Example of High Dimensional Contract

- An exotic high dimensional option is the ING-Coconote option (Conditional Coupon Note), whose lifetime is 8 years (2004-2012). The interest rate paid is flexible.
- In the first 2 years, a payment of 6 %, after that interest rate depends on asset prices.
- It depends on the value of the average value of the stocks in the Dow Jones Global Titan index:
- The starting price (2/2004) was Euro 100.75, i.e., the value of 50 stocks of the Dow Jones Global Titans index.
- 15 / 2 / 2012: Eur 100 will be paid guaranteed + the variable coupon

Dow Jones Global Titans index, 50 stocks

Abbott Laboratories New York Medical Supplies
Altria Group Inc. New York Tobacco
American International Group Inc. New York Full Line Insurance
Astrazeneca PLC London Pharmaceuticals
Bank of America Corp. New York Banks
Barclays PLC London Banks
BellSouth Corp. New York Fixed Line Telecommunications
BP PLC London Integrated Oil & Gas
Chevron Corp. New York Integrated Oil & Gas
Cisco Systems Inc. NASDAQ Telecommunications Equipment
Citigroup Inc. New York Banks
Coca-Cola Co. New York Soft Drinks
DaimlerChrysler AG NA XETRA Automobiles
Dell Inc. NASDAQ Computer Hardware
Eli Lilly & Co. New York Pharmaceuticals

Dow Jones Global Titans index, 50 stocks

ENI S.p.A. Milan Integrated Oil & Gas
Exxon Mobil Corp. New York Integrated Oil & Gas
General Electric Co. New York Diversified Industrials
GlaxoSmithKline PLC London Pharmaceuticals
HBOS PLC London Banks
HSBC Holdings PLC (UK Reg) London Banks
ING Groep N.V. Amsterdam Life Insurance
Intel Corp. NASDAQ Semiconductors
International Business Machines New York Computer Services
Johnson & Johnson New York Pharmaceuticals
JPMorgan Chase & Co. New York Banks
Merck & Co. Inc. New York Pharmaceuticals
Microsoft Corp. NASDAQ Software
Morgan Stanley New York Investment Services

Dow Jones Global Titans index, 50 stocks

Nestle S.A. VIRTX Food Products
Nokia Corp. Helsinki Telecommunications Equipment
Novartis AG VIRTX Pharmaceuticals
PepsiCo Inc. New York Soft Drinks
Pfizer Inc. New York Pharmaceuticals
Procter & Gamble Co. New York Nondurable Household Products
Roche Holding AG Part. Cert. VIRTX Pharmaceuticals
Royal Bank of Scotland Group PLC London Banks
Royal Dutch Petroleum Co. Amsterdam Integrated Oil & Gas
Samsung Electronics Co. Ltd. Korea Semiconductors
SBC Communications Inc. New York Fixed Line Telecommunications
Siemens AG XETRA Electronic Equipment
Time Warner Inc. New York Broadcasting & Entertainment

Dow Jones Global Titans index, 50 stocks

Total S.A. Paris Integrated Oil & Gas
Toyota Motor Corp. Tokyo Automobiles
UBS AG VIRTX Banks
Verizon Communications Inc. New York Fixed Line Telecommunications
Vodafone Group PLC London Mobile Telecommunications
Wal-Mart Stores Inc. New York Broadline Retailers
Walt Disney Co. New York Broadcasting & Entertainment
Wyeth New York Pharmaceuticals

- If an asset increases more than 8 %, a maximum of 8 % is considered. If it decreases more than 20 % a minimum -20 % is considered.
- Then, there is a maximum payment of 8 %, and a minimum of 0 % each year. However, the interest rate cannot decrease from year to year.
- The required information to value a basket option is the volatility of each asset σ_i and the correlation between each pair of assets $\rho_{i,j}$.
- With Payoff:

$$\frac{1}{50}\sum_{i=1}^{50}\max\{\min\{\frac{S_{j+1}^{(i)}-S_j^{(i)}}{S_j^{(i)}},8\%\},-20\%\},\ j=t_j$$

