FE522 homework 1

Yuwen Jin

March 2020

1 Problem 1

For this problem, the 5 distributions I choose are: Normal distribution; Poisson distribution; Chi-squared distribution; Gamma distribution and Bernoulli distribution. Here is the parameters I set for each distribution to get random numbers.

```
// Problem 1
// Generate 5 random number distributions
vector<double> normal = normalDist( mean: 10,  sd: 5,  length: 10000);
vector<double> poisson = poissonDist( k: 5,  length: 10000);
vector<double> chi_squared = chisquaredDist( k: 5,  length: 10000);
vector<double> gamma = gammaDist( mean: 10,  sd: 5,  length: 10000);
vector<double> bernoulli = bernoulliDist( p: 0.5,  length: 10000);
```

Figure 1: Parameter setting for Problem 1

And for output path, I'm not sure if it's better to put outputs of all problems into a single file, or separately. Finally I create folders for both paths but use only one in my code. Just to make sure I know the usage of "../". Each "../" means jumping back to once to the parent folder. I use "../../" at the beginning of my output path so I finally put all outputs together.

akeLists.txt | main.cpp | Q1Output.txt | Q1.h

Distribution	Mean	Medium	Sample sd
Normal	9.98721	9.95241	4.99278
Poisson	5.0049	5	2.23691
ChiSquared	5.01319	4.38232	3.16895
Gamma	49.8788	48.3033	15.7761
Bernoulli	0.5022	1	0.500016

Figure 2: Output of Problem 1

2 Problem 2

For this problem, the polynomial function I use was $f(x) = -4 + x^2$ because the root for this one is easy to check.

Then, my C++ function will take input of an index vector and two different initial guesses. The index vector, for example, while solving $a_0 + a_1x + a_2x^2 + \cdots + a_nx^n = 0$, the index vector should be $a_0, a_1, a_2, \ldots, a_n$. And I made my functions print a report instead of only a number. Here is my function and output:

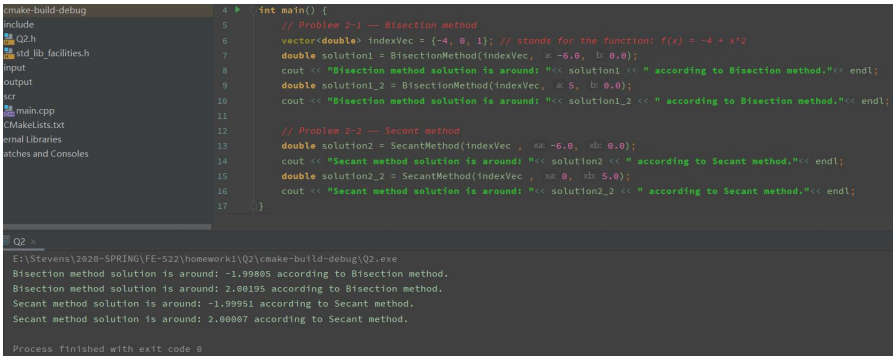


Figure 3: Output of Problem 2

Root of my polynomial function which falls between interval [-6, 0] is -2, and that falls between interval [0, 5] is 2. The results I got from functions of both methods are very close to the exact values. Therefore, they work well.

3 Problem 3

Based on the requirements, I created a class named Money which takes string as input and output. When creating a Money variable, the default value will be "$0.0", and I can also set values using similar format. For example, "$1234.567".

In test cases below, in order to make it more convenient for graders to check my work, I make output more detailed. Here, all the "money1" and "money2" four decimal places are printed out because this is the initial value I set and professor didn't ask us to make any change on it. The calculation results follow 4/5 rounding rules.

