七、Implied Volatility 與二分法尋根*

(一)Implied Volatility

- 波動性，Volatility，乃衡量資產價格變動的程度。
- 為何需要知道波動性
  - 權利型式的金融工具，其價值受到波動性的影響
  - Example: Stock Option, Interest Rate Option, Convertible Bond
  - 計算 Greeks 與風險值時，需要波動性
- 問題：大部份金融資產價格的波動性無法直接由市場觀察到
  - OTC Products：直接報 Vol
  - Listed Products：報價格

➢ 歷史波動性

- ✦ 使用過去之歷史資料求得資產價格的波動性
  - 波動性與採樣時間週期的平方根成正比

- ✦ 估計步驟
  - 取得歷史價格資料 P_0, P_1, P_2, …, P_N。
  - 計算報酬資料 R_1, R_2, R_3, …, R_N。

$$R_n = Ln\left[\frac{P_n}{P_{n-1}}\right]$$

  - 求取報酬之標準差
  - 轉換為年標準差

➢ 隱含波動性

✦ 有限的來源與產品

- Market Vol 可由一些資訊源取得，如 Reuters、Bloomberg。
- 只限於特定的產品。

✦ 資料的透明性

- 只有交易所交易的產品價格有較佳可靠性。

✦ 模型的缺點

- Black-Scholes 模型有太多的假設。
- 隱含波動性只適用一特定期限，無法內、外差。

$$C = f(S, K, T, r_d, r_f, \sigma)$$

$$\sigma = f^{-1}(C, S, K, T, r_d, r_f)$$

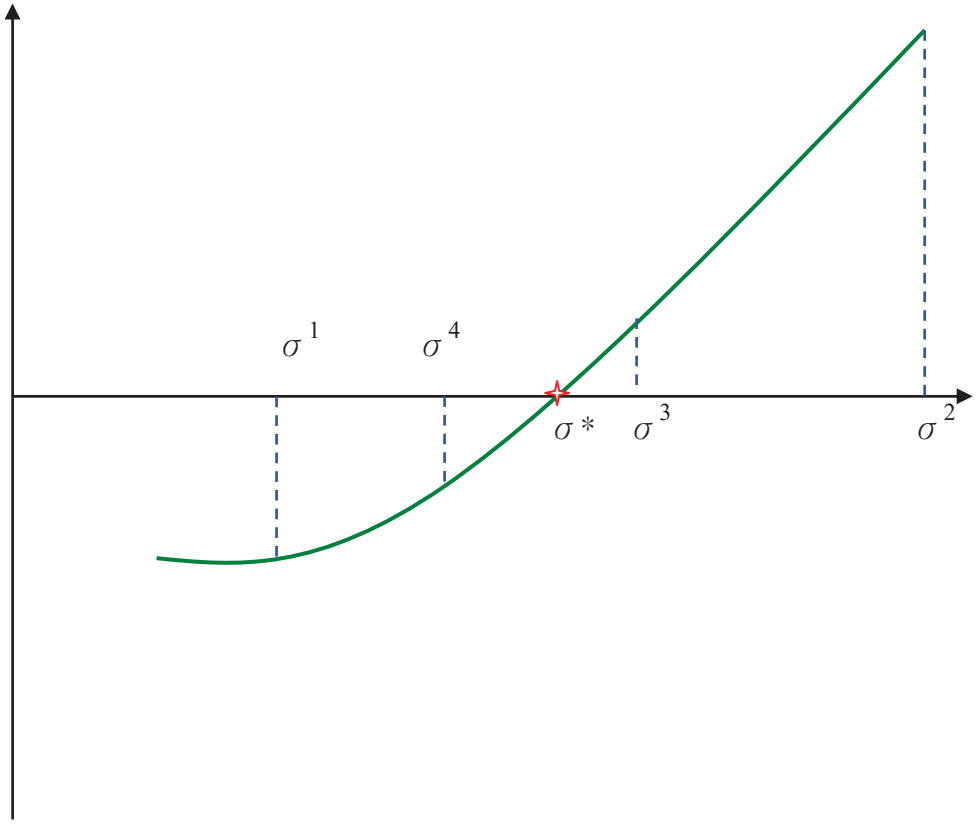
(二)Bisection Method

- 已知 C*，要求 σ *，如下式

$$C^* = f(S,K,T,r_d,r_f,\sigma^*)$$

- 轉化為下式求根

$$g(\sigma^*) = f(S,K,T,r_d,r_f,\sigma^*) - C^* = 0$$



➢ Pseudo Code

```
Input: ε, a, and b(b>a and g(a)*g(b)<0);
Real: length, c;
Length:=b-a;
While[length>ε]
{
   c:=(a+b)/2
   if[g(c)=0] return c;
   else if[g(a)g(c)<0] b:=c;
   else a:=c;
   length:= b-a;
}
return c;
```

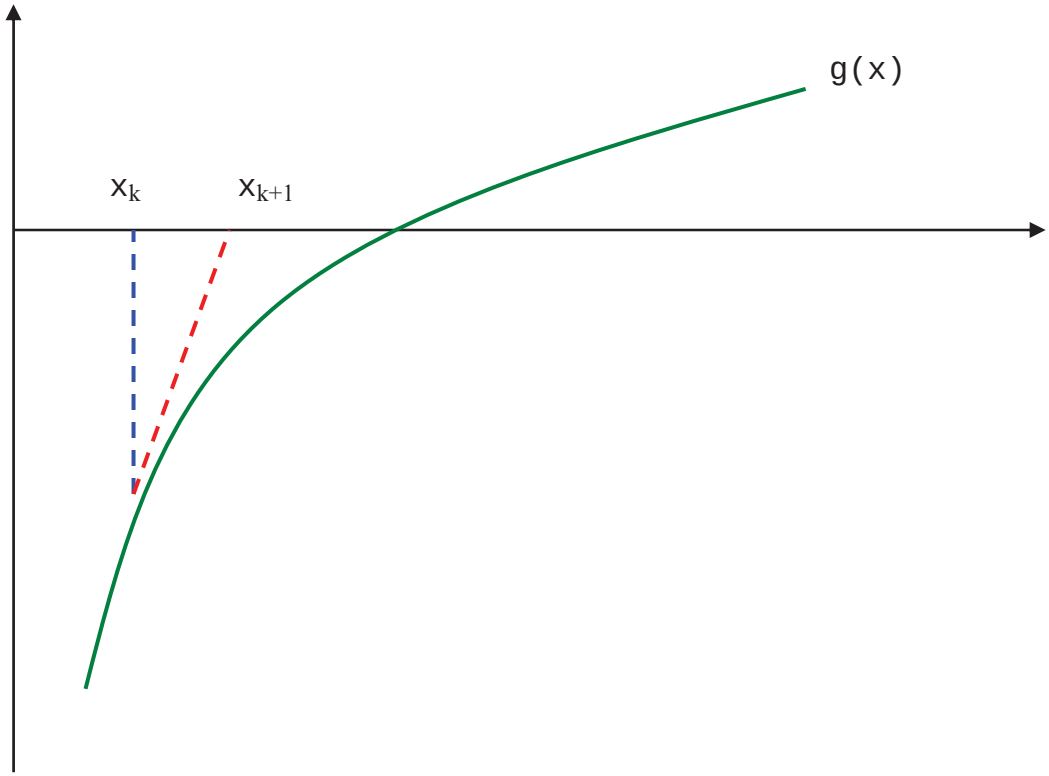
(三)Code for BS Options

```
Const pi = 3.1415926
Public Sub Implied_Vol()
  Dim S As Double, K As Double, r As Double, f As Double
  Dim sd1 As Double, sd2 As Double, T As Double
  Dim P As Double, std As Double, err As Double
  Dim tol As Double, value As Double
  S = Worksheets("Implied Vol").Range("A4").value
  K = Worksheets("Implied Vol").Range("B4").value
  r = Worksheets("Implied Vol").Range("E4").value
  f = Worksheets("Implied Vol").Range("F4").value