Simulation: Euler Discretization

Suppose that we want to generate random paths from the process of the following form:

$$dS_t = \mu S_t dt + \sigma S_t dW_t$$

first: we choose a time step $\Delta t = T/N$, and generate iteration of

$$S_{t+\Delta t} - S_t = \mu S_t \Delta t + \sigma S_t \left(W_{t+\Delta t} - W_t\right)$$

Algorithm

```
clear all; clc; close all;
S0=50; N=1000;T=1; delta=T/N;
S=S0; mu=0.1;sigma=0.2;
X=[];

for i=1:N
    Z=random('normal',0,sqrt(delta));
    S(i+1)=S(i)+mu*S(i)*delta+sigma*S(i)*Z;
end
plot([0:delta:1],S)
```

Simulation: Euler Discretization

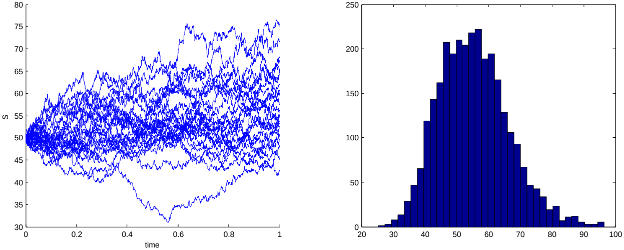


Figure: Euler discretization for: $S_0 = 50$, $\mu = 0.1$, $\sigma = 0.2$, $T = 1$ and $N = 1000$, LEFT: generated paths, RIGHT: histogram of the prices at maturity T

Exercises

Stochastic Integration: For a given deterministic function $g(s)$ how to find:

$$\int_0^t g(s)\mathrm{d}W_s = ?$$

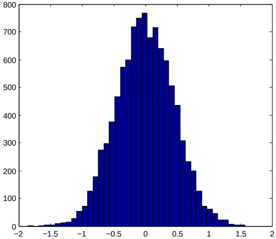
First we define partition: $0 = t_0 < t_1 < \cdots < t_n = T$ then

$$\int_0^T g(s)\mathrm{d}W_s = \sum_{k=0}^{n-1} g(t_k)\left(W(t_{k+1}) - W(t_k)\right)$$

Example: let us take $g(t) = t^2$ and $T = 1$. Theoretically we have:

$$\begin{aligned} \mathbb{E}\left(\int_0^1 t^2\mathrm{d}W_t\right) &= 0 \\ Var\left(\int_0^1 t^2\mathrm{d}W_t\right) &= \mathbb{E}\left(\int_0^1 t^2\mathrm{d}W_t\right)^2 = \int_0^1 t^4\mathrm{d}t = 0.2 \end{aligned}$$

Exercises




```
clear all; clc; close all;
T=1;
N=100;
delta=T/N;
X=[];
t=[0:delta:T-delta];
f=t.^2;
for i=1:10000;
dW_t=random('normal',0,sqrt(delta),[N,1]);
X=[X;sum(f.*dW_t')];
end
hist(X,40)
```

Figure: LEFT: Histogram of simulated stochastic integral with $T = 1$, $N = 100$, $g(t) = t^2$. RIGHT: Matlab code

$$\mathbb{E}\left(\int_0^1 t^2 \mathrm{d}W_t\right) \approx -0.0037, \;\; Var\left(\int_0^1 t^2 \mathrm{d}W_t\right) \approx 0.2047$$

Exercises

Stochastic Integration: For a given Brownian motion W_t how to find:

$$\int_0^T W_s \mathrm{d}W_s = ?$$

Analytically:

$$\mathbb{E}\left(\int_0^T W_s \mathrm{d}W_s\right) = 0$$

In order to calculate the variance we define a function $g = x^2$, from Itô we have:

$$\mathrm{d}g = g_x \mathrm{d}x + \frac{1}{2} g_{x,x} (\mathrm{d}x)^2 = 2x\mathrm{d}x + \frac{1}{2} \cdot 2(\mathrm{d}x)^2$$

so:

$$\mathrm{d}W_t^2 = 2W_t \mathrm{d}W_t + (\mathrm{d}W_t)^2 = 2W_t \mathrm{d}W_t + \mathrm{d}t$$

