Modèles financiers en temps continu

Introduction to derivative pricing by Monte Carlo

2014-2014
Derivative Pricing by Monte Carlo

- We consider a contingent claim of maturity T (e.g. an equity option on one or several underlying assets, within a Black-Scholes framework)

- Its price at t is given by:

\[ P(t) = P(t, S) = E_Q \left[ e^{-rT} \text{payoff}(T) \mid S(t) = S \right] \]

where \( r \) is the (constant, deterministic) risk free rate and \( S(t) \) the underlying price(s)

- The principle of Monte Carlo pricing is to approximate numerically this expectation by:

\[ P(t) \approx \frac{1}{\# Simul} e^{-rT} \sum_{scen=1}^{\# Simul} \text{payoff}_{scen}(T) \]

where \( \text{payoff}_i(T) \) are realisations of i.i.d. random variables having the same distribution as the payoff of the derivative
This is a simple application of the law of large numbers, that says that if $X_i \sim X$, $i = 1, \ldots, N$, are i.i.d. with $E(X) < \infty$, then:

$$\frac{1}{N} \sum_{i=1}^{N} X_i \rightarrow E(X)$$

The idea is hence to find algorithms to “simulate” i.i.d. realisations of the payoff of the option.

As the expectation is taken under the risk-neutral measure, the distribution of the payoff has to be considered under this measure.

In practice, we often don’t know the payoff distribution (it is generally not given explicitly), and this is precisely in these cases that Monte Carlo simulations become helpful.

In the subsequent, we see how to simulate such payoffs in an i.i.d. way.
Simulation of a Brownian motion:

- A simple method consists to simulate normal distributions:

- If $X_i, i \leq n$, are independent standard Gaussian variables, and if we define the sequence:

$$S_0 = 0, S_{n+1} = S_n + \sqrt{\delta} X_n$$

then $(S_0, S_1, \ldots, S_n)$ has the same distribution as:

$$(W_0, W_\delta, W_{2\delta}, \ldots, W_{n\delta})$$

- More generally for every function $g(x)$, $(g(S_0), g(S_1), \ldots, g(S_n))$ has the same distribution as:

$$(g(W_0), g(W_\delta), g(W_{2\delta}), \ldots, g(W_{n\delta}))$$

- This already provides a way to approximate numerically European options in the framework of models with explicit solutions of the EDS governing the underlying – which is the case of the Black-Scholes model.
Simulation of a Brownian motion:

- In practice, we might need to price exotic options with payoff function of the underlying at predefined instants $0 = t_0 < t_1 < \cdots < t_k$, requiring simulating one (or several) Brownian motion(s) at these instants.
- To simulate a standard B.M. at instants $0 = t_0 < t_1 < \cdots < t_k$:
  - Simulate $k$ rvs $\epsilon_i \ i.\ i.\ d. \sim N(0,1)$
  - Apply the following scheme:
    - $W(t_1) = \sqrt{t_1} \epsilon_1$
    - $W(t_2) = W(t_1) + \sqrt{t_2 - t_1} \epsilon_2 = \sqrt{t_1} \epsilon_1 + \sqrt{t_2 - t_1} \epsilon_2$
    - ...
    - $W(t_k) = \sum_{i=1}^{k} \sqrt{t_i - t_{i-1}} \epsilon_i$
Simulation within Black-Scholes model (1/2):

- We know that the solution of the EDS of the geometric Brownian is:
  \[ S(t) = S(0)e^{(r-\frac{1}{2}\sigma^2)t+\sigma W_t} \]  
  (1)

- So if we want to calculate the price of a European option with payoff
  \[ H(T) = g(S(t)_{t\in[0,T]}) \], it suffices to divide the time interval [0,T] in n subintervals of length \( \Delta t \), or to consider predefined instants \( t_1 < t_2 < \cdots t_k \) (depending on the problem...) and apply:
  
  - For each simulation:
    - Simulate \( W(t) \) on [0,T]
    - Apply transformation (1) above to get a simulation of \( S(t) \) on [0,T]
    - Calculate the payoff as \( g(S(t)_{t\in[0,T]}) \)
  
