



CUDA programming on GPU for option pricing

Valerio Firmano 918239
Matteo Tajana 962031
Enrico Fornasa 960633

1. Introduction

- Goal of the project —————> pricing options' derivatives
- Use of GPU —————> fastening code execution

2. Mathematical and Financial Background

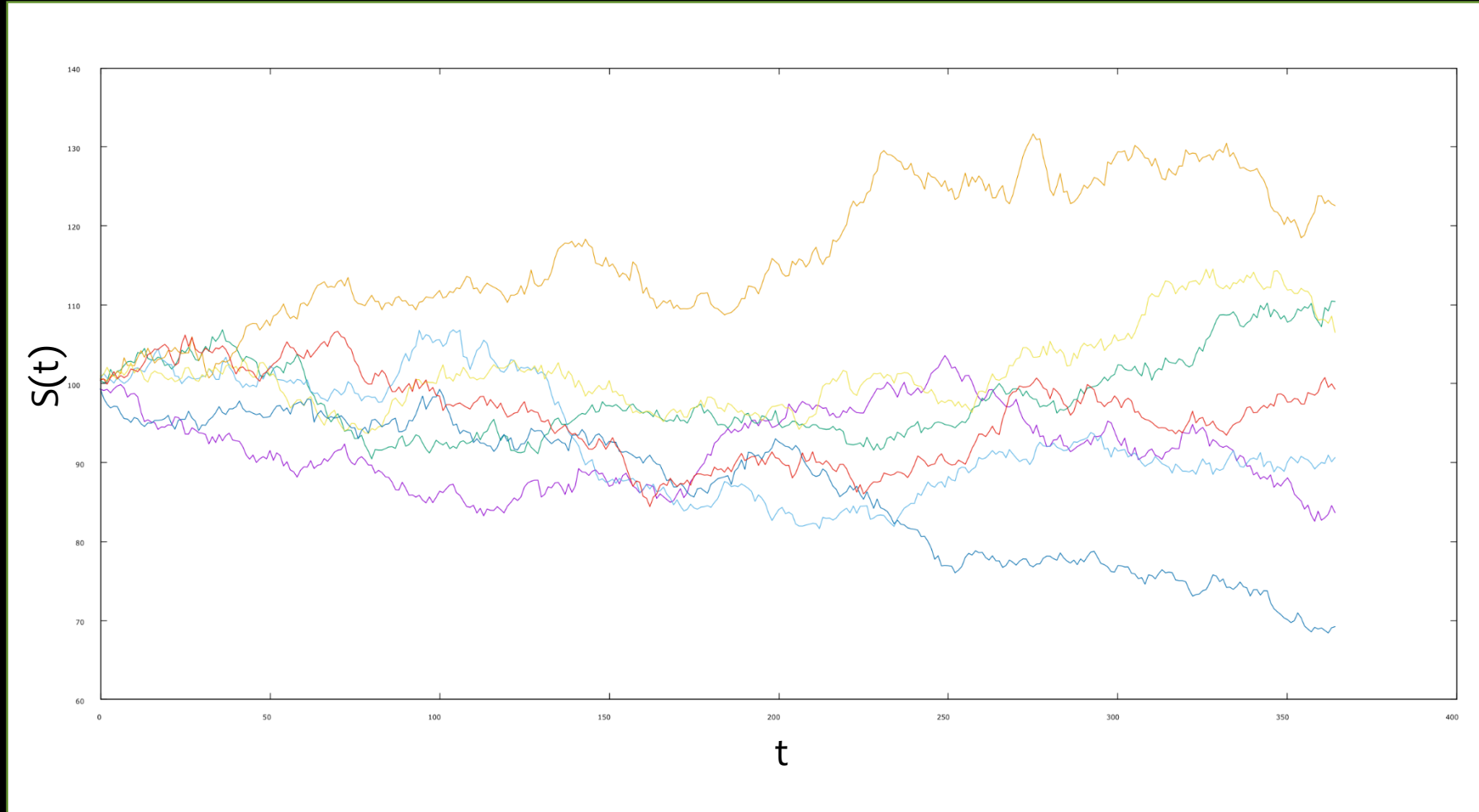
2.1 Stock price evolution model

- (Stochastic) Markovian process
- Lognormal evolution: $\frac{dS}{S} = rdt + \sigma\sqrt{dt}\omega$

two solutions:

- i. Exact solution: $S(t_{i+1}) = S(t_i)e^{\left(r - \frac{\sigma^2}{2}\right)\Delta t + \sigma\sqrt{\Delta t}\omega}$
- ii. Euler's solution: $S(t_{i+1}) = S(t_i)(1 + r\Delta t + \sigma\sqrt{\Delta t}\omega)$

2. Mathematical and Financial Background

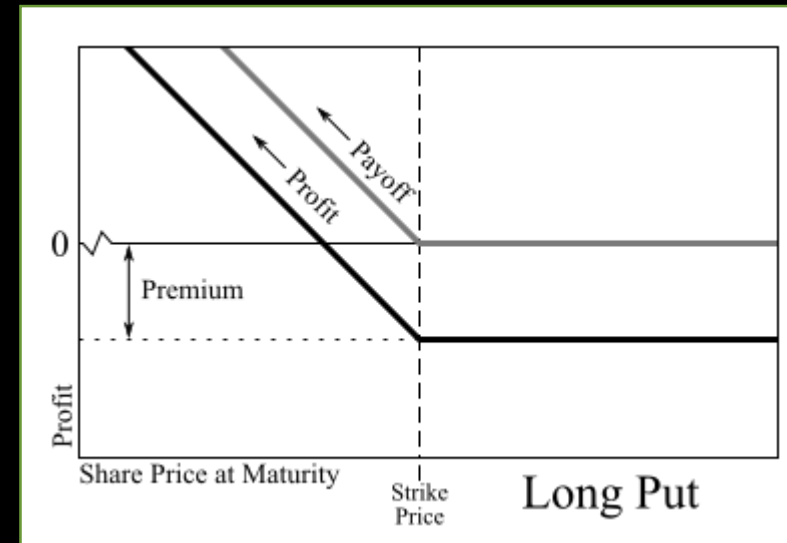
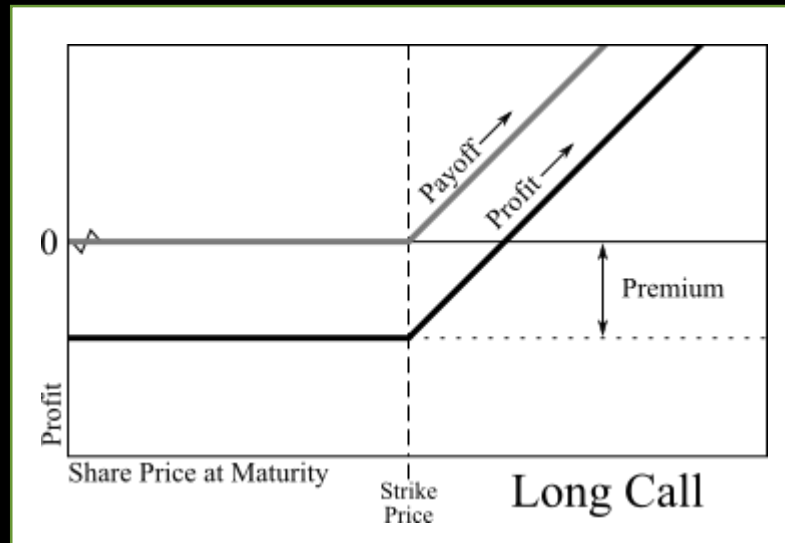


2.2 Options and Payoffs

Options: contract which give the buyer the right to buy or sell an underlying asset (stock, other options...)

