Monte Carlo Fundamentals

Checks for Randomness

Overview

Checking for randomness is just hypothesis testing where the null hypothesis is:

H_0: The numbers are random

H_1: They are not random

So to implement such a hypothesis test we need to think of features we would expect to see in truly random numbers

❓ *Do we need to check for randomness of normal variables if we use Box-Muller or other distribution if we use the inverse CDF method*

Interval test

On average we would expect the proportion of random numbers $X \sim N(0, 1)$ that are less than c to be c

We need to develop this into a more formal statistical test though so we realise than then number of random number less than c will be binomially distributed

$X \sim Bin(N, c)$ where:

- X is the number of random numbers less than c drawn from N trials

Given we are going to test with very large values of N we can assume a normal distribution as so we assume

$$X \sim N(Nc, Nc(1 - c))$$

Example 1

Suppose we run 10000 simulations and we find that 5200 of the random number are less than 0.5. Do we accept or reject the null hypothesis that our numbers are random

We expect $X \sim N(5000, 2500)$ and so our 95% confidence interval is $(5000 - 1.96 \times \sqrt{2500}, 5000 + 1.96 \times \sqrt{2500}) = (4902, 5098)$

How many numbers fall within a given set

We can easily extend the above test to any set on the interval $U(0, 1)$

? *Why is this important*

Example 2

Suppose we run 1,000,000 simulations from $U(0, 1)$ and 803,000 of them fall between 0.1 and 0.9

Falling between 0.1 and 0.9 has a binomial distribution

$X \sim Bin(1000000 \times 0.8, 1000000 \times 0.8 \times 0.2) = N(800,000, 400^2)$

So again our 95% confidence interval is

$(800,000 - 1.96 \times 400, 800,000 + 1.96 \times 400) = (799216, 800784).$

and we can reject H_0

? *Why might this test be particularly useful*

Further Examples

We could also test how many random numbers are in the set
$[0, 0.1) \cup [0.2, 0.3) \cup [0.4, 0.5) \cup [0.6, 0.7) \cup [0.8, 0.9)$

That is: do 50% of the random numbers have a first decimal place that is even

easily we could extned this to counting how many random number have a third decimal place which is even

? *How would you test for the third decimal place being even*

```
--- endif
```

Visual Frequency Test

We should not under-estimate the power of the human eye to detect problems with our random numbers

Many patterns that are difficult to detect by statistical methods will be instantly visible to the human eye

Method

The method is very simple:

- Generate thousands or random numbers and group them into buckets
- Plot a histogram of the number in each bucket

In purely statistical terms this is a mere extension of the above and does not really add anything, but any bias or pattern will be clearly visible

We cannot use this type of test to measure the extent of the randomness but only to detect any obvious unrandom patterns

Exercise

Design an LCG: choose m, a, c and X_0

Plot a histogram of the returns to see if any obvious non - random features are visible

Frequency Distribution Test

We can extend the above to look at the distribution of the frequency that numbers fall with each interval.

This is useful because simply counting across the interval $U(0, 1)$ would fool the above test if it was done in sufficiently small intervals

Method

We need an array to count the numbers of times we fall in each interval.

Supposing we subdivide the interval $U(0, 1)$ into 100 intervals $U(\frac{n-1}{100}, \frac{n}{100})$

We then need to group our data into a frequency array

We finally need to compare this frequency array with the normal distribution we would expect to get if true randomness were being observed

This is slightly confusing as there is effectively a double frequency count going on. Lets look at an example

If we do 1000000 (1m) random numbers then we expect the number that are in the interval (0.230,0.231) to be $Bin(1000000, 0.001)$ or approximately $N(1000, 999)$

In fact this is the same distribution for all the intervals

So we have 1000 numbers all from $N(1000, 999)$

We then group these into a histogram and compare with the normal distribution

Exercise

Design an LCG: choose m, a, c and X_0

Perform the above statistical test, which should result in a spreadsheet with your histogram with the normal distribution overlaid on top

Extensions

You could test whether individual frequencies in your histogram fell within an appropriate confidence interval

You could could that the number of frequencies that fell with a 95% confidence interval was about 95%

You could do a chi-squared test on the actual versus expected from the histogram

Independence

For truly random numbers each number will be completely independent of the previous one

Thinking back to the definition of independence

$$f_{X,Y}(x,y) = f_X(x) \times f_Y(y)\ \forall x, y$$

We also need independence of all preceding random numbers not just the previous one so our definition must extend to:

$$f_{X_1,X_2,\ldots,X_n}(x_1, x_2, .., x_n) = f_{X_1}(x_1) \times f_{X_2}(x_2) \times \ldots \times f_{X_n}(x_n)$$

$$\forall x_1, x_2, \ldots, x_n$$

Really this is just another way of saying that whatever the values of $x_1, x_2, \ldots, x_{n-1}$ are the values of x_n should be unaffected

We can test for this by selecting subsets of our random numbers x_n such that the previous numbers $x_1, x_2, \ldots, x_{n-1}$ have any given property and do all the same randomness tests on this subset of x_n

Example 1

Suppose there is a strong serial correlation between successive values in our random number generator.

values of $x_n > 0.5$ would be more likely when the previous value was $x_{n-1} > 0.5$

selecting only those values of x_n for which $x_{n-1} > 0.5$ and then testing that the proportion of these values is $Bin(n, 0.5)$ would cause a fail in H_0 of independence

Example 2

You are concerned that the random number generator tends to pick a number less than the previous one if the previous two numbers were higher that the one which preceded them

Solution

In the middle of your test algorithm:

```
do
  n=n+1
  new_num(n) = my_LCG()
 loop until ((new_num(n-1)>new_num(n-2)) and (new_num(n-2)>new_num(n-3)))
```

`new_num(n)` is now in your subset of random numbers on which to perform tests

Take care

? *If* $x_{n-1} > x_{n-2}$ *what is* $P(x_n > X_{n-1})$

Variance Reduction Techniques

The big problem with Monte Carlo is that it is always an approximation

The solution is to do lots of runs

The problem with lots of runs is that they take time

However we can increase the rate of convergence using certain techniques which we call ***variance reduction techniques***

Motivating example

Suppose I wish to calculate the expected value of $U(0,1)$

I could take 1000 random numbers from $U(0,1)$ and take the average of them

The standard deviation of my error would be $\frac{1}{\sqrt{12}\times\sqrt{1000}} = 0.009129$

The problem I have here is that I am taking numbers randomly from all over the line.

Suppose I wanted to do something more akin to numerical integration where we would go step by step across the interval adding up each number

The compromise solution is

Take a random number from $U(0, 0.1)$ and then synthesise a number from each of $U(0.1, 0.2)$ etc by adding 0.1 then 0.2 etc to the original number

If we repeat this 100 times so that we have a total of 1000 random numbers (100 random + 900 generated) then

The variance of the first number is $\frac{1}{12} \times 0.1^2 = \frac{1}{1200}$

The variance of the average of the first 10 numbers is then the same: $\frac{1}{12} \times 0.1^2 = \frac{1}{1200}$

The variance of the 1000 numbers is then $\frac{1}{100} \times \frac{1}{12} \times 0.1^2 = \frac{1}{120000}$

and the standard error is $\sqrt{\frac{1}{120000}} = 0.002887$

So the standard error is reduced by a factor of $\sqrt{10}$

This spreadsheet performs the experimental confirmation of this

Monte Carlo Fundamentals

Introduction

Monte Carlo calculations allow us to use random numbers to find approximate solutions to mathematical problems that may otherwise be too complicated to solve

There are many different ways in which this can help

- Analysis of future uncertain events
- Analysis of probability of uncertain outcomes
- Solution of complex probability problems
- Solution of integral type equations

The fundamental issue around all Monte Carlo analysis is the the results are approximate

Law of Large Numbers

The ***Law of Large Numbers*** states that if a large number of random vents occur then the average outcome will converge on the expected outcome

It is also known as the ***law of averages***

Sometimes it is used when to few trials have been used - this is known as ***Gambler's fallacy***

Example:
If 1000 random numbers are drawn from the distribution $N(48, 20^2)$ what is the probability that the average lies between 47 and 49

Solution:

The sum of 1000 trials is $N(48000, 1000 \times 20^2)$

So
$P(47 < Average < 49) = P(47000 < Sum < 49000)$

=

=

=

=

Example 2

If random numbers are drawn from the distribution $U(0,1)$ how many have to be drawn before the average is between 0.4999 and 0.5001 with a probability of 95%

Solution

Clearly $E(U(0.1)) = 0.5$ but what is $Var(U(0,1))$

We use $Var(X) = E(X^2) - (E(X))^2$

$$Var(X) = \int_0^1 x^2 dx - (0.5)^2$$

Then we need to assume that the central limit theorem will apply and so the sum of n trials will be:

$$Sum \sim N\left(\frac{n}{2}, \frac{n}{12}\right)$$

The average of n trials will be

By symmetry we need to solve:

Extension 1:

How many trials would we need to for the average to be between 0.49 and 0.51 with a probability of 95%

Extension 2:

If we do 10,000 trials what accuracy would we expect?

Solve

$$A = \frac{1.96}{\sqrt{12000}} = 0.005658$$

therefore the 95% confidence interval is $(0.4943, 0.5057)$

Using VBA and Excel to Generate Random Numbers

Excel

In Excel we use the function `rand()` to generate a random number from $U(0,1)$

VBA

In VBA we use the function `rnd()` to generate a random number from $U(0,1)$

Exercise:

Test the above calculation by finding the average of 3,201 random numbers generated from $U(0,1)$ and test how many times it falls between 0.49 and 0.51.

Spreadsheet is here

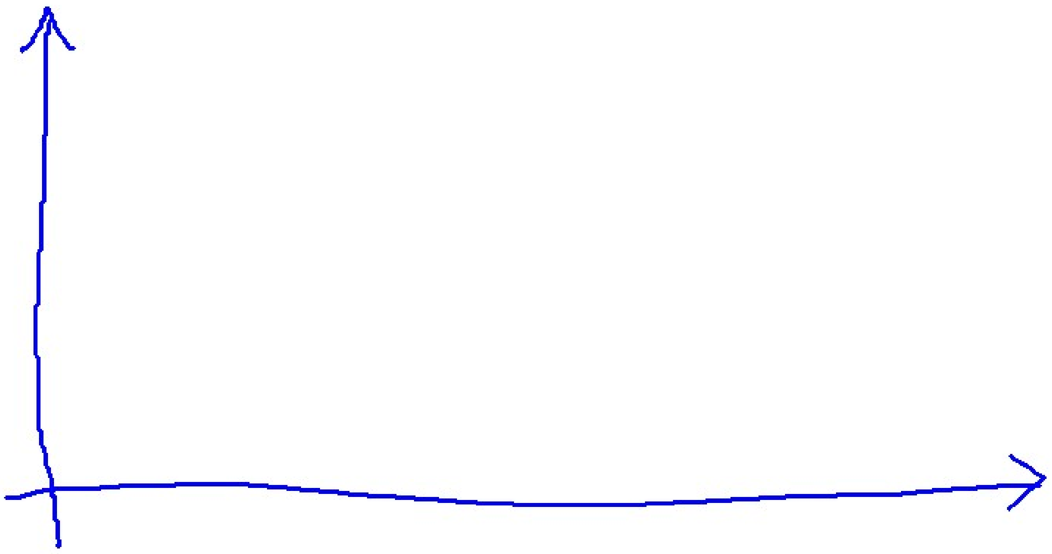
Inverse CDF method

We use the inverse CF method to generate random variables from different probability distributions given we can start from the uniform distribution

The steps are:

- Integrate the probability density function (p.d.f.) to get the cumulative density function (c.d.f.)
- Invert the c.d.f. so you can find the 'value' from the cumulative probability.
- Generate random variables from U(0,1) and pass these into the inverted c.d.f.