  - The option price is then obtained by taking the arithmetic average on simulations of the discounted payoff
Simulation within Black-Scholes model (2/2):

- Solution of the Black-Scholes EDS:
  \[ S(t) = S(0) e^{(r-\frac{1}{2}\sigma^2)t + \sigma W_t} \]  

- In practice, we generally directly simulate the Brownian motion with drift appearing in (1), \( Z_t = \left( r - \frac{1}{2} \sigma^2 \right) t + \sigma W_t \)

- This leads (e.g.) to:
  - Choose \( \Delta t = \frac{T}{n} \) for some \( n \) (depending on the problem...)
  - Simulate \( \epsilon_i \sim N(0,1) \) \( i.i.d. \) \( (i = 1, \ldots, n) \)
  - Initialisation \( Z_0 = 0 \)
  - \( Z_{i\Delta t} = Z_{(i-1)\Delta t} + \left( r - \frac{1}{2} \sigma^2 \right) \Delta t + \sigma \sqrt{\Delta t} \epsilon_i \)
  - \( S_{i\Delta t} = S(0) e^{Z_{i\Delta t}} \)
Example: fixed strike lookback option*

- Maturity: $T = 5$ years, $r = 2\%$, $\sigma = 20\%$
- Payoff:
  \[
  (\max_{t \in [0,5Y]} S(t) - K)^+
  \]
- Price within BS model based on a discretisation scheme with $\Delta t = 1/250$ and 500000 simulations leads to a price close to 0.3490

- Cf. Matlab code

* Remark that an analytical formula exists in the BS framework
Example:

Payoff on 1000 simulations
Not all models are characterized by explicit solutions of the EDS of the underlying asset(s) or rate(s)

When the EDS cannot be solved analytically, we need to use discretization methods

Let us consider a general Itô process:

\[ dX_t = \mu(t, X_t)dt + \sigma(t, X_t)dW_t \]

The simplest method is the **Euler discretisation**:

We subdivide \([0,T]\) into steps of size \(\Delta t = \frac{T}{N}\). Euler scheme corresponds to:

\[
\tilde{X}_{t_{i+1}} = \tilde{X}_{t_i} + \mu(\tilde{X}_{t_i}, t_i)(t_{i+1} - t_i) + \sigma(\tilde{X}_{t_i}, t_i)(W(t_{i+1}) - W(t_i)) \\
\tilde{X}_0 = X_0
\]
The Euler scheme derives from a simple integration rule and the definition of the Ito integral

\[
X_{t_{i+1}} = X_{t_i} + \int_{t_i}^{t_{i+1}} \mu(X_t, t) \, dt + \int_{t_i}^{t_{i+1}} \sigma(X_t, t) \, dW(t)
\]

where:
\[
\int_{t_i}^{t_{i+1}} \mu(X_t, t) \, dt \approx \int_{t_i}^{t_{i+1}} \mu(X_{t_i}, t_i) \, dt = \mu(X_{t_i}, t_i) \Delta t_i
\]

and:
\[
\int_{t_i}^{t_{i+1}} \sigma(X_t, t) \, dW(t) \approx \sigma(X_{t_i}, t_i) \Delta W(t_i)
\]

One can show that this scheme converges in \(L^2\) to the solution of the EDS, in the following sense:

\[\exists C > 0 \text{ s.t. } E \left[ \left| X_{t_k} - \tilde{X}_{t_k} \right|^2 \right] \leq C \Delta t \quad \forall k \in \{0, 1, \ldots, N - 1\}\]

(Maruyama, 1955)
Derivative Pricing by Monte Carlo

- If we want to calculate $E[g(X_T)]$, by the law of large numbers implies, if $(\tilde{X}_t)^(i)$ denote different independent occurrences of $(X_t)$, $i = 1, \ldots, N$, then

$$\frac{1}{N} \sum_{i=1}^{N} g(\tilde{X}_T(i))$$

provides an approximation of this expectation.

