Assignment #5: Monte Carlo Methods - Improving Accuracy and Efficiency

B473 Numerical Methods in Finance
Prof. Dr.-Ing. Rainer Schöbel

Winter Term 2016-2017

Konstantin Smirnov
Student-ID: 3980253
Economics and Finance
Tübingen, March 28, 2017

List of Figures

Contents

1 Introduction

This assignment discusses another famous and important tool of computational finance, the Monte-Carlo approach. Due to its simplicity and flexibility it is widely used in practice to solve numerical issues. In this assignment, the different Monte Carlo techniques will be applied on option pricing theory.
As we know, in option-pricing it is required to compute the risk-neutral probability of an underlying, with respect to the stoachstic differential equation (SDE), which can alternatively expressed as an integral I. Generally speaking, in a stochastic framework, this can also be expressed as the expected value **E**[g(U)]. In a simple case where the random variable $U \sim (0,1)$ is uniformally distributed, we get following expression (Brandimarte, 2006, p. 221):

$$I = \mathbf{E}[g(U)] = \int_0^1 g(x)d(x)$$

The expected value can be estimated by simply computing the sample mean:

$$\hat{I}_m = \frac{1}{m}\sum_{i=1}^{m} g(U_i)$$

Under the Law of Large Numbers (LLN) this leads, with a probability of 1, to the population mean I:

$$\lim_{m\to\infty} \hat{I}_m = I$$

To get a more informal idea, Glasserman (2003, p. 30) simplified the methodology of the risk-neutral derivative pricing framework in the following way ("Monte Carlo Recipe for Cookbook Pricing"):

- replace the drift η of your SDE with the risk-free interest rate and simulate paths
- calculate payoff of derivative security on each path
- discount payoffs at the risk-free rate
- calculate average over paths

The advantages of a Monte-Carlo approach are its generality and simplicity. It can handle stochastic volatility models (discussed in chapter 3) or complex exotic options in a relative easy fashion.
But there are also difficulties this approach has to face. In general, three major problems do occur:

- generation of (pseudo)-random numbers
- discretisation error
- sampling errors

In this assignment we will focus mainly on *discretisation* and *sampling* erros, which are closely connected to accuracy and efficiency questions (Brandimarte, 2006, p. 429 - 431). *Discretisation errors* arise from the discretisation of the continuous sample path whereas *sampling errors* occur due to a too small number of simulations *N*, closely related to the LLN.
Accuracy can be improved with an increase in the number of time increments n or number of simulations run N but this would come with an increase in computational effort. This paper discusses how *discretisation errors* can be decreased by using different discretisation

techniques. Specifically, it will be discussed how the *Milstein scheme* can reduce discretisation errors. There exist also various techniques to reduce sampling errors. In this assignment the focus lays on the *control variate technique* to reduce the variance of the sample. Besides this we will consider technical implementation in *MATLAB*.
Note that regardless the just listed theoretical approaches to increase accuracy and efficiency, CPU and GPU performance increased heavily in the last century which is another major source of efficiency gain, in general. Betkaoui, Thomas and Luk (2010) discuss this topic from a technical point of view. They compared technical efficiency by running 1,000,000 simulations of an Asian option. It could be shown that the use of new processor technologies lead to a performance improvement of up to 18-times over a multi-threaded software implementation.

2 Improving strong convergence with the Milstein scheme

As we just noted, discretisation errors arise from the discretisation of a continuous function. Therefore we will compare the *Euler Scheme* and a refinement, the *Milstein scheme*, with the continuous path of a stock. Specifically, it will be shown how the *Milstein* scheme can reduce the discretisation error significantly.

2.1 Theoretical Framework

Since we are interested in the generation of derivatives, the simulation path of the underlying plays a crucial role in computing the final price (Brandimarte, 2006, p. 431).
A general representation of a typical SDE is:

$$dS_t = a(S_t, t)dt + b(S_t, t)dW_t \tag{1}$$

The most simple discretisation approach with respect to (1) is known as the *Euler scheme*:

$$dS_t = S_{t+\Delta t} - S_t = a(S_t, t)\Delta t + b(S_t, t)\sqrt{\Delta t}\epsilon \tag{2}$$

where Δt is the discretisation step and $\epsilon \sim \mathbf{N}(0, 1)$.
More specifically, under geometric Brownian Motion the continuous path of an underlying stock S is defined as:

$$dS_t = \eta S_t dt + \sigma S_t dW_t \tag{3}$$

and with the associated Euler scheme with respect to (2):

$$S_{t+\Delta t} = (1 + \eta\Delta t)S_t + \sigma S_t\sqrt{\Delta t}\epsilon \tag{4}$$

Representation (4) is a simple and straight-forward representation of the underlying path. Reducing Δt would decrease the discretisation error but this also would come with higher computational effort.
Applying Ito's lemma on equation (3) would also lead to another alternative straight-forward representation:

$$S_T = S(t)exp[(\eta - \frac{1}{2}\sigma^2)\Delta t + \sigma\epsilon\sqrt{\Delta t}] \tag{5}$$

but keep in mind that this kind of transformation (5) is not always applicable, especially for more complex SDE (Brandimarte,2006, p. 433).
One way to reduce computational burden would be the use of a different discretisation scheme. An alternative to the Euler Scheme is the *Milstein scheme* which is defined as (Glasserman, 2004, p. 343.):

$$\Delta S = a(S, t)\Delta t + b(S, t)\epsilon\sqrt{\Delta t} + \frac{1}{2}bb'(\epsilon^2 - 1)\Delta t \tag{6}$$

where *b'(S,t)* is the partial derivative of the constant b with respect to S.
In our specific case, when b(S,T)=ηS with b'(S,T)=η, (6) this leads to:

$$\Delta S = rS\Delta t + \eta S\varepsilon\sqrt{\Delta t} + \frac{1}{2}\eta^2 S(\varepsilon^2 - 1)\Delta t \tag{7}$$