Exercises

Further we have:

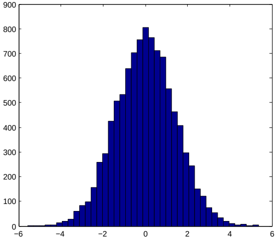
$$\begin{aligned} \int_0^T \mathrm{d}W_t^2 &= 2\int_0^T W_t \mathrm{d}W_t + \int_0^T \mathrm{d}t \\ W_T^2 - W_0^2 &= 2\int_0^T W_t \mathrm{d}W_t + T \end{aligned}$$

so finally:

$$\int_0^T W_t \mathrm{d}W_t = \frac{1}{2}W_T^2 - \frac{1}{2}T.$$

Example: Let us take $T = 2$. Theoretically we have:

$$\mathbb{E}\left(\int_0^2 W_s \mathrm{d}W_s\right) = 0, \ \ Var\left(\int_0^2 W_s \mathrm{d}W_s\right) = 2$$




```
clear all; clc; close all;
T=2;
N=100;
delta=T/N;
X=[];
t=[delta:delta:T];
for i=1:10000;
 W_t=random('normal',0,sqrt([0:delta:T-delta]));
 dW_t=random('normal',0,sqrt(delta),[N,1]);
 X=[X;sum(W_t*dW_t)];
end
mean(X)
var(X)
```

Figure: LEFT: Histogram of simulated stochastic integral with $T = 2$, $N = 100$. RIGHT: Matlab code

$$\mathbb{E}\left(\int_0^2 W_t \mathrm{d}W_t\right) \approx 0.0083, \;\; Var\left(\int_0^2 W_t \mathrm{d}W_t\right) \approx 1.9883$$

Analytic vs Monte-Carlo of BS model

Exercise: We set: $S_0 = 5$, $\sigma = 0.3$, $r = 0.06$, $T = 1$, $M = 500$ ($= \#$ time steps), and $K = S_0$. Exact solutions from BS formula are:

$$Call_{t=0} = 0.7359, \; Put_{t=0} = 0.4447, \text{ with time } 0.0003[s]$$

Table: Call and Put prices depending on number of Monte-Carlo Paths (Euler Approach)

n ($N = 2^n$) # paths	2	4	6	10	14
Call price at $t = 0$	1.0770	0.9217	0.8622	0.7729	0.7227
Put price at $t = 0$	0.1737	0.1881	0.4132	0.4258	0.4544
Time [s]	0.07	0.10	0.20	2.15	34.56

Analytic vs Numerical Solution of BS model

Algorithm

```
S_0=5; K=5; sigma=0.3; r=0.06; T=1; %defining variables
N=500; delta=T/N;                   %time steps
NoOfPaths=2^14;                     %number of generated paths
S_T=[];
tic
noise=random('normal',0,sqrt(delta),[N,NoOfPaths]);
for path=1:1:NoOfPaths
    S=zeros(N,1);
    S(1)=S_0;
    time=0;
    TT=0;
    for i=1:1:N;
        time=time+delta;
        TT=[TT;time];
        S(i+1)=S(i)+r*S(i)*delta+sigma*S(i)*noise(i,path);
    end;
    S_T=[S_T;S(end)];
end
    Call=exp(-r*T)*mean(max(S_T-K,0))    %MC Call price
    Put=exp(-r*T)*mean(max(K-S_T,0))     %MC Put  price
    toc
    [CallExact,PutExact] = blsprice(S_0, K, r, T, sigma)  %% exact solution
```

Numerical Solutions

Monte Carlo simulation vs. BS formula for Call & Put Prices as a function of strikes.

