Introduction to Simulation Methods in Finance

Giovanni Della Lunga

May 5, 2016

Outline

What is Monte Carlo?

- From a quite general point of view (not a really precise one, actually) with the term Monte Carlo usually one means a numerical technique which makes use of random numbers for solving a problem.
- For the moment we assume that you can understand, at least intuitively, what a random number is.
- Later we will return to the definition of a random number, and, as we shall see, this will lead to absolutely not trivial issues.
- Let's start immediately with some practical examples (we'll try to give a more formal definition later).

What is Monte Carlo?

- Let's consider two problems apparently very different in nature;
- The first one is of probabilistic nature: the assessment of the premium for an option of European type option on a stock that does not pay dividends;
- The second one is an issue of purely deterministic nature: the determination of the area enclosed by a plane figure, such as a circle.

What is Monte Carlo?

- Let's start with the first problem. The pricing of an option is usually dealt with in the context of so-called risk-neutral valuation.
- Indicating with $f[S(T)]$ where S is the value of the underlying asset, the value of the option at maturity T, the value today, $f[S(t)]$, is given by

$$f(S(t)) = \mathbb{E}^{\mathbb{Q}}\left[P(t,T)f[S(T)]\right]$$

- $\mathbb{E}^{\mathbb{Q}}$ being the risk-neutral expectation value and $P(t,T)$ the discount function between t and T.
- Let's assume, for simplicity, to know with certainty the value of the discount function so the problem can be put in the form

$$f(S(t)) = e^{-r(T-t)}\mathbb{E}^{\mathbb{Q}}\left[f[S(T)]\right]$$

What is Monte Carlo?

- The formulation of the problem makes clear its inherently probabilistic nature.
- The application of the Monte Carlo method in this case is reduced essentially to the generation of a sufficiently high number of estimates of $f[S(T)]$ from which to extract the average value.
- ⚠ To this end it is necessary first to introduce a hypothesis on how the underlying stock price evolves over time;

What is Monte Carlo?

- Let's suppose for example that the asset price follows a geometric Brownian motion, according to this hypothesis the rate of change of the price in a range of infinitesimal time is described by

$$dS = rSdt + S\sigma dw$$

where r is the risk free rate, σ is the volatility of S returns and dw is a brownian motion;

- First of all we choose a discrete version of this SDE:

$$\Delta S = rS\Delta t + S\sigma\epsilon\sqrt{\Delta T}$$

where ϵ is a random number drawn from a normal distribution (we assume for the moment to have some procedure that allows us to generate random numbers with probability distribution assigned);

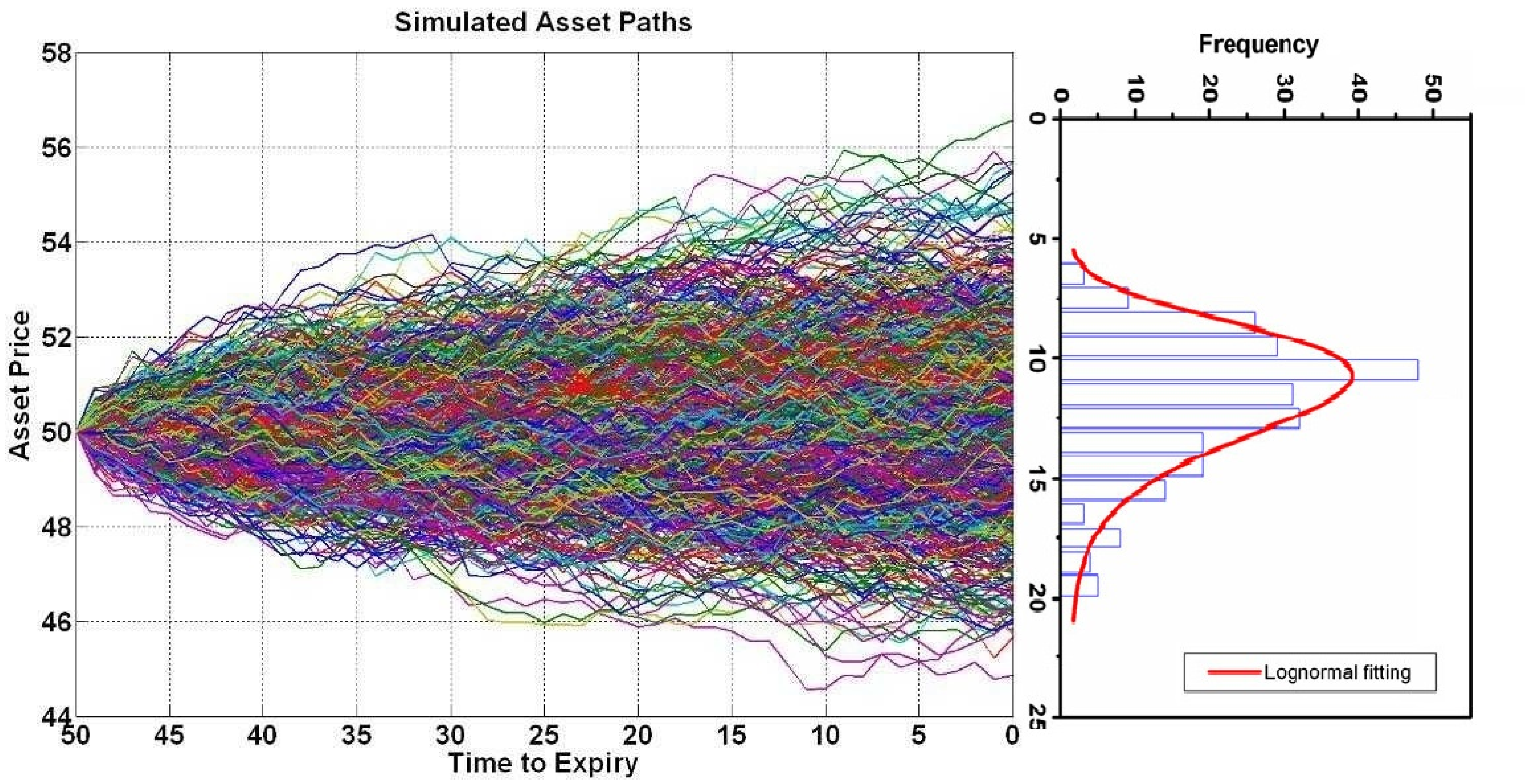
What is Monte Carlo?

- Once we have the simulated value of the underlying at time T, we are able to derive the value of the option at the same date;
- Assuming for example that the option is a CALL we simply write

$$f[S(T)] = \max(S(T) - K, 0)$$

  where K is the strike price.
- By repeating the above procedure a very large number of times we are able to obtain a distribution of values for $f[S(T)]$ from which it is possible to extract the expectation value ...

What is Monte Carlo?

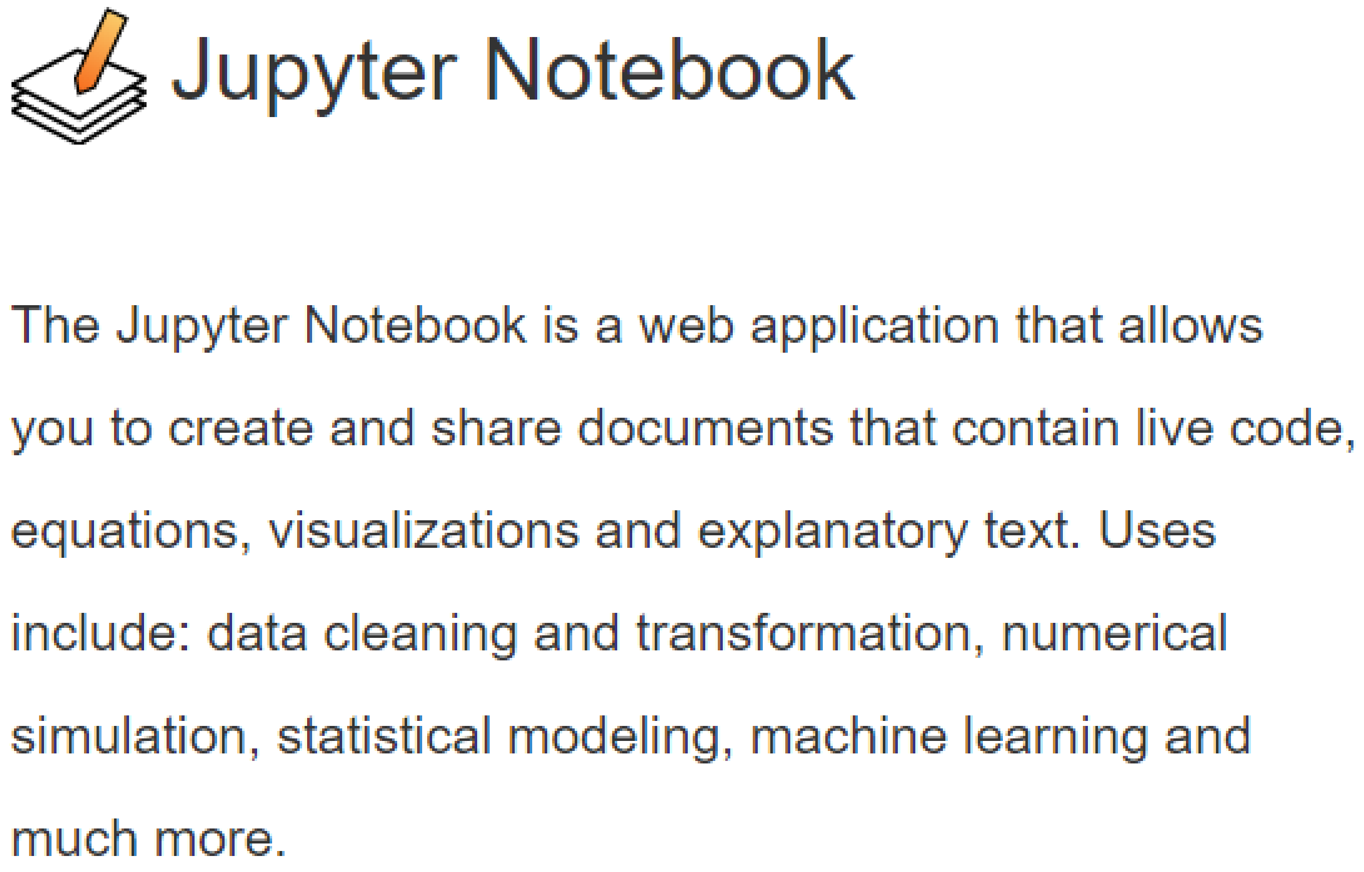


Interlude - Let's start coding...

Out toolbox: Jupyter, Python, R

- Python, R
- The Python world developed the IPython notebook system.
- Notebooks allow you to write text, but you insert code blocks as "cells" into the notebook.
- A notebook is interactive, so you can execute the code in the cell directly!
- Recently the Notebook idea took a much enhanced vision and scope, to explicitly allow languages other than Python to run inside the cells.
- Thus the Jupyter Notebook was born, a project initially aimed at Julia, Python and R (Ju-Pyt-e-R). But in reality many other languages are supported in Jupyter.

