QF4102 Financial Modelling and Computation Assignment 3

G01 Wang Zexin, Chen Penghao

November 19, 2017

1 Transformed Black-Scholes PDE model

Consider the **transformed** Black-Scholes PDE model:

$$\begin{cases} \frac{\partial u}{\partial t} + \frac{\sigma^2}{2}\frac{\partial^2 u}{\partial x^2} + (r - q - \frac{\sigma^2}{2})\frac{\partial u}{\partial x} - ru = 0, \quad x \in (-\infty, \infty), t \in [0, T) \\ u(x, T) = \varphi(x), \end{cases}$$

1.1 Derivation of fully implicit scheme

Evaluate partial derivatives at (x_n^i, t_n) where $t_n = n\Delta t, x_n^i = i\Delta x, n \in [0, \frac{T}{\Delta t}), x \in [-x_{max}, x_{max}], I = \frac{x_{max}}{\Delta x}$

$$\text{Use the forward time finite difference formulae : } \left.\frac{\partial u}{\partial t}\right|_{(x_n^i, t_n)} = \frac{u_{n+1}^i - u_n^i}{\Delta t} + O(\Delta t)$$

$$\text{Use the centred space finite difference formulae : } \left.\frac{\partial u}{\partial x}\right|_{(x_n^i, t_n)} = \frac{u_n^{i+1} - u_n^{i-1}}{2\Delta x} + O[(\Delta x)^2]$$

$$\text{Use the centred space finite difference formulae : } \left.\frac{\partial^2 u}{\partial x^2}\right|_{(x_n^i, t_n)} = \frac{u_n^{i+1} - 2u_n^i + u_n^{i-1}}{(\Delta x)^2} + O[(\Delta x)^2]$$

The finite difference equation is hence :

$$\frac{u_{n+1}^i - u_n^i}{\Delta t} + O(\Delta t) + \frac{\sigma^2}{2}\frac{u_n^{i+1} - 2u_n^i + u_n^{i-1}}{(\Delta x)^2} + (r - q - \frac{\sigma^2}{2})\frac{u_n^{i+1} - u_n^{i-1}}{2\Delta x} + O[(\Delta x)^2] - ru_n^i = 0$$

$$\frac{u_{n+1}^i - u_n^i}{\Delta t} + O(\Delta t) + O[(\Delta x)^2] = -\frac{\sigma^2}{2}\frac{u_n^{i+1} - 2u_n^i + u_n^{i-1}}{(\Delta x)^2} - (r - q - \frac{\sigma^2}{2})\frac{u_n^{i+1} - u_n^{i-1}}{2\Delta x} + ru_n^i$$

$$\frac{U_{n+1}^i - U_n^i}{\Delta t} = rU_n^i - \frac{\sigma^2}{2}\frac{U_n^{i+1} - 2U_n^i + U_n^{i-1}}{(\Delta x)^2} - (r - q - \frac{\sigma^2}{2})\frac{U_n^{i+1} - U_n^{i-1}}{2\Delta x}$$

$$U_{n+1}^i = U_n^i + \Delta t[rU_n^i - \frac{\sigma^2}{2}\frac{U_n^{i+1} - 2U_n^i + U_n^{i-1}}{(\Delta x)^2} - (r - q - \frac{\sigma^2}{2})\frac{U_n^{i+1} - U_n^{i-1}}{2\Delta x}]$$

$$U_{n+1}^i = U_n^i(1 + r\Delta t) - \frac{\Delta t}{2(\Delta x)^2}[\sigma^2(U_n^{i+1} - 2U_n^i + U_n^{i-1}) + \Delta x(r - q - \frac{\sigma^2}{2})(U_n^{i+1} - U_n^{i-1})]$$

