Chapter 7

Simulating more interesting stochastic processes

7.1 Generating correlated random variables

The lectures contained a lot of motivation and pictures. We'll boil everything down to pure algebra in these notes.

Definition. *An n dimensional symmetric matrix S is said to be positive semi-definite if $\mathbf{x}^T S\mathbf{x} \geq 0$ for all vectors $\mathbf{x}$.*

Definition. *An n dimensional symmetric matrix S is said to be positive definite if $\mathbf{x}^T S\mathbf{x} > 0$ for all non zero vectors $\mathbf{x}$. Note the strict inequality.*

Lemma 1. *A covariance matrix is always positive semi-definite.*

Proof. Let Σ be the covariance matrix of random variables X_1, X_2, $\ldots X_n$. Let x be a vector in $\mathbb{R}^n$ with components x_i. Then the variance of the random variable $\sum_{i=1}^n x_i X_i$ is given by $x^T \Sigma x$. Hence $x^T \Sigma x$ is positive semi-definite. □

7.1.1 Multivariate normal distributions

In this section we will motivate the definition of the multivariate normal distribution and the notion of a pseudo-square root.

Lemma 2. *Let $\tilde{X}_1$, $\tilde{X}_2$, $\ldots$, $\tilde{X}_n$ all be independent normally distributed random variables each of mean 0 and standard deviation 1 then their joint density function is*

$$\frac{1}{(\sqrt{2\pi})^n} \exp\left(-\frac{1}{2}\sum_{i=1}^n \tilde{x}_i^2\right)$$

Proof. The p.d.f. of each $\tilde{X}_i$ is

$$\frac{1}{\sqrt{2\pi}} \exp\left(-\frac{1}{2}\tilde{x}_i^2\right).$$

They are independent so their joint density function is

$$\begin{aligned}\Pi_{i=1}^{n}\frac{1}{\sqrt{2\pi}}\exp\left(-\frac{1}{2}\tilde{x}_i^2\right) &= \frac{1}{(\sqrt{2\pi})^n}\Pi_{i=1}^{n}\exp\left(-\frac{1}{2}\tilde{x}_i^2\right)\\ &= \frac{1}{(\sqrt{2\pi})^n}\exp\left(-\frac{1}{2}\sum_{i=1}^{n}\tilde{x}_i^2\right)\end{aligned}$$

□

Suppose we introduce new random variables X_i which are linearly related to the original independent variables $\tilde{X}_i$ so that

$$X_i = \sum_{j=1}^{n} a_{ij}\tilde{X}_j$$

for some constants a_{ij}.

We can calculate the covariance of X_i and X_j it is:

$$\begin{aligned}\mathrm{Cov}(X_i, X_j) &= \mathrm{Cov}(\sum_{\alpha=1}^{n} a_{i\alpha}\tilde{X}_\alpha, \sum_{\beta=1}^{n} a_{j\beta}\tilde{X}_\beta)\\ &= \sum_{\alpha=1}^{n}\sum_{\beta=1}^{n} a_{i\alpha}a_{j\beta}\,\mathrm{Cov}(\tilde{X}_i, \tilde{X}_j)\\ &= \sum_{\alpha=1}^{n}\sum_{\alpha=1}^{n} a_{i\alpha}a_{j\alpha}.\end{aligned}$$

The first line follows by the bi-linearity of Cov. The second line follows from the fact that $\mathrm{Cov}(\tilde{X}_i, \tilde{X}_j)$ is equal to 1 if $i = j$ and 0 otherwise.