- One can show that within Euler scheme, $|E[g(\tilde{X}_T)] - E[g(X_T)]| \rightarrow 0$ with a speed of the order of $\frac{1}{n}$, where $n$ is the discretization step.

- In practice, a reasonable time step is the month (in the case of a time horizon of the order of 10 - 20 years), but it depends on the problem:
  - In that case $t_{i+1} - t_i = 1/12$

- Take care about the time unit of the model parameters!
  - In general, if given, all parameters correspond to the year as time unit (e.g. « annual » volatility etc), but take care if you estimate them from historical data.
Milstein scheme improves this discretization if \( \mu \) and \( \sigma \) do not depend on time

It is based on a Taylor development of \( \sigma(x) \) (based on Ito lemma)

It consists to the following approximation

\[
\tilde{X}_{t_{i+1}} = \tilde{X}_{t_i} + \mu(\tilde{X}_{t_i}) \Delta t_i + \sigma(\tilde{X}_{t_i}) \Delta W(t_i)
+ 0.5\sigma(\tilde{X}_{t_i}) \frac{d\sigma}{dx}(\tilde{X}_{t_i})(\Delta W(t_i)^2 - \Delta t_i)
\]

One can show that the scheme converges a.s. and in \( L^2 \), with a greater speed than the Euler scheme
Idea of the derivation of Milstein scheme:

Let $t_i, t_{i+1}$ be two consecutive points in the time discretization.

$$X_{t_{i+1}} = X_{t_i} + \int_{t_i}^{t_{i+1}} a(X_s)ds + \int_{t_i}^{t_{i+1}} b(X_s)dW_s.$$ 

We apply the Itô formula to the expressions $a(X_s)$ and $b(X_s)$, which are the coefficients in our SDE:

$$X_{t_{i+1}} = X_{t_i} + \int_{t_i}^{t_{i+1}} \left( a(X_t) + \int_{t_i}^{s} \left( a'(X_u)a(X_u) + \frac{1}{2} a''(X_u)b^2(X_u) \right) du \right) ds$$

$$+ \int_{t_i}^{t_{i+1}} \left( a'(X_u)b(X_u)dW_u \right) ds$$

$$+ \int_{t_i}^{t_{i+1}} \left( b(X_t) + \int_{t_i}^{s} \left( b'(X_u)a(X_u) + \frac{1}{2} b''(X_u)b^2(X_u) \right) du \right) ds$$

$$+ \int_{t_i}^{t_{i+1}} \left( b'(X_u)b(X_u)dW_u \right) dW_s.$$ 

(source: J. Palczewski, course computation in finance, Uni. Leeds):
Idea of the derivation (continued):

The differentials $dW$ and $dt$ are replaced by the corresponding discrete versions $\Delta W$ and $\delta t$.

We have

$$\delta t \cdot \delta t = O((\delta t)^2),$$
$$\Delta W \cdot \delta t = O((\delta t)^{3/2}),$$
$$\Delta W \cdot \Delta W = O(\delta t).$$

Therefore: If we are up for a method which converges strongly of order 1, we can neglect the double integrals from the previous slide, which are of type $dW_u \cdot ds$, $dW_s \cdot du$ and $du \cdot ds$.

(source: J. Palczewski, course computation in finance, Uni. Leeds)
Derivative Pricing by Monte Carlo

- Idea of the derivation (continued):

We then obtain

\[
X_{t_{i+1}} \approx X_{t_i} + \int_{t_i}^{t_{i+1}} a(X_{t_i}) ds \\
+ \int_{t_i}^{t_{i+1}} \left( b(X_{t_i}) + \int_{t_i}^{s} b'(X_u)b(X_u) dW_u \right) dW_s \\
\approx X_{t_i} + a(X_{t_i})\delta t + b(X_{t_i})\Delta W_i + \int_{t_i}^{t_{i+1}} \int_{t_i}^{s} b'(X_u)b(X_u) dW_u dW_s
\]

The first two summands in the equation above are well known from the Euler-Maruyama scheme. The third one is new.

