Computational Methods in Financial Mathematics Course Project

by

Zhipeng Lin

A Project

Submitted to the Faculty

of the

WORCESTER POLYTECHNIC INSTITUTE

In partial fulfillment of the requirements for the

Degree of Master of Science

in

Mathematics Science

by

May 2009

APPROVED:

Professor Marcel Blais, Major Project Advisor

Professor Bogdan M. Vernescu, Head of Department

Abstract

This course project is made up of two parts. Part one is an investigation and implementation of pricing of financial derivatives using numerical methods for the solution of partial differential equations. Part two is an introduction of Monte Carlo methods in financial engineering. The name of course is MA573:Computational Methods in Financial Mathematics, spring 2009, given by Professor Marcel Blais.

Acknowledgements

I would like to express my gratitude to my advisor Professor Marcel Blais who help me accomplish my project.

Contents

List of Figures

Part I

Introduction

This project is a summary of the course : Computational Methods in Financial Mathematics, it introduces the Finite Difference Method and Monte Carol Methods to solve the option pricing problem. The text of this project is based on the course lecture and added some extra examples.

Part II

Finite Differnce Methods

0.1 Options

Call Option:

An European call option is a financial contract with the following conditions:
At a prescribed time in the future, the expire date, the holder of the option may purchase a prescribed asset, known as the underlying asset, for orescribed amount, called the strike price. It has the properites as follows:

1. Holder has a right not a obligation.

2. Seller potentially has an obligation.

3. It has value.

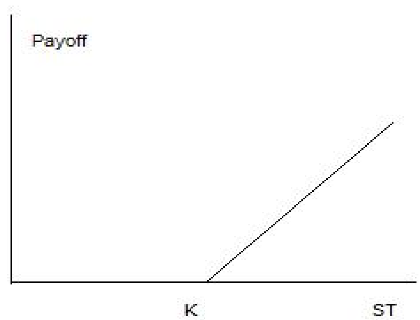


Figure 1: Call Option

Put Option:

An European put option is a financial contract with the same conditions as a European call except the holder has the right to sell the underlying to the writer at expiry for the strike price.

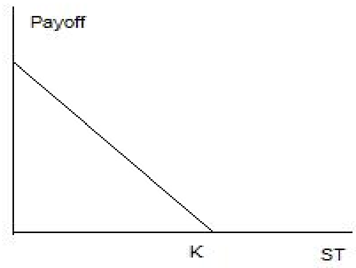


Figure 2: Put option

Some other types

European digital call option

payoff= \$1 if $S_T \geq K$

= \$0 if $S_T \leq K$

American option

It is a financial contract between two parties, the holder and the writer, with expiry time T. At any time t, the holder may exercise the option and receive the payoff $g(t, S_t)$, where S_t is the time t value of the underlying.

Asian call option

An asian option is an option where the payoff is not determined by the underlying price at maturity but by the average underlying price over some period of time.

payoff=max$(A_T - K, 0)$, where

A_T=avg(S_t:$0 \leq t \leq T$)

$=\frac{1}{T}\int_0^T S_t dt$

Discrete time model

Model the underlying asset price movement using a sequence of coin tosses, at time$t \in \{0, 1, 2, ...\}$

Black-Scholes Model

It is a continuous time model, and it is the limit of the binomial with a specific choice of parameters as $\Delta t \to 0$

It's SDE:

$$\frac{dS_t}{S_t} = udt + \sigma dW_t \tag{1}$$

Option Pricing Approachs

1. Find the theory of asset pricing.

   - use of probability - price option by taking the discounted expected payoff under the risk-neutral measure.

   - get algorithmic solution in some cases.

2. Option replication

   - create a synthetic option, use positions to replicate the option value.

3. Solving PDE's

   Often time, an option's value can be determined by solving a boundary value problem, which is a PDE and a set of boundary conditions.

   -sometimes can find algorithmic solutions(usually approximate the solution using finite difference methods)

   -deal with discretizing the continuous models

4. Monte Carlo Methods

   -approximate an option's value by simulation

0.2 Stochastic Process

A stochastic process X is a collection of random variables

$(X_t, t \in [0, \pi]) = X_t(\omega) : t \in [0, \pi], \omega \in \Omega)$, where Ω is sample space.

For fixed t, $X_t(\omega)$ is a random variable for $\omega \in \Omega$ fixed$X_t = X_t(\omega)$ is a function of time, called sample path.

A stochastic process $\omega = (\omega_t, t \in [0, \infty])$ is called a standard Brownian motion if the followings are satisfied:

1. $\omega_0 = 0$

2. for 0$\leq s \leq t$

   $\omega_t - \omega_s \in N(0, t - s)$

3. Independence of increments for$0 \leq s < u < v$

   $W_t - W_s \perp (W_v - W_u)$

4. W has continuous sample paths

Example:

Brownian Motion with drift $X_t = \mu t + \sigma W_t$

E(X_t)=E($\mu t + \sigma W_t$)

=E(μt)+E(σW_t)

=μt+σE(W_t)

=μt

Black-Scholes-Merton

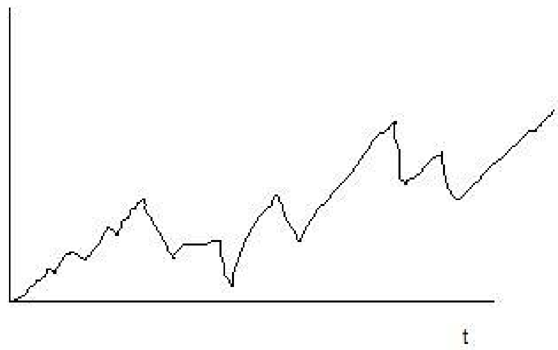


Figure 3: BrownianMotion

SDE: consider a small time interal dt, during which an asset price changes from s to s+ds, decompose it into 2 parts

1. One part comes from a fixed rate of return over dt:
   μdt, μ is called the drift

2. The random compoment is given by a random sample drawn from a normal dist, with mean 0 and variance $dt, \sigma dW_t$, σ is called the volatility.

$$\frac{dS}{S} = \mu dt + \sigma dW_t \tag{2}$$

Thm:

The stochastic process that solves the SDE $\frac{dS}{S} = \mu dt + \sigma dW_t$ is the geometric

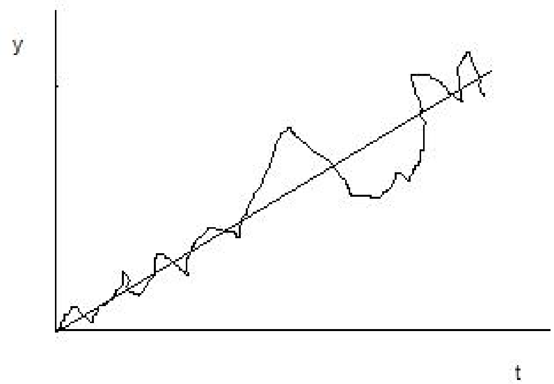


Figure 4: BrownianMotion

Brownian Motion.

$$S_t = S_0 e^{\mu - \frac{1}{2}\sigma^2} t + \sigma W_t \tag{3}$$

Ito process

X_t is an Ito process if it is a stochastic process that can be written as:

$$dX_t = u_t dt + v_t dW_t \tag{4}$$

Ito's formula is a basic tool for determining the Ito process followed by a function of other Ito processes.

Suppose X_t is an Ito's process and $g_{t,x} \in C^2([0, \infty] \times R)$, then $X_t = g(t, X_t)$ is also an Ito process and

$$dY_t = g_t(t, X_t)dt + g_x(t, X(t))dX_t + \frac{1}{2}g_{xx}(t, X_t)dX_t dX_t \tag{5}$$

$$dtdt = 0, dtdW_t = 0, dW_t dW_t = dt \tag{6}$$

Example:

$S_t = S_0 e^{\mu - \frac{1}{2}\sigma^2 t} + \sigma W_t$

$g(t,x) = S_0 e^{\mu - \frac{1}{2}\sigma^2 t} + \sigma X$

$g_t = (\mu - \frac{1}{2}\sigma^2)g, g_x = \sigma g, g_{xx} = \sigma^2 g$

Back to option

Consider a European option with time-t,value $V(t, S_t)$, the option payoff is

$V(T, S_T)$

Let $V(t, S_t) = g(t, S_t)$

Using Ito's formula, we get

$$dV = g_t dt + g_x dS_t + \frac{1}{2}g_{xx}dS_t dS_t \tag{7}$$

$$dS_t = \mu S_t dt + \sigma S_t dW_t \tag{8}$$

Plus (8) into (7)

$dV = g_t dt + g_x(\mu S_t dt + \sigma S_t dW_t) + \frac{1}{2}g_{xx}(\mu^2 S_t^2 dtdt + 2\mu\sigma S_t^2 dtdW_t + \sigma^2 S_t^2 dW_t dW_t)$

$= (g_t + \mu g_x S_t + \frac{1}{2}g_{xx}\sigma^2 S_t^2)dt + g_x \sigma S_t dW_t$

This is the ODE for option price

Form a portfolio with one option and Δ units of underlying, at time t, the value of the portfolio is

$$\Pi_t = V_t - \Delta S_t \tag{9}$$

$$d\Pi_t = dV_t - \Delta dS_t \tag{10}$$

where Δ is fixed at the beginning of dt

Plug in dV, we get

$d\Pi_t = (\frac{\partial V}{\partial t} + \mu S_t \frac{\partial V}{\partial S} \frac{1}{2}\sigma^2 S_t^2 \frac{\partial^2 V}{\partial S^2})dt + \frac{\partial V}{\partial S}\sigma S_t dW_t - \Delta(\mu S_t dt + \sigma S_t dW_t)$

$\quad = (\frac{\partial V}{\partial t} + \mu S_t \frac{\partial V}{\partial S} + \frac{1}{2}\sigma^2 S_t^2 \frac{\partial^2 V}{\partial S^2} - \mu S_t \Delta)dt + (\sigma S_t \frac{\partial V}{\partial S} - \sigma \Delta S_t)dW_t$

Then choose the value of Δ, let $\Delta = \frac{\partial V}{\partial S} \Rightarrow 0$

$$d\Pi_t = (\frac{\partial V}{\partial t} + \frac{1}{2}\sigma^2 S_t^2 \frac{\partial^2 V}{\partial S^2})dt \tag{11}$$

We find that there is no randomness

Arbritrage

Suppose the amount Π_t was invested in the money market account(MMA)

$d\Pi = r\Pi dt$ must hold, otherwise there would be arbitrage.

$$d\Pi_t = r(V_t - \frac{\partial V}{\partial S} S_t)dt \tag{12}$$

Combining (11) and (12), we get

$$\frac{\partial V}{\partial t} + \frac{1}{2}\sigma^2 S_t^2 \frac{\partial^2 V}{\partial S^2} + rS_t \frac{\partial V}{\partial S} - rV = 0 \tag{13}$$

This is the Black-Scholes PDE, it has the properities as follows:

1. Any option whose price depends only on t and S_t is paid for up front must satisfy this PDE

2. Boundary conditions are needed to price an option

Example:

European call option with strike price K, expiry T final condition : $C(T, S_T) = max(S_T - K, 0)$

Spatial condition

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

Since S can not escape from 0, we set $C(0, t) = 0$

As $S \to \infty$, the call option becomes more likely to be exercised.

So we set $C(S, T) \to S$ as $S \to \infty$

The Boundary Value Problem(B.V.P) works backward in time before we give an end condition instead of an initial

Make transformation $t\hat{\wedge} = T - t$

Rewrite the BSM PDE as

$$\frac{\partial V}{\partial t\hat{\wedge}} = \sigma S\frac{\partial V}{\partial S} + \frac{1}{2}\sigma^2 S^2\frac{\partial^2 V}{\partial S^2} - rV \tag{14}$$

To approximate the partial deviratives in the above PDE, we use finite differnce methods to solve it.

First of all, let us recall Taylor's Theory

Suppose $u \in C^\infty$, then $u(x) = \sum_{n=0}^{\infty} \frac{u^u(a)}{n!}(x-a)^n$

Consider u(x+h) and u(x-h), we expand both in taylor series at a=x

$u(x+h) = \sum_{n=0}^{\infty} \frac{u^n(x)}{n!}((x+h)-x)^n$

$=\sum_{n=0}^{\infty} \frac{u^n(x)}{n!}h^n$

$u(x-h) = \sum_{n=0}^{\infty} \frac{u^n(x)}{n!}h^n(-1)^n$

And u(x+h)+u(x-h)=$2u^{'}(x)h + 2\frac{u^{''}(x)}{3!}h^3 + O(h^4)$

Divide both sides by 2h, we get

$\frac{u(x+h)+u(x-h)}{2h} = u^{'}(x) + \frac{u^{''}}{3!}h^2 + O(h^3)$

Rearrange it , we get $\frac{\partial u}{\partial x} = \frac{u(x+h)-u(x-h)}{2h} + O(h^2)$

and it is a first-order central finite difference method

$u(x+h)+u(x-h)=2u(x)+u''(x)h^2+\frac{2u'''(x)h^4}{4!}+O(h^6)$

Divide both sides by h^2, we get

$\frac{\partial^2 u}{\partial x^2}=\frac{u(x+h)-2u(x)+u(x-h)}{h^2}$

So we get

$$\frac{\partial u}{\partial x}(x,t)=\frac{u(x+\Delta x,t)-u(x-\Delta,t)}{2\Delta x}-TE \tag{15}$$

$$\frac{\partial^2 u}{\partial x^2}=\frac{u(x+\Delta x,t)-2u(x,t)+u(x-\Delta x,t)}{(\Delta x)^2}-TE_2 \tag{16}$$

where TE, TE_2 are both $O(\Delta x^2)$

0.3 Order Notation

Given two sequences, $\{a_n\}_{n=1}^{\infty}$ and $\{b_n\}_{n=1}^{\infty}$, we write $a_n=O(b_n)$, if $\exists$ C >0 such that $|a_n|\leq C|b_n|\forall n$, and we write $a_n=o(b_n)$, if $\lim_{n\to\infty}\frac{a_n}{b_n}=0$.

