

Valuation, Pricing of Options / Use of MATLAB

1.0 Put-Call Parity (review)

Given a European option with no dividends, let

t = current time

T = exercise time

K = strike price

r = interest rate

S = stock price at time zero

put-call parity ensures that

$$c(t;T, K) + Ke^{-rT} = p(t;T, K) + S$$

where $c(t;T, K)$, $p(t;T, K)$ are the prices of a European call and a European put option on S respectively.

Proof:

Construct two portfolios,

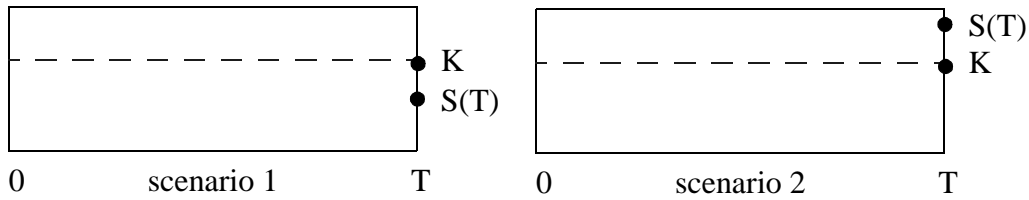
Portfolio A: 1 call + Ke^{-rT} dollars

Portfolio B: 1 put + 1 share of the underlying

$$\text{Payoff}(A) = \text{Payoff}(B) \Rightarrow \text{val}(A) = \text{val}(B) \Rightarrow \text{p-c parity}$$

no arbitrage

There are two nontrivial scenarios:



Payoff(A) = k
 Payoff(B) = K-S(T)+S(T) = K

Payoff(A) = S(T) - K + K
 Payoff(B) = S(T)

2.0 MATLAB Black-Scholes Functions: European Options

The Black-Scholes (BS) equation can be written as:

$$\frac{\partial V}{\partial t} + rS \frac{\partial V}{\partial S} + \frac{1}{2} \sigma^2 S^2 \frac{\partial^2 V}{\partial S^2} = rV \quad (2.1)$$

where $V(S, t)$ is the option price, r is the risk-free interest rate, S is the current (underlying) non-dividend paying stock price and σ is the volatility of the stock price. $\sigma^2 S^2$ can be interpreted as the instantaneous variance of S .

The derivation of this equation is based on assumptions which will be discussed in detail in the next lectures by Robert Jarrow.

The type of option under consideration determines the boundary conditions which apply to this equation. For a European option, the boundary conditions at T are given by

- $V(T, S(T)) = \max\{K - S(T), 0\}$, for a put option.
- $V(T, S(T)) = \max\{S(T) - K, 0\}$, for a call option.

Solving the BS equation is done in three different ways:

1. Explicit analytic solution (uses cumulative probability distributions, which requires an approximation or a table lookup)
2. Binomial approximation
3. PDE discretization (explicit or implicit)

The first approach, i.e., solving the BS equation exactly, is the one used in MATLAB. The other two only provide approximations, but they can be extended for option pricing in more general settings (american, exotic options).

MATLAB EXAMPLE

Verification of Put-Call Parity

The BS pricing function in MATLAB uses the following syntax:

```
[call, put] = blsprice(S0, K, R, T, SIG, Q);
```

S_0 is the current asset price, K is the exercise price, R is the risk-free interest rate,

T is the time to maturity of the option, and SIG is the volatility.

Suppose $S_0 = 100$, $K = 95$, $R = 0.1$, $T = 0.25$, $SIG = 0.5$, and $Q = 0$.

```
[c, p] = blsprice(S0, K, R, T, SIG, Q);
```

 returns

$c = 13.7$ and $p = 6.3$.

We are able to verify the put-call parity since

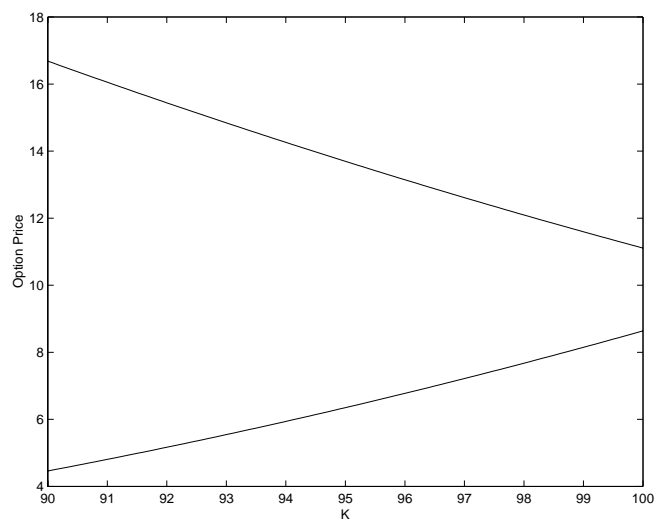
$p + S_0 = 106.3$ and $c + K \cdot \exp(-R \cdot T) = 106.3$.