We approximate the third term above by

\[
\int_{t_i}^{t_{i+1}} \int_{t_i}^{s} b'(X_u)b(X_u) dW_u dW_s \approx b'(X_{t_i})b(X_{t_i}) \int_{t_i}^{t_{i+1}} \int_{t_i}^{s} dW_u dW_s.
\]

- (source: J. Palczewski, course computation in finance, Uni. Leeds):
One can also accelerate convergence

- Ex: “antithetic variates” within Euler scheme:

\[
X_{t_i}^+ = X_{t_i}^+ + \mu(X_{t_i}^+, t_i)(t_{i+1} - t_i) + \sigma(X_{t_i}^+, t_i)\sqrt{t_{i+1} - t_i} \epsilon_i
\]

\[
X_{t_i}^- = X_{t_i}^- + \mu(X_{t_i}^-, t_i)(t_{i+1} - t_i) - \sigma(X_{t_i}^-, t_i)\sqrt{t_{i+1} - t_i} \epsilon_i
\]

- In practice, within a pricing tool:
  - One generates only ONE set of random numbers \( \epsilon_i \sim N(0,1) \) per simulation path
  - BUT cash-flows (payoffs) are projected along the TWO sets of simulation paths in parallel
  - Consider then for each couple of simulated path (with the “+” and with the “-” but the same \( \epsilon_i \)) the average payoff(s)
  - The result is finally a simple set of simulated payoffs, from which we take the mean
In practice, options might involve also (risk-free) interest rates

In that case, we work in a stochastic framework for interest rates, and we will see that the general pricing formula becomes:

$$P(t) = E_Q \left[ \sum_{i=1}^{N} \text{payoff} \left( t_i \right) \exp \left( - \int_{t}^{t_i} r(s) ds \right) \right]$$

where \( r(s) \) denotes the short rate at time \( s \) (see later)

Monte Carlo pricing becomes hence:

$$P(t) \approx \frac{1}{\text{NbSimul}} \sum_{\text{scen}=1}^{\text{NbSimul}} \left[ \sum_{i=1}^{N} \text{payoff}_{\text{scen}} \left( t_i \right) \prod_{k=1}^{i} \exp \left( -r_{\text{scen}} (t_i - t_{i-1}) \right) \right]$$

$$= \frac{1}{\text{NbSimul}} \sum_{\text{scen}=1}^{\text{NbSimul}} \left[ \sum_{i=1}^{N} \text{payoff}_{\text{scen}} \left( t_i \right) \frac{DF_{\text{sto}}_{\text{scen}}}{DF_{\text{sto}}_{\text{scen}}} \right]$$
Simulations
Basics about pseudo-Random Numbers Generation

- In order to perform the Monte-Carlo simulation of the processes by discretization (Euler or other), we need to be able to generate pseudo-random numbers

- Example:
  - if \( t_{i+1} - t_i = 1/12 \) (1 month), if projection horizon \( N = 20 \) years, \( \rightarrow \) 240 independent generations by scenario are required
  - \( \rightarrow \) for 10000 generated scenarios, this leads to 2 400 000 random numbers by Brownian motion present in an ESG

- \( \rightarrow \) we need to have a pseudo-random numbers generator with a sufficient cycle

- Pre-programmed function in numerical software like Matlab, SAS, R are in general sufficient for option pricing: rand, random, ranuni,…
  - It is important however to pay attention to their cycle, depending on the number of factors
  - It is becoming insufficient if about \( 10^9 \) simulations of random numbers are required
The first step is to construct pseudo-random numbers following a uniform law on [0,1]

A well-known method consists to use congruential generators

Simple linear congruential generators:

- The idea is to choose huge integers a, c, m, and to generate pseudo-random numbers by:

\[ x_{i+1} = (ax_i + c) \mod m \]

\[ u_{i+1} = \frac{x_{i+1}}{m} \]

for a choice of initial seed x1.
Some conditions on the numbers $a$, $c$, $m$ guarantee that the generator will generate a complete cycle:
- “Complete cycle” means that for any choice of initial seed, the $m-1$ subsequent generated numbers will all be different