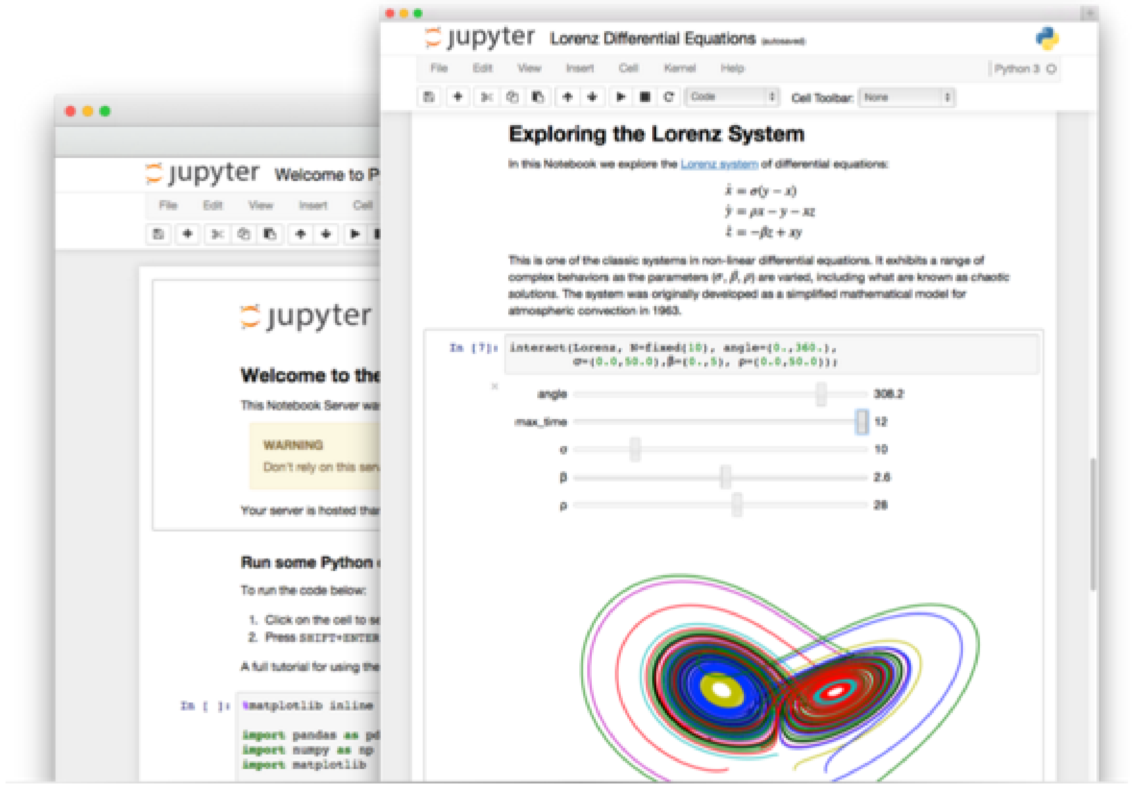
Out toolbox: Jupyter, Python, R



Jupyter Notebook

The Jupyter Notebook is a web application that allows you to create and share documents that contain live code, equations, visualizations and explanatory text. Uses include: data cleaning and transformation, numerical simulation, statistical modeling, machine learning and much more.

Out toolbox: Jupyter, Python, R


jupyter Welcome to P
File Edit View Insert Cell
jupyter Lorenz Differential Equations (autosaved)
File Edit View Insert Cell Kernel Help
Python 3
Code
Cell Toolbar: None

jupyter

Welcome to the

This Notebook Server wa

WARNING

Don't rely on this serv

Your server is hosted tha

Run some Python

To run the code below:

1. Click on the cell to se
2. Press SHIFT+ENTER

A full tutorial for using the

```
In [ ]: %matplotlib inline

        import pandas as pd
        import numpy as np
        import matplotlib
```

Exploring the Lorenz System

In this Notebook we explore the Lorenz system of differential equations:

$$\dot{x} = \sigma(y - x)$$
$$\dot{y} = \rho x - y - xz$$
$$\dot{z} = -\beta z + xy$$

This is one of the classic systems in non-linear differential equations. It exhibits a range of complex behaviors as the parameters (σ, β, ρ) are varied, including what are known as chaotic solutions. The system was originally developed as a simplified mathematical model for atmospheric convection in 1963.

```
In [7]: interact(Lorenz, N=fixed(10), angle=(0.,360.),
                 σ=(0.0,50.0),β=(0.,5), ρ=(0.0,50.0));
```

angle	308.2
max_time	12
σ	10
β	2.6
ρ	28

Out toolbox: Jupyter, Python, R

- **The IRKernel**
- To enable support of a new language means that somebody has to write a "kernel".
- The kernel for R is called IRKernel (available at github).
- **How do you use Jupyter?**
- Once Jupyter is up and running, you interact with it on a web page.

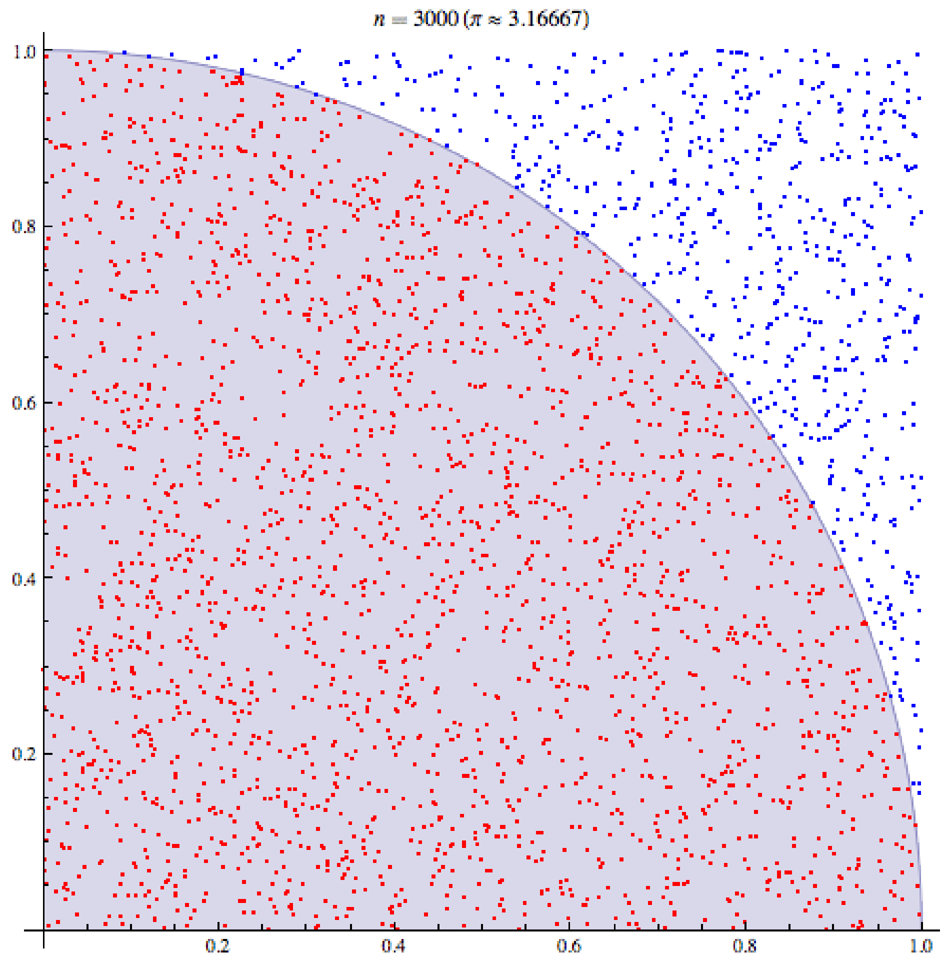
Out toolbox: Jupyter, Python, R

- **Benefits of using Jupyter**
- Jupyter was designed to enable sharing of notebooks with other people. The idea is that you can write some code, mix some text with the code, and publish this as a notebook. In the notebook they can see the code as well as the actual results of running the code.
- This is a nice way of sharing little experimental snippets, but also to publish more detailed reports with explanations and full code sets. Of course, a variety of web services allows you to post just code snippets (e.g. gist). What makes Jupyter different is that the service will actually render the code output.
- One interesting benefit of using Jupyter is that Github magically renders notebooks. See for example, the github Notebook gallery.

Notebook

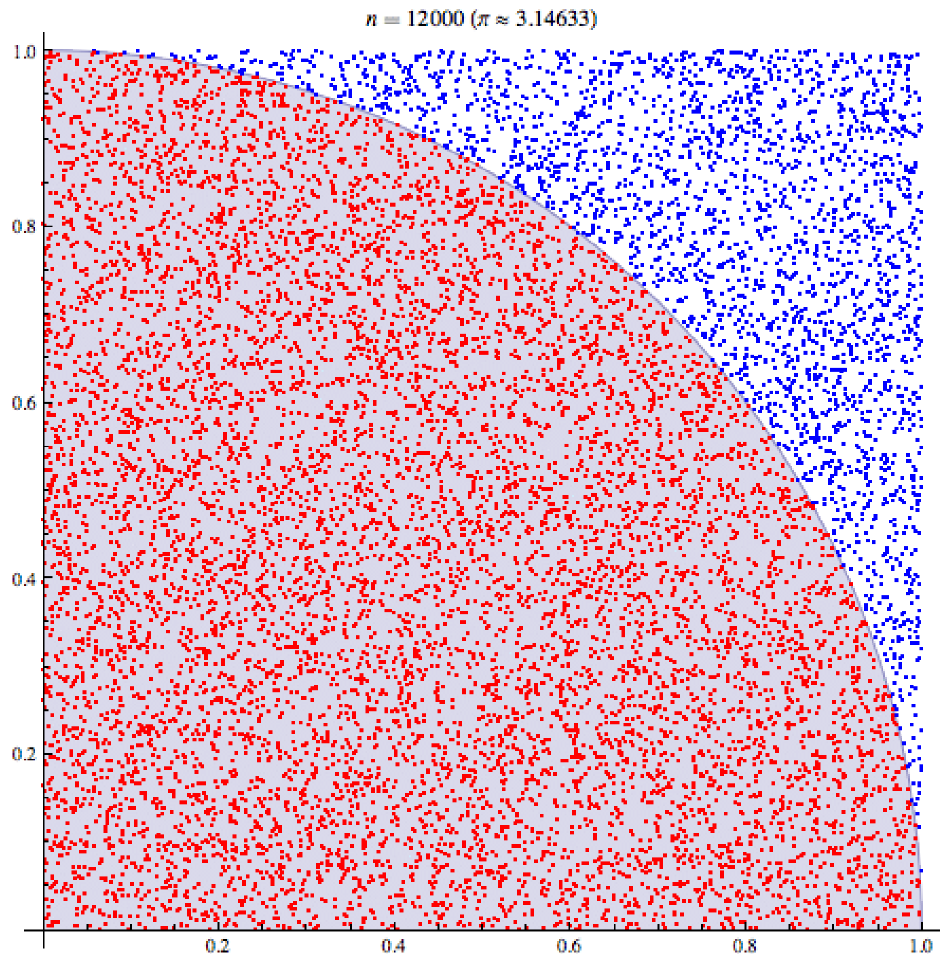
- **GitHub :** polyhedron-gdl;
- **Notebooks :** mcs_1;

What is Monte Carlo?