$$U_{n+1}^i = U_n^{i-1}[\frac{\Delta t(r - q - \frac{\sigma^2}{2})}{2\Delta x} - \frac{\sigma^2\Delta t}{2(\Delta x)^2}] + U_n^i[1 + r\Delta t + \frac{\sigma^2\Delta t}{(\Delta x)^2}] + U_n^{i+1}[-\frac{\Delta t(r - q - \frac{\sigma^2}{2})}{2\Delta x} - \frac{\sigma^2\Delta t}{2(\Delta x)^2}]$$

$$U_{n+1}^i = aU_n^{i-1} + bU_n^i + cU_n^{i+1}, \forall -I + 1 \le i \le I - 1$$

$$\text{where } a = \gamma - \frac{\alpha}{2}, b = \beta + \alpha, c = -\gamma - \frac{\alpha}{2}, \alpha = \frac{\sigma^2\Delta t}{(\Delta x)^2}, \beta = 1 + r\Delta t, \gamma = \frac{\Delta t(r - q - \frac{\sigma^2}{2})}{2\Delta x}$$

The boundary conditions are as follows:

$$U_n^I = e^{-q(T - n\Delta t)}\exp(I\Delta x) - e^{-r(T - n\Delta t)}X, \text{ when the underlying value is very large at } \exp(I\Delta x)$$

$$U_n^{-I} = 0, \text{ when the underlying value is very small at } \exp(-I\Delta x)$$

With the values of U_n^{-I} and U_n^I specified, we can express the FDE into matrix form.

$$\begin{bmatrix} b & c & \dots & \dots & \dots & \dots & \dots \\ a & b & c & \dots & \dots & \dots & \dots \\ \dots & a & b & c & \dots & \dots & \dots \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ \dots & \dots & \dots & a & b & c & \dots \\ \dots & \dots & \dots & \dots & a & b & c \\ \dots & \dots & \dots & \dots & \dots & a & b \end{bmatrix} \begin{bmatrix} U_n^{-I+1} \\ U_n^{-I+2} \\ U_n^{-I+3} \\ \dots \\ U_n^{I-3} \\ U_n^{I-2} \\ U_n^{I-1} \end{bmatrix} = \begin{bmatrix} U_{n+1}^{-I+1} \\ U_{n+1}^{-I+2} \\ U_{n+1}^{-I+3} \\ \dots \\ U_{n+1}^{I-3} \\ U_{n+1}^{I-2} \\ U_{n+1}^{I-1} \end{bmatrix} + \begin{bmatrix} -aU_n^{-I} \\ 0 \\ \vdots \\ \vdots \\ 0 \\ -cU_n^I \end{bmatrix}$$

More concisely, we can name the tridiagonal matrix A and the right hand side vector F to express the FDE in this form: $AU_n = U_{n+1} + F \rightarrow U_n = A^{-1}(U_{n+1} + F)$

1.2 Finite Difference Scheme Algorithm on fully implicit scheme

Data: S_0, X, r, T, σ, I, N, x_{max}

Result: c_{IDS}: Option Premium

$\Delta t = \dfrac{T}{N}$, $\Delta x = \dfrac{x_{max}}{I}$;

$\alpha = \frac{\Delta t(r-q-\frac{\sigma^2}{2})}{2\Delta x}$;

$\beta = 1 + r\Delta t$;

$\gamma = \frac{\sigma^2 \Delta t}{2(\Delta x)^2}$;

$a = \alpha - \gamma$;

$b = \beta + \alpha$;

$c = -\alpha - \gamma$;

for $i = -I+1, -I+2, \dots, I-2, I-1$ **do**

  $U_N^i = max(\exp(i\Delta x) - X, 0)$;

end

Generate a tridiagonal matrix A of dimension $(2I-1) * (2I-1)$,

with $A_{i,i} = b \forall i = 1, 2, \dots 2I-1$,$A_{i,i-1} = b \forall i = 2, \dots 2I-1$, $A_{i,i+1} = b \forall i = 1, 2, \dots 2I-2$.