  T = Worksheets("Implied Vol").Range("D4").value
  P = Worksheets("Implied Vol").Range("B7").value
  tol = 0.0001
  sd1 = 0.0001
  sd2 = 2
  std = (sd1 + sd2) / 2
  Call Op_Cvalue(S, K, r, f, std, T, value)
  err = Abs(value - P)

  Do While (err >= tol)
    If (value >= P) Then sd2 = std
    If (value < P) Then sd1 = std
    std = (sd1 + sd2) / 2
    Call Op_Cvalue(S, K, r, f, std, T, value)
    err = Abs(value - P)
  Loop

  Worksheets("Implied Vol").Range("B9").value = std
End Sub

Public Sub Op_Cvalue(S As Double, K As Double, r As Double,
  f As Double, sd As Double, T As Double, Price As Double)
  Dim d1 As Double, d2 As Double
  Dim Class As String, Position As String

  d1 = (Log(S / K) + (r - f + sd * sd / 2) * T) / (sd * Sqr(T))
  d2 = d1 - sd * Sqr(T)
  Price = S * Exp(-f * T) * NorCDF(d1) - K * Exp(-r * T) * NorCDF(d2)
End Sub

Public Sub Op_Pvalue(S As Double, K As Double, r As Double, f As
Double, sd As Double, T As Double, Price As Double)
  Dim d1 As Double, d2 As Double
  Dim Class As String, Position As String

  d1 = (Log(S / K) + (r - f + sd * sd / 2) * T) / (sd * Sqr(T))
  d2 = d1 - sd * Sqr(T)
  Price = K * Exp(-r * T) * NorCDF(-d2) - S * Exp(-f * T) * NorCDF(-d1)
End Sub
```

```
Public Function NorCDF(d As Double) As Double
  Dim ans As Double, g As Double
  Const a1 = 0.4361836
  Const a2 = -0.1201676
  Const a3 = 0.937298

  g = 1 / (1 + 0.33267 * d)
  If d >= 0 Then
    ans = 1 - (a1 * g + a2 * g * g + a3 * g * g * g) * NorPDF(d)
  Else
    ans = 1 - NorCDF(-d)
  End If

  NorCDF = ans
End Function

Public Function NorPDF(x As Double) As Double
  NorPDF = Exp(-x * x / 2) / Sqr(2 * pi)
End Function
```