When it comes to truncation errors, it can be shown that the Euler scheme approximation (4) has an error of order $O(\Delta t)$ for the drift term and $O(\sqrt{\Delta t})$ for the diffusion term. The truncation error of the diffusion term in the Milstein scheme is of order $O(\Delta t)$, which is an improvement compared to the Euler scheme (Glasserman, 2004, p. 340 - 343). However, this improvement is not generally the case, so the choice of the scheme depends on the model itself. Indeed, if the diffusion coefficient is independent of the state variable b=b(t), one can show that the truncation errors of both schemes are of the same order. In that case, it would make sense to use an one-step simulation if a closed form solution exists. Only if b depends on the state variable b(S,t), as in our case (7), then the Milstein scheme is truly superior regarding accuracy and efficiency. Alternatively, one could try to use a transformation to make b independent of the state variable.
In the next section we will compare these schemes by application. Note that a further theoretical discussion can be found in Glasserman (2004, p. 344 - 348).

2.2 Discussion of Results

To discuss accuracy and efficiency of the Milstein scheme, following parameters were used to simulate the different stock price paths with a continuous function, the Euler scheme and the Milstein scheme:

Table 1: Sample parameters of problem 1

Description	**Variable**	**Value**
stock price	S	100
Time to maturity	T - t	0.5
risk-free interest rate	r	0.0953
volatility	η	0.25
initialization value of the random generator	seed	7777

In figure 1 all three simulated paths with respect to equations (3) (continuous), (4) (Euler) and (7) (Milstein) were simulated with the given parameters. We used a discretisation step size of 1000 steps in this illustration. In the full scale representation, the scheme approximations look already relatively accurate. However, as you can see in the right plot, the Milstein scheme delivers a more accurate version of the continuous path simulation compared to Euler scheme.
To get a better understanding of accuracy and efficiency, the final terminal value of the continuous path is opposed to the terminal values of both schemes for increasing discretisation step size n from 1^2 to 14^2, represented in figure 2. The right figure of this plot additionaly takes account of the discretisation error which is simply the difference between the terminal value and the schemes approximations, respectively for a given n.[1]

[1]This form of step-size increasement was chosen due to graphical illustration.

