Question 1

The initial condition for the PDE is:-

$$V(0,S) = \max(0, S-K)$$ — Initial conditions.

Let $\tau = T-t$. At $S=0$, we get

$$V_\tau + rV = 0$$

And at S_{max}, $V(\tau, S_{max}) \approx S_{max}$.

Hence the boundary conditions are:

At $S=0$, $V_\tau + rV = 0$

At $S = S_{max}$, $V(\tau, S_{max}) \approx S_{max}$

— boundary conditions

Question 2

The path of the strike price can be simulated using the following difference-equation:

$$S_{t+1} = S_t + r S_t \Delta t + \sigma(S_t, t) S_t \cdot \sqrt{\Delta t}\, \phi$$

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

We can also develop an integral form for Monte-Carlo simulations (for better accuracy):

$$S(t+\Delta t) = S(t) \times e^{r\Delta t - \frac{\sigma^2}{2}\Delta t + \sigma(S_t, t)\sqrt{\Delta t}\cdot \phi}$$

We now get the following pseudo-code:

Let M = total divisions on asset price
For $p = 1$ to M: (where m = total paths)
  For $i = 1$ to $N-1$
    ϕ = randn ($\sim N(0,1)$)
    $\sigma(S,t) = \alpha_1 + \alpha_2 S + \alpha_3 S^2$
    $S(i+1) = S(i) \cdot \exp\left[r\Delta\tau - \sigma^2(S,t)\frac{\Delta\tau}{2}\right]$

$$+ \sigma(S,t)\cdot\sqrt{\Delta\tau}\cdot\phi$$

End

$f(p) = \max[S(N) - K, 0]$

End

Finally: option price $= e^{-rT} \cdot \frac{\text{sum}(f)}{M}$

Please see attached matlab code for program.

Question 3

At $S=0$, $V_\tau = -rV \Rightarrow \frac{V_0^{n+1} - V_0^n}{\Delta\tau} = -rV_0^n$

$$\therefore \quad V_0^{n+1} = (1 - r\Delta\tau)\, V_0^n$$

We also define:

$$\alpha_{central} = \left[\frac{\sigma S_i^2}{(S_i - S_{i-1})(S_{i+1} - S_{i-1})} - \frac{r S_i}{S_{i+1} - S_{i-1}} \right]$$

$$\beta_{central} = \left[\frac{\sigma S_i^2}{(S_{i+1} - S_i)(S_{i+1} - S_{i-1})} + \frac{r S_i}{S_{i+1} - S_{i-1}} \right]$$

$$\alpha_{forward} = \left[\frac{\sigma S_i^2}{(S_i - S_{i-1})(S_{i+1} - S_i)} \right]$$

$$\beta_{forward} = \left[\frac{\sigma^2 S_i^2}{(S_{i+1} - S_i)(S_{i+1} - S_{i-1})} + \frac{r S_i}{S_{i+1} - S_{i-1}} \right]$$

Using these equations and previous initial/boundary conditions, we have the following pseudo-code:

$V_i^0 = \text{payoff}(S_i) = (S_i - K)^+$

for $n = 0, 1, \cdots N-1$

$V_0^{n+1} = (1 - r\Delta\tau) V_0^n$ (boundary condition at $S=0$)

$V_m^{n+1} = V_m^n$ (boundary condition at $S = S_{max}$)

for $i = 1, 2, \cdots M-1$

sigma = max(0, $x_1 + x_2 \cdot S + x_3 \cdot S^2$) (LVF function)

Compute $\alpha_{central}$ & $\beta_{central}$

if $\alpha_{central} \geq 0$ & $\beta_{central} \geq 0$

$\alpha_i = \alpha_{central}$, $\beta_i = \beta_{central}$

else $\alpha_i = \alpha_{forward}$, $\beta_i = \beta_{forward}$

} central diff. and upstream weighting

$V_i^{n+1} = V_i^n [1 - (\alpha_i + \beta_i + r)\Delta t] + \alpha_i \Delta\tau V_{i-1}^n + \beta_i \Delta\tau V_{i+1}^n$

end

end

Please see attached matlab code for implementation.

Question 4

Please run attached PriceTest.m

Program Output:

Monte-Carlo result = 0.044251
Explicit Finite-Diff result = 0.038112

Question 5(a)

Please see attached myfun.m.

Sample output (single output):

```
>> F = myfun([0.1; 0; 0])

F =

-0.1511
-0.1238
-0.1082
-0.1145
-0.1188
-0.1148
-0.0669
```

Sample output (two outputs):