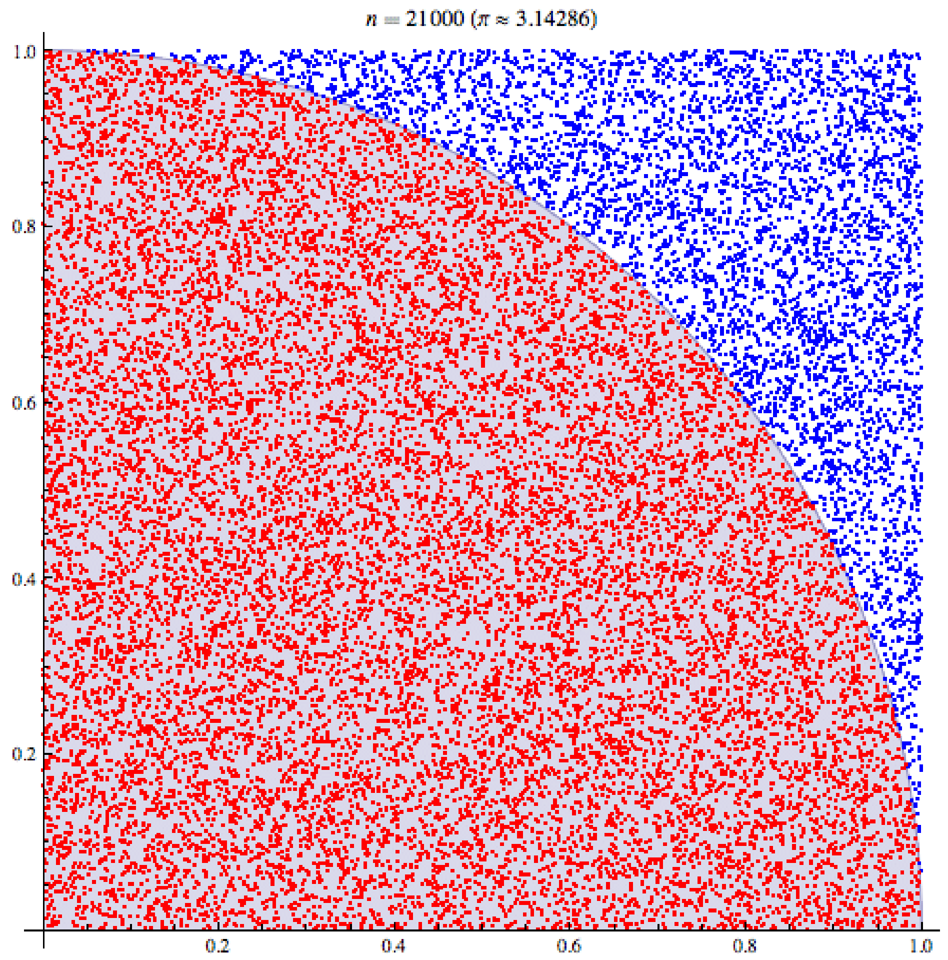
- Let's now consider the second problem;
- Take a circle inscribed in a unit square. Given that the circle and the square have a ratio of areas that is $\pi/4$, the value of π can be approximated using a Monte Carlo method:

What is Monte Carlo?



1. Draw a square on the ground, then inscribe a circle within it.
2. Uniformly scatter some objects of uniform size (grains of rice or sand) over the square.
3. Count the number of objects inside the circle and the total number of objects.
4. The ratio of the two counts is an estimate of the ratio of the two areas, which is $\pi/4$. Multiply the result by 4 to estimate π.

What is Monte Carlo?



- In this procedure the domain of inputs is the square that circumscribes our circle.
- We generate random inputs by scattering grains over the square then perform a computation on each input (test whether it falls within the circle).
- Finally, we aggregate the results to obtain our final result, the approximation of π.

Notebook

- **GitHub :** polyhedron-gdl;
- **Notebooks :** mcs_2;

Monte Carlo is Integration!

- There is a formal connection between the use of the Monte Carlo method and the concept of integration of a function.
- First of all we observe how the problems discussed in the previous paragraph can be attributed both to the calculation of integrals.
- The case related to the area of the circle is evident
- The price of an option as we have seen is nothing more than the discounted value of the expectation value of the price at maturity, the underlying risk factor (the stock price) is distributed according to a log-normal distribution, therefore, we have (for the CALL case):

$$C(t,S) = \frac{e^{-r(T-t)}}{\sqrt{2\pi}} \int_{-\infty}^{+\infty} h\left[S\exp\left((r-q-\sigma^2/2)(T-t)-\sigma x\sqrt{T-t}\right) e^{-\frac{1}{2}x^2}\right] dx$$

where $h(S) = |S-K|^+$.

Monte Carlo is Integration!

- More in general we can state that each extraction of a sample of random numbers can be used as an estimator of an integral.
- As an example consider the case relating to the integration of a function of a real variable;
- by a suitable change of variable, we can always bring us back to the simplest case in which the integration interval is between 0 and 1:

$$I = \int_0^1 f(x)dx$$

Monte Carlo is Integration!

- The key point of our argument is to recognize that the expression written above is also the expectation value of the function f at values of a random variable uniformly distributed in the range $[0, 1]$.
- It becomes possible to estimate the value of our integral using an arithmetic mean of n values of $f(U_i)$ where each U_i is a sample from a uniform distribution in $[0, 1]$.
- In other words we can say that the quantity

$$\tilde{I}_n = \frac{1}{n}\sum_{i=1}^{n} f(U_i)$$

is an unbiased estimator of I.

Monte Carlo is Integration!

- The variance of the estimator is

$$var\left(\tilde{I}_n\right) = \frac{var(f(U_i))}{n}$$

- the mean square error of the estimator, which can be interpreted as the mean square error of the Monte Carlo simulation, decreases with increasing n.
- This result is completely independent of the dimensionality of the problem.
- It's this last characteristic that makes attractive the Monte Carlo method for solving problems with a large number of dimensions.
- In this case typically the Monte Carlo method converge to the final value faster than the traditional numerical methods.

Pricing a Call Option

- It's worth to recast the pricing problem into a simple integral formulation in order to gain some insight into the general problem;
- So let's consider again the payoff of a simple plain vanilla option

$$e^{-rT}\mathbb{E}^{\mathbb{Q}}[h(S_T)] = e^{-rT}\mathbb{E}^{\mathbb{Q}}\left[h\left(S_0 e^{\log(S_T/S_0)}\right)\right]$$

- By a simple application of Ito's lemma is easy to demonstrate that the variable $X = \log(S_T/S_0)$ has a normal distribution with mean $m = (r - \frac{1}{2}\sigma^2)T$ and variance $s = \sigma^2 T$.
- So we can write

$$C(S,t) = e^{-rT} S_0 \int_0^{+\infty} \max[e^X - K, 0] e^{-\frac{(X-m)^2}{2s^2}} dX$$

Pricing a Call Option

- Let's now recall the fundamental **probability integral transform** which relates to the result that data values that are modelled as being random variables from any given continuous distribution can be converted to random variables having a uniform distribution
- This result states that if a random variable X has a continuous distribution for which the cumulative distribution function (CDF) is F_X. Then the random variable Y defined as

$$Y = F_X(X)$$

has a uniform distribution;

Pricing a Call Option

- This means that, in our case, we can say that the variable

$$u = \Phi[X; m, u], \quad u \to 1 \, when \, X \to +\infty$$

has a uniform distribution;

- From the previous relation we find (within a normalization factor)

$$du = \frac{d\Phi[X; m, u]}{dX} dX \Rightarrow dX = \frac{1}{e^{-\frac{(X-m)^2}{2s^2}}} du$$

- and finally we can write our integral in the form

$$C(S, t) = \int_0^1 f(u) du$$

where $f(u) = e^{-rT} \max[S_0 \exp(\Phi^{-1}(u; m, s)) - K, 0]$

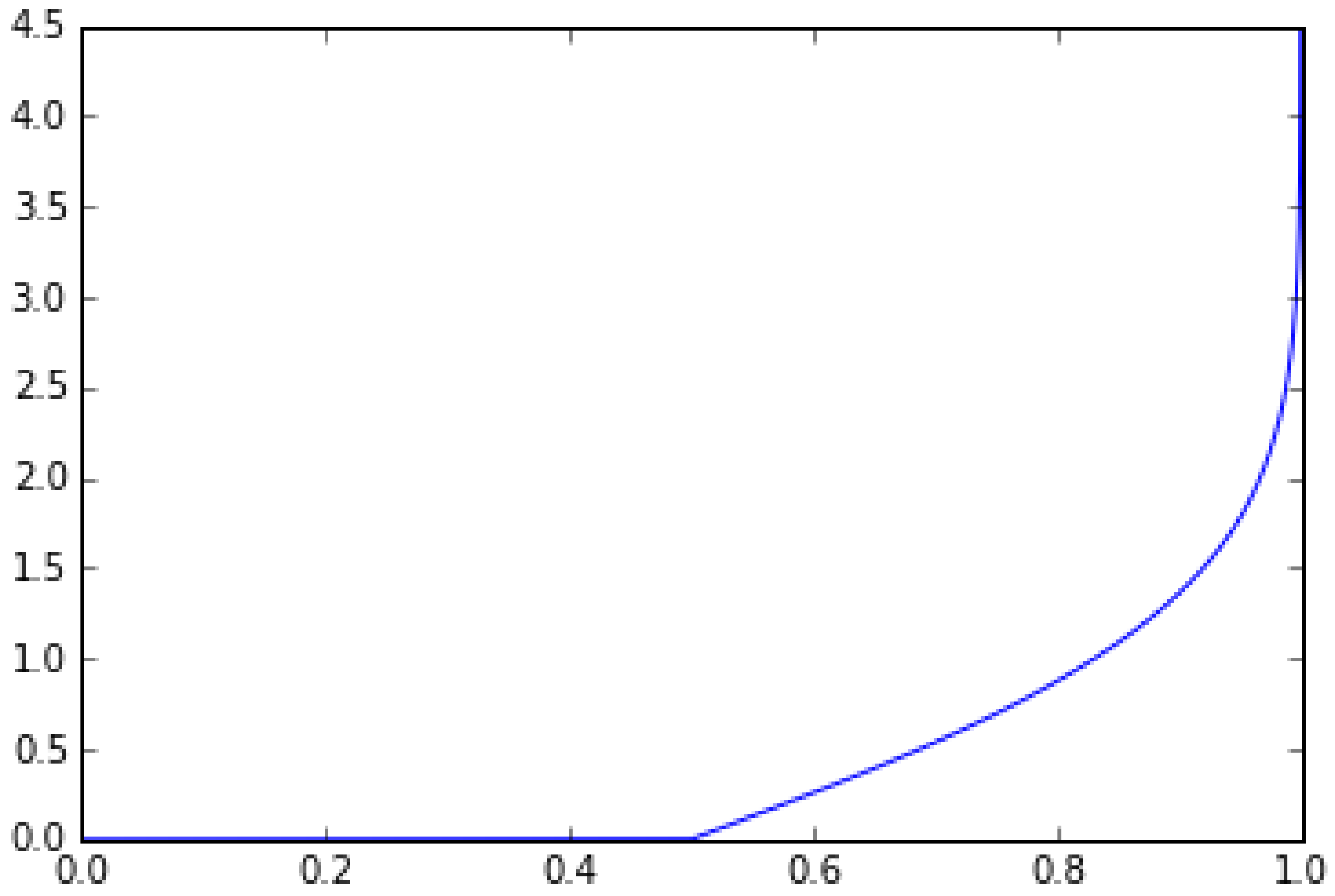
Pricing a Call Option - The Python Code

```
def f(u, S0, K, r, sigma, T):
    m       = (r - .5*sigma*sigma)*T
    s       = sigma*sqrt(T)
    f_u     = exp(-r*T) *
              np.maximum(S0*exp(scnorm.ppf(u, m, s))-K,0)
    return f_u

u       = rand(1000000)
f_u     = f(u,S0,K,r,sigma,T)

print mean(f_u)
```