for $j = N-1, N-2, \dots, 0$ **do**

  Generate a vector F of length $(2I-1)$, with $F_{2I-1} = c\exp(-r(T-j\Delta t))(S_{max} - X)$, $F_i = 0$

  otherwise;

  $U_j = A^{-1}(U_{j+1} + F)$;

end

$i_0 = floor\left(\dfrac{\ln S_0}{\Delta x}\right)$;

c_{IDS} is interpolated between $U_0^{i_0}$ and $U_0^{i_0+1}$;

For the European vanilla call option with strike price \$5, time to maturity of 1 year, current underlier price of \$5.25, volatility of 30%, risk free rate of 3%, dividend yield of 10%. Using a grid with values of x in the truncated domain $[-5, 5]$, with $N = 1500$ and I taking values from 100 to 1500 with increments of 100, the option value estimates are obtained as tabulated below:

I	N	Option price
100	1500	0.522776022894615
200	1500	0.523228505305835
300	1500	0.523105617704949
400	1500	0.522656345235669
500	1500	0.522797471560081
600	1500	0.522898874252548
700	1500	0.522950382492270
800	1500	0.522963742476891
900	1500	0.522959159142429
1000	1500	0.522890228302100
1100	1500	0.522914330062009
1200	1500	0.522936922834744
1300	1500	0.522947345908465
1400	1500	0.522949937081634
1500	1500	0.522947446794060

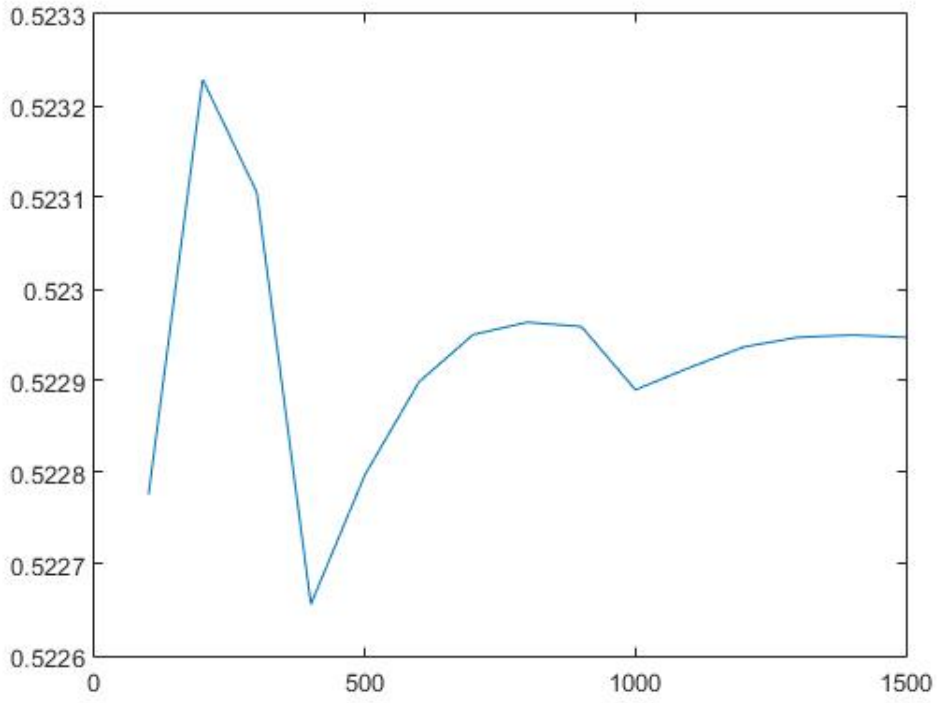

Figure 1: European vanilla call option value estimates against I with increments of 100

Since the value obtained at $I = 100$ is already quite close to the true value, it may be difficult to observe the convergence in the diagram of I going from 100 to 1500 with increments of 100 (figure 1). Hence, we plotted the diagram of I going from 100 to 1500 with increments of 20 in order to investigate further.