Space discrelization

Use finite difference from last time on:

$\frac{\partial u}{\partial t}=r\frac{\partial u}{\partial x}+\frac{1}{2}\sigma^2\frac{\partial^2 u}{\partial x^2}-ru$

We set up a grid with grid points $x_0, x_1, x_2....x_I$, Δx=$x_{i+1}-x_i$

Let $u_i=u(x_i)$, use finite difference to approximate $\frac{\partial u}{\partial x}$ and $\frac{\partial^2 u}{\partial x^2}$ at the grid points.

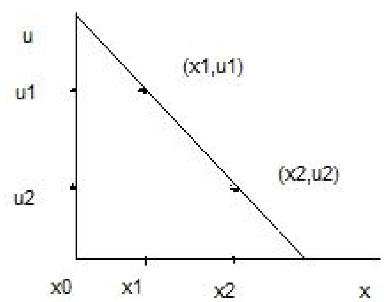
$\frac{\partial u}{\partial x}(x_i,t)=\frac{u(x_i+\Delta x,t)-u(x_i-\Delta x,t)}{2\Delta x}-TE$=$\frac{u_{i+1}-u_{i-1}}{2\Delta x}-TE_1$

$\frac{\partial^2}{\partial x^2}(x_i,t)=\frac{u(x_i+\Delta x,t)-2u(x_i,t)+u(x_i-\Delta x,t)}{(\Delta x)^2}-TE_2$=$\frac{u_{i+1}-2u_i+u_{i-1}}{(\Delta x)^2}-TE_2$

for i=1, 2, 3 ..., I-1

Boundary value u_0 and u_I

Example: for linear condition



line: $u - u_i = m(x - x_i)$

$$m = \frac{u_2 - u_1}{x_2 - x_1} = \frac{u_2 - u_1}{\Delta x}$$

$$u - u_1 = \frac{u_2 - u_1}{\Delta x}(x - x_1),\ u_0 - u_1 = \frac{u_2 - u_1}{\Delta x}(x_0 - x_1)$$

$$u_0 - u_1 = u_1 - u_2,\ u_0 = 2u_1 - u_2$$

Similarly, $u_I = 2u_{I-1} - u_{I-2}$

$$\frac{\partial u}{\partial t} = r\left(\frac{u_{i+1} - u_{i-1}}{2\Delta x}\right) + \frac{1}{2}\sigma^2\left(\frac{u_{i+1} - 2u_i + u_{i-1}}{\Delta x^2}\right) - ru_i + \epsilon$$

$$= u_{i-1}\left(\frac{-r}{2\Delta x} + \frac{\sigma^2}{2\Delta x^2}\right) + u_i\left(\frac{-\sigma^2}{\Delta x^2} - r\right) + u_{i+1}\left(\frac{r}{2\Delta x} + \frac{\sigma^2}{2\Delta x^2}\right) + \epsilon$$

$$= \beta u_{i-1} - \gamma u_i + \alpha u_{i+1}$$

where $\alpha = \frac{1}{2}[\frac{r}{\delta x} + \frac{\sigma^2}{\Delta x^2}]$, $\gamma = \frac{\sigma^2}{\Delta x^2} + r$,$\beta = \frac{-r}{2\Delta x} + \frac{\sigma^2}{2\Delta x^2}$

Then we need to vectorize,

Let $\underline{u} = \begin{bmatrix} u_0 \\ u_1 \\ . \\ . \\ . \\ u_I \end{bmatrix}$ boundary : $u_0 = 2u_1 - u_2, u_I = 2u_{I-1} - u_{I-2}$

$A\underline{u} = \begin{bmatrix} u_0 - 2u_1 + u_2 \\ \beta u_0 - \gamma u_1 + \alpha u_2 \\ \beta u_1 - \gamma u_2 + \alpha u_3 \\ ...... \\ ...... \\ u_I - 2u_{I-1} + u_{I-2} \end{bmatrix}$ $\epsilon = \begin{bmatrix} \epsilon_1 \\ \epsilon_2 \\ \epsilon_3 \\ . \\ . \\ \epsilon_I \end{bmatrix}$ $A = \begin{bmatrix} 1 \ -2 \ 1 \ 0 \ 0 \ ... \ 0 \\ \beta \ -\gamma \ \alpha \ 0 \ 0 \ ... \ 0 \\ 0 \ \beta \ -\gamma \ \alpha \ 0 \ ... \ 0 \\ .... \\ .... \\ .... \\ 0 \ 0 \ ... \ 0 \ 1 \ -2 \ -1 \end{bmatrix}$

$$\frac{d\underline{u}}{dt} = A\underline{u} + \underline{\epsilon} \tag{17}$$

Use $\frac{d\underline{u}}{dt} \approx A\underline{u}$, we solve this system numerically.

Discretize form $\{t_0, t_1, ..., t_N\}$

Denote $u_i^n = u_i(t_n) = u(x_i, t_n)$, then we can find a solution at time t_n using information from $t_0, t_1, ..., t_{n-1}$, fix i, we drop subscript, and get

$u_i^n = u^n = u(t_n) = u(t_{n-1} + \Delta t)$

Using Taylor's series to $u(t + \Delta t)$ about $a = t_{n-1}$

$u(t + \Delta t) = \sum_{k=0}^{\infty} \frac{u^{(k)}(t_{n-1})}{k!}(t + \Delta t - t_{n-1})$

$\quad\quad =u(t_{n-1}) + \frac{du}{dt}(t_{n-1})(t + \Delta t - t_{n-1}) + \frac{d^2u}{dt^2}(t_{n-1})(t + \Delta t - t_{n-1})$

Use a first-order approximation, we get

$u(t_{n-1} + \Delta t) = u(t_{n-1}) + \frac{du}{dt}(t_{n-1})\Delta t + TE_{\Delta t}$

As we mentioned above, replace $\frac{du}{dt}(t_{n-1})$ by $(\frac{du}{dt})^{n-1}$, we get

$u^n = u^{n-1} + (\frac{du}{dt})^{n-1}\Delta t + TE_{\Delta t}$

Vary i from 1 to I, it gives

$\underline{u}^n = \underline{u}^{n-1} + (\frac{d\underline{u}}{dt})^{n-1}\Delta t + TE_{\Delta t}$

$\Rightarrow \underline{u}^n = \underline{u}^{n-1} + (A^{n-1}\underline{u}^{n-1})\Delta t$

Numercal Issues

1. Consistency: A numercal scheme is consistent if the finite difference scheme converges to the PDE as the space and time steps $\rightarrow 0$,error terms are $O(\Delta x^2)$ and $O(\Delta t^2)$

2. Stability: A numercal scheme is stable if the difference between the numbercal solution and exact solution remains bounded as the number of time steps $\rightarrow 0$

Another Finite Differnce Discretization

$\frac{\partial u}{\partial x} = \frac{4u(x+\Delta x,t) - u(x+2\Delta x,t) - 3u(x,t)}{2\Delta x} + O(\Delta x^2)$

Stability Analysis

We use stability analysis to judge whether a method fail or not.

Pricing using finite difference

1. discretize in space, transforms the PDE into a system of ODE's

2. discretize in time to solve ODE's system of equations(PDE's)

Analysis of the finite difference equations(PDE's)

1. find a local analytic solution to the system of ODE's in (1) above
2. find a local analytic solution of the PDE system that came from (2) above
3. compare them-relates Δx and Δt

Linear Algebra Review

Eigenvalues: let $A \in R^{n\times n}$, if $\exists$ a vector $\underline{x} \in R^n$, such that $\underline{x} \neq \underline{0}$ and $A\underline{x} = \lambda\underline{x}$, then λ is called an eigenvalue of A.

Consider

$A\underline{x} = \lambda I \underline{x}$

$A\underline{x} - \lambda I \underline{x} = 0$

$(A - \lambda I)\underline{X} = \underline{0} \Longleftrightarrow Det(A - \lambda I) = 0$

Defination: the space of the set of all solutions to $A\underline{x} - \lambda I \underline{x} = 0$ is called the eigenspace of A.

Example: $A = \begin{bmatrix} 7 & -1 \\ 4 & 3 \end{bmatrix}$

λ of A is 5

so for $(\lambda I - A)\underline{x} = \underline{0}$ $\begin{bmatrix} -2 & 1 \\ -4 & 2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$

Solution is $\begin{matrix} x_1 = \frac{1}{2}t \\ x_2 = t \end{matrix}$

The eigenspace of A is $i = \left\{ \begin{bmatrix} \frac{1}{2} \\ 1 \end{bmatrix} t : t \in R \right\}$

Space Discretization

Consider $\frac{\partial u}{\partial t}$= Lu, where L is a partial differential operator with no time derivatives. For example:

$\frac{\partial u}{\partial t} = \frac{\partial^2 u}{\partial x^2} - \frac{4\partial u}{\partial x}$

L=$(\frac{\partial^2}{\partial x^2} - \frac{4\partial}{\partial x})$

In our example, space discretization gave $\frac{d\underline{u}}{dt} = A\underline{u}$

Assume A is non-singular, rank M, so $\underline{X_m}\ for\ m = 1, 2, 3, ..., M$ and eigenvalues

$\lambda_1, \lambda_2, ..., \lambda m$

Define $A\underline{X}_m = \lambda_m \underline{X}_m$

$X = [\underline{X}_1|\underline{X}_2|\underline{X}_3|...|\underline{X}_m]$, non-singular $\lambda = \begin{bmatrix} \lambda_1\ 0\ 0\ ...\ 0 \\ 0\ \lambda_2\ 0\ ...\ 0 \\ ... \\ ... \\ 0\ 0\ ...\ 0\ \lambda_m \end{bmatrix}$

$AX = [A\underline{X}_1|A\underline{X}_2|...|A\underline{X}_m] = [\lambda_1\underline{X}_1|\lambda_2\underline{X}_2|..|\lambda_m\underline{X}_m] = \lambda X$

$\Longrightarrow AX = \lambda X$

So we get $X^{-1}AX = \lambda$

$\longrightarrow\ X^{-1}\frac{d\underline{u}}{dt} = X^{-1}A\underline{u} = X^{-1}A(XX^{-1})\underline{u}$,where $\lambda = X^{-1}AX$

$\longrightarrow X^{-1}\frac{d\underline{u}}{dt} = \lambda X^{-1}\underline{u}$

If the elements of X are independent of time, $\frac{d(X^{-1}\underline{u})}{dt} = \lambda X^{-1}\underline{u}$

Define $\underline{V} = X^{-1}\underline{u}$, then $\frac{d\underline{V}}{dt} = \lambda\underline{V}$

We get $\begin{bmatrix} \frac{d\underline{V}_1}{dt} \\ \frac{d\underline{V}_2}{dt} \\ ... \\ ... \\ \frac{d\underline{V}_m}{dt} \end{bmatrix} = \begin{bmatrix} \lambda_1 V_1 \\ \lambda_2 V_2 \\ ... \\ ... \\ \lambda_m V_m \end{bmatrix}$ with solution $\underline{V}^* = \begin{bmatrix} C_1 e^{\lambda_1 t} \\ C_2 e^{\lambda_2 t} \\ ... \\ ... \\ C_m e^{\lambda_m t} \end{bmatrix}$

where C_i are constants

0.4 Time discretization

Let $V^n = V(n\Delta t)$, the shift operator E^i is defined by $E^i V^n = V^{n+i} = V[(n+i)\Delta t]$.

Time discretized finite difference can be expressed as polynomial in the shift operator. For example:

Suppose $\frac{\partial u}{\partial t} \approx \frac{u(x,t+\Delta t)-u(x,t)}{\Delta t}$,$t_n = t_0 + n\Delta t = n\Delta t$

$\frac{\partial u}{\partial t}(t_n) = (\frac{\partial u}{\partial t})^n \approx \frac{u(t_n+\Delta t)-u(t_n)}{\Delta t}$

$= \frac{u(n\Delta t+\Delta t)-u(n\Delta t)}{\Delta t} = \frac{u^{n+1}-u^n}{\Delta t} = \frac{Eu^n-u^n}{\Delta t}$

$= \frac{u^n}{\Delta t}E - \frac{u^n}{\Delta t}$

Consider a polynomial in the shift operator

$P(E) = \sum_{i=0}^{n} a_i E^i$

An equation of the form $P(E)V^n = 0$ is known as a homogeneous difference equation and has the solution $V^n = \sum_{k=1}^{K} b_k(\Lambda_k)^n$, where the b_k are constants and the Λ_k are the roots of the polynomial P(Λ).

Consider this system $\frac{d\underline{V}}{dt} = \lambda \underline{V}$, jth component $\frac{dV_j}{dt} = \lambda_j V_j$

has the analytic solution: $v_j(t) = c_j e^{\lambda_j t}$

Finite difference expression:

$\frac{dV_j}{dt}(t_n) = \frac{dV_t}{dt}(n\Delta t) \cong \frac{1}{\Delta t}\sum_{i=1}^{M} C_i V_j^{n+i}$ for some constants C_i

Recall homogeneous difference equation $\hat{P}(\Lambda) - \lambda_j \Delta t = 0$, the λ_j resulted from spatial discretization, $\frac{d\underline{v}}{dt} = \lambda \underline{v}$

The Λ_{jk} resulting from time discretization are resulted to the λ_j by

$\hat{P}(\Lambda_{jk}) - \lambda_j \Delta t = 0$

Thus a space-time discretization gives rise to a set Λ_{jk}. They are called amplification errors.

Since $v_j^n = \sum_{k=1}^{K} C_{jk}(\Lambda_{jk})^n$,if $|\Lambda_{jk}| > 1$, then v_{jk} will blow up as the number of n increases. If any one $|\Lambda_{jk} > 1|$, the scheme is unstable.

The relationship between time and space discretizations is embodied in relationship between λ_j and Λ_{jk}, so we set t=$n\Delta t$ and $v_j(t) = C_j e^{\lambda j t}$ gives $v_j(t) = v_j(n\Delta t) = C_j e^{\lambda_j n\Delta t} = C_j(e^{\lambda_j \Delta t})^n$ and expand the right-hand side by Taylor series:

$v_j(t) = c_{jk}[1 + \lambda_j\Delta t + \frac{1}{2!}(\lambda_j\Delta t)^2 + ...]^n$

To get $v_j(n\Delta t)$ to converge as $\Delta t \to 0$, the following equation must hold for some (at least 1) pair j,k

$\Lambda_{jk} = 1 + (\lambda_j\Delta t) + \frac{(\lambda_j\Delta t)^2}{2!} + ... + \frac{(\lambda_j\Delta t)^P}{P!} + O(\Delta t^{P+1})$

0.5 Specific finite difference methods

1. The Explicit Euler Scheme

   ODE:$\frac{dv}{dt} = \lambda v$, we use the explicit approximation
   $\frac{dv}{dt}|^n \cong \frac{v^{n+1}-v^n}{\Delta t}$, plug into the ODE, we get:

   $\frac{v^{n+1}-v^n}{\Delta t} \approx \lambda v^n$ or $v^{n+1} = \lambda\Delta t v^n + v^n$,which is called explicit because v^{n+1} depends on previous vales of v.