full scale zoomed in

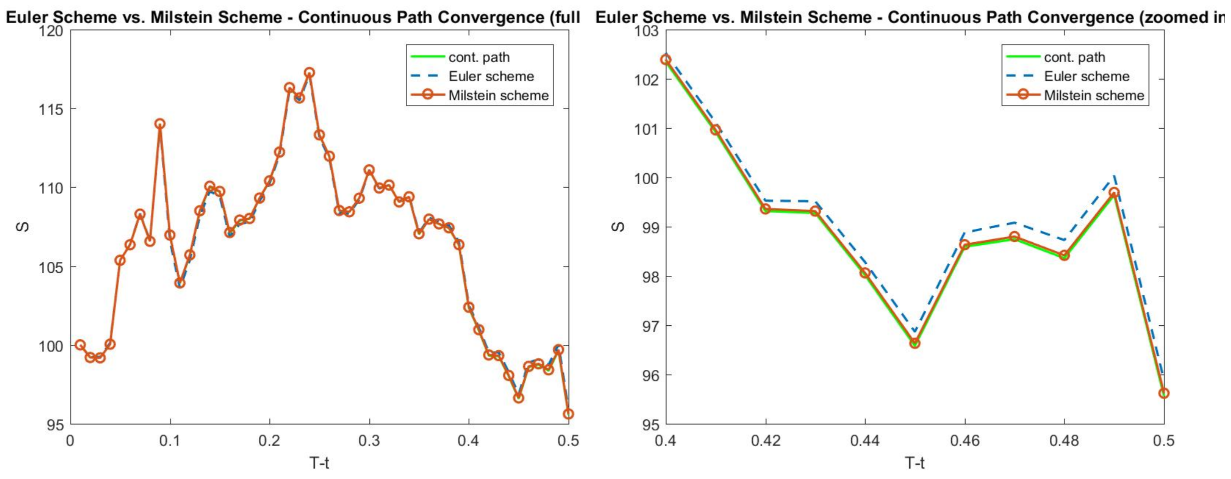

Figure 1: Euler Scheme vs. Milstein Scheme - Continuous Path Convergence

Cont. Path Convergence Error Convergence

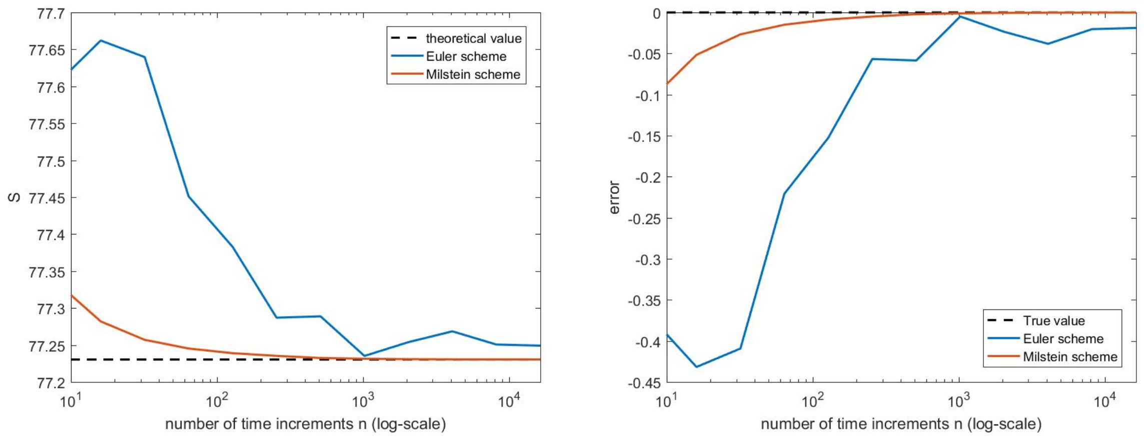

Figure 2: Euler Scheme vs. Milstein Scheme - Cont. Path Convergence and Discretization Error

Clearly, in both illustrations, the Milstein scheme delivers more precise results for a small step-size parameter compared to the Euler scheme. With a step-size of n=128 the error of the Milstein schemes has the value of -0.0086 while the error of the Euler scheme still has a relatively big value of -0,1526. With a step-size of n=2048 error of the Milstein scheme is already close to 0 with a value of -0,0005 while the Euler scheme only improved to -0,0234 which is in line with the theory described in the first part of this chapter. Indeed, in our case, the Euler scheme does not directly converge to the true value, even for relatively large discretisation steps leading to systematic errors which makes the Milstein scheme superior when it comes to accuracy.

Note that in this computations for the continuous path, the terminal value of equation (5) was used which leads to more accurate results compared to continuous path representation (3). In fact, the Milstein scheme does not converge to the terminal value of equation 3. This may come from the fact that (3) contains a discretisation error on itself since in practice the function is also discretised. Hence the more accurate approach (5) was used, where the Milstein scheme converged to the true value.

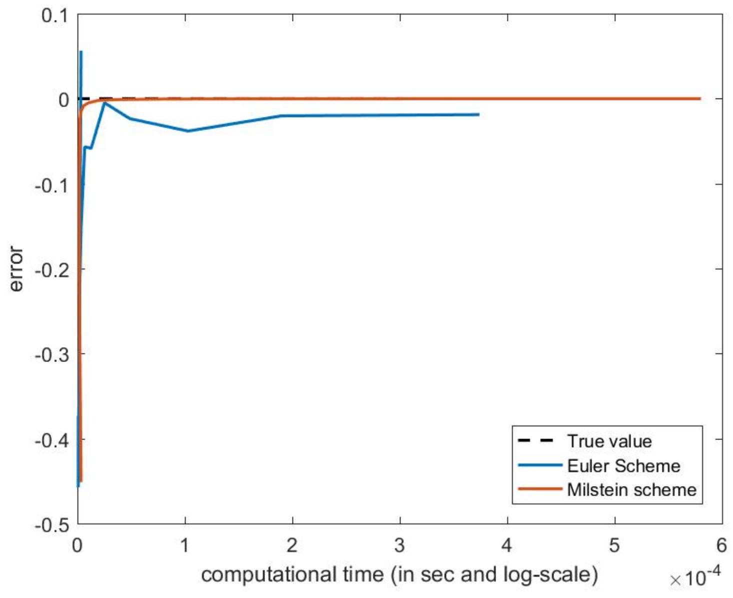

Figure 3: Euler Scheme vs. Milstein Scheme - Computational Effort

Since the error of the Milstein scheme vanishes for relative small discretisation steps, generally speaking, it converges faster to the true value but this does not imply that it is the general faster approach. It could be the case that the Milstein algorithm is time costly in its nature. But this can be disproved while having a look on figure (3). As you can see the Milstein scheme is not only superior when it comes to accuracy but also when it comes to time efficiency in general compared to the Euler scheme. Since, as we just have seen, the accuracy is relatively high, it reaches the true value fast.[2]

Summarized, the Milstein scheme is, in that simple case, superior to the Euler Scheme regarding efficiency and accuracy which is in line with the theory stated in the beginning of the chapter. Keep in mind that these result only holds if the coefficient b depends on the state variable, which is the case in our example.

2.3 Technical Implementation

All associated computations of chapter 2 can be called in the main script *problem1.m*. Since we need a stream of pseude-random number stream for the Brownian Motion, the *Rand.Generator* command was used to create a stream of (pseudo)-random numbers with a combined multiple recursive random number generator (’*mrg32k3a*’).

The functions *cont.m, euler.m and milstein.m* are called in the main script and basically contain the algorithms with respect to the mentioned equations in the theoretical part of this chapter.

The given parameters of the assignment and the (pseudo)-random stream, as well as the number of overall increments *nn1* and the discretisation step size *nn2* serve as input parameters. As an output the algorithms deliver the stock price paths as well as terminal values of the stock price path with the given parameters. The *Euler* and *Mils* algorithms additionally output the meisured time of one run.

Since in problem 1 we are interested in the simulated paths in general, the functions were simply called with a *nn11* of 1000 and a *nn12* of 50.

