sub-section 10.6.2. A Monte Carlo simulation estimator for the value of the ZCB at t is derived as follows. Consider a certain path i of the I simulated paths for the short rate process with time grid $t \in \{0, \Delta t, 2\Delta t, ..., T\}$. We discount

[1]There are number of alternative Euler schemes available which section 10.3 presents and compares with regard to their performance, i.e. accuracy and speed.

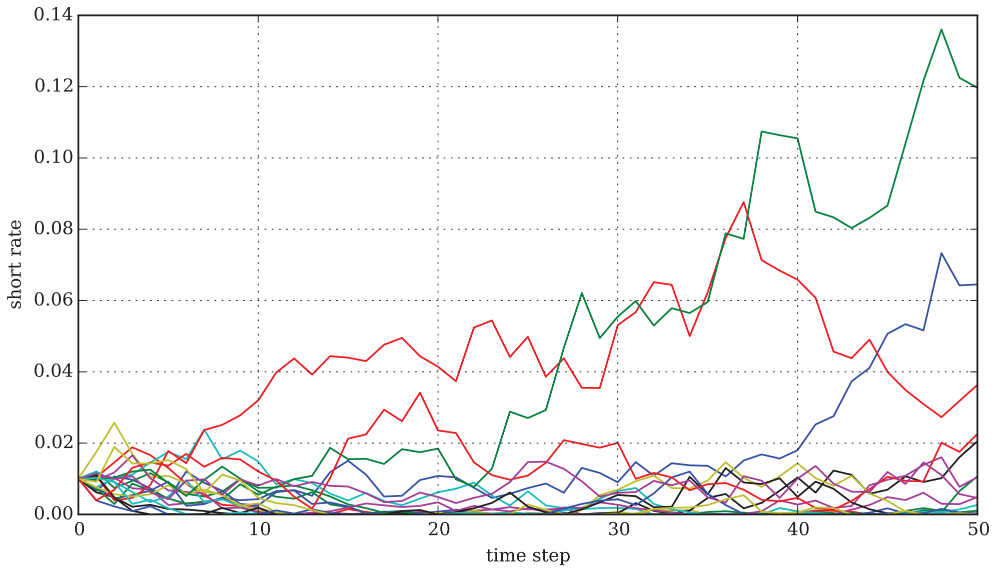

FIGURE 10.1 Twenty simulated paths for the CIR85 short rate process

the terminal value of the ZCB, i.e. 1.0, step by step backwards. For $t < T$ and $s \equiv t - \Delta t$ we have

$$B_{s,i} = B_{t,i} e^{-\frac{r_t + r_s}{2}\Delta t}$$

The Monte Carlo simulation estimator of the ZCB value at t then is

$$B_t^{MCS} = \frac{1}{I}\sum_{i=1}^{I} B_{t,i}$$

Figures 10.2 and 10.3 present valuation results for both the exact scheme and the Euler scheme compared, respectively, to the analytical values for a ZCB maturing at $T = 2$. The figures illustrate that with $M = 50$ time steps and $I = 50{,}000$ paths both schemes deliver Monte Carlo simulation estimates really close to the analytical values. However, the Euler scheme shows a systematically low bias in this particular case. The errors for the exact scheme are not only smaller but also negative as well as positive.

In terms of speed, the Euler scheme is indeed much faster. The generation of $I = 50{,}000$ sample paths with $M = 50$ time steps takes only about one-quarter of the time with the Euler scheme compared to the exact scheme. As a consequence, one could, for example, double the number of time steps to $M = 100$ to increase accuracy of the Euler scheme.

These numbers and comparisons are illustrative only. They are by no means a “proof” that the exact scheme easily outperforms a Euler scheme. The subsequent section revisits this issue in the context of the stochastic volatility model of H93—in this case we need to correlate the square-root diffusion with a second process which introduces a new problem area.

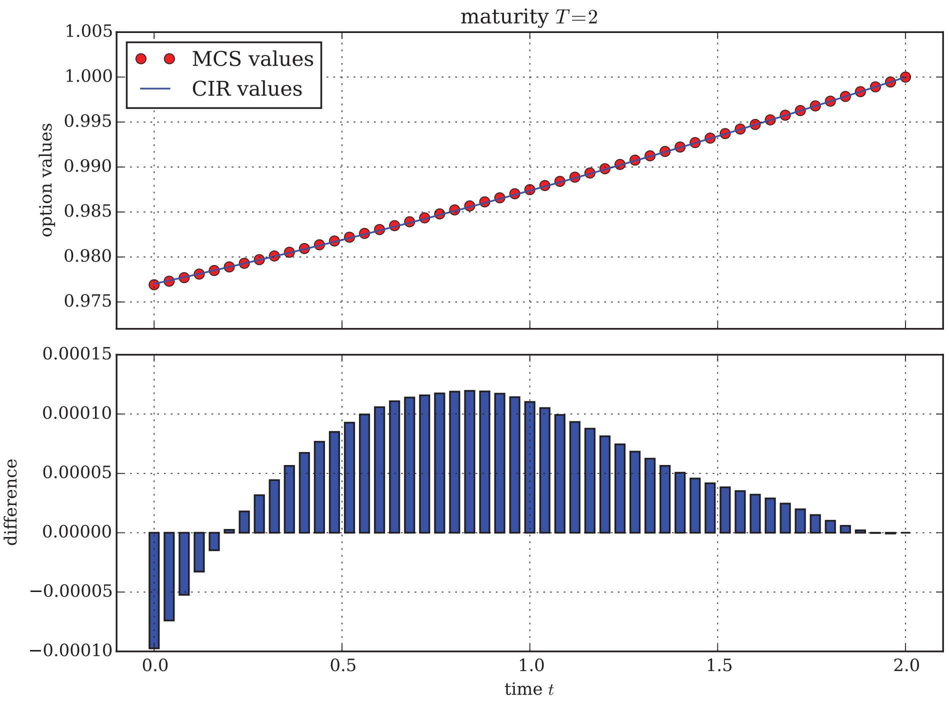

FIGURE 10.2 Values for a ZCB maturing at $T = 2$; line = analytical values, dots = Monte Carlo simulation estimates from the exact scheme for $M = 50$ and $I = 50{,}000$

