Problems:

1. **Programming a Black-Scholes Implied Volatility Solver:** The purpose of this project is to build a flexible and and adaptable implied volatility solver in an object oriented frame-work using *C++*. Graduate and actuarial students may use *R* or *Matlab* if they wish but they shall be required to implement everything from scratch without using any built-in functions.

   The basic elements required shall be

   - A function that computes the values of the normal density $\phi(x)$ and cumulative distribution function $\Phi(x)$ (cdf). For the cdf you may use the approximation

     $$\Phi(x) \approx 1 - \phi(x)(b_1 t + b_2 t^2 + b_3 t^3 + b_4 t^4 + b_5 t^5)$$

     where $t = \frac{1}{1+b_0 x}$, $b_0 = 0.2316419$, $b_1 = 0.319381530$, $b_2 = -0.356563782$, $b_3 = 1.781477937$, $b_4 = -1.821255978$, and $b_5 = 1.330274429$. All students should explore the accuracy of the normal cdf approximation given for a wide range of x values. Compare the accuracy of this approximation to the to another implementation (e.g. the pnorm function in R). There are intervals where where some adjustments are necessary but these are easy to make if you use simple properties of the normal cdf.
   - A function that computes Black-Scholes prices of European Call and Put Options. Having your function also return the option Delta would be useful.
   - A nonlinear equation solver (either bisection, Newton-Rhapson, or both) for $f(x) = y$.
   - An implied volatility calculator which accepts as inputs the observed option price, the market price of the stock, the strike price, the time to maturity, the interest rate, and an initial guess of the volatility (for Newton-Rhapson) or another method of initializing the algorithm (for bisection you need two values (a,b) that satisfy the conditions $f(a) < y$ and $f(b) > y$)).

   Write a program that uses the data from July 29, 2002 to calculate the implied volatilities for call and put option ask prices which expire on September 26, 2002. Assume $S_0 = 4.75$, and $r = 0.0492$ per annum (use a time scale in units of years with 365 days per year).

K	Call price-ask	Put price-ask
4.00	-	0.02
4.25	-	0.04
4.50	0.33	0.09
4.75	0.16	0.20
5.00	0.06	0.38
5.25	0.02	0.59
5.50	0.01	-
5.75	0.01	-

   You will need to calculate the number of years to expiry.

   What do you observe with this data (is there a notable 'volatility smile)? Compare your results to those you obtained in Assignment 1 for the same data.

   Solution:
   The following code implements the given approximation in the R programming implementation:

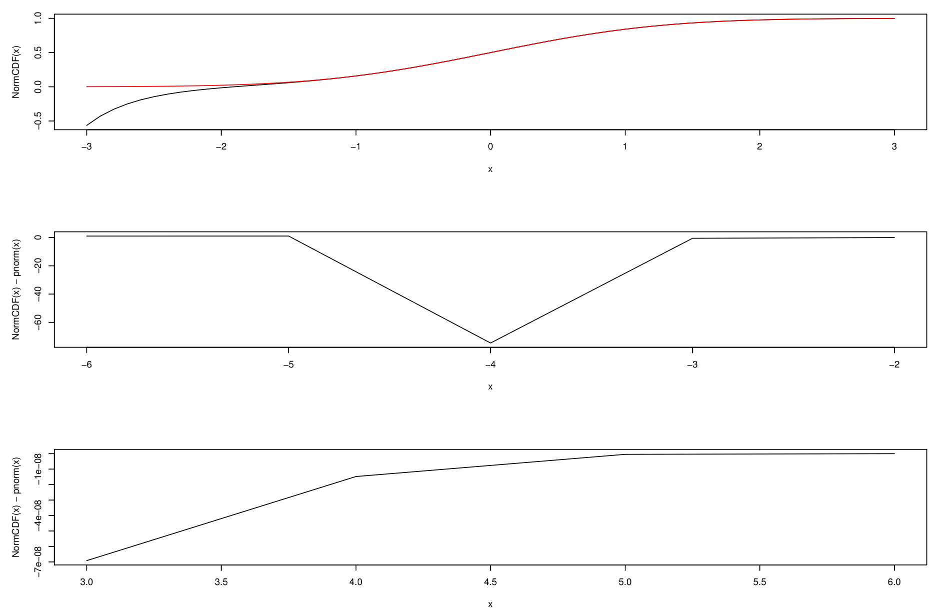

Figure 1: Rational Approximation of the Normal CDF (black) versus `pnorm` (red)

```
NormCDF <- function(x){
  b0 <- 0.2316419
  b1 <- 0.319381530
  b2 <- -0.356563782
  b3 <- 1.781477937
  b4 <- -1.821255978
  b5 <- 1.330274429
  t <- 1/(1+b0*x)
  phi_x <- (1/sqrt(2*pi))*exp(-0.5*x^2)
  y <- 1 - phi_x*(t*(b1 + t*(b2 + t*(b3 + t*(b4 + t*b5)))))
  return(y)
}
```

The top plot in Figure 1 compares the results of the approximation with the built-in R function `pnorm` over the interval $(-3, 3)$. The middle plot looks at the difference between the two function over the interval $(-6, -2)$. The bottom plot looks at the difference between the two functions over the interval $(3, 6)$.