八、牛頓法尋根*

(一)Newton-Raphson Method



$$g(\sigma^*) = f(S,K,T,r_d,r_f,\sigma^*) - C^* = 0$$

$$g(x_k) = -(x_{k+1} - x_k) \times g'(x_k)$$

$$x_{k+1} \equiv x_k - \frac{g(x_k)}{g'(x_k)}$$ Price_error / Vega

$$g'(x) = \frac{\partial g}{\partial x} = \frac{\partial f}{\partial \sigma} = \text{vega} = \frac{\partial C}{\partial \sigma}$$

$$\frac{\partial C}{\partial \sigma} = S\sqrt{T}e^{-yT}\Phi(d_1)$$

➢ Pseudo Code

```
Input: ε , x_init;
Real: x_new, x_old;
X_old := x_init;
X_new := ∞;
While[|x_new-x_old|> ε ]
{
   x_new = x_old -g(x_old)/g'(x_old);
}
return x_new;
```

(二)Code for BS Options

```
Function ImpVolCall(C As Double, K As Double, T As Double, S As
Double, r As Double, f as Double) As Double
  Dim vol as Double
  Dim err As Double
  Dim dv As Double
  Dim d1 As Double, d2 As Double
  Dim price_err As Double
  Dim vega As Double

  vol = 0.2
  err = 0.0001
  dv = err + 1
  While ( Abs(dv)>err )
    d1 = (Log(S/K) + (r-f+vol*vol/2)*T)/(vol*Sqr(T))
    d2 = d1 - vol*Sqr(T)

    price_err = (S*Exp(-f*T)*NorCDF(d1) _
        - K*Exp(-r*T)*NorCDF(d2))  - C
    vega = S*Exp(-f*T)*Sqr(T)*Exp(-0.5*d1*d1)/Sqr(2*PI)
    dv = price_err/vega
    vol = vol - dv
  Wend

  ImpVolCall = vol
End Function
```