Plotting p vs. K :

```
K = [90:0.1:100];
```

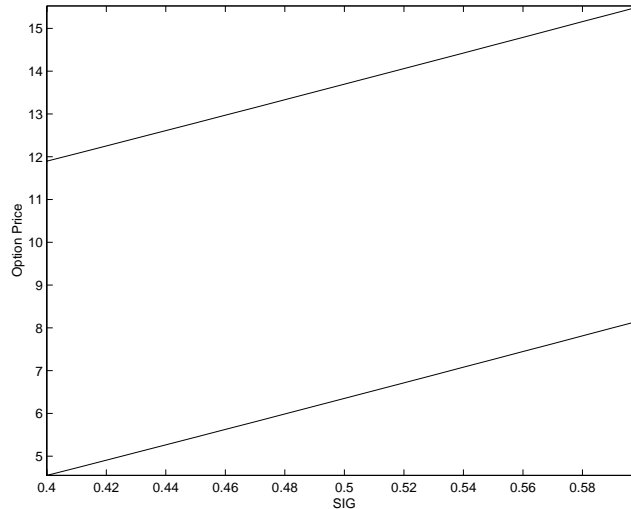
```
[c, p] = blsprice(S0, K, R, T, SIG, Q);
```

```
plot(K, c); hold on; plot(K, p);
```

 generates the following figure:

Plotting p, c as a function of σ :

```
SIG = [0.4: 0.01: 0.6]; K = 95;
[p, c] = blsprice(S0, K, R, T, SIG, Q);
plot(SIG, c); hold on; plot(SIG, p); generates the following:
```



3.0 Sensitivity Measures

Delta	Gamma	Rho	Theta	Vega (Kappa)	Lambda
$\frac{\partial V}{\partial S}$	$\frac{\partial^2 V}{\partial S^2}$	$\frac{\partial V}{\partial r}$	$\frac{\partial V}{\partial T}$	$\frac{\partial V}{\partial \sigma}$	$\frac{\Delta V}{V}$

4.0 Implied Volatility

Implied volatility can be obtained using a MATLAB command `blsimpv`;

```
SIG = blsimpv(S0, K, R, T, c);
```

with the current asset price S_0 , the exercise price K , the risk free interest rate R , the time to maturity T , and call option value c . Note the call option value c is give as an input. This is one nonlinear equation in one unknown: MATLAB uses the Newton method to solve this.

To find numerical solutions to BS, the “formal” problem is to solve

$$F(V, \sigma, S, T, r, q) = 0 \text{ for } V$$

where σ, S, T, r, q are known and V is linear. The implied problem is to solve

$$F(V, \sigma, S, T, r, q) = 0 \text{ for } \sigma$$

where σ is unknown and nonlinear.

5.0 Example: A Delta-neutral portfolio of two options

(from the MATLAB User's Guide)

Given two options, OP1 (call) and OP2 (put) and total resources of \$1000.-. The objective is to find x_1, x_2 which satisfy:

$$P = \text{OP1}x_1 + \text{OP2}x_2$$

such that

$$\text{cost}(P) = 1000, \text{ and}$$

$$\frac{\partial P}{\partial S} = 0$$

Step 1: Get V_1 and V_2 , the prices for OP1, OP2 using `blsprice()`

Step 2: Get D_1, D_2 , the Deltas for the above options using `blsdelta()`.

Step 3: Solve the linear system

$$x = A \setminus b; \text{ where}$$

$$A = [D_1 \ D_2; \ V_1 \ V_2]; \text{ and}$$

$$b = [0; \ 1000];$$

i.e., we spend $x_1 V_1$ dollars on OP1 and $x_2 V_2$ dollars on OP2.

Note that positions x_1, x_2 may be negative, which corresponds to short-selling the options.

One of the exercises in Problem set #1 is to generalize this. We now have a system of four options, OP1, OP2, OP3 and OP4 subject to a maximum expenditure of \$17'000.- which we want to hedge against Delta, Gamma and Vega.

Again,

Step 1: Get the prices V_i using `blsprice()`.

Step 2: Get the vectors of Deltas, Gammas and Vegas using `blsdelta()`, `blsgamma()`, and `blsvega()`.

Step 3: Set up and solve the linear system $Ax = b$ given by:

$$A = \begin{bmatrix} \Delta_1, \dots, \Delta_4 \\ \gamma_1, \dots, \gamma_4 \\ \kappa_1, \dots, \kappa_4 \\ V_1, \dots, V_4 \end{bmatrix} \quad b = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 17000 \end{bmatrix}$$

Questions to think about:

When solving this type of problem, one should investigate the case when A is singular or near-singular, and find the financial setting which describes this. The same question applies to an over/under-determined system.

6.0 Plotting the sensitivity

The relationship between the stock price of the underlying and the sensitivity equations can be obtained with MATLAB, by extending the arguments of the functions to a vector form. As an example, the relationship between S and γ can be obtained using the MATLAB function *blsgamma(...)* as follows:

```
range = 10:70;
span = length(range);
j = 1:0.5:12;
newj = j(ones(span,1),:)/12;
jspan = ones(length(j),1);
newrange = range(jspan,:);
pad = ones(size(newj));
zval = blsgamma(newrange, 40*pad, 0.1*pad, newj, 0.35*pad);
color = blsdelta(newrange, 40*pad, 0.1*pad, newj, 0.35*pad);
mesh(range, j, zval, color);
axis([10 70 1 12 -inf inf]);
```

Call Option Sensitivity Measures

