Package User Manual

This package allows you to perform various computations on European call options, barrier options and look-back options in the `C++` language.

Table of Contents

Features

- Compute the European call option, the barrier option and the look-back price
- Compute the European call option, the barrier option and the look-back option greeks:
    - delta
    - gamma
    - vega
- Provide statistical informations

Installation

Building the project

The current repository comes with precompiled libraries in the `/lib` folder. However, if you edit the functions in the code, or simply prefer to import the header, we have also included a

`Makefile` for your convenience.
Most users will skip this part to go directly to importing the package.

In order to compile and run the project, you can use the following command in the `src` folder.

```
make
./run
```

In case of an issue, run the following command.

```
make clean
```

Another cause of issue is the absence of the directory `/src/bin`. In this case, you can create it manually using the following command.

```
mkdir bin
```

Importing the package

Importing with headers

Importing via shared libraries (recommended)

Linux

Windows NT

MacOS

Usage

Random numbers generation

In order to generate a sample of numbers drawn from normal distribution, a separate class 'random_normal' was implemented. **Note that if you use either the European call option, the barrier option or the look-back option, the generation of random samples is already done automatically.**

You can specify parameters of the normal distribution from which numbers are to be drawn. In this declaration 'name' can be arbitrary, 'm' and 'v' correspond to mean and variance respectively. By default, standard normal distribution will be created:

```
random_normal name_1(m, v);
random_normal name_2();    // same as:  random_normal name_2(0,1)
```

Now the method *generate* can be used with an integer argument 'n', to draw n numbers from distributions specified:

```
name_2.generate(n);
```

After n numbers have been drawn, you can access i-th number by using square brackets [], for instance:

```
double d = name_2[i];
```

You can also generate a single random number from standard normal distribution N(0,1) simply using global function:

```
get_random();    // the result is long double
```

Finally, you can compute the cdf and pdf of the standard normal distribution by using following functions:

```
normal_cdf(d);    // input/output are double types
normal_pdf(d);    // input/output are double types
```

European call option

Initialising the European call option

Before you use any of the functions of the package, you must create a `european_option` object to interact with. This can be done using the following method. You will need the following parameters:

- `S_t0 (double)` the value of the underlying stock at time t0. Default is `100`.
- `strike (double)` the value of the strike of the European option. Default is `100`.
- `interest_rate (double)` the decimal value of the interest rate. For an interest rate of 15%, enter `0.15`. Default is `0.05`.
- `volatility (double)` the decimal value of the volatility of the underlying stock. For an volatility of 15%, enter `0.15`. Default is `0.40`.
- `time_maturity_T (double)` the time of maturity of the call. Default value is `1 .0`.
- `initial_time_t0 (double)` the initial time at which `S_t0` was recorded. Default value is `0.0`.
- `number_iterations_approximation (int)` the number of iterations for the approximation methods. The default value is 10,000. Note than the larger this number is, the slower but more accurate computations are. Industry standards are 100,000.

```
european_option call(S_t0, strike, interest_rate, volatility, time_maturity_T, time
```

You can access most of the program's functionalities by accessing the functions as follows `my_european_call_option.the_function_i_use()`.

Price

The price is computed using the Monte Carlo method with the number of iterations specified when initialising the `european_call_option` class. Recall that the value is 10,000 iterations by default.

```
double call.price();
```

You can however specify any other number of iterations by specifying an `int` number in the parameters as follows.

```
double call.price(100000);
```

Greeks

Delta

In order to compute the call option delta, you can use the following method. By default (with any string input or no input at all), the method used is *likelihood ratios*.

```
double call.delta();
```

You can alsospecify the method you want to use.

- To use the *pathwise estimates* method, enter the argument `"pw"`.
- To use the *likelihood ratios* method, enter the argument `"lr"` or alternatively do not enter any argument.

```
double call.delta("pw");
double call.delta("lr");
```

Gamma

In order to compute the call option gamma, you can use the following method. By default (with any string input or no input at all), the method used is *pathwise estimates - likelihood ratios*.

```
double call.gamma();
```

You can also specify the method you want to use.

- To use the *likelihood ratios - pathwise estimates* method, enter the argument `"lrpw"`.
- To use the *likelihood ratios - likelihood ratios* method, enter the argument `"lrlr"`.
- To use the *pathwise estimates - likelihood ratios* method, enter the argument `"pwlr"` or alternatively do not enter any argument.

```
double call.gamma("lrpw");
double call.gamma("lrlr");
double call.gamma("pwlr");
```

Vega

In order to compute the call option vega, you can use the following method. By default (with any string input or no input at all), the method used is *likelihood ratios*.

```
double call.vega();
```

You can alsospecify the method you want to use.