If $c \neq 0$, sufficient conditions are given by:
- $c$ and $m$ are relatively prime
- Any prime number dividing $m$ also divides $a$
- $a-1$ is divisible by 4 if $m$ also is

If now $c=0$ and if $m$ is prime, a complete cycle can be generated for any seed $x_1 \neq 0$ if:
- $a^{m-1} - 1$ is a multiple of $m$
- $a^{j-1} - 1$ is not a multiple of $m$ for any $j = 1, \ldots, m-2$
In practice, a well known congruential generator corresponds to:

\[ m = 2^{31} - 1 = 2147483647, \quad a = 16807, \quad c=0 \]

(Park-Miller generator)

Remark: this choice of \( m \) corresponds to the greatest integer that can be stored in a 32 bits computer

- 1 bit for the sign \( \rightarrow \) 31 bits for the digits in a binary basis,
- This was important in the past when deciding to work with variables declared in the “long integer” type and not with the “double [precision]” type, which was more efficient in terms of computational speed
Other possibilities are given by:

<table>
<thead>
<tr>
<th>m</th>
<th>a</th>
</tr>
</thead>
<tbody>
<tr>
<td>$2^{31} - 1$</td>
<td>16 807</td>
</tr>
<tr>
<td>(= 2 147 483 647)</td>
<td>39 373</td>
</tr>
<tr>
<td></td>
<td>742 938 285</td>
</tr>
<tr>
<td></td>
<td>950 706 376</td>
</tr>
<tr>
<td></td>
<td>1 226 874 159</td>
</tr>
<tr>
<td>2 147 483 399</td>
<td>40 692</td>
</tr>
<tr>
<td>2 147 483 563</td>
<td>40 014</td>
</tr>
</tbody>
</table>
Simulations
Basics about pseudo-Random Numbers Generation

• In practice, as the sequence of generated numbers can be arbitrarily close to e \( m \), by multiplying these by \( a \) during the generation, we arrive always to an over-flow if we work with an “integer” type
  
  → in order to avoid overflows, it is possible to decompose \( m \) into:

  \[
  m = a \cdot q + r,
  \]
  
  where \( r = m \mod a \).

• In that case, we use the fact that

  \[
  ax_i \mod m = a(x_i \mod q) - \left\lfloor \frac{x_i}{q} \right\rfloor r + \left( \left\lfloor \frac{x_i}{q} \right\rfloor - \left\lfloor \frac{ax_i}{m} \right\rfloor \right) m
  \]

  and that the second term above is always equal to \( m \) or 0

• As the result is between 0 and \( m-1 \), we are sure that the last term is equal to \( m \) only if :

  \[
  a(x_i \mod q) - \left\lfloor \frac{x_i}{q} \right\rfloor r < 0.
  \]
Hence, this decomposition of $m$ never leads to an overflow when working with a “long integer” within a 32 bits environment.

\[
\begin{align*}
\text{a} & \leftarrow 16807; \\
\text{m} & \leftarrow 2147483647; \\
\text{q} & \leftarrow 127773; \\
\text{r} & \leftarrow 2836; \\
\text{Seed initialisation} \\
\text{For} \ i=1:M \\
\text{xx} & \leftarrow \text{seed mod q}; \\
\text{k} & \leftarrow (\text{seed-xx})/q; \\
\text{x} & \leftarrow \text{a} \times \text{xx} - \text{r} \times \text{k}; \\
\text{If} \ \text{x} < 0 \\
\text{x} & \leftarrow \text{x} + \text{m}; \\
\text{seed} & \leftarrow \text{x} \\
\text{End}
\end{align*}
\]
Simulations
Basics about pseudo-Random Numbers Generation

- Other methods have been developed and are used (ex: multiple congruential...)