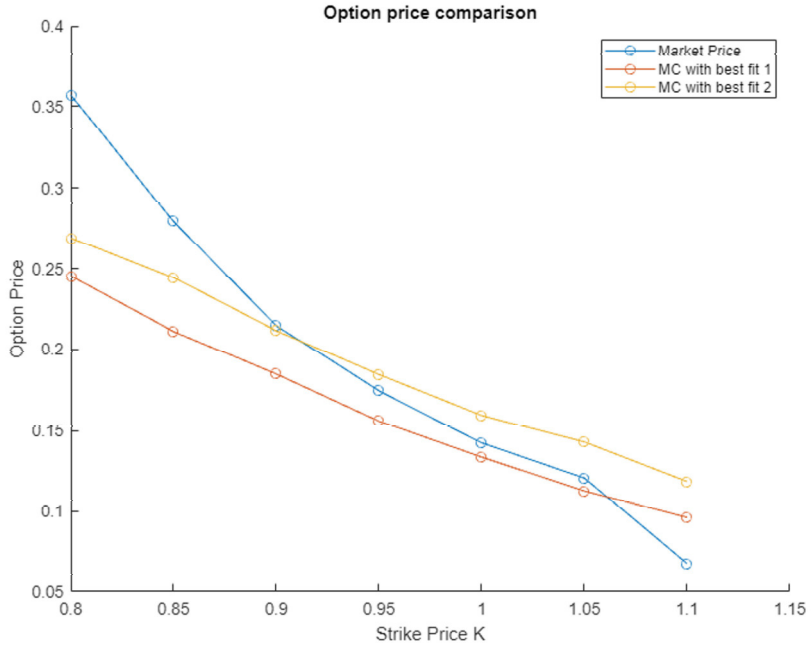
```
[F, J] = myfun([0.1; 0; 0])

F =

-0.1513
-0.1232
-0.1081
-0.1141
-0.1193
-0.1147
-0.0668

J =

 0.1353   -0.0007    0.1646
-0.0169    0.1406    0.0682
-0.0327    0.0434   -0.0438
 0.0682    0.0633   -0.0019
 0.2289    0.2141    0.2767
 0.1368    0.1197    0.1337
 0.0535    0.0416    0.0565
```

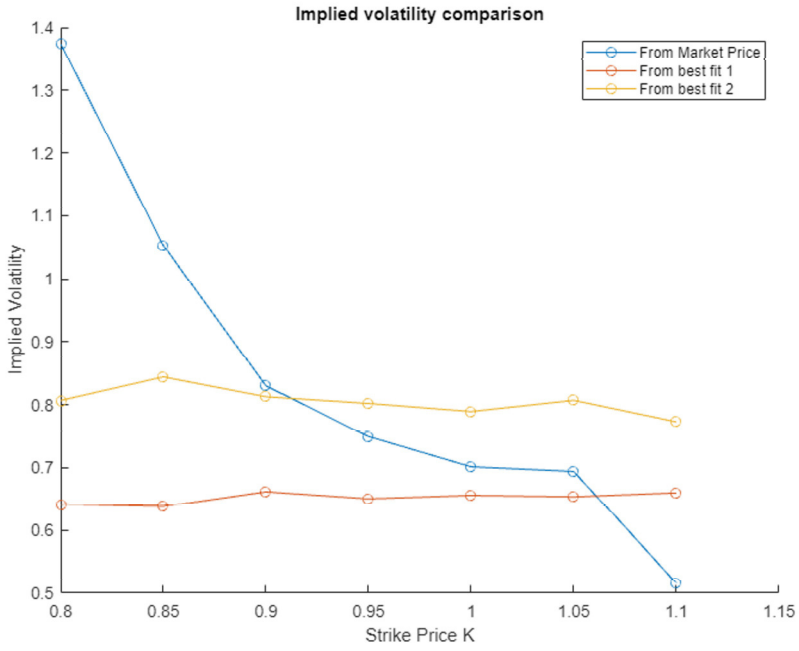
Question 5(b)

Please run Calibration.m. The sample output is shown below.

```
Please give this program a few min to complete

 Iteration  Func-count     Residual      First-Order optimality      Lambda          Norm of step
     0          1       0.0801528          0.0724                   0.01
     1          2       0.0247665           0.117                   0.001          0.416367
     2          3       0.0213934          0.0209                   0.0001         0.227949
     3          4       0.0197325          0.0708                   1e-05          0.0660373
     4          8       0.0195859           0.224                   0.01           0.299954
     5          9       0.0182625          0.0637                   0.001          0.173762
     6         14       0.0170453           0.142                   10             0.0074308
     7         20                                                                  2.15845e-07

Local minimum possible.
lsqnonlin stopped because the relative size of the current step is less than
the value of the step size tolerance.

<stopping criteria details>

 Iteration  Func-count     Residual      First-Order optimality      Lambda          Norm of step
     0          1       0.0808935           0.126                   0.01
     1          2       0.0128765          0.0468                   0.001          0.588238
     2         11                                                                  6.15901e-07

Local minimum possible.
lsqnonlin stopped because the relative size of the current step is less than
the value of the step size tolerance.

<stopping criteria details>
Best Fit X1 = [0.38061     0.45204    -0.16809] with error = 0.017045
Best Fit X2 = [0.68959     0.28021    -0.16676] with error = 0.012876
>>
```

The output graphs are shown below:





Discussion

We can see that the second starting point gives a better calibration and a lower error. This is expected since it can use the S & S^2 terms from the beginning.

We also see that both starting points approximately track the real market-price, but it is not perfect. Hence a simple quadratic LVF might be insufficient and we can go for a cubic (S^3) or higher order LVF for better results.

The program does not do too well in predicting the implied volatility. Note that our objective function considers the option prices, not implied volatility. Hence the program does not give a very good result. To improve this, we can use the implied volatility in the objective function instead of option price.