#### C Monte Carlo Method

###### b

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | NT | NSIM | Call | Put |
| Batch 1 | 100 | 1000 | 2.14877 | 5.88702 |
| 100 | 10000 | 2.13932 | 5.84287 |
| 500 | 1000 | 2.17492 | 6.13859 |
| 500 | 10000 | 2. 12329 | 5.94285 |
| 500 | 22000 | 2.13676 | 5.84864 |
| Batch 2 | 500 | 1000 | 7.93263 | 8.46301 |
| 500 | 10000 | 7.90847 | 8.13933 |
| 500 | 20000 | 7.92754 | 8.0651 |
| 500 | 100000 | 7.96121 | 7.95912 |

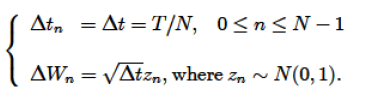
###### c

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | NT | NSIM | Call | Put |
| Batch 4 | 100 | 1000 | 92.715 | 1.3291 |
| 100 | 10000 | 87.7119 | 1.31043 |
| 100 | 100000 | 91.8968 | 1.2511 |
| 500 | 1000 | 95.5659 | 1.36483 |
| 500 | 10000 | 91.1309 | 1.28566 |
| 500 | 100000 | 91.845 | 1.25428 |

**Analysis**:

The NormalGenerator.cpp creates class that can be used the boost library to generate random number, which can be used to get dw as it can be considered as normally distribution random variable.

In the test.cpp code, we are required to input the NT and NSIM with difference value of batches, then the code defines the Wiener increments with the following equation:



After that, the code uses for loop to iterate large amount of simulations following the normal distribution and get the option value, then calculate the sum of the option price and divided by the simulation numbers to get the average price.

Finally, the code discounts the price and get the result.

here is the test result, I change the NT and NSIM several times (just list some of them) and found that it is difficult for us to find exact result. I supposed that within the increase of the simulation frequencies, the data will be closer than before, the value of call and input change irregularly in reality, I don’t think the convergence is monotone.