2.2.1 Plain vanilla options: $\text{Payoff} = F(S(t = T))$

- i. Plain vanilla Call options: $\text{Payoff}_{\text{call}} = \max[S(T) - E, 0]$
- ii. Plain vanilla Put options: $\text{Payoff}_{\text{put}} = \max[E - S(T), 0]$



2.2.2 Path dependent option:

I. Positive Performance Corridor (PPC):

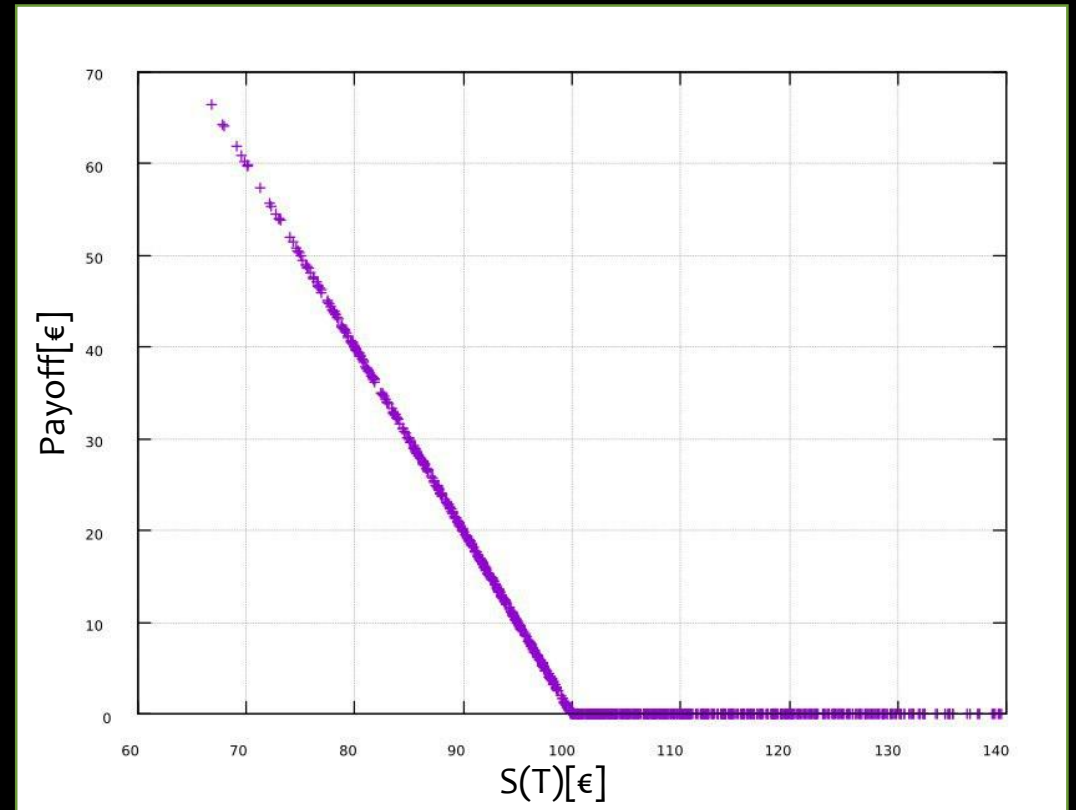
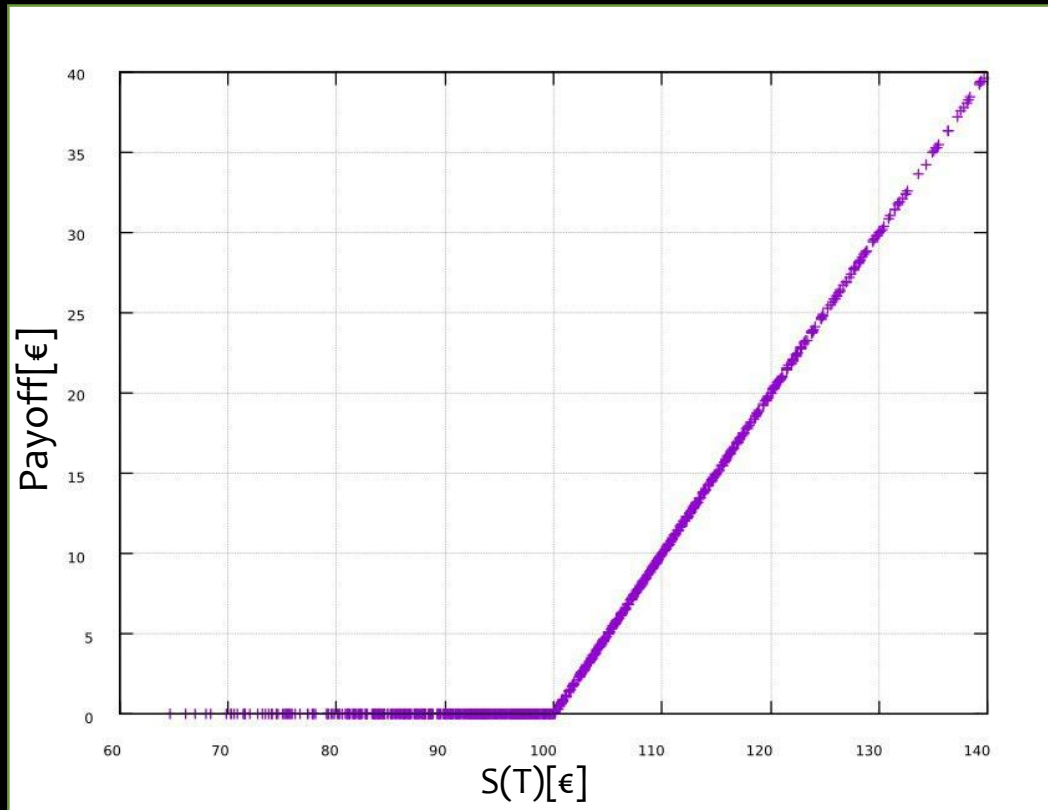
$$\text{Payoff} = N \left[\left(\frac{1}{m} \sum_{i=0}^{m-1} P_i \right) - K \right]^+$$

$$P_i = \begin{cases} 1 & \text{if } 0 < \frac{1}{\Delta t} \ln \left(\frac{S_{t+1}}{S_t} \right) < B\sigma \\ 0 & \text{otherwise} \end{cases}$$

3. Simulations

3.0 Check Plain Vanilla option's payoff.

Showing the shape of plain vanilla options' payoffs



3.1.1 Plain Vanilla Call options

Evaluate of **Call** options payoffs with different number of time steps.