From figure 2, it is obvious that the option value converges to the true value at around \$0.52294 as the fluctations become smaller and smaller as I increases.

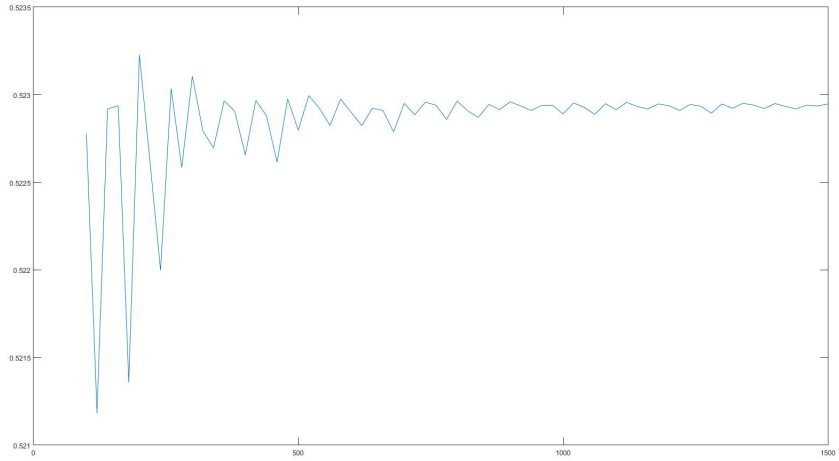

Figure 2: European vanilla call option value estimates against I with increments of 20

1.3 American vanilla call option using PSOR

For the American vanilla call option with same parameters as the above European vanilla call, we can use the projected SOR algorithm for the calculation of U_n from V_{n+1} with parameters $\epsilon = 1.0 * 10^{-6}, \omega = 1.3$.

I	N	Option price
100	1500	0.582849189993941
200	1500	0.583505358758593
300	1500	0.583457275858546
400	1500	0.582944640173874
500	1500	0.583166191075223
600	1500	0.583290840686203
700	1500	0.583345916640081
800	1500	0.583358092446549
900	1500	0.583351410333823
1000	1500	0.583260621906101
1100	1500	0.583286109957217
1200	1500	0.583301049237271
1300	1500	0.583302179614822
1400	1500	0.583294042928569
1500	1500	0.583280020736888

We have plotted the values estimated with I going from 100 to 1500 with increments of 100 in figure 3.

Similar to the previous section, we plotted another diagram of I going from 100 to 1500 with increments of 25 in order to investigate further as figure 4.

From figure 4, it is obvious that the option value converges to the true value at around \$0.5834 as the fluctations become smaller and smaller as I increases.

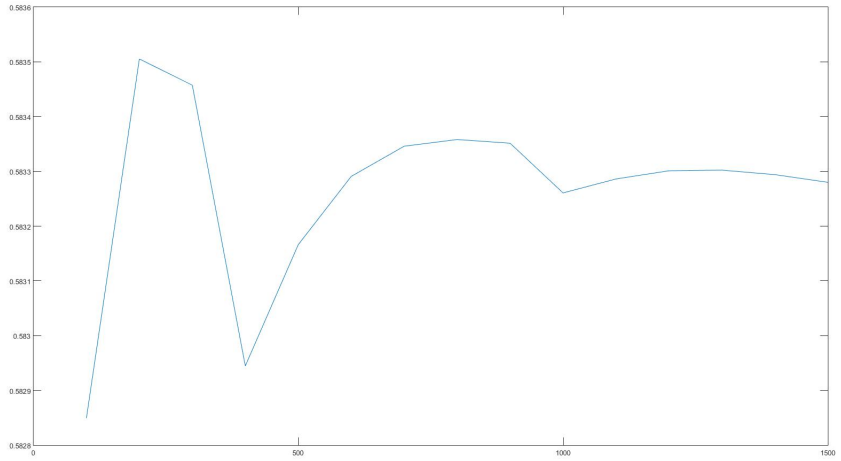

Figure 3: American vanilla call option value estimates against I with increments of 100