In the second part of the assignment, the Euler and Milstein algorithms are looped for different

[2]Note that time meisurement highly depends on the hardware and software used for computations.

squared step sizes *nn22*=1^2, 2^2..., to 14^2 (due to illustrative reasons) with respect to the input parameters and an overall number of increments *nn21* = 10000. As it was already mentioned, the analytic solution was used (*SPath*) instead of the disc. continuous version *cont.m.* All computations can be found in the *accuracy* output matrix where the terminal values and the step-size values are saved. The errors and the time are additionally saved in the *errors_euler* or *errors_milstein* and the *time_efficiency* vectors.
Note here that time measurement is highly subjective since it depends highly on the used hardware and the software of the system. In this assignment, all computations were run on an Windows 10 64-bit operation system with an Intel(R) Core(TM) i7-5500U CPU 2.40 Ghz.

3 Variance reduction using control variate techniques

As we described in the beginning of the chapter, inaccuracy in the Monte Carlo approach might arise as a sampling error. In that case, due to the LLN, the variance of the estimate can also be reduced by simply increasing the number of samples N but this would come with an increase in computational time.
In this section we will discuss how the control variate technique (CVT) can increase efficiency by decreasing the relative sampling error without increasing the number of sampled paths. Specifically, we apply this technique on a stochastic volatility derivative model of Schöbel and Zhu (1999).

3.1 Theoretical Framework

The theoretical framework of the variance reduction technique is described by Glasserman (2003, p. 185 - 186). Assume there are $Y_1, Y_2, ..., Y_i$ samples of a discounted payoff of the sampled path.
If Y_i are identical and independent distributed (i.i.d), the usual estimator of the sample is described as:

$$\hat{Y} = \mathbb{E}[Y_i] = \frac{1}{n}\sum_{i=1}^{n} Y_i \tag{8}$$

which converges under LLN to a probability of 1 with $n \to \infty$.
Suppose there is another estimate X_i along with Y_i which is also i.i.d. and with the same distribution as Y_i with a known expectation $\hat{X} = \mathbb{E}[X]$. The control variate estimator can then be described, for any fixed b, as:

$$\hat{Y}(b) = \hat{Y} - b(\hat{X} - E[X]) = \frac{1}{n}\sum_{i=1}^{n} Y_i - b(X_i - \mathbb{E}[X]) \tag{9}$$

where $\hat{X} - E[X]$ serves as a control in estimating $\mathbb{E}[Y]$.
One can show that the estimator (9) is unbiased and consistent (with probability 1):

$$\hat{Y}(b) = \mathbb{E}[Y] \quad and \quad \lim_{n\to\infty} = \mathbb{E}[\mathbb{Y}]$$

and has a variance of:

$$Var[Y_i(b)] = \sigma_Y^2 - 2b\sigma_X\sigma_Y\rho_{XY} + b^2\sigma_X^2 = \sigma^2(b)$$

with $\sigma_X^2 = Var[X]$, $\sigma_Y^2 = Var[Y]$ and with the correlation factor ρ_{XY} between the random variables X and Y.

The estimated variance of the usual estimator is:

$$\hat{\sigma}_Y^2 = Var[Y_i]/N; \tag{10}$$

and the estimated variance of the CVT is defined as:

$$\hat{\sigma}_{CVT}^2 = Var[Y_i(b)]/N; \tag{11}$$

If $b^2\sigma_X < 2b\sigma_Y\rho_{XY}$, then the estimated variance of the control variance estimator $\hat{\sigma}_{CVT}^2$ is smaller then the estimated variance of the usual estimator $\hat{\sigma}_Y^2$.
One can show that the optimal b, which minimizes the variance, is:

$$b^* = \frac{\sigma_Y}{\sigma_X}\rho_{XY} = \frac{Cov[X,Y]}{Var[X]} \tag{12}$$

In practice, $\mathbb{E}[Y]$ is unknown. Therefore, we use the sample counterparts which are defined as:

$$\hat{b}_N = \frac{\sum_{i=1}^N (X_i - \hat{X})(Y_i - \hat{Y})}{\sum_{i=1}^N (X_i - \hat{X})^2} \tag{13}$$

Substituting (12) into (9) results in the variance ratio between the controlled variance estimator and the uncontrolled estimator, which measures the effectiveness of a control variate, we will get:

$$\frac{Var[\hat{Y} - b^*(\hat{X} - \mathbb{E}[X])]}{Var[\hat{Y}]} = 1 - \rho_{XY}^2. \tag{14}$$

If the algorithms of the usual estimator and the CVT are roughly the same regarding computational time, then the speed up resulting from the control variate can be computed, with respect to equation (14), by:

$$\varrho = \frac{N}{1 - \rho_{XY}^2} \tag{15}$$

In other words this expression tells you the number of replications N needed with the usual estimate to get the same estimated CVT variance (11).
Equation (14) shows that a high correlation factor would lead to a sharp increase regarding efficiency. On the other side a relative low correlation factor would only marginally increase efficiency. For example, for N=1, a ρ of $= 0.95$ would lead to a speed up of around 10x compared to the usual estimator, while an $\rho = 0.9$ would achieve only half of the speed up of around 5x. Hence a relative high ρ is needed in order to gain efficiency when using this kind of variance reduction technique.
In the following chapter we will apply the usual estimator and the CVT modification on a European call with stochastic volatility (Schöbel and Zhu, 1999). The Schöbel and Zhu model (SZ-model) is defined as :