The diagrams below illustrate this:



0	1	2	3	4	5	6	7

Exercise

Write a function which generates numbers randomly from a Weibull distribution

Remember the pdf is $f(x) = \frac{k}{\lambda}\left(\frac{x}{\lambda}\right)^{k-1} e^{-(x/\lambda)^k}$, for x >= 0 and 0 elsewhere

Hint: Use the substitution $u = e^{-(x/\lambda)^k}$ to calculate $F(x)$

This spreadsheet shows the method and compares the results with the theoretical pdf of the Weibull distribution

Box-Muller

? *Can we apply the inverse c.d.f. method to the normal distribution?*

Fortunately there is an algorithm which we can use to produce randomly distributed normal variables

This is called the ***Box-Muller*** algorithm and it produces pairs of independent random variables from the standard normal distribution: $N(0,1)$

Algorithm

Let U_1 and U_2 be r.v.s from the uniform distribution $U(0,1)$ then if

$Z_1 = \sqrt{-2lnU_1}cos(2\pi U_2)$ and

$Z_2 = \sqrt{-2lnU_1}sin(2\pi U_2)$

then Z_1 and Z_2 are independent random variables both with a distribution of $N(0,1)$

Exercise

Write a VBA routine to produce normal random variables using the Box-Muller algorithm

A spreadsheet can be found here

- *Is there a short cut we can use in excel?*

- *Is this as good as the Box Muller approach*

- *How could we generate random numbers from the distribution* $N(\mu, \sigma^2)$

- *How would we generate lognormal r.v.s such as* $Y \sim lnN(\mu, \sigma^2)$

Proof

The proof is motivated by thinking about random variables on a circular plane

we define $R = \sqrt{-2ln(U_1)}$

so $P(R \leq r) = P(-2ln(U_1) \leq r^2)$

$= P(ln(U_1) \geq -\frac{r^2}{2})$

$= 1 - P\left(U_1 < exp\left(-\frac{r^2}{2}\right)\right)$

As U_1 is uniform on $U(0,1)$ so we can say

$$P(R \leq r) = 1 - \int_0^{exp(-r^2/2)} dt = 1 - exp\left(-\frac{r^2}{2}\right)$$

So we can differentiate the cdf to get the pdf and we see

$f_R(r) = exp\left(-\frac{r^2}{2}\right) \times r$ for $r > 0$

We define a new random variable Φ to be $\Phi = 2\pi U_2$

and we can see that: $f_\Phi(\phi) = \frac{1}{2\pi}$ for $0 < \phi \leq 2\pi$

Given U_1 and U_2 are independent then so must R and Φ be

so we can multiply pdf to get joint density functions so

$f_{R,\Phi}(r,\phi) = f_R(r) f_\Phi(\phi) = \frac{1}{2\pi} exp\left(-\frac{r^2}{2}\right) . r$

so the probability that an event lies in a little bit of the $R - \Phi$ plane is:

$P = \frac{1}{2\pi} exp\left(-\frac{r^2}{2}\right) . r . dr . d\phi$

But we are interested in random variables over the $X - Y$ plane not the $R - \Phi$ plane

So we need to equate probabilities for a little bit of the area of the $R - \Phi$ plane and the $X - Y$

i.e. $\frac{1}{2\pi}exp\left(-\frac{r^2}{2}\right).r.dr.d\phi = f_{X,Y}(x,y)dxdy$

geometrically we can see that $drd\phi$ describes an area length dr and width $r \times d\phi$ that is an area of $r \times drd\phi$, whereas $dxdy$ simply describes an area of $dx \times dy$

So to transform from the $R - \Phi$ plane to the $X - Y$ plane we simply divide the pdf by r and we have:

$$f_{X,Y}(x,y) = \frac{1}{r} \times \frac{1}{2\pi}exp\left(-\frac{r^2}{2}\right).r$$

$$\therefore f_{X,Y}(x,y) = \frac{1}{2\pi}exp\left(-\frac{r^2}{2}\right)$$

$$\therefore f_{X,Y}(x,y) = \frac{1}{2\pi}exp\left(-\frac{x^2+y^2}{2}\right)$$

$$\therefore f_{X,Y}(x,y) = \sqrt{\frac{1}{2\pi}}exp\left(-\frac{x^2}{2}\right).\sqrt{\frac{1}{2\pi}}exp\left(-\frac{y^2}{2}\right)$$

and so we have both X and Y are normally distributed and independent.

Inside the Random Number Generator

Computers cannot generate true random numbers as they only perform deterministic calculations.

If you really need true random numbers (and we generally do not) then this company takes atmospheric data to produce true random numbers

For our purposes we use pseudo random numbers which computers can generate.

The standard algorithm is called a ***linear congruential generator***

Linear Congruential Generator

These are defined by a recurrence relationship such as:

$$X_{n+1} = (aX_n + c)\ mod\ m$$

The choice of a, m and c is critical to the effective working of the method.

Let's build a spreadsheet and see how this works

What factors do you think would make a good choice of a, m and c

Try experimenting with different values until you have produced a good LCG

Excel random numbers

The random number generator in excel is a pseudo random number generator but it is a very advanced one and we can use it as if it produces true random numbers

Simple Probability Calculation

If you roll 4 dice what is the probability that the sum is 7

Try to do this calculation with a traditional probability method

Now try to use a Monte Carlo method

Calculation of π

The area of a circle is $A = \pi r^2$

Can you use this to develop a Monte Carlo algorithm to calculate π

Integration of Well Behaved Functions

When you price options we can use Monte Carlo methods but we can also use the Black-Scholes formula.

? *Why can two such different methods do the same thing*

- Because essentially Monte Carlo is about adding up lots of different values
- So it is like an integration of sorts
- The Black Scholes formula is proved by integrating over the possible share prices

Can you use a Monte Carlo method to find:

a) $\int_0^{\pi/2} sin(x) dx$

b) $\int_0^1 e^{-x^2/2} dx$

? *Is this a sensible method to use for these integrations?*

Integration of Poorly Behaved Functions

What about function which cannot be solved analytically and would be unstable if quadrature techniques were applied

For example how about the calculation:

$$\int_0^1 sin\left(\frac{1}{x}\right) dx$$

Here you will find a Monte Carlo technique will give you a robust but approximate way of calculation the integral

Simple Option Pricing

Introduction

So far you will be familiar with the binomial model and the Black Scholes model for option pricing

Fundamentally as a result of the Cameron Martin Girsanov Theorem and the Central Limit Theorem both these methods are effectively the same thing

As the number of steps tends to ∞ the binomial model tends to the Black Scholes model

The Monte Carlo methods are also fundamentally the same thing - but with the Monte Carlo methods we can start from a binomial perspective or from a Black Scholes perspective and gain insight into how all the methods are effectively equivalent

Additionally with the Monte Carlo method it is much easier to extend the method to deal with more advanced modelling of equity prices with features such as skew and stochastic volatility

$\mathbb{P}$ and $\mathbb{Q}$ Review

Remember when you first looked at the binomial model

You started with an uptick and a downtick

Then do some algebra and the formula that emerges is $f_n = e^{-r\Delta t} \times E_{\mathbb{Q}}\left(f_{n+1}\right)$, that is effectively an expected value over some 'other' probability measure '$\mathbb{Q}$'

Review of CRR model

We start with a very simple model of the price evolution of our underlying stock

The stock has price S_0 at time 0 and the model advances in units of time Δt

The term of the option is T

We assume the volatility of the stock is σ, which leads to a natural choice of u and d of:

$u = e^{\sigma\sqrt{\Delta t}}$ and $d = \frac{1}{u}$

and $q = \frac{e^{r\Delta t} - d}{u - d}$ - from arbitrage free construction

The value at each node is then worked out from the subsequent nodes with the formula: $f_n = e^{-r\Delta t} \times E_{\mathbb{Q}}\left(f_{n+1}\right)$

These formulas are derived by considering a hedging portfolio of stocks and cash

The important thing about u and d is the gap between them as the Girsanov Theorem can sort out the drift but not the volatility

The $\sqrt{\Delta t}$ is to make the unit time volatility σ

So simply by changing the probabilities of the uptick and the downtick we can express the market price of an option (or any other financial asset) as an expected value

This extends through the Cameron Martin Girsanov Theorem to the full Black Scholes Model.

An illustration of this can be seen in this graphical simulation

Binomial Monte Carlo

Weighted Probabilities

Much of the mathematics of the binomial method has to be reproduced to use the Monte Carlo method

although Monte Carlo methods are often simpler than other methods we ***CANNOT***

- Use arbitrary values of u and d
- Use arbitrary $\mathbb{Q}$ measures

All we can do in the Monte Carlo method is simplify the algorithm by being able to pick the up tick and the down tick randomly rather than having to write the code to go through every single permutation

For given values of S_0, K, T, ΔT and σ can you write a Monte Carlo version of the binomial model

? *Can you explain how this is simpler to code than the normal binomial model*

? *What other advantages does it have?*

Distribution Shift

Fundamentally the CMG is about $\frac{d\mathbb{Q}}{d\mathbb{P}}$ but the visible impact of this is to appear to shift the distribution from the expected return on equity to the risk free rate as we have seen in the above demonstration

Can you change your model to reflect this directly in the VBA

Hint: Preserve the ratio of $u : d$ and solve $q = 0.5$

Solution:

$u = C \times e^{\sigma\sqrt{\Delta t}}$ and $d = C \times e^{-\sigma\sqrt{\Delta t}}$

$$q = \frac{e^{r\Delta t} - d}{u - d} = 0.5$$

$$q = \frac{e^{r\Delta t} - C \times e^{-\sigma\sqrt{\Delta t}}}{C \times e^{\sigma\sqrt{\Delta t}} - C \times e^{-\sigma\sqrt{\Delta t}}} = 0.5$$

$$C = \frac{e^{r\Delta t}}{e^{-\sigma\sqrt{\Delta t}} + 0.5 \times e^{\sigma\sqrt{\Delta t}} - 0.5 \times e^{-\sigma\sqrt{\Delta t}}}$$

$$C = \frac{2 \times e^{r\Delta t}}{e^{\sigma\sqrt{\Delta t}} + e^{-\sigma\sqrt{\Delta t}}}$$

Multiple Step Lognormal

It is a necessary limitation of the binomial model to only have discrete state spaces.

This clearly means an uptick and a downtick in the case of the binomial model but a basic extension is the trinomial model which has three ticks from each state

However for a Monte Carlo approach we are not limited to discrete state spaces.

? *Why*

Therefore it is possible (and sensible) to extend the model by allowing the return at each point to be generated fully from a lognormal distribution

Extend your model to allow for full lognormal steps

Monte Carlo Options

? *How many steps is it now sensible to have to get to an accurate result (You may assume sufficient Monte Carlo runs).*

One Step Monte Carlo

Test your model using just one step

? *Can you reconcile the mathematics of what we have done to the proof of the Black Scholes formula*

Ito's Lemma is Still Necessary

If you have followed the steps above you will probably expect to have replicated Black Scholes results with a one-step lognormal model.

However (unless you have thought ahead to this section) it is highly likely you will have made a mistake and you answers will be too high

The mistake we have made is to assume that the solution of the integral equation: $\frac{dS_t}{S_t} = rdt + \sigma dW_t$ is $S_t = S_0 e^{rt+\sigma W_t}$

When in fact it is $S_t = S_0 e^{(r-0.5\sigma^2)t+\sigma W_t}$

Proof

$$\frac{dS_t}{S_t} = rdt + \sigma dW_t \implies dS_t = rS_t dt + \sigma S_t dW_t$$

Now look at Ito's Lemma:

$$df(t, X_t) = \left[\frac{df}{dt} + \mu(t, X_t)\frac{df}{dx} + \frac{1}{2}\sigma^2(t, X_t)\frac{d^2f}{dx^2}\right]dt + \sigma(t, X_t)\frac{df}{dx}dW_t$$

Where $dX_t = \mu(t, X_t)dt + \sigma(t, X_t)dW_t$

We set this up as follows:

Rewrite with S_t instead of X_t

Then our μ is rS_t and our σ is σS_t

The we use the function $f(t, S_t) = ln(S_t)$ (there is no way of knowing this other than it works)

so $df(t, S_t) = \left[\frac{df}{dt} + \mu(t, S_t)\frac{df}{ds} + \frac{1}{2}\sigma^2(t, S_t)\frac{d^2f}{ds^2}\right]dt + \sigma(t, S_t)\frac{df}{ds}dW_t$

then $df(t, S_t) = \left[\frac{df}{dt} + rS_t\frac{df}{ds} + \frac{1}{2}\sigma^2 S_t^2\frac{d^2f}{ds^2}\right]dt + \sigma S_t\frac{df}{ds}dW_t$

Now we need to calculate our derivatives:

$$f(t, S_t) = ln(S_t) \implies \frac{df}{dt} = 0, \frac{df}{ds} = \frac{1}{S_t}, \frac{d^2f}{ds^2} = \frac{-1}{S_t^2}$$

So putting it all together:

$$dln(S_t) = \left[0 + rS_t\frac{1}{S_t} + \frac{1}{2}\sigma^2 S_t^2 \frac{-1}{S_t^2}\right] dt + \sigma S_t \frac{1}{S_t} dW_t$$