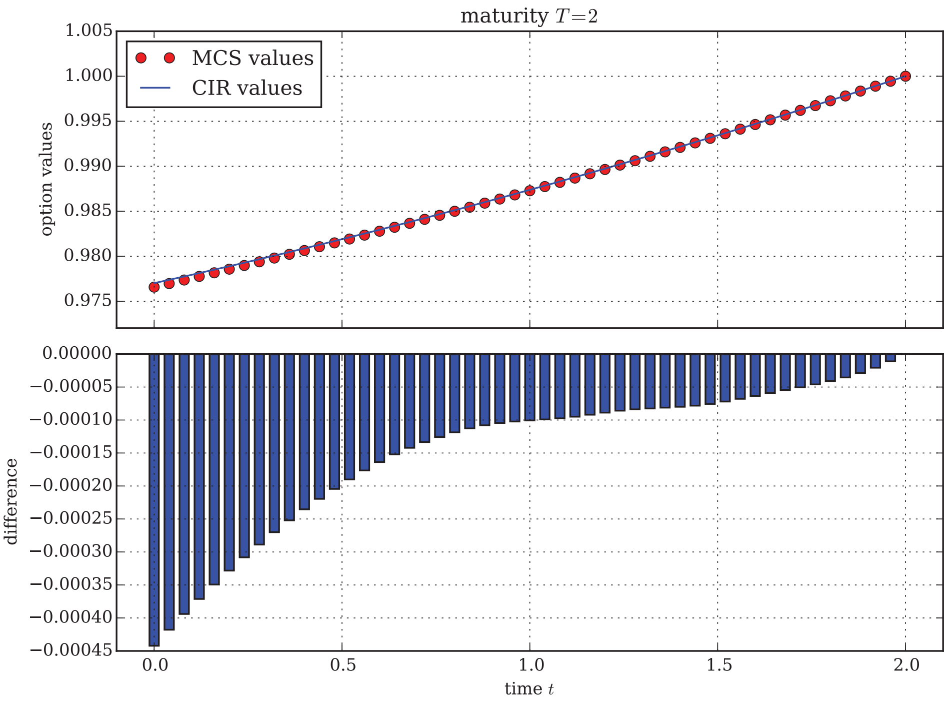

FIGURE 10.3 Values for a ZCB maturing at $T = 2$; line = analytical values, dots = Monte Carlo simulation estimates from the Euler scheme for $M = 50$ and $I = 50{,}000$

10.3 VALUATION OF EUROPEAN OPTIONS

As the next special case of the general framework $\mathcal{M}^{BCC97}$ with risk-neutral dynamics given by (9.1)–(9.3), we consider the H93 stochastic volatility model $\mathcal{M}^{H93}$ with constant short rate. This section values European call and put options in this model by Monte Carlo simulation. As for the ZCB values, we also have available a (semi-analytical) pricing formula which generates natural benchmark values against which to compare the Monte Carlo simulation estimates.

For $0 \leq t \leq T$, the risk-neutral dynamics of the index in the H93 stochastic volatility model are given by

$$dS_t = rS_t dt + \sqrt{v_t} S_t dZ_t^1 \tag{10.4}$$

with the variance following the square-root diffusion

$$dv_t = \kappa_v(\theta_v - v_t)dt + \sigma_v \sqrt{v_t} dZ_t^2 \tag{10.5}$$

The two Brownian motions are instantaneously correlated with $dZ_t^1 dZ_t^2 = \rho$. This correlation introduces a new problem dimension into the discretization for simulation purposes (cf. Broadie and Kaya (2006)). To avoid problems arising from correlating normally distributed increments (of S) with chi-squared distributed increments (of v), we consider in the following only Euler schemes for both the S and v processes. This has the advantage that the increments of v become normally distributed as well and can therefore be easily correlated with the increments of S.

In total, we consider *two* discretization schemes for S and seven discretization schemes for v. For S, we consider the exact **log** Euler scheme with

$$S_t = S_s e^{(r-v_t/2)\Delta t + \sqrt{v_t}\sqrt{\Delta t} z_t^1} \tag{10.6}$$

where $s \equiv t - \Delta t$ and z_t^1 standard normal. This scheme is obtained by considering the dynamics of $\log S_t$ and applying Itô's lemma to it. For illustration purposes, we also consider the **naive** Euler discretization (with $s \equiv t - \Delta t$)

$$S_t = S_s \left(e^{r\Delta t} + \sqrt{v_t}\sqrt{\Delta t} z_t^1 \right) \tag{10.7}$$

These schemes can be combined with any of the following Euler schemes for the square-root diffusion.[2]

- **Full Truncation**

$$\begin{aligned} \tilde{x}_t &= \tilde{x}_s + \kappa(\theta - \tilde{x}_s^+)\Delta t + \sigma\sqrt{\tilde{x}_s^+}\sqrt{\Delta t} z_t \\ x_t &= \tilde{x}_t^+ \end{aligned}$$

[2]In the following, x^+ is notation for $\max[x, 0]$.