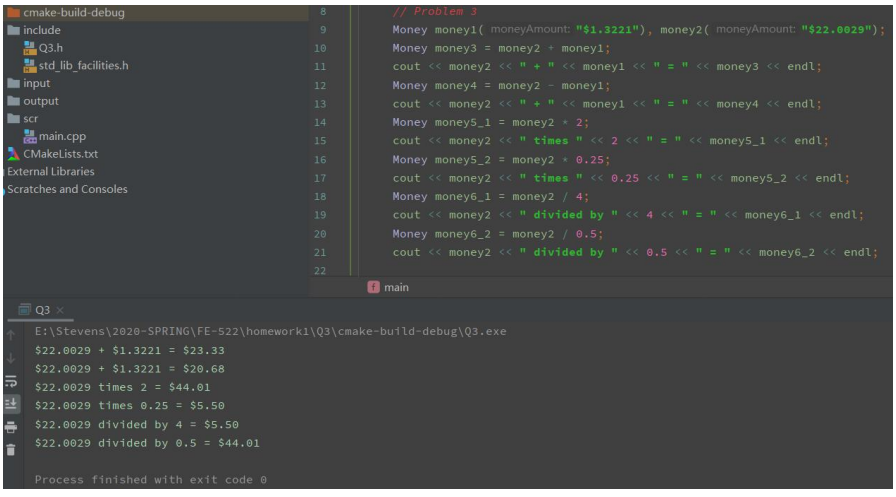


Figure 4: Some test cases for Problem 3

This is how I make my bool value "visible". And variable "money7" is used to test the "¿¿" operator.

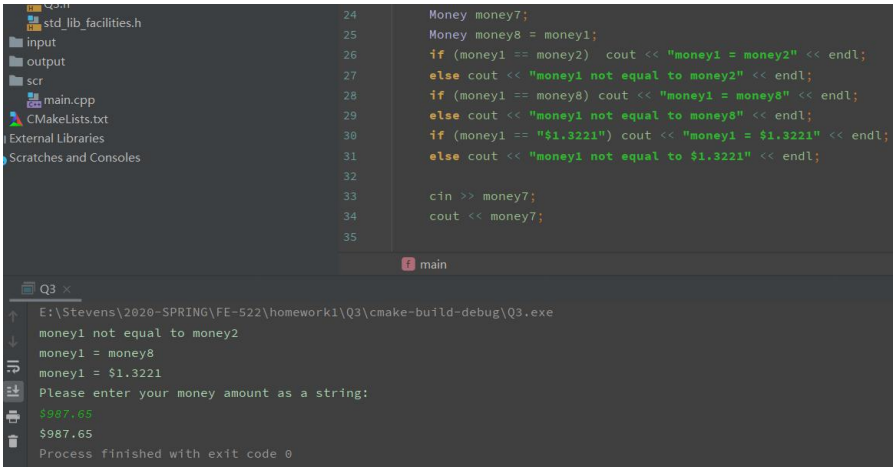


Figure 5: Some other test cases for Problem 3

4 Problem 4

Here is what my class contains. Two grey functions will be used in Problem 5.

```
// For problem 4:
class EuropeanOption{
private:
    string Type;
    double call();
    double put();
    double St, K, r, vol, d1, d2;
    int T;

public:
    EuropeanOption(string type, double spotPrice, double strikePrice,
                   double interestRate, double volatility, int timeToMature);
    double getPrice();
    double getBisectionVol(double price);
    double getSecantVol(double price);
};
```

Figure 6: Test cases and result for Problem 4

Here, input values are option type, spot price, strike price, interest rate, volatility and time to mature(year). As required, I didn't define an N() function myself.

```
double EuropeanOption::getPrice() {
    if(Type == "call") return call();
    else return put();
}
double EuropeanOption::call() {
    return (0.5 * erfc(-d1 * M_SQRT1_2))*St-(0.5 * erfc(-d2 * M_SQRT1_2))*K*exp(-r*T);
}
double EuropeanOption::put() {
    return (0.5 * erfc(d2 * M_SQRT1_2))*K*exp(-r*T)-(0.5 * erfc(d1 * M_SQRT1_2))*St;
}
```

Figure 7: Test cases and result for Problem 4

This is the test cases I use and price I get. The exit code is not 0 because professor said we should not accept

illegal input and I tested wrong type setting, wrong volatility input and wrong time to mature input.