Simplify

$$dln(S_t) = \left[r + \frac{1}{2}\sigma^2\right] dt + \sigma dW_t$$

Integrate:

$$\int dln(S_t) = \int (r + \frac{1}{2}\sigma^2) dt + \int \sigma dW_t$$

$$ln(S_t) = (r + \frac{1}{2}\sigma^2)t + \sigma W_t + lnS_0$$

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

Code for Different Methods

We can now see below the code for the three different methods

- The Binomial Model
- The Binomial Monte Carlo Model
- The LogNormal Model

```
Option Explicit
Function max(x As Double, y As Double) As Double
  If x > y Then
    max = x
   Else
    max = y
   End If
End Function

Function CRR(S0, K, r, T, sigma As Double, n As Integer) As Double
  Dim S() As Double
  ReDim S(0 To n, 0 To 2 ^ n - 1)
  Dim f() As Double
  ReDim f(0 To n, 0 To 2 ^ n - 1)
  Dim delta_t, u, d As Double
  Dim j As Double
  Dim q As Double
  Dim node_x, node_y As Integer
  delta_t = T / n
  u = Exp(sigma * delta_t ^ 0.5)
  d = Exp(-sigma * delta_t ^ 0.5)
  q = (Exp(r * delta_t) - d) / (u - d)
  S(0, 0) = S0
  For node_x = 1 To n
    For node_y = 0 To 2 ^ node_x - 1
      If node_y Mod 2 = 0 Then
        S(node_x, node_y) = S(node_x - 1, node_y / 2) * u
       Else
        S(node_x, node_y) = S(node_x - 1, (node_y - 1) / 2) * d
       End If
     Next node_y
   Next node_x

  For node_y = 0 To 2 ^ n - 1
    f(n, node_y) = max(S(n, node_y) - K, 0)
   Next node_y

 For node_x = n - 1 To 0 Step -1
   For node_y = 0 To 2 ^ node_x - 1
     f(node_x, node_y) = Exp(-r * delta_t) * (q * f(node_x + 1, node_y * 2) + (1 - q) * f(node_x + 1, node_y * 2 + 1))
    Next node_y
  Next node_x
CRR = f(0, 0)
End Function

Function Monte_Carlo_Binomial(S0, K, r, T, sigma As Double, n, runs As Integer) As Double
  Dim delta_t, u, d As Double
  Dim j As Long
  Dim q As Double
  Dim S As Double
  Dim sum As Double
  Dim payoff As Double
  Dim run As Long
  delta_t = T / n
  u = Exp(sigma * delta_t ^ 0.5)
  d = Exp(-sigma * delta_t ^ 0.5)
  q = (Exp(r * delta_t) - d) / (u - d)
  sum = 0
  For run = 1 To runs
    S = S0
    For j = 1 To n
      If (Rnd() < q) Then
        S = S * u
       Else
        S = S * d
       End If
     Next j
    payoff = max(S - K, 0)
    sum = sum + payoff
   Next run
  Monte_Carlo_Binomial = Exp(-r * T) * sum / runs
End Function

Function Monte_Carlo_LogNormal(S0, K, r, T, sigma As Double, n, runs As Integer) As Double
  Dim delta_t, u, d As Double
  Dim j As Long
  Dim q As Double
  Dim S As Double
  Dim jump As Double
  Dim sum As Double
  Dim payoff As Double
  Dim run As Long
  delta_t = T / n
  sum = 0
  For run = 1 To runs
    S = S0
    For j = 1 To n
      jump = Exp(Application.NormSInv(Rnd()) * sigma * Sqr(delta_t) + (r-0.5*sigma^2) * delta_t)
      S = S * jump
     Next j
    payoff = max(S - K, 0)
    sum = sum + payoff
   Next run
  Monte_Carlo_LogNormal = Exp(-r * T) * sum / runs
End Function
```

Monte Carlo Adaptations

Introduction

So far we have considered the basic lognormal model for stock prices.

This is perfectly valid as a starting point but in reality stock prices movements are not adequately modelled by the lognormal model

Notably they:

- Exhibit sustained periods of higher volatility (Conditional Heteroscedasticity)
- Sometimes experience discrete price jumps or falls
- Are more kurtose than would be predicted from a lognormal distribution

When options are 'at the money' these are not great considerations but when an option is severely out of the money then it is very important because:

We look at a number of ways of modelling these additional features within the Monte Carlo framework

The main ones are

- Price Jumps
- Stochastic volatility

We always have a problem with models in that the data we have to fit them has considerable randomness and so there is an ever present danger of over fitting two many parameters

This spreadsheet shows how daily returns on the FTSE100 compares with the lognormal distribution

Jumps

Jumps in share prices are more formally known as Levy processes

Given that share prices are responding to information then any time when a significant piece of information which might effect a share price becomes known then there is a possibility that a share price may jump and so a simple lognormal model of share price movements may not be appropriate

This is particularly important for out of the money options close to maturity

The are two different types of jumps we need to consider:

- Those with known times but uncertain size
- Those with uncertain times

Price Jumps at Known Times

What fundamental fact about the modelling of share prices with a lognormal Monte Carlo process can we use to simply to introduction of price jumps into our analysis?

Consequently the incorporation of known jumps into a MC valuation is very simple:

- Divide the process into intervals defined by when each jump is
- Add or subtract the price jump at each point in time to the share price
- Proceed as usual with option valuation

How should the size of the jump be modelled

It is important to note here that for consistency with arbitrage free pricing each jump should have an expected value of 0 under Q

Price Jumps at Unknown Times

This is very similar in principle to having price jumps at known times with the exception that we now have to model when the jumps occur as well

? *If we believe jumps are going to arise as a Poisson process how do we model the times of the jumps?*

? *How do we model exponential waiting times?*

Adapt your option valuation code from the previous chapter to allow for share price jumps with size $N(\mu, \sigma^2)$ which arise at a Poisson rate of λ

Merton's Jump Diffusion Process

Clearly there are other ways of including jumps into a share price model.

The important thing to retain though is the no arbitrage condition

In Merton's Jump diffusion Process The governing SDE is

$$\frac{dS_t}{S_t} = (r - \lambda\bar{k})dt + \sigma dW_t + kdq_t$$

The jump event is governed by a compound Poisson process q_t with Poisson rate λ

and k is the magnitude of the random jump given by: $ln(1 + k) \sim N(\gamma, \delta^2)$

so $\bar{k} = E(k) = e^{\gamma + \delta^2/2} - 1$

The key thing to note here though is that the expected return of the process under Q will still be $e^{r\Delta t}$ because the expected size of the jumps is backed out of the drift component of the diffusion

Stochastic Volatility

Sometimes we may notice that share prices not only move in occasional discrete jumps but can become more volatile after a given event.

This reflects the reality of traders trying to settle to a new view of the market as they take on new information

There are many different views we could take about how to model this depending to a large part on how we interpret the real World events that drive it.

Here are a few examples

- We could make the volatility parameter increase for a given period after an individual jump event
- We could make the volatility parameter higher or lower depending on whether the share price was higher or lower
- We could make the volatility parameter increase (or decrease) depending on the actual realised sample volatility over a previous time period
- We could observe that there are discrete periods of higher or lower volatility which appear to act like two discrete states with the system moving between the two states

Increased volatility post jump

This is easy to model but very difficult to parameterise as there are now multiple modelling variables that would need to be extracted from random historic share price data

? *What additional parameters would now be needed?*

- Increase in volatility and how this related to the size and direction of the jump
- How long the increase in volatility lasts for (and whether this is a function of the size of the jump)
- Whether the increase in volatility stops at some point, or whether it simply fades away

Adapt your model so that a jump in price is followed by one month of a volatility increase of proportionately the same size as the price jump.

? *Could you adapt your code so that the period of increased volatility was random*

? *Could you adapt your code so that the increased volatility faded away*

Volatility and share price correlation

This is the Monte Carlo version of the Heston Stochastic Volatility Model

Formally the set us is as follows

The Heston SV Model

It is an observed fact of markets that there are periods when volatility is higher than others

The traditional Black-Scholes model does not pick this up as there is a constant volatility assumption σ

What market evidence have you seen of this?

The ***Heston Stochastic Volatility Model (HSV)*** seeks to address this problem by treating volatility itself as its own stochastic process

Maths

Formulation

The original 1993 paper by Heston can be found here

Suppose under a real world probability measure Q, the price process S_t of the underlying security is governed by the following diffusion process

$$dS_t = \mu S_t dt + \sigma_t S_t dW_{1t}$$

and the volatility process σ_t follows the diffusion process:

$$d\sigma_t = -\alpha\sigma_t dt + \beta dW_{2t}$$

where μ is the expected rate of return, W_{1t} and W_{2t} are two standard Brownian motions so that:

$$Cov(dW_{1t}, dW_{2t}) = \rho dt$$

where ρ is the constant correlation coefficient between W_{1t} and W_{2t}

We also assume the continuously compounded rate of interest is a constant r

To make the algebra easier we let: $V_t = \sigma_t^2$

Analysis $\mathbb{P}$-World

Applying Ito's Lemma to $V_t = \sigma_2^2 = f(\sigma_t)$ where $f(x) = x^2$ gives:

$$dV_t = \kappa(\theta - V_t)dt + \gamma\sqrt{V_t}dW_{2t}$$

where:

- $\theta = \frac{\beta^2}{2\alpha}$ is the long term mean of the variance
- $\kappa = 2\alpha$ is the mean-reverting speed of the variance
- $\gamma = 2\beta$ is the volatility of the variance

<u>Proof</u>

$$df(t, X_t) = \left[\frac{\partial f}{\partial t} + \mu(t, X_t)\frac{\partial f}{\partial x} + \frac{1}{2}\sigma^2(t, X_t)\frac{\partial^2 f}{\partial x^2}\right]dt + \sigma(t, X_t)\frac{\partial f}{\partial x}dWt$$

where

$dX_t = \mu dt + \sigma dW_t$

But now we have

$d\sigma_t = -\alpha\sigma_t dt + \beta dW_{2t}$

so we re-write Ito's Lemma for σ rather than X

$$df(t, \sigma_t) = \left[\frac{\partial f}{\partial t} + -\alpha\sigma_t\frac{\partial f}{\partial \sigma} + \frac{1}{2}\beta^2\frac{\partial^2 f}{\partial \sigma^2}\right]dt + \beta\frac{\partial f}{\partial \sigma}dWt$$

We want to get to dV_t and $V_t = \sigma_t^2$ so we use $f(x) = x^2$

So $\frac{\partial f}{\partial t} = 0$, $\frac{\partial f}{\partial x} = 2x$ and $\frac{\partial^2 f}{\partial x^2} = 2$

So Ito's Lemma becomes

$$dV_t = d\sigma_t^2 = df(\sigma_t) = \left[0 - \alpha\sigma_t.\, 2\sigma_t + \frac{1}{2}\beta^2.\, 2\right]dt + \beta.2\sigma.\, dWt$$

$$dV_t = \left[\beta^2 - 2\alpha\sigma_t^2\right]dt + 2\beta\sigma.\,dWt$$

$$dV_t = 2\alpha\left[\frac{\beta^2}{2\alpha} - \sigma_t^2\right]dt + 2\beta\sigma.\,dWt$$

$$dV_t = 2\alpha\left[\frac{\beta^2}{2\alpha} - V_t\right]dt + 2\beta\sqrt{V_t}.\,dWt$$

Defining $x_t = lnS_t$, we see that by Ito's Lemma

$$dx_t = \left(\mu - \frac{1}{2}V_t\right)dt + \sqrt{V_t}dW_{1t}$$

The Cholesky Decomposition gives us that:

$$W_{2t} = \rho W_{1t} + \sqrt{1-\rho^2}W_{0t}$$

where W_{0t} and W_{1t} are two independent Brownian motions under P. This gives us the correlation of ρ between W_{1t} and W_{2t}

<u>Check</u>

$$cov(W_{1t}, W_{2t}) = cov(W_{1t}, \rho W_{1t} + \sqrt{1-\rho^2}W_{0t}),$$

$$cov(W_{1t}, W_{2t}) = cov(W_{1t}, \rho W_{1t}) + cov(W_{1t}, \sqrt{1-\rho^2}W_{0t}),$$

$$cov(W_{1t}, W_{2t}) = \rho \times var(W_{1t}) + \sqrt{1-\rho^2} \times cov(W_{1t}, W_{0t}),$$

$$cov(W_{1t}, W_{2t}) = \rho,$$

We can now rewrite the HSV as follows:

$$dx_t = \left(\mu - \frac{1}{2}V_t\right)dt + \sqrt{V_t}dW_{1t}$$

$$dV_t = \kappa(\theta - V_t)dt + \rho\gamma\sqrt{V_t}dW_{1t} + \sqrt{1-\rho^2}\gamma\sqrt{V_t}dW_{0t}$$

Analysis Q-World

So far this is all well and good. Now we need to map our stochastic volatility process into Q world

<u>Let us start with a Cameron Martin Girsanov review</u>

If W_t is a S.B.M. under P and γ_t is a previsible process then:

$\tilde{W}_t = W_t + \int_0^t \gamma_s ds$ is a S.B.M under Q where Q is defined by:

$$\frac{dQ}{dP} = exp\left(-\int_0^T \gamma_t dW_t - \frac{1}{2}\int_0^T \gamma_t^2 dt\right)$$

This is far more powerful than the special case we actually need. In practice what we do is to chose the γ_t to take out the market price of risk and so Q becomes the risk-neutral probability measure

We start with the following inspired definition:

$$\eta_t = \left(\frac{\gamma\rho(\mu-r)+\lambda(t,S_t,V_t)}{\gamma\sqrt{1-\rho^2}\sqrt{V_t}}, \frac{\mu-r}{\sqrt{V_t}}\right)' \in R^2$$

where $\lambda(t, S_t, V_t)$ is the market price of volatility risk

we now write $\mathbf{W}_t = (W_{0t}, W_{1t})' \in R^2$ for convenience and define Λ_t by

$$\Lambda_t = \exp\left(\int_0^t \eta_u' d\mathbf{W}_u - \frac{1}{2}\int_0^t ||\eta_u||^2 du\right)$$