If we write Σ for the covariance matrix of the X_i and A for the matrix a_{ij} then we can write this equation in matrix form as:

$$\Sigma = AA^T.$$

The term

$$\sum_{i=1}^{n}\tilde{x}_i^2$$

that appears in the probability density function can be written in vector notation as $\tilde{\mathbf{x}}^{\mathbf{T}}\tilde{\mathbf{x}}$. Let us write $\mathbf{x} = \mathbf{A}\tilde{\mathbf{x}}$. Let us assume that A is invertible so we may write $A^{-1}\mathbf{x} = \tilde{\mathbf{x}}$. We can now compute

$$\begin{aligned}\sum_{i=1}^{n}\tilde{x}_i^2 &= (A^{-1}\mathbf{x})^{\mathbf{T}}(\mathbf{A^{-1}x})\\ &= \mathbf{x}^{\mathbf{T}}(\mathbf{A^{-1}})^{\mathbf{T}}(\mathbf{A^{-1}})\mathbf{x}\\ &= \mathbf{x}^{\mathbf{T}}(\mathbf{A}^{\mathbf{T}})^{-\mathbf{1}}(\mathbf{A^{-1}})\mathbf{x}\\ &= \mathbf{x}^{\mathbf{T}}(\mathbf{AA}^{\mathbf{T}})^{-\mathbf{1}}\mathbf{x}\\ &= \mathbf{x}^{\mathbf{T}}\mathbf{\Sigma}^{-\mathbf{1}}\mathbf{x}.\end{aligned}$$

This computation allows us to write down the joint density function for the X_i it is:

$$\frac{1}{(\sqrt{2\pi})^n}|\det A|^{-1}\exp\left(-\frac{1}{2}\mathbf{x^T\Sigma^{-1}x}\right)$$

We have $\Sigma = AA^T$ so $\det\Sigma = (\det A)(\det A^T) = (\det A)^2$ so we can eliminate A entirely from this formula. We obtain:

$$\frac{1}{(\sqrt{2\pi})^n}|\det\Sigma|^{-\frac{1}{2}}\exp\left(-\frac{1}{2}\mathbf{x^T\Sigma^{-1}x}\right).$$

Let us summarize our findings:

Proposition 1. *Let $\tilde{X}_1$, $\tilde{X}_2$, ..., $\tilde{X}_n$ be independently normally distributed with mean 0 and standard deviation 1. Let A be an invertible $n\times n$ matrix with components a_{ij}. Then defining random variables X_i by:*

$$X_i = \sum_{j=1}^{n} a_{ij}\tilde{X}_j$$

we have that the X_i have mean 0 and covariance matrix $\Sigma = AA^T$. The joint density function of the X_i is:

$$\frac{1}{(\sqrt{2\pi})^n}|\det\Sigma|^{-\frac{1}{2}}\exp\left(-\frac{1}{2}\mathbf{x^T\Sigma^{-1}x}\right). \tag{7.1}$$

Since A is invertible, so is A^T. So if $x\neq 0$, $A^Tx\neq 0$. Write $y = A^T$ then since $y\neq 0$ we have $y^Ty > 0$. Expanding this we have $x^TAA^Tx = x^T\Sigma x > 0$. Hence Σ is positive definite if A is invertible.

This motivates the following definition.

Definition. *The probability density function for a multivariate normal distribution of dimension n with mean μ and covariance matrix Σ (where Σ is positive definite) is defined to be:*

$$\frac{1}{(\sqrt{2\pi})^n}|\det\Sigma|^{-\frac{1}{2}}.\exp\left(-\frac{1}{2}(\mathbf{x}-\mu)^\mathbf{T}\mathbf{\Sigma^{-1}}(\mathbf{x}-\mu)\right)$$

where $\mathbf{x}$ is vector in $\mathbb{R}^n$.

Note that if we have this joint density function, then the change of variables $\mathbf{x}\to\mathbf{x}+\mu$ will take us to a density function of the form (7.1). So the mean and covariance of this distribution are μ and Σ as has been tacitly claimed in this definition.

One thing to observe is that a multivariate normal distribution (7.1.1) is defined using just Σ. We do not need to know A. Σ is a covariance matrix so is easy to measure statistically, so typically when modelling Σ will be given but A will not be given.

The following definition gives a formal name for the relationship between Σ and A.

Definition. *If S is a symmetric $n \times n$ matrix then A is said to be a pseudo square root of S if $S = AA^T$.*

It is important to be aware that pseudo square roots are far from unique.

7.2 Simulating multivariate normal distributions

Algorithm. *Suppose we wish to simulate random variables $X_1, X_2, \ldots X_n$ with joint density* (7.1.1)*. To do this we:*

(i) Find a pseudo square root A of Σ.