Pricing a Call Option - The Integrand Function



Notebook

- **GitHub :** polyhedron-gdl;
- **Notebooks :** mcs_3;

Feynman–Kac formula

- The **Feynman–Kac formula** named after Richard Feynman and Mark Kac, establishes a link between parabolic partial differential equations (PDEs) and stochastic processes.
- It offers a method of solving certain PDEs by simulating random paths of a stochastic process. Conversely, an important class of expectations of random processes can be computed by deterministic methods.
- Consider the PDE

$$\frac{\partial u}{\partial t}(x,t)+\mu(x,t)\frac{\partial u}{\partial x}(x,t)+\tfrac{1}{2}\sigma^2(x,t)\frac{\partial^2 u}{\partial x^2}(x,t)-V(x,t)u(x,t)+f(x,t)=0$$

subject to the terminal condition

$$u(x,T)=\psi(x)$$

Feynman–Kac formula

- Then the Feynman–Kac formula tells us that the solution can be written as a conditional expectation

$$u(x,t) = E^Q\left[\int_t^T e^{-\int_t^r V(X_\tau,\tau)\,d\tau} f(X_r,r)dr + e^{-\int_t^T V(X_\tau,\tau)\,d\tau}\psi(X_T)\middle| X_t = x\right]$$

under the probability measure $\mathbb{Q}$ such that X' is an Ito process driven by the equation

$$dX = \mu(X,t)\,dt + \sigma(X,t)\,dW^{\mathbb{Q}}$$

Valuing a derivative contract

- A derivative can be perfectly replicated by means of a self-financing dynamic portfolio whose value exactly matches all of the derivative flows in every state of the world. This approach shows that the values of the derivative (and of the portofolio) solves the following PDE

$$\frac{\partial f}{\partial t} + \frac{\partial f}{\partial S} rS + \frac{1}{2}\frac{\partial f}{\partial S}\sigma^2 S^2 = fr \tag{1}$$

with the terminal condition at T that is the derivative's payoff

$$f(T, S(T)) = payoff$$

Valuing a derivative contract

- According to the Feynmann-Kac formula, if $f(t_0, S(t_0))$ solves the B-S PDE, then it is also solution of

$$f(t_0, S(t_0)) = \mathbb{E}\left[e^{-r(T-t_0)} f(T, S(T) | \mathcal{F}_{t_0}\right]$$

- i.e. it's the expected value of the discounted payoff in a probability measure where the evolution of the asset is

$$dS = rSdt + \sigma Sdw$$

- This probability measure is the **Risk Neutral Measure**

Valuing a derivative contract

- Since there exist such an equivalence, we can compute option prices by means of two numerical methods
- PDE: finite difference (explicit, implicit, crank-nicholson)suitable for optimal exercise derivatives;
- Integration
  - Quadrature Methods;
  - Monte Carlo Methods

Outline

Scenario Nomenclature

- As we have have repeated ad nauseam, the value of a derivative security with payoffs at a known time T is given by the expectation of its payoff, normalized with the numeraire asset;
- The value of an European derivative whose payoff depends on a single underlying process, $S(t)$, is given by

$$\boxed{V[S(0),0] = B(0)E^{B}\left[\frac{V(S(T),T)}{B(T)}\right]} \tag{2}$$

where all stochastic processes in this expectation are consistent with the measure induced by the numeraire asset $B(t)$.

Scenario Nomenclature

- We consider an underlying process $S(t)$ described by the sde

$$\boxed{dS(t) = a(S,t)dt + b(S,t)dW} \tag{3}$$

- A scenario is a set of values $\hat{S}^j(t_i)$, $i = 1, \ldots, I$ that are an approximation to the $j - th$ realization,$S^j(t_i)$, of the solution of the sde evaluated at times $0 \le t_i \le T$, $i = 1, \ldots, I$;
- A scenario is also called a trajectory
- A trajectory can be visualized as a line in the state-vs-time plane describing the path followed by a realization of the stochastic process (actually by an approzimation to the stochastic process).

Scenario Nomenclature

- For example, the Black and Scholes model assumes a market in which the tradable assets are:
  - A risky asset, whose evolution is driven by a geometric brownian motion

$$dS = \mu S dt + \sigma S dw \Rightarrow S(T) = S(t_0) e^{(\mu - \frac{1}{2}\sigma^2)(T - t_0) + \sigma[w(T) - w(t_0)]} \quad (4)$$

  - the money market account, whose evolution is deterministic

$$dB = Brdt \Rightarrow B(T) = B(t_0) e^{r(T - t_0)} \quad (5)$$

Scenario Contruction

- There are several ways to construct scenario for pricing
- Constructing a path of the solution to the SDE at times ti by exact advancement of the solution;
  - This method is only possible if we have an analytical expression for the solution of the stochastic differential equation
- Approximate numerical solution of the stochastic differential equation;
  - This is the method of choice if we cannot use the previous one; Just as in the case of ODE there are numerical techniques for discretizing and solving SDE.

Exact Solution Advancement

- Example: Log-normal process with constant drift and volatility

$$\text{SDE} \Rightarrow \frac{dS}{S} = \mu dt + \sigma dw$$

$$\Downarrow$$

$$\text{SOLUTION} \Rightarrow S(T) = S(t_0) e^{(\mu - \frac{1}{2}\sigma^2)(T - t_0) + \sigma[w(T) - w(t_0)]}$$

- How to obtain a sequence of Wiener process?

$$w(t_i) = w(t_{i-1}) + \sqrt{t_i - t_{i-1}} Z \quad Z \sim N(0, 1)$$

Exact Solution Advancement

- Defining the outcomes of successive drawings of the random variable Z corresponding to the $j-th$ trajectory by Z_i^j, we get the following recursive expression for the $j-th$ trajectory of $S(t)$:

$$S^j(t_i) = S^j(t_{i-1}) exp \left[\left(\mu - \frac{1}{2}\sigma^2 \right)(t_i - t_{i-1}) + \sigma \sqrt{t_i - t_{i-1}} Z_i^j \right]$$

Exact Solution Advancement

- Some observations are in order...
- The set $w(t_i)$ must be viewed as the components of a vector of random variables with a multidimensional distribution. This means that for a fixed j Z_j^i are realizations of a multidimensional standard normal random variable which happen to be independent;
- Wheter we view the Z_j^i as coming from a multidimensional distribution of independent normals or as drawings from a single one-dimensional distribution does not affect the outcome as long as the Z_j^i are generated from pseudo-random numbers;
- This distinction, however, is conceptually important and **it becomes essential if we generate the Z_j^i not from pseudo-random numbers but from quasi-random sequences**.

Numerical Integration of SDE

- The numerical integration of the SDE by finite difference is another way of generating scenarios for pricing;
- In the case of the numerical integration of ordinary differential equations by finite differences the numerical scheme introduces a discretization error that translates into the numerical solution differing from the exact solution by an amount proportional to a power of the time step.
- This amount is the truncation error of the numerical scheme.

Numerical Integration of SDE

- In the case of the numerical integration of SDE by finite differences the interpretation of the numerical error introduced by the discretization scheme is more complicated;
- Unlike the case of ODE where the only thing we are interested in computing is the solution itself, when dealing with SDE there are two aspects that interest us:
    - One aspect is the accuracy with which we compute the trajectories or paths of a realization of the solution
    - The other aspect is the accuracy with which we compute functions of the process such as expectations and moments.

Numerical Integration of SDE

- The order of accuracy with which a given scheme can approximate trajectories of the solution is not the same as the accuracy with which the same scheme can approximate expectations and moments of functions of the trajectories;
- The convergence of the numerically computed trajectories to the exact trajectories is called strong convergence and the order of the corresponding numerical scheme is called order of strong convergence;
- The convergence of numerically computed functions of the stochastic process to the exact values is called weak convergence and the related order is called order of weak convergence.

Numerical Integration of SDE

- We assume that the stock price St is driven by the stochastic di§erential equation (SDE)

$$dS(t) = \mu(S,t)dt + \sigma(S,t)dW_t \tag{6}$$

  where W_t is, as usual, Brownian motion.
- We simulate S_t over the time interval $[0;T]$, which we assume to be is discretized as $0 = t_1 < t_2 < \cdots < t_m = T$, where the time increments are equally spaced with width dt.
- Equally-spaced time increments is primarily used for notational convenience, because it allows us to write $t_i - t_{i-1}$ as simply dt. All the results derived with equally-spaced increments are easily generalized to unequal spacing.

Numerical Integration of SDE

- **Euler Scheme**
- The simplest way to discretize the process in Equation (6) is to use Euler discretization

$$\text{EULER} \Rightarrow \hat{S}(t_{i+1}) = \hat{S}(t_i) + \mu[\hat{S}(t_i), t_i]\Delta t + \sigma[\hat{S}(t_i), t_i]\left(w(t_{i+1}) - w(t_i)\right)$$

- **Milshstein Scheme**

$$\text{MILSHSTEIN} \Rightarrow \text{EULER} + \frac{1}{2}\sigma[\hat{S}(t_i)]\frac{\partial\sigma[\hat{S}(t_i)]}{\partial S}\left[\left(w(t_{i+1}) - w(t_i)\right)^2 - \Delta t\right]$$

This scheme is described in Glasserman and in Kloeden and Platen for general processes, and in Kahl and Jackel for stochastic volatility models. The scheme works for SDEs for which the coefficients $\mu(S_t)$ and $\sigma(S_t)$ depend only on S, and do not depend on t directly

Notebook

- **GitHub :** polyhedron-gdl;
- **Notebooks :** mcs_sde_solution;
- **Code :** mcs_sde_solution.py;

The Brownian Bridge

- Assume you have a Wiener process defined by a set of time-indexed random variables $W(t_1), W(t_2), ..., W(t_n)$.
- How do you insert a random variable $W(t_k)$ where $t_i \leq t_k \leq t_{i+1}$ into the set in such a manner that the resulting set still consitutes a Wiener process?
- The answer is: with a Brownian Bridge!
- The Brownian Bridge is a sort of interpolation tat allows you to introduce intermediate points in the trajectory of a Wiener process.

The Brownian Bridge

- Brownian Bridge Construction
- Given $W(t)$ and $W(t + \delta t_1 + \delta t_2)$ we want to find $W(t + \delta t_1)$;
- We assume that we can get the middle point by a weighted average of the two end points plus an independent normal random variable:

$$W(t + \delta t_1) = \alpha W(t) + \beta W(t + \delta t_1 + \delta t_2) + \lambda Z$$

where α, β and λ are constants to be determined and Z is a standard normal random variable.