Since the approximation has large errors in the interval $(-6, -2)$ but appears reasonable for $x > 0$ we can adjust the function to use the symmetry of the normal distribution since $\Phi(-x) = 1 - \Phi(x)$ for $x \geq 0$. The following function incorporates this adjustment

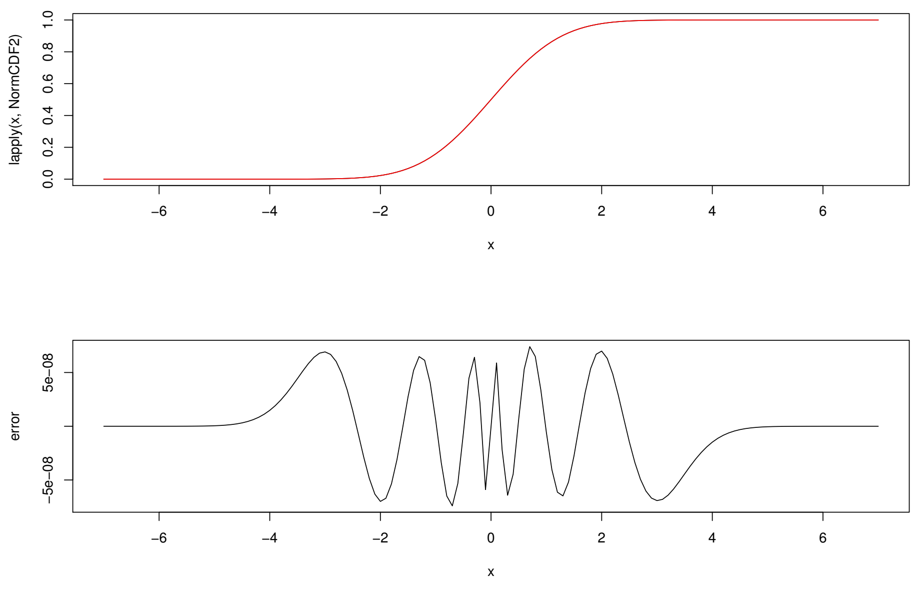

Figure 2: Rational Approximation of the Normal CDF (black) versus `pnorm` (red)

```
NormCDF2 <- function(x){
  b0 <- 0.2316419
  b1 <- 0.319381530
  b2 <- -0.356563782
  b3 <- 1.781477937
  b4 <- -1.821255978
  b5 <- 1.330274429
  z <- abs(x)
  t <- 1/(1+b0*z)
  phi_z <- (1/sqrt(2*pi))*exp(-0.5*z^2)
  y <- 1 - phi_z*(t*(b1 + t*(b2 + t*(b3 + t*(b4 + t*b5)))))
  if (x > 0){
    return(y)
  } else return(1-y)
}
```

Figure 2 plots the result of the approximation over the interval $(-7, 7)$ and the errors over $(-10, 10)$. While the errors show some fluctuation we find that the maximum absolute difference is 7.405715×10^{-8}.

The following functions return the Black-Scholes Call and Put prices respectively.

```
BScall <- function(t,T,S,K,r,sigma){
  d1 <- (log(S/K) + (r+0.5*sigma^2)*(T-t))/(sigma*sqrt(T-t))
  d2 <- d1 - sigma*sqrt(T-t)
  N1 <- NormCDF2(d1)
  N2 <- NormCDF2(d2)
  y <- S*N1 - K*exp(-r*(T-t))*N2
  return(y)
}

BSput <- function(t,T,S,K,r,sigma){
  d1 <- (log(S/K) + (r+0.5*sigma^2)*(T-t))/(sigma*sqrt(T-t))
  d2 <- d1 - sigma*sqrt(T-t)
  N1 <- NormCDF2(-d1)
  N2 <- NormCDF2(-d2)
  y <-  K*exp(-r*(T-t))*N2 - S*N1
  return(y)
}
```

The code on the following page uses the bisection algorithm to compute implied volatilities for the call data. We find

Strike	Observed Call Price	Implied Volatility
4.50	0.33	0.1955344
4.75	0.16	0.1850393
5.00	0.06	0.1810154
5.25	0.02	0.1872637
5.50	0.01	0.2144398
5.75	0.01	0.2666544

Similar code gives us the results for the observed put prices.