- **Partial Truncation**

$$\begin{aligned}\tilde{x}_t &= \tilde{x}_s + \kappa(\theta - \tilde{x}_s)\Delta t + \sigma\sqrt{\tilde{x}_s^+}\sqrt{\Delta t} z_t \\ x_t &= \tilde{x}_t^+\end{aligned}$$

- **Truncation**

$$x_t = \max[0, x_s + \kappa(\theta - x_s)\Delta t + \sigma\sqrt{x_s}\sqrt{\Delta t} z_t]$$

- **Reflection**

$$\begin{aligned}\tilde{x}_t &= |\tilde{x}_s| + \kappa(\theta - |\tilde{x}_s|)\Delta t + \sigma\sqrt{|\tilde{x}_s|}\sqrt{\Delta t} z_t \\ x_t &= |\tilde{x}_t|\end{aligned}$$

- **Higham-Mao**

$$\begin{aligned}\tilde{x}_t &= \tilde{x}_s + \kappa(\theta - \tilde{x}_s)\Delta t + \sigma\sqrt{|\tilde{x}_s|}\sqrt{\Delta t} z_t \\ x_t &= |\tilde{x}_t|\end{aligned}$$

- **Simple Reflection**

$$x_t = \left|x_s + \kappa(\theta - x_s)\Delta t + \sigma\sqrt{x_s}\sqrt{\Delta t} z_t\right|$$

- **Absorption**

$$\begin{aligned}\tilde{x}_t &= \tilde{x}_s^+ + \kappa(\theta - \tilde{x}_s^+)\Delta t + \sigma\sqrt{\tilde{x}_s^+}\sqrt{\Delta t} z_t \\ x_t &= \tilde{x}_t^+\end{aligned}$$

This list contains only Euler schemes and is not exhaustive with regard to discretization schemes for the square-root diffusion (cf. Andersen et al. (2010), Andersen (2008), Broadie and Kaya (2006), Glasserman (2004) and Haastrecht and Pelsser (2010)). However, all these schemes share the convenient feature that correlation of the variance square-root diffusion with the index process is easily accomplished.

In the literature, there are a lot of tests and numerical studies available that compare efficiency of different discretization schemes. But since the approach of this book is a practical one, we want to implement our own test and comparison procedures. Moreover, we want to use Python in combination with the data management and analysis library pandas to automate our tests.

For our tests, we take four different parametrizations for the H93 model as found in Medvedev and Scaillet (2010), table 3. In these four model economies, we value the following:

- **options**: European call and put options
- **maturities**: we take $T \in \{\frac{1}{12}, 1, 2\}$
- **strikes**: we take $K \in \{90, 100, 110\}$ for $S_0 = 100$
- **time steps**: we take $M \in \{25, 50\}$ steps per year

- **paths**: we take $I \in \{25{,}000, 50{,}000, 75{,}000, 100{,}000\}$
- **discretization**: we combine all schemes (two for index with seven for variance = 14 schemes)

This makes a total of 36 option values per option type. In view of the empirical results about option spreads and tick sizes, as presented in section 3.5, we say that a valuation is accurate if

1. the absolute value difference is smaller than **2.5 cents or**
2. the absolute value difference is smaller than **1.5%**

To improve upon valuation accuracy, we use both moment matching and antithetic paths for our Python implementation found in sub-section 10.6.3. This script writes all results into a pandas DataFrame object and saves this in HDF5 format to disk (e.g. for later usage and analysis).

To generate antithetic paths (cf. Glasserman (2004), sec. 4.2), we use both the pseudo-random number $z_{t,i}$ and its negative value $-z_{t,i}$ (where we generate only $I/2$ random numbers). With regard to moment matching (cf. Glasserman (2004), sec. 4.5.), we correct the first two moments of the pseudo-random numbers delivered by Python. The respective code for both antithetic paths and moment matching looks like this:

```
def random_number_generator(M, I):
    if antipath:
        rand = np.random.standard_normal((2, M + 1, I / 2))
        rand = np.concatenate((rand, -rand), 2)
    else:
        rand = np.random.standard_normal((2, M + 1, I))
    if momatch:
        rand = rand / np.std(rand)
        rand = rand - np.mean(rand)
    return rand
```

Depending on the time interval Δt used, the drift of the index level process may also show a non-negligible bias (even after correction of the random numbers). We can correct the first moment of the index level process in a fashion similar to the pseudo-random numbers:

```
for t in range(1, M + 1, 1):
        ran = np.dot(CM, rand[:, t])
        if momatch:
            bias = np.mean(np.sqrt(v[t]) * ran[row] * sdt)
        if s_disc == 'Log':
            S[t] = S[t - 1] * np.exp((r - 0.5 * v[t]) * dt +
                    np.sqrt(v[t]) * ran[row] * sdt - bias)
        elif s_disc == 'Naive':
            S[t] = S[t - 1] * (math.exp(r * dt) +
                    np.sqrt(v[t]) * ran[row] * sdt - bias)
```

TABLE 10.1 Valuation results for European call and put options in H93 model for parametrizations from Medvedev and Scaillet (2010) and $M_0 = 50, I = 100{,}000$. Performance yardsticks are $PY_1 = 0.025$ and $PY_1 = 0.015$.[a]