Call option

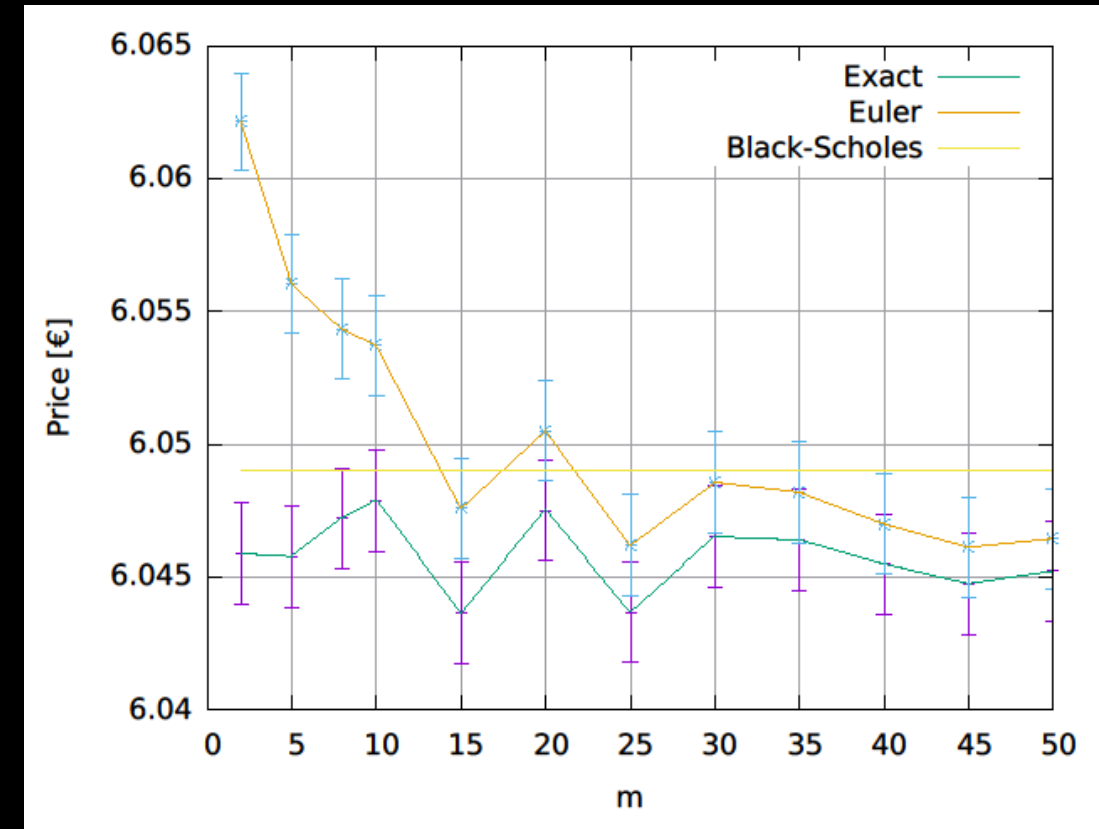
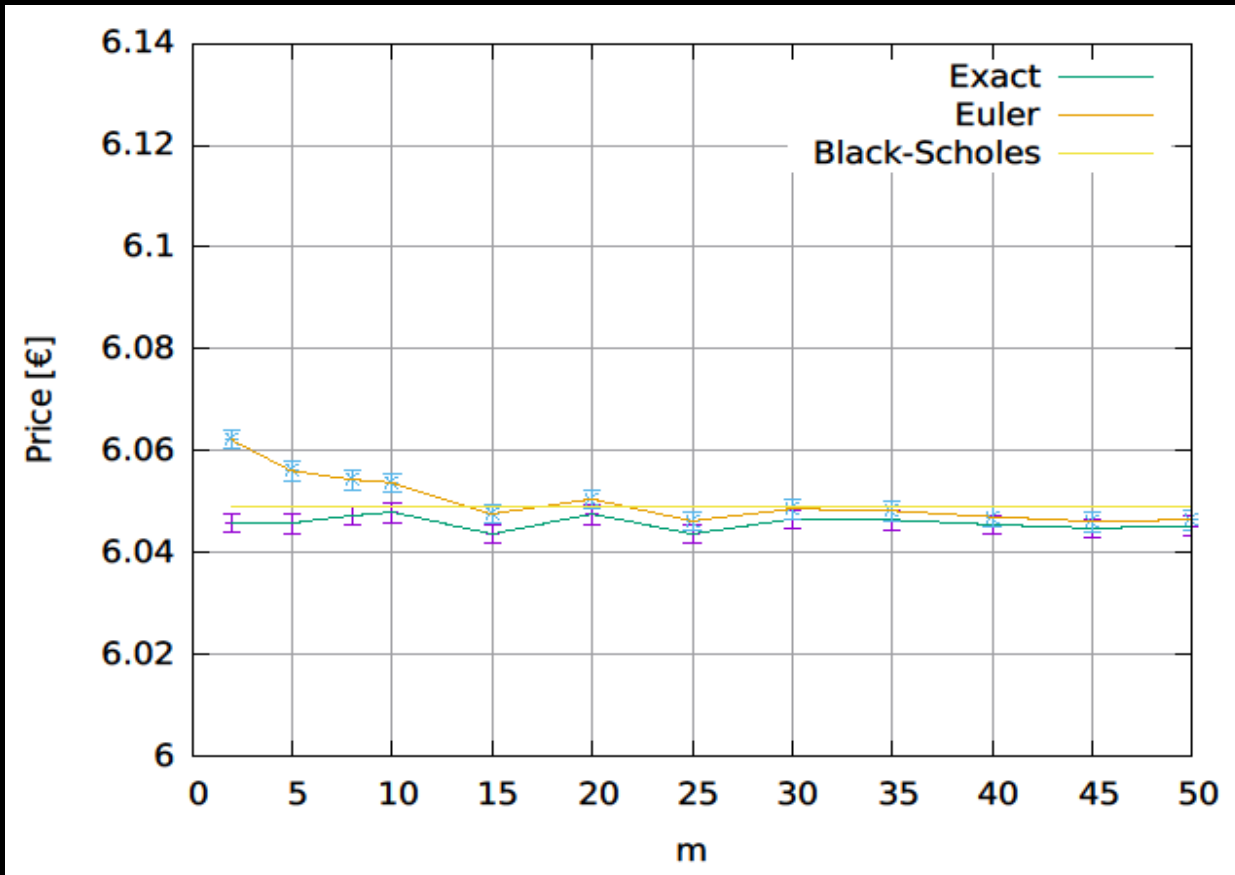
$S = 100\text{€}$ $E = 100\text{€}$

$r = 0.15\%$ $\sigma = 15\%$

$T = 1\text{yr}$

5120 threads

$N = 5000$



3.1.1 Plain Vanilla Call options

Evaluate of Call options payoffs with different number of time steps.