   $v^{n+1} - v^n - \lambda\delta t v^n = 0 \Rightarrow Ev^n + v^n - \lambda\Delta t v^n = 0 \Rightarrow \underbrace{(E - 1 - \lambda\Delta t)}_{P(E)} v^n = 0$

   To solve $P(E) = 0$,

   $\Lambda - 1 - \lambda\Delta t = 0 \Rightarrow \Lambda = 1 + \lambda\Delta t$

   $\Rightarrow e^{\lambda\Delta t} - \Lambda = \frac{1}{2}(\lambda\Delta t)^2 + \frac{1}{3!}(\lambda\Delta t)^3 + ...$, the model is first-order accurate.

   For stability, we require

   $|\Lambda| \le 1 \Rightarrow -1 \le 1 + 1\lambda\Delta t \le 1 \Rightarrow -2 \le \lambda\Delta t \le 0 \Rightarrow |1 + \lambda\Delta t| \le 1$

   So if $\lambda > 0 \Rightarrow$unstable, if $\lambda < 0 \Rightarrow$ if $\Delta t \le -\frac{2}{\lambda} \Rightarrow$stable

2. The Implicit Euler Scheme

$\frac{dv}{dt}|^{n+1} \approx \frac{v^{n+1}-v^n}{\Delta t}$, plug into ODE $\frac{dv}{dt} = \lambda v$

$\frac{v^{n+1}-v^n}{\Delta t} \approx \lambda v^{n+1}$

$(1-\lambda\Delta t)v^{n+1} - v^n = 0 \Rightarrow (1-\lambda\Delta t)Ev^n - v^n = 0 \Rightarrow \underbrace{[(1-\lambda\Delta t)E-1]}_{P(E)}v^n = 0$

Solve $P(\Lambda) = 0 \Rightarrow (1-\lambda\Delta t)\Lambda - 1 = 0$

$\Lambda = \frac{1}{1-\lambda\Delta t}$=$1 + (\lambda\Delta t) + \frac{1}{2}(\lambda\Delta t)^2 + \frac{1}{3!}(\lambda\Delta t)^3$

Thus, $e^{\lambda\Delta t} - \Lambda = -\frac{1}{2}(\lambda\Delta t)^2 + O(\Delta t\lambda)^3$, so this method is first-order accurate.

For stability we require $|\Lambda| \leq 1 \Rightarrow |\frac{1}{1-\lambda\Delta t}| \leq 1$, that is if $\lambda < 0$, it is stable.

3. Crank-Nicolson Scheme

We average the Explicit and Implicit Euler schemes, and get the

Crank-Nicolson scheme:

$\frac{1}{2}[\frac{dv}{dt}|^n + \frac{dv}{dt}|^{n+1}] = \frac{v^{n+1}-v^n}{\Delta t}$

Rewrite as $P(E)v^n = 0$

$(1-\frac{1}{2}\lambda\Delta t)v^{n+1} - (1+\frac{1}{2}\lambda\Delta t)v^n = 0 \Rightarrow (1-\frac{1}{2}\lambda\Delta t)Ev^n - (1+\frac{1}{2}\lambda\Delta t)v^n = 0$

$P(E) = (1-\frac{1}{2}\lambda\Delta t)E - (1+\frac{1}{2}\lambda\Delta t)$

Solve $P(\Lambda) = 0$, we get

$\Lambda = \frac{1+\frac{1}{2}\lambda\Delta t}{1-\frac{1}{2}\lambda\Delta t} = \frac{1}{1-\frac{1}{2}\lambda\Delta t} + \frac{1}{2}\lambda\Delta t(\frac{1}{1-\frac{1}{2}\lambda\Delta t})$

$\Rightarrow \Lambda = 1 + (\lambda\Delta t) + \frac{1}{2}(\lambda\Delta t)^2 + \frac{1}{4}(\lambda\Delta t)^3 + ...$

So $e^{\lambda\Delta t} - \Lambda = \frac{1}{12}(\lambda\Delta t)^3 + O(\lambda\Delta t)^4$

The scheme is second-order accurate.

Stability Condition:

Suppose $|\Lambda| = |\frac{1+\frac{1}{2}\lambda\Delta t}{1-\frac{1}{2}\lambda\Delta t}| \leq 1$

$\Rightarrow |1+\frac{1}{2}\lambda\Delta t| \leq |1-\frac{1}{2}\lambda\Delta t|$

And $\Lambda = 1 + (\lambda\Delta t) + \frac{1}{2}(\lambda\Delta t)^2 + \frac{1}{4}(\lambda\Delta t)^3 + ...$, so if $\lambda \leq 0, |\Lambda| \leq 1$

0.6 Implementation of the Time Advancement

$\underline{u}^{n+1} = \underline{u}^n + 2t[\frac{du}{dt}]^n + TE_{\Delta t}$

Using Crank-Nicolson, we get

$\underline{u}^{n+1} \approx \underline{u}^n + \frac{\Delta t}{2}[\frac{du}{dt}|^n + \frac{du}{dt}|^{n+1}]$, then insert the spatial discretization $\frac{du}{dt} = A\underline{u}$, gives

$\underline{u}^{n+1} = \underline{u}^n + \frac{\Delta t}{2}[A\underline{u}^n + f^n + A\underline{u}^{n+1} + f^{n+1}$, where$f^n$ and f^{n+1} are vectors.

Rewrite it as $\underline{u}^{n+1} - \frac{\Delta t}{2}A\underline{u}^{n+1} = u^n + \frac{\Delta t}{2}A\underline{u}^n + \frac{\Delta t}{2}(\underline{f}^n + \underline{f}^{n+1})$

$$\Rightarrow \underbrace{(I - \frac{\Delta t}{2})}_{\hat{A}}\underline{u}^{n+1} = \underbrace{(I + \frac{\Delta t}{2}A)\underline{u}^n + \frac{\Delta t}{2}(\underline{f}^n + \underline{f}^{n+1})}_{\underline{b}}$$

$\hat{A}\underline{u}^{n+1} = \underline{b}$, where $\hat{A}$ is sparse, large and bonded.

Solving sparse systems of linear equations

1. Direct Solver

   properties:

   -solve in a finite number of steps

   -can not control the accuracy and it depends on the implementation and the algorithm

   The most popular direct solver is the tridiagonal solver which is applicable to 1-dimensional problems.

   The tridiagonal solver is the Gaussian elimination method.

$$\begin{bmatrix} b_1\ c_1\ 0\ 0\ ...\ 0 \\ a_2\ b_2\ c_2\ 0\ ...\ 0 \\ ...\ ...\ ...\ ...\ ...\ ... \\ ...\ ...\ ...\ ...\ ...\ ... \\ 0\ 0\ ...\ a_{n-1}\ b_{n-1}\ c_{n-1} \\ 0\ 0\ 0\ 0\ a_n\ b_n \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ ... \\ ... \\ u_{n-1} \\ u_n \end{bmatrix} = \begin{bmatrix} f_1 \\ f_2 \\ ... \\ ... \\ f_{n-1} \\ f_n \end{bmatrix}$$

Step 1 Normalization

$$\begin{bmatrix} 1\ \frac{c_1}{b_1}\ 0\ 0\ ...\ 0 \\ a_2\ b_2\ c_2\ 0\ ...\ 0 \\ ...\ ...\ ...\ ...\ ...\ ... \\ ...\ ...\ ...\ ...\ ...\ ... \\ 0\ 0\ ...\ a_{n-1}\ b_{n-1}\ c_{n-1} \\ 0\ 0\ 0\ 0\ a_n\ b_n \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ ... \\ ... \\ u_{n-1} \\ u_n \end{bmatrix} = \begin{bmatrix} \frac{f_1}{b_1} \\ f_2 \\ ... \\ ... \\ f_{n-1} \\ f_n \end{bmatrix}$$

Step 2 Elimination

$$\begin{bmatrix} 1\ \frac{c_1}{b_1}\ 0\ 0\ ...\ 0 \\ 0\ b_2 - \frac{a_2 c_1}{b_1}\ c_2\ 0\ ...\ 0 \\ ... \\ ... \\ 0\ 0\ ...\ a_{n-1}\ b_{n-1}\ c_{n-1} \\ 0\ 0\ 0\ ...\ a_n\ b_n \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ ... \\ ... \\ u_{n-1} \\ u_n \end{bmatrix} = \begin{bmatrix} \frac{f_1}{b_1} \\ f_2 - \frac{a_2 f_1}{b_1} \\ ... \\ ... \\ f_{n-1} \\ f_n \end{bmatrix}$$

Step 3 Normalization

$$
\begin{bmatrix} 1 \ \frac{a_1}{b_1} \ 0 \ 0 \ ... \ 0 \\ 0 \ 1 \ \frac{c_2}{b_2 - \frac{a_2 c_1}{b_1}} \ 0 \ ... \ 0 \\ ... \\ ... \\ 0 \ 0 \ ... \ a_{n-1} \ b_{n-1} \ c_{n-1} \\ 0 \ 0 \ 0 \ ... \ 0 \ a_n \ b_n \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ ... \\ ... \\ u_{n-1} \\ u_n \end{bmatrix} = \begin{bmatrix} \frac{f_1}{b_1} \\ \frac{f_2 - \frac{a_2 f_1}{b_1}}{b_2 - \frac{a_2 c_1}{b_1}} \\ ... \\ ... \\ f_{n-1} \\ f_n \end{bmatrix}
$$

This is continued until the system looks as follows:

$$
\begin{bmatrix} 1 \ x_1 \ 0 \ 0 \ ... \ 0 \\ 0 \ 1 \ x_2 \ 0 \ ... \ 0 \\ ... \\ ... \\ 0 \ 0 \ 0 \ ... \ 1 \ x_{n-1} \\ 0 \ 0 \ 0 \ 0 \ ... \ 1 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ ... \\ ... \\ u_{n-1} \\ u_n \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ ... \\ ... \\ y_{n-1} \\ y_n \end{bmatrix}
$$

The solution follows in the upward sweep:

$u_{n-1} = y_{n-1} - x_{n-1} u_{n-i+1}, i = 1, 2, ..., n-1$

2. Iterative Solvers

   There are two main types of iterative solvers:

   (a) Stationary methods-using iteration schemes with parameters that remain fixed during the iterations -the Jacobi, Gauss-Seidel, SOR methods

   (b) Nonstationary methods-using parameters that are updated as the iteration proceeds -the conjugate gradient family and minimal residual methods

   Preconditioning $\hat{A}\underline{u} = \underline{f}$, for some non-singular matrix C, we use

$\hat{A}C^{-1}C\underline{u} = \underline{f}$, could choose C such that if $\lambda_{\min}$ and $\lambda_{\max}$ are the smallest and largest eigenvalues of $\hat{A}C^{-1}$, then $\frac{\lambda_{\max}}{\lambda_{\min}}$ is close to 1.

The Jacobi method

Consider the system of linear equations

$\sum_{i=1}^{j=n} a_{ij}u_j = f_i$

Solve for u_i, assume that we konw all the u_j for $j \neq i$, we get the solution
$u_i = \frac{1}{a_{ii}} \left\{ f_i - \sum_{i \neq j} a_{ij}u_j \right\}$

This equation suggests an iterative algorithm of the form
$u_i^{n+1} = \frac{1}{a_{ii}} \left\{ f_i - \sum_{i \neq j} a_{ij}u_j^n \right\}$

The Gauss-Seidel Method

This method is a modification of Jacobi method, its property is that the updates to the unknowns are incorporated into the scheme as they occur. The solution is as follows:

$u_i^{n+1} = \frac{1}{a_{ii}} \left\{ f_i - \sum_{j<i} a_{ij}u_j^{n+1} - \sum_{j>i} a_{ij}u_j^n \right\}$

The SOR Method

It is constructed by averaging the Gauss-Seidel iterate with a previous iterate:

$\tilde{u}_i^{N+1} = \frac{1}{a_{ii}} \left\{ f_i - \sum_{j<i} a_{ij}u_j^{n+1} - \sum_{j>i} a_{ij}u_j^n \right\}$
$\tilde{u}_i^{n+1} = \omega\tilde{u}_i^{n+1} + (1+\omega)u_i^n$, where ω is a overrelaxation parameter.

Finite Difference Approach to American Options

Partial Differntial Complementarity Problem(PDCP)

1. $v \geq f$

   - the option value must be above its immediate exercise value.

2. $\frac{\partial v}{\partial t} + rs\frac{\partial v}{\partial s} + \frac{1}{2}\sigma^2 s^2 \frac{\partial^2 v}{\partial s^2} \leq rv$

   - if the growth rate of the option value is lower than the mma, exercise.

3. $(\frac{\partial v}{\partial t} + rs\frac{\partial v}{\partial s} + \frac{1}{2}\sigma^2 s^2 \frac{\partial^2 v}{\partial s^2} - rv)(v - f) = 0$

   - if v=f, that means early exercise

   - or if $\frac{\partial v}{\partial t} + rs\frac{\partial v}{\partial s} + \frac{1}{2}\sigma^2 s^2 \frac{\partial^2 v}{\partial s^2} - rv$, it means that Black-Scholes PDE is satisfied, in such condition, it is like a European option.

4. v(T,S)=f(S)-payoff

The Linear Complementarity Problem

Given a matrix A and vectors $\underline{b}$and $\underline{c}$, the Linear Complementarity Problem(PCM) makes $\underline{x}$ satisfy the following:

$A\underline{x} \geq \underline{b}$,$\underline{x} \geq \underline{c}$,$(\underline{x} - \underline{c})(A\underline{x} - b) = 0$

Then we use PDCP to fit this form.

