David Šiška School of Mathematics Sciences, University of Edinburgh

2018/19 Semester 1

Object Oriented Programming with Applications

Big Project - Tuesday 13th November 2018[1]

1 Key Points:

i) Deadline: Monday 14th January 2019 (start of Week 1 / Semester 2) at *noon*.

ii) Two parts to submit:

a) Written report, main part no more that 5 pages A4 with at least 10pt font. Figures, Tables, Plots etc. *do not count for the page limit*. Appendix should include all your C# code and *does not count for the page limit*. A printed copy of this should be submitted to MTO (Room 5211, JCMB). An electronic version should be submitted via Learn.

b) Working implementation of the code submitted via a private GitHub repository (just like the assignments)

iii) Late submissions will follow standart university policy. See Section 28.1 in Taught Assesment Regulations, (link).

iv) The maximum number of marks for this 'Big Project' is 80 and the marking scheme is the following:

a) *Correctness* of the code (i.e. the code producing correct numerical answers for a range of standard inputs) and robustness (what the code does, how it throws exceptions / sensible error messages for non-standard inputs): *30 marks*.

b) *Written report* and code documentation (provide XML documentation for all your public classes and methods): *20 marks*.

c) *Code structure* (avoiding code duplication, sensible use of classes, sensible use of public / private methods and variables) and style (indentation, class, method and variable names etc.): *10 marks*.

d) *Exploration* If you wish to obtain a mark of over $60/80$ (i.e. over 75%) for this "big project" then you have to do something related to the project topic but that was not explicitly described here. The ideas that one can think of would include: pricing of other exotic derivatives, parallel implementation of the Monte-Carlo algorithm, variance reduction method in the Monte-Carlo algorithm, a finite-difference algorithm that prices derivatives, calculate the error in all Monte-Carlo simulations to a supplied confidence interval .... Of course you can do more than one.

v) Each task below is of roughly equal importance.

vi) This project is worth 80% of the final mark for the OOPA course.

[1] Last updated 13th November 2018

Strike K	Option exercise T	Price $C(0,S,v)$
100	1	
100	2	
100	3	
100	4	
100	15	

Table 1: Prices of "at the money" call options in the Heston model using Heston formula with parameters (1). Precision: two decimal places.

2 The Tasks

The theme of the project is the Heston stochastic volatility model. See [1] and references therein. The project will explore aspects of calibration and option pricing in the Heston model.

Please read all the tasks and think of how you will structure your code, projects and solution before you start. You need to submit *one* solution containing code solving *all* the tasks.

Task 2.1 (Code set-up)**.** Modify the Heston.cs class in the template solution to expose the required functionality. Please note you shouldn't implement your code in that class. You should only use the provided inputs to initialize the classes that you've designed, pass the arguments to them and return the outputs. We will use this class to run unit tests on your code.
Do not change class or method names, leave the method signatures intact and keep the namespace as it is. Please feel free to add a new project to the solution to implement your work, but do remember that you'll need to reference it in the project contianing Heston.cs to expose the relevant functionality.

Task 2.2 (Heston formula)**.** Write C# class for pricing European put and call options in the Heston model using the Heston formula, see [1] for details.

Your report should include Table 1 completed with values generated by your code for the Heston formula run with the following parameters:

$$\begin{aligned} &r = 2.5\%, \\ &\theta^* = 3.98\%, \quad \kappa^* = 157.68\%, \quad \sigma = 57.51\%, \quad \rho = -57.11\%, \quad v = 1.75\%, \\ &S = 100\,. \end{aligned} \tag{1}$$

Task 2.3 (Heston Call/Put with Monte Carlo)**.** Write C# class for pricing European put and call options in the Heston model using a Monte-Carlo algorithm, see [1] for details.

You probably want to have a class that generates the paths separate as this will be used in further tasks.

Your algorithm only needs to run if the Feller condition $2\kappa^*\theta^* > \sigma^2$ holds.

Your report should include Table 2 completed with values generated by your code for the Monte Carlo method (using at least 10^5 different paths generated with at least 365

Strike K	Option exercise T	Price $C(0, S, v)$
100	1	
100	2	
100	3	
100	4	
100	15	

Table 2: Prices of "at the money" call options in the Heston model using a Monte Carlo method with parameters (2). Precision: one decimal place.

time-steps per year) run with the parameters:

$$\begin{aligned} r &= 10\%, \\ \theta^* = 6\%, \quad \kappa^* &= 200\%, \quad \sigma = 40\%, \quad \rho = 50\%, \quad v = 4\%, \\ S &= 100\,. \end{aligned} \tag{2}$$

Task 2.4 (Checking Heston formula and Monte Carlo)**.** Experimentally (i.e. run the code) confirm that your option prices match between the Heston formula and Monte Carlo code. Plot convergence using log-log scale.

