Hand-in Assignment 2

Quantitative Methods in Fixed Income, Michel vd Wel, Erasmus U. Rotterdam

IMPORTANT INFORMATION

Below is the second hand-in assignment. At most 5 points can be earned. You will receive a grade on a scale of 1-5 (0 if you hand in nothing or in case of plagiarism), which will be added to the points of the first hand-in assignment and the exam to get your course grade (divided by 100). The assignment is a CodeGrade assignment, available through Canvas, with deadline of Friday December 8 at 23:59 (1 minute before midnight). Make the assignment on your own; there will be plagiary scan. This assignment is a lot less work than the first hand-in assignment, but very important as well since it deals with the practical and applied topic of pricing a derivative using simulation. Start early and don't underestimate it! You can start making this assignment immediately after lecture 9 so have almost nine working days time to make it.

ASSIGNMENT

In this assignment we simulate the price of a European call bond option, for a setting where the underlying bond is modelled using a 1-factor Vasicek model. It is your goal to write two functions for this simulation. Input variables for the pricing problem are:

- Strike price K
- Expiry of bond option T_0
- Maturity of bond T_1
- Short-rate speed of mean reversion κ
- Long-run mean short-rate μ
- Short-rate volatility σ
- Number of replications in simulation study R
- Euler step-length Δ

In terms of timing, at the time of expiry of the bond, given by T_0, the time to maturity of the bond will be $T_1 - T_0$. The bond price is modelled using the 1-factor Vasicek model, for which a closed-form solution exists. Hence, the short-rate needs to be simulated only through T_0 to calculate both the discounting as well as the short-rate at this time as input for the bond pricing function. Simulate under the risk-neutral measure (thus take $\lambda = 0$) and initialize with $r_0 = \mu$.

YOUR CODE

You can make this assignment with either Octave (the free version of Matlab), Python or R. There are two functions that need to be programmed. First, because there is an exact solution for the one-factor Vasicek model, a function that returns the price of a zero-coupon bond given input values of κ, μ, σ, r_t and τ. These should be, in the given order, the arguments of the function.

Use the closed-form solution of slide 21 from lecture 6. Name this function `get_vasicek_price`. Second, a function that returns the simulated bond option price given input values of κ, μ, σ, r_0, R, Δ, T_0, T_1 and K (make this, in the given order, arguments of the function). (Note that in the assignment we have $r_0 = \mu$, but it is good to program this function as general as possible. Do not hard-code this relation.) Assume T_0 and T_1 are always given in full years. In Python and R name this function `main`, and in Octave/Matlab name it `getSimBondOptionPrice`. Make sure you call the Vasicek bond price function in the bond option price calculation code. The bond price function is worth 1 point and the bond option price function 3 points for the grading of the assignment. In addition, 1 points is earned for the quality of your coding; for which CodeGrade will provide detailed feedback.

We will now structure the bond option price calculation code a bit such that, given a certain random seed set in CodeGrade, only one price will be provided by your code and your assignment can thus be easily graded. Use a two for-loop structure as in the examples from slides 14 and 19 of lecture 9. Note that other ways of implementation are possible, but that has the risk of getting different random numbers used and thus a different answer. To calculate the simulated short rate, use the following random number generator functions to obtain scalar random numbers:

- Octave/Matlab: `randn`
- Python: `numpy.random.normal(0,1)`
- R: `rnorm(1)`

We are thus drawing from the standard normal distribution, which can be multiplied by the standard deviation to get draws from a normal distribution with the right variance.

Program the functions yourself first on your own computer. Make sure to use the exact names for each of the functions, with arguments in the given order! Then, after you are happy with the result, you can simply drag and drop the files to the CodeGrade environment. Immediately after the assignment is available, you can in fact already upload the files and test them on CodeGrade. Using some pre-set input values, CodeGrade then checks whether the function returns the right value and if it doesn't provide an error. It is strongly advised to test your code! You can upload files as many times as you want. The files last-uploaded before the deadline will be used for grading the assignment. If you are working in Octave/Matlab, give the file for each function the name of that function. For Python and R, give the file with both functions the name "simbondoption".