Define differential operator

$L = rs\frac{\partial}{\partial s} + \frac{1}{2}\sigma^2 s^2 \frac{\partial^2}{\partial s^2} - r$

So for an option value u(s,t) at time-t, it satisfies the following PDCP:

1. $u(s,t) \geq F(s,t)$

2. $\frac{\partial u}{\partial t} - Lu \geq 0$

3. $(\frac{\partial u}{\partial t} - Lu)(u - F) = 0$,$0 \leq t \leq T, 0 \leq S \leq \infty$

4. $u(S,0) = F(S,0), 0 \leq S \leq \infty$

Then use Crank-Nicolson scheme to approximate Lu, we get

$Lu \approx \frac{1}{2}(A\underline{u}^{n+1} + A\underline{u}^n) + \frac{1}{2}(\underline{f}^{n+1} + \underline{f}^n)$

$\Rightarrow M\underline{u}^{n+1} = \underline{b}$, where $M = I - \frac{1}{2}\Delta t A$,$\underline{b} == (I + \frac{\Delta t}{2}A)\underline{u}^n + \frac{\Delta t}{2}(\underline{f}^n + \underline{f}^{n+1})$

Let $\underline{F}$ be a discrete approximate to the exercise value F, apply to above PDCP, then we get:

1. $\underline{u}^{n+1} \geq \underline{F}$
2. $M\underline{u}^{n+1} \geq \underline{b}$
3. $(M\underline{u}^{n+1} - \underline{b})^T(\underline{u}^{n+1} - \underline{F}) = 0$
4. $\underline{u}^0 = \underline{F}$

To get an equivalent and more compact version, we make the following substitutions:

$\underline{Z} = \underline{u} - \underline{F}$,$q = M\underline{F} - \underline{b}$

$\underline{z} \geq 0 \Longleftrightarrow \underline{u} - \underline{F}$, (1) holds.

$q + M\underline{z} = M\underline{F} - \underline{b} + M(\underline{u} - \underline{F}) = M\underline{u} - \underline{b} \geq 0$ (2)holds

$\underline{z}^T(q + M\underline{z}) = (\underline{u} - \underline{F})^T(M\underline{u} - \underline{b}) = 0$,(3) holds

Part III

Monte Carlo Methods

0.7 Foundations

Monte Carlo Methods are based on the analogy between probability and volume. By drawing samples randomly from a universe of possible outcomes, use these samples as an estimate of the set's volume. The law of large numbers ensures this estimate converges to the currect value as the number of draws goes to very large.

Since it needs probability background, recall some notions of probability

Probability

Sample Space Ω (Ω, F, P)
Define A the set of outcomes in Ω that leads to this event occuring. A $\in F$, P(A) is the probability of the event occuring.
However Monte Carlo Method is a different approach to above calculation, its properties are as following:

1. Randomly sample $\omega \in \Omega$ many times
2. For each sample ω, determine whether or not event occurs
3. P(A) is approximated by the fraction of outcomes
4. The laws of large numbers ensure that this estimate converges to P(A) as the number of draws goes to ∞
5. The central limit theorey gives us information about the error of our approximation

Assume $x_1, x_2, x_3, ..., x_n$ is a sequence of independent identically distributed (iid) random variables
E(x_i)=$\mu < \infty$ i=1,2,3...
Denote $\bar{x}_n$=$\frac{1}{n}\sum_{i=1}^{n} x_i$

Thm Strong Law of Large Numbers

P($\lim_{n\to\infty} \bar{x}_n = \mu$)=1

we can simplely write it as $\bar{x}_n \overset{a.s}{\to}$ as $n \to \infty$

The strong law can imply the weak law, which means that the events for which $\bar{x}_n$ does not converge to μ have probability zero.

$P(\omega \in \Omega : \lim_{n\to\infty} \bar{x}_n(\omega) = \mu) = 1$

Thm Central Limit Theorem

If var(x_i)=$\sigma^2 < \infty$, then for all a $\leq$ b, $\lim_{n\to\infty} P(a \leq \frac{\bar{x}_n - \mu}{\frac{\sigma}{\sqrt{n}}} \leq b) = \Phi(b) - \Phi(a)$, where Φ is the CDF of the standard normal distribution.

$\Phi(y) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{y} e^{-\frac{x^2}{2}} dx$

Then standardize $\bar{x}_n$ to get a Z_n,$Z_n = \frac{\bar{x}_n - \mu}{\frac{\sigma}{\sqrt{n}}}$, close to a N(0,1) random variable. For large n, $\bar{x}_n$ has a distribution that is approximate N(μ, $\frac{\sigma^2}{n}$)

Two Monte Carlo Examples

The first one is to calculate $\alpha = \int_0^1 f(x)dx$

Thinking of α as an expectation E(f(U)), where U is uniformly distributed on[0,1], sample $U_1, U_2, ...$ independently and uniformly from [0,1]. Evaluating the function f at n random points and averaging the results produces the Monte Carlo estimate

$\hat{\alpha} = \frac{1}{n} \sum_{i=1}^{n} f(U_i)$

If f is integrable on [0,1], then by the strong law of large numbers, we get $\hat{\alpha}_n = \alpha$ with probability 1 as $n \to \infty$

P($\lim_{n\to\infty} \hat{\alpha}_n = \alpha$)=1

If f is square integrable on [0,1], set

$\sigma_f^2 = var(f(U)) = E[(f(U) - E(f(U)))^2] = E[(f(U) - \alpha)^2] = \int_0^1 [f(x) - \alpha]^2 dx$

Analysis of the error of our approximation $\hat{\alpha}_n - \alpha$ by CLT, the distribution of $\hat{\alpha}_n$ is N($\alpha, \frac{\sigma_f^2}{n}$)

Error estimate

Since we know $S_f = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}[f(U_i) - \hat{\alpha}_n]^2}$ and $\frac{\sigma_t}{\sqrt{n}} \approx \frac{S_f}{\sqrt{n}}$
we can use $\frac{S_f}{\sqrt{n}}$ as an error estimate. As we mentioned, Monto Carlo Method is not a competitive method for one-dimensional integrals, the accuray of the error is $O(n^{-\frac{1}{2}})$, in contrast, the error in the simple trapezoidal rule
$\alpha \approx \frac{f(0)+f(1)}{2n} + \frac{1}{n}\sum_{i=1}^{n-1} f(\frac{i}{n})$ is $O(n^{-2})$

Multiple integrals

$\int_{[0,1]^d} f(\underline{x})d\underline{x}, \underline{x} \in R^d$ based on n draws from $[0,1]^d$, we get error estimates: Monte Carlo-$O(\frac{1}{\sqrt{n}})$, Trapezoidal Rule-$O(\frac{1}{n^{\frac{2}{d}}})$
So once $n > 4$, Monte Carle Method is better.

The second one is Pricing Options.

European Call Option with strike K, underlying S_t, maturity T, interest rate r.

The underlying asset price must follow Black-Schole formula, so its SDE is:

$\frac{dS_t}{S_t} = rdt + \sigma dW_t \Rightarrow$

$$S_t = S_0 e^{(r-\frac{1}{2}\sigma^2)t+\sigma W_t} \tag{18}$$

$$S_T = S_0 e^{(r-\frac{1}{2}\sigma^2)T+\sigma W_T} \tag{19}$$

where $W_t \sim N(0,1)$.

If $Z \sim N(0,1)$, then $\sqrt{T}Z \sim N(0,T)$
(19) can be rewrited as $S_T = S_0 e^{(r-\frac{1}{2}\sigma^2)T+\sigma\sqrt{T}Z}$

The logarithm of the stock price is normally distributed, and the stock price itself has a lognormal distribution.

$\text{Log}(S_T) \sim N((r - \frac{1}{2}\sigma^2)T, \sigma^2 T)$

The expectation $E[e^{-rT}(S(T) - K^+)]$ is an integral with respect to the lognormal density of S(T), this integral can be evaluated as

BS(S(0), σ,T,r,K)=$S\Phi(\frac{\log(\frac{S}{K})+(r+\frac{1}{2}\sigma^2)T}{\sigma\sqrt{T}}) - e^{-rT}K\Phi(\frac{\log(\frac{S}{K})+(r-\frac{1}{2}\sigma^2)T}{\sigma\sqrt{T}})$

This is a Black-Scholes formula for a call option.

Monte Carlo Pricing

$V_0 = \hat{E}[e^{-rt}(S_T - K)^+]$

We can estimate $E[e^{-rt}(S_T - K)^+]$ using the following algorithm:

for i=1,2,3,...,n

generate Z_i

set $S_i(T) = S(0)\exp([r - \frac{1}{2}\sigma^2]T + \sigma\sqrt{T}Z_i)$

set $C_i = e^{-rT}(S(T) - K)^+$

Set $\hat{C}_n = \frac{C_1+...+C_n}{n}$

If n$\geq$ 1,

E($\hat{C}_n$)=E($\frac{1}{n}\sum_{i=1}^{n} C_i) = \frac{1}{n}\sum_{i=1}^{n} E(C_i) = \frac{1}{n}(nE(C_i)) = C_0 = E(e^{-rT}(S_T - K)^+)$

By the Strong Law of Large Numbers, P($\lim_{n\to\infty} \hat{C} = C_0$)=1

Error estimate

Let $S_C = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(C_i - \hat{C}_n)^2}$ denote the sample standard deviation of $C_1, C_2..., C_n$ and let z_δ denote the $1 - \delta$ quantile of the standard normal distribution. Then $\hat{C}_n = \pm z_{\delta/2}\frac{s_C}{\sqrt{n}}$ is an asymptotically (as n$\rightarrow \infty$) valid 1-δ confidence interval for C.

0.8 Random Number Generation

The core of Monte Carlo simulation is to generate uniformly distributed random variables.

Pseudo random number generator

Generator a sequence of random variables U_1, U_2 with two properties:

1. U_i is uniformly distributed on [0,1]

2. the U_i are mutually independent.

The Linear Congruential Generator

The general linear congruential generator takes the form:

$$x_{i+1} = (ax_i + c) \mod m, u_{i+1} = x_{i+1}/m \tag{20}$$

, where a, m, c $\in Z$

a is called the multiplier, m is the modulus. the initial seed x_0 is required

$1 \leq x_0 \leq m - 1$.

If c$\neq$ 0, it is called mixed, if c=0, it is called pure.

Some examples:

8 mod 5 =3, 2 mod 7=2

use (20) with a=6, m=11,$x_0 = 1$

$x_1 = ax_0$ mod 11=6, x_2=a$x_1$mod 11=3, $x_3 = 7$, $x_4 = 9$, $x_5 = 10$, $x_6 = 5$, $x_7 = 8$,

$x_8 = 4$, $x_9 = 2$, $x_{10} = 1$, $x_{10} = x_0$, the sequence repeats.

Full Period

Definition: A linear congruential generator that produces all m-1 distinct values is said to have a full period.

In general, we choose m large, and a needs to be chosen carefully.

Issues for random number generator

1. Period length-any random generator will eventuall repeat itself, and the longer period is much better, since it contains more distinct values before repeating. The longest period is m-1(m is the modulus). For a full period linear congruential generator, the gap between two variables is 1/m, so the larger m is the more accuracy values that approximate a uniform distribution.

2. Reproducibility-the linear congruential generator can reproduce the same random sequence by using the same seed x_0

3. Speed-since a generator can be used many times in a single simulation, it must be fast.

4. Portability-An algorithm for generating random numbers should produce the same sequence of values on all platforms.

5. Randomness-two broad aspects to constrain generators : theoretical properties and statistical tests.

Thm: suppose $c \neq 0$, for any seed x_0, (20) generates m-1 distinct values if

1. c and m are relaively prime

2. every prime number that divides m also divides a-1

3. a-1 is divisible by 4 if m is

As a simple consequence, if m is power of 2, the generator has full period if c is odd and a=4n+1 for some integer n.

If c=0 and m is prime, full period is achieved from any $x_0 \neq 0$ if

1. $a^{m-1} - 1$ is a multiple of m
2. $a^j - 1$ is not a multiple of m for j=1,2...,m-2.

A number a satisfying above 2 properties is called a primitive root of m.

General Sampling Methods

Assume the availability of a sequence $U_1, U_2, ...$ of independent random variables, each satisfying:

$$P(U_i \leq u) = \begin{pmatrix} 0, u < 0 \\ u, 0 \leq u \leq 1 \\ 1, u > 1 \end{pmatrix}$$

i.e each uniformly distributed between 0 and 1.

So We want to find a algorithm to transform these random variables into paths of stochastic processes.

The Inverse Transform Method

Set random variable X with cumulative distribution function(CDF) :

$F(x) = P(X \leq x), \forall x \in R$, if f(x) is the density function of x, then

$F(x) = \int_{-\infty}^{x} f(y)dy$.

If F is strictly increasing, the inverse transform method sets $X = F^{-1}(U), U \sim$ Unif[0,1], where F^{-1} is the inverse of F and Unif[0,1] denotes the unform distribution on[0,1].

Otherwise, we need a rule to break ties. For example, we may set $F^{-1}(u) = \inf x : F \geq u$ for the points that have more than one values.

Verification:

To make sure that the inverse transform generates samples from F, we check the distribution of the X it produces:

$P(X \leq x) = P(F^{-1}(U) \leq x) = P(U \leq F(x)) = F(x)$

Some examples:

Exponential Distribution:

The exponential distribution with mean θ has distribution: $F(x) = 1 - e^{\frac{-x}{\theta}}, x \geq 0$

Invert F(x)

$\Rightarrow U = 1 - e^{\frac{-x}{\theta}} \Rightarrow \frac{-x}{\theta} = \ln(1-U) \Rightarrow x = -\theta \ln(1-U) \Rightarrow x = -\theta \ln(U)$, because U and 1-U have the same distribution.

Arcsine law:

For t$\in [0,1]$, the time at which a standard brownian motion attains its maximum over the time interval [0,1] has distribution: $F(x) = \frac{2}{\pi} \arcsin(\sqrt{x}), 0 \leq x \leq 1$

Invert $U = \frac{2}{\pi} \arcsin(\sqrt{x}) \rightarrow \sin(\frac{\pi U}{2}) = \sqrt{x} \rightarrow \sin^2(\frac{\pi U}{2}) = x$

Using $2\sin^2(t) = 1 - \cos(2t)$ for $0 \leq t \leq \frac{\pi}{2}$, we get $X = \frac{1}{2} - \frac{1}{2}\cos(U\pi), U \sim$ Unif[0,1]

0.9 Acceptance-Rejection Method

This method is one of the most widely used applicable mechanisms for generating random samples. First, it generates samples from a convenient distribution, then reject a random subset. The reject mechanism is designed so that the accepted samples are distributed according to the target distribution.