$$dS = rSdt + vSdW_1 \tag{16}$$

where the instantaneous volatility follows a mean-reverting Ornstein-Uhlenbeck process:

$$dv = \kappa(\theta - v)dt + \sigma dW_2 \tag{17}$$

where the Brownian motions dW_1 and dW_2 are correlated:

$$\rho dt = dW_1 dW_2$$

with

$$dW_1 = dW_1^*$$
$$dW_2 = \rho dW_1^* + \sqrt{1-\rho^2} dW_2^*$$
$$dW_1^* dW_2^* = 0$$

where the payoff function of a European Call is defined as:

$$Y_i = e^{-r(T-t)} max(S_{iT} - K, 0) \tag{18}$$

As a control variate the standard one-dimensional Black-Scholes-Merton (BSM) model is used, which is defined as:

$$dS = rSdt + S\sigma dW_1 \tag{19}$$

with dW_1:

$$dW_1 = \sqrt{dt}\epsilon$$

The population of mean of $\mathbb{E}$[X] is known since there exists a closed-form solution of the BSM. It is defined as:

$$C_{BCSM}(S,t) = Se^{(h-r)(T-t)} N(d_1) - Xe^{-r(T-t)} N(d_2) \tag{20}$$

with

$$d_1 = \frac{ln(\frac{S_t}{X}) + (h + 0.5\sigma^2)(T-t)}{\sigma\sqrt{T-t}}$$
$$d_2 = d_1 - \sigma\sqrt{T-t}$$

where N is normally distributed $N \sim (0, 1)$.

3.2 Discussion of Results

In the following section following parameters were used for our computations:

Table 2: Input parameters for problem 2

Description	**Variable**	**Value**
stock price	S	100
strike	K	100
Time to maturity	T - t	0.5
risk-free interest rate	r	0.0953
mean reversion parameter	κ	4.0
"volatility of volatility"	σ	0.1
long-run volatility	θ	0.2
correlation coefficient	ρ	-0.5
instantaneous volatility of the stock price	v	0.2
initialization value of the random generator	seed	7777

In the next subsections the effect of an increase in number of samples N and in an increase in number of time increments n will be analyzed and discussed.

3.2.1 Accuracy and Efficiency: the effect of an increase in simulations *N*

In this examination, the model was discretised with the Euler scheme, where the number of time increments n was fixed to n=5000. According to the assignment, the real price of this

call is 8.243.
First, the efficiency factor ϱ given in (15) was computed. The variables X and Y of our CVT estimator are highly correlated with $\rho_{XY} = 0.9851$ (for a N = 10,000) and hence the CVT model should reduce variance significantly. Indeed, our computations proofed this finding. For a N=1000, the ϱ is approximately $\sim$ 30491, so to achieve the same variance as the CVT with N=1000, the usual estimator would need N = 30491 simulations. Hence the CVT in our case is around 30.50x times more accurate then the usual estimator.
To illustrate how accuracy is improving, we additionally compute the 95%-confidence intervals (CI) of the estimates. The CVT CI's are defined as (Glasserman, 2003, p. 195):

$$CI_{+/-} = \hat{Y}(b) \pm 1.96 \frac{\hat{\sigma}}{\sqrt{N}} \tag{21}$$

and for the usual estimator, the 95% - CIs are defined as:

$$CI_{+/-} = \hat{Y} \pm 1.96 \frac{\hat{\sigma}}{\sqrt{N}} \tag{22}$$

where $\hat{\sigma}$ is the estimated standard deviation.

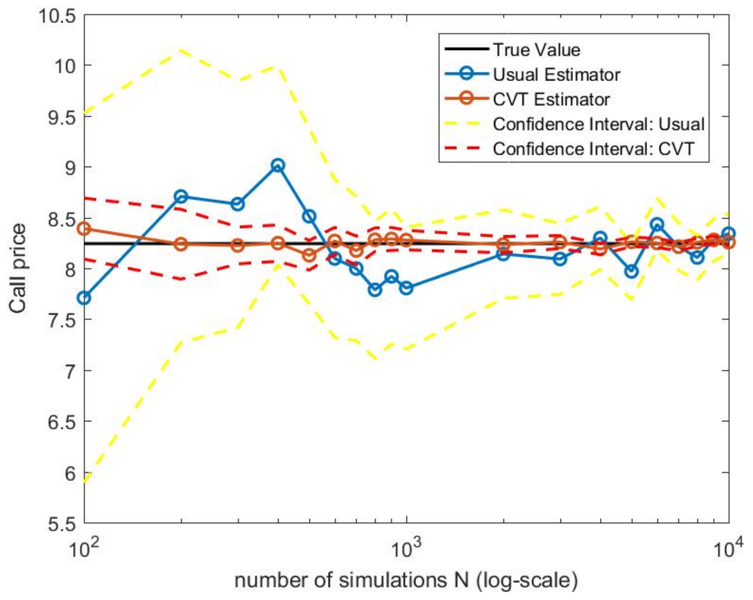

Figure 4: Accuracy of the CVT for increasing N - Usual Estimator vs. CVT

Illustration 4 shows the convergence behavior of the simulation with of the usual (Euler) estimator and the CVT estimator in an increase in number of simulations N, reaching from 100:100:1000 and 1000:1000:10000.[3]
The findings in these plots are in line with the theory. The CVT estimator already converges smoothly close to true value (8.243) even for relatively small number of simulation paths N where as the usual estimator oscillates vigorously, specifically in the beginning, around the true value. The CI's of the CVT are narrow where as the CI's of the usual estimator are relatively bright and only slowly get closer with increasing N. However, for that given intervals, even for large N, the usual estimator is not able to achieve the same accuracy as the improved CVT.