Call option

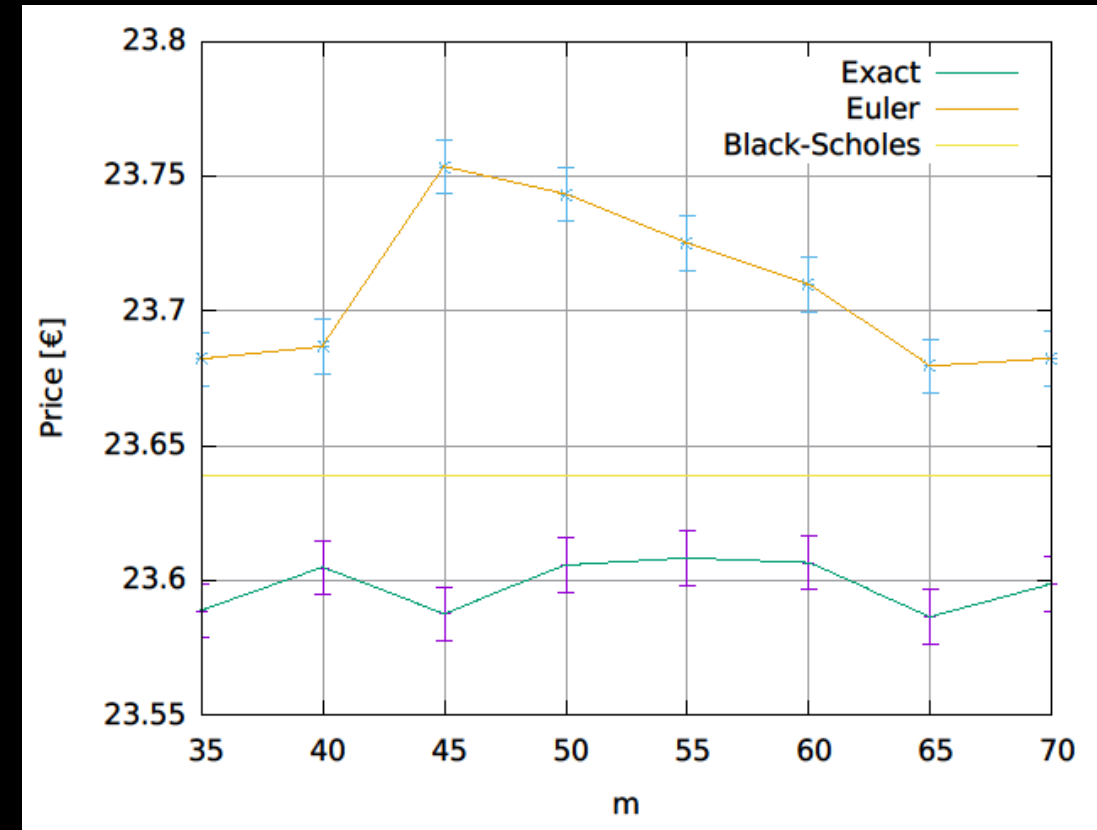
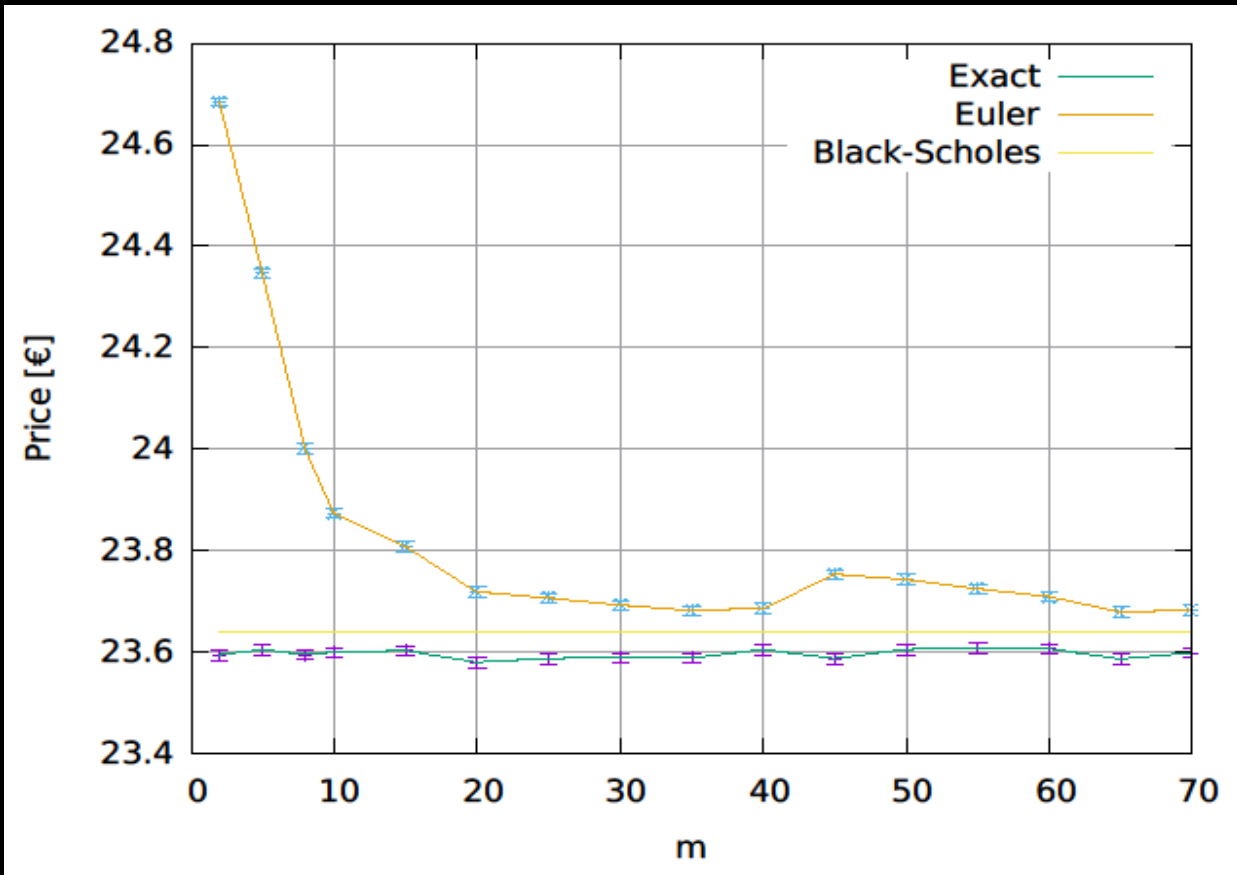
$S = 100\text{€}$ $E = 100\text{€}$

$r = 0.15\%$ $\sigma = 60\%$

$T = 1\text{yr}$

5120 threads

$N = 5000$



3.1.2 Plain Vanilla Put options

Evaluate of Put options payoffs with different number of time steps.

