CHAPTER 2

The GARCH(1,1) Model

2.1. The Model

In this chapter, we discuss what is known as the GARCH(1,1) model, introduced by Bollerslev [1]. The distinctive feature of the model is that volatilities of asset price returns are not constant. Under the stochastic regime, price return r_t between for example the end of previous day $t - 1$ and the end of day t can be generated through random normal drawings as

$$r_t = \varepsilon(\ \mu\ ,\ \sigma_t\ ) \tag{2.1}$$

with dynamical volatility σ_t and constant mean μ. The model attempts to keep track and forecast the variations in the volatility through time. Applications of this so-called GARCH volatility are widespread especially in the assessment of portfolio risk exposure over a short period of time.

In GARCH(1,1), future variance σ^2_{t+1} is a weighted average of its immediate past estimation σ^2_t, the most recent observation of squared residual $(\ r_t - \mu\ )^2$, and a long-run average variance V_L. It follows an iteration equation given by

$$\sigma^2_{t+1} = \gamma\ V_L + \alpha\ (\ r_t - \mu\ )^2 + \beta\ \sigma^2_t \tag{2.2}$$

with weight factors $\alpha > 0$, $\beta > 0$, and $\gamma > 0$. Since the total weight must sum up to one, we have

$$\gamma = 1 - \alpha - \beta$$

Note that the constant mean μ in equation (2.2) can be estimated based on its historical average. There are all together three parameters in the model namely $(\ V_L,\ \alpha,\ \beta\ )$ that satisfy the constraints,

$$V_L > 0\ ,\ \alpha > 0\ ,\ \ \beta > 0\ ,\ \text{and}\ \alpha + \beta < 1 \tag{2.3}$$

They can be estimated under the notion of maximum likelihood of seeing the historical data. Given historical time series of price returns $\{\ r_1, r_2, \ldots, r_n\ \}$, we can first estimate the constant mean historically as

$$\mu \cong (1/n)(\ r_1 + \ldots + r_n\ )$$

For a particular choice of model parameters, GARCH volatilities $\{\ \sigma_1, \sigma_2, \ldots, \sigma_n\ \}$ can be generated through equation (2.2) where the iteration starts off from observation r_1 and estimate $\sigma^2_1 \cong (\ r_1 - \mu\ )^2$. According to the random normal assumption in equation (2.1), the likelihood or chance of the entire historical data set to be observed is proportional to

$$L \propto \frac{exp\{\ -½\ (\ r_1 - \mu\ )^2/\sigma^2_1\ \}}{\sqrt{2\pi\sigma^2_1}} \times \ldots \times \frac{exp\{\ -½\ (\ r_n - \mu\ )^2/\sigma^2_n\ \}}{\sqrt{2\pi\sigma^2_n}} \tag{2.4}$$

[1] T. Bollerslev, "Generalized Autoregressive Conditional Heteroscedasticity", *Journal of Econometrics*, 31 (1986) p307-27. See also Robert Engle, "GARCH 101 : The use of ARCH/GARCH Models in Applied Econometrics", *Journal of Economic Perspectives*, Vol. **15**, No. 4 (2001) p157-168.