OT	*R*	*I*	ID	XD	MM	AP	#ER	#OP	AE	MSE
CALL	5	100,000	L	A	False	False	146	180	0.07670	3.74152
CALL	5	100,000	L	A	False	True	146	180	0.07288	3.75574
CALL	5	100,000	L	A	True	False	3	180	0.00633	0.00136
CALL	5	100,000	L	A	True	True	3	180	0.00468	0.00149
CALL	5	100,000	L	F	False	False	146	180	0.03556	3.63164
CALL	5	100,000	L	F	False	True	148	180	0.03462	3.62426
CALL	5	100,000	L	F	True	False	1	180	−0.01659	0.00113
CALL	5	100,000	L	F	True	True	1	180	−0.01299	0.00090
CALL	5	100,000	L	P	False	False	144	180	0.03942	3.68678
CALL	5	100,000	L	P	False	True	145	180	0.04079	3.64441
CALL	5	100,000	L	P	True	False	1	180	−0.01474	0.00108
CALL	5	100,000	L	P	True	True	1	180	−0.01128	0.00111
CALL	5	100,000	L	T	False	False	147	180	0.07196	3.72847
CALL	5	100,000	L	T	False	True	145	180	0.07256	3.74803
CALL	5	100,000	L	T	True	False	3	180	0.00340	0.00155
CALL	5	100,000	L	T	True	True	3	180	0.01147	0.00162
PUT	5	100,000	L	A	False	False	143	180	0.04343	0.93155
PUT	5	100,000	L	A	False	True	141	180	0.04284	0.93065
PUT	5	100,000	L	A	True	False	14	180	0.00445	0.00110
PUT	5	100,000	L	A	True	True	20	180	0.00657	0.00149
PUT	5	100,000	L	F	False	False	141	180	0.03198	0.94487
PUT	5	100,000	L	F	False	True	142	180	0.03797	0.94874
PUT	5	100,000	L	F	True	False	9	180	−0.01349	0.00068
PUT	5	100,000	L	F	True	True	10	180	−0.01379	0.00083
PUT	5	100,000	L	P	False	False	143	180	0.03593	0.96873
PUT	5	100,000	L	P	False	True	141	180	0.03330	0.94941
PUT	5	100,000	L	P	True	False	3	180	−0.00881	0.00041
PUT	5	100,000	L	P	True	True	5	180	−0.00987	0.00056
PUT	5	100,000	L	T	False	False	143	180	0.04830	0.92206
PUT	5	100,000	L	T	False	True	142	180	0.04231	0.92051
PUT	5	100,000	L	T	True	False	10	180	0.00687	0.00111
PUT	5	100,000	L	T	True	True	10	180	0.00445	0.00117

[a]Monte Carlo simulation values benchmarked against semi-analytical values from Fourier-based pricing approach. The columns report the following values: **OT** = the option type (call or put), ***R*** = number of valuation runs, ***I*** = number of paths per single option valuation, **ID** = (first letter of) discretization scheme for index process, **XD** = (first letter of) discretization scheme for variance process, **MM** = moment matching, **AP** = antithetic paths, **#ER** = number of errors out of #OP, **#OP** = number of options valued, **AE** = average error over all valuations, **MSE** = mean squared error of all valuations.

Table 10.1 reports valuation results for European call and put options with $M_0 = 50$ and $I = 100{,}000$. Here, M_0 means steps per year. For example, if time-to-maturity is 2 years, we set $M = 2 \cdot M_0 = 100$. The table uses the exact scheme for the index process and combines this with four different schemes for the variance process (Full Truncation, Partial Truncation, Truncation and Absorption). It is evident that variance reduction techniques are indispensable

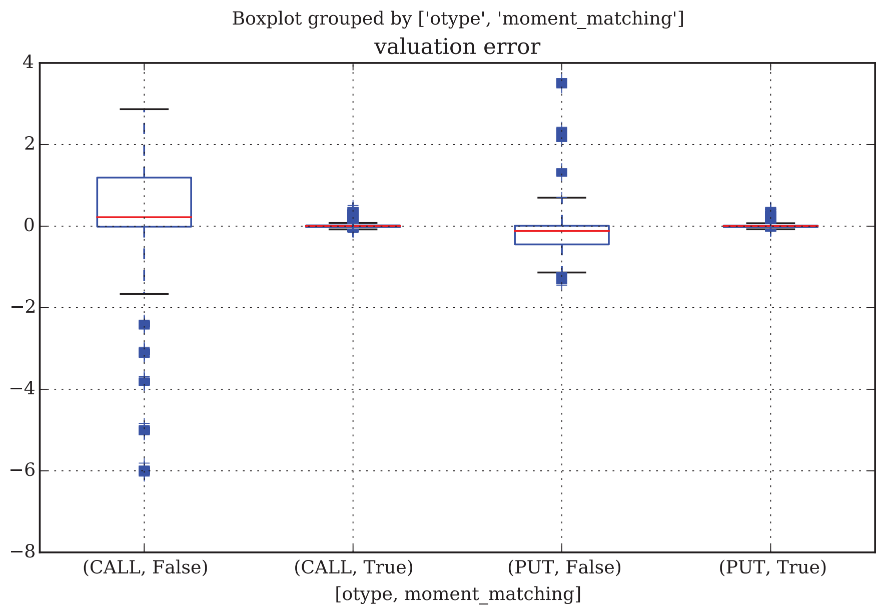

FIGURE 10.4 Boxplot of Monte Carlo valuation errors without and with moment matching

in this setting. While antithetic paths have no noticeable influence ceteris paribus, moment matching significantly increases accuracy of the valuation process. Using moment matching (with or without antithetic paths) the Full Truncation scheme shows the best overall performance. However, for some configurations the Absorption scheme, for example, also performs quite well. The average absolute valuation errors for the total set of 180 options when using moment matching mainly range between 0.5 cents and 1 cent. The good performance of the Full Truncation scheme is in line with results obtained in Lord et al. (2006) who name this particular scheme as their winner.