Explain your reasoning (why your experiments make sense) and include appropriate tables / figures in your report.

Task 2.5 (Calibration)**.** Use the BFGS algorithm (as implemented in Alglib and used in the last workshop) to create a "Heston calibrator" class capable of calibrating the Heston model to option prices observed in the market. This should be done by minimising mean square error between model prices and market prices. You can also use the objective function to penalise undesirable model parameters.

You should treat the initial asset price S and risk-free-rate r as given but the initial variance v as a model parameter. Thus your minimiser should minimise over $\kappa^*, \theta^*, \sigma, \rho$ and v.

Your calibrator class should have "guess parameters" as input for the starting point from which the minimisation starts. It should be possible to set accuracy and maximum number of iterations. Your calibrator should keep track of whether the minimisation method completed successfully or failed.

The calibrator class should use the Heston formula code from Task 2.2.

Your report should include the values of $\kappa^*, \theta^*, \sigma, \rho$ and v your calibrator reached when the accuracy is set to 10^{-3}, maximum iterations to 1000 when calibrated with $S = 100$, $r = 2.5\%$ and with market data from Table 3.

Hint. This will take a lot more time than calibrating the Vasicek model from the workshop. This is because you need numerical integration every time you price an option using the Heston formula. Each step in the minimisation algorithm will price the option at least six times.

Task 2.6 (Checking calibration)**.** Experimentally (i.e. run the code) confirm that your calibration method works. Explain, in the report, the steps taken. Provide tables etc. to document your checks.

There will be situations where the calibration procedure fails. Find and document some of these.

Strike K	Option exercise T	Market Call Price
80	1	25.72
90	1	18.93
80	2	30.49
100	2	19.36
100	1.5	16.58

Table 3: Observed market prices for an asset with underlying price 100 with risk-free-rate 2.5%.

Strike K	Option exercise T	$T_1, \ldots, T_M$	MC Price
100	1	$0.75, 1.00$	
100	2	$0.25, 0.50, 0.75, 1.00, 1.25, 1.50, 1.75$	
100	3	$1.00, 2.00, 3.00$	

Table 4: Prices of "at the money" Asian call options in the Heston model using a Monte Carlo method with parameters (2). Precision: one decimal place.

Task 2.7 (Pricing Asian arithmetic option)**.** Create a C# class for pricing Asian arithmetic call / put options using a Monte a Carlo algorithm. Your algorithm only needs to run if the Feller condition $2\kappa^*\theta^* > \sigma^2$ holds.

Such options have the following payoff: given a set of dates

$$T_1 \leq T_2 \leq \cdots \leq T_M \leq T$$

and a strike K the option payoff at time T is given by

$$\xi = \max\left(\frac{1}{M}\sum_{m=1}^{M} S(T_m) - K, 0\right)$$

for a call option while

$$\xi = \max\left(K - \frac{1}{M}\sum_{m=1}^{M} S(T_m), 0\right)$$

for a put option.

Your report should include Table 4 completed with values generated by your code (with at least 10^5 different paths generated with at least 365 time-steps per year) run with the parameters given in (2).

Hint. You should use the same code in Task 2.3, Task 2.7 and Task 2.8 for generating the sample paths.

Task 2.8 (Pricing lookback option)**.** Create a C# class for pricing a "lookback option" using a Monte a Carlo algorithm. Your algorithm only needs to run if the Feller condition $2\kappa^*\theta^* > \sigma^2$ holds.

Such options have the payoff:

$$\xi = S(T) - \inf_{0\leq t\leq T} S(t).$$

Your report should include Table 5 completed with values generated by your code (with at least 10^5 different paths generated with at least 365 time-steps per year) run with the parameters given in (2).

Option exercise T	MC Price
1	
3	
5	
7	
9	

Table 5: Prices of lookback options in the Heston model using a Monte Carlo method with parameters (2). Precision: one decimal place.

Hint. You should use the same code in Task 2.3, Task 2.7 and Task 2.8 for generating the sample paths.

3 Additional comments

You do not actually need to read all the details of [1], just find the relevant formulae there. These are in Section 2.4 (for Heston formula) and Section 3 (for Monte Carlo).

References

[1] Šiška, D. A note on the Heston model, *OOPA course website*.