Then Λ_t is a martingale, given the way it is defined and this is largely irrespective of η_t providing the Novikov condition is satisfied:

$$E\left[\exp\left(\frac{1}{2}\int_0^T ||\eta_t||^2 dt\right)\right] < \infty$$

So given $E(\Lambda_0) = 1$ then $E(\Lambda_T) = 1$

The next thing we do is to define a new probability measure Q such that

$$\frac{dQ}{dP} = \Lambda_T$$

So by Girsanov's Theorem, the process $\mathbf{W}_t^*$ defined by:

$$\mathbf{W}_t^* = \mathbf{W}_t - \int_0^t \eta_u du$$

is a standard Brownian motion under Q

It may be helpful to write out $\mathbf{W}_t^*$ longhand at this point to have a more detailed look at this vector

$$W_{0t}^* = W_{0t} + \int_0^t \frac{\lambda(u, S_u, V_u)}{\gamma\sqrt{1-\rho^2}\sqrt{V_u}} du + \int_0^t \frac{\rho(\mu - r)}{\sqrt{1-\rho^2}\sqrt{V_u}} du$$

$$W_{1t}^* = W_{1t} + \int_0^t \frac{\mu - r}{\sqrt{V_u}} du$$

So W_{0t}^* and W_{1t}^* are two independent standard Brownian motions under Q

This brings us to the question of what is λ. Clearly λ is a parameter which allows us to adjust for the market's attitude to the uncertainty around the volatility of the volatility

Unlike the first order market price of risk there is no arbitrage free construction of what λ must be and so we have to observe this value from the market

Work by Breeden (1979) suggests that $\lambda(t, S_t, V_t)$ is proportional to V_t and hence equal to λV_t where λ is a constant so we will proceed on that basis from here

Ultimately this is a matter of fitting a parameter to the data, This analysis by PIMCO suggests the parameter fitting will be fairly stable

Then under $\mathbb{Q}$, the risk neutral dynamics of the logarithmic price process and the variance process are:

$$dx_t = \left(r - \frac{1}{2}V_t\right)dt + \sqrt{V_t}dW_{1t}^*$$

Check

$$dx_t = (\mu - \frac{1}{2}V_t)dt + \sqrt{V_t}dW_{1t}$$

$$W_{1t} = W_{1t}^* - \int_0^t \frac{\mu - r}{\sqrt{V_u}}du$$

$$dW_{1t} = dW_{1t}^* - \frac{\mu - r}{\sqrt{V_t}}$$

$$dx_t = (\mu - \frac{1}{2}V_t)dt + \sqrt{V_t}\left(dW_{1t}^* - \frac{\mu - r}{\sqrt{V_t}}\right)$$

$$dx_t = (r - \frac{1}{2}V_t)dt + \sqrt{V_t}dW_{1t}^*$$

$$dV_t = \kappa^*(\theta^* - V_t)dt + \rho\gamma\sqrt{V_t}dW_{1t}^* + \sqrt{1-\rho^2}\gamma\sqrt{V_t}dW_{0t}^*$$

where

$\kappa^* = \kappa + \lambda$ and

$$\theta^* = \frac{\kappa\theta}{\kappa+\lambda}$$

so κ^* and θ^* are risk neutral model parameters

The check for this line would work the same way as the dx_t check but is simply a little more complicated

We now define the process W_{2t}^* by putting:

$W^*_{2t} = \rho W^*_{1y} + \sqrt{1-\rho^2} W^*_{0t}$

It can be shown that W^*_{2t} is a Brownian motion under Q (fairly trivial as W^*_{0t} and W^*_{1t} are) and that: $Cov(dW^*_{1t}, dW^*_{2t}) = \rho dt$ under Q

So we can re-write the price and variance dynamics as:

$$dS_t = rS_t dt + \sqrt{V_t} S_t dW^*_{1t}$$

$$dV_t = \kappa^*(\theta^* - V_t)dt + \gamma\sqrt{V_t} dW^*_{2t}$$

Because we have got our x_t and V_t equations under Q-World into the same form as we had them under P-World then these equations must follow as they were established as equivalent under P-world.

In summary, so far we have managed to define our simultaneous stochastic differential equations under P and then we have recalculated them under Q. You will notice the μ in P world has become a r in Q world in particular

we now have two options for valuing an option:

- Monte Carlo
- finding a closed form solution

We start with the Monte Carlo option

The Monte Carlo Implementation

Stochastic volatility lends itself very well to Monte Carlo methods as the complexity makes it very difficult (although not impossible) to do via an analytical solution

Set up a function to take the parameters for the HSV model

Write code to generate thousands of values of W_{1t}

Using the Cholesky Decomposition write code to generate thousands of values of W_{2t}

The formulas above are under the real-world probability measure P

what do you need to do next to allow you to create arbitrage free prices for your options

Extend your function to create a single price path using a small time increment dt

Extend your function to create many prices paths under Q

Extend your function to calculate the risk neutral value of a European call option

Extend your function to calculate the value of a European call or put option

The graphical Computer program to illustrate these concepts InterestProj is here

Could you extend your code to value American options

Multiple state volatility model

The multiple state volatility model is a simple way of producing an observed feature of share prices.

? *What observed feature do you think this is*

This is because we can use a multiple state model to simulate periods of high volatility and periods of low volatility, therefore our overall share price movements will have more small movements and more very large movements than would be predicted by a simple lognormal model

Extend your model to allow for a two volatility state solution

Assume that the changes between the states are separated by exponentially distributed lengths of time

Can you now adjust the additional parameters you have σ_1, $\sigma_2 \lambda_1$ and λ_2 (the waiting time rates) to recreate the distribution of returns we can observe from the actual FTSE100 data

Complex Options

American Options

American Options cannot be simply priced by Monte Carlo methods

? *Why do you think this is the case*

It may help to review the Binomial tree method for American option pricing

CRR Binomial Tree for American Option Pricing

American Options allow the holder to exercise at any time up to and including maturity.

When you do a CRR tree you need to adapt your code so that at each node you calculate two different values:

- The value of the option assuming you do not exercise: the discounted expectation under Q
- The value of the option assuming you do exercise: simply $Max(0, S_t - K)$ for a call option

Check out the code in this spreadsheet: Binomial Model to see how American Options are handled

Remember how we worked out the payoff at the end and then for each node decide if we would have exercised at that point given the follow-on value

Clearly there is little point in being able to price American call options (on non-dividend paying shares) as they have the same price as the European Call option. We will largely just consider ***Put*** options from now on

Available Methods

There are some methods available for approximating American option prices using Monte Carlo methods.

A comprehensive paper by Caflish and Chaudary of UCLA is presented here for further reading.

I have given a simple but slow method and a surprisingly fast but more difficult method below

The Critical Decision Method

The critical decision method is based on the idea that (for simple call and put options) we can divide the decision space into (for put options):

- All share prices above Y_t would involve not exercising
- All share prices below Y_t would involve exercising

and visa versa for call options

where we need to work out what the threshold values Y_t are, for each t for which we are allowed to make a decision

Method

- Working backwards from the term T of the option, start with the last time t_{n-1} at which we could make a decision to exercise
- Calculate the value of the option assuming we do not exercise before t_{n-1} and then do exercise at t_{n-1} if the share price is below $Y_{t_{n-1}}$
- Repeat this exercise with multiple values of $Y_{t_{n-1}}$ until we find the value such that the value of the put option is a maximum
- This is then the threshold we will actually use
- Repeat whole process at time $Y_{t_{n-2}}$ and so on until we have a set of thresholds Y_{t_i} for $i = 1..n-1$
- The value of the American Put option is the value of the option with deterministic 'exercise decision' at each point t_i based on the value of Y_{t_i}

A Spreadsheet can be found here. It includes a very simple evolutionary search method

Advantages

Ultimately can be used to produce arbitrarily accurate results

Disadvantages

Extremely slow

Only uses discrete decision points, which although there can be arbitrarily many of them will make an already slow method even slower

Improvements

Using the same set of random numbers for each threshold Y_{t_i} will make comparison converge quicker

Efficient selection of Y_{t_i} will allow optimum to be found more quickly

Longstaff Schwartz Method

This method which can also be called the least squares method, can only ultimately produce an approximation but is much faster and has been tested to produce remarkably accurate results

The essence is to fit a quadratic (or some other appropriate function) to the follow-on values as you go backwards through the set of all the Monte Carlo runs

Method is as follows

- Generate N price paths of the stock
- At time $T = t_n$ it is axiomatic that you exercise the put option if the share price is less than the strike price
- So we have a full payoff profile at time t_n
- At time t_{n-1} we have a decision to make:
- We know the intrinsic value is $max(K - S_{t_{n-1}}, 0)$ but we do not know the follow-on value
- However we do know the follow-on value in the particular run of the model we are looking at
- We also know the follow-on value for all the other runs and also what the share price was at time: t_{n-1} for each of these runs
- So if we fit a function: ***follow-on value = f(current share price)*** to the data for each run of current share price and follow-on value then we can use this function to estimate our follow-on value for each run at time t_{n-1}
- We now have (for each run) an intrinsic value and a follow-on value at time t_{n-1} and so we can decide which is the greatest of the two and this becomes our payoff at time t_{n-1}
- We then go back to t_{n-2} and repeat the whole exercise

A common function for f is $f(a + b \times X + c \times X^2) = Y$ where X is the vector of share prices at time t_i and Y is the payoffs discounted back to t_i from t_{i+1}

Exotic Options

This is where we get to be a little more creative with our programming. We will go back to our basic Monte Carlo routine so as to keep the coding as simple as possible.

The things you will need to consider when programming exotic options are:

- Do we need extra variables to carry data such as running maximums?
- Do we need extra variables to carry information such as whether an option has kicked in or not?
- How do you pass to the function the type of option you want evaluated?
- How are we going to use conditional programming to ensure we make the right payoff calculations for each different type of option?

There is ultimately no limit to the amount of different options you could consider, but we will look at a few key ones here. The point is once you can program then you can change your code to reflect any particular option you want.

Single Barrier Options

A single barrier option is typically an option which pays out zero unless it has reached a certain level. Once it has reached that level then it acts as a normal option.

You can have options which are up-and-in, up-and-out, down-and-in and down-and-out.

For example if we turned our standard option into a down and out option with a knock-out point of £90 then it would behave just like any other call option unless the price of the underlying share went below £90, at any point, at which point the option would become worthless.

So you need an extra variable say `knocked_out as boolean`.

Initially you set this to false:

```
knocked_out = false
```

Then you will need to work out where to add the lines:

```
if S < 90 then
  knockout_out = true
 end if
```

What would be wrong with

```
if S < 90 then
  knockout_out = true
 else
  knocked_out = false
 end if
```

What else will you need to change

Digital Options

Whereas for a standard call option the payoff would be $Max(S_n - K, 0)$, for a digital option the payoff could be $I(S_n > K)$, that is the payoff is £1 if the share price is above the strike price and nothing otherwise.

The code for this would need to be something like:

```
  if S > K then
    payoff = 1    'or whatever payoff is
   else
    payoff = 0
   end if
```

You could also write an actual indicator function, but this would probably be overkill

```
function indicator(condition as boolean) as double
  if condition then
    indicator = 1
   else
    indicator = 0
   end if
 end function
```

Notice here how we have used the `boolean` variable - this can only take values `true` or `false`

We can also set the return value of the function in the middle of the function if we wish as is shown above

You CANNOT use the return value as an interim variable though as the function will think this is a further function call

Lookback Options

Lookback option can come in many forms, but the classic variety is for example an option where instead of the payout being $Max(S_n - K, 0)$ it might be $Max\left(Max_{0 \le i \le n}(S_i) - K, 0\right)$, so you have to "look back" over the price evolution of the share to calculate the payoff of the option.

To include lookback options you will need an extra variable which 'carries' the look back amount

For example the line:

```
if S > max_S then max_S = S
```

will use the variable max_S to store the running maximum value of the share price

? *What else do you need to do*

Adapt your monte carlo model code so that it can handle Barrier options, digital options and lookback options

Wider Applications

Monte Carlo Merton

The Merton Model is a simple extension of the Black Scholes model in which instead of modelling the value of an option with a payoff which is the amount by which the share price exceeds the strike price, we model the value of an equity and a bond where the bondholder receives all of the enterprise value up to the redemption payment amount and the equity holder receives the rest

How does this equate to the Black Scholes model

- Treat the redemption payment on the bond as the strike price
- The value of the equity is then calculated as if it were the call option
- The value of the bond is the enterprise value of the firm MINUS the Black Scholes value of the equity

So we have the following equivalence

Black Scholes	Merton Model
Share price	Enterprise value
Strike price	Par value of Bond
Value of call option	Value of share
Value of share - call option	Value of bond

Interest Rates Models

There are many different models for interest rates which use simple stochastic techniques to to model different paths for interest rates in the future.

We can use these to project forward thousands of scenarios and from this calculate the the price of a bond would be if each path were the known future path of interest rates.

We then take the average of these prices and compare this with the actual bond price to calculate the 'shift' of dB_t (the real world to risk neutral measure shift)

From this we can then re-engineer a full yield curve given our model and the knowledge of the current price of the zero coupon bond

This program: interestproj.exe shows this concept graphically

Mortgage Back Securities

Mortgage back securities achieved notoriety in the credit crunch.

They are not as complicated as often thought and can easily be modelled using Monte Carlo techniques

What is a MBS

A mortgage is a set of payments that a house buyer makes to pay off the loan he took out to buy the house

If the bank receiving those payments collects them and then agrees to sell on those payments IF they are made, then different investors can buy different tranches of payments. The tranche of payments you can buy is called a mortgage back security.