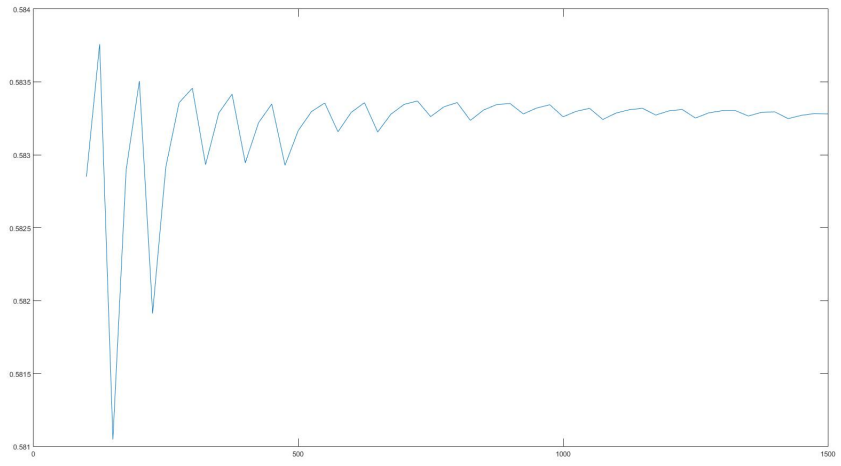

Figure 4: American vanilla call option value estimates against I with increments of 25

2 Valuation of digital call option

2.1 Pricing Formula of digital call option

First, we implemented the function `BS_DigitalCall(S0, X, r, q, T, sigma)` for pricing the digital call option given:

Data: S_0, X, r, q, T, σ

Result: c_{DC}, Option Premium

$x = \dfrac{\log\left(\frac{S_0}{X}\right) + (r - q - \frac{\sigma^2}{2})T}{\sigma\sqrt{T}}$;

$c_{\mathrm{DC}} = e^{-rT}N(x)$;

2.2 Implementation of Monte Carlo Simulation without control variate

Next up, we implemented the Monte Carlo simulation algorithm for estimating the 3-asset digital call option price, given the correlation matrix C and the number of samples generated N, as well as the other parameters, S_0, X, r, q, T, same as those used in pricing of Digital Call option.

Data: S_0, X, r, q, T, σ, C, N

Result: `MC_noCV`, Option Premium

First use the in-built function `randn` *in MATLAB to initiate a* $3 \times N$ *matrix* $\mathbf{R}$.

Each column of the matrix corresponds to the values of x *for the 3 assets in the portfolio.*

$\mathbf{R} =$ `randn(3, N)`;

Use Cholesky factorization to obtain a triangular matrix from the correlation matrix C

The in-build function `chol` *in MATLAB will lead to an upper triangular matrix.*

$\mathbf{L} =$ `chol`$(\mathbf{C})$;

$\mathbf{E} = \mathbf{R}^T\mathbf{L}$;

Construct the row matrix $\mathbf{p}$ *for expected price.*

for $j = 1, 2, 3$ **do**

  $\boldsymbol{\mu}_j = r - \mathbf{q}_j - \frac{\boldsymbol{\sigma}_j^2}{2}$;

  $\mathbf{p}_j = S_0 e^{\boldsymbol{\mu}_j T}$;

end

Replicate the row matrices $\boldsymbol{\sigma}$ *and* $\mathbf{p}$ *into* $N \times 3$ *matrices* $\mathbf{S}$ *and* $\mathbf{P}$

$\mathbf{S} =$ `repmat`$(\boldsymbol{\sigma}, N, 1)$;

$\mathbf{P} =$ `repmat`$(\mathbf{p}, N, 1)$;