(ii) Generate independent normally distributed random variables $\tilde{X}_i$

(iii) Simulate X_i using the formula:

$$X_i = \sum_{j=1}^{n} a_{ij}\tilde{X}_j.$$

The challenge is to find a pseudo square root of Σ. One approach that is not recommended is to diagonalize Σ. This approach is discussed in more detail in the lecture notes. A more efficient algorithm is given by Cholesky decomposition.

Theorem 1. *If S is a symmetric, positive definite $n \times n$ matrix then there exists a unique lower triangular matrix L with positive diagonal such that L is a pseudo square root of S. This matrix is called the* Cholesky decomposition *of S.*

Proof. Suppose by induction that the result is true for $(n-1)\times(n-1)$ matrices. We write S in block diagonal form as

$$S = \left(\begin{array}{c|c} S_{n-1} & v_{n-1} \\ \hline v_{n-1}^T & s \end{array}\right) \tag{7.2}$$

where S_{n-1} is an $(n-1)\times(n-1)$ symmetric matrix, v_{n-1} is a vector of length $(n-1)$ and s is a scalar.

We now define L by

$$L = \left(\begin{array}{c|c} L_{n-1} & 0 \\ \hline w_{n-1}^T & w \end{array}\right)$$

where w_{n-1} is some n vector to be determined, and w is a scalar to be determined. We require that L_{n-1} is lower triangular with positive diagonal and that

$$LL^T = S.$$

This last condition is equivalent to the three conditions

$$L_{n-1}L_{n-1}^T = S_{n-1}, \tag{7.3}$$

$$L_{n-1}w_{n-1} = v_{n-1} \tag{7.4}$$

and

$$w_{n-1}^T w_{n-1} + w^2 = s. \tag{7.5}$$

It is easy to check that S_{n-1} is positive definite. So there is a unique choice for L_{n-1} by our induction hypothesis.

Since L_{n-1} is lower triangular with positive diagonal, it has a non-zero determinant and so is invertible. Hence there is a unique w_{n-1} solving (7.4). (Indeed this equation will already be in echelon form so it is quick and easy to solve). We now require that:

$$w^2 = s - w_{n-1}^T w_{n-1}.$$

There will be a unique positive solution to this equation if and only if

$$s - w_{n-1}^T w_{n-1} > 0.$$

To see that this inequality will hold, we define a vector v in block diagonal form by:

$$v = \begin{pmatrix} -(L_{n-1}^T)^{-1} w_{n-1} \\ \hline 1 \end{pmatrix}$$

we know that $v^T S v > 0$ as S is positive definite. We compute

$$\begin{aligned}
v^T S v &= \begin{pmatrix} -w_{n-1}(L_{n-1})^{-1} & | & 1 \end{pmatrix} \left(\begin{array}{c|c} S_{n-1} & v_{n-1} \\ \hline v_{n-1}^T & s \end{array}\right) \begin{pmatrix} -(L_{n-1}^T)^{-1} w_{n-1} \\ \hline 1 \end{pmatrix} \\
&= \begin{pmatrix} -w_{n-1}(L_{n-1})^{-1} & | & 1 \end{pmatrix} \left(\begin{array}{c|c} L_{n-1}L_{n-1}^T & L_{n-1}w_{n-1} \\ \hline w_{n-1}^T L_{n-1}^T & s \end{array}\right) \begin{pmatrix} -(L_{n-1}^T)^{-1} w_{n-1} \\ \hline 1 \end{pmatrix} \\
&= \begin{pmatrix} -w_{n-1}(L_{n-1})^{-1} & | & 1 \end{pmatrix} \begin{pmatrix} -L_{n-1}w_{n-1} + L_{n-1}w_{n-1} \\ \hline -w_{n-1}^T w_{n-1} + s \end{pmatrix} \\
&= \begin{pmatrix} -w_{n-1}(L_{n-1})^{-1} & | & 1 \end{pmatrix} \begin{pmatrix} 0 \\ \hline -w_{n-1}^T w_{n-1} + s \end{pmatrix} \\
&= s - w_{n-1}^T w_{n-1}.
\end{aligned}$$

This quantity is positive, so there exists a unique w solving (7.5). □

This formal algebraic proof is essentially equivalent to the argument that was given in the lectures. It is more rigorous but may be a little harder to read. The argument in the lectures contains a hand-waving "and so on" which we have tidied up in this rigorous proof using induction. In the lectures we also skipped the step where we checked that the right hand side of (7.5) is positive.