Call option

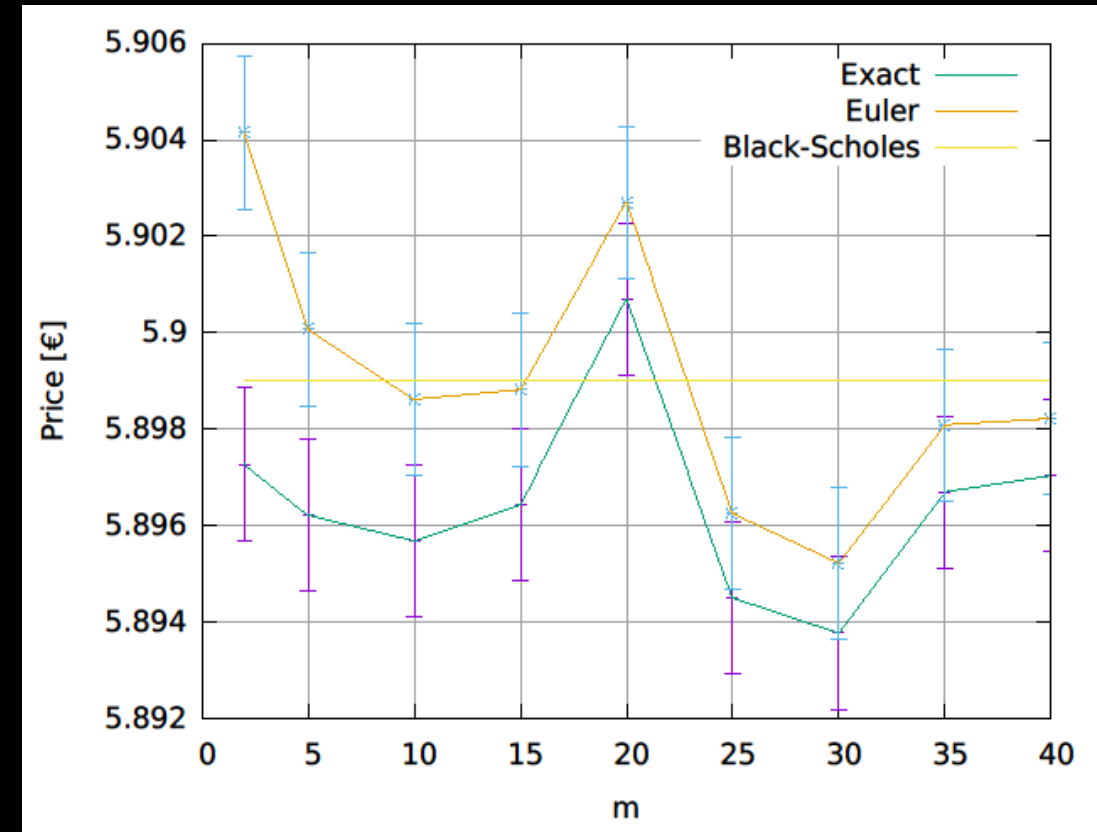
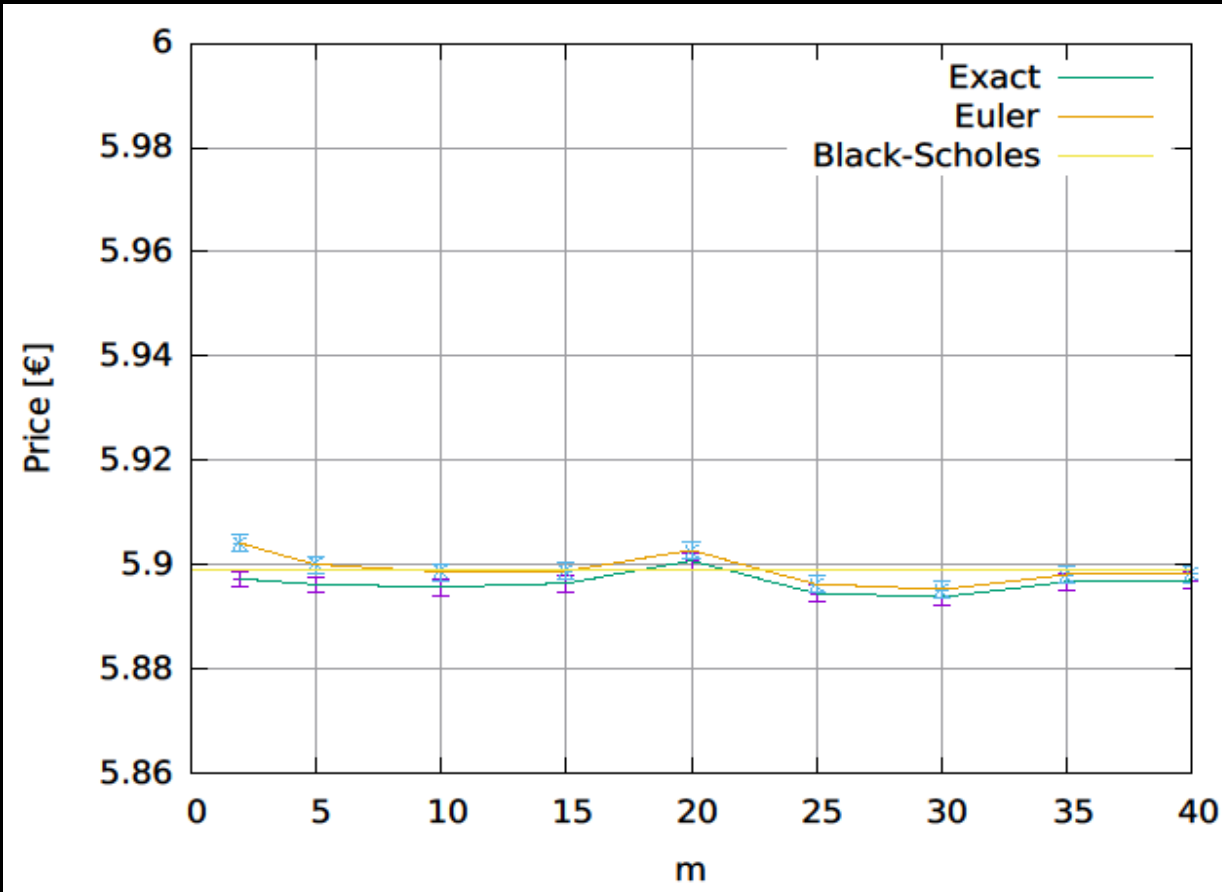
$S = 100\text{€}$ $E = 100\text{€}$

$r = 0.15\%$ $\sigma = 15\%$

$T = 1\text{yr}$

5120 threads

$N = 5000$



3.1.2 Plain Vanilla Put options

Evaluate of Put options payoffs with different number of time steps.