Figure 8: Test cases and results for Problem 4

Figure 9: Other test case and result for Problem 4

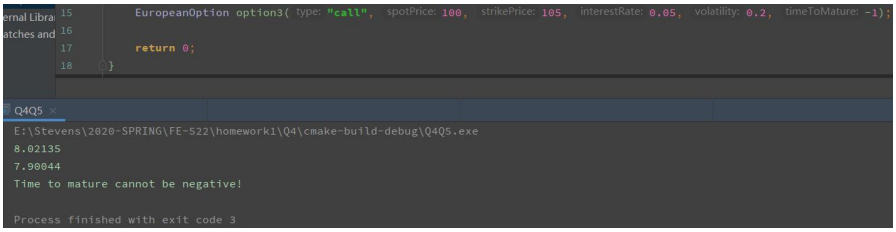
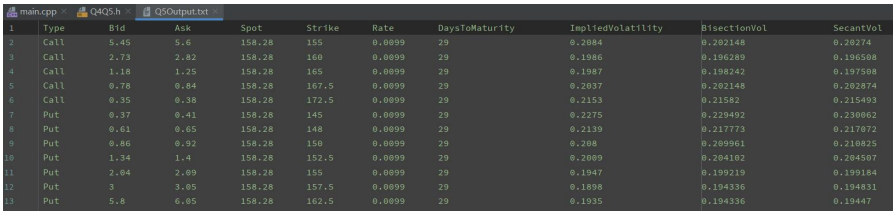


Figure 10: Other test case and result for Problem 4

5 Problem 5

For this problem, there isn't too much to say. This time the input value "timeToMature" in the constructor of EuropeanOption class should be mature day. I did change "time to mature" from day to year while calculating. Below is screenshot of my output file. All results from both methods are close to given volatility.



	Type	Bid	Ask	Spot	Strike	Rate	DaysToMaturity	ImpliedVolatility	BisectionVol	SecantVol
2	Call	5.45	5.6	158.28	155	0.0099	29	0.2084	0.202148	0.20274
3	Call	2.73	2.82	158.28	160	0.0099	29	0.1986	0.196289	0.196508
4	Call	1.18	1.25	158.28	165	0.0099	29	0.1987	0.198242	0.197508
5	Call	0.78	0.84	158.28	167.5	0.0099	29	0.2037	0.202148	0.202874
6	Call	0.35	0.38	158.28	172.5	0.0099	29	0.2153	0.21582	0.215493
7	Put	0.37	0.41	158.28	145	0.0099	29	0.2275	0.229492	0.230062
8	Put	0.61	0.65	158.28	148	0.0099	29	0.2139	0.217773	0.217072
9	Put	0.86	0.92	158.28	150	0.0099	29	0.208	0.209961	0.210825
10	Put	1.34	1.4	158.28	152.5	0.0099	29	0.2009	0.204102	0.204507
11	Put	2.04	2.09	158.28	155	0.0099	29	0.1947	0.199219	0.199184
12	Put	3	3.05	158.28	157.5	0.0099	29	0.1898	0.194336	0.194831
13	Put	5.8	6.05	158.28	162.5	0.0099	29	0.1935	0.194336	0.19447

Figure 11: Output of Problem 5

Because d_1 and d_2 will change when volatility changes, I bulit a new class "dInBS" (stands for d in BS model), in orderto update d_1 and d_2 when do Iteration to vol.

6 Problem 6

For this problem, I first use R to download stocks data from June 30^{th}, 2019 to Dec. 31^{st}, 2019, and use C++ to do later work. I choose 5 stocks from different industries, generate random weight and see whether current sharp ratio I get is larger than all before. If so, I mark down the weight as temp optimal. Finally I got a best one for that simulation. Also, in order to get sharp ratio, I find risk free rate on a website. It turns out that if I start from June 6^{th}, risk free rate is 2.19%.

In CLion, I print the optimal weights I got from that simulation, as well as the shares of stocks I should buy in the beginning. Then, I output the change of my portfolio value into a csv file and use R to visualize it.