- References:
Simulations
Generation of standard normal numbers (1/4)

- We need to generate numbers $\varepsilon_i$ following standard normal distributions $N(0,1)$
- The first step is to generate **independent** realisations of standard normal distributions
- Several methods can be considered. One of these: **Polar rejection method**.
  - Generate two numbers following independent uniform laws on $[-1, 1]$: $u_1, u_2$
  - Calculate $w = (u_1)^2 + (u_2)^2$, and if $w < 1$, set:
    - $e_1 = (-2 \ln(w)/w)^{1/2} u_1$
    - $e_2 = (-2 \ln(w)/w)^{1/2} u_2$
  - If $w \geq 1$, reject the couple $(u_1, u_2)$ and generate another couple
**Simulations**

Generation of standard normal numbers (2/4)

- **Alternative method: Box-Muller:**
  - \( R = -2 \ln(u_1) \)
  - \( W = 2\pi u_2 \)
  - \( e_1 = \sqrt{R} \cos(W) \)
  - \( e_2 = \sqrt{R} \sin(W) \)

- This method is linked to the choice of polar coordinates and to the fact that if \( Z=(Z_1,Z_2) \sim N(0,Id) \) is a bivariate Gaussian vector, then:
  - \( R = Z_1^2 + Z_2^2 \sim \text{Exponential} \text{ distribution with mean 2} \)
  - Conditionally to \( R \), the vector \( (Z_1,Z_2) \) follows a bivariate uniform distribution (with independent margins) on the circle of radius \( \sqrt{R} \) centered at the origin
Simulations
Generation of standard normal numbers (3/4)

- Other method: direct inversion of the cumulative distribution function $\Phi$ of the standard normal $N(0,1)$
- No analytical expression of $\Phi$ in terms of elementary functions (exp, log, sin, cos, polynomials,…).
- Approximation of the inverse of $\Phi$:

\[
\Phi^{-1}(u) \approx \frac{\sum_{n=0}^{3} a_n (u - 0.5)^{2n+1}}{1 + \sum_{n=0}^{3} b_n (u - 0.5)^{2n}} \quad 0.5 \leq u \leq 0.92
\]

\[
\Phi^{-1}(u) \approx \sum_{n=0}^{8} c_n \left(\ln(-\ln(1-u))\right)^n \quad 0.92 \leq u \leq 1
\]

\[
\Phi^{-1}(u) = -\Phi^{-1}(1-u) \quad 0 \leq u \leq 0.5
\]
Simulations
Generation of standard normal numbers (4/4)

where:

\[
a_0 = 2.50662823884 \quad b_0 = -8.47351093090 \\
a_1 = -18.61500062529 \quad b_1 = 23.08336743743 \\
a_2 = 41.39119773534 \quad b_2 = -21.06224101826 \\
a_3 = -25.44106049637 \quad b_3 = 3.13082909833 \\
c_0 = 0.3374754822726147 \quad c_5 = 0.0003951896511919 \\
c_1 = 0.9761690190917186 \quad c_6 = 0.000321767881768 \\
c_2 = 0.1607979714918209 \quad c_7 = 0.0000002888167364 \\
c_3 = 0.0276438810333863 \quad c_8 = 0.0000003960315187 \\
c_4 = 0.0038405729373609
\]
This becomes important in option pricing for basket options, or options depending on several underlying variables.

The different variables are generally dependent.

A simple possibility is to incorporate linear correlations between the different Brownian motions of the different processes.

In practice: Cholesky decomposition
- At the level of the pseudo-random numbers generated following $N(0,1)$
- Correlation levels need of course to be estimated, like any other parameter

First step: Generation of the independent numbers $\varepsilon_i$, one sequence of numbers by Brownian motion, and introduction of correlations between the different sequences.
Simulations
Introduction of correlations between Brownian motions (1/4)

- Cholesky decomposition: decomposition of the correlation matrix in a product of an upper and a lower triangular matrices.
- As the correlation matrix $\rho$ is symmetric, one can show that there exists a lower triangular matrix $L$ such that $LL^T = \rho$.
- We then use the following result:
  - If $Z = (Z_1, \ldots, Z_d)$ is a vector of independent Gaussian variables, if $LL^T = \rho$, then $W = LZ^T$ is a Gaussian vector with correlation matrix $\rho$.
- Thanks to this result, it suffices to find $L$, and to multiply the Gaussian vectors (obtained by using one of the previous method for generating i.i.d. Gaussian numbers) corresponding to the same simulation instant by the matrix $L$. 