Thence, obtain the simulated terminal prices for the three assets, in an $N \times 3$ *matrix* $\mathbf{S}_T$.

for $i = 1 \ldots N$ **do**

  for $j = 1, 2, 3$ **do**

    $w_{i,j} = \mathbf{S}_{i,j} \times \mathbf{E}_{i,j}$;

    $(\mathbf{S}_T)_{i,j} = \mathbf{p}_{i,j} \times \exp(w_{i,j}\sqrt{T})$

  end

end

Filter into a column matrix `max_prices` *with only the maximum terminal prices in each sample.*

`max_prices` $=$ `max`$(\mathbf{S}_T, [], 2)$;

Filter out those in `max_prices` *that are no more than strike* X *into a column matrix* `option_values`

`option_values` $= (\texttt{max\_prices} > X)$;

Finally, obtain the option value by discounting the average of the filtered terminal prices.

`MC_noCV` $=$ `mean`$(\texttt{option\_values}) \times e^{-rT}$;

2.3 Results from different number of price-path bundles and Comment

Using the set of parameters, with the number of price-path bundles at 100, 1000, 10000, and 100000, and each round running 30 times of simulations, we obtain the following results in one of runs:

X	N	Option price estimate	Standard errors
8.5	100	0.88325	0.027956
8.5	1000	0.87561	0.0095148
8.5	10000	0.87745	0.0028249
8.5	100000	0.87833	0.00081011
9.5	100	0.70923	0.053627
9.5	1000	0.71379	0.014814
9.5	10000	0.71402	0.0033345
9.5	100000	0.71369	0.0013023
10.5	100	0.51114	0.058064
10.5	1000	0.50082	0.013599
10.5	10000	0.50434	0.0038309
10.5	100000	0.50371	0.0013951

The option price estimate decreases for increase in value of X for same values of N. This is explained by the pricing formula of the digital call option, where the value of $\frac{\partial c}{\partial X}$ is negative.

From the option price estimate for different strike prices, we do not observe an explicit overall increasing or decreasing trend within the same strike price. However, as N gets larger, the change in estimated option price compared to the previous value of N gets to decrease.

This trend in the variance of the estimated values is more evidenced in the measurements of standard errors, which effectively decreased from smaller values of N to larger values within the same strike price. Each time when N increased by 10 times, the value of standard errors is lowered to approximately between $\frac{1}{4}$ to $\frac{1}{3}$ of the previous value, which is similar to the value of $\frac{1}{\sqrt{10}}$.

2.4 Variance Control and Effectiveness

2.4.1 Algorithm

- **Inputs:** S_0, X, r, q, T, σ, C, N
- **Output:** `MC_CV`, Option Premium
- **Initiation:**
  $F_1 = \texttt{BS\_DigitalCall}(S_0^1, X, r, q_1, T, \sigma_1)$;
  $F_2 = \texttt{BS\_DigitalCall}(S_0^2, X, r, q_2, T, \sigma_2)$;
  $F_3 = \texttt{BS\_DigitalCall}(S_0^3, X, r, q_3, T, \sigma_3)$;
  $\bar{F} = \frac{F_1 + F_2 + F_3}{3}$
- Pilot simulation using the same Monte Carlo simulation method but for only $\frac{1}{5}$ of original sample size, to obtain β. The results of interest are:
  - `pilotOptionValues`: the vector containing the simulated 3-asset digital call option values using samples in the pilot simulation;
  - `pilotBasketValues`: the vector of the mean digital call option value on each single asset in all samples in the pilot simulation.