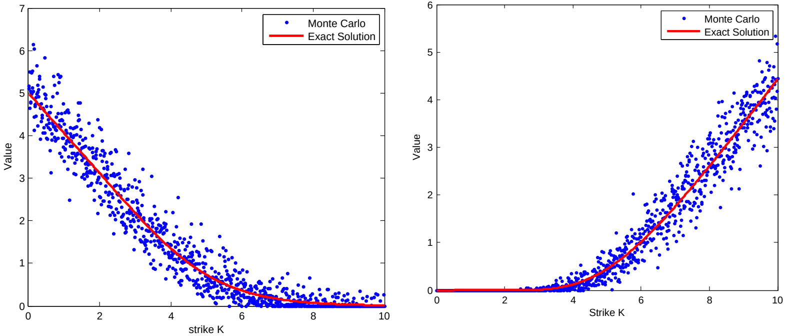


Figure: Call and Put prices as a function of Strikes, K, $K = [0.01 : 0.01 : 10]$, Steps=500

Simulation

Again X_t denotes a stochastic process and solution of an SDE,

$$dX_t = \mu(Y_t, t)dt + \sigma(X_t, t)dW_t \text{ for } 0 \leq t \leq T.$$

where the driving process W_t is a Wiener process. We already know the Euler discretization:

$$\begin{cases} x_{i+1} &= x_i + \mu(x_i, t_i)\Delta t + \sigma(x_i, t_i)\Delta W_j \ , \ t_j = j\Delta t \\ \Delta W_j &= W_{t_{i+1}} - W_{t_i} = Z\sqrt{\Delta t} \text{ with } Z \sim N(0,1) \end{cases} \tag{1}$$

where length Δt is assumed equidistant, i.e., $\Delta t = \frac{T}{M}$.

Definition (Absolute error)

The absolute error at time T is defined as:

$$\epsilon(h) := \mathbb{E}\left(|X_T - x_T^h|\right).$$

Approximation Error

Example Suppose we study a linear SDE of the form:

$$dX_t = \mu X_t dt + \sigma X_t dW_t,$$

We know that the solution is of the following form:

$$X_T = X_0 \exp\left((\mu - \frac{1}{2}\sigma^2)T + \sigma W_T \right)$$

Now, we perform an experiment in order to check the Euler method's convergence. In the experiment we use a discrete measure of the absolute error:

$$\widetilde{\epsilon}(h) = \frac{1}{N}\sum_{k=1}^{N} |X_{T,k} - X^h_{T,k}|$$

Approximation Error- Euler

Example (Euler Scheme) We set $X_0 = 50, \mu = 0.06, \sigma = 0.3$, $T = 1$ and find

$$\widetilde{\epsilon}(h) = \frac{1}{N} \sum_{k=1}^{N} |X_{T,k} - X_{T,k}^{h}|$$

Table: Table of the absolute error $\epsilon(h)$, wrt time and h.

error:ϵ	M=100	M=1000	M=2000	M=3000	M=5000
seed 1	0.259	0.082	0.062	0.052	0.037
seed 2	0.270	0.087	0.053	0.050	0.035
seed 3	0.300	0.081	0.065	0.045	0.038
time [s]	0.15	0.94	2.14	3.90	8.7

Approximation Error

Example The numerical results for obtained estimates $\widetilde{\epsilon}$ are assumed to be valid for ϵ. We postulate:

$$\epsilon(h) \leq C \cdot h^{\frac{1}{2}} = \mathcal{O}(h^{\frac{1}{2}})$$

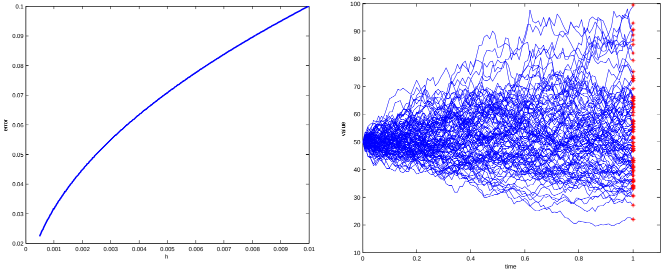


Figure: LEFT: error against value of step size h for Euler discretization. RIGHT: Generated paths for $M = 100$, red stars indicate value of exact solution.