This proof gives a recursive algorithm for computing the Cholesky decomposition. First find the Cholesky decomposition of the $(n-1) \times (n-1)$ matrix L_{n-1} in the top left and then solve equations (7.5) and (7.4). The Cholesky decomposition is then given by (7.2).

7.3 Solving SDEs numeically

7.3.1 Multi-dimensional Brownian motion

Definition. *d-dimensional Brownian motion with correlation matrix P is defined to be a Markov process whose increments over time δt are independent random vectors which are multivariate normally distributed with mean 0 and covariance matrix $P\delta t$.*

This is exactly analagous to the definition of 1-d Brownian motion.

Algorithm. *We can simulate d-dimensional Brownian motion as follows:*

(i) Take A to be a pseudo-square root of P, so $P = AA^T$.

(ii) Generate X_t by the difference equation:

$$X_{t+\delta t} = X_t + A\sqrt{\delta t}\epsilon$$

7.3.2 Simulating stochastic differential equations

The material in this section is a brief summary of some key results from the book "Numerical Solution of Stochastic Differential Equations" by Kloeden and Platen. We won't give proofs.

A 1-dimensional stochastic differential equation can be written as:

$$\mathrm{d}X_t = a(X,t)\mathrm{d}t + b(X,t)\mathrm{d}W_t.$$

An n dimensional stochastic differential equation looks essentially identical but one uses vector notation.

$$\mathrm{d}\mathbf{X}_t = \mathbf{a}(\mathbf{X},t)\mathrm{d}t + \mathbf{b}(\mathbf{X},t)\mathrm{d}\mathbf{W}_t \tag{7.6}$$

where

$$\mathbf{X}_t \in \mathbb{R}^n$$

$$\mathbf{a}(\mathbf{X},t) \in \mathbb{R}^n$$

$$\mathbf{b}(\mathbf{X},t) \text{ is an } n \times d \text{ matrix}$$

$\mathbf{W}_r$ is a d-dimensional vector of correlated Brownian motions, with correlation matrix P.

Whether we are working in 1 or more dimensions we will have an initial condition X_0. For the rest of this section we will drop the bold face for vectors and matrices, as the formulae are essentially the same in one or more dimensions.

Definition. *The Euler scheme for the SDE* (7.6) *with time step δt is given by the difference equation:*

$$\tilde{X}_i = \tilde{X}_{i-1} + a(X_{i-1},t)\delta t + b(X_{i-1},t)\delta W_i$$

where

$$\delta W_i := W_{i\delta t} - W_{(i-1)\delta t}.$$

Note that $\tilde{X}_i$ approximates the value of X at time point i, so i is always an integer and corresponds to the actual time $i\delta t$. (This notation is convenient when working with vectors on a computer.)

The following result is proved in Kloeden and Platen.

Theorem 2. *Suppose that:*

$$|a(x,t) - a(y,t)| + |b(x,t) - b(y,t)| < K_1|x-y|$$

and

$$|a(x,t)| + |b(x,t)| < K_2(1+|x|)$$

and

$$|a(x,s) - a(x,t)| + |b(x,s) - b(x,t)| < K_3(1+|x|)|s-t|^{-\frac{1}{2}}$$

for some constants K_1, K_2, K_3 and all s,t,x,y. Then

$$E(|X_T - \tilde{X}_{(T/\delta t)}|) \leq K_4 \delta t^{\frac{1}{2}}$$

for some constant K_4.

This shows that the $\tilde{X}_i$ converge to $X_{i\delta t}$ in expectation and give the rate of convergence. Note that in the exam I don't expect you to remember the full statement of this theorem, but I do expect you to remember the rate of convergence in expectation.