The Brownian Bridge

- We have to satisfy the following conditions:

$$\begin{cases} cov[W(t+\Delta t_1), W(t)] = min(t+\Delta t_1, t) = t \\ cov[W(t+\Delta t_1), W(t+\Delta t_1+\Delta t_2)] = t+\Delta t_1 \\ var[W(t+\Delta t_1)] = t+\Delta t_1 \end{cases}$$

$$\begin{cases} \alpha+\beta = 1 \\ \alpha t + \beta(t+\Delta t_1+\Delta t_2) = t+\Delta t_1 \\ \alpha^2 t + 2\alpha\beta t + \beta^2(t+\Delta t_1+\Delta t_2) + \lambda^2 = t+\Delta t_1 \end{cases}$$

The Brownian Bridge

- which are equivalent to:

$$\begin{cases} \alpha = \frac{\Delta t_2}{\Delta t_1 + \Delta t_2} \\ \beta = 1 - \alpha \\ \gamma = \sqrt{\Delta t_1 \alpha} \end{cases}$$

The Brownian Bridge

- We can use the brownian bridge to generate a Wiener path and then use the Wiener path to produce a trajectory of the process we are interested in;
- The simplest strategy for generating a Wiener path using the brownian bridge is to divide the time span of the trajectory into two equal parts and apply the brownian bridge construction to the middle point. We then repeat the procedure for the left and right sides of the time interval.

The Brownian Bridge

- Notice that as we fill in the Wiener path, the additional variance of the normal components we add has decreasing value;
- Of course the total variance of all the Wiener increments does not depend on how we construct the path, however the fact that in the brownian bridge approach we use random variables that are multiplied by a factor of decreasing magnitude means that the importance of those variables also decreases as we fill in the path;
- The dimension of the random variables with larger variance need to be sampled more efficiently than the dimension with smaller variance;

Notebook

- **GitHub :** polyhedron-gdl;
- **Code :** mcs_brownian_bridge.py;

Outline

Outline

Choleski Decomposition

- The **Choleski Decomposition** makes an appearance in Monte Carlo Methods where it is used to simulating systems with correlated variables.
- Cholesky decomposition is applied to the correlation matrix, providing a lower triangular matrix A, which when applied to a vector of uncorrelated samples, u, produces the covariance vector of the system. Thus it is highly relevant for quantitative trading.
- The standard procedure for generating a set of correlated normal random variables is through a linear combination of uncorrelated normal random variables;
- Assume we have a set of n independent standard normal random variables Z and we want to build a set of n correlated standard normals Z' with correlation matrix Σ

$$Z' = AZ, \qquad AA^t = \Sigma$$

Choleski Decomposition

- We can find a solution for A in the form of a triangular matrix

$$\begin{pmatrix} A_{11} & 0 & \dots & 0 \\ A_{21} & A_{22} & \dots & 0 \\ \vdots & \vdots & \ddots & \dots \\ A_{n1} & A_{n2} & \dots & A_{nn} \end{pmatrix}$$

- **diagonal elements**

$$a_{ii} = \sqrt{\Sigma_{ii} - \sum_{k=1}^{i-1} a_{ik}^2}$$

- **off-diagonal elements**

$$a_{ij} = \frac{1}{a_{ii}} \left(\Sigma_{ij} - \sum_{k=1}^{i-1} a_{ik} a_{jk} \right)$$

Choleski Decomposition

- For example, for a two-dimension random vector we have simply

$$A = \begin{pmatrix} \sigma_1 & 0 \\ \sigma_2\rho & \sigma_2\sqrt{1-\rho^2} \end{pmatrix}$$

- say one needs to generate two correlated normal variables x_1 and x_2
- All one needs to do is to generate two uncorrelated Gaussian random variables z_1 andz_2 and set

$$x_1 = z_1$$

$$x_2 = \rho z_1 + \sqrt{1-\rho^2} z_2$$

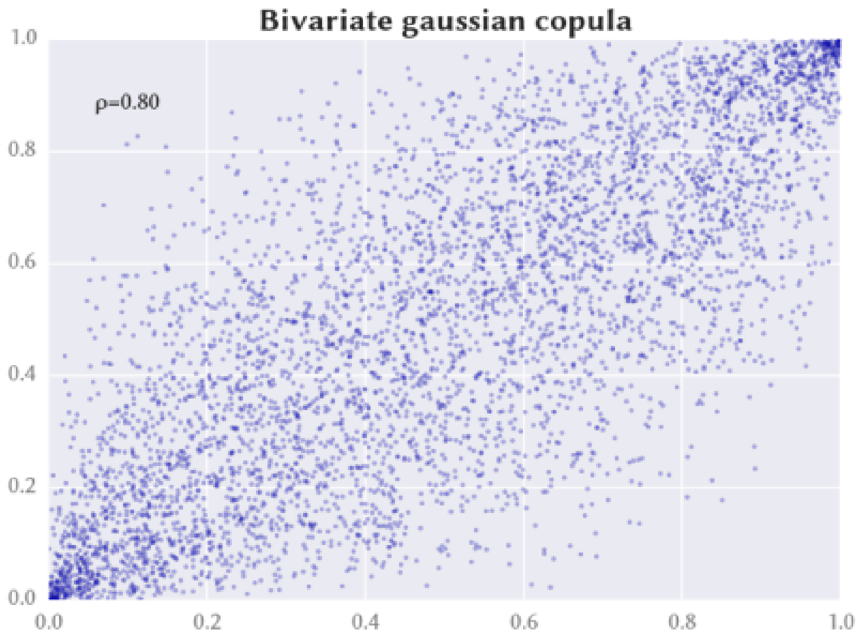
Copula Functions

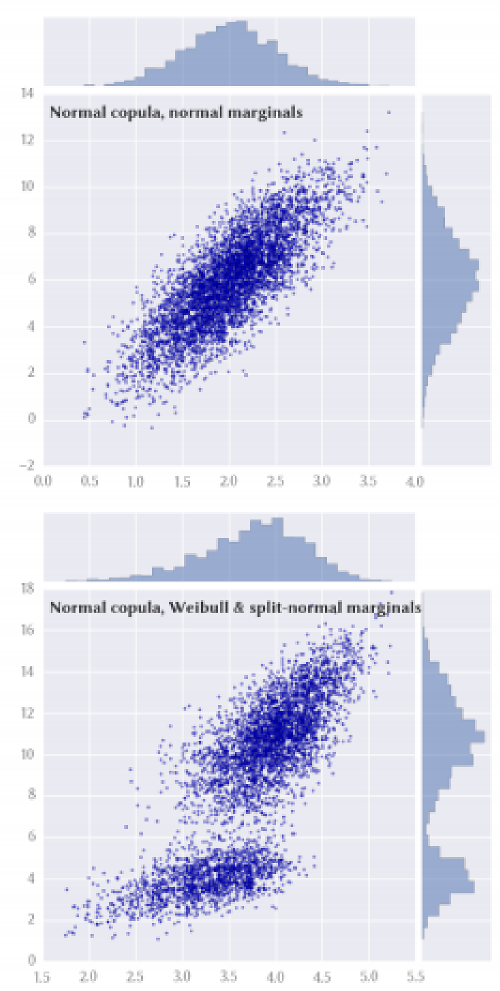
- **Why Copula?**
- A copula is a function that links univariate marginals to their multivariate distribution;
- Non-linear dependence
- Be able to measure dependence for heavy tail distributions
- Very flexible: parametric, semi-parametric or non-parametric
- Be able to study asymptotic properties of dependence structures
- Many others... (ask to Cherubini for everything you would like to know about copula)

Copula Functions

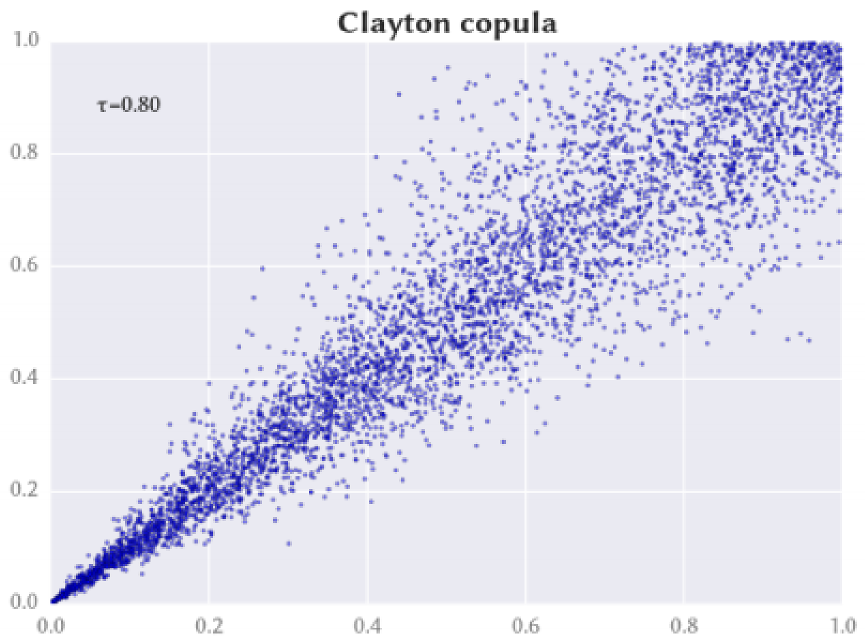
- In most applications, the distribution is assumed to be a multivariate gaussian or a log-normal distribution for tractable calculus, even if the gaussian assumption may not be appropriate.
- Copulas are a powerful tool for finance, because the modelling problem can be splitted into two steps:
  • the first step deals with the identification of the marginal distributions;
  • and the second step consists in defining the appropriate copula in order to represent the dependence structure in a good manner.

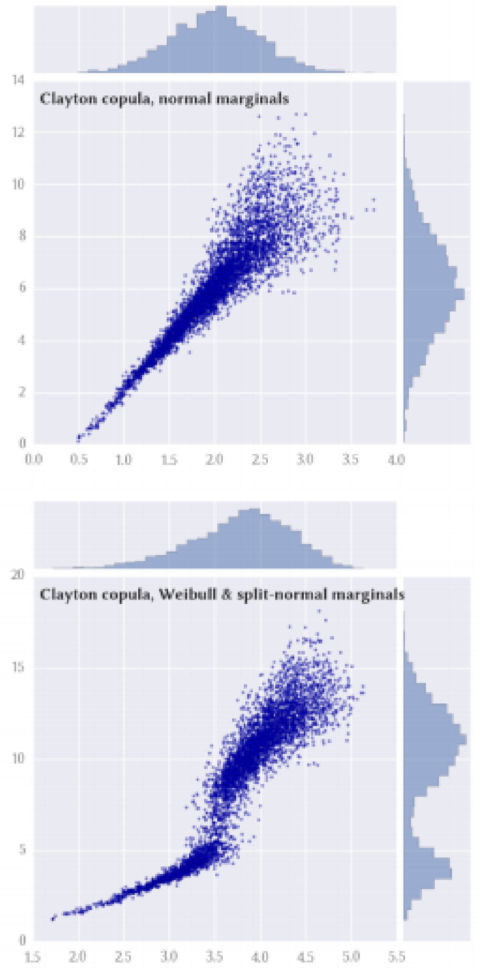
Copula Functions





Copula Functions





Notebook

- **GitHub :** polyhedron-gdl;
- **Notebook :** mcs_multi_asset_path;

