Pricing options using Monte Carlo simulations

Published on 29 Aug 13 monte-carlo options

Previously we introduced the concept of Monte Carlo simulations, and how to build a basic model that can be sampled stochastically. We're now going to expand on our modelling and show how these simulations can be applied to some financial concepts.

An option is a contract that gives the buyer the *right* to buy or sell an asset at a particular price, at a point in the future. These contracts, known as derivatives, are traded for a number of reasons, but a common usage is to hedge away exposure to an asset's price moving in an undesirable way.

The option, the right to buy or sell, has a price too. The Black Scholes model describes one way of determining a fair price for an option, but there are also many other methods for determining a price.

Options & what they're worth

A *European style* option can only be used (or *exercised*) once the option has reached a predetermined date in the future, known as its expiry date. This is referred to with the letter T .

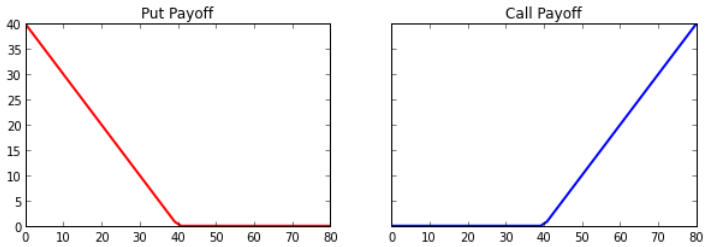
A *call* option gives the holder of the option the right to buy at a known price. A call makes money if the price of the asset at maturity, denoted by S_T , is above the *strike* price K , otherwise it's worth nothing.

$$C_T = max(0, S_T - K)$$

Similarly, a *put* option is the right to sell an asset. A put makes money when the asset is *below* the strike price at maturity, otherwise it's worth nothing.

$$P_T = max(0, K - S_T)$$

Below are the payoff graphs of a put and a call at maturity. Our asset price is the x-axis and the payoff is the y-axis.



Risk-neutral valuation

To price an option using a Monte Carlo simulation we use a risk-neutral valuation, where the fair value for a derivative is the expected value of its future payoff.

So at any date before maturity, denoted by t , the option's value is the present value of the expectation of its payoff at maturity, T .

$$C_t = PV(E[max(0, S_T - K)])$$

$$P_t = PV(E[max(0, K - S_T)])$$

Under the risk-neutral valuation we assume that underlying asset is going to earn, on average, the risk-free interest rate. So to calculate an option payoff for any time t we discount the payoff by that interest rate. Now we have a way of calculating the present value PV .

$$C_t = e^{-r(T-t)} E[max(0, S_T - K)]$$

At this point all of these variables are known, with the exception of S_T , which is what our simulation will provide.

To price our option, we're going to create a simulation that provides us with many observations for the final price of the asset S_T . By averaging all of the pay offs, we're provided with a value for our expectation of the payoff.

Simulating asset prices

The model of stock price behaviour used in the Black Scholes model assumes that we have a known volatility, we have a risk-free interest rate, and that the price of an asset follows a geometric Brownian motion.

A geometric Brownian motion is a random process where the logarithm of the random variable follows a normal distribution. This type of process distributes prices over a lognormal distribution.

So now we have a method for calculating asset prices at time T :

$$S_T = S_t e^{(r-\frac{1}{2}\sigma^2)(T-t)+\sigma\sqrt{T-t}\epsilon}$$

For this, we need to know:

- **r** is our risk free interest rate to discount by.
- σ is volatility, the annualised standard deviation of a stock's returns.
- **(T-t)** gives us the annualised time to maturity. E.g. for a 30 day option this would be $30/365 = 0.082...$
- **S** at time t . The price of the underlying asset.
- ϵ is our random value. Its distribution must be standard normal (mean of *0.0* and standard deviation of *1.0*)

Pricing options

To price an option in this simulation we: generate many prices that the asset might be at maturity, calculate option payoffs for each of those generated prices, average them, and then discount the final value.

Before creating a full simulation, we'll go through a small example of 10 runs. Suppose we have an asset with the following values: S = $100.00 and σ = 20% and we want to price a call option expiring in half a year's time, with a strike of $110.00, we also have our risk-free interest rate, which is 1%.