Convergence in expectation implies convergence in distribution.

Normally we are not actually given the Browniam motion process W_t, we have to simulate this. As we have discussed, this can be done using Cholesky decomposition. The net effect is that we first simulate a d-dimensional vector of normally distributed ϵ_i with mean 0 and standard deviation 1 and correlation matrix P using Cholesky decomposition. We then define:

$$\tilde{X}_i = \tilde{X}_{i-1} + a(X_{i-1},t)\delta t + b(X_{i-1},t)(\sqrt{\delta t})\epsilon_i$$

Note the $\sqrt{\delta t}$.

Example 1: For the process

$$\mathrm{d}X_t = a\mathrm{d}t + b\mathrm{d}W_t$$

with a and b constants then the solution is Euler scheme is exact. Note this is elementary and does not use the above general result on convergence.

When we simulate Brownian motion, we can choose the time steps δt to be as large as we like because the Euler scheme is exact. However, for more general processes, we will need to choose rather small δt because of the slow rate of convergence.

Note that for geometric Brownian motion, we can make a change of variables to turn this into Brownian motion. We can then simulate this exactly. This is why we do not use the Euler scheme to simulate geometric Brownian motion directly.

Although proving the general convergence result is beyond the scope of the course we can at least check it is true. We simply want to test if:

$$E(|X_T - \tilde{X}_T|) \leq K_4 \delta t^{\frac{1}{2}}$$

for an example process. Let's find an interesting process we can solve. Take

$$X_t = \sin(W_t)$$

so

$$\mathrm{d}X_t = -\frac{1}{2} X_t \mathrm{d}t + \sqrt{1 - X_t^2}\mathrm{d}W_t$$

Assuming that we have a vector of the increments of Brownian motion stored in the parameter `dW`, the following code can be used to simulate X_t using the Euler scheme.

```
function [ X ] = simulateSinEuler( X0, dW, dt, nSteps )
% Simulate the following process
% dX = -1/2 X + sqrt(1-X^2) dW
% Note this is obtained by taking the sin of brownian motion

currX = X0;
nPaths = size( dW, 1);
X = zeros(nPaths, nSteps );
for i=1:nSteps
    currDW = dW(1:end,i);
    X(1:end,i) = currX - 0.5*currX*dt + sqrt(1-currX.^2).* currDW;
    currX = X(1:end,i);
end

end
```

On the other hand, we know the exact solution, so we can compute the ahsolute value of the difference between the euler scheme and the true solution. We store this in a variable called `eulerErrors`.

We then want to compute the expectation of this variable. We use a helper function `ninetyPercentConfidence` to actually find an 90% confidence upper bound on this expectation.

```
    dW = randn( nPaths, nSteps(j) )*sqrt(dt);

    exactPaths = sin(X0+cumsum(dW,2));

    eulerPaths = simulateSinEuler( X0, dW, dt, nSteps(j) );
```

```
    eulerErrors = abs(eulerPaths(1:end,end)-exactPaths(1:end,end));
    eulerError(j) = ninetyPercentConfidence(eulerErrors);
```

If we generate a log-log plot of the upper level of the confidence interval against the number of steps we expect to see a slope of gradient $-\frac{1}{2}$. See **??**.

7.3.3 An application to finance

We recall that by simulating asset price processes in the risk neutral measure we can compute derivative prices using risk-neutral valuation.

So if we have a complex asset price model, we can try to simulate it using the Euler scheme. This is unncessary for the Black–Scholes model as we know a better way to simulate geometric Brownian motion. However, it can be useful for more general processes.

You will have seen interesting $\mathbb{Q}$ measure models in your interest rate course and I would encourage you to simulate some of them using the Euler scheme and use this to price derivatives.

As a concrete example, we will simulate the Heston model. This is quite a complex modell that requires all that we have learned in this lecture.