Call option

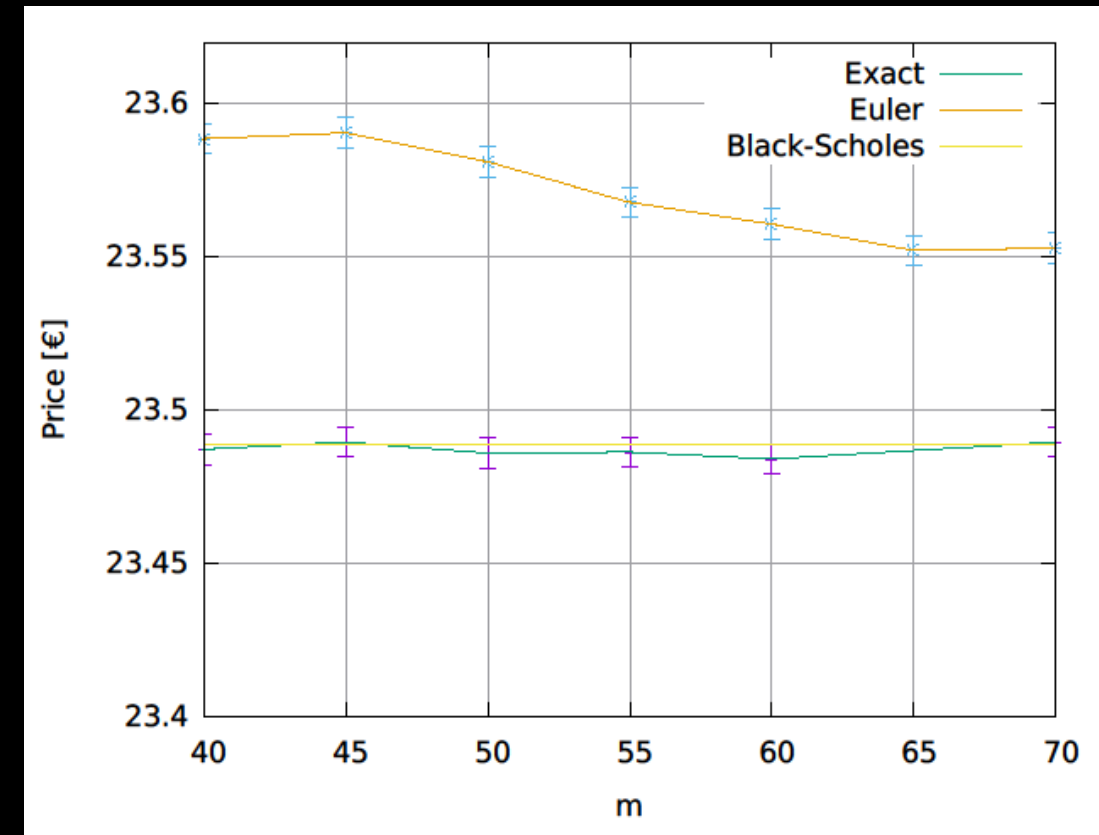
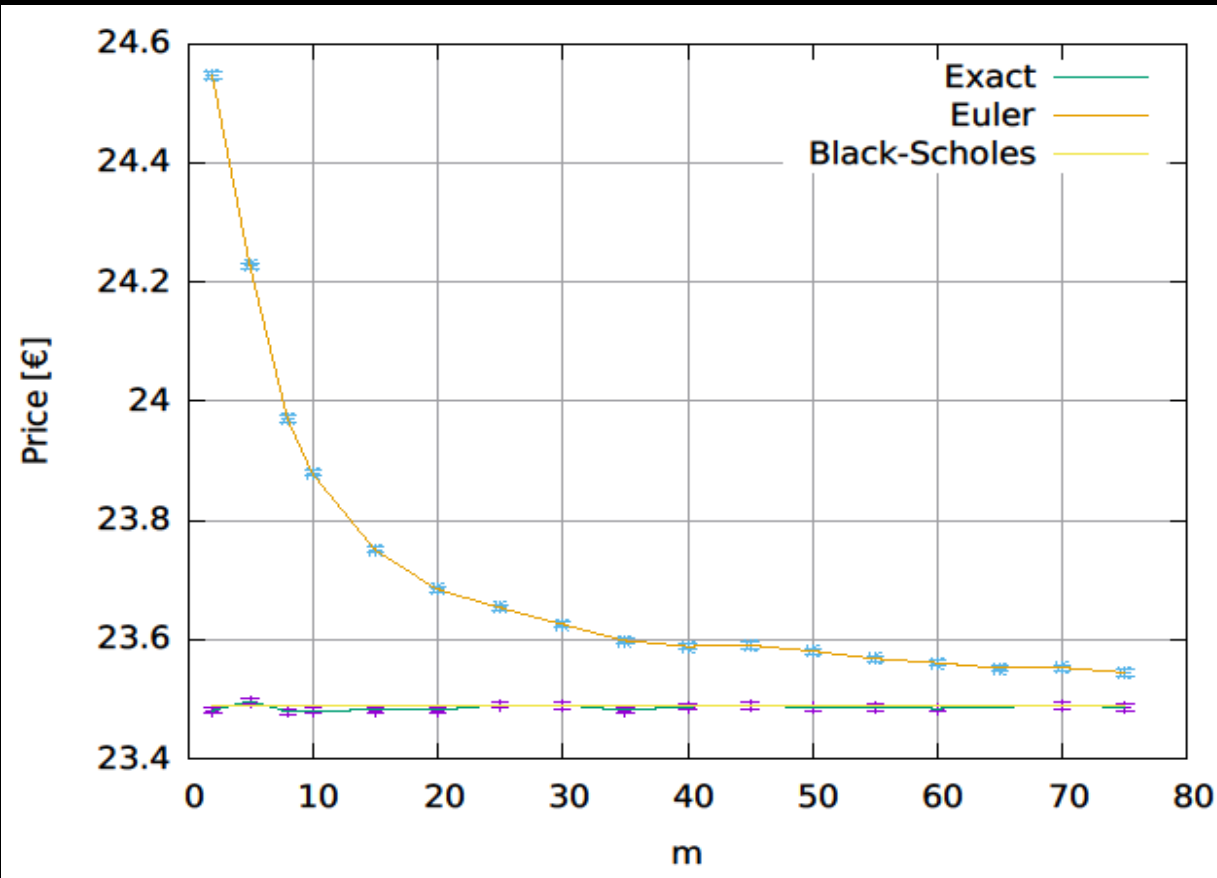
$S = 100\text{€}$ $E = 100\text{€}$

$r = 0.15\%$ $\sigma = 60\%$

$T = 1\text{yr}$

5120 threads

$N = 5000$



3.2 Positive Performance Corridor options

Evaluate of PPC options payoffs with different market data versus B.