Outline

CIR Model

- In this section, we consider the stochastic short rate model MCIR85 of Cox- Ingersoll-Ross which is given by the SDE:

$$\boxed{dr_t = \kappa_r(\theta_r - r_t)dt + \sigma_r\sqrt{r_t}dZ_t} \tag{7}$$

- To simulate the short rate model, it has to be discretized. To this end, we divide the given time interval $[0, T]$ in equidistant sub-intervals of length t such that now $t \in \{0, \Delta t, 2\Delta t, \dots, T\}$, i.e. there are $M + 1$ points in time with $M = T/t$.
- The exact transition law of the square-root diffusion is known. Consider the general square- root diffusion process

$$dx_t = \kappa(\theta - x_t)dt + \sigma\sqrt{x_t}dZ_t \tag{8}$$

CIR Model

- It can be show that x_t, given x_s with $s = t - \Delta t$, is distributed according to

$$x_t = \frac{\sigma^2(1 - e^{-\kappa\Delta t})}{4\kappa} \chi_d'^2 \left(\frac{4^{-\kappa\Delta t}}{\sigma^2(1 - e^{-\kappa\Delta t})} x_s \right)$$

where $\chi_d'^2$ denotes a non-central chi-squared random variable with

$$d = \frac{4\theta\kappa}{\sigma^2}$$

degrees of freedom and non-centrality parameter

$$l = \frac{4^{-\kappa\Delta t}}{\sigma^2(1 - e^{-\kappa\Delta t})} x_s$$

CIR Model

- For implementation purposes, it may be convenient to sample a chi-squared random variable χ_d^2 instead of a non-central chi-squared one, $\chi_d'^2$.
- If $d > 1$, the following relationship holds true

$$\chi_d'^2(l) = (z + \sqrt{l})^2 + \chi_{d-1}^2$$

  where z is an independent standard normally distributed random variable.
- Similarly, if $d \leq 1$, one has

$$\chi_d'^2(l) = \chi_{d+2N}^2$$

  where N is now a Poisson-distributed random variable with intensity $l/2$. For an algorithmic representation of this simulation scheme refer to Glasserman, p. 124.

CIR Model: Pricing ZCB

- A MC estimator for the value of the ZCB at t is derived as follows.
- Consider a certain path i of the I simulated paths for the short rate process with time grid $t \in \{0, \Delta t, 2\Delta t, \ldots, T\}$.
- We discount the terminal value of the ZCB, i.e. 1, step-by-step backward. For $t < T$ and $s = t - \Delta t$ we have

$$B_{s,i} = B_{t,i} e^{-\frac{r_t + r_s}{2}\Delta t}$$

- The MC estimator of the ZCB value at t is

$$B_t^{MC} = \frac{1}{I}\sum_{i=1}^{I} B_{t,i}$$

CIR Model: Pricing ZCB

- The present value of the ZCB in the CIR model takes the form:

$$B_0(T) = b_1(T)e^{-b_2(T)r_0}$$

where

$$b_1(T) = \left[\frac{2\gamma\, exp((\kappa_r + \gamma)T/2)}{2\gamma + (\kappa_r + \gamma)(e^{\gamma T} - 1)}\right]^{\frac{2\kappa_r\theta_r}{\sigma_r^2}}$$

$$b_2(T) = \frac{2(e^{\gamma T} - 1)}{2\gamma + (\kappa_r + \gamma)(e^{\gamma T} - 1)}$$

$$\gamma = \sqrt{\kappa_r^2 + 2\sigma_r^2}$$

Notebook

- **GitHub :** polyhedron-gdl;
- **Notebook :** mcs_cir;

Outline

Valuation of American Option by Simulation

- As we have seen Monte Carlo simulation is a flexible and powerful numerical method to value financial derivatives of any kind.
- However being a forward evolving technique, it is per se not suited to address the valuation of American or Bermudan options which are valued in general by backwards induction.
- Longstaff and Schwartz provide a numerically efficient method to resolve this problem by what they call Least-Squares Monte Carlo.
- The problem with Monte Carlo is that the decision to exercise an American option or not is dependent on the continuation value.

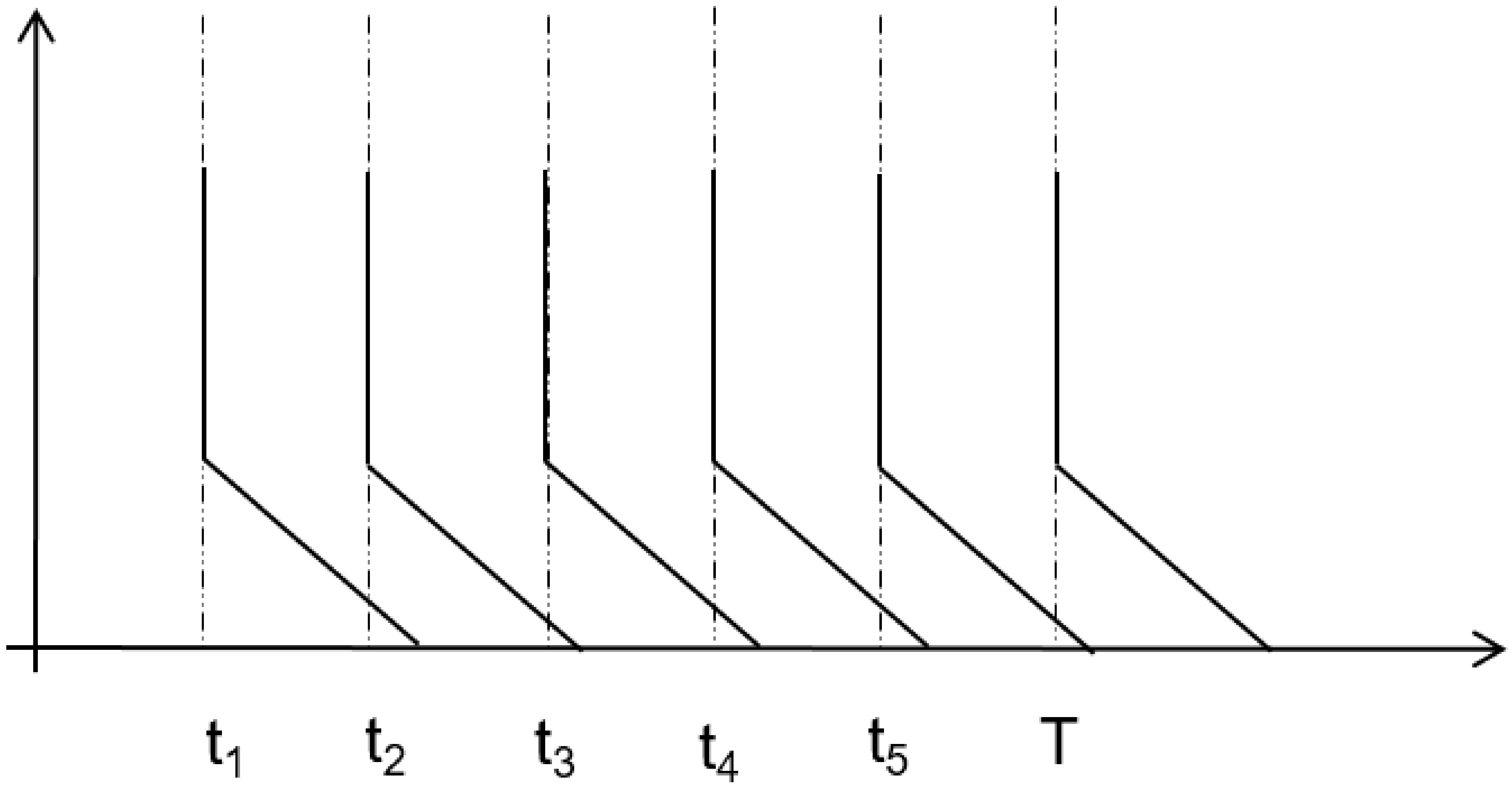
Valuation of American Option by Simulation

- Consider a simulation with $M + 1$ points in time and I paths.
- Given a simulated index level $S_{t,i}, t \in \{0, ..., T\}, i \in \{1, ..., I\}$, what is the continuation value $C_{t,i}(S_{t,i})$, i.e. the expected payoff of not exercising the option?
- The approach of Longstaff-Schwartz approximates continuation values for American options in the backwards steps by an ordinary least-squares regression.
- Equipped with such approximations, the option is exercised if the approximate continuation value is lower than the value of immediate exercise. Otherwise it is not exercised.

Valuation of American Option by Simulation

- In order to explain the metodology, let's start from a simpler problem.
- Consider a bermudan option which is similar to an american option, except that it can be early exercised once only on a specific set of dates.
- In the next figure, we can represent the schedule of a put bermudan option with strike K and maturity in 6 years. Each year you can choose whether to exercise or not ...

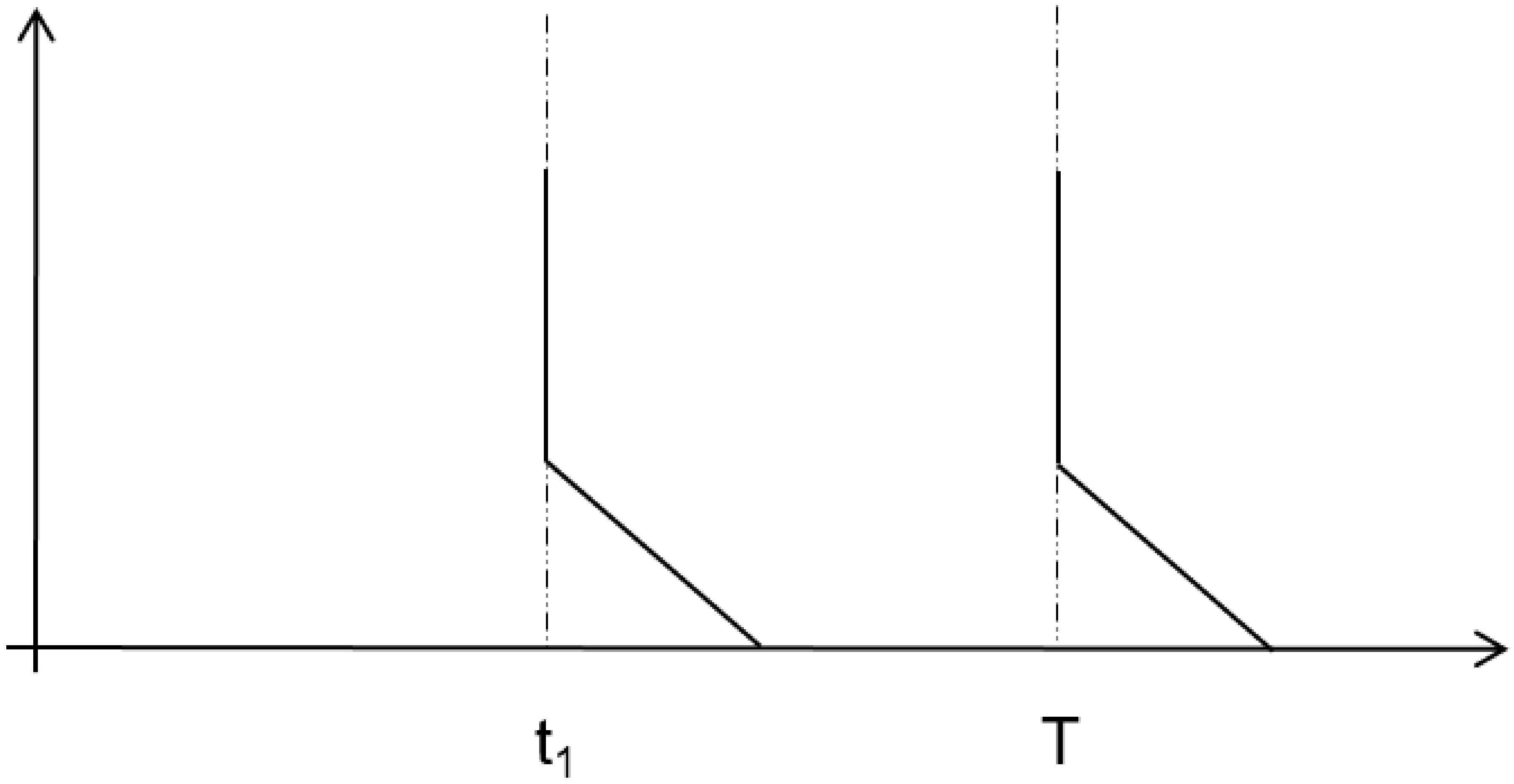
Valuation of American Option by Simulation



Valuation of American Option by Simulation

- Let's consider a simpler example: a put option which can be exercised early only once ...