[3]This intervals were choosen due to illustrative reasons.

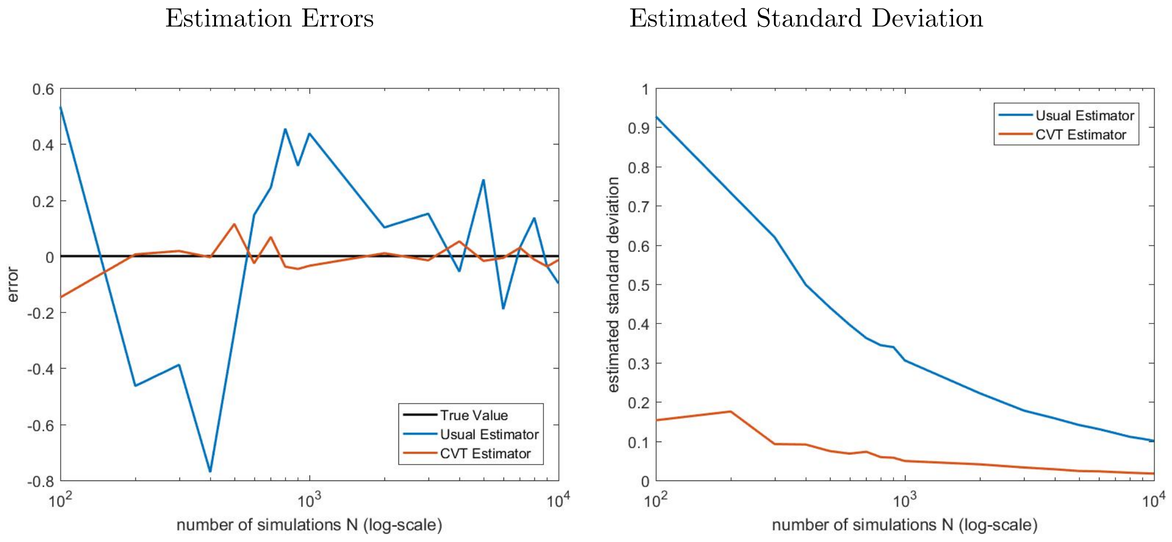

Figure 5: Accuracy of the CVT for increasing N - Estimated Error and Estimated Standard Deviation

Figure 5 highlights this result. The left figure shows the estimation errors for an increase in sample paths N with respect to the true value of 8.243. The estimated error for the CVT converges smoothly to the 0-error line, whereas the usual estimator has high oscillations which weaken only slowly with high N. The right plot shows the estimated standard deviation $\hat{\sigma}_Y/\sqrt{N}$ and $\hat{\sigma}_{CVT}/\sqrt{N}$. While for the CVT, the standard deviation starts already with relative small values, the values of the usual estimator starts with a high deviation and only drive slowly towards 0. However, it can never reach the precision of the CVT estimator for that specific N. You can also observe the big difference between the two estimated standard deviations, which approves the findings regarding ϱ that the CVT is around 30x more accurate.

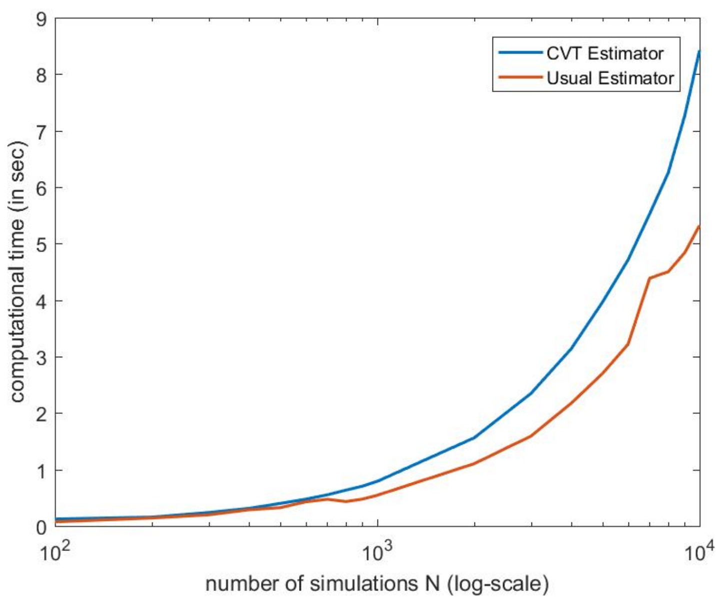

Figure 6: Efficiency of the CVT for increasing N - absolute Computational Effort

Summing up, the CVT modification, due to that highly correlated ρ_{XY} in that specific case, is truly superior to the usual estimator. This is also proven graphically. However, from an efficiency point of view, all these findings would not matter if computational burden of the CVT algorithm, f.e. due to too many complex operations, would be too time costly compared to the variance reduction improvement.

Figure 6 shows how many seconds the usual estimator or the CVT need to estimate a price for N number of generated sample paths. For a small number of observations paths, the time effort is only marginal, while for N=10000, the computation indeed needs 3 seconds longer then the usual estimator. From an absolute point of view, the CVT algorithm is more time costly which is not surprising since by its nature it has to handle more operations then the usual estimator algorithm.