Simulations
Introduction of correlations between Brownian motions (2/4)

- In practice, the first step is to find the matrix $L$ of the Cholesky decomposition
  - The algorithm corresponds simply to write the equation $LL^T = \rho$ on an element by element basis:

$$L(i, j)L(i, i) + \sum_{k=1}^{n} L(k, i)L(k, j) = \rho(i, j)$$

and isolate in a member of the equation $L(k=i,j)$ (first treat the case $i=j$, then the case $i>j$)
Simulations
Introduction of correlations between Brownian motions (3/4)

- This leads to the following algorithm:

```plaintext
for i=1:N
  for j=1:N
    A(i,j)=0;
  end
end
for j=1:N
  for i=j:N
    V(i)=correl(i,j);
    for k=1:(j-1)
      V(i)=V(i)-A(j,k)*A(i,k);
    end
    A(i,j)=V(i)/sqrt(V(j));
  end
end
resultat=A;
```
The correlations are then introduced between the numbers $\varepsilon_i$ (we suppose here that we have $M$ Brownian motions) by a loop:

$$\text{For } j = 1 \ldots M : \quad \varepsilon_{\text{corr}}(i, j) = \sum_{k=1}^{j} L_{jk} \varepsilon(M(i-1) + k)$$

where $i$ represents a given instant in a given generated scenario.

The index $i$ is used later within the generation of the process associated to one asset or variable by Euler (or variant) discretisation scheme, for the increase of Brownian motion for the $j^{th}$ process.

**In practice:**

One considers groups of $M$ numbers generated following standard Gaussian variables, and we introduce correlations within each group.

At a given instant in a given scenario, the $M$ numbers $\varepsilon_{\text{corr}}(i,j)$ correspond to realisations of variables that are correlated with the correlation matrix $\rho$. 
• **Vasicek model** - **Euler discretisation**:

\[
\text{dr}(t) = (\theta(t) - ar(t))dt + \sigma dW(t)
\]

\[
\Rightarrow r_{t_{i+1}} = r_{t_i} + (\theta(t_i) - ar(t_i))(t_{i+1} - t_i) + \sigma \sqrt{t_{i+1} - t_i} \varepsilon_i
\]

where \( \varepsilon_i \) are normal variables \( N(0,1) \) i.i.d.

• Now, in the case of Vasicek model, some alternative discretization schemes exist, deduced from the explicit expression of the short rate

\[
\text{dr}(t) = k(\mu - r(t))dt + \sigma dW(t)
\]

\[
\Rightarrow r_t = r_0 \exp(-kt) + \mu(1 - \exp(-kt)) + \int_0^t \sigma \exp(-k(t - u)) dW_u
\]

\[
\Rightarrow r(t + \Delta t) = r(t) \exp(-k\Delta t) + \mu(1 - \exp(-k\Delta t)) + \varepsilon_s \sigma \sqrt{\frac{1 - \exp(-2k\Delta t)}{2k}}
\]

• The same methodology can be applied for Hull-White model from the explicit solution for the short rate
This can also be applied for the formulation of Hull-White of the form “Vasicek + deterministic function” previously introduced:

\[
\begin{align*}
  r(t) &= x(t) + \alpha(t) \\
  dx(t) &= -ax(t)dt + \sigma dW(t) \\
  \alpha(t) &= f^M(0, t) + \frac{\sigma^2}{2a^2} (1 - e^{-at})^2
\end{align*}
\]

