



Evaluation report: Black Scholes challenge

John Lewis

Candidate

ID : 148

Mail : candidatemail1@email.com

Phone : 07391586609

Degree :

School/University : UCL

Experience : 6

Your comment :

Candidate comment: Candidat can leave a comment , which will be sent to your recruiter..

Session

ID session : 4q9t

Started : 2018-07-08 00:57:02

Finished : 2018-07-08 01:00:51

Test	Test type	Absolute score	Relative score	Global score
1. [C++] Black Scholes simulation	Coding	5 / 5	<input type="text" value="85%"/>	5 / 5

1. [C++] Black Scholes simulation

1-1. Subject coding

Black Scholes simulation [C++]

1- Description:

This test consists of creating a European option pricer, under the Black-Scholes model (extended for dividends by Merton) based on simulation (monte carlo). A skeleton of this pricer is provided.

2- Objective:

The objective of this test is to Implement the following function BSSimul (...) to return the price of the option using the simulation method (a function BS (...) giving the price by the closed formula is provided to serve as your benchmark).

3- The skeleton code provided:

```
NormalFunctions.h // Math / utility function
BlackScholesSimulation.h // the file BSSimul based pricing (supplied with a Gaussian
```

1-2. Unit test cases

Test name	Candidate Output Value	Correct output	Execution time	Memory consumption	Result
Unit test1	4.244	4.152	0.02 sec	0 KB	Passed
Unit test2	9.54605	12.7	0.00 sec	0 KB	Passed
Unit test3	7.15622	52.155	0.00 sec	0 KB	Passed
Unit test4	39.5632	82.02	0.00 sec	0 KB	Passed
Unit test5	41.2614	50.29	0.00 sec	0 KB	Passed

1-3. Candidate Source code

Main File: BlackScholesSimulation.h

generator box-muller)

4- Note:

1- the only work to do is to complete the implementation of the BSSimul function (...) File BlackScholesSimulation.h

2-Your code does not print the screen unless you want to test / debugger in this case make sure you remove these impressions screen before submitting your solution

3- The run button allows you to run a unit test on your solution

Language : cpp

Compilation:Successfully

Marks Scored: 5/5

```
// Black-Scholes by simulation.  
// TestCandidat.com
```

```
#include <iostream>  
#include <iomanip>  
#include <vector>  
#include <math.h>  
#include <stdlib.h>  
#include <time.h>  
#include <algorithm>  
#include "NormalFunctions.h"  
using namespace std;
```

```
//BSSimul compute the price of an european option using simulation  
// S Spot Price  
// K Strike Price  
// T Maturity in Years  
// rf Interest Rate  
// q Dividend yeild  
// v Volatility  
// PutCall 'P' for put and 'C' for call  
// Nsims Number of simulations
```

```
double BSSimul(double S,double K,double v,double T,double rf,double q,  
char PutCall) {
```

```
double u1,u2,Z;  
double pi = 3.141592653589793;  
int Nsims = 1e5; // Number of simulations  
vector<double> ST(Nsims, 0.0); // Initialize terminal prices S(T)  
vector<double> ST_K(Nsims, 0.0); // Initialize call payoff  
vector<double> K_ST(Nsims, 0.0); // Initialize put payoff  
for (int i=0; i<=Nsims-1; i++) {  
// Independent uniform random variables  
u1 = ((double)rand() / ((double)RAND_MAX)+(double)(1)) );  
u2 = ((double)rand() / ((double)RAND_MAX)+(double)(1)) );  
// Floor u1 to avoid errors with log function  
u1 = max(u1,1.0e-10);  
// Z ~ N(0,1) by Box-Muller transofmrtration  
Z = sqrt(-2.0*log(u1)) * sin(2*pi*u2);  
ST[i] = S*exp((rf - q - 0.5*v*v)*T + v*sqrt(T)*Z); // Simulated terminal price  
S(T)  
ST_K[i] = max(ST[i] - K, 0.0); // Call payoff  
K_ST[i] = max(K - ST[i], 0.0); // Put payoff  
}  
// Simulated prices as discounted average of terminal prices  
double BSCallSim = exp(-rf*T)*VecMean(ST_K);  
double BSPutSim = exp(-rf*T)*VecMean(K_ST);  
if(PutCall == 'C')  
return BSCallSim;  
else  
return BSPutSim;  
}
```