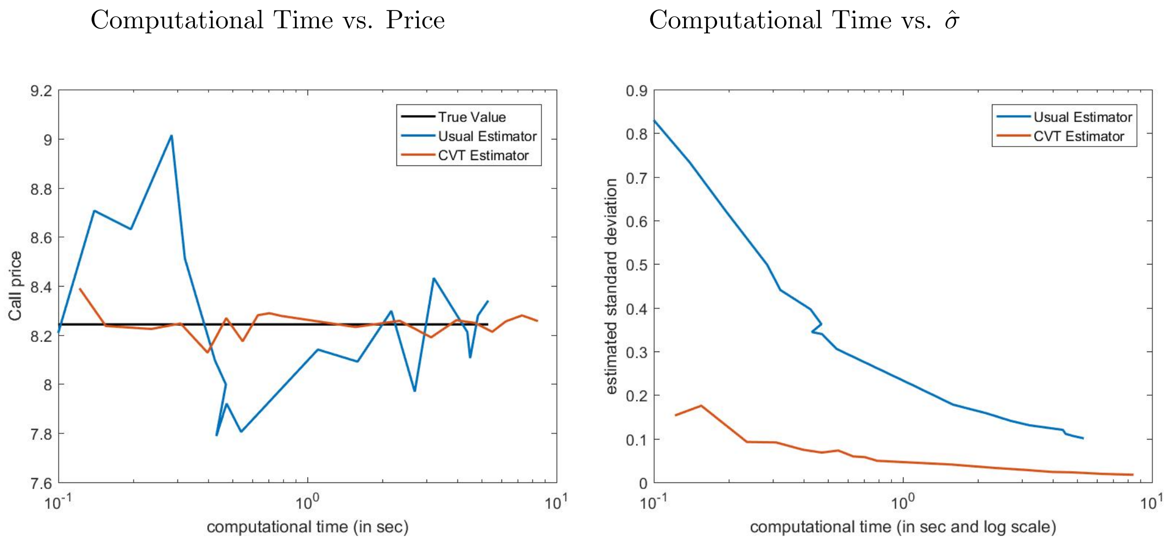

Figure 7: Efficiency of the CVT for increasing N - Price Convergence and $\hat{\sigma}$

Nonetheless, in plot 7 these findings can be put into perspective. The illustrations show the computational effort plotted against the Call price and the estimated standard deviation. The left plot again shows the high osciliation of the usual estimator. The computations might be shorter from an absolute point of view, but indeed the usual estimator is not able to deliver precise results as good as with CVT, in that given time. These findings are also substantiated when observing the measured computational time vs. estimated standard deviation, represeted in the right figure of the plot.

Summing up, the CVT is not only superior from an absolute accuracy point of view, but also superior when it comes to time efficiency. Keep in mind that the CVT is only efficient if the correlation ρ_{XY} is high enough which might not be the usual case when it comes to practical application.

3.2.2 Accuracy and Efficiency: the effect of refining the number of time increments n

In the next part of this subchapter, we will briefly discuss the effect of an increase in the number of time increments n, as it was discussed in chapter 2, on accuracy and efficiency of the usual estimate and the CVT. The algorithms were run with an increase in n from 2^1, 2^2 up to 2^{14} for a fixed number of simulations $N = 1000$ and N = 10,000.

In figure 8 it is examined which effect an increase in time increments n has on the accuracy for a fixed N=1000 and N=10,000. What you can see is that the refinement of n does not improve accuracy, either in the case of N=1000, nor in the case of N=10,000, independent of

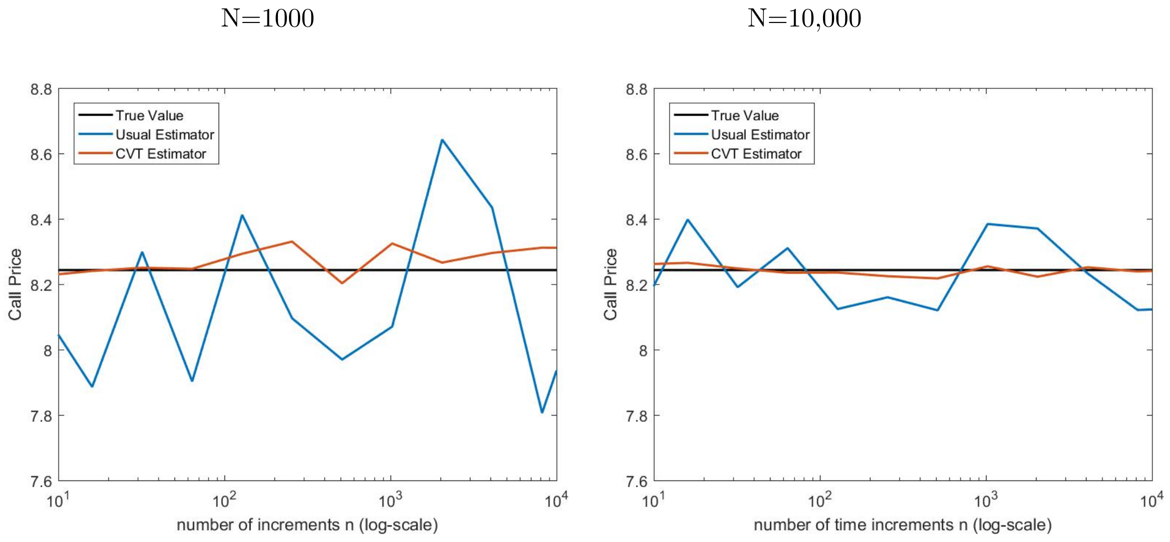

Figure 8: Comparison of Time Increment Convergence for different N=1000 and N=10,000