Suppose we have a density function f defined on set $\chi \in R^d$, let g be a density on χ from which we can generate samples such that $f(x) \leq cg(x)$, for some constant c$\geq$ 1,$\forall x \in \chi$

In acceptance-rejection method, we generate a sample X from g, accept the sample with probability $\frac{f(x)}{cg(x)} \leq 1$;this can be implemented by sampling U uniformly over (0,1) and accept X if U$\leq \frac{f(X)}{cg(X)}$, if X is rejected, sample X from g again, and repeat.

Verification: suppose Y is returned by our algorithm then Y has the distribution of X conditional on $U \leq \frac{f(X)}{cg(X)}$, then for any A $\in \chi$

$$P(Y \in A) = P(X \in A|U \leq \frac{f(X)}{cg(X)}) = \frac{P(X \in A, U \leq \frac{f(X)}{cg(X)})}{P(U \leq \frac{f(X)}{cg(X)})} \tag{21}$$

Given X, $P(U \leq \frac{f(X)}{cg(X)}) = \frac{f(X)}{cg(X)}$ because U$\sim$Unif[0,1]

For the random variable X, $P(U \leq \frac{f(X)}{cg(X)}) = E(\frac{f(X)}{cg(X)}) = \int \frac{f(x)}{cg(x)} g(x)dx = \frac{1}{c}$

Plug into (21), we get

$P(Y \in A) = cP(X \in A, U \leq \frac{f(X)}{cg(X}) = c\int_A \frac{f(x)}{cg(x)} g(x)dx = \int_A f(x)dx$

So this verifies that Y has density f.

Example:

Normal from Double Exponential

Standard Normal Density: f(x)=$\frac{1}{\sqrt{2\pi}} e^{-\frac{x^2}{2}}$

Double-normal exponential on $(-\infty, \infty)$ density: $g(x) = \frac{1}{2}e^{-|x|}$

Ratio $\frac{f(x)}{g(x)} = \frac{\frac{1}{\sqrt{2\pi}} e^{-\frac{x^2}{2}}}{\frac{1}{2}e^{-|x|}} = \sqrt{\frac{2}{\pi}} e^{-\frac{x^2}{2}+|x|} \leq \sqrt{\frac{2e}{\pi}} \approx 1.3155 \equiv c$

To sample a double exponential draw on standard exponential X=-$\theta \ln(U)$, where $U \sim$ Unif[0,1]

Then randomize the sign.

Rejection test: $U \geq \frac{f(x)}{cg(x)} = \frac{\sqrt{\frac{2}{\pi}} e^{-\frac{1}{2}x^2+|x|}}{\sqrt{\frac{2e}{\pi}}}$=$e^{-\frac{1}{2}(|x|-1)^2}$

The combined steps are as follows:

1. generate U_1, U_2, U_3 from Unif[0,1]

2. X$\leftarrow$ - $\log(U_1)$

3. if $U_2 > \exp(-0.5(X-1)^2)$

   go to step 1

4. if $U_3 \leq 0.5$

   X$\leftarrow$-X

5. return X

Normal Random Variables and Vectors

Basic Properties:

If $X \sim N(\mu, \sigma^2)$, then it has density $\Phi(x) = \Phi_{\mu,\sigma}(x) = \frac{1}{\sqrt{2\pi\sigma}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$

If Z$\sim N(0,1)$, then $\mu + \sigma Z \sim N(\mu, \sigma^2)$

So if we want to generate normal random variables, we can only generate standard normal random variables.

A d-dimensional normal distribution is characterzed by $\underline{\mu} \in R^d$ and $\Sigma \in R^{d\times d}$

Properties of Σ

1. Σ is symmetric, that is $\Sigma = \Sigma^T$

2. Σ is positive, semidefinite

Definition: A matrix $\Sigma \in R^{d\times d}$ is positive definite if $\underline{X}^T \Sigma \underline{X} > 0, \forall \underline{X} \in R^d$ with $\underline{X} \neq 0$.

A matrix Σ is positive semi-definite if $\underline{X}^T \Sigma \underline{X} \geq 0, \forall \underline{X} \in R^d$

If Σ is positive definite, then N($\underline{\mu}, \Sigma$) has density function:

$\Phi_{\underline{\mu},\Sigma}(\underline{x}) = \frac{1}{(2\pi)^{\frac{d}{2}} |\Sigma|^{\frac{1}{2}}} e^{-\frac{1}{2}(\underline{x}-\underline{\mu})^T \Sigma (\underline{x}-\underline{\mu})}$ for $\underline{x} \in R^d$

If $\underline{x} \sim N(\underline{\mu}, \Sigma)$, then its ith component x_i has density: $x_i \sim N(\mu_i, \sigma_{ii}^2)$, where $\sigma_{ii}^2 = \Sigma_{ii}$, further, cov$(x_i, x_j)$=E$[(x_i - \mu_i)(x_j - \mu_j)] = \Sigma_{ij}$

The correlation bewteen x_i and x_j is : $\rho_{ij} = \frac{\Sigma_{ij}}{\sigma_i \sigma_j}$

Generating univariate normals:

Now we discuss algorithms for generating samples from univariate normal distributions. Assume that we have a sequence of independent uniform on[0,1] $U_1, U_2, ...$

Box-Muller Method

First, generate a sample from the bivariate standard normal, so each component is a standard normal. This algorithm is based on the following two properties of the bivariate normal: if $Z \sim N(0, I_2)$,

1. R=$Z_1^2 + Z_2^2$ is exponentially distributed with mean 2: $P(R \leq x) = 1 - e^{-\frac{x}{2}}$;

2. given R, the point(Z_1, Z_2) is uniformly distributed on the circle of radius $\sqrt{R}$,centered at $\underline{0}$

To generate (Z_1, Z_2)

1. generate R

2. choose a point uniformly, from the circle of radius $\sqrt{R}$

Generate R

R=$-2\ln(U_1), U_1 \sim$ Unif[0,1]

Generate the point

generate a random angle uniformly between 0 and 2π, V=$2\pi U_2$, $U_2 \sim$ Unif[0,1]

point on circle:$(\sqrt{R}\cos V, \sqrt{R}\sin V)$

Algorithm:

Generate $U_1, U_2 \sim$ Unif[0,1] independently, R=$-2\ln(U_1)$, V=$2\pi U_2$

$Z_1 = \sqrt{R}\cos V, Z_2 = \sqrt{R}\sin V$

Return Z_1, Z_2

0.10 Multivariate Normals

Basic properties:

$\underline{Z} \sim N(\underline{\mu}, \Sigma)$,where

$$\Sigma = \begin{bmatrix} \sigma_{11}\ \sigma_{12}\ \dots\ \sigma_{1d} \\ \sigma_{21}\ \sigma_{22}\ \dots\ \sigma_{2d} \\ \dots \\ \dots \\ \dots \\ \sigma_{d1}\ \sigma_{d2}\ \dots\ \sigma_{dd} \end{bmatrix}$$

Using the correlations $\rho_{ij} = \frac{\sigma_{ij}}{\sigma_i \sigma_j}$

$$\Sigma = \begin{bmatrix} \sigma_1\ 0\ 0\ \dots\ 0 \\ 0\ \sigma_2\ 0\ \dots\ 0 \\ \dots \\ \dots \\ \dots \\ 0\ 0\ \dots\ 0\ \sigma_d \end{bmatrix} \begin{bmatrix} \rho_{11}\ \rho_{12}\ \cdots\ \rho_{1d} \\ \rho_{21}\ \rho_{22}\ \cdots\ \rho_{2d} \\ \dots \\ \dots \\ \rho_{d1}\ \rho_{d2}\ \cdots\ \rho_{dd} \end{bmatrix} \begin{bmatrix} \sigma_1\ 0\ 0\ \dots\ 0 \\ 0\ \sigma_2\ 0\ \dots\ 0 \\ \dots \\ \dots \\ \dots \\ 0\ 0\ \dots\ 0\ \sigma_d \end{bmatrix}$$

If Σ is positive semidefinite, but not positive semidefinite, $\exists \underline{X} \neq \underline{0}$, such that

$\underline{X}^T \Sigma \underline{X} = 0$

-Σ is sigular

-there is no normal density with covariance matrix Σ

-if $\underline{Z} \sim N(0,1)$, $\underline{X} = \underline{\mu} + A\underline{Z} \sim N(\underline{\mu}, \Sigma)$

Thm:Linear Transformation Property

Any linear transformation of a normal vector is normal. If $\underline{X} \sim N(\underline{\mu}, \Sigma)$, then A$\underline{X} \sim N(A\underline{\mu}, A\Sigma A^T)$ for any $\underline{\mu} \in R^d$,$\Sigma \in R^{d\times d}$ and A$\in R^{k\times d}$

Generate multivariate normals

First, generate independent $Z_1, Z_2, ..., Z_d \sim N(0,1)$ and put them in a vector $\underline{Z} \sim N(\underline{0}, I_d)$, then $A\underline{Z} \sim N(\underline{0}, AA^T)$

Then, the problem of sampling $\underline{X}$ from N($\underline{\mu}, \Sigma$) reduces to finding a matrix A for which $AA^T = \Sigma$

Thm Cholesly factorization

Suppose $\Sigma \in R^{d\times d}$ is positive definite, then $\exists$ a lower trangular matrix $A \in R^{d\times d}$ such that $\Sigma = AA^T$, and A is unique up to changes in sign.

Consider the component of $\underline{X} = \underline{\mu} + A\underline{Z}$

$$\begin{bmatrix} x_1 \\ x_2 \\ . \\ . \\ . \\ x_d \end{bmatrix} = \begin{bmatrix} \mu_1 \\ \mu_2 \\ . \\ . \\ . \\ \mu_d \end{bmatrix} + \begin{bmatrix} A_{11}\ A_{12}\ ...\ A_{1d} \\ A_{21}\ A_{22}\ ...\ A_{2d} \\ ... \\ ... \\ ... \\ A_{d1}\ A_{d2}\ ...\ A_{dd} \end{bmatrix} \begin{bmatrix} Z_1 \\ Z_2 \\ . \\ . \\ . \\ Z_d \end{bmatrix}$$

Example: 2×2 case

Suppose $\Sigma = \begin{bmatrix} \sigma_1^2 & \rho_{12}\sigma_1\sigma_2 \\ \rho_{21}\sigma_2\sigma_1 & \sigma_2^2 \end{bmatrix}$, we want to $\Sigma = AA^T = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} \begin{bmatrix} A_{11} & A_{21} \\ A_{12} & A_{22} \end{bmatrix}$

$$\Rightarrow \begin{bmatrix} \Sigma_{11} & \Sigma_{12} \\ \Sigma_{21} & \Sigma_{22} \end{bmatrix} = \begin{bmatrix} A_{11} & 0 \\ A_{21} & A_{22} \end{bmatrix} \begin{bmatrix} A_{11} & A_{12} \\ 0 & A_{22} \end{bmatrix} = \begin{bmatrix} A_{11}^2 & A_{21}A_{11} \\ A_{21}A_{11} & A_{21}^2 + A_{22}^2 \end{bmatrix}$$

$\Rightarrow A_{11} = \sqrt{\Sigma_{11}}, A_{21} = \frac{\Sigma_{12}}{A_{11}} = \frac{\Sigma_{12}}{\sqrt{\Sigma_{11}}}, A_{22} = \sqrt{\Sigma_{22} - A_{21}^2}$

$\Sigma_{11} = \sigma_1^2, \Sigma_{22} = \sigma_2^2, \Sigma_{12} = \sigma_1\sigma_2\rho$

$A_{11} = \sigma_1, A_{21} = \sigma_2\rho, A_{22} = \sigma_2\sqrt{1-\rho^2}$

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

General Case $\Sigma \in R^{d\times d}$

$\Sigma = AA^T$

$$\Sigma = \begin{bmatrix} A_{11} \ 0 \ 0 \ ... \ 0 \\ A_{21} \ A_{22} \ 0 \ ... \ 0 \\ ... \\ ... \\ ... \\ A_{d1} \ A_{d2} \ A_{d3} \ ... \ A_{dd} \end{bmatrix} \begin{bmatrix} A_{11} \ A_{21} \ A_{31} \ ... \ A_{d1} \\ 0 \ A_{22} \ A_{32} \ ... \ A_{d2} \\ ... \\ ... \\ ... \\ 0 \ 0 \ 0 \ ... \ A_{dd} \end{bmatrix}$$

over row 1 of Σ

$\Sigma_{11} = A_{11}^2, \Sigma_{12} = A_{11}A_{21}, \Sigma_{13} = A_{11}A_{31}, ...\Sigma_{1d} = A_{11}A_{d1}$

over row 2 of Σ

$\Sigma_{21} = A_{21}A_{11}, \Sigma_{22} = A_{21}^2 + A_{22}^2, \Sigma_{23} = A_{21}A_{31} + A_{22}A_{32}, ..., \Sigma_{2d} = A_{21}A_{d1} + A_{22}A_{d2}$

...

...

over row d of Σ

$\Sigma_{d1} = A_{11}A_{d1}, \Sigma_{d2} = A_{21}A_{d1} + A_{22}A_{d2}, ..., \Sigma_{dd} = A_{d1}^2 + A_{d2}^2 + ... + A_{dd}^2$

General solution of Σ_{ij} is $\Sigma_{ij} = \sum_{k=1}^{j} A_{ik}A_{jk}, j \leq i$

we get,$A_{ij} = \frac{(\Sigma_{ij} - \sum_{k=1}^{j-1} A_{ik}A_{jk})}{A_{jj}}, j < i$ and $A_{ii} = \sqrt{\Sigma_{ii} - \sum_{k=1}^{i-1} A_{ik}^2}$

Algorithm:

Input: Symmetric positive definite matrix d$\times d$ matrix Σ

Output: Lower triangular A with $AA^T = \Sigma$

$A \leftarrow 0(d \times d$ zero matrix)

for j=1,2,...,d

for i=j,...,d

$v_i \leftarrow \Sigma_{ij}$

for k=1,2,...,j-1

$v_i \leftarrow v_i - A_{jk}A_{ik}$

$A_{ij} \leftarrow \frac{v_i}{\sqrt{v_j}}$

return A

The Semidefinite Case

If Σ is positive semidefinite but not positive definite, then Σ is singular and if $AA^T = \Sigma$, then A is singular

Suppose A is lower triangular since its rank deficient, some diagonal element $A_{jj} = 0$, so the Cholesky algorithm fails because of a division by 0.