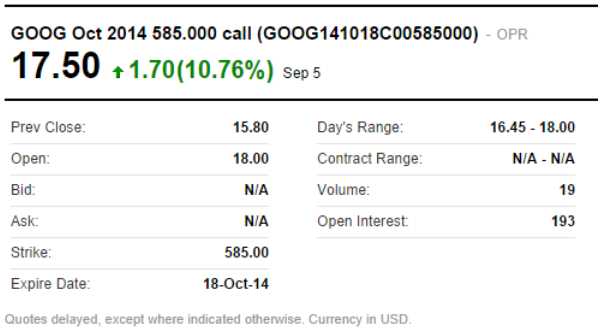
Random Var	Asset price	Payoff	Discounted payoff
1.3620	120.64	10.64	10.58
-0.7784	89.13	0.00	0.00
-0.9408	87.11	0.00	0.00
0.2227	102.69	0.00	0.00
-0.0364	98.99	0.00	0.00
-1.4303	81.28	0.00	0.00
-0.8306	88.47	0.00	0.00
1.5155	123.28	13.28	13.21
-1.5679	79.71	0.00	0.00
-1.6718	78.55	0.00	0.00

Averaging our discounted payoff values gives a price for our call option of *$2.38*. The more simulations we perform, the more accurate the price.

Now we can see how the simulation generates a price, let's build up a small python script that can price an option and see if it matches the real world. Let's look at an option on Vodafone.

Pricing real world options

In the below image we have a quote for a *call* option on Google, with a strike of *$860.00* which expires on 21 Sep 2013. We can also see the last price it traded for, *$14.50*, which gives us our target when we try and price this option.



GOOG Oct 2014 585.000 call (GOOG141018C00585000) - OPR

17.50 ↑ 1.70(10.76%) Sep 5

Prev Close:	15.80	Day's Range:	16.45 - 18.00
Open:	18.00	Contract Range:	N/A - N/A
Bid:	N/A	Volume:	19
Ask:	N/A	Open Interest:	193
Strike:	585.00		
Expire Date:	18-Oct-14		

Quotes delayed, except where indicated otherwise. Currency in USD.

What isn't specified here is the volatility, the risk-free interest rate, or the current Vodafone stock price. Volatility is a rather complex topic, so for the purposes of this article we're going to assume we know that the volatility for this particular option is *20.76%*.

Vodafone's current price can be found by from looking at a variety of sources, but at the time of writing it is *$857.29*.

For the risk-free interest rate we can use the US LIBOR rate with the same time to maturity as our option; our option expires in around three weeks, and as there is no three week rate, we'll approximate it by using the two week rate, which is currently *0.14%*.

The source code for the entire simulation will be given at the bottom, but first we'll write up how we're going to generate our asset prices.

```
def generate_asset_price(S,v,r,T):
    return S * exp((r - 0.5 * v**2) * T + v * sqrt(T) * gauss(0,1.0))
```

We know all of the input values, so we can plug them in like so:

```
S = 857.29 # underlying price
v = 0.2076 # vol of 20.76%
r = 0.0014 # rate of 0.14%
T = (datetime.date(2013,9,21) - datetime.date(2013,9,3)).days / 365.0

print generate_asset_price(S,v,r,T)
>>> 862.1783726682384
```

Now we need to be able to calculate the payoff of this generated price. Recall earlier we said that a call is worth either $S_T - K$ or 0 at expiry, we express this as a function and apply it to our generated asset price.

```
def call_payoff(S_T, K):
    return max(S_T - K, 0.0)

print call_payoff(862.18, 860)
>>> 2.1799999999
```

Complete simulation

Now let's glue this together and price our Google option.

```
import datetime
from random import gauss
from math import exp, sqrt

def generate_asset_price(S,v,r,T):
    return S * exp((r - 0.5 * v**2) * T + v * sqrt(T) * gauss(0,1.0))

def call_payoff(S_T,K):
    return max(0.0,S_T-K)

S = 857.29 # underlying price
v = 0.2076 # vol of 20.76%
r = 0.0014 # rate of 0.14%
T = (datetime.date(2013,9,21) - datetime.date(2013,9,3)).days / 365.0
K = 860.
simulations = 90000
payoffs = []
discount_factor = math.exp(-r * T)

for i in xrange(simulations):
    S_T = generate_asset_price(S,v,r,T)
    payoffs.append(
        call_payoff(S_T, K)
    )

price = discount_factor * (sum(payoffs) / float(simulations))
print 'Price: %.4f' % price
```

Which gives us the output of:

```
Price: 14.5069
```

Which matches the price we observed in the market for this Google option.

Misc

The Google option we just replicated is actually an American-style option, and we just priced it like it was a European-style option. We didn't consider the possibility that the option could be exercised early, but despite that we still arrived at the correct price.

The reason this works is that the price of an American-style call option on a non-dividend paying stock, such as Google, is the same as the price of a European-style call option. This occurs as it is not, in theory, ever optimal to exercise early when the stock doesn't pay a dividend. If the option won't ever be exercised early, its price can be calculated like a European-style option.

← Previous post
← Back to all posts
Next post →