Simulating Jump-Diffusion Processes

1 Instructions

Project 1 is due on September 6th by 10:00 pm. This is a hard deadline, so no exceptions. You must push your local repository back to GitHub before the deadline. Your repository must contain:

- The Matlab code you used to complete the project;
- A script named `main.m` file that generates all required plots;
- A `report.pdf` file with your answers to the project questions.
- All plots in the report must be self-contained. Self-contained means that a reader who only sees your figure (image and caption, but not the surrounding text) can understand what you are plotting. This translates to all plots having axis titles, correct units on the axis, and a caption that summarizes what is plotted.

If you are familiar with Git and Github, you can **go to this link** to obtain the repository for this project, otherwise read the step-by-step guide on the next section.

2 Obtaining the Project Repository on Github

This section will guide you on how to use Git and GitHub to clone the project repository to your computer. The repository you will clone has a skeleton of how the project should be structured. This structure is also discussed below.

2.1 Create a Folder for Projects and Data

Open your terminal application (or git bash) and change directory (cd) to your home folder.

```
cd ~/
```

The `~` at the beginning is interpreted by the terminal as the path to your home folder. In my case, this is the same as typing `/Users/guilhermesalome`. (If you type `echo $HOME` you should get back the path to your home folder.)

Now, make a new directory (mkdir) to hold the solution to the projects:

```
mkdir Projects
```

Move inside this folder and create a new folder that will hold all the data you will use in the projects.

```
cd ~/Projects/
mkdir Data
```

You should save the data you were assigned to inside the `Data/` folder.

2.2 Cloning the Repository

Follow the next steps to get the repository for Project 1. Read all of the steps before beginning.

1. Go to GitHub and create a new account if you do not already have one.
2. Click on **this link** to be directed to Github. The following should happen:
   (a) When you click on the link, you will be asked to login with your GitHub account.
   (b) You will then see a page with a list of names (Last Name, First Name). Click on yours. This will link your Github account to your name.
   (c) Then you will be asked to accept the assignment. When you do, GitHub will create a new repository on your account. This is the repository you will use to submit your solution to this project.
   (d) After Github is done creating the repository, you will get a link to your assignment. Follow that link.
   (e) Lastly, click on the green button `Clone or download` and copy the URL to the repository.
3. Use the `URL` to clone the repository to your computer:

```
cd ~/Projects/
git clone URL
```

The command will clone the repository from Github to your computer. It will create a new folder with the project name, in this case `project-1`, and download all the files to that folder.

Now, if you list files (ls) in your directory:

```
ls
```

You should see a new folder for this project. Inside that folder you will see several empty files and folders. These are discussed next.

2.3 Folder Structure

The folder you just cloned has the following structure:

```
project-1/
|
|-- report.tex
|-- preamble.tex
|-- report.pdf
|-- figures/
|-- functions/
|-- scripts/
|   |-- main.m
```

Let's go over each item:

- `report.tex` and `preamble.tex` are the tex files that should generate your `report.pdf` file with the solutions to this project;

- `figures/`: folder that contains all the figures used in your report
- `functions/`: folder with all Matlab functions you created for solving this project
- `scripts/`: folder with all Matlab scripts you created for solving this project. Notice the file `main.m`, which should be a script that (ideally) generates all of the solutions for this project.

The purpose of the `functions/` folder is to break the solutions to the individual exercises into smaller pieces, which is necessary for simplifying your code and making it more readable/easier to debug. The `scripts/` folder has a similar purpose to `functions/`, but the files in it should break down `main.m` into separate parts (i.e., one file for each problem). For most projects, the `scripts/` folder will usually contain scripts that generate plots/figures. The code to generate plots in Matlab can easily become big and repetitive. So, the idea is to move that code into scripts, say `exercise1A.m`, so that we can simply call this script in the right place of `main.m` and have the plot be created. This also helps separating the computational part of the project from the plotting counterpart. Lastly, the `figures/` folder should contain the figures you were asked to create in this project. These figures will be used to construct your report, and this folder provides a clean way to keep them separate from everything else.

2.4 On Making Mistakes and Debugging

The lectures are responsible for discussing the theory, but the projects are meant for you to implement that theory and see it in practice. Implementing the theory means creating software to compute the estimates we are interested in, and then analyzing the results.

The process of writing the software can be divided in two stages:

- Knowing what we want the computer to do
- Telling the computer what to do

The first stage is knowing what to compute, which is given by the theory we are studying. The second stage is about transforming that theory into computer instructions. It is in this step that most mistakes occur, and we always make mistakes when programming.

> If debugging is the process of removing software bugs, then programming must be the process of putting them in. – E. W. Dijkstra

The easiest mistakes to spot are syntax mistakes. They are easy to spot because you will get an error when you run your code, and so are also easy to fix.

The hardest mistakes to identify are logical mistakes. These happen when you think your code is doing something, but your instructions are telling it to actually do something else. When this happens, you will end up computing wrong values, and, because there is no easy way to identify these mistakes, you will spend most of your time fixing these types of errors.

Your job is to find your own logical mistakes and fix them. Do not worry if you do not succeed at first, it is normal to make mistakes when coding. Look at your code and understand what is actually doing (debug, go line by line). Then fix it and try again.

3 Questions

The purpose of this project is to get a better understanding of Jump-Diffusion processes. In this project you will learn how to simulate specific versions of a Jump-Diffusion process. You will simulate both the diffusion and jump parts of a Jump-Diffusion process, under constant and stochastic volatility.

Exercise 1

The objective of this exercise is to simulate a Gaussian diffusion model with constant coefficients.