In sitution of $A_{jj} = 0$, we set column j of A to $\underline{0}$, and change the algorithm slightly:

Given Σ symmetric positive semidefinite but not posiive definite, same as previous case, but we replace $A_{ij} \leftarrow \frac{v_i}{\sqrt{v_j}}$ with :

If $v_j > 0$, then $A_{ij} \leftarrow \frac{v_i}{\sqrt{v_j}}$

Thus if $v_j = 0$, the entry A_{ij} is left at its intital value of zero.

However, there are two problems in practice.

1. It may lead to a round-off error

2. Reduction: $\underline{X} \sim N(\underline{0}, \Sigma)$, suppose rank($\Sigma$)=k<d, the components of $\underline{X} \in R^d$ can be represented as a linear combination of k components, situation arises if d variables are generated using k<d sources of uncertainty .

0.11 Money Market Account

In Money Market Account(MMA), if we invest \$1 at time t=0, then it has the value $\beta(t) = e^{rt}$ at time t.

Suppose stock pays no dividends, $\frac{dS_t}{S_t} = \mu dt + \sigma dW_t$

In no arbitrage condition, under the risk-neutral measure, $\mu = r$, and all assets have the same average rate of return.

Further, under the risk-neutral measure, $\frac{S_t}{\beta(t)}$ is a martingale,

$\frac{S_u}{\beta(u)} = E[\frac{S_t}{\beta(t)}|S_\tau : 0 \leq \tau \leq u]$

Path Dependent Payoff

path = GBM(μ, σ, n)

Asian Option with Discrete Monitoring

For call option: Payoff= $(\overline{S} - K)^+$, for put option: Payoff= $(K - \overline{S})^+$, where K is the strike price and $\overline{S} = \frac{1}{n}\sum_{i=1}^{n} S(t_i)$ is the average stock price over different time monitoring dates.

Asian Option with Continuous Monitoring

The only difference is $\overline{S} = \frac{1}{t-u}\int_u^t S(\tau)d\tau$, compared with Asian option with discrete monitoring.

Barrier Option

It is a Down-and-Out call option with barrier b, strike K, and expiry T, which means that spot price starts above the barrier level and has to move down for the option to become null and void

Payoff: $1_{\tau(b)>T}$, where $\tau(b) = \inf\{t_i : S_{t_i<b}\}$ is the first time in $t_1, t_2, ..., t_n$ that underlying price drops below b.

Look back option

The Lookback options are a type of exotic options with path dependency. The payoff depends on the optimal (maximum or minimum) underlying asset's price occurring over time slots.

Payoff for put option: $\max_{i=1,2,...,n}\{S_{t_i} - S_{t_n}\}$

Payoff for call option: $S_{t_n} - \min_{i=1,2,...,n}\{S_{t_i}\}$

-gain profit from buying the underlying at the lowest price over $t_1, .., t_n$ and selling at the final price.

Incorporate a Term Structure of Interest Rates

If we have a constant interset rate r, the time-t price of a zero-coupon bond paying \$ 1 at time $T > t$ is $B(t, T) = e^{r(T-t)}$

However in real world, r is not constant, so we determine the term structure of interest rates using a collection of bond prices $B(0, T) : i = 1, 2, ..., n$

Then we define the time-varying interest rate r(u) by $r(u) = \frac{-\partial}{\partial T}[B(0, T)]|_{T=u}$

solve for B(0,T) and we get $B(0, T) = e^{-\int_0^T r(u)du}$

Under risk-neutral measure, the only nomics of an asset price are

$\frac{dS_t}{S_t} = r(t)dt + \sigma dW_t$ with solution $S_t = S_0 e^{\int_0^t r(u)du - \frac{1}{2}\sigma^2 t + \sigma W_t}$.

We can simulate this over $0 = t_0 < t_1 < ... < t_n$ using

$S_{t_{i+1}} = S_{t_i} e^{\int_{t_i}^{t_{i+1}} r(u)du - \frac{1}{2}\sigma^2(t_{i+1}-t_i) + \sigma\sqrt{t_{i+1}-t_i}Z_{i+1}}$, where $Z_1, ..., Z_n$ are independent and have standard normal distribution

Suppose we observe bond prices $B(0, t_1), B(0, t_2), ..., B(0, t_n)$,

$$\begin{aligned}\frac{B(0,t_i)}{B(0,t_{i+1})} &= \frac{e^{-\int_0^{t_i} r(u)du}}{e^{-\int_0^{t_{i+1}} r(u)du}}\\ &= e^{\int_0^{t_i} r(u)du + \int_0^{t_{i+1}} r(u)du}\\ &= e^{\int_{t_i}^{t_{i+1}} r(u)du}\end{aligned}$$

Simulation:

$S_{t_{i+1}} = S_{t_i} \frac{B(0,t_i)}{B(0,t_{i+1})} e^{-\frac{1}{2}\sigma^2(t_{i+1}-t_i)+\sigma\sqrt{t_{i+1}-t_i}Z_{i+1}}, i = 0, 1, ..., n-1$

Asset with Dividends

Suppose we hold a single share of an asset which is no longer self-financing, then our strategy must deal with the dividends.

In this case, neither withdraw nor deposits are allowed and number of shares changes over time.

First, we construct the model:

S_t: underlying asset price, $\widetilde{S}_t$: asset price with dividends reinvested

$\frac{dS_t}{S_t} = \mu dt + \sigma dW_t$

$\frac{d\widetilde{S}_t}{\widetilde{S}_t} = \frac{dS_t + dD_t}{S_t}$, where dD_t is the dividend payment over dt, return an origiral investment with dividends and captital gains reinvested.

In such case, $\frac{\widetilde{S}}{\beta(t)}$ is a martingale under the risk-neutral measure instead of $\frac{S_t}{\beta(t)}$

Suppose an asset pays a continuous dividend yield at a rate δ, then $dD_t = \delta S_t dt$

So $\frac{\widetilde{S}_t}{\widetilde{S}_t} = \frac{dS_t + \delta S_t dt}{S_t}$

$\quad = \frac{S_t}{S_t} + \delta dt$

$\quad = (\mu dt + \sigma dW_t) + \delta dt$

$\quad = (\mu + \delta)dt + \sigma dW_t$

Since it is a no arbitarge case, $\mu + \delta = r \rightarrow \mu = r - \delta$

$\frac{dS_t}{S_t} = \mu dt + \sigma dW_t \rightarrow \frac{dS_t}{S_t} = (r - \delta)dt + \sigma dW_t$

-Risk-neutral dynamics of an asset price with continuous dividend yield δ

Solving above equation, we get:

$S_t = S_0 e^{(r-\delta-\frac{1}{2}\sigma^2)t+\sigma W_t}$

Comparing with the original formula, $r - \delta - \frac{1}{2}\sigma^2$ takes the place of r, so the dividend yield reduced the growth rate of the underlying.

Applications:

1. Equity Indices

   we often construct an index as a Geometric Brownian Motion, the index itself does not pay dividends, but the stocks that make up the index might. Wide range of dividends on different dates can be simulated by a continuous dividend yield.

2. Exchange Rates

   Exchange rate=$\frac{\#units\ of\ domestic\ currency}{1\ unit\ of\ foreign\ currency}$

   a unit of foreign currency earns interset at rate r_f, can be considered as a dididend stream.

3. Commodities Some physical commodities as gold , oil and etc have extra cost of storage, such cost can be considered as negative dividend yield (set $\delta < 0$)

0.12 Multi-Dimensions

We specify a multi-dimensional Geometric Brownian motion

$$\frac{dS_i(t)}{S_i(t)} = \mu_i dt + \sigma_i dX_i(t), i = 1, 2, ..., d \tag{22}$$

where $X_i(t)$ is a standard 1-dimension Brownian motion, and X_i and X_j have correlation ρ_{ij}

Let $\sum_{ij} = \sigma_i \sigma_j \rho_{ij}$

Define $\sum \in R^{d \times d}$

then $\underline{X}(t) = \begin{bmatrix} \sigma_1 X_1(t) \\ \sigma_2 X_2(t) \\ . \\ . \\ . \\ \sigma_d X_d(t) \end{bmatrix} \sim BM(\underline{0}, \sum)$

We denote $\underline{S} = \begin{bmatrix} S_1(t) \\ S_2(t) \\ . \\ . \\ . \\ S_d(t) \end{bmatrix}$ as GBM($\underline{\mu, \sum}$) with $\underline{\mu} = \begin{bmatrix} \mu_1 \\ \mu_2 \\ . \\ . \\ . \\ \mu_d \end{bmatrix}$

$\sum$ is the convariance matrix for $\underline{X}(t)$ and the actual drift vector for $\underline{S}$ is $\begin{bmatrix} \mu_1 S_1(t), \mu_2 S_2(t), ..., \mu_d S_d(t) \end{bmatrix}^T$

$S_i(t) = S_i(0)e^{(\mu_i - \frac{1}{2}\sigma^2)t + \sigma_i X_i(t)}, i = 1, 2, ..., d$

Recall that a vector $\sim BM(\underline{0}, \sigma)$ can be represented as $A\underline{W}(t)$, where $\underline{W}(t)$ is a standard Brownian motion with drift $\underline{0}$ and covariance I and A is any matrix such that $AA^T = \sum$

Apply this to $\underline{X}(t)$ to (22), we get

$\frac{dS_i(t)}{S_i(t)} = \mu_i dt + \underline{a}_i d\underline{W}(t)$, where $\underline{a}_i$ is the ith row of A.

Explicitly, $\frac{dS_i(t)}{S_i(t)} = \mu_i dt + \sum_{j=1}^{d} A_{ij} dW_j(t)$

Simulation:

$S_i(t) = S_i(0)e^{(\mu_i - \frac{1}{2}\sigma^2)t + \sum_{j=1}^{d} A_{ij} W_j(t)}$,where $\underline{\mu}$ and $\sum$ are the drift and covariance for $\underline{X}(t)$, the underlying Brownian motion.

At discrete monitoring $0 = t_0 < t_1 < ... < t_n$

$S_i(t_{k+1}) = S_i(t_k)e^{(\mu_1 - \frac{1}{2}\sigma^2)(t_{k+1}-t_k)+\sqrt{t_{k+1}-t_k}\sum_{j=1}^{d} A_{ij}Z_{k+1}}$, i=1,2,...,d, k=0,1,...,n-1,

where $\underline{Z}_k = \begin{bmatrix} Z_{k,1}, Z_{k,2}, ..., Z_{k,d} \end{bmatrix}^T$

If asset S_i has dividend yield δ_i, set $\mu_i = r - \delta_i$ for no arbitrage.

Applications:

1. spread option

   A call option on the spread between two assets S_1 and S_2 with strike K, expiry T.

   Payoff=$([S_1(T) - S_2(T)] - K)^+$.

   For example, crack spread options traded on the New York Mercantile Exchange are options on the spread between heating oil and crude oil futures.

2. Basket Option.

   A basket option is an option on a portfolio of underlying assets and has a payoff of $([c_1S_1(T) + c_2S_2(T) + ... + c_dS_d(T)] - K)^+$.

   Typical examples would be options on a portfolio of related assets - bank stocks or Asian currencies.

3. Outperformance option.

   A outperformance option is an option that the holder gains the best performance out of multiple assets and have payoff of the form

   :$(\max c_1S_1(T), c_2S_2(T), ..., c_dS_d(T) - K)^+$

4. Barrier options

   A two-asset barrier option may have such a payoff like the form:

   $1_{\min_{i=1,2,...,n}\{S_2\}(t_1)<b}(K - S_1(T))^+$

   This example is a down and in put option on S_1 that knocks in when S_2

drops below a barrier at b, where S_1 could be a stock and S_2 could be an index. If so, the put on the stock is knocked in only if the market drops.

5. Quantos

   Quantos are options that depends on both a stock price and an exchange rate. For example, an option to buy a stock denominated in a foreign currency with teh strike price fixed on the foreign currency, but payoff is to be made in the domestic currency.

   Let S_1 denote the stock price and S_2 the exchange rate $(\frac{units\ domestic\ currency}{1\ unit\ foreign\ currency})$

   The payoff in the domestic currency is : $S_2(T)(S_1(T) - K)^+$

   Another variation payoff is : $(S_1(T) - \frac{K}{S_2(T)})$, it corresponds to a quanto in which the level of the strike is fixed in the domestic currency and the payoff is made in the foreign currency.

0.13 Generating sample paths

This chapter introduces some methods for simulating paths of a variety of stochastic processes important in financial engineering.

In many applications, we need entire path of an asset price $\{S_t : 0 \leq t \leq T\}$

Brownian Motion

A standard one-dimensional Brownian motion on [0,T], we mean a stochastic process W(t), $0 \leq t \leq T$ with the following properties:

1. W_0=0;
2. the mapping t$\rightarrow W_t$ is, with probability 1, a continuous function on [0,T];
3. the increments $W_{t_1} - W_{t_0}, W_{t_2} - W_{t_1}, ..., W_{t_k} - W_{t_{k-1}}$ are independent for any k and any $0 \leq t_0 < t_1 < ... < t_k \leq T$;

4. $W_t - W_s \sim N(0, t-s)$ for any $0 \le s \le t \le T$.

For constants μ and $\sigma > 0$, we call a process X(t) a Brownian motion with drift μ and diffusion coefficient $\sigma^2 (X \sim BM(\mu, \sigma^2))$ if $\frac{X_t - \mu t}{\sigma}$ is a standard Brownian motion.

Given a standard Brownian motion W_t, we construct a Brownian motion $X \sim BM(\mu, \sigma^2)$ by setting $X_t = \mu t + \sigma W_t$. Further, X_t solves the SDE $dX_t = \mu dt + \sigma dW_t$, we can also define a Brownian motion with deterministic drift $\mu(t)$, and diffusion coefficient $\sigma(t)$ through $dx_t = \mu(t)dt + \sigma(t)dW_t$

Then we need stochastic integration to get the solution

$X_t = X_0 + \int_0^t \mu(s)ds + \int_0^t \sigma(s)ds$

In this case $(X_t - X_s) \sim N(\int_0^t \mu(s)ds, \int_0^t \sigma(s)ds)$

Random Walk Construction

Simulate Brownian motion at a fixed set of times $0 < t_1 < t_2 < ... < t_n$. Because Brownian motion has independent normally distributed increments, simulating the $W(t_i)$ from their increments is straightforward.