Algorithm

```
%%euler discretization- speed of convergence
clear all;clc;close all;
X0=50; mu=0.06; sigma=0.3;T=1; N=100; M=5000;
delta=T/M;
x=[];
ti=0;
tic
for j=1:N        %number of paths
Z=random('normal',0,1,[M,1]);
X=X0;
path=0;
N=[];
x=X0;
     for i=1:M; %number of steps
         ti=ti+delta;
         X(i+1)=X(i)+mu*X(i)*delta+sigma*X(i)*sqrt(delta)*Z(i); %Euler
         path=path+sqrt(delta)*Z(i);
     end
    y=X0*exp((mu-0.5*sigma^2)*T+sigma*path);                     %Exact
    error(j)=y-X(end);
end
epsilon=mean(abs(error))
toc
```

Strong and weak convergence

Definition (Strong Convergence)

x_T^h converges strongly to X_T with order $\gamma > 0$ if

$$\epsilon(h) = \mathbb{E}\left(|X_T - x_T^h|\right) = \mathcal{O}(h^\gamma),$$

x_T converges strongly, if $\lim_{h\to 0} \mathbb{E}\left(|X_T - x_T^h|\right) = 0$.
The Euler method converges strongly with order $1/2$.

Definition (Weak Convergence)

x_T^h converges weakly to X_T with respect to g with order $\beta > 0$, if

$$|\mathbb{E}(g(X_t)) - \mathbb{E}(g(x_T^h))| = \mathcal{O}(h^\beta)$$

The Euler scheme is weakly $\mathbb{O}(h^1)$ convergent wrt polynomials g.

Milstein Algorithm

As before X_t denotes a stochastic process and solution of an SDE,

$$dX_t = \mu(Y_t, t)dt + \sigma(X_t, t)dW_t \text{ for } 0 \leq t \leq T.$$

where the driving process W_t is a Wiener process. Now we introduce another discretization method:
The Milstein Scheme.

$$\begin{cases} x_{i+1} & = & x_i + \mu\Delta t + \sigma\Delta W_j + \frac{1}{2}\sigma\sigma' \cdot \left((\Delta W)^2 - \Delta t\right) \\ \Delta W_j & = & W_{t_{i+1}} - W_{t_i} = Z\sqrt{\Delta t} \text{ with } Z \sim N(0,1) \end{cases} \tag{2}$$

where length Δt is assumed equidistant, i.e., $\Delta t = \frac{T}{M}$, and where

$$\sigma' = \frac{\partial\sigma(x_i, t_i)}{\partial x_i}$$

Milstein Algorithm

In the case of GBM we obtain:

$$\begin{cases} x_{i+1} & = & x_i + \mu x_i \Delta t + x_i \sigma \Delta W_j + \frac{1}{2}\sigma^2 x_i \cdot \left((\Delta W)^2 - \Delta t\right) \\ \Delta W_i & = & W_{t_{i+1}} - W_{t_i} = Z\sqrt{\Delta t} \text{ with } Z \sim N(0,1) \end{cases} \tag{3}$$

The additional correction term in the Milstein scheme improves the speed of convergence compared to Euler method. The 'improved' method is convergent with order one.

Although the Milstein Scheme is definitely manageable in the one-dimensional case, its general multidimensional extension may be very difficult.