The Heston model (with interest rate of 0) assumes that in the $\mathbb{Q}$ measure the stock price and volatility respectively obey:

$$\mathrm{d}S_t = \sqrt{v_t} S_t \mathrm{d}W_t^1$$

$$\mathrm{d}v_t = \kappa(\theta - v_t)\mathrm{d}t + \xi\sqrt{v_t}\mathrm{d}W_t^2$$

where $\mathrm{d}W_t^1$ and $\mathrm{d}W_t^2$ are Brownian motions with correlation ρ.

- θ is the long run variance
- κ is the mean reversion rate
- ξ is the volatility of volatility

Because of the $\sqrt{v_t}$ in these formulae we need to be careful to ensure that v_t is always positive. This can be done by requiring that $2\kappa\theta > \xi^2$.

We note that$r = 0$ so we require that S is a martingale for this to be a valid $\mathbb{Q}$ measure model. This is why there is no drift term.

Notice that in Black–Scholes model there is a unique compatible $\mathbb{Q}$ model for a given $\mathbb{P}$. This isn't true for more general models, so one usually takes the $\mathbb{Q}$ model as given and does not attempt to derive it from a $\mathbb{P}$ measure model.

Note that I do not expect you to remember the formulae for the Heston model in the exam.

Definition. *Given the market price of a European put or call option, the implied volatility is the value that you must put into the Black–Scholes formula to get that market price.*

According to the Black–Scholes model, the implied volatility will be the same for all options irrespective of their strike and maturity. In practice one find that the implied volatility depends on both of these parameters. If one plots the implied volatility against strike one typically obtains a convex curve or "smile". This is called the volatility smile.

We would like to use the Heston model to price options to see if it can explain the volatility smile.

Our existing pricing code looks like this:

```
function ret = priceByMonteCarlo(...)
    paths = generatePricePaths(...);
    payoffs = computeOptionPayoffs(...);
    ret = mean(payoffs)*exp(-r*T);
end
```

All we need to do is to change this to use the Heston model to generate price paths.

```
function [ prices, variances ] = generatePricePathsHeston( ...
    S0, v0, ...
    kappa, theta, xi, rho, ...
    T, nPaths, nSteps)
%GENERATEPRICEPATHSHESTON Generate price paths according to the
%    Heston model
prices = zeros( nPaths, nSteps );
variances = zeros( nPaths, nSteps );

currS = S0;
currv = v0;
dt = T/nSteps;
for i=1:nSteps
    epsilon = randnMultivariate( [1 rho; rho 1], nPaths  );
    dW1 = epsilon(1,:)*sqrt(dt);
    dW2 = epsilon(2,:)*sqrt(dt);
    currS = currS + sqrt( currv).* currS .* dW1';
    currv = currv + kappa*(theta - currv)*dt + xi*sqrt( currv).* dW2';
    currv = abs( currv ); % Forcibly prevent negative variances
    prices( :, i) = currS;
    variances( :, i) = currv;
end
```

This code contains a hack. Although the exact SDE for the volatility ensures that it is always positive, our approximation is only an approximation and sometimes negative volatilities do occur. We simply take the absolute value of the volatility at each time step to prevent this occuring.

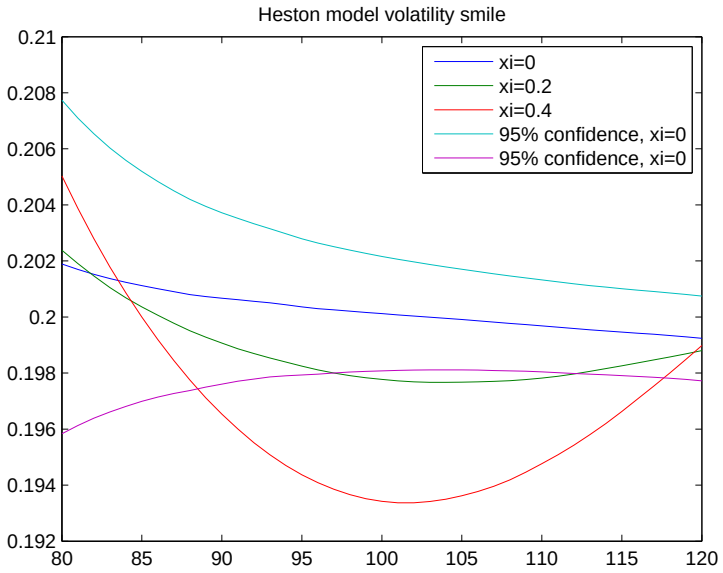
Note that our code to simulate the Heston model is quite easy to adapt to other SDEs, simply change the two lines of code that correspond to the equations of the Heston model and delete the hack where we take absolute values.