Figure 10.4 illustrates the importance of moment matching techniques in this context. The correction effect moment matching has for Monte Carlo valuation is impressively illustrated in that figure.

Table 10.2 presents further valuation results. This time all possible combinations of the discretization schemes are considered. Again, the Full Truncation scheme generates good valuation results—and, which may come as a surprise, the naive discretization scheme for the index process also performs relatively well throughout.

10.4 VALUATION OF AMERICAN OPTIONS

We now turn to American (put) options which are a little bit harder to value efficiently, i.e. accurately and fast, by Monte Carlo simulation.[3] The setup for this section is the H93 stochastic

[3]This section is mainly based on a 2011 working paper by the author titled “Fast Monte Carlo Valuation of American Options under Stochastic Volatility and Interest Rates.” The results of the paper were presented at the EuroScipy 2011 conference in Paris.

TABLE 10.2 Valuation results for European call and put options in H93 model for parametrizations from Medvedev and Scaillet (2010) and $M_0 = 50, I = 100{,}000$. Performance yardsticks are $PY_1 = 0.025$ and $PY_1 = 0.015$.[a]

OT	*R*	*I*	ID	XD	MM	AP	#ER	#OP	AE	MSE
CALL	5	100,000	L	A	True	True	3	180	0.00468	0.00149
CALL	5	100,000	L	F	True	True	1	180	−0.01299	0.00090
CALL	5	100,000	L	H	True	True	4	180	0.01169	0.00175
CALL	5	100,000	L	P	True	True	1	180	−0.01128	0.00111
CALL	5	100,000	L	R	True	True	5	180	0.01208	0.00253
CALL	5	100,000	L	S	True	True	13	180	0.02979	0.00659
CALL	5	100,000	L	T	True	True	3	180	0.01147	0.00162
CALL	5	100,000	N	A	True	True	1	180	0.00805	0.00101
CALL	5	100,000	N	F	True	True	2	180	−0.01566	0.00104
CALL	5	100,000	N	H	True	True	4	180	0.00545	0.00226
CALL	5	100,000	N	P	True	True	4	180	−0.01526	0.00120
CALL	5	100,000	N	R	True	True	6	180	0.01221	0.00279
CALL	5	100,000	N	S	True	True	11	180	0.03020	0.00549
CALL	5	100,000	N	T	True	True	3	180	0.00692	0.00174
PUT	5	100,000	L	A	True	True	20	180	0.00657	0.00149
PUT	5	100,000	L	F	True	True	10	180	−0.01379	0.00083
PUT	5	100,000	L	H	True	True	14	180	0.00808	0.00157
PUT	5	100,000	L	P	True	True	5	180	−0.00987	0.00056
PUT	5	100,000	L	R	True	True	18	180	0.01005	0.00207
PUT	5	100,000	L	S	True	True	29	180	0.02747	0.00592
PUT	5	100,000	L	T	True	True	10	180	0.00445	0.00117
PUT	5	100,000	N	A	True	True	16	180	0.00084	0.00123
PUT	5	100,000	N	F	True	True	9	180	−0.01466	0.00075
PUT	5	100,000	N	H	True	True	13	180	0.00402	0.00176
PUT	5	100,000	N	P	True	True	7	180	−0.01529	0.00070
PUT	5	100,000	N	R	True	True	17	180	0.00987	0.00217
PUT	5	100,000	N	S	True	True	30	180	0.02535	0.00556
PUT	5	100,000	N	T	True	True	11	180	0.00622	0.00106

[a]Monte Carlo simulation values benchmarked against semi-analytical values from Fourier-based pricing approach. For the meaning of column headings refer to Table 10.1.

volatility (SV) model in combination with the CIR85 stochastic short rate (SI) model. This model $\mathcal{M}^{SVSI}$ is a special case of the general market model $\mathcal{M}^{BCC97}$ and exhibits risk-neutral dynamics as follows:

$$\begin{aligned} dS_t &= r_t S_t dt + \sqrt{v_t} S_t dZ_t^1 \\ dv_t &= \kappa_v(\theta_v - v_t)dt + \sigma_v \sqrt{v_t} dZ_t^2 \\ dr_t &= \kappa_r(\theta_r - r_t)dt + \sigma_r \sqrt{r_t} dZ_t^3 \end{aligned}$$

The major algorithm we apply is the LSM of Longstaff-Schwartz (Longstaff and Schwartz, 2001). However, in addition to moment matching and antithetic variates, we introduce a further variance reduction technique: control variates.

The LSM estimator (7.15) provides us with an estimate for an American option's value. We correct this estimator by the simulated differences gained from a control variate. Consider that we have simulated I paths of $X_t = (S_t, v_t, r_t)$ to value an American put option with maturity T and strike K. Then there are also I simulated present values of the corresponding European put option. They are given by $P_{0,i} = B_0(T)h_T(X_{T,i}), i \in \{1, ..., I\}$, with $h_T(x) \equiv \max[K - x, 0]$. The correction for the estimator (7.15) is as follows

$$\hat{V}_0^{CV} = \frac{1}{I}\sum_{i=1}^{I}\left(V_{0,i} - \lambda \cdot (P_{0,i} - P_0^{H93})\right) \tag{10.8}$$

For λ one can use the statistical correlation between the simulated European and American option present values. However, results from a number of numerical experiments indicate that simply setting $\lambda \equiv 1$ yields more accurate results in the test cases covered in this section.