Strike	Observed Put Price	Implied Volatility
4.00	0.02	0.3178383
4.25	0.04	0.2861530
4.50	0.09	0.2727943
4.75	0.20	0.2878284
5.00	0.38	0.3360327
5.25	0.59	0.3863758

```
# Parameters
S0 <- 4.75
r <- 0.0492
T <- 59/365

C_Strike <- c(4.5,4.75,5.0,5.25,5.5,5.75)
C_obs <- c(.33,.16,.06,.02,.01,.01)
Calls <- data.frame(C_Strike,C_obs)

P_Strike <- c(4.0,4.25,4.5,4.75,5.0,5.25)
P_obs <- c(.02,.04,.09,.2,.38,.59)
Puts <- data.frame(P_Strike,P_obs)

# Use the bisection algorithm to find the implied volatilities
# Note that the starting interval [0.01,1] provides a good interval
# for each option

n_calls <- length(Calls$C_obs)
sigma_imp <- rep(0,n_calls)

for (i in 1:n_calls){
  sigmaa <- 0.01
  sigmab <- 1
  error <- 1
  while (error > 10^-6){
    m <- (sigmaa+sigmab)/2
    if (BScall(0,T,S0,Calls$C_Strike[i],r,m)<Calls$C_obs[i]){
      sigmaa <- m}
    if (BScall(0,T,S0,Calls$C_Strike[i],r,m)>Calls$C_obs[i]){
      sigmab <- m}
    error <- abs(BScall(0,T,S0,Calls$C_Strike[i],r,m) - Calls$C_obs[i])
  }
  sigma_imp[i] <- m
}

# Verify the prices computed with the implied volatilities agree with
# observed values

for (i in 1:n_calls){
y <- BScall(0,T,S0,Calls$C_Strike[i],r,sigma_imp[i])
print(y)
}

Calls <- data.frame(Calls,sigma_imp)
```

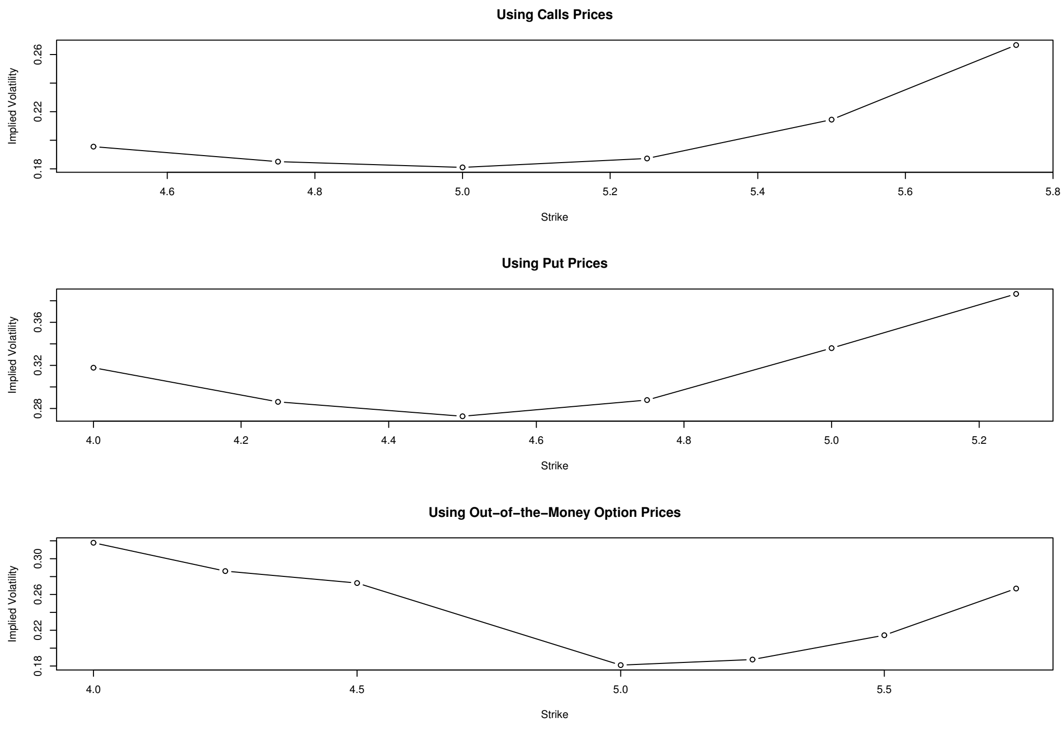

Figure 3: Implied Volatilities

Since investors may believe that options which are in the money with a short time to expiration are more likely to payoff there could be a premium to in the money options.

The plots in Figure 3 show the calculated implied volatilities for calls, puts, and out of the money options. We observe a weak smile (more of a smirk) which contradicts the constant volatility assumption of the Black-Scholes model.