In the case of G2++, the Vasicek schemes for \(x(t)\) and \(y(t)\) lead to the following discretization:

\[
\begin{align*}
  r(t_i) &= x(t_i) + y(t_i) + \varphi(t_i) \\
  x(t_{i+1}) &= x(t_i) \exp(-a \Delta t) + \varepsilon_i \sigma \sqrt{\frac{1 - \exp(-2a \Delta t)}{2a}} \\
  y(t_{i+1}) &= y(t_i) \exp(-b \Delta t) + \varepsilon_i' \eta \sqrt{\frac{1 - \exp(-2b \Delta t)}{2b}} \\
  \text{corr}(\varepsilon_i, \varepsilon_i') &= \rho
\end{align*}
\]
Euler scheme:

\[ x(t_{i+1}) = x(t_i) + k(\theta - x(t_i))(t_{i+1} - t_i) + \sigma \sqrt{x(t_i)} \sqrt{t_{i+1} - t_i} \varepsilon_i \]

In theory, the probability to get negative rates is zero if \( \Phi(t) > 0 \). The probability that \( x(t) \) reaches 0 is generally 0 as well, but discretisation errors can imply in practice that we reach the origin and even cross it.

In order to avoid such a situation, one can replace in the square root the term \( x(t_i) \) by its positive value, and by forcing negative values of \( x(t) \) to 0, i.e. using the following modified Euler scheme:

\[ x(t_{i+1}) = x(t_i) + k(\theta - x(t_i))(t_{i+1} - t_i) + \sigma \sqrt{(x(t_i))^+} \sqrt{t_{i+1} - t_i} \varepsilon_i \]

if \( x(t_{i+1}) < 0 \), then \( x(t_{i+1}) = 0 \)

An alternative discretisation scheme is due to Scott (cf. references in [Glassermann, pages 120-124]). The method is summarized below.
The distribution of \( r(t) \) given \( r(s) \) for \( s < t \) is known explicitly, and appears as a non central chi-square distribution. This property is used to simulate the process.

The transition law of the CIR factor \( x(t) \) is indeed given by:

\[
x(t) = \frac{\sigma^2 (1 - e^{-k(t-s)})}{4k} \chi^2_d \left( \frac{4ke^{-k(t-u)}}{\sigma^2 (1 - e^{-k(t-s)})} x(s) \right), \quad t > s
\]

where \( d = \frac{4\theta k}{\sigma^2} \)

and \( \chi^2_d (\lambda) \) is a noncentral chi squared distribution with non centrality parameter \( \lambda \).
Now, in order to simulate a noncentral chi-square variable with noncentrality parameter $\lambda$ and with an integer $\nu>1$ degrees of freedom, we first can decompose this variable in an ordinary (i.e. centered) chi-square variable and an independent normal. This can also be generalised in the case $\nu$ is not an integer.

Now, if $\nu>0$, by using the definition of a chi-square distribution with a non integer number of degrees of freedom, we can represent any noncentral chi-square variable with noncentrality parameter $\lambda$ as an ordinary (i.e. central) chi-square variable with a random degrees of freedom parameter, where this stochastic parameter follows actually a Poisson distribution with mean $\lambda/2$.

The method consists hence in this case to simulate a central variable $\chi^2_{\nu+2N}$ where $N$ is a Poisson distribution.

In clear, if $\nu>1$, the variable $\chi^2_{\nu}$ is simulated as $(Z + \sqrt{\lambda})^2 + \chi^2_{\nu-1}$, where $Z$ is a normal distribution, and if $\nu$ is smaller or equal to 1, as $\chi^2_n$ where $N$ is a Poisson variable with mean $\lambda/2$. 

Derivative Pricing by Monte Carlo
Interest rates models – CIR++
Illustration of both discretization methods:

- Averages on simulations of the short rate, 5 and 10 years zero-coupon rates and theoretical mean (dame parameters). Right: alternative method, left: standard Euler scheme. Only 1000 simulations (insufficient for a good convergence towards the theoretical mean).
- The main interesting property of this simulation algorithm is that we avoid now negative rates when they should in theory not appear.