10.4.1 Numerical Results

This sub-section presents numerical results from the simulation study based on American put options as implemented in the Python script of sub-section 10.6.4.

Parametrized Financial Model We consider all model parametrizations for the H93 and the CIR85 parts of the financial model from table 3 in Medvedev and Scaillet (2010). These are four different parameter sets for the financial model.

Per parameter set, American put options for three different maturities and moneyness levels, respectively, are valued:

- $T \in \left\{\frac{1}{12}, \frac{1}{4}, \frac{1}{2}\right\}$
- $K \in \{90, 100, 110\}$

All parameter sets and all values for the single options (for a total of 36 option values) are included in the Python script provided in sub-section 10.6.4. The script uses all seven discretization schemes for the square-root diffusions. With respect to the index process, it relies on an additive version of the log Euler scheme. The script implements the LSM algorithm with certain options to alter algorithm features (like control variates, moment matching or antithetic paths).

To measure accuracy, we consider the absolute difference between our script's values and the benchmark values from Medvedev and Scaillet (2010). As benchmark values we take their LSM estimates obtained by simulations with 50 exercise dates, 500 time steps and 1,000,000 paths. We say that our value estimates are accurate if the absolute difference is either smaller than 2.5 cents or 1.5% (i.e. we take the same yardsticks as for the European options).

Medvedev and Scaillet (2010) derive in their paper approximations for American option values under stochastic volatility (of H93 type) and stochastic interest rates (of CIR85 type) which can be evaluated very fast. They write on page 16:

> "To give an idea of the computational advantage of our method, a Matlab code implementing the algorithm of Longstaff and Schwartz (2001) takes dozens of minutes to compute a single option price while our approximation takes roughly a tenth of a second."

Apart from accuracy, we therefore want to take a look at how fast we can value options accurately with our Python implementation. This obviously is an important issue since "dozens of minutes" per single option price are of course unacceptable for practical applications.

Example Results from Simulation A numerical experiment with 10 simulation runs—for a total of 360 American put option values—yielded the following results (using control variate, moment matching and antithetic paths techniques):

- **discretization**: Full Truncation
- **time steps**: 20
- **paths**: 20,000
- **simulation runs**: 10
- **number of options**: 360
- **number of errors**: 13
- **average error**: −0.00096
- **total time**: 29.18 seconds
- **time per option**: 0.08 seconds

Three hundred and forty-seven out of 360 American put options are valued accurately given our yardsticks. The average valuation error is about −0.001 cents and therewith well below 1 cent in absolute value. The average relative error is not quite representative since the relative error for option values of about 0.01 cents easily reaches 100% and more. Nevertheless, it is only about +4%. Average time per option is about 0.08 seconds—which has to be compared with the "dozens of minutes" reported in Medvedev and Scaillet (2010). Our approach seems to be 1,000+ times faster (if we assume a 'single' dozen of minutes) with an accurateness that is consistent with a typical market microstructure.

Simulation Results Table 10.3 shows simulation results for different configurations of the LSM algorithm. Each of the 36 options is valued five times making for a total of 180 option valuations per configuration.

Interpretation of Results What are the reasons for the combination of reasonable accuracy and valuation speed of the Python script? Actually, there are a number of reasons:

- **implementation**: the LSM algorithm has been implemented in Python using the fast numerical library NumPy which runs at the speed of C code for certain operations; for some applications this may be faster than Matlab or other domain-specific environments like R
- **discretization**: we only use Euler discretization schemes which provide "sufficient" accuracy at a high speed; we let the simulated index level paths according to (10.4) drift step-by-step by the average of the two relevant short rate values
- **control variates**: the use of European put options as control variates (cf. Glasserman (2004), sec. 4.1) is of high or even highest importance for variance reduction and accuracy of the LSM estimator
- **moment matching**: we correct the set of standard normal pseudo-random numbers generated by Python to match the first two moments correctly (cf. Glasserman (2004), sec. 4.5),

TABLE 10.3 Valuation results for American put options in H93 and CIR85 model for parametrizations from Medvedev and Scaillet (2010). Performance yardsticks are $PY_1 = 0.025$ and $PY_1 = 0.015$.[a]

R	*M*	*I*	XD	CV	MM	AP	#ER	#OP	AE	MSE
5	20	25,000	A	True	True	True	1	180	−0.00117	0.00064
5	20	25,000	F	True	True	True	1	180	−0.00105	0.00042
5	20	25,000	H	True	True	True	5	180	0.00043	0.00046
5	20	25,000	P	True	True	True	4	180	−0.00379	0.00047
5	20	25,000	R	True	True	True	5	180	−0.00187	0.00058
5	20	25,000	S	True	True	True	3	180	−0.00290	0.00044
5	20	25,000	T	True	True	True	2	180	0.00072	0.00062
5	20	35,000	A	True	True	True	1	180	−0.00836	0.00050
5	20	35,000	F	True	True	True	1	180	−0.00289	0.00043
5	20	35,000	H	True	True	True	5	180	−0.00205	0.00057
5	20	35,000	P	True	True	True	4	180	−0.00328	0.00039
5	20	35,000	R	True	True	True	4	180	−0.00543	0.00051
5	20	35,000	S	True	True	True	2	180	−0.00408	0.00044
5	20	35,000	T	True	True	True	2	180	−0.00283	0.00035
5	25	25,000	A	True	True	True	2	180	−0.00171	0.00045
5	25	25,000	F	True	True	True	1	180	0.00126	0.00039
5	25	25,000	H	True	True	True	2	180	−0.00177	0.00043
5	25	25,000	P	True	True	True	1	180	−0.00083	0.00037
5	25	25,000	R	True	True	True	2	180	−0.00016	0.00059
5	25	25,000	S	True	True	True	4	180	0.00044	0.00050
5	25	25,000	T	True	True	True	3	180	−0.00173	0.00054
5	25	35,000	A	True	True	True	2	180	−0.00162	0.00043
5	25	35,000	F	True	True	True	1	180	−0.00135	0.00045
5	25	35,000	H	True	True	True	2	180	−0.00166	0.00041
5	25	35,000	P	True	True	True	1	180	−0.00199	0.00036
5	25	35,000	R	True	True	True	0	180	−0.00436	0.00028
5	25	35,000	S	True	True	True	5	180	−0.00429	0.00048
5	25	35000	T	True	True	True	1	180	−0.00144	0.00045