Valuation of American Option by Simulation



Valuation of American Option by Simulation

- Can we price this product by means of a Monte Carlo? Yes we can! Let's see how.
- Let's implement a MC which actually simulates, besides the evolution of the market, what an investor holding this option would do (clearly an investor who lives in the risk neutral world). In the following example we will assume the following data, $S(T) =$, $K =$, $r =$, $\sigma =$, $t_1 = 1y$, $T = 2y$.
- We simulate that 1y has passed, computing the new value of the asset and the new value of the money market account

$$S(t_1 = 1y) = S(t_0)e^{(r - \frac{1}{2}\sigma^2)(t_1 - t_0) + \sigma\sqrt{t_1 - t_0}N(0,1)}$$

$$B(t_1 = 1y) = B(t_0)e^{r(t_1 - t_0)}$$

Valuation of American Option by Simulation

- At this point the investor could exercise. How does he know if it is convenient?
- In case of exercise he knows exactly the payoff he's getting.
- In case he continues, he knows that it is the same of having a European Put Option.
- So, in mathematical terms we have the following payoff in t_1

$$\max\left[K - S(t_1), P(t_1, T; S(t_1), K)\right]$$

where $P(t_1, T; S(t_1), K)$ is the price of a Put which we compute analytically! In the jargon of american products, P is called the continuation value, i.e. the value of holding the option instead of early exercising it.

Valuation of American Option by Simulation

- So the premium of the option is the average of this discounted payoff calculated in each iteration of the Monte Carlo procedure.

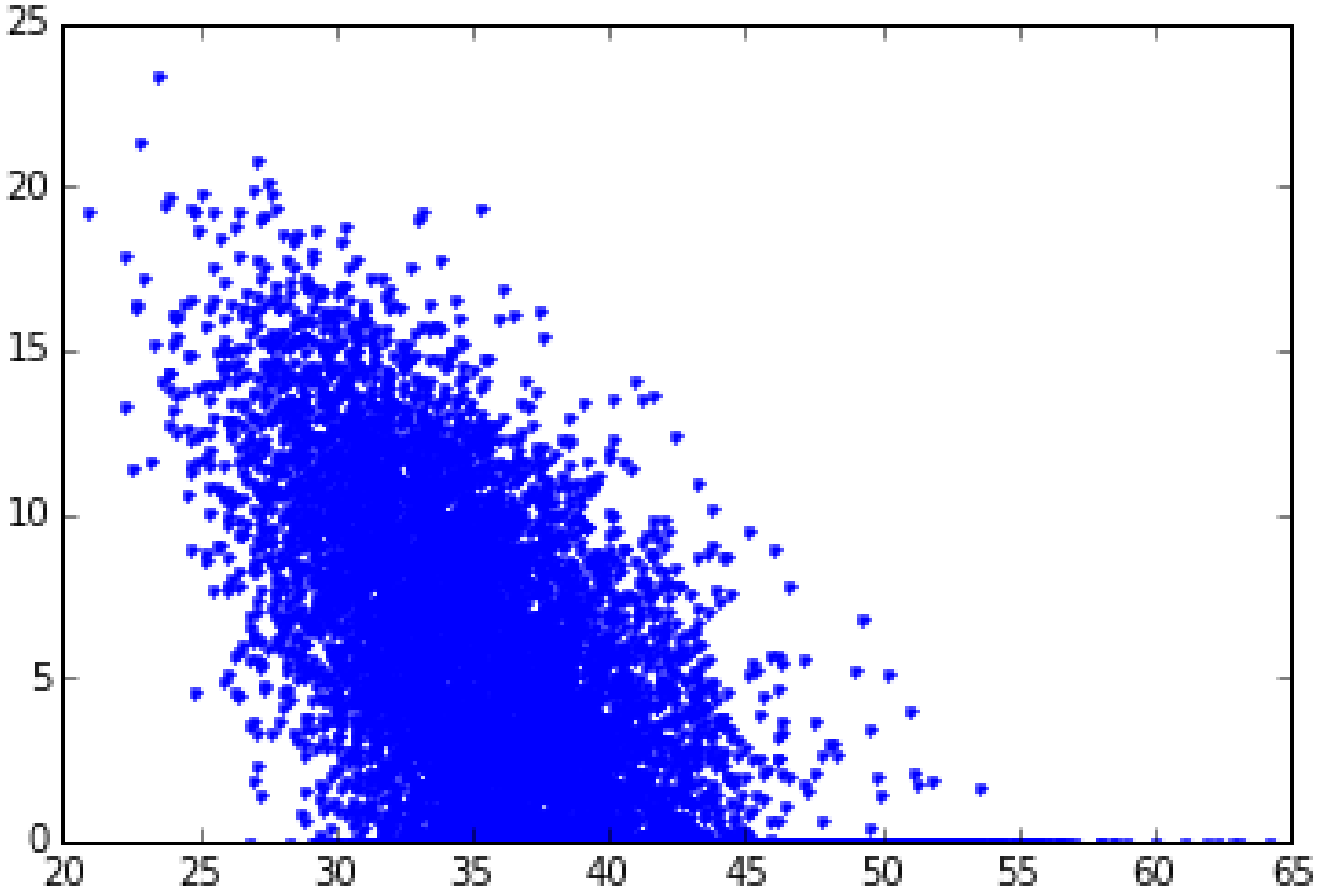
$$\frac{1}{N}\sum_i \max\left[K - S_i(t_1), P(t_1, T; S_i(t_1), K)\right]$$

- Some considerations are in order.
- We could have priced this product because we have an analytical pricing formula for the put. What if we didn't have it?

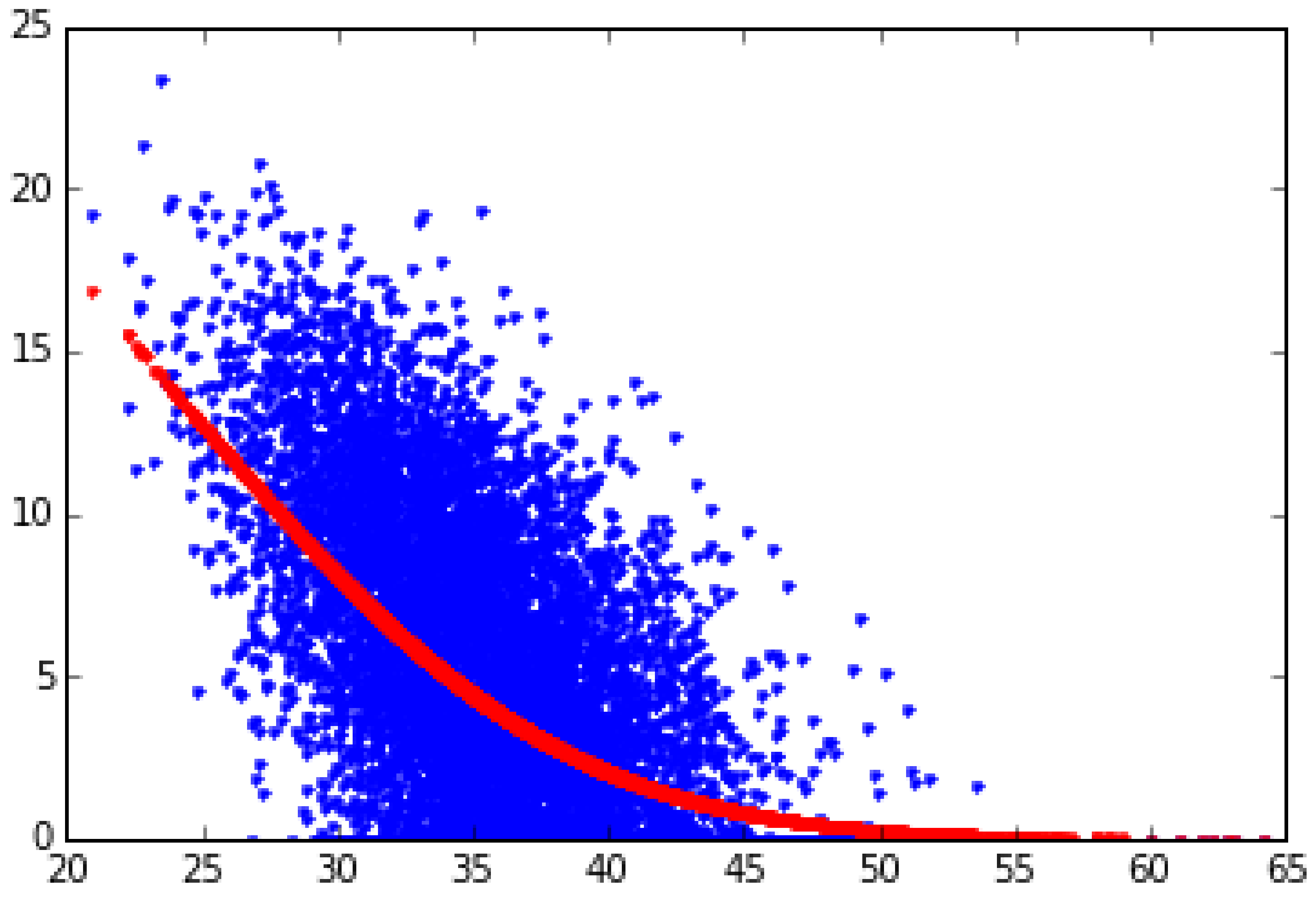
Valuation of American Option by Simulation

- Brute force solution: for each realization of $S(t_1)$ we run another Monte Carlo to price the put.
- This method (called Nested Monte Carlo) is very time consuming. For this very simple case it's time of execution grows as N^2, which becomes prohibitive when you deal with more than one exercise date!
- Let's search for a finer solution analyzing the relationship between the continuation value (in this very simple example) and the simulated realization of S at step t_1.
- let's plot the discounted payoff at maturity, P_i, versus $S_i(t_1)$ ...