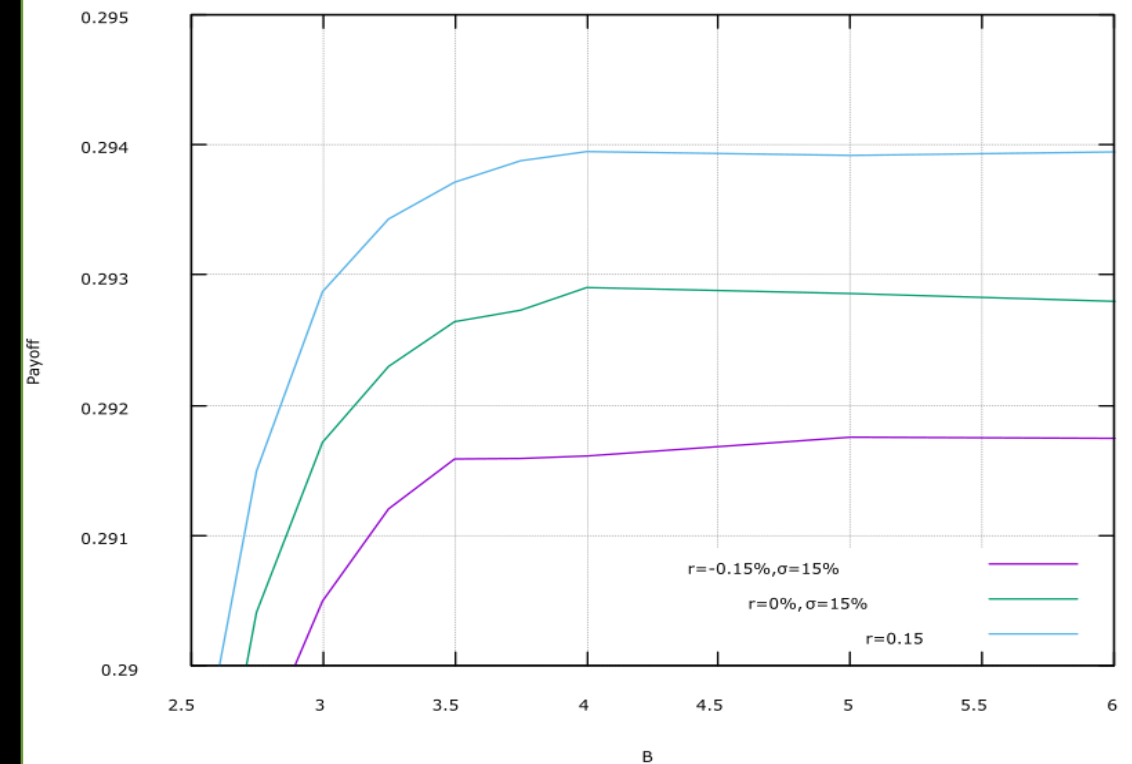
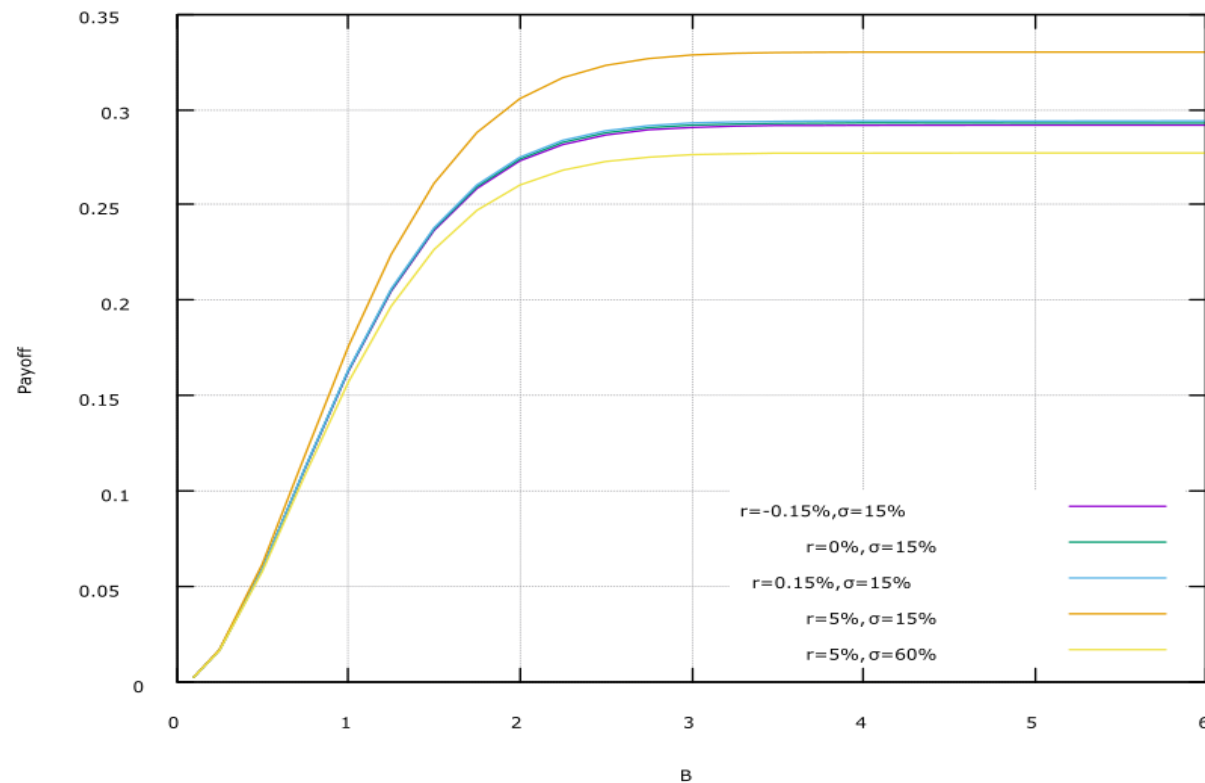
PPC option

$S = 100\text{€}$ $E = 100\text{€}$

$K = 0,15$ $T = 1\text{yr}$

256 threads

$N = 1000$



3.3 Unit Test

We implemented a Unit Test to check the exactness of GPU results, evolving on GPU and CPU several stochastic processes and calculating payoffs of a european call and a forward contract on them .

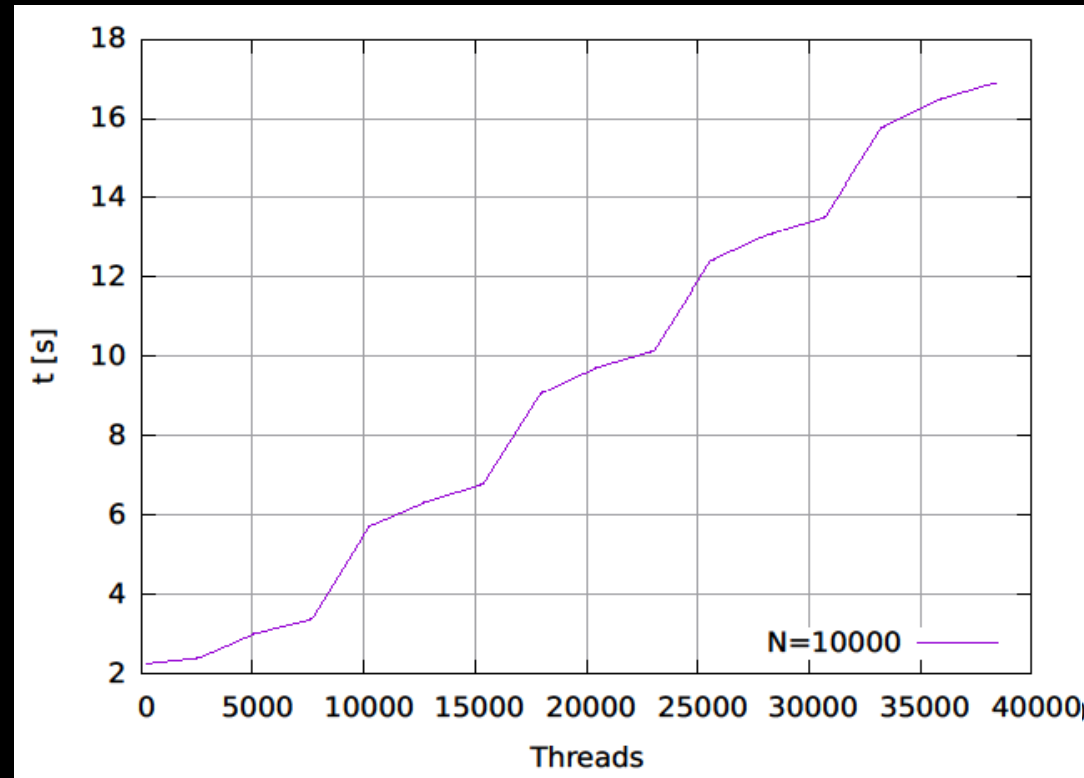
OEC GPU [€]	OEC CPU [€]	FC GPU [€]	FC CPU [€]
39.5355802861	39.5355802861	139.5355802861	139.5355802861
58.8089015441	58.8089015441	158.8089015441	158.8089015441
8.8584289951	8.8584289951	108.8584289951	108.8584289951
2.2724059387	2.2724059387	102.2724059387	102.2724059387
18.6201200624	18.6201200624	118.6201200624	118.6201200624