[a]Monte Carlo simulation estimates benchmarked against LSM values from Medvedev and Scaillet (2010). The columns report the following values: ***R*** = number of valuation runs, ***M*** = number of time steps, ***I*** = number of paths per single option valuation, **XD** = (first letter of) discretization scheme for square-root diffusions, **CV** = control variates, **MM** = moment matching, **AP** = antithetic paths, **#OP** = number of options valued, **#ER** = number of errors out of #OP, **AE** = average error over all valuations, **MSE** = mean squared error of all valuations.

i.e. the mean is adjusted to 0.0 and the standard deviation to 1.0; we also correct the first moment of the simulated index level paths according to (10.4) step by step to account for some remaining errors

- **antithetic paths**: as a general variance reduction technique we generate, as in Medvedev and Scaillet (2010), antithetic paths (cf. Glasserman (2004), sec. 4.2) such that convergence of the algorithm may improve somewhat

- **use of paths**: we use only in-the-money paths such that both the estimation of the regressions becomes faster (in particular for out-of-the-money options) and convergence of the algorithm may improve
- **basis functions**: we use all in all ten different basis functions for the regressions in the LSM implementation
- **exercise at** $t = 0$: we allow for exercise at $t = 0$ such that we get at least the inner value as the option price for the in-the-money cases
- **paths**: our LSM implementation allows a significant reduction in the number of discretization intervals (25 instead of 500 as in Medvedev and Scaillet (2010)) and paths (35,000 instead of 1,000,000); our approach reduces the number of necessary simulated values by a factor of more than 500 and halves the number of regressions (25 exercise dates instead of 50)
- **recycling**: we use the same set of random numbers for the 36 options to be valued per simulation run; we also use the same simulated processes for each of the three options per time-to-maturity
- **hardware**: of course, hardware also plays a role; the computational times reported for the script are from a server with Intel Xeon CPU E3-1231 v3 @ 3.40GHz; Python 2.7 and NumPy ran on a Linux 64 bit operating system; however, better hardware or parallelization techniques could further speed up calculations

Importance of Algorithm Features In this sub-section, we report further simulation results for variants of the LSM algorithm implementation. The aim is to identify those features of the implementation that indeed contribute to accuracy. Using the same seed value for the Python pseudo-random number generator, we replicate the 180 American option valuations several times—changing, respectively, features of the algorithm implementation. Table 10.4 shows the results.

It is obvious that the use of control variates is of paramount importance for accuracy. By contrast, moment matching and antithetic variates may be beneficial or not (if at all, then on a small scale). In view of the rather small additional computational time needed to include control variates they should be used whenever possible in such a context.

10.4.2 Higher Accuracy vs. Lower Speed

In some circumstances, our yardsticks used to assess accuracy may be too lax. Even if only for theoretical reasons, one might be interested in the LSM estimator (corrected with the help of a control variate) being even closer to the true (i.e. theoretical) option value. To this end, we set the performance yardsticks now to $PY_1 \equiv 0.01$ currency units (i.e. 1%) and $PY_2 \equiv 0.01$ (i.e. 1 percent). In particular, the 1 cent threshold is reasonable since it represents the smallest currency unit in general. Therefore it is often used to judge accuracy. For example, Longstaff and Schwartz (2001), p. 127, write: "Of the 20 differences shown in Table 1, 16 are less than or equal to one cent in absolute value."

To better meet the new yardsticks, we increase the number of time intervals to 50 as well as the number of paths to 100,000 and 200,000, respectively. As the results in Table 10.5 show, there are six valuation errors for the 180 options in the case of 50 time steps, 100,000 paths

TABLE 10.4 Valuation results for American put options in H93 and CIR85 model for parametrizations from Medvedev and Scaillet (2010). Performance yardsticks are $PY_1 = 0.025$ and $PY_1 = 0.015$.[a]