- To use the *pathwise estimates* method, enter the argument `"pw"`.
- To use the *likelihood ratios* method, enter the argument `"lr"` or alternatively do not enter any argument.

```
double call.vega("pw");
double call.vega("lr");
```

Barrier option

Initialising the barrier option

The `barrier_option` object takes the same arguments as the `european_option` plus the value of the barrier.
You will need the following parameters:

- `barrier_value (double)` the value of the barrier.
- `S_t0 (double)` the value of the underlying stock at time t0. Default is `100`.
- `strike (double)` the value of the strike of the European option. Default is `100`.
- `interest_rate (double)` the decimal value of the interest rate. For an interest rate of 15%, enter `0.15`. Default is `0.05`.
- `volatility (double)` the decimal value of the volatility of the underlying stock. For an volatility of 15%, enter `0.15`. Default is `0.40`.
- `time_maturity_T (double)` the time of maturity of the call. Default value is `1 .0`.
- `initial_time_t0 (double)` the initial time at which `S_t0` was recorded. Default value is `0.0`.
- `number_iterations_approximation (int)` the number of iterations for the approximation methods. The default value is 10,000. Note than the larger this number is, the slower but more accurate computations are. Industry standards are 100,000.

```
barrier_option boption(barrier_value, S_t0, strike, interest_rate, volatility, time_
```

You can access most of the program's functionalities by accessing the functions as follows `my_barrier_option.the_function_i_use()`.

Price

The price is computed using the Monte Carlo method with the number of iterations specified when initialising the `barrier_option` class. Recall that the value is 10,000 iterations by default.

```
double boption.price();
```

Greeks

Delta

In order to compute the barrier option delta, you can use the following method. By default (with any string input or no input at all), the method used is *likelihood ratios*.

```
double boption.delta();
```

You can alsospecify the method you want to use.

- To use the *theoretical value*, enter the argument `"th"`.
- To use the *likelihood ratios* method, enter the argument `"lr"` or alternatively do not enter any argument.

```
double boption.delta("th");
double boption.delta("lr");
```

Gamma

In order to compute the barrier option gamma, you can use the following method. By default (with any string input or no input at all), the method used is *likelihood ratios*.

```
double boption.gamma();
```

You can alsospecify the method you want to use.

- To use the *theoretical value*, enter the argument `"th"`.
- To use the *likelihood ratios* method, enter the argument `"lr"` or alternatively do not enter any argument.

```
double boption.gamma("th");
double boption.gamma("lr");
```

Vega

In order to compute the barrier option vega, you can use the following method. By default (with any string input or no input at all), the method used is *likelihood ratios*.

```
double boption.vega();
```

You can alsospecify the method you want to use.

- To use the *theoretical value*, enter the argument `"th"`.
- To use the *likelihood ratios* method, enter the argument `"lr"` or alternatively do not enter any argument.

```
double boption.vega("th");
double boption.vega("lr");
```

Gamma

In order to compute the barrier option gamma, you can use the following method. By default (with any string input or no input at all), the method used is *likelihood ratios*.

```
double boption.gamma();
```

You can alsospecify the method you want to use.

- To use the *theoretical value*, enter the argument `"th"`.
- To use the *likelihood ratios* method, enter the argument `"lr"` or alternatively do not enter any argument.

```
double boption.gamma("th");
double boption.gamma("lr");
```

Vega

In order to compute the barrier option vega, you can use the following method. By default (with any string input or no input at all), the method used is *likelihood ratios*.

```
double boption.vega();
```

You can alsospecify the method you want to use.

- To use the *theoretical value*, enter the argument `"th"`.
- To use the *likelihood ratios* method, enter the argument `"lr"` or alternatively do not enter any argument.

```
double boption.vega("th");
double boption.vega("lr");
```

Look-back option

You can access most of the program's functionalities by accessing the functions as follows `my_lookback_option.the_function_i_use()`.

Price

The price is computed using the Monte Carlo method with the number of iterations specified when initialising the `lookback_option` class. Recall that the value is 10,000 iterations by default.

```
double lboption.price();
```

Greeks

Delta

In order to compute the barrier option delta, you can use the following method. By default (with any string input or no input at all), the method used is *likelihood ratios*.

```
double boption.delta();
```

You can alsospecify the method you want to use.

- To use the *theoretical value*, enter the argument `"th"`.
- To use the *likelihood ratios* method, enter the argument `"lr"` or alternatively do not enter any argument.

```
double boption.delta("th");
double boption.delta("lr");
```

Gamma

In order to compute the barrier option gamma, you can use the following method. By default (with any string input or no input at all), the method used is *likelihood ratios*.

```
double boption.gamma();
```

You can alsospecify the method you want to use.

- To use the *pathwise derivatives* method, enter the argument `"pw"`.
- To use the *likelihood ratios* method, enter the argument `"lr"` or alternatively do not enter any argument.

```
double boption.gamma("pw");
double boption.gamma("lr");
```

Vega

In order to compute the barrier option vega, you can use the following method.

```
double lboption.vega();
```

Gamma

In order to compute the barrier option gamma, you can use the following method.

```
double lboption.gamma();
```

Vega

In order to compute the barrier option vega, you can use the following method.

```
double lboption.vega();
```