$$S = 100\text{€} \quad E = 100\text{€} \quad r = 20\% \quad \sigma = 15\% \quad T = 1\text{yr} \quad m = 30$$

4. Performance and comparison between GPU and CPU

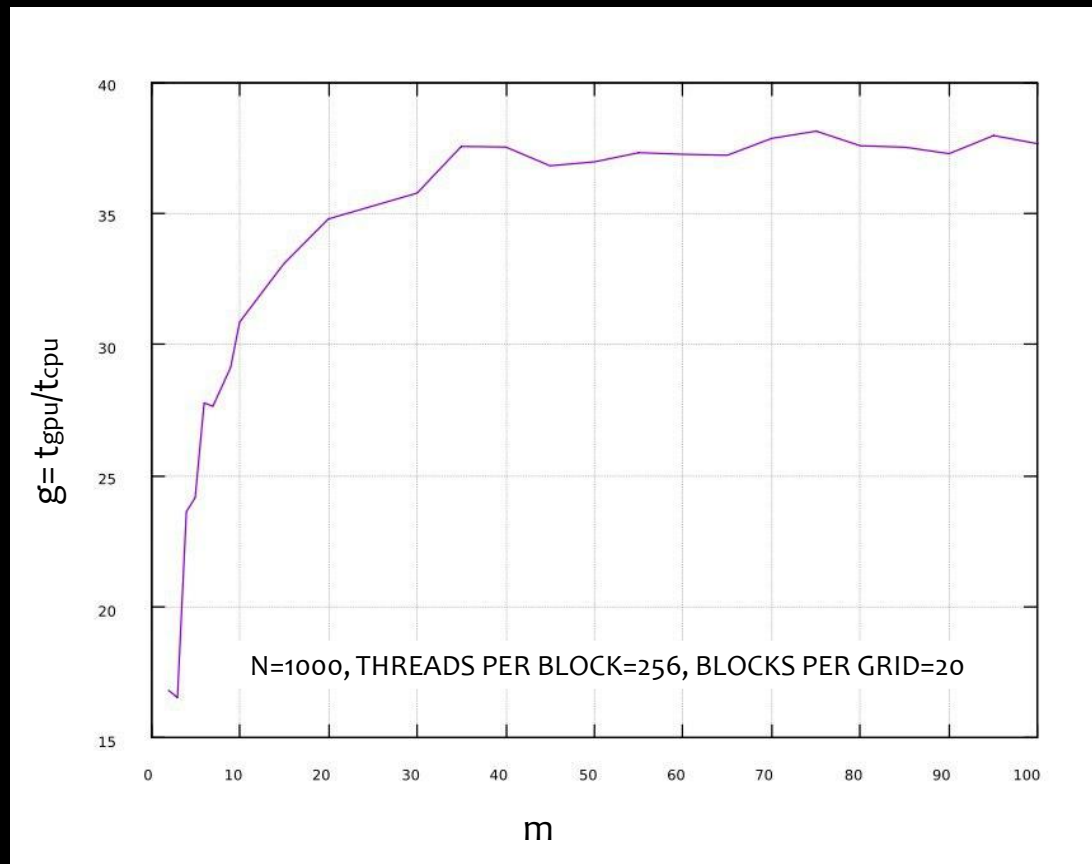
4.1 Time of GPU

Time needed for calculating the payoff of a European option call with $N=10000$ simulations as a function of number of threads



4.2 Time gain factor

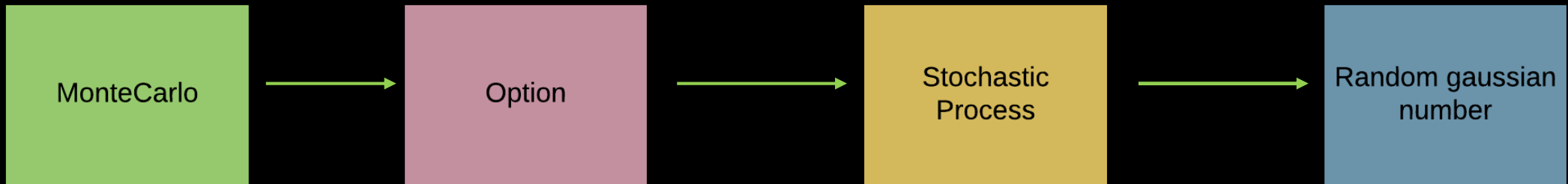
Using GPU gives an advantage in terms of time of calculation. The time gain factor between GPU and CPU is calculated as $g = \frac{t_{GPU}}{t_{CPU}}$, and is showed as a function of m with different GPU parameters.



5. Implementation of the library


5.1 Structure of the main routine

Each thread performs the same routine N times → path integrate results with discount factor $\exp(-rT)$.



5.2 Classes used in the Library

5.2.1 Rng

- Generation of uniform random numbers combining four different generators with bitwise XOR: three Tausworthe and a Linear Congruential Generator.
- Box-Muller method  no numbers wasted

5.2 Classes used in the Library

5.2.2 Stochastic_Process

- Points to an Rng object
- Creates and stores stock price evolution path
- Two methods implemented: *Exact* and *Euler* (finite differences)

5.3 Classes used in the Library

5.3.3 Opt

- Opt abstract base class \longrightarrow virtual method Payoff()
- Points to a Stochastic_Process object \longrightarrow path
- European Call, European Put, Exotic Positive Performance Corridor
- Different payoffs, same method \longrightarrow user friendly
- Plain vanilla options have also BlackScholes() method: option's price computed as analytical solution of Black-Scholes equation

5.3 Classes used in the Library

5.3.4 Montecarlo

- Points to an option
- Performs N simulations \longrightarrow N paths
- Stores cumulative payoff and standard deviation

6. Conclusions

- Advantages of using GPU \longrightarrow 30X fastening
- Comparison between Exact and Euler's method
- Compatibility of Montecarlo simulations for Plain Vanilla options and analytical results
- Trend of PPC price with different values of B