If the bank receives 100 mortgage payments each month and you buy the first ten received then you have a very secure investment as it is unlikely that over 90 people will default. If you buy the last 10 then you have a very risk asset as there is a high chance that several (if not all ten) of your payments will not be received.

The e-lecture below explains the working in detail

The spreadsheet used in this e-lecture can be found here

Ruin Theory

Ruin theory is the mathematics of predicting how likely an insurance company is to become insolvent given a certain set of starting conditions and a certain claims experience

It is difficult to analyse in a closed form mathematical way and tends to be considered by Monte Carlo simulations

We would typically consider the following factors when performing a simple ruin theory calculation:

- Initial capital
- Rate of premium income
- Claim frequency
- Claim distribution

The probability of ruin can then be calculated over any given time horizon by running thousands of random simulations as in the spreadsheet below:

Ruin Theory Spreadsheet

Multiple Decrement Models - Pensions

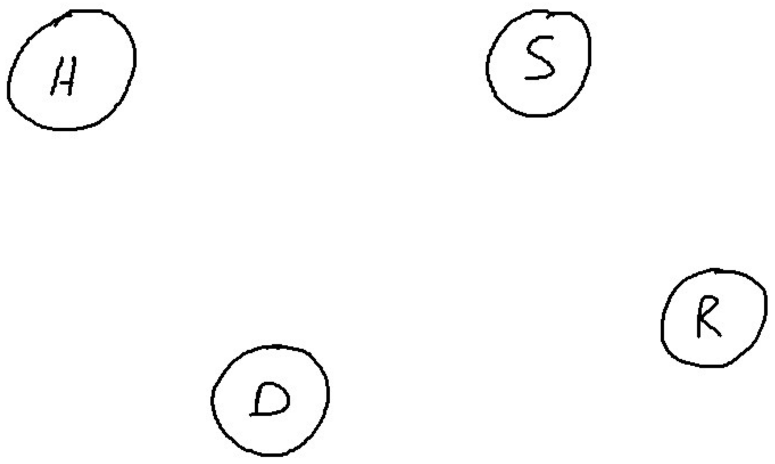
In a simple pension valuation you assume people work until they retire at say age 65 and then they receive a pension until they die

It is relatively easy to calculate the value of this as a financial liability

However in practice things will often be more complicated due to other states each pension scheme member might be in

For example a multiple state model can include sickness as well as death and retirement

The 'state space' may then appear as follows:



0 1 2 3 4

The normal state is healthy. However the life can also be Dead or Sick or Retired

A simple mortality model would have a transition rate to death

But a more sophisticated model would have transitions to sick and back (recovery) and also transitions from sick to dead at a different rate from healthy to dead

We will ignore retirement for the time being and just analyse the multiple state model

The simple two state model: healthy and dead is simple to analyse analytically for a constant μ as it is simple exponential decay

However there is no closed form solution to the multiple state model and so we can use a Monte Carlo solution

Problem:

The sponsoring company wishes to know the cost of providing a sickness benefit to employees of £50 per day while they are sick. Develop a Monte Carlo model to price the value of this benefit given parameters of the transition rates μ, σ, ρ and λ as defined in the diagram

Hint: each transition time will be distributed exponentially

Solution is Healthy Sick Dead Model.xls

Parameterisation

Method of Moments

Recall that the method of moments involves calculating sample statistics such as the mean and variance of an actual distribution and then fitting this to a known analytic distribution

Example

You have a set of data with a mean of 23 and a standard deviation of 12. You believe the data comes from a lognormal distribution: $lnN(\mu, \sigma^2)$. calculate μ and σ using the method of moments

Fitting the Lognormal Distribution to Price Data

Price data is not a simple set of values from a distribution.

There are a number of issues we need to be careful about:

- We would naturally assume it to be geometric
- The distribution should describe the price changes not the price
- We will typically have values on M,T,W,T and F but not at the weekends

? *How do we solve these problems*

- First we need to take logs of our data
- Then we take differences

We now have data we would expect to fit a normal distribution

We still have to solve the weekend problem

We could

- Make the assumption that the weekend is a three day period and hence multiply both the mean and variance of our distribution by 3
- Make the assumption that as market's are closed we can treat Friday to Monday as one day

Both assumptions are assumptions and it makes more sense to actually test the validity of either using the data that we have

Essentially we are dealing with a composite process which combines a one day process with a three day process

By viewing the spreadsheet: Fitting volatility to weekly data.xls we observe a very interesting phenomenon

SDE interpretation of the 'log difference' method

We believe our data comes from the SDE: $\frac{dS_t}{S_t} = \mu dt + \sigma dW_t$

The solution of this is: $S_t = S_0 e^{(\mu - \frac{\sigma^2}{2})t + \sigma W_t}$

Or in other words: $S_{\Delta t} = S_t e^{(\mu - \frac{\sigma^2}{2})\Delta t + \sigma W_{\Delta t}}$

$\therefore ln(S_{t+\Delta t}) - ln(S_t) = (\mu - \frac{\sigma^2}{2})\Delta t + \sigma W_{\Delta t}$

And so $var(ln(S_{t+\Delta t} - ln(S_t)) = \sigma^2 \Delta t$

Cholesky Decomposition

The Cholesky decomposition allows us to take a set of known correlations between random variables and then construct a matrix which will allow us to recreate these correlations from a set of independent random variables.

2 Random Variables

In the case of two random variables we know that X and Y are random variables drawn from $N(0,1)$ and have a correlation of ρ.

If we are now given a new random variable $X' \sim N(0,1)$ such that $corr(X, X') = 0$ then how do we derive Y such that $corr(X, Y) = \rho$

In other words what is the matrix $\begin{bmatrix} a & b \\ c & d \end{bmatrix}$ such that

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix} \begin{bmatrix} X \\ X' \end{bmatrix} = \begin{bmatrix} X \\ Y \end{bmatrix}$$

We have $X = aX + bX'$ and $Y = cX + dX'$ $\therefore a = 1, b = 0$

So $cov(X,Y) = a.c.var(X) + d.b.var(X') + (a.d + b.c)cov(X,X')$

$\therefore cov(X,Y) = a.c + d.b = c$

$\therefore c = \rho$

Given $var(Y) = 1$ we have that $var(cX + dX') = 1$

$\therefore c^2Var(X) + d^2var(X') = 1$

$\therefore \rho^2 + d^2 = 1$

$\therefore d = \sqrt{1-\rho^2}$

So our Cholesky decomposition is:

$$\begin{bmatrix} X \\ Y \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ \rho & \sqrt{1-\rho^2} \end{bmatrix} \begin{bmatrix} X \\ X' \end{bmatrix}$$

What is $L \times L^T$

If we now multiply our matrix by its transpose we observe the following:

$$\begin{bmatrix} 1 & 0 \\ \rho & \sqrt{1-\rho^2} \end{bmatrix} \times \begin{bmatrix} 1 & \rho \\ 0 & \sqrt{1-\rho^2} \end{bmatrix} = \begin{bmatrix} 1 & \rho \\ \rho & 1 \end{bmatrix}$$

We notice that we have reconstructed the correlation matrix of the original relationship between the random variables X and Y

Generalising the triangular matrix

We can extend this logic to more than 2 random variables

For example for three random variables the Cholesky decomposition is:

$$\begin{bmatrix} 1 & \rho_{1,2} & \rho_{1,3} \\ \rho_{1,2} & 1 & \rho_{2,3} \\ \rho_{1,3} & \rho_{2,3} & 1 \end{bmatrix} = \begin{bmatrix} t_{1,1} & 0 & 0 \\ t_{1,2} & t_{2,2} & 0 \\ t_{1,3} & t_{2,3} & t_{3,3} \end{bmatrix} \times \begin{bmatrix} t_{1,1} & t_{1,2} & t_{1,3} \\ 0 & t_{2,2} & t_{2,3} \\ 0 & 0 & t_{3,3} \end{bmatrix}$$

Solving gives us:

$$\begin{bmatrix} t_{1,1} & 0 & 0 \\ t_{1,2} & t_{2,2} & 0 \\ t_{1,3} & t_{2,3} & t_{3,3} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ \rho_{1,2} & \sqrt{1-\rho_{1,2}^2} & 0 \\ \rho_{1,3} & \frac{\rho_{2,3}-\rho_{1,2}\rho_{1,3}}{\sqrt{1-\rho_{1,2}^2}} & \sqrt{1-\rho_{1,3}^2-\frac{(\rho_{2,3}-\rho_{1,2}\rho_{1,3})^2}{1-\rho_{1,2}^2}} \end{bmatrix}$$

This may look complicated but if this system of 9 equations in 9 unknowns reduces naturally one equation at a time and and is very easy to solve.

A numerical algorithm is easy to implement

Example

A correlation matrix between A, B and C is given by $\begin{bmatrix} 1 & 0.6 & 0.4 \\ 0.6 & 1 & 0.5 \\ 0.4 & 0.5 & 1 \end{bmatrix}$

Show how you would generate the random varaibles A, B and C from uniform independent random variables U_1 , U_2 and U_3

Curve Fitting and Least Squares Regression

If you wish to fit a function of several parameters to a set of data - this can be a very computationally intensive task

If there is only one parameter then using goal seek in Excel will probably be perfectly serviceable, but you cannot use this method if you have multiple parameters

First you need to decide what is meant by best fit. For most purposes minimising the square of the deviation between the actual data and the fitted data will be a sensible metric, hence the expression "Least squares Regression"

Then we need to decide how to search the vector space of all possible parameter combinations for the one for which the sum of the squares of the deviations is the least

We list below a number of methods in order of increasing sophistication.

Simple grid search

We assume:

- there are n parameters
- the sum least squares function is denoted by f
- the current set of parameters is stored in a vector denoted by P
- Calculate f at every point in a n dimensional grid and then select the point in the grid for which f is least. This is the set of parameters P which gives the best fit

Orthogonal Grid Search

As per the simple grid search but:

- Minimise for parameter 1, keeping the other parameters constant, then do the same for parameter 2 and so on, starting again with parameter 1 and repeating until the function cannot be reduced by changing any further parameters

Steepest Decent Method

- Calculate the gradient vector at P_0 and then minimise in the steepest direction. Once the minimum has been found in this direction recalculate the gradient and repeat the process until you reach a minimum in all directions

Methods 2 and 3 both require us to be able to minimise along an individual line within our hyperspace

Line minimisation

Which ever method we use (apart from the grid) we will need to minimise our function along any particular line in the vector space of the parameters

Line minimisation is analogous to Newton-Rhapson root finding although we are not solving $f(x) = 0$ but rather we are solving $f'(x) = 0$ i.e. the point which the function (least squares deviation) is at a minimum

For Newton Rhapson the assumption we made was of local linearity - sop this time the assumption we make is that the function is locally quadratic

We can use the first and second derivatives if we have them (a vector and matrix respectively) but if not a simple numerical procedure will suffice

calculate $f_0 = f(\mathbf{P}), f_1 + f(\mathbf{P} + h\mathbf{x}), f_2 = f(\mathbf{P} + 2h\mathbf{x})$, where $\mathbf{x}$ is the unit vector in parameter space of the direction in which we are minimising and h is a small increment

Then $f_0'' = \frac{(f_2 - f_1) - (f_1 - f_0)}{h^2}$ is the rate at which the (negative) gradient is getting shallower and so given $f_0' = \frac{(f_1 - f_0)}{h}$ is the current gradient in the direction $\mathbf{x}$ then our initial line minimisation guess is $P_{n+1} = P_n - \mathbf{x} \times \frac{f_0'}{f_0''}$

We then repeat this process along the same line until we have found our one dimensional minimum - at which point we continue with whatever multi-dimensional method we were using

Convergence Speed

On first inspection the second and third methods would appear to be quite good methods, but in practice they both tend to zig zag very slowly towards a solution and a much more efficient method is found by the method of conjugate gradients

Conjugate Gradient Method

The conjugate gradient method is as follows:

Step 1: Initialise the set of directions $\mathbf{u}_i$ to the basis vectors $\mathbf{e}_i$

Step 2: Save your starting position as $\mathbf{P}_0$

Step 3: For $i = 1, \ldots, N$, move $\mathbf{P}_{i-1}$ to the minimum along direction $\mathbf{u}_i$ and call this point $\mathbf{P}_i$

Step 4: For $i = 1, \ldots, N$, set $\mathbf{u}_i$ to be $\mathbf{u}_{i+1}$

Step 5: Set $\mathbf{u}_N$ to be $\mathbf{P}_N - \mathbf{P}_0$

Step 6: Move $\mathbf{P}_N$ to the minimum along the direction $\mathbf{u}_N$ and call this point $\mathbf{P}_0$

Step 7: Repeat this basic algorithm N times and any quadratic form will be minimised

The result in step 7 was proved by Powell in 1964

This spreadsheet Powell.xls contains an implementation of the method

Bootstrapping

Simple Stock Returns

The essence of bootstrapping is that we don't know what the distribution of stock returns is, so we simply use historic stock returns as our sample space

This spreadsheet gives FTSE100 data

Use it to develop a Monte carlo method which prices an option on the FTSE100

Method

- Pick a random day in the history of the data you have
- Take the market return for that day
- Apply this to your current simulated share price
- Repeat for a long as you are simulating the share price behaviour
- Run thousands of times to perform your Monte Carlo calculation

Issues

❓ *Can you think of a problem with this method*

❓ *Can you think of solutions*

Making Bootstrapping more Realistic

We are going to assess the log returns of the FTSE100 data over 1 year periods.