Approximation Error- Milstein

Example (Milstein Scheme) We set $X_0 = 50, \mu = 0.06, \sigma = 0.3$, $T = 1$ and find

$$\widetilde{\epsilon}(h) = \frac{1}{N} \sum_{i=1}^{N} |X_T - X_{T,k}^h|$$

Table: Table of the absolute error $\epsilon(h)$, wrt time and h.

error:ϵ	M=100	M=1000	M=2000	M=3000	M=5000
seed 1	0.005	7E-4	4E-4	2E-4	2E-4
seed 2	0.007	6E-4	4E-4	3E-4	2E-4
seed 3	0.005	8E-4	4E-4	3E-4	2E-4
time [s]	0.108	0.392	0.999	2.35	8.44

Approximation Error- Milstein

```
%%Milstein discretization- speed of convergence
clear all;clc;close all;
X0=50; mu=0.06; sigma=0.3;T=1; N=100; M=100;
delta=T/M;
x=[];
ti=0;
tic
for j=1:N                          %number of paths
Z=random('normal',0,1,[M,1]);
X=X0;
path=0;
x=X0;
     for i=1:M;                    %number of steps
         ti=ti+delta;
         X(i+1)=X(i)+mu*X(i)*delta+sigma*X(i)*sqrt(delta)*Z(i)+...
             0.5*sigma^2*X(i)*((sqrt(delta)*Z(i))^2-delta);
         path=path+sqrt(delta)*Z(i);
     end
     x=X0*exp((mu-0.5*sigma^2)*T+sigma*path);                    %Exact
     error(j)=x-X(end);
end
epsilon=mean(abs(error))
toc
```

Euler vs Milstein- Trajectories

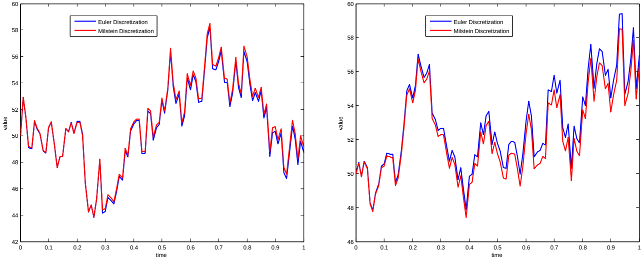


Figure: LEFT: Trajectories generated for $\sigma = 0.1$, RIGHT: Trajectories generated for $\sigma = 0.3$

Antithetic sampling

It is well known that if a random variable $Z \sim N(0,1)$, then $-Z \sim N(0,1)$. We can use this property to drastically reduce the number of paths needed in the Monte-Carlo simulation. Suppose that $\hat{V}$ is the approximation obtained from MC, and $\widetilde{V}$ is the one obtained using $-Z$. By taking average

$$V = \frac{1}{2}\left(\widetilde{V} + \hat{V}\right)$$

we obtain a new approximation. Since $\hat{V}$ and V are both random variables we aim at:

$$Var(V) < Var(\hat{V}).$$

We have:

$$Var(V) = \frac{1}{4}Var(\widetilde{V} + \hat{V}) = \frac{1}{4}Var(\widetilde{V}) + \frac{1}{4}Var(\hat{V}) + \frac{1}{2}Cov(\widetilde{V}, \hat{V}).$$

So it is clear that: $Var(V) \leq \frac{1}{2}\left(Var(\hat{V}) + Var(\widetilde{V})\right)$.

Antithetic sampling

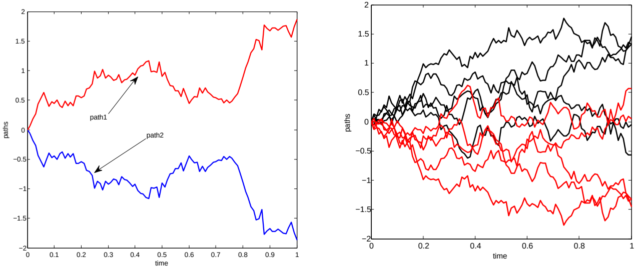


Figure: Usual and Antithetic paths. LEFT: one path, RIGHT: five paths

Antithetic sampling

Now, we check different implementations vs. number of paths, accuracy and time. For standard GBM model we set: $T = 1$, $\mu = 0.04$, $\sigma = 0.3$, $\Delta = 0.01$, $S_0 = 50$.

