MATH5350M: Computations in Finance Coursework 1 (30%): Due Thursday 26th February at 14:00

Nadhir Ben Rached (n.benrached@leeds.ac.uk)
University of Leeds, 2026

Your work should be submitted on Minerva before 14:00 on Thursday 26th February. Late submissions will be penalized 5% for each day that they are late. Work that is submitted more than a week late will get zero marks.

Further details on how to submit are given below.

In this coursework, you will first implement the binomial model as it is described in Algorithm 1.2 and use it to compute the price of a European put option. The underlying security of the option is a stock which follows the Black–Scholes model with risk-free rate r and volatility σ.

You will then explore the computational complexity of several different algorithms to compute the binomial model. Write your programs in python and please use the function names and arguments as suggested in the detailed descriptions below. You should write a report with the answers to the questions in Parts 3, 4, 5, 6 and 7.

1 Inputting data (15 marks)

Implement a function responsible for inputting data. This function should be informative, user friendly and must verify correctness of arguments. You are expected to deal with situations when an inputted parameter is clearly wrong: for example, when the stock price is negative or a letter, not a number. Please use the function name and arguments:

```
input_data(S_0,r,sigma,T,K,M)
```

The meaning of parameters is as follows:

- `S_0` – initial price of the stock, i.e. S_0,
- `r` – risk-free interest rate r (per annum),
- `sigma` – the volatility of the stock price σ (per annum),
- `T` – expiry time T of the option,
- `K` – strike price K of the option,
- `M` – number of periods in the binomial tree, M.

2 The binomial model (15 marks)

Implement the pricing of a vanilla European put option on a recombining binomial tree. Base your program as closely as you can to Algorithm 1.2 in the notes and store the option price at all nodes of the tree. Use the approximate CRR method for the calibration of the binomial model. The computed price should be returned by the function. Please use the function name and arguments:

```
compute_binomial_algorithm1_2(S_0,r,sigma,T,K,M)
```

The parameters have the same meaning as in Part 1. This function should be standalone (ready to use in other programmes without your function `input_data`), so it must perform its own verification of parameters. It does not mean that you are relieved from checking correctness of parameters in the function `input_data`! The above function must not expect input from the user and it should not print anything on the screen, except for errors, e.g. because the parameters have not be chosen appropriately.

3 Verification (15 marks)

In your report, state the price as computed by your program of a European put option with strike price £90. The option matures in 18 months. The underlying stock costs £80 now and has a volatility of 30% p.a. The risk-free rate is 2.5% p.a. Use a tree with $M = 50$ periods to compute the price.

Compare the result of your program with the price computed using the exact formula (as provided in the lecture notes). Explain how you evaluate the Φ function in the exact formula.

Convince yourself (as well as you can) that your program is correct. Document your verification process so that your hypothetical manager may be sure that she can rely on your code. Your documentation must be concise (don't produce 20 screenshots) but detailed enough to give your manager confidence.

4 Computational complexity (15 marks)

As in Part 3, use the strike price £90, option maturity time 18 months, current stock price £80, volatility of 30% p.a and risk-free rate 2.5% p.a. Use the `perf_counter()` function from the `time` module to time your code for different numbers of periods M. For example, you can time your function in the following way:

```
time_start = time.perf_counter()
compute_binomial_algorithm12(S_0,r,sigma,T,K,M)
time_end = time.perf_counter()
totaltime = time_end - time_start
```

Plot a graph with the number of periods on the horizontal axis and the computation time on the vertical axis using the `loglog` plot command in `pyplot` module in `matplotlib`. What is the slope of the line in the `loglog` plot? What does this tell you about the computational complexity of the algorithm? (Tip: you may find it helpful to write a function that creates the `loglog` plot and computes the slope.)