We only have 34 data points but we can still measure the skew and kurtosis of the returns from this amount of data.

We need the following formulas

$$\text{Sample Excess Kurtosis} = \frac{\frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})^4}{\left(\frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})^2\right)^2} - 3$$

$$\text{Sample Skewness} = \frac{\frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})^3}{\left[\frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})^2\right]^{3/2}}$$

Note: Ideally we would use the unbiased estimators at this point but given the reasonably large sample and the intrinsic lack of accuracy in this process the pure sample statistics will probably suffice

Exercise: Write VBA functions to calculate the mean standard deviation, skewness and kurtosis of a data set

Random Walk in Share Price History

This time instead of taking random days from the past we start with one random day from the past and then randomly choose an adjacent day

We continue this process taking the price moves on successive adjacent days either going forward of backwards

By running this simulation thousand of times you can measure the skew and kurtosis of the distribution so generated

Using a Longer Period

This time we will use randomly chosen days in the past but then take the return over a one month period.

This model will run quicker as as we onlty need 12 return values per one year simulation

Can you see a problem with this method

Can you also change your model so that you use whole year segments of the share price return

Ratioing up one days returns

This time we randomly choose days in the past and then ratio up the return from that day by an appropriate amount to simulate the return for a whole year

We could also do this for 12 separate months or even take a monthly return and ratio up by $\sqrt{12}$

What are we trying to achieve

Each time we perform the bootstrapping in a different way we are trying to recreate the features of the actual share price return distribution.

For each method there are a number of different parameters we can change and it makes sense to try out different methods until we have a realistic simulation of the actual share we are modelling (in our case the whole FTSE100

Generating Functions

Generating Functions

Generating functions are a strange concept when you first come across them but they have many applications and often make otherwise complicated calculations much more straighforward

Probability generating functions

The probability generating function of a random variable X is
$G(t) = E\left(t^X\right)$

For a discrete random variable (on the integers) this is:

$$G(t) = \sum_{x=-\infty}^{\infty} p(x)t^x$$

For a continuous random variable this is: $G(t) = \int\limits_{-\infty}^{\infty} f(x)t^x dx$, where $f(x)$ is the probability density function.

What is the probability generating function which describes a dice roll?

Now calculate $\left(G(t)\right)^2$

Can you see the practical application that probability generating functions might have?

Moment generating functions

The moment generating function of a random variable X is
$M(t) = E\left(e^{tX}\right)$

For a continuous random variable this is: $M(t) = \int\limits_{-\infty}^{\infty} f(x)e^{tx}dx$, where $f(x)$ is the probability density function.

Calculate the moment generating function for the the normal distribution:

Proof can be found here

Now differentiate $M_X(t)$ and find $\frac{dM_X(0)}{dt}$ and $\frac{d^2M_X(0)}{dt^2}$ again for $N(\mu, \sigma^2)$

proof (first moment)

proof (second moment)

Characteristic Functions

The characterisitc function is very closely related to the moment generating function in that it is defined as

$$\varphi_X(t) = E\left(e^{itX}\right)$$

It is the ***Fourier transform*** of the probability density function and as such is particularly important

Cumulant generating functions

The cumulant generating function of a random variable X is
$K(\theta) = logE\left(e^{\theta X}\right)$

We can see that for $X \sim N(\mu, \sigma^2), K_X(\theta) = \theta\mu + \frac{1}{2}\sigma^2\theta^2$

So $K'(\theta) = \mu + \sigma^2\theta, K''(\theta) = \sigma^2$ and $K'''(\theta) = 0$

The cumulants are $k_1 = \mu, k_2 = \sigma^2, k_n = 0$ for $n \geqslant 3$

How do you think the cumulants are defined

This allows us to connect the cumulant generating functions with a Maclaurin expansion and write:

$$K(\theta) = \frac{\kappa_1\theta^1}{1!} + \frac{\kappa_2\theta^2}{2!} + \frac{\kappa_3\theta^3}{3!} + \frac{\kappa_4\theta^4}{4!} + \ldots$$

The dummy variable t has been replaced by θ in the above to avoid confusion with time

Fourier Analysis

Quadratures

Introduction

A quadrature is just a name we use to describe methods of numerical integration

In this chapter we will look at some simple methods such as the trapezium rule, which you will already be familiar with and then we will consider some more sophisticated developments, such as Simpson's and Boole's rule, which interpolate function points with polynomials of increasing degree

Finally we will look at Gaussian quadratures where the assumption of evenly spaced abscissa points is dropped and a more general form of quadrature is developed which can produce remarkably accurate results with relatively little calculation

Very basic quadrature methods

The most basic form of numerical integration is called the left point rule: In this case we simply divide the function into intervals of width Δx and then add up the area of the rectangles width: Δx and height: $f(x_i)$, where x_i is the value of x at the left of the interval

This gives: $\int_a^b f(x)dx \approx \Delta x \times (f(x_0) + f(x_1) + \ldots + f(x_{n-1}))$

where $a = x_0$, $b = x_n$ and the x_is are evenly spaced across the interval.

The right point rule and the mid-point rule follow by analogy

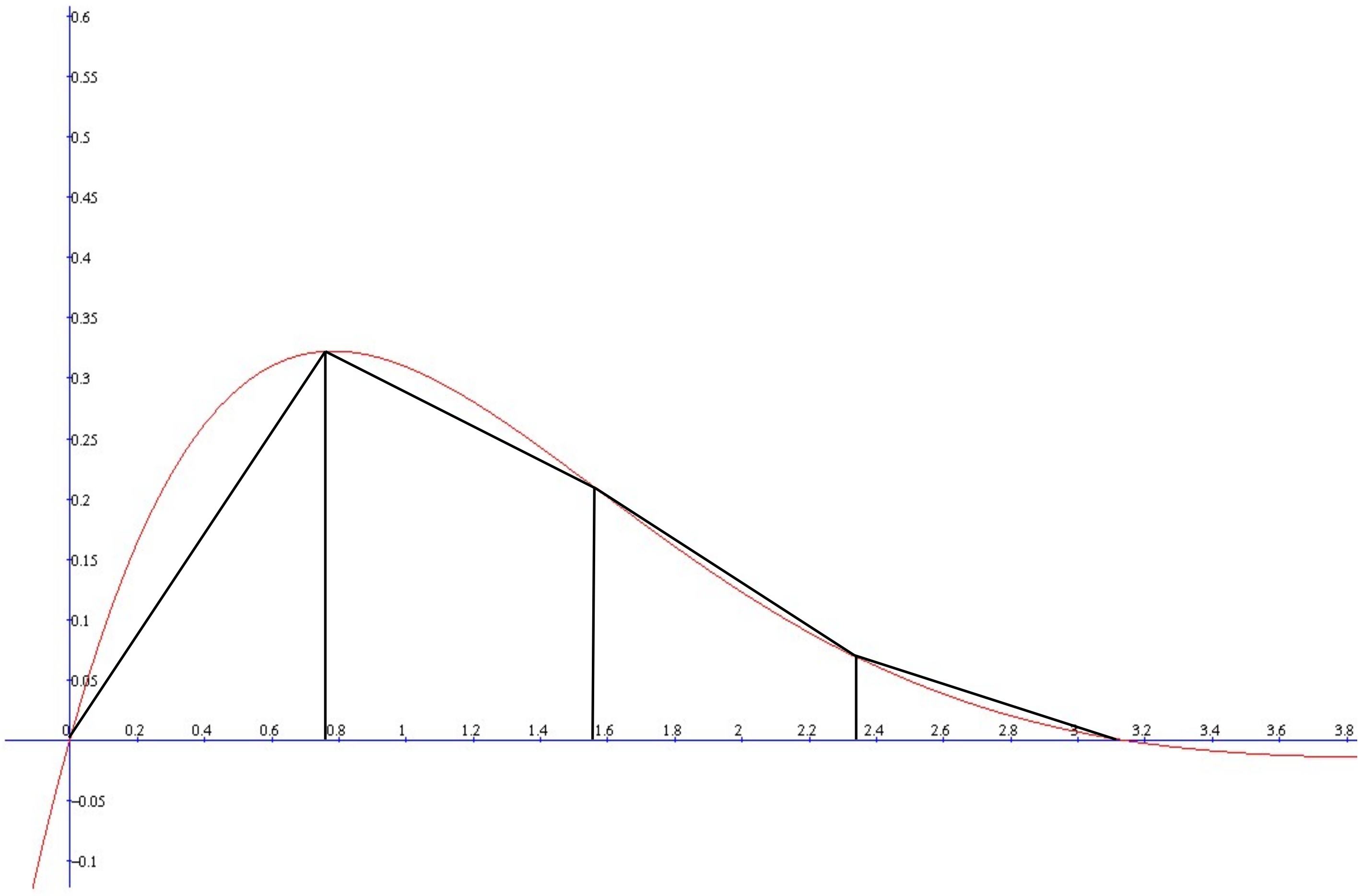
Clearly these methods are very primitive and it is easy to improve on them

The Trapezium Rule

The trapezium rule represents the first level of sophistication in the numerical calculation of integrals

consider the problem of how to calculate the area under the curve:
$y = e^{-x} \times sinx$ between 0 and π

The following graph shows how we could use a series of trapeziums to estimate this integral



We can easily see that the area under the trapeziums reduces to:

$$A = \frac{\pi}{8} \times \left(f(0) + 2f(\tfrac{\pi}{4}) + 2f(\tfrac{\pi}{2}) + 2f(\tfrac{3\pi}{4}) + f(\pi)\right)$$

So more generally the trapezium rule is given by:

$$\int_a^b f(x)dx \approx \frac{\Delta x}{2} \times (f(x_0) + 2f(x_1) + 2f(x_2) + \ldots + 2f(x_{n-1}) + f(x_n))$$

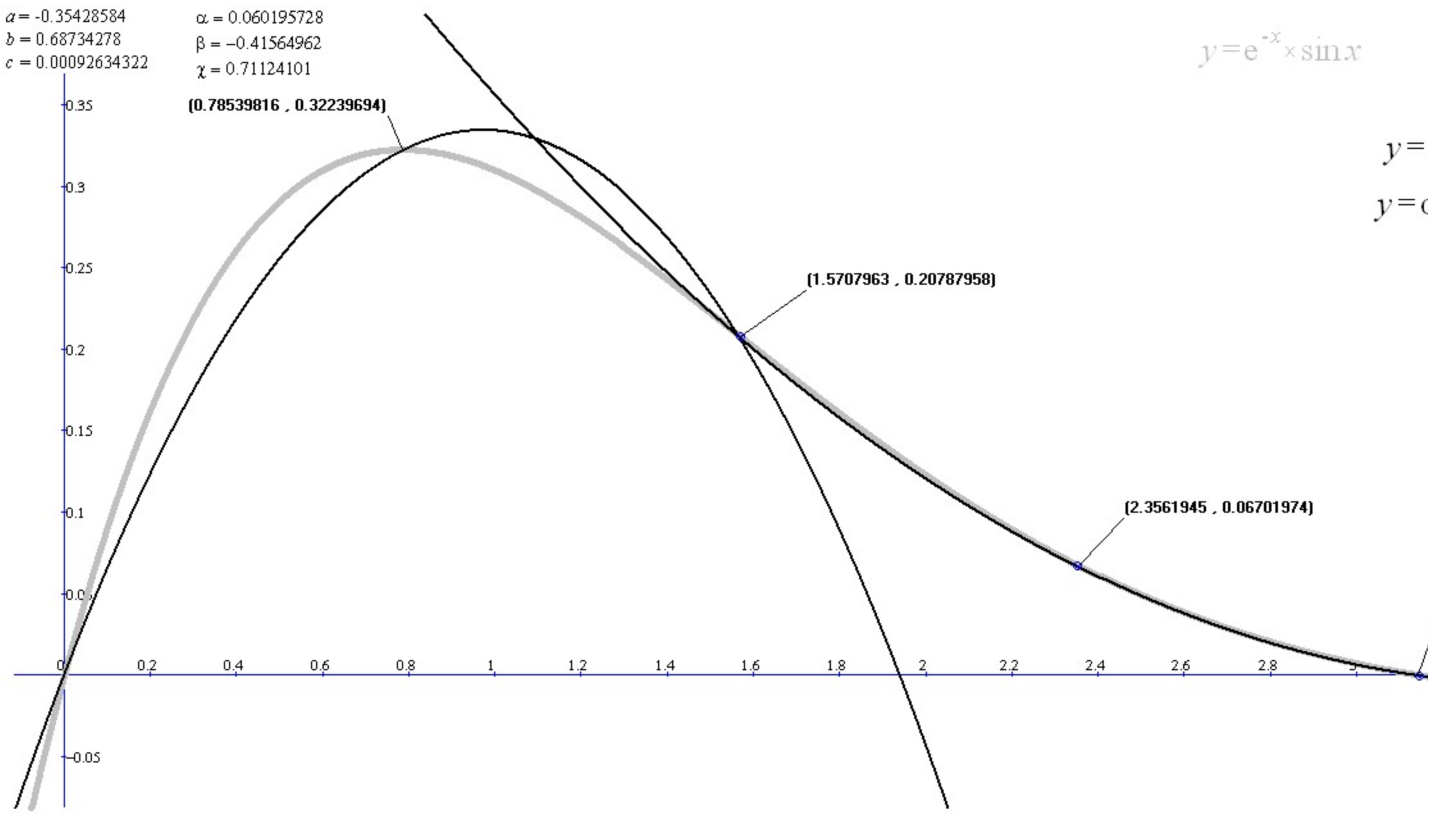
Write a VBA routine to approximate the integral of a general function between abscissas a and b with any given number of intervals using the trapezium rule