Suppose $Z_1, Z_2, ..., Z_n \sim N(0,1)$ and independent set $t_0 = 0$ and $W_0 = 0$, we generate a standard Brownian motion, using $W_{t_{i+1}} = W_{t_i} + \sqrt{t_{i+} - t_i} Z_{i+1}$ for i=0,1,2,...,n-1.

For Brownian motion with constant μ and σ and given X(0),set

$X_{t_{i+1}} = X_{t_i} + \int_{t_i}^{t_{i+1}} \mu(s)ds + \sqrt{\int_{t_i}^{t_{i+1}} \sigma^2(s)ds} Z_{i+1}$,i=0,1,2,...,n-1

To generate $X \sim BM(\mu, \sigma^2)$, given X_0, set

$X_{t_{i+1}} = X_{t_i} + \mu(t_{i+1} - t_i) + \sigma\sqrt{t_{i+1} - t_i} Z_{i+1}$

The methods are exact in the sence that the joint distribution of the simulated values $(W_{t_1}, ..., W_{t_n})$ coincides with the joint distribution of the corresponding

Brownian motion at $t_1, ..., t_n$.

Alternative Construction

The vector $(W_{t_1}, ..., W_{t_n})$ is a linear transformation of the vector of increments $(W_{t_1}, W_{t_2} - W_{t1}, ..., W_{t_n} - W_{tn-1})$, which are independent and normal.

So $[W_{t_1}, ..., W_{t_n}]$ is multivariate normal.

E(W_{t_i})=0, and

Cov(W_s, W_t)=Cov$(W_s, W_s + (W_t - W_s))$

=Cov(W_s, W_s)+Cov$(W_s, W_t - W_s)$

=Var(W_s)+0=S

Let C be the covariance matrix for $[W_{t_1}, ..., W_{t_n}]^T$, then $C_{ij} = \min(t_i, t_j)$, and this vecter has mean $\underline{0}$

Since $[W_{t_1}, ..., W_{t_n}] \sim N(\underline{0}, C)$

The choleskey factorization of C gives:

$$A=\begin{bmatrix} \sqrt{t_1}\ 0\ 0\ ...\ 0 \\ \sqrt{t_1}\ \sqrt{t_2 - t_1}\ 0\ ...\ 0 \\ ... \\ ... \\ \sqrt{t_1}\ \sqrt{t_2 - t_1}\ \sqrt{t_3 - t_2}\ ...\ \sqrt{t_n - t_{n-1}} \end{bmatrix}$$

Definition: A process $\underline{W_t} = [W_1(t), W_2(t), ..., W_d(t)]^T, 0 \leq t \leq T$ is a standard Brownian motion on R^d. If

1. $\underline{W_0} = \underline{0}$
2. $\underline{W}$ has continuous sample paths almost surely
3. $\underline{W}$ has independent increments
4. $(\underline{W_t} - \underline{W_s}) \sim N(\underline{0}, (t-s)I), \forall 0 \leq s < t \leq T$

-Each $W_i(t), i = 1, ..., d$ is a standard Brownian motion

-$W_i \perp W_j$ for $i \neq j$

Definition: suppose $\mu \in R^d$ and $\Sigma \in R^{d\times d}$ positive semidefinite we say $\underline{X}$ is a Brownian motion with drift $\underline{\mu}$ and covariance $\Sigma(\underline{X} \sim BM(\underline{\mu}, \Sigma))$ if $\underline{X}$ has continuous sample paths and independent increments with
$(\underline{X}_t - \underline{X}_s) \sim N((t-s)\underline{\mu}, (t-s)\Sigma)$

If $B \in R^{d\times d}$ is a vector such that $BB^T = \Sigma$ and $\underline{W}$ is a standard Brownian motion on R^d then $\underline{X}_t = \underline{\mu}t + B\underline{W}_t \sim BM(\underline{\mu}, \Sigma)$

$\underline{X}$ solves $d\underline{X}_t = \underline{\mu}dt + Bd\underline{W}_t$

Simulation

Let $Z_1, Z_2, ..., Z_n \sim N(0,1)$, independent to simulate $\underline{W_t}$, apply the 1-dimensional random walk construction to each component of $\underline{W_t}$:
$W_j(t_{i+1}) = W_j(t_i) + \sqrt{t_{i+1} - t_i} Z_{i+1}, i = 0, 1, 2, .., n-1$
To simulate $\underline{X_t} \sim BM(\underline{\mu}, \Sigma)$, we need find B definited above $\sim R^{d\times d}$, such that $BB^T = \Sigma$
Set $\underline{X_t} = 0, \underline{X_{t_{i+1}}} = \underline{X_{t_i}} + \underline{\mu}(t_{i+1} - t_i) + \sqrt{t_{i+1} - t_i} B\underline{Z_{i+1}}$

Geometric Brownian Motion

Definition: a stochastic process S_t is a geometric Brownian motion if $ln(S_t)$ is a Brownian motion with initial value $ln(S_0)$

The properties of Geometric Brownian motion:
If S_t is Geometric Brownian motion, S_t does not have independent increments.
Instead,
$\frac{S_{t_2}-S_{t_1}}{S_{t_1}}, \frac{S_{t_3}-S_{t_2}}{S_{t_2}}, ..., \frac{S_{t_n}-S_{t_{n-1}}}{S_{t_{n-1}}}, t_0 < t_1 < ... < t_n$ are independent.

Suppose W is a standard Brownian motion and X satifies $dX_t = \mu dt + \sigma dW_t$.

Then X$\sim BM(\mu, \sigma^2)$

Let $S_t = S_0 e^{X_t} = f(X_t)$, by Ito's formula, we get:

$dS = f_t(X_t)dt + f_x(X_t)dx + \frac{1}{2} f_{xx}(X_t dx^2), f_t = 0, f_x = S_0 e^x, f_{xx} = S_0 e^x$

$dX_i dX_i = (\mu dt + \sigma dW_t)^2 = \sigma^2 dt$

$\Rightarrow dS_t = 0 + S_0 e^{X_t}(\mu dt + \sigma dW_t) + \frac{1}{2}\sigma^2 S_0 e^{X_t} dt$

$$dS_t = S_t(\mu + \tfrac{1}{2}\sigma^2)dt + S_t \sigma dW_t$$

$$\frac{dS_t}{S_t} = (\mu + \tfrac{1}{2}\sigma^2)dt + \sigma dW_t$$

This is a different SDE than what is usually used for Geometric Brownian motion, here μ is the drift for the Brownian motion $X_t = \ln S_t$

If $S_t \sim GBM(\mu, \sigma^2)$, then $S_t = S_0 e^{(\mu - \frac{1}{2}\sigma^2)^2)t + \sigma W_t}$, for u¡t,$S_t = S_0 e^{(\mu - \frac{1}{2}\sigma^2)(t-u) + \sigma(W_t - W_u)}$

Simulation:

for $0 = t_0 < t_1 < ... < t_n$,$S_{t_{i+1}} = S_{t_i} e^{(\mu - \frac{1}{2}\sigma^2)(t_{i+1} - t_i) + \sigma\sqrt{t_{i+1} - t_i} Z_{i+1}}$, for i=0,1,...,n-1, where $Z_1, Z_2, ..., Z_n \sim N(0,1)$ and independent.

This method is exact i.e $S_{t_{i+1}} - S_{t_i}$ has the joint distribution of $S_t \sim GBM(\mu(t_{i+1} - t_i), \sigma^2(t_{i+1} - t_i))$ and that $(S_{t_{i+1}} - S_{t_i}) \perp S_{t_i} - S_{t_{i-1}}$.

0.14 Variance Reduction Techniques

In this chapter, we develop some methods for increasing the efficiency of Monte Carlo simulation by reducing the cariance of simulation estimates. The greatest gains in efficiency from variance reduction techniques results from expoliting specific features of a problem, rather than from generic applications of generic methods.

Control Variates

This method is to expolit information about the error in estimates of known quantities to reduce the error in an estimate of an unknown quantity.

Let $Y_1, Y_2, ..., Y_n$ be outputs of n runs of a simulation. For example,Y_i could be the discounted payoff of an option in the ith simulation path.

Assume the Y_i are independent and identically distributed(iid) and our objective is to estimate $E(Y_i)$.

Estimator: $\overline{Y} = \frac{1}{n}\sum_{i=1}^{n} Y_i$, it is unbiased and converges with probability 1 as n $\to \infty$.

Suppose on each replication we calculate another output X_i in addition to Y_i

Assume the pairs $(X_i, Y_i), i = 1, 2, ..., n$ are independent and identically distributed and $E(X_i)$ is known.

For any fixed b, we can calcucate $Y_i(b) = Y_i - b[X_i - E(X)]$ for the ith replication.

Calculate the sample mean: $\overline{Y}(b) = \overline{Y} - b[\overline{X} - E(X)] = \frac{1}{n}\sum_{i=1}^{n}[Y_i - b[X_i - E(X)]]$.

This is a control variate estimator, and the observed error $X_i - E(X)$ is a control in estimating $E(Y)$.

$$\begin{aligned}
E(\overline{Y}(b)) &= E[\overline{Y} - b[\overline{X} - E(X)]] \\
&= E(\overline{Y}) - b[E(\overline{X} - E(X))] \\
&= E(Y) - b[\underbrace{E(X) - E(X)}_{0}]
\end{aligned}$$

$\Rightarrow \overline{Y}(b)$ is an unbiased estimator of E(Y)

$$\begin{aligned}
\lim_{n\to\infty} \overline{Y}(b) &= \lim_{n\to\infty} \frac{1}{n}\sum_{i=1}^{n}[Y_i - b(X_i - E(X))] \\
&= \lim_{n\to\infty} \frac{1}{n}\sum_{i=1}^{n} Y_i - \lim_{n\to\infty} \frac{1}{n}\sum_{i=1}^{n} X_i + \\
&\quad b * \lim_{n\to\infty} \frac{1}{n}\sum_{i=1}^{n} E(X) \\
&= E(Y) - bE(X) \text{ almost surely.}
\end{aligned}$$

$\Rightarrow \overline{Y}(b)$ is a consistent estimator of E(Y)

Var$[Y_i(b)]$=Var$[Y_i - b(X_i - E(X))]$

$$
\begin{aligned}
&= E[Y_i - b(X_i - E(X))]^2 - E[Y_i - b(X_i - E(X))]^2 \\
&= E(Y_i^2) - 2bE[Y_i(X_i - E(X))] + b^2E[(X_i - E(X))^2] - [E(Y_i)]^2 \\
&= \underbrace{\sigma_y^2 - 2b\sigma_x\sigma_y\rho_{xy} + \sigma^2 b^2}_{\sigma^2(b)}
\end{aligned}
$$

The control estimator $\overline{Y}(b)$ has variance:

$$
\begin{aligned}
Var(\overline{Y}(b)) = Var[\tfrac{1}{n}\textstyle\sum_{i=1}^{n} Y_i(b)] \\
&= \tfrac{1}{n^2} Var(\textstyle\sum_{i=1}^{n} Y_i(b)) \\
&= \tfrac{1}{n^2} \textstyle\sum_{i=1}^{n} Var(Y_i(b)) \\
&= \tfrac{1}{n^2} nVar(Y_i(b)) = \tfrac{\sigma^2(b)}{n}
\end{aligned}
$$

The sample mean $\overline{Y}$ has variance: $Var(\overline{Y}) = \frac{\sigma_y^2}{n}$

Hence, the control variate estimator has smaller variance than the standard estimator if $b^2\sigma_x < 2b\sigma_y\rho_{xy}$, because:

$Var(\overline{Y}(b)) < Var(\overline{Y}) \iff \frac{\sigma^2(b)}{n} < \frac{\sigma_y^2}{n} \iff b^2\sigma_x < 2b\sigma_y\rho_{xy}$

To find a b^* to minimize $\sigma^2(b)$, we derivate $\sigma^b(b)$ and let it equal to 0, then we get:

$-2\sigma_x\sigma_y\rho_{xy} + 2b\sigma_x^2 = 0$

So $b^* = \frac{\sigma_x\sigma_y}{\sigma_x} = \frac{Cov(X,Y)}{Var(X)}$

The minimized variance $\sigma^2(b^*) = \sigma_y^2 - 2(\frac{\sigma_y\rho_{xy}}{\sigma_x})\sigma_x\sigma_y\rho_{xy} + (\frac{\sigma_y\rho_{xy}}{\sigma_x})^2\sigma_x^2 = \sigma_y^2(1 - \rho_{xy}^2)$

Compare the ratio of the optimally controlled estimator to that of the uncontrolled estimator:

$$
\frac{\frac{\sigma^2(b^*)}{n}}{\frac{\sigma^2(0)}{n}} = 1 - \rho_{xy}^2 \tag{23}
$$

Through this formula, we find that the effectiveness of th control variate is determined by the correlation of X and Y.

$\frac{1}{1-\rho_{xy}^2}$ is called the variance reduction factor.

What we discuss above depends on that the optimal coefficient b^* is known. If we don't know σ_x and σ_y, we can also gain most of the benefit of a control variate using an estimate of b^*. For example, replacing the population parameters in $b^* = \frac{\sigma_x \sigma_y}{\sigma_x} = \frac{Cov(X,Y)}{Var(X)}$ with their sample counterparts yields the estimate:

$$\hat{b}_n = \frac{\sum_{i=1}^{n}(X_i - \overline{X})(Y_i - \overline{Y})}{\sum_{i=1}^{n}(X_i - \overline{X})^2} \tag{24}$$

By the strong law of large numbers, $\hat{b} \to b^*$ almost surely as $n \to \infty$, so we can use $\overline{Y}(\hat{b}_n)$ as an estimator.

This suggests using the estimator $\overline{Y}(\hat{b}_n)$, the sample mean of

$$Y_i(\hat{b}_n) = Y_i - \hat{b}_n(X_i - E[X]), i = 1, 2, ..., n. \tag{25}$$

$\hat{b}_n$ is the slope of the least squares regression line through the points(X_i, Y_i).

