Columbia University
MATH GR5260 Spring 2022
Programming for Quant and Computational Finance
Ka Yi Ng

Exam 1
Mar 11th 2022 (Fri) 8:15pm – 9:45pm

GUIDELINES

This is an open book exam. You may use any notes, reference materials, internet, Jupyter Notebook or any Python IDEs during the exam period. However, you only have limited time to complete the exam, so you must use your time wisely.

All answers must be written legibly.
In the Python code, add comments to your code where you think will help grader understand your logic.

Solution submission

- Answer the theory questions on the exam paper provided.
- Submit your python code (and output results) as a Jupyter notebook or HTML or pdf files onto Courseworks under 'Exam 1' section, in the same way you uploaded your homework previously. If you have technical issues in submitting your solution, please notify your TAs or Prof Ng and email your solution directly to your TAs to get a proper timestamp of submission.
- You may name these files whichever way you like.
- Make sure that in each of these files, you have your name and UNI on it.

During the exam, if you suspect of any typos or have questions, notify your TAs or Prof Ng.
Prof Ng: kyn@math.columbia.edu
Zheng Sun: zs2413@columbia.edu
Ying Wu: yw3521@columbia.edu

Question	1	2	3	Total
Points	40	15	10	65

HONOR CODE AFFIRMATION

I affirm that I will not plagiarize, use unauthorized materials, or give or receive illegitimate help on assignments, papers, or examinations. I will also uphold equity and honesty in the evaluation of my work and the work of others. I do so to sustain a community built around this Code of Honor.
(https://www.college.columbia.edu/honorcode#:~:text=)

First name	Last name	UNI	Signature

QUESTION 1 (40 POINTS)

This question is about applying Monte Carlo simulation to the valuation of a call option on the spread of two commodity prices (Brent crude vs WTI crude). Spread options were discussed in class and were covered in homework assignment 2.

Suppose that the two commodity prices $S_1(t)$ and $S_2(t)$ follow a 2-dimensional correlated Geometric Brownian Motion (GBM). That is, $dS_1(t) = S_1(t)(\mu_1 dt + \sigma_1 dW_1)$ and $dS_2(t) = S_2(t)(\mu_2 dt + \sigma_2 dW_2)$ where $W_1(t)$ and $W_2(t)$ are correlated standard Brownian Motions with constant correlation ρ.

The present value of the option can be written as follows:

$$V = E[e^{-rT}\max(S_1(T) - S_2(T) - K, 0)]$$

Here T is the time to option expiry, K is the strike of the option and r is the zero coupon rate for the time to option expiry.

Let's assume that the drift and volatility parameters in the GBM and the zero rate r are known constants.

a) Write down the expression for $S_1(t)$ and $S_2(t)$ in terms of their initial prices, drift, volatility parameters and the bivariate standard normal random variable (Z, W) with correlation ρ.

Answer

Suppose $(z_1, w_1), (z_2, w_2), \dots, (z_N, w_N)$ are independent random samples of (Z, W).

b) Write down an expression for the simulated price V_N of the option in terms of $(z_1, w_1), (z_2, w_2), \dots, (z_N, w_N)$ of (Z, W). Explain why the estimator is unbiased.

Answer

c) As N goes to infinity, the theoretical value V of the option will fall into an interval $(V_N - \varepsilon_N, V_N + \varepsilon_N)$ with 95% confidence where ε_N is expressed in terms of $(z_1, w_1), (z_2, w_2), \ldots, (z_N, w_N)$. Write down the expression for ε_N.

Answer

d) Write a Python function to compute the simulated price V_N of a spread call option. The input parameters shall include: the option strike, time to option expiry and any market data required. The number of simulations N shall be made as an input. The function shall return **both** the simulated price and the 95% confidence interval. (You may use the input parameters without validating them within the function)

e) Write a Python function that uses antithetical sampling technique to compute and return the simulated price $\tilde{V}_N$ of the spread call option. Again you may use the function inputs without validating them within the function. (It's not required to return the 95% confidence interval, but feel free to do so if you want.)

Trader Joe explains:

"In antithetical sampling, if a random sample (z, w) is drawn, we compute the simulated payoff as the average of two payoffs, one for (z, w) and one for $(-z, -w)$. I think we will achieve even better results if we also use $(-z, w)$ and $(z, -w)$. That is, for each random sample, we compute the simulated payoff as the average of four payoffs, one for (z, w), one for $(-z, w)$, one for $(z, -w)$ and one for $(-z, -w)$. And the estimate $\hat{V}_N$ for the option price becomes the average of the N simulated discounted payoffs."

f) Write a Python function that uses Joe's method to compute and return the simulated price $\hat{V}_N$ of the spread call option. The function shall have the same inputs as in part d) or e).

Consider this spread call option example: $S_1(0) = 98$, $S_2(0) = 92$, $\mu_1 = 2\%$, $\mu_2 = 1.5\%$, $\mathrm{K} = 7$, $\sigma_1 = \sigma_2 = 20\%$, $r = 0.05\%$, $\rho = 70\%$, $T = 1/12$

g) Use your functions to compute the estimates V_n, $\tilde{V}_n$, $\hat{V}_N$ for the option using n=200000 simulations.

h) Do you think that Joe's method will give a better convergence than the antithetical method? Explain why you think so.