Valuation of American Option by Simulation

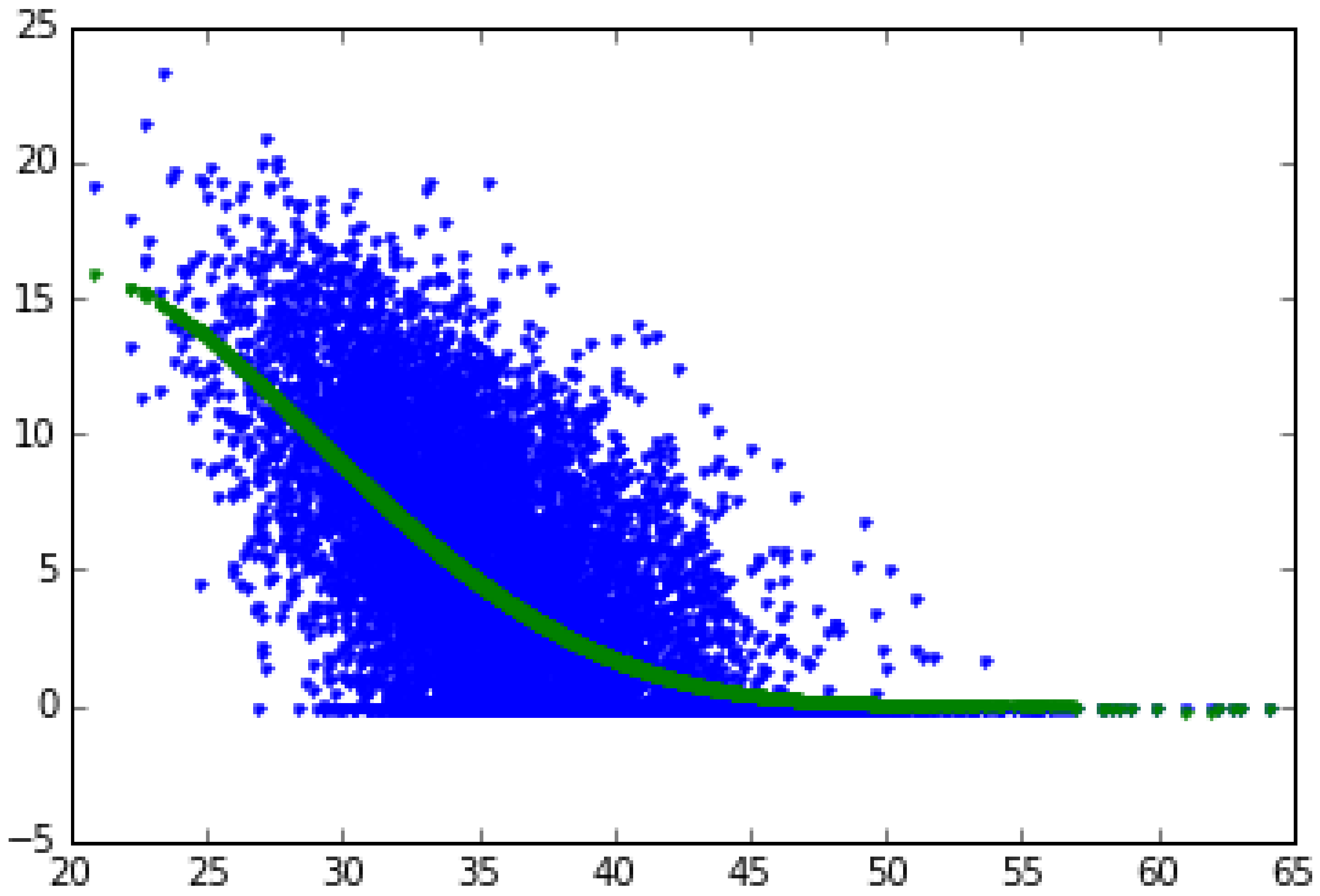


Valuation of American Option by Simulation

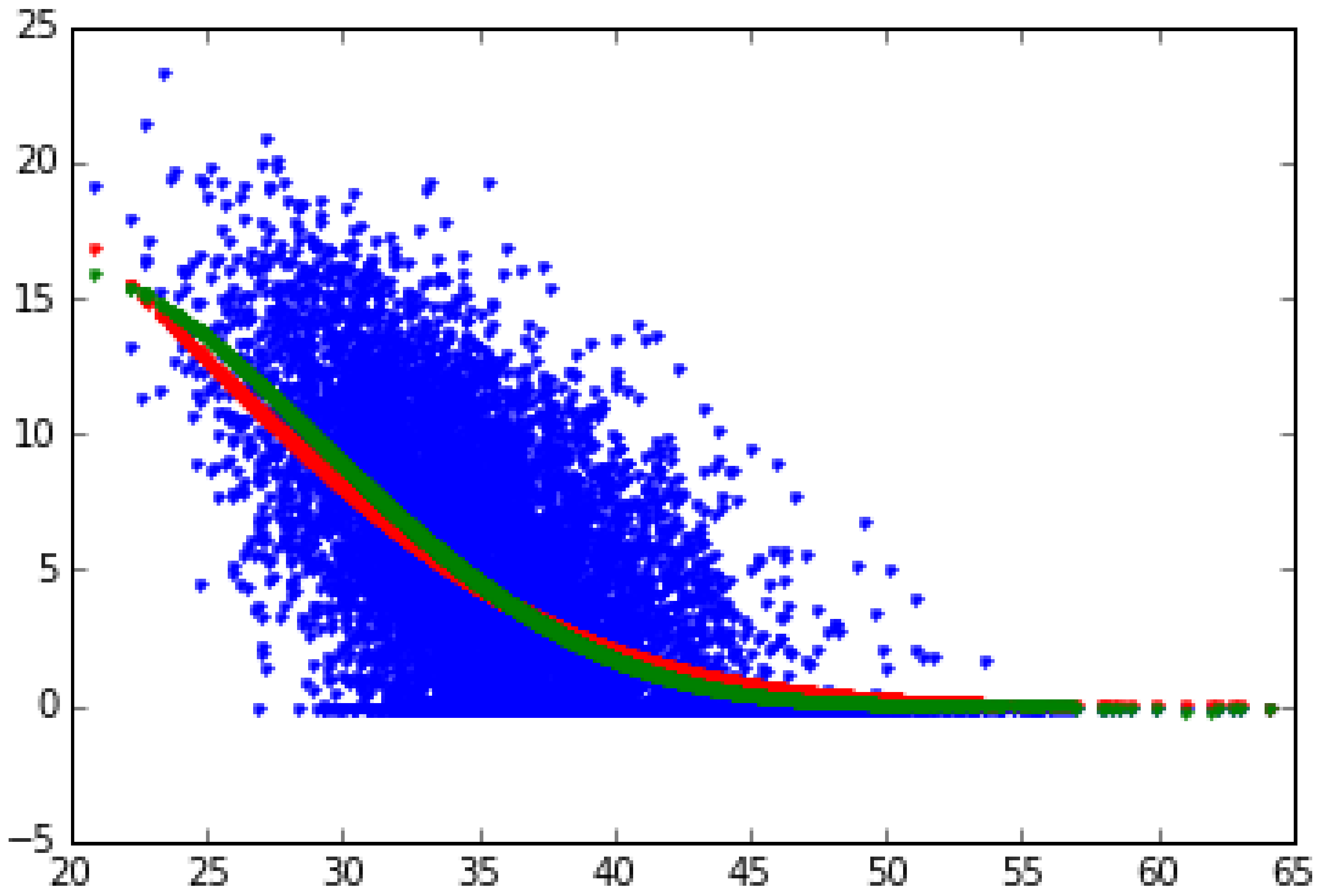


- As you can see, the analytical price of the put is a curve which kinds of interpolate the cloud of Monte Carlo points.
- This suggest us that the price at time t_1 can be computed by means of an average on all discounted payoff (i.e. the barycentre of the cloud made of discounted payoff)
- So maybe... the future value of an option can be seen as the problem of finding the curve that best fits the cloud of discounted payoff (up to date of interest)!!!
- In the next slide, for example, there is a curve found by means of a linear regression on a polynomial of 5th order...

Valuation of American Option by Simulation



Valuation of American Option by Simulation



Valuation of American Option by Simulation

- We now have an empirical pricing formula for the put to be used in my MCS

$$P(t_1, T, S(t_1), K) = c_0 + c_1 S(t_1) + c_2 S(t_1)^2 + c_3 S(t_1)^3 + c_4 S(t_1)^4 + c_5 S(t_1)^5$$

- The formula is obviously fast, the cost of the algorithm being the best fit.
- Please note that we could have used any form for the curve (not only a plynomial).
- This method has the advantage that it can be solved as a linear regression, which is fast.

The Longstaff-Schwartz Algorithm

- The major insight of Longstaff-Schwartz is to estimate the continuation value $C_{t,i}$ by ordinary least-squares regression, therefore the name "Least Square Monte Carlo" for their algorithm;
- They propose to regress the I continuation values $Y_{t,i}$ against the I simulated index levels $S_{t,i}$.
- Given D basis functions b with $b_1, \ldots, b_D : \mathbb{R}^D \rightarrow \mathbb{R}$ for the regression, the continuation value $C_{t,i}$ is according to their approach approximated by:

The Longstaff-Schwartz Algorithm

- Given D basis functions b with $b_1, \ldots, b_D : \mathbb{R}^D \to \mathbb{R}$ for the regression, the continuation value $C_{t,i}$ is according to their approach approximated by:

$$\hat{C}_{t,i} = \sum_{d=1}^{D} \alpha^{\star}_{d,t} b_d(S_{t,i}) \quad (1)$$

- The optimal regression parameters $\alpha^{\star}_{d,t}$ are the result of the minimization

$$\min_{\alpha_{1,t},\ldots,\alpha_{D,t}} \frac{1}{I} \sum_{i=1}^{I} \left(Y_{t,i} - \sum_{d=1}^{D} \alpha_{d,t} b_d(S_{t,i}) \right)^2$$

The Longstaff-Schwartz Algorithm

- Simulate I index level paths with $M+1$ points in time leading to index level values $S_{t,i}, t \in \{0, ..., T\}, i \in \{1, ..., I\}$;
- For $t = T$ the option value is $V_{T,i} = h_T(S_{T,i})$ by arbitrage
- Start iterating backwards $t = T - \Delta t, \ldots, \Delta t$:
    - regress the $T_{t,i}$ against the $S_{t,i}, i \in \{1, \ldots, I\}$, given D basis function b
    - approximate $C_{t,i}$ by $\hat{C}_{t,i}$ according to (1) given the optimal parameters $\alpha^\star_{d,t}$ from (2)
    - set

$$V_{t,i} = \begin{cases} h_t(S_{t,i}) & \text{if } h_t(S_{t,i}) > \hat{C}_{t,i} \quad \text{exercise takes place} \\ Y_{t,i} & \text{if } h_t(S_{t,i}) \leq \hat{C}_{t,i} \quad \text{no exercise takes place} \end{cases}$$

  repeat iteration steps until $t = \Delta t$;
- for $t = 0$ calculate the LSM estimator

$$\hat{V}_0^{LSM} = e^{-r\Delta t} \frac{1}{I} \sum_{i=1}^{I} V_{\Delta t,i}$$

Notebook

- **GitHub :** polyhedron-gdl;
- **Notebook :** mcs_american;

Outline

Outline