APPENDIX

Matlab Codes For Monte Carlo Simulation

Spatialcoeff.m

This function takes inputs of the spatial step δx, the number of space grid points N, the volatility σ, the interest rate r and returns a matrix A that results from the spatial discretization of $\frac{\partial u}{\partial t} = r\frac{\partial u}{\partial x} + \frac{1}{2}\sigma^2\frac{\partial^2 u}{\partial x^2} - ru$

```
function [ A ] = spatialcoeffs( deltaX,r,n,sigma )
N = N+1;
gamma = sigma2̂/deltaX2̂+r;
beta = -r/(2*deltaX)+sigma2̂/(2*deltaX);
alpha = .5*(r/deltaX+sigma2̂/deltaX2̂);
A = zeros(N);
gammas = gamma*ones(1,N-2);
betas = beta*ones(1,N-2);
alphas = alpha*ones(1,N-2);
A(2:end-1,2:end-1) = A(2:end-1,2:end-1) + diag(-gammas);
A(2:end-1,1:end-2) = A(2:end-1,1:end-2) + diag(betas);
A(2:end-1,3:end) = A(2:end-1,3:end) + diag(alphas);
A(1,1) = 1;
A(1,2) = -2;
A(1,3) = 1;
A(N,N-2) = 1 ;
A(N,N-1) = -2;
A(N,N) = 1 ;
```

Intergalmc.m

This function takes n as an input and return both the estimate of the integral and the error of the approximation.

```
function[a,b]=intergalmc(n)
u=unifrnd(0,1,1,n);
a=(1/n)sum(u.^2);
Sf=sqrt(1/(n-1))*sum((u.^2-a).^2);
b=Sf/sqrt(n);
end
```

EuroOption.m

This function takes inputs of the number of draws n, the initial underlying asset price S_0, the interest rate r, and the volatiliy σ, it returns the estimated prices of the put and call, along with 95% and 99% confidence intervals for each estimate.

```
function[C,ci1,ci2,P,ci3,ci4]=EuroOptionmc(n,S0,r,sigma,K,T)
z=normrnd(0,1,1,n);
S=zeros(1,n);
C1=zeros(1,n);
P1=zeros(1,n);
for i=1:n
S(i)=S0*exp((r-0.5*sigma))*T+sigma*sqrt(T)*Z(i));
C1(i)=exp(-r*T)max(S(i)-K,0);
P1(i)=exp(-r*T)max(K-S(i),0);
end
C=sum(C1)/n;
P=sum(P1)/n;
Sc1=sqrt(1/(n-1))*sum((C1-C).^2);
```

```
Sc2=sqrt(1/(n-1))*sum((P1-P).2̂);

z1 = norminv([0.025 0.975],0,1);

z2 = norminv([0.005 0.995],0,1);

ci1= zeros(1,2); ci1 = C+z1*Sc1/sqrt(n); ci2 = zeros(1,2); ci2 =
C+z2*Sc1/sqrt(n); ci3= zeros(1,2); ci3 = C+z1*Sc2/sqrt(n); ci4 = zeros(1,2); ci4
= C+z2*Sc2/sqrt(n); end
```

LinConGenerator.m

This function takes modulus m, multiplier a, c, seed x_0 and N as inputs. It returns a vector of length N containing the pseudorandom values.

```
function u = linConGenerator( a,c,x0,N,m )
x = [x0 zeros(1,N-1)];
u = [x0/m ones(1,N-1)];
for i = 1:N-1
x(i+1) = mod(a*x(i)+c,m);
u(i+1) = x(i+1)/m;
end
if c ≠ 0 u;
else if isprime(m) == 0 warning('The generator does not have full period') u;
else a1 = zeros(1, m-2);
for i = 1: m-2
a1(i) = aî-1;
end
t = mod(a1,m);
if mod(a(̂m-1)-1, m) == 0 && isempty(t) u;
else warning('The generator does not have full period') u;
```

```
end
end
end
```

Exponentialgenerator.m

This function returns a vector of length N containing the pseudo- random values.

```
function u = exponentialgenerator( N,lamda )
X = rand(1,N) ;
u = zeros(1,N);
for i = 1:N
u(i) = -log(X(i))/lamda;
end
return
end
```

Boxmuller.m

This function takes N, μ, σ as inputs and returns a vector of length N containing the pseudorandom values.

```
function [V] = boxmuller(N, mu, sigma)
V = zeros (1,N);
for i = 1:2:N
u1 = rand(1);u2 = rand(1);
R = -2*log(u1);v = 2*pi*u2;
Z1 = sqrt(R)*cos(v);Z2 = sqrt(R)*sin(v);
x1 = mu+sigma*Z1;x2 = mu+sigma*Z2;
V(i) = x1;
if i ¡= N-1
```

```
V(i+1) = x2;
end
end
end
```

Bivariatenormal.m

This function takes a 1×2 vector and a 2×2 matrix as inputs and returns a vector of containing the pseudorandom values.

```
function [V]=bivariatenormal(mu,m,N)
A = zeros(2,2);
rho = m(1,2)/sqrt(m(1,1)*m(2,2));
V = zeros(2,N);
for i = 1:N
if m(1,1)¿0 && det(m)¿0
A(1,1) = sqrt(m(1,1));A(1,2) = 0;
A(2,1) = rho*sqrt(m(2,2));A(2,2) = sqrt(m(2,2))*sqrt(1-rho^2);
u1 = rand(1);u2 = rand(1);
R = -2*log(u1);v = 2*pi*u2;
Z1 = sqrt(R)*cos(v);Z2 = sqrt(R)*sin(v);
Z = [Z1;Z2];
X = mu + A*Z;
V(:,i) = X;
else
warning('m is not positive definite')
end
end
```

```
end
```

Brownianmotion.m

This function takes drift μ, volatility σ, time T, and step number N as inputs and returns a vector of simulated values from a Brownian motion.

```
function W = brownianmotion(mu,sigma,T,N,W0)
deltaT = T/N;
Z = randn(1,N);
W = [W0 zeros(1,N-1)];
for i = 1:N-1
W(i+1) = W(i) + mu*deltaT+sigma*sqrt(deltaT)*Z(i+1);
end
plot(1:N,W)
end
```

Geobrownianmotion.m

This function takes drift μ, volatility σ, time T, and step number N as inputs and returns a vector of simulated values from a geometric Brownian motion.

```
function S = geobrownianmotion(mu,sigma,T,N,S0)
deltaT = T/N;
Z = randn(1,N);
S = [S0 zeros(1,N-1)];
for i = 1:N-1
S(i+1) = S(i) * exp((mu-0.5*sigma^2)*deltaT+sigma*sqrt(deltaT)*Z(i+1));
end
plot(1:N,S)
end
```

MultiVarNormal.m

This function takes a take mean vector and covariance matrix as inputs and returns a d-vector of normally distributed random variables.

function [randValues] = multiVarNormal(mu, Sigma, N)

d=length(mu);

$$muRows, muCols$$

= size(mu);

if muRows == 1

mu = mu';

end

$$n, m$$

= size(Sigma);

if $d \neq n | d \neq m$

error('Dimensions must agree.')

end

if Sigma $\neq$ Sigma

error('Sigma must be symmetric.')

end

if min(eig(Sigma))¡= 0

error('Sigma must be positive definite.')

end

A=zeros(d);

for jj=1:d

```
for ii=jj:d
v(ii)= Sigma(ii,jj);
for kk=1:jj-1
v(ii)=v(ii)-A(jj,kk)*A(ii,kk);
end
A(ii,jj)=v(ii)/sqrt(v(jj));
end
end
randValues = zeros(d,N);
for i=1:N
Z=randn(d,1);
randValues( ,i)= mu + A*Z;
end
```

AsianOption.m

This function takes interest rate r, drift μ, volatility σ, time T, and step number N as inputs and returns the estimated prices of asian call and put option with 95% confidence interval.

```
function[callPrice,putPrice,callCI,putCI] =
asianOption(n,S0,r,sigma,K,T,N,M,delta)
if mod(N,M) ≠ 0,
error('asianOption(n,S0,r,sigma,K,T,N,M): N should be a multiple of M.');
end
monitorFactor = N/M;
callPayoffs = zeros(n,1);
putPayoffs = zeros(n,1);
for count = 1:n
```

```
S = geoBrownianMotion(r,sigma,N,T,S0);
SatMdates = zeros(M+1,1);
for index = 1:M+1
monitorDate = 1 + monitorFactor*(index-1);
SatMdates = S(monitorDate);
end
callPayoffs(count) = max(0,mean(SatMdates)-K);
putPayoffs(count) = max(0,K-mean(SatMdates));
end
callPrice = exp(-r*T) * mean(callPayoffs);
putPrice = exp(-r*T) * mean(putPayoffs);
zScore = norminv(1-delta/2,0,1);
callSampleStDev = callPayoffs - callPrice;
callSampleStDev = 1/(n-1)* sum( callSampleStDev.^2);
callCI = [callPrice - zScore*callSampleStDev/sqrt(n),callPrice +
zScore*callSampleStDev/sqrt(n)]
putSampleStDev = putPayoffs - putPrice;
putSampleStDev = 1/(n-1)* sum( putSampleStDev.^2);
putCI = [putPrice - zScore*putSampleStDev/sqrt(n),putPrice +
zScore*putSampleStDev/sqrt(n)]
```

BarrierOption.m

This function takes interest rate r, drift μ, volatility σ, time T, strike price K, initial price S_0 as inputs and returns the estimated price of a barrier option.

```
function [ci1, ci3, C, P] = barrieroptionlzp(mu, sigma, b, T, K, S0, V, N, M)
deltaT = T/N;
```

```
Z = randn(1,N);
S = [S0 zeros(1,N-1)];
for i =1:N-1
S(i+1) = S(i) * exp((mu-0.5*sigma^2)*deltaT+sigma*sqrt(deltaT)*Z(i+1));
end
if mod(N,M) ≠0
error('N must be an integer multiple of M')
else
C1 = zeros (1,M); P1 = zeros (1,M);
for j = 1:M
r = mu;
C1(j) = exp(-r*T)*max(S(30*j)-K,0); P1(j) = exp(-r*T)*max(K-S(30*j),0);
end
SC = zeros(1,M);
if strcmp(V,'down and out')
for m = 1:M
if S(30*m)<b
SC(m) = 0;
else
SC(m) = 1;
end
end
if SC == 1
C = sum(C1)/M; P = sum(P1)/M;
Sc1 = sqrt(1/(M-1)*sum((C1-C).^2));
Sc2 = sqrt(1/(M-1)*sum((P1-P).^2));
```

```
z1 = norminv([0.025 0.975],0,1);
ci1 = C+z1*Sc1/sqrt(M) ;
ci3 = C+z1*Sc2/sqrt(M);
else
warning('The option has been knocked out')
end
elseif strcmp(V,'down and in')
for m = 1:M
if S(30*m)> b
SC(m) = 1;
else
SC(m) = 0;
end
end
if SC ≠ 1
C = sum(C1)/M; P = sum(P1)/M;
Sc1 = sqrt(1/(M-1)*sum((C1-C).^2));
Sc2 = sqrt(1/(M-1)*sum((P1-P).^2));
z1 = norminv([0.025 0.975],0,1);
ci1 = C+z1*Sc1/sqrt(M) ;
ci3 = C+z1*Sc2/sqrt(M);
else
warning('The option has not been activated')
end
elseif strcmp(V,'up and out')
for m = 1:M
```

```
if S(30*m)< b
SC(m) = 1;
else
SC(m) = 0;
end
end
if SC == 1
C = sum(C1)/M; P = sum(P1)/M;
Sc1 = sqrt(1/(M-1)*sum((C1-C))^2));
Sc2 = sqrt(1/(M-1)*sum((P1-P).^2));
z1 = norminv([0.025 0.975],0,1);
ci1 = C+z1*Sc1/sqrt(M) ;
ci3 = C+z1*Sc2/sqrt(M);
else
warning('The option has been knocked out')
end
else strcmp(V,'up and in')
for m = 1:M
if S(30*m)> b
SC(m) = 0;
else
SC(m) = 1;
end
end
if SC ≠ 1
C = sum(C1)/M; P = sum(P1)/M;
```

```
Sc1 = sqrt(1/(M-1)*sum((C1-C).2̂));
Sc2 = sqrt(1/(M-1)*sum((P1-P).2̂));
z1 = norminv([0.025 0.975],0,1);
ci1 = C+z1*Sc1/sqrt(M) ;
ci3 = C+z1*Sc2/sqrt(M);
else
warning('The option has not been activated')
end
end
end
end
```

Basketoption.m

This function takes a drift vector, a covariance matrix, an initial underlying price vector, maturity time T, step number N, a weights vector, interest rate r, strike price K as inputs and returns a estimated price of basket option.

```
function [optionprice] = basketoption(mu,Sigma,S0,N,K,c,T)
S = multipleGeoBrownianMotion(mu, Sigma, T, N, S0);
callpayoffs = zeros(1,N);
for i = 1:N
callpayoffs(i) = max(sum(c.*S(:,i)'- K),0);
end
optionprice = mean(callpayoffs);
end
```

0.15 References

Andersen, L.B.G. & Brotherton-Ratcliffe, R.(1997). The Equity Option Volatility Smile: An Implicit Finite-Difference Approach. *Journal of Computational Finance*,1,5-38

Brennan, M.J., & Schwartz, E.S. (1980). *Jounral of Financial and Quantitative Analysis, Proceedings Issue*,15,(4),907-929.

Clewlow, L., & Strickland, C. (1998). *Implementing Derivative Models.* Chichester,England: Wiley.

Derman, E., kani, I., & Chriss, N. (1996). Implied Trinomial Trees of the Volatility Smile. *Journal of Derivatives*, 3, 7-22.

Rogers, L.C.G.,& Zane, O. (1997). Valuing Moving Barrier Options. *Journal of Computational Finance,* 1, 5-11.

Abken, P.A. (2000) An Empirical Evaluation of Value at Risk by Scenario Simulation, *Journal of Derivatives 7(Summer):* 12-30.

Beasley, J.D., and Springer, S.G. (1977) *The Percentage Points of the Normal Distribution*, Applied Statistics 26: 118-121.

Chung, K.L. (1974) *A Course in Probability Theory*, Second Edition, Acdemic Press, New York.

Devroye, L. (1986) *Non-Uniform Random Variate Genenration*, Springer-Verlag, New York.

Niederreiter, H. (1992) *Random Number Generation and Quasi-Monte Carlo Methods*, Society for Industrial and Applied Mathematics, Philadelphia.