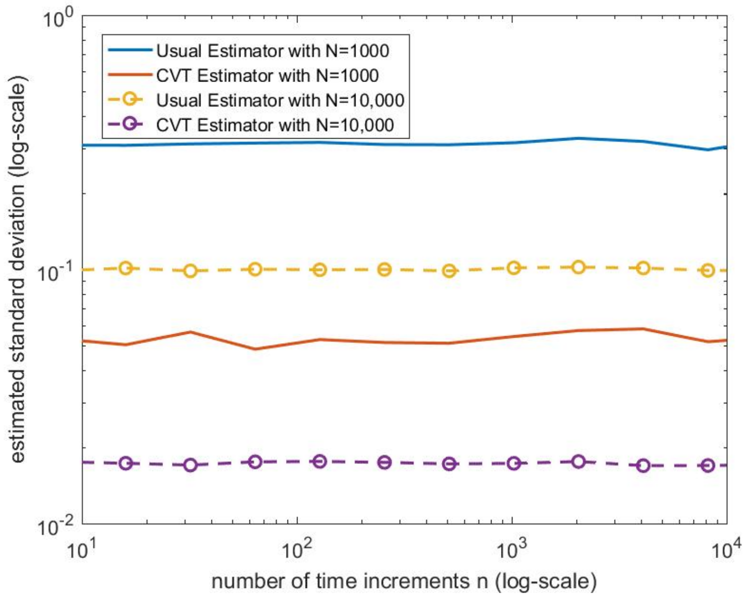

Figure 9: Time Increment Convergence - Estimated Standard Deviation

the choice of usual estimate or CVT. Volatility only reduces significantly for an increase in the number of simulated paths N.
These findings are confirmed in illustration 9. Indeed, the estimated standard deviation stays constant in all four cases for an increase in n. Hence an increase in n does not influence accuracy or efficiency. This is not surprising because the CVT modification by its nature affects the sampling error, not the discretisation error. Furter examinations regarding time efficiency were therefore skipped.

3.3 Technical Implementation

All associated computations of chapter 3 are splitted into two main scripts *Problem2a.m* and *Problem2b.m* where the main algorithms *MC_USUAL.m* for the usual estimator and *MC_CVT.m* for the CVT modification are called.
As an input of the algorithms we used the given parameters, the discretisation steps *nn* and the number of simulation paths *NN*. Again, two pseudo-random numbers with a combined multiple recursive random number generator were generated with help of the specified *Rand.Generator* and the *randn* command.
What was noticeable is how these errors behaved. The errors marginally differed when calling them in the algorithm function itself compared if they are generated outside the function in the main script. It seems that *MATLAB* does not only generate pseudo-random numbers specified by the initialization seed value but also on the location where these numbers were generated. Generating random numbers in the algorithm *MC_USUAL.m* hence slightly differed compared to the errors in the algorithm *MC_CVT.m* since they are not generated in the same location. Indeed, this would have an effect on comparability of the findings. Therefore we decided to generate the errors outside the algorithms in the main script and handle them as additional input parameters of the functions *MC_USUAL.m* and *MC_CVT.m.*
Both algorithms basically are constructed in the same manner as it was discussed in the theoretical framework of this chapter. The first part of *Problem2a.m* consists of the proof that efficiency factor ϱ holds as it was described in the theoretical part. By calling this subsection you can see that the estimated variances Var_CVT and Var_U are equal for a CVT run with N=1000 or a run with the usual estimator with *N_EF*=30491.
The second part of the main script *Problem2a.m* consists of the loops for increasing *NN* and fixed *nn.* The findings are saved in the output matrix *SZ_usual* and *SZ_CVT* for the usual and CVT case, respectively. The exact specification of each column can be found in the code. All these findings are plotted in the usual manner with help of the *plot* or *semilogx* commands. The time was again measured with the *tic* and *toc*commands and can differ depending on the system.
The main script *Problem2b.m* is constructed in the same fashion but in that case we fixed *NN* and looped over *nn.* All findings again are saved in specified output vectors which are then plotted.
Note that the Monte Carlo approach can be relatively time consuming, so the number of simulations was set to a maximum of *NN*=10,000 because it delivers precise results in a reasonable time. However, the program was written dynamically, so the parameter NN can be increased when running on more powerful system.
Again all computations were run on an Windows 10 64-bit operation system with an Intel(R) Core(TM) i7-5500U CPU 2.40 Ghz.

4 References

Betkaoui, Brahim, David B. Thomas, and Wayne Luk. *Comparing performance and energy efficiency of FPGAs and GPUs for high productivity computing.* Field-Programmable Technology (FPT), 2010 International Conference on IEEE, 2010.

Brandimarte, Paolo. *Numerical Methods in Finance and Economics. - A MATLAB- Based Introduction*, 2nd ed., John Wiley and Sons, Hoboken NJ, 2006.

Glassermann, Paul. Monte Carlo Methods in Financial Engineering, Springer Verlag, New York 2003.

Schöbel, Rainer and Jianwei Zhu. *Stochastic Volatility With an Ornstein-Uhlenbeck Process: An Extension.* European Finance Review 3, 1999, 23 - 46.