- Hence with the result, we can obtain β by:
  - First obtain the covariance matrix between the option values and the basket values by $\mathbf{C}^* = \texttt{cov}(\texttt{pilotOptionValues}, \texttt{pilotBasketValues})$
  - We would have:
$$\mathbf{C}^* = \begin{bmatrix} \sigma_o^2 & \sigma_{o,b} \\ \sigma_{o,b} & \sigma_b^2 \end{bmatrix}$$
    where σ_o^2 and σ_b^2 are the variance of the option values and the basket values respectively, and $\sigma_{o,b}$ is the covariance between option values and basket values.
  - We hence could have an estimate for β to use:
$$\beta = \frac{\sigma_{o,b}}{\sigma_b^2}$$
- Followed that, we carry out the actual computation for the final option value using the β obtained.
- After obtaining the terminal stock price matrix $\mathbf{S}_T$, we also obtain the vectors for the 3-asset option values and the mean call option values for all samples.
  - `optionValues`: the vector containing the simulated 3-asset digital call option values in the full simulation;
  - `basketValues`: the vector containing the mean digital call option value on each single asset in all samples in the full simulation;
- Then we have the vector `controlledOptionValues` obtained as
$$\texttt{controlledOptionValues} = \texttt{optionValues} - \beta(\texttt{basketValues} - \bar{F})$$
- Taking the mean of the `controlledOptionValues` vector, we have the final option value
$$\texttt{MC\_CV} = \texttt{mean}(\texttt{controlledOptionValues})$$

2.4.2 Results

Using the set of parameters and the control variate algorithm, with the number of price-path bundles at 100, 1000, 10000, and 100000, and each round running 30 times of simulations, we obtain the following results in one of runs:

X	N	Option price estimate	Standard errors
8.5	100	0.88306	0.026009
8.5	1000	0.87657	0.0059658
8.5	10000	0.87712	0.0015879
8.5	100000	0.87799	0.00058616
9.5	100	0.71548	0.025147
9.5	1000	0.71364	0.0061404
9.5	10000	0.7145	0.0025493
9.5	100000	0.71406	0.0005965
10.5	100	0.49626	0.028081
10.5	1000	0.50429	0.0069935
10.5	10000	0.50457	0.002156
10.5	100000	0.50401	0.00039769

2.4.3 Comments

Using Monte Carlo simulation with control variate, we obtained results whose trends are to the previous runs with same set of parameters and without control variate. The similarity is in terms of:

- Option premium converges with less variation as N increases;
- The option premium decreases with increase in X;
- The standard error of the results decreases for increase in N within the same X.

However compared to the runs with the same X and same N without control variate, we can see that the option premium estimates are of similar values, and the standard errors decreases.

The following table summarizes the change in standard errors from methods with control variate and without control variate:

X	N	Standard errors WITHOUT control variate	Standard errors WITH control variate	Percentage of original
8.5	100	0.027956	0.026009	0.9304
8.5	1000	0.0095148	0.0059658	0.6270
8.5	10000	0.0028249	0.0015879	0.5621
8.5	100000	0.00081011	0.00058616	0.7235
9.5	100	0.053627	0.025147	0.4689
9.5	1000	0.014814	0.0061404	0.4144
9.5	10000	0.0033345	0.0025493	0.7645
9.5	100000	0.0013023	0.0005965	0.4580
10.5	100	0.058064	0.028081	0.4836
10.5	1000	0.013599	0.0069935	0.5142
10.5	10000	0.0038309	0.002156	0.5629
10.5	100000	0.0013951	0.00039769	0.2850

We can see evident reduction in the standard errors of the results across all set of values of X and N, when control variate is applied compared to when it is not.

From observation, the ρ, the estimated correlation coefficient between the control variate and the option value for different strike prices are as follows:

X	ρ	$\sqrt{1-\rho^2}$
8.5	0.68	0.7332
9.5	0.81	0.5864
10.5	0.88	0.4750

According to the lecture notes, the theoretical value of the standard error after control variate is applied would be reduced to $\sqrt{1-\rho^2}$ of that of the original values obtained without using control variate. From the data of standard errors before and after applying control variate, we could see the ratio of after-control-variate values to the original values for the same strike price, are clustering around the theoretical percentage predicted using the values of $\sqrt{1-\rho^2}$