Answer

QUESTION 2 (15 POINTS)

This question is about using Pandas to handle P&L calculations for the equity transactions in a file.

Download the .csv files from Courseworks under the Assignment section, 'Exam 1'.

The file eq_trans.csv consists of a list of equity transactions for 2 customers.

TradeID	TradeDate	Sector	Ticker	Shares	TradePrice	Customer
200171	2022-02-01	Financials	BAC	5000	47.0	CUST_A
200172	2022-02-01	HealthCare	MRK	5000	82.0	CUST_B
200173	2022-02-01	HealthCare	JNJ	4500	169.0	CUST_A

Header descriptions:
TradeDate: date on which the transaction is traded
Ticker: ticker symbol for the company
Sector: the sector which the company belongs to.
Shares: number of shares purchased or sold. Positive number refers to a long position and negative number refers to a short position
TradePrice: price at which the shares are purchased or sold
Customer: alias name for the customer

The file eq_prices.csv keeps the adjusted closing prices of the stocks in February.

Date	AAPL	AXP	BAC	CME	CSCO	JNJ	MRK	NVDA	PFE	XOM
2022-02-01	174.39	183.54	46.94	232.02	55.37	169.79	81.92	246.38	53.07	79.94
2022-02-02	175.62	184.16	46.89	239.37	56.17	171.67	82.01	252.42	53.86	79.74
2022-02-03	172.68	184.04	46.43	241.07	55.20	171.66	79.01	239.48	53.38	78.81

a) Write a Python program that reads the given input files and displays a DataFrame that keeps the End-of-day value, daily PnL and total PnL as of Feb 28, 2022 for **all transactions**.

The EOD value of an equity transaction on a date T is defined as:

(number of shares) x (adjusted closing price on date T)

The daily PnL of an equity transaction on a date T is defined as:

(EOD value on date T) – (EOD value on date T-1)

The total PnL of an equity transaction on a date T is defined as:

(number of shares) x (adjusted closing price on date T - TradePrice)

Here we assume zero transaction cost and date T-1 means the previous business day.

The format of the DataFrame should look something like this.

TradeID	TradeDate	Sector	Ticker	Shares	TradePrice	Customer	EOD value	Daily_PnL	Total_PnL
200171	2022-02-01	Financials	BAC	5000	47.0	CUST_A	221000.0	-4100.0	-14000.0
200172	2022-02-01	HealthCare	MRK	5000	82.0	CUST_B	382900.0	1300.0	-27100.0
200173	2022-02-01	HealthCare	JNJ	4500	169.0	CUST_A	740565.0	-6435.0	-19935.0

b) Provide an aggregated view of the EOD value, daily PnL and total PnL by customer by ticker. The view should look something like this.

Customer	Ticker	Shares	EOD value	Daily_PnL	Total_PnL
CUST_A	AXP	2500	486350.0	2075.0	23850.0
	BAC	5000	221000.0	-4100.0	-14000.0

QUESTION 3 (10 POINTS)

The duration formula for a bond that has n remaining coupon payments is given as follows:

$$D = \sum_{i=1}^{n} t_i \left[\frac{CF_i e^{-yt_i}}{B} \right]$$

Where: B is the bond price

y is the yield to maturity (continuously compounded)

CF_i is the bond cashflow for the ith remaining coupon period

t_i is the time in years from value date to the ith coupon date (assuming 365 days in a year)

Below is a class that provides bond calculations.

```
class Bond:
    def __init__(self, start, maturity, coupon, freq):
        self.start = start    # start date of bond issue
        self.end = maturity   # maturity date of bond issue
        self.coupon = coupon  # annualized coupon rate in percents (eg. 5)
        self.freq = freq      # no. of periods per year (eg. 1: annual, 2: semi-annual)
```

Two class methods are already defined and available in this class.

```
def cc_yield(self, value_date, price):
    # returns yield to maturity (continuously compounded) for a given bond price input
    # input price: in percents (eg. 101.05)
    # output yield: in decimals (eg. 0.045)
```

```
def period_dates(self, value_date):
    # returns a numpy array of remaining coupon period dates after value_date
    # example:
    #    for a bond that matures on 2022-12-10, paying coupons semi-annually
    #    if value_date is '2022-02-15', then
    #    period_dates(value_date) returns an ndarray of ['2022-06-10','2022-12-10']
```

A function to compute the number of calendar days between two dates is also defined and available.

```
def days(start, end):
    # returns the number of calendar days between start and end date inputs
```

Use the above defined class methods and function to write a class method (called **duration**) for the Bond class to compute and return the duration of a bond given an input price quoted in percents. The inputs for the method are: **value_date** and **price**.

Note: you **don't need to run** this piece of code as the Bond class is not explicitly provided to you.

--end of exam--
--Happy spring break--