I ran the simulation with the following parameters

- `nPaths=100000`
- `nSteps=50`
- $T = 1$
- $S_0 = 1$
- $\kappa = 2$
- $\theta = 0.04$
- $v_0 = 0.04$
- $\rho = 0$

I then used three different values of ξ.

I plotted the Black–Scholes implied volatility for a number of strikes and for the three different values of ξ. Also, since the Monte Carlo calculation of the option prices is only approximate, I included a 95% confidence interval for the case when $\xi = 0$.



If our pricing code was completely accurate, the curve shown for $\xi = 0$ should actually be a straight line value 0.2. That is because if there is no volatility of volatility, the Heston model simplifies to the Black-Scholes model. This expected result is between the error bars, so this gives a test of our calculation.

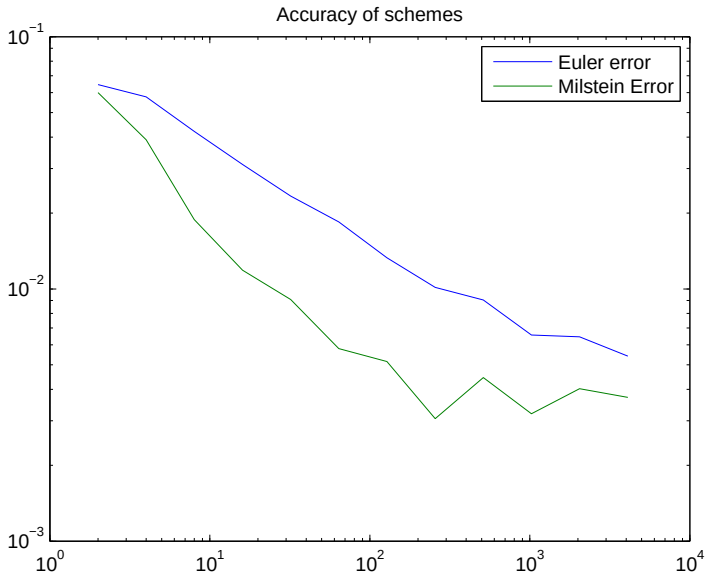


Figure 7.1: Convergence of Euler and Milstein schems. x-axis is number of steps. Vertical axis is Expectation of absolute value of error.

One can clearly see a smile in the case $\xi = 0.4$. This smile extends beyond the error bars, so it seems that the smile really is a consequence of the Heston model and not simply inaccuracies in our Monte Carlo calculation.

7.3.4 Other schemes

The Euler scheme isn't the end of the story. There are many other numerical schemes and the book Kloeden and Platen discusses many of them in detail.

As an example, the *Milstein scheme* for:

$$\mathrm{d}X_t = a(X_t, t)\mathrm{d}t + b(X_t, t)\mathrm{d}W_t$$

is to take

$$\tilde{X}_i = \tilde{X}_{i-1} + a\delta t + b\delta W_t + \frac{1}{2} b \frac{\partial b}{\partial x} \left((\delta W_t)^2 - \delta t \right).$$

This is Euler scheme plus one more term Under certain bounds on the coefficients, it can be shown to converges in expectation at a rate $O(\delta t)$. n-d versions exist but are more complex and so are not often used in practice.

It is easy to implement the Milstein scheme for the sin of Brownian motion to test this rate of convergence. Figure 7.1 shows a plot I obtained of the results.

7.4 Further Reading

The full theory of simulating discrete time stochastic processes can be found in [1].

Bibliography

[1] P. Kloeden and E. Platen. *Numerical solution of Stochastic Differential Equations*. Springer, 1999.