Let X denote the process for the log-price of an asset. Then, we want to simulate the model:

$$dX_t = \mu dt + \sqrt{c} dW_t$$

Assume that:

$$\begin{aligned} n &= 80 \\ T &= 1.25 \times 252 \\ \mu &= 0.03782\% \\ \sqrt{c} \equiv \sigma &= 0.011 \\ X_0 &= \log 292.58 \end{aligned}$$

A.

Interpret the meaning of the parameters n, T, μ and σ. How does μ relate to the market index returns (hint: there are 252 business days in a typical year)? What about σ?

B.

Simulate the model.

Remember, to simulate the diffusion you have to compute:

$$X_i = X_{i-1} + \mu\Delta_n + \sigma\sqrt{\Delta_n} Z_i \text{ for } i = 1, 2, \ldots, nT$$

Convert the simulated log-prices to prices:

$$P_i = \mathrm{e}^{X_i} \text{ for } i = 0, 1, 2, \ldots, nT$$

Plot the time series of the prices. Interpret the plot.

Exercise 2

The objective of this exercise is to simulate a compound Poisson process. Consider the parameters:

$$
\begin{aligned}
n &= 80 \\
T &= 1.25 \times 252 \\
\sigma &= 0.011 \\
\lambda &= \frac{15}{252}
\end{aligned}
$$

Assume the jump sizes are drawn from a normal distribution: $Y_k \overset{d}{\sim} \mathcal{N}\left(0, \sigma_j^2\right)$ for $k = 1, 2, \cdots, N$. Also, consider:

$$\sigma_j = 18 \times \sqrt{\sigma^2 / n}$$

A.

Interpret the parameters λ and σ_j. How does σ_j compare to σ? Why is there an n in the denominator of σ_j?

B.

Simulate the compound Poisson process:

$$J_i = \sum_{k=1}^{N} 1_{\left\{U_k \leq \frac{i}{nT}\right\}} Y_k \text{ for } i = 0, 1, 2, \ldots, nT$$

Make a time-series plot of the simulated compound Poisson process. Interpret the plot.

Exercise 3

Consider a jump-diffusion model with constant coefficients:

$$dX_t = \mu dt + \sqrt{c} dW_t + dJ_t$$

On this exercise you will simulate this process.

A.

By combining the simulations from Exercise 1 and Exercise 2, you can obtain a simulation for the jump-diffusion model with constant coefficients.

Let $\tilde{X}_i$ denote the simulations from Exercise 1, and J_i the simulations from Exercise 2. How can you combine the results from the previous two exercises? Consider the alternatives:

1. $X_i = \tilde{X}_i + J_i$
2. $X_i = \tilde{X}_i + \mathrm{e}^{J_i}$
3. $\mathrm{e}^{X_i} = \mathrm{e}^{\tilde{X}_i \times J_i}$
4. $\mathrm{e}^{X_i} = \mathrm{e}^{\tilde{X}_i + J_i}$

Which alternatives (if any) are correct and why?

B.

Combine the simulations from the previous exercises, convert the values from log-prices to prices, and create a time-series plot. Interpret the plot.

Exercise 4

The objective of this exercise is to simulate from a jump-diffusion model with stochastic volatility.

Let X denote the process for the log-price of an asset. We want to simulate:

$$\begin{aligned} dX_t &= \sqrt{c_t} dW_t \\ dc_t &= \rho(\mu_c - c_t)dt + \sigma_c \sqrt{c_t} dW_t^c \end{aligned}$$

where W and W^c are independent.

Assume that:

$$\begin{aligned} n &= 80 \\ T &= 1.25 \times 252 \\ n_E &= 20 \times n \\ \rho &= 0.03 \\ \mu_c &= 0.011^2 \\ \sigma_c &= 0.001 \\ c_0 &= \mu_c \\ X_0 &= \log 292.58 \end{aligned}$$

A.

Interpret the parameters ρ, μ_c and σ_c.

B.

Simulate the stochastic variance process c_t. An Euler discretization scheme is now necessary to reduce the discretization error.

To simulate the process of c_t you have to compute:

$$c_j = c_{j-1} + \rho(\mu_c - c_{j-1})\delta_E + \sigma_c \sqrt{c_{j-1}\delta_E}\tilde{Z}_j \text{ for } j = 1, 2, \ldots, n_E T$$

We also employ an additional rule. If the value of c_j for some j gets too small, we will reflect it to a bigger value. That is, if $c_j < \frac{\mu_c}{2}$ for some j, make it equal to $\frac{\mu_c}{2}$ instead.

Plot the ultra high-frequency values of c_j for $j = 0, 1, 2, \ldots, n_E T$. Interpret the plot.

C.

Generate the ultra high frequency log-prices X_j for $j = 0, 1, 2, \ldots, n_E T$.

Remember, to simulate the log-prices when the variance is stochastic we compute:

$$X_j \equiv X_{j-1} + \sqrt{c_{j-1}\delta_E} Z_j \text{ for } j = 1, 2, \ldots, n_E T$$

Convert the log-prices to prices and make a time series plot. Interpret the plot.

D.

Use the simulated log-prices to compute the log-returns $\Delta_j^{n_E} X$ for $j = 1, 2, \ldots, n_E T$. Create a time-series plot the ultra high frequency returns. Does the return volatility display a pattern?

E.

Sample the simulated log-prices X_i at a coarser frequency. Compute the log-returns for $i = 1, 2, \ldots, nT$. Plot the returns sampled at the coarser frequency. Does the volatility pattern remain?

F. (Optional, PhD required)

What happens to the stochastic volatility when the value of ρ increases? Plot the same sample path for increasing values of ρ.

G. (Optional, PhD required)

What happens to the stochastic volatility when σ_c increases? Plot the same sample path for increasing values of σ.