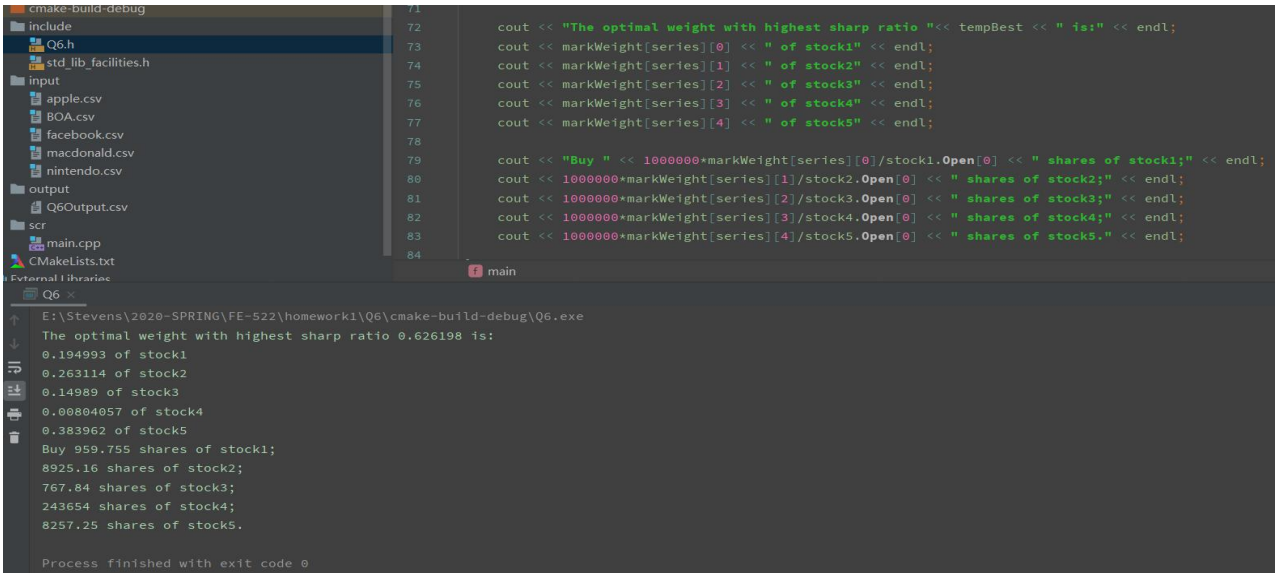


Figure 12: CLion output for Problem 6

This time, basically portfolio value is rising. But I replicated simulation for multiple times and often the value fluctuates a lot.

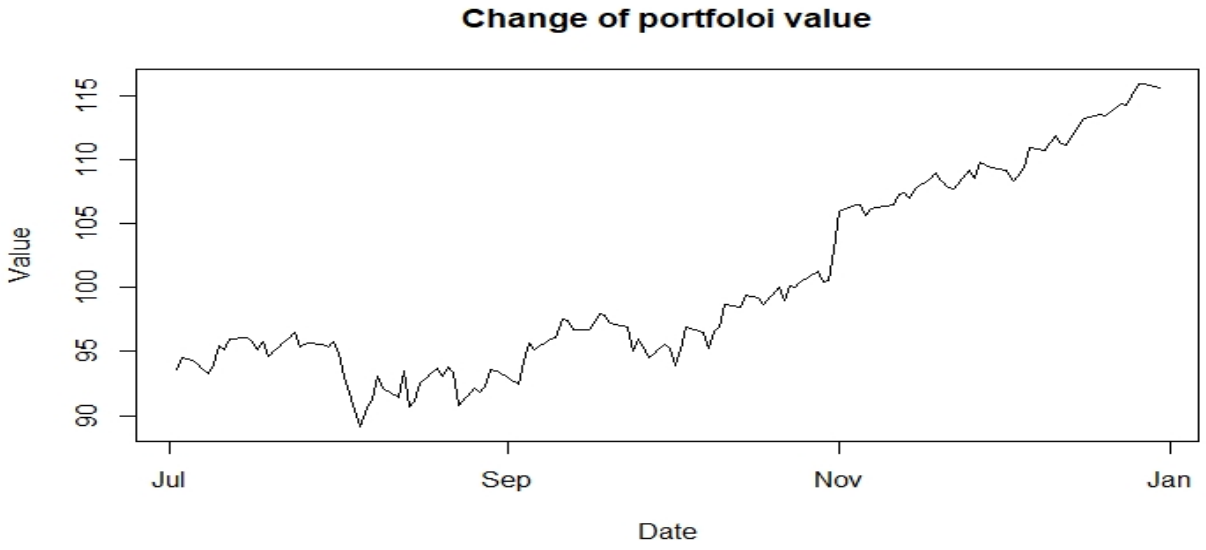


Figure 13: Portfolio value change