Standard	$N = 10^1$	$N = 10^3$	$N = 10^4$	$N = 10^5$
$\lvert\epsilon\rvert$	7.0915	0.5027	0.2301	0.0457
Time	0.066	0.2153	2.1408	21.3213

Antithetic	$N = 10^1$	$N = 10^3$	$N = 10^4$	$N = 10^5$
$\lvert\epsilon\rvert$	0.1639	0.0460	0.0037	0.000138
Time	0.066	0.2155	2.1418	21.3321

The antithetic approach converges much faster, without any extra time needed for calculation.

Antithetic sampling

```
% antithetic variates
clear all;clc;close all;
T=1; mu=0.04; sigma=0.3;
N=10^3; % no of paths
M=100; % number of steps
delta=T/M; S_0=50;
tic
Noise=random('normal',0,sqrt(delta),[N,M]);
path(:,1)=zeros(N,1);
path2(:,1)=zeros(N,1);
for i=1:M
    path(:,i+1)=path(:,i)+Noise(:,i);
    path2(:,i+1)=path2(:,i)-Noise(:,i);
end
S=S_0.*exp((mu-0.5*sigma^2)*T+sigma*path(:,end));
toc
S_2=0.5*(S+S_0.*exp((mu-0.5*sigma^2)*T+sigma*path2(:,end)));
toc
[p1,p2] = lognstat((mu-sigma^2/2)*T,sqrt(sigma^2*T));
theore=p1*S_0;
e1=abs(mean(S)-theore)
e2=abs(mean(S_2)-theore)
```

Figure: Matlab implementation- semi-efficient way of programming.

Antithetic sampling

Efficient Implementation

No.Samp.	$N = 10^1$	$N = 10^3$	$N = 10^4$	$N = 10^5$
$\lvert\epsilon\rvert$	0.1639	0.0460	0.0037	0.000138
Time	0.0651	0.0721	0.1411	0.8992

```
% antithetic variates
clear all;clc;close all;
T=1;mu=0.04;sigma=0.3;
N=10^5; % no of paths
M=100; % number of steps
delta=T/M; S_0=50;

tic
Noise=random('normal',0,sqrt(delta),[N,M]);
P=cumsum(Noise,2);
S=0.5*S_0.*(exp((mu-0.5*sigma^2)*T+sigma*P(:,end))+exp((mu-0.5*sigma^2)*T-sigma*P(:,end)));
toc
[p1,p2] = lognstat((mu-sigma^2/2)*T,sqrt(sigma^2*T));
theore=p1*S_0;
error=abs(mean(S)-theore)
```

Figure: Matlab implementation-efficient way of programming.

Simulating Jumps

$$dS_t = dN_t$$

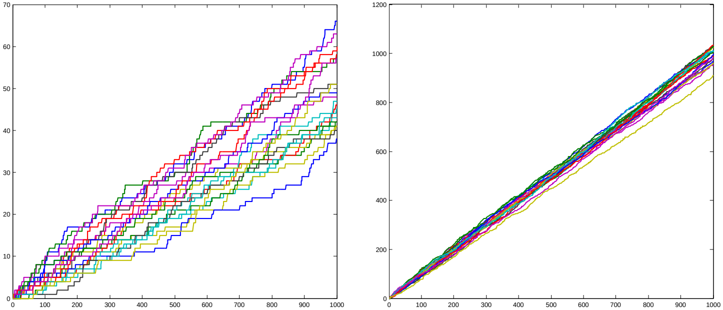


Figure: Poisson Process realizations with $\Delta = 1$, LEFT: $\lambda = 0.01$, RIGHT: $\lambda = 1.0$

Simulating Jumps

Suppose we have given a stock variable S_t which jump at time τ_j. We denote τ^+ the moment after one particular jump and τ^- the moment before.

- The absolute size of the jump is:

$$\Delta S = S_{\tau^+} - S_{\tau^-},$$

  which we model as a *proportional jump*,

- $S_{\tau^+} = qS_{\tau^-}$ with $q > 0$, so $\Delta S = qS_{\tau^-} - S_{\tau^-} = (q-1)S_{\tau^-}$.
- The jump sizes equal $q - 1$ times the current asset price.
- Assuming that for given set of i.i.d. $q_{\tau_1}, q_{\tau_2}, \ldots$ r.v.the process

$$dS_t = (q_t - 1)S_t dJ_t,$$

  is called Compound Poisson Process.