R	*M*	*I*	XD	CV	MM	AP	#ER	#OP	AE	MSE
5	20	35,000	A	False	False	False	54	180	−0.01504	0.00412
5	20	35,000	A	False	False	True	42	180	−0.01279	0.00365
5	20	35,000	A	False	True	False	43	180	−0.01126	0.00368
5	20	35,000	A	False	True	True	42	180	−0.01366	0.00351
5	20	35,000	A	True	False	False	0	180	−0.00319	0.00041
5	20	35,000	A	True	False	True	2	180	−0.00594	0.00046
5	20	35,000	A	True	True	False	1	180	−0.00364	0.00037
5	20	35,000	A	True	True	True	0	180	−0.00499	0.00045
5	20	35,000	F	False	False	False	44	180	−0.00894	0.00394
5	20	35,000	F	False	False	True	43	180	−0.01226	0.00379
5	20	35,000	F	False	True	False	42	180	−0.01283	0.00374
5	20	35,000	F	False	True	True	46	180	−0.01179	0.00422
5	20	35,000	F	True	False	False	4	180	−0.00525	0.00059
5	20	35,000	F	True	False	True	3	180	−0.00366	0.00038
5	20	35,000	F	True	True	False	2	180	−0.00416	0.00046
5	20	35,000	F	True	True	True	2	180	−0.00618	0.00042
5	20	35,000	P	False	False	False	45	180	−0.01272	0.00432
5	20	35,000	P	False	False	True	44	180	−0.01313	0.00407
5	20	35,000	P	False	True	False	42	180	−0.01241	0.00371
5	20	35,000	P	False	True	True	41	180	−0.01339	0.00385
5	20	35,000	P	True	False	False	3	180	−0.00337	0.00046
5	20	35,000	P	True	False	True	4	180	−0.00540	0.00047
5	20	35,000	P	True	True	False	1	180	−0.00322	0.00036
5	20	35,000	P	True	True	True	3	180	−0.00154	0.00045
5	20	35,000	T	False	False	False	53	180	−0.01329	0.00431
5	20	35,000	T	False	False	True	40	180	−0.01140	0.00374
5	20	35,000	T	False	True	False	42	180	−0.01210	0.00400
5	20	35,000	T	False	True	True	42	180	−0.01345	0.00367
5	20	35,000	T	True	False	False	1	180	−0.00513	0.00050
5	20	35,000	T	True	False	True	1	180	−0.00530	0.00042
5	20	35,000	T	True	True	False	3	180	−0.00479	0.00042
5	20	35,000	T	True	True	True	2	180	−0.00446	0.00039

[a]Monte Carlo simulation estimates benchmarked against LSM values from Medvedev and Scaillet (2010). For the meaning of column headings refer to Table 10.3.

and the Full Truncation scheme. The average valuation error in this case is around −0.5 cents. However, further increasing the number of paths to 200,000 ceteris paribus does not guarantee better valuation results, as is also illustrated in Table 10.5.

These results illustrate the trade-off between valuation accuracy and speed quite well. By increasing the number of time intervals and paths per simulation, you can get closer to the true (theoretical) value—just as the convergence results of Clément et al. (2002) imply. Longer valuation times are the price to pay.

TABLE 10.5 Valuation results for American put options in H93 and CIR85 model for parametrizations from Medvedev and Scaillet (2010). Performance yardsticks are $PY_1 = 0.01$ and $PY_1 = 0.01$.[a]

R	*M*	*I*	XD	CV	MM	AP	#ER	#OP	AE	MSE
5	20	35,000	A	True	True	True	10	180	−0.00417	0.00038
5	20	35,000	F	True	True	True	11	180	−0.00241	0.00043
5	20	35,000	P	True	True	True	10	180	−0.00323	0.00051
5	20	35,000	T	True	True	True	15	180	−0.00318	0.00051
5	20	100,000	A	True	True	True	14	180	−0.00928	0.00048
5	20	100,000	F	True	True	True	15	180	−0.00857	0.00033
5	20	100,000	P	True	True	True	20	180	−0.00855	0.00047
5	20	100,000	T	True	True	True	15	180	−0.00981	0.00047
5	20	200,000	A	True	True	True	17	180	−0.00966	0.00040
5	20	200,000	F	True	True	True	19	180	−0.01116	0.00044
5	20	200,000	P	True	True	True	15	180	−0.01055	0.00043
5	20	200,000	T	True	True	True	16	180	−0.01032	0.00041
5	50	35,000	A	True	True	True	12	180	−0.00211	0.00040
5	50	35,000	F	True	True	True	16	180	−0.00004	0.00048
5	50	35,000	P	True	True	True	14	180	0.00020	0.00043
5	50	35,000	T	True	True	True	15	180	−0.00258	0.00047
5	50	100,000	A	True	True	True	7	180	−0.00478	0.00031
5	50	100,000	F	True	True	True	6	180	−0.00536	0.00028
5	50	100,000	P	True	True	True	7	180	−0.00657	0.00032
5	50	100,000	T	True	True	True	9	180	−0.00591	0.00034
5	50	200,000	A	True	True	True	7	180	−0.00783	0.00034
5	50	200,000	F	True	True	True	7	180	−0.00720	0.00038
5	50	200,000	P	True	True	True	3	180	−0.00709	0.00029
5	50	200,000	T	True	True	True	9	180	−0.00743	0.00032

[a]Monte Carlo simulation values benchmarked against LSM values from Medvedev and Scaillet (2010). For the meaning of column headings refer to Table 10.3.

10.5 CONCLUSIONS

Monte Carlo simulation is an indispensable tool for the valuation of non-vanilla equity derivatives and for risk management purposes. However, even valuing simple products correctly by simulation in more complex models—like the ones of CIR85, H93 or BCC97—is already a daunting task. This chapter first shows how to correctly discretize the square-root diffusion in the CIR85 model and value zero-coupon bonds numerically. It then proceeds and values European call and put options in the H93 model where the variance process is discretized by a Euler scheme—a total of seven schemes is implemented to allow for numerical comparisons.

Section 4 then adds the CIR85 short rate model to the H93 model to value American put options by Monte Carlo simulation and the LSM algorithm. We show that our Python implementation allows for quite fast valuations in this context—the script needs only about a tenth of a second for a single option valuation. This is, among others, accomplished by the use of three variance reduction techniques: control variates, moment matching and antithetic paths. In this context, control variates play a dominant role in increasing valuation accuracy.