5 Faster algorithms (15 marks)

Now consider how to improve Algorithm 1.2. Write a function to test whether using a single list (or array) to store the option prices speeds up the function. To do this, adjust Algorithm 1.2 as described in Exercise 1.2(b). Please use the function name and arguments:

```
compute_binomial_single_array(S_0,r,sigma,T,K,M)
```

Plot the computational time for this algorithm, as in Part 4, and compare to the results of `compute_binomial_algorithm12`. Does using a single list (or array) make much difference?

Investigate other potential improvements to Algorithm 1.2 (ideally at least one). For example, can you avoid using a nested loop? Is there a formula you can use to compute the option price? How does this perform compared to the other algorithms? Compare your algorithms using plots of the computational time, as in Part 4 and discuss why each has the computational complexity suggested by your plots.

In each case, you should write your algorithms as functions with the same arguments as `compute_binomial_algorithm12` and check that they produce the same option price. Give your functions appropriate names and refer to them in your report.

6 Tilted tree and Richardson Extrapolation (15 marks)

Plot the option price as a function of M with and without a tilted tree. Explain the improvement you get using a tilted tree. Are the computed prices using a titled tree more accurate? Show how Richardson extrapolation can improve the accuracy. Does it matter whether M is even or odd for the accuracy to get improved? To study the accuracy between different methods, plot the absolute option price error as a function of M.

7 Results and discussion (10 marks)

Summarise your findings from Parts 3 to 6 and discuss the strengths and limitations of the algorithms you have investigated.

8 Directions for submission

You need to submit a report and the code.

Report: Your report must be typed in Microsoft Word or Latex. Do not include the code in your report. Structure your report as the assignment (Part 3, Part 4, Part 5, Part 6, Part 7), leaving out Part 1 and Part 2 which do not require an answer. Attach the **Academic Integrity Form** to your report.

Your report (as a PDF file) must be uploaded in the Submit My Work area using the TurnItIn submission tool entitled *"Report submission for Coursework 1"*.

Code: Before submission, clean up the code and check that the programme runs. Make sure you have not shared your code or a piece of thereof with anyone else as this qualifies as plagiarism.

The python file `cw1-studentnumber.py` (replace `studentnumber` with your student number; NO pdf, doc, ps, etc.) must be uploaded in the Submit My Work area using the assignment submission tool entitled *"Program submission for Coursework 1"*

Marking: The following aspects of your report are important in the marking: quality of the answers, presentation (easy to understand, looking professional, quality and labelling of figures), relevance, conciseness (excluding any figures, you can do good work in two pages of text, five is probably more than necessary; think carefully about what figures you include and how you present your results).

The following aspects of your programme are important in the marking: correctness (the programme does what it is supposed to do), quality of code (e.g., meaningful names for variables, comments that facilitate understanding of the code) and efficiency (speed and memory usage).

Deadline: Both the report and the code have to be submitted by the deadline. Please ensure that you leave sufficient time to complete the online submission process, as upload times can vary. It is your responsibility to ensure you upload the correct files to Minerva, and that it has uploaded successfully.

9 Python code suggestions

- Python has several different ways that programs can handle errors, for example `try`, `except`, `raise Exception` and `assert`.
- You can time bits of python code using the `time` module:

  ```
  import time
  start_time=time.perf_counter()
  # Put the code you want to time here
  end_time=time.perf_counter()
  total_time=end_time-start_time
  ```

- The `stats` module of `scipy` has the cumulative distribution function of the normal distribution, `norm.cdf`.
- The `special` module of `scipy` has a function to compute the binomial coefficients:

  ```
  from scipy.special import binom
  ```

  However, some binomial coefficients can get very large so you should investigate whether this function will work for the values you need. Can you think of any other way to compute the binomial coefficients? (Hint: if you log a factorial, it becomes a sum, and you often want to multiply the binomial coefficients with something small.)
- The `numpy` module has a function called `roll` that permutes the elements of an array in a cycle. How could you use this?
- You can do arithmetic on `numpy` arrays, which is often much quicker than using a `for` loop.
- You can pass function names as arguments in python.