Simulating Jumps- Jump Diffusion Process

If we combine geometric Brownian motion and jump process we obtain:

$$dS_t = \mu S_t dt + \sigma S_t dW_t + (q_t - 1) S_t dJ_t.$$

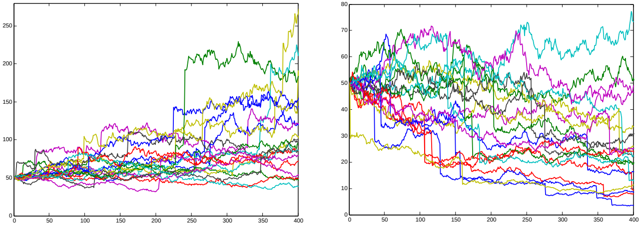


Figure: Geometric Brownian motion with jumps: $\Delta = 0.01$, $\mu = 0.04$, $\sigma = 0.2$, $\lambda = 0.5$, LEFT: $q = 1.4$,RIGHT $q = 0.6$.

Simulating Jumps

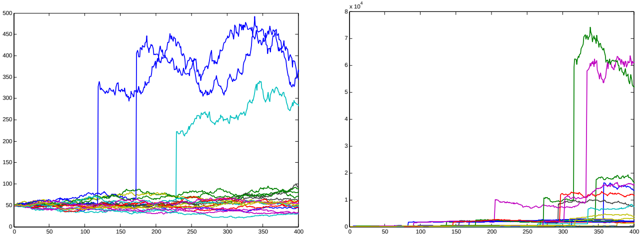


Figure: Geometric Brownian motion with jumps: $\Delta = 0.01$, $\mu = 0.04$, $\sigma = 0.2$, $q = 6$, LEFT: $\lambda = 0.04$,RIGHT $\lambda = 0.5$.

Market and Jumps- Algorithm

```
S_0=5; K=5; sigma=0.3; r=0.06; T=1; %defining variables
N=500; delta=T/N;                   %time steps
NoOfPaths=2^14;                     %number of generated paths
S_T=[];
tic
noise=random('normal',0,sqrt(delta),[N,NoOfPaths]);
for path=1:1:NoOfPaths
    S=zeros(N,1);
    S(1)=S_0;
    time=0;
    TT=0;
    for i=1:1:N;
        time=time+delta;
        TT=[TT;time];
        S(i+1)=S(i)+r*S(i)*delta+sigma*S(i)*noise(i,path);
    end;
    S_T=[S_T;S(end)];
end
    Call=exp(-r*T)*mean(max(S_T-K,0))    %MC Call price
    Put=exp(-r*T)*mean(max(K-S_T,0))     %MC Put  price
    toc
    [CallExact,PutExact] = blsprice(S_0, K, r, T, sigma)  %% exact solution
```

Simulating Jumps- Jump Diffusion Process

An analytical solution of the equation

$$dS_t = \mu S_t dt + \sigma S_t dW_t + (q_t - 1) S_t dJ_t,$$

can be calculated on each of the jump-free subintervals $\tau_j < t < \tau_{j+1}$ where the SDE is just a GBM.
When at time τ_1 a jump of size:

$$(\Delta S) = (q_{\tau_1} - 1) S_{\tau_1^-},$$

occurs, and thereafter the solution is given by:

$$S_t = S_0 \exp\left((\mu - \frac{\sigma^2}{2}) t + \sigma W_t \right) + (q_{\tau_1} - 1)\, S_{\tau_1^-}$$

In general we obtain:

$$S_t = S_0 \exp\left((\mu - \frac{\sigma^2}{2}) t + \sigma W_t \right) + \sum_{j=1}^{J_t} S_{\tau_j^-} \left(q_{\tau_j} - 1 \right).$$