Approximate the area under $y = e^{-x} \times sinx$ between 0 and π using 4, 12 and 120 intervals

Simpson's Rule

Suppose instead of fitting straight line segments to our function we attempt to find a better fit by fitting quadratic segments

Look at the same function below (grey) with two black quadratics fitted to the first three and last three calculated function points



By considering a function f on the interval $[-\theta,\theta]$ and using abscissa points $-\theta$, 0 and θ, calculate the quadratic which will interpolate the three points

Using basic calculus, calculate the area under this quadratic on the interval $[-\theta,\theta]$ as a function of θ, $f(-\theta)$, $f(0)$ and $f(\theta)$

This leads us to Simpson's Rule which is stated generally as:

$$\int_a^b f(x)dx \approx \frac{\Delta x}{3} \times (f(x_0) + 4f(x_1) + 2f(x_2) + 4f(x_3) + \ldots + 2f(x_{n-2}) + 4f(x_{n-1}) + f(x_n))$$

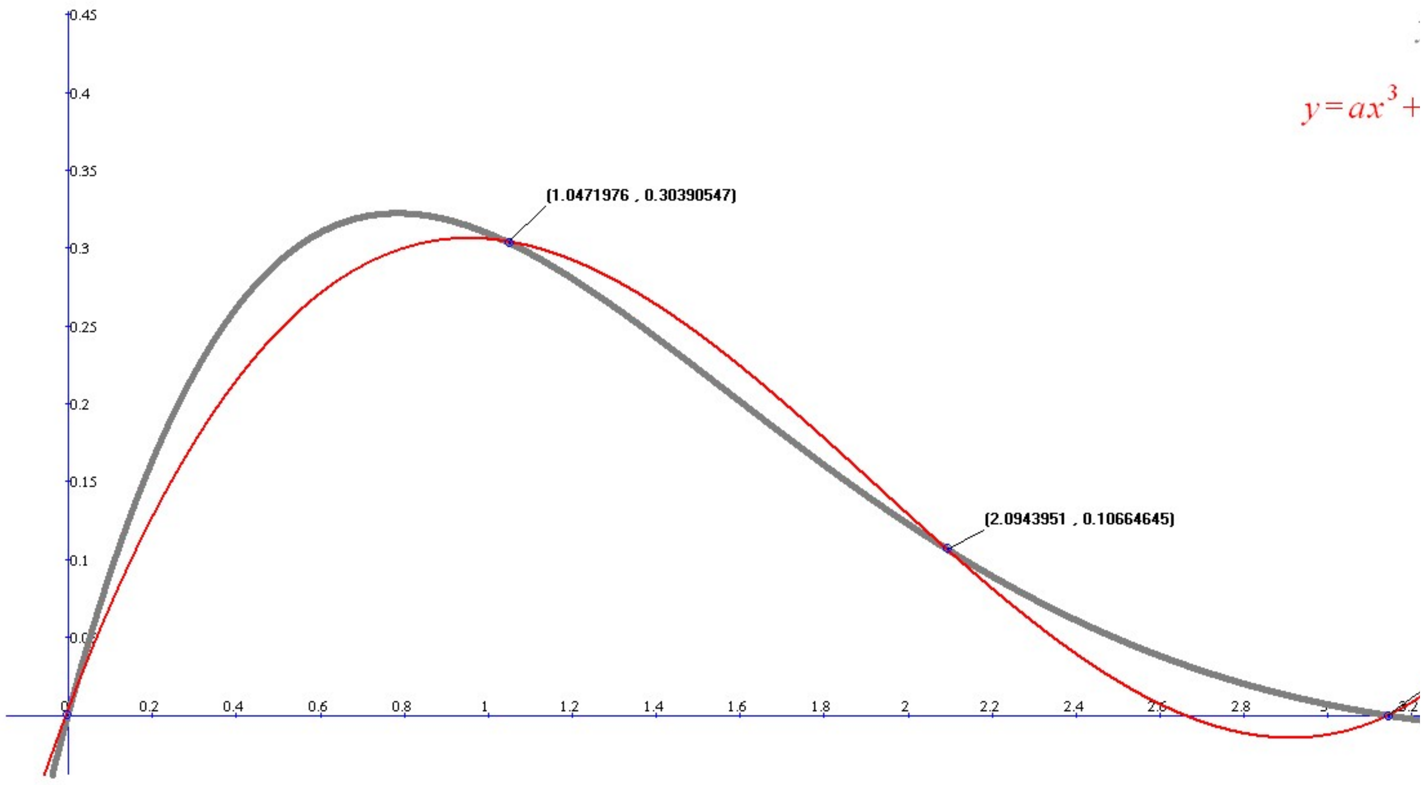
Write a VBA routine to approximate the integral of a general function between abscissas a and b with any given number of intervals using Simpson's Rule

Approximate the area under $y = e^{-x} \times sinx$ between 0 and π using 4, 12 and 120 intervals using Simpson's Rule (2,6 and 60 quadratic segments)

Simpson's three eighths Rule

The next level is to fit cubic segments

Look at the same function again (grey) with the red cubic fitted to four calculated function points



The principles are exactly the same but the algebra is a bit more complicated, so Simpson's 3/8ths Rule over a $3\Delta x$ interval can be expressed as:

$$\int_a^b f(x)dx \approx \frac{3\Delta x}{8} \times (f(x_0) + 3f(x_1) + 3f(x_2) + f(x_3))$$

Write a single VBA routine to approximate the integral of a general function between abscissas a and b with any given number of intervals and allowing the user to choose between the methods given so far

Approximate the area under $y = e^{-x} \times sinx$ between 0 and π using 3, 6, 12 and 120 intervals using Simpson's 3/8ths Rule

Boole's Rule

The next level is to fit quartic segments

This time the interpolating function is a quartic

The principles are again the same, so Boole's Rule over $4\Delta x$ interval can be expressed as:

$$\int_a^b f(x)dx \approx \frac{\Delta x}{45} \times (14f(x_0) + 64f(x_1) + 24f(x_2) + 64f(x_3) + 14f(x_4))$$

Extend your VBA routine to include Boole's Rule

Approximate the area under $y = e^{-x} \times sinx$ between 0 and π using 4, 12 and 120 intervals using Boole's Rule

Programming Extensions

There are a number of things you should now do to tidy up your routine

- Ensure the routine checks for an appropriate number of intervals for each integration method
- You could splice different methods where n does not fit
- You could include an option to leave out n and let the routine try different values until some convergence criterion is reached

Gaussian Quadratures

So far all of our methods have assumed that the abscissa points have been equally spaced along the x-axis

We could have changed this if we wanted but there seemed little point and it would have made the algebra more complicated

The question therefore arises: Can we improve the speed or accuracy of our routine by using a different set of abscissa points?

General Considerations

In general all quadrature methods are based on the formula:

$$\int_a^b f(x)dx \approx \sum_{i=1}^{n} \omega_i f(x_i)$$

Where ω_i are a set of weights and x_i are a set of abscissa points

As we are now free to choose $2n$ values we should be able to approximate our integral much more accurately - in fact if we motivate our selection of ω_i and x_i by fitting polynomials then this method will calculate the integral of an order $2n-1$ polynomial exactly

2 Point Gaussian Quadratures

W.l.o.g we work on the interval $[-1, 1]$

We wish to find ω_1, ω_2, x_1 and x_2 such that

$\int_{-1}^{1} f(x)dx \equiv \sum_{i=1}^{2} \omega_i f(x_i)$, for a cubic $f(x)$, i.e.:

$\int_{-1}^{1} a_0 + a_1 x + a_2 x^2 + x_3 x^3 dx \equiv \omega_1 f(x_1) + \omega_2 f(x_2)$, i.e. this relation must hold for all choices of a_i

We proceed with basic calculus

$$\left[a_0 x + \frac{a_1 x^2}{2} + \frac{a_2 x^3}{3} + \frac{x_3 x^4}{4}\right]_{-1}^{1} \equiv \omega_1 f(x_1) + \omega_2 f(x_2)$$

$$a_0(1-(-1)) + a_1 \frac{1^2-(-1)^2}{2} + a_2 \frac{1^3-(-1)^3}{3} + a_3 \frac{1^4-(-1)^4}{4} \equiv \omega_1 f(x_1) + \omega_2 f(x_2)$$

$$2a_0 + \tfrac{2}{3}a_2 \equiv \omega_1 \left(a_0 + a_1 x_1 + a_2 x_1^2 + a_3 x_1^3\right) + \omega_2 \left(a_0 + a_1 x_2 + a_2 x_2^2 + a_3 x_2^3\right)$$

$$2a_0 + \tfrac{2}{3}a_2 \equiv a_0(\omega_1 + \omega_2) + a_1(\omega_1 x_1 + \omega_2 x_2) + a_2(\omega_1 x_1^2 + \omega_2 x_2^2) + a_3(\omega_1 x_1^3 + \omega_2 x_2^3)$$

As this is an identity we can match the co-efficients of the a_i

$a_0 : 2 = \omega_1 + \omega_2$

$a_1 : 0 = \omega_1 x_1 + \omega_2 x_2$

$a_2 : \frac{2}{3} = \omega_1 x_1^2 + \omega_2 x_2^2$

$a_3 : 0 = \omega_1 x_1^3 + \omega_2 x_2^3$

The best way to solve these equations is guess and check as follows:

as symmetry seems likely assume $\omega_1 = \omega_2$ therefore $\omega_1 = \omega_2 = 1$

a_1 gives $x_1 = -x_2$

a_2 gives $\frac{2}{3} = \omega_1 x_1^2 + \omega_2 x_2^2 = 2x_1^2$

so $3x_1^2 - 1 = 0$ - this is the Legendre Polynomial order 2

w.l.o.g. $x_1 = -\sqrt{\frac{1}{3}}$ and $x_2 = \sqrt{\frac{1}{3}}$

Calculate $\int_{-1}^{1} 2 + 6x - 3x^2 + x^3 dx$, both analytically and using 2 point Gaussian quadrature

The last thing we need to do to make our method useful is to drop the requirement to work on the $[-1, 1]$ interval

There are two additional things we need to consider to to this:

- the ω_i need to be transposed from the [-1,1] interval to the [5,10] interval
- The area calculated at the end needs to be ratioed up from a width of 2 (1 - (-1)) to a width of 5 (10-5)

Calculate $\int_{5}^{10} 2 + 6x - 3x^2 + x^3 dx$, both analytically and using 2 point Gaussian quadrature

n Point Gaussian Quadratures

We could repeat the above exercise for 3 and then 4,5 ,6 etc points. The algebra does get complicated, but fortunately we have the following theorem:

If we perform n point quadrature by selecting the x_i to be the roots of the Legendre Polynomials and the weighting function to be $\omega_i = \frac{2}{(1-x_i^2)(P_n'(x_i))^2}$, where $P_n'(x_i)$ is the differential of the n^{th} Legendre Polynomial then our resulting quadrature will integrate polynomials of order $2n - 1$ exactly

We do not attempt to prove this result here - but the above analysis is the proof of this result for 2 point quadrature. You can see how the Legendre Polynomial arises in the above analysis

The Legendre Polynomials are defined by:

$$P_n(x) = \frac{1}{2^n} \sum_{k=0}^{n} \left(\binom{n}{k}^2 \times (x-1)^{n-k} \times (x+1)^k \right)$$

Formula 98 files are Legendre Polynomials and Legendre Polynomials wi

Calculating $P_n'(x)$ is a simple application of the product rule

So we can see that the first few of the Legendre Polynomials are:

- $n = 0 : P_0(x) = 0$
- $n = 1 : P_1(x) = x$
- $n = 2 : P_2(x) = \frac{1}{2}\left(3x^2 - 1\right)$
- $n = 3 : P_3(x) = \frac{1}{2}\left(5x^3 - 3x\right)$
- $n = 4 : P_4(x) = \frac{1}{8}\left(35x^4 - 30x^2 + 3\right)$

Produce a spreadsheet which draws graphs of each of the Legendre Polynomials

What do you notice about the dispersion of the roots?

Now compare the roots of the n-th order Legendre Polynomial with the value of $x = cos\left(\frac{(i-0.25)\pi}{n+0.5}\right)$

How might we use the formula: $x = cos\left(\frac{(i-0.25)\pi}{n+0.5}\right)$

1. Develop a VBA function which can perform an n point Gaussian quadrature over a given interval for any given function

2. change your code so that the calculation of the Legendre Polynomial roots is only calculated once, however many times the routine is called

Hint: you will need to store the ω_i and x_i values outside of the function so that they have module wide scope and use a boolean array to determine whether the values have already been calculated or not

3. Use 2, 4, 6, 8 and 10 point Gaussian quadrature to calculate

$$\int_0^{\pi} sinxdx$$

The answer spreadsheet : quadrature.xls is here

Fourier Analysis

Fourier analysis allows us to convert waveforms into spectral densities and back again

In excel draw a graph of $y = sinx$

On the same axes draw $y = sinx + \frac{sin2x}{2}$

Create a VBA function for $y = \sum_{k=1}^{n} \frac{sinkx}{k}$ and again plot on the same axes for different values of n

You can soon see that as $n \to \infty$ this function becomes a sawtooth wave

This raises the question: If we started with the sawtooth wave wave could we calculate the co-efficients

This is where fourier analysis comes in

For our purposes we will skip Fourier series and go onto the continuous Fourier transform

The fourier tranform $\hat{f}$ of an integrable function $f : \Re \to C$ is given by

$$\hat{f}(\xi) = \int_{-\infty}^{\infty} f(x) e^{2\pi i x \xi} dx$$

f is the original wave function of time x and $\hat{f}$ is the spectral density of frequency ξ

The process can be inverted by the inverse transform:

$$f(x) = \int_{-\infty}^{\infty} \hat{f}(\xi) e^{-2\pi i x \xi} d\xi$$

The above formulas look rather daunting but we start by breaking them down with Euler's formula:

$$e^{ix} = cosx + isinx$$

Write a VBA function which takes a function f, a parameter ξ, specifies bounds a and b outside of which the function will be close to zero, and returns the Fourier transform $\hat{f}(\xi)$

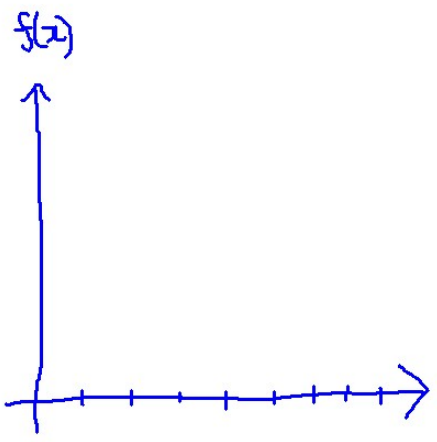
Discrete Fourier transform

Before we look at the algorithm for the Fast Fourier transform we must understand the discrete Fourier transform.

This is simply the discretised version of the Fourier transform - the numerical integration version.

Instead of performing the integral: $\hat{f}\left(\xi\right) = \int_{-\infty}^{\infty} f(x)e^{2\pi i x\xi}dx$ we perform the sum $\hat{f}\left(\xi\right) = \sum_{x=i}^{n} f(x)e^{2\pi i \frac{x}{n}\xi}$

The following diagrams illustrate how this works to produce the spectral density of the function



0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16

You should note that there are alternative ways of setting this up:

- We could go from x=1/n to 1 and leave the divide by n out
- We could have the e^{-} in the Fourier transform and the positive in the inverse transform

The principles are the same in each case

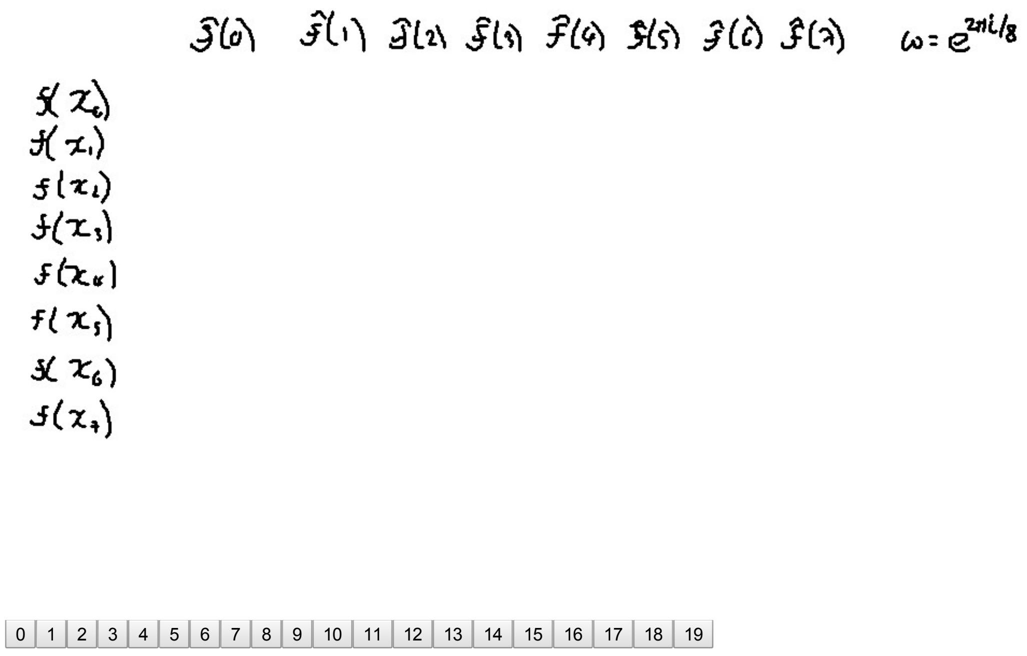
The spreadsheet DFT.xls shows how a discrete Fourier transform can be performed in excel

Fast Fourier Transform

The Fast Fourier Transform takes advantage of the fact that the discrete Fourier transform set up as above allows us to repeat the use of the $e^{2\pi i x\xi}$ and so avoid having to keep calculating a computationally expensive number

We then divide each transform into an odd and an even section and show how different transforms can be calculated from already calculated numbers.

The following slides illustrate the point



Fast Fourier transform Exercise

HARD (I will try and demonstrate before end of term)

Generate 1024 numbers in excel with some sort of periodicity

- Write a visual basic subroutine (behind a button) which takes these values into an array
- Calculate all the $e^{2\pi i \frac{x}{n}}$ values you will need, just using separate x and y values
- Use the fastest algorithm you can to calculate the Fourier transform $\hat{f}(\xi)$ for all values of $\xi \in [1..1024]$
- Output back to excel the Fourier transform you have created (the modulus of the complex numbers)
- Test your routine with different signal numbers $f(x)$

The issue here lies in how to structure the arrays to carry the required information in the FFT algorithm

Gram Charlier Method

Gram-Charlier Introduction

The purpose of the Gram-Charlier Model is to incorporate both the skewness and kurtosis of the underlying share into the option valuation

Developed by Backus, Foresi and Wu (2004)

The reason for doing this is that is we look at historic share prices we can see that the level of skew and kurtosis they exhibit is not consistent with that predicted by the lognormal model

This spreadsheet shows this feature with FTSE100 data

Key idea is to use the third and fourth moments of the cumulant generating function to include skew and kurtosis and then invert the cumulant generating functions to produce a probability density function which better approximates actual share returns and then integrate this up as in the Black Scholes model to produce a closed form valuation method for options

The shortcoming of this model is that it does not allow for variable volatility

Setup of Maths

Let S_t be the price of the underlying security at time t.

The one-period logarithmic return Y_t is given by:

$$Y_t = lnS_t - lnS_{t-1}$$

The return over T periods is

$$Y_{t+1}^T = lnS_{t+T} - lnS_t = \sum_{k=1}^{T} Y_{t+k}$$

Let $K(\theta)$ be the cumulant generating function of Y_{t+1}. Then

$$K(\theta) = \sum_{i=1}^{\infty} \kappa_i \frac{\theta^i}{i!}$$

where κ_i is the i^{th}-order cumulant of Y_{t+1}

Suppose that the one-period returns are independent and identically distributed. Then the cumulants of the sum Y_{t+1}^T are $T \times \kappa_i$, for $i = 1, 2, \ldots,$ In particular, the mean and variance of Y_{t+1}^T are respectively $\mu_T = T\mu, \sigma_T^2 = T\sigma^2$

<u>Aside</u>

Note that: $K_X(\theta) = logE(e^{\theta X})$

$$\therefore K_{X_1+X_2}(\theta) = logE\left(e^{\theta(X_1+X_2)}\right)$$

$$\therefore K_{X_1+X_2}(\theta) = logE\left(e^{\theta(X_1)} \times e^{\theta(X_2)}\right)$$

$$\therefore K_{X_1+X_2}(\theta) = log\left(E\left(e^{\theta(X_1)}\right) \times E\left(e^{\theta(X_2)}\right)\right)$$

$$\therefore K_{X_1+X_2}(\theta) = logE\left(e^{\theta(X_1)}\right) + logE\left(e^{\theta(X_1)}\right)$$

$$\therefore K_{X_1+X_2}(\theta) = K_{X_1}(\theta) + K_{X_2}(\theta)$$

And the Y_t is the log process and so we are adding these successive random variables together

It turns out that κ_3 and κ_4 etc give us the higher moments albeit indirectly:

Skew: $\gamma_1 = E\left[\left(\frac{X-\mu}{\sigma}\right)^3\right] = \frac{\kappa_3}{\kappa_2^{1.5}}$

Kurtosis: $\gamma_2 = E\left[\left(\frac{X-\mu}{\sigma}\right)^4\right] = \frac{\kappa_4}{\kappa_2^2}$

The aside above shows us that doubling the time period doubles all of the cumulants so it is easy to see that:

$$\gamma_1^T = \frac{\gamma_1}{\sqrt{T}}, \gamma_2^T = \frac{\gamma_2}{T}$$

Let $Z_T = \frac{Y_{t+1}^T - \mu_T}{\sigma_T}$, be the standardized T-period return of the underlying security. (This just keeps things a little more simple)

Using the above results and the first four terms on the Gram-Charlier expansion, Backus, Foresi and Wu (2004) gives the following probability density function for Z_T:

$$f(Z_T) = \phi(Z_T)\left(1 - \frac{\gamma_1^T}{3!}\phi^{(3)}(Z_T) + \frac{\gamma_2^T}{4!}\phi^{(4)}(Z_T)\right)$$

where $\phi(x)$ is the probability density function of the standard normal distribution and $\phi^{(i)}(x)$ is the i^{th} order derivative of $\phi(x)$

By direct differentiation,

$$\phi^{(3)}(x) = (3x - x^3)\phi(x)$$

$$\phi^{(4)}(x) = (x^4 - 6x^2 + 3)\phi(x)$$

Write a VBA function to calculate the above formula

Investigate the shape of the probability density function

Formula 98 file is here. (save as .fm2 before opening in Formula 98.

The time-t price of a European call option maturing at time T is:

$$C(t) = e^{-rT}E\left[max(S_{t+T} - K, 0)|F_t\right]$$

$$= e^{-rT}\int_{ln(K/S_t)}^{\infty}(S_t e^x - K)g(x)dx$$

where r is the risk free rate of interest, K is the strike price of the call and $g(x)$ is the probability density function of Y_{t+1}^T under a risk neutral probability measure

We now simply substitute in for $g(x)$ the probability density function we have developed above:

$$C(t) = e^{-rT}\int_{Z^*}^{\infty}(S_t e^{r_T + \sigma_T Z_T} - K)\phi(Z_T)\left(1 - \frac{\gamma_1^T}{3!}\phi^{(3)}(Z_T) + \frac{\gamma_2^T}{4!}\phi^{(4)}(Z_T)\right)dZ_T$$

where $r_T = rT$ and $Z^* = \frac{ln(K/S_t) - r_T}{\sigma_T}$

'All' that remains is to crunch through the great big integration

backus, Foresti and Wu obtained the following approximation to the call price:

$$C(t) \approx S_t\Theta(d_1) - Ke^{-rT}\Theta(d_2) + S_t\phi(d_1)\sigma_T\left(\frac{\gamma_1^T}{3!}(2\sigma_T - d_1) - \frac{\gamma_2^T}{4!}(1 - d_1^2 + 3d_1\sigma_T - 3\sigma_T^2)\right)$$

where

$$d_1 = \frac{ln(S_t/K) + (r + \sigma^2/2)T}{\sigma_T}$$

$$d_2 = d_1 - \sigma_T$$

As you can see this reduces to the Black-Scholes formula when $\gamma_1^T = \gamma_2^T = 0$

Write a VBA function to calculate European Call prices using the Gram-Charlier Model

Test you routine produces the correct answers when $\gamma_1^T = \gamma_2^T = 0$

Investigate the impact of changing the skew and kurtosis of the share price according to the model

Use put-call parity to develop a formula for the Gram-Charlier price of a put option and write a VBA function for this

Solution is Gram Charlier calculation.xls

There are obvious limitations to the Gram-Charlier model

By considering the sum of two normal distributions with different variances investigate the the skew, kurtosis and higher order moments of this distribution

Calculate the higher order cumulants of this distribution

Discuss how you would improve the Gram-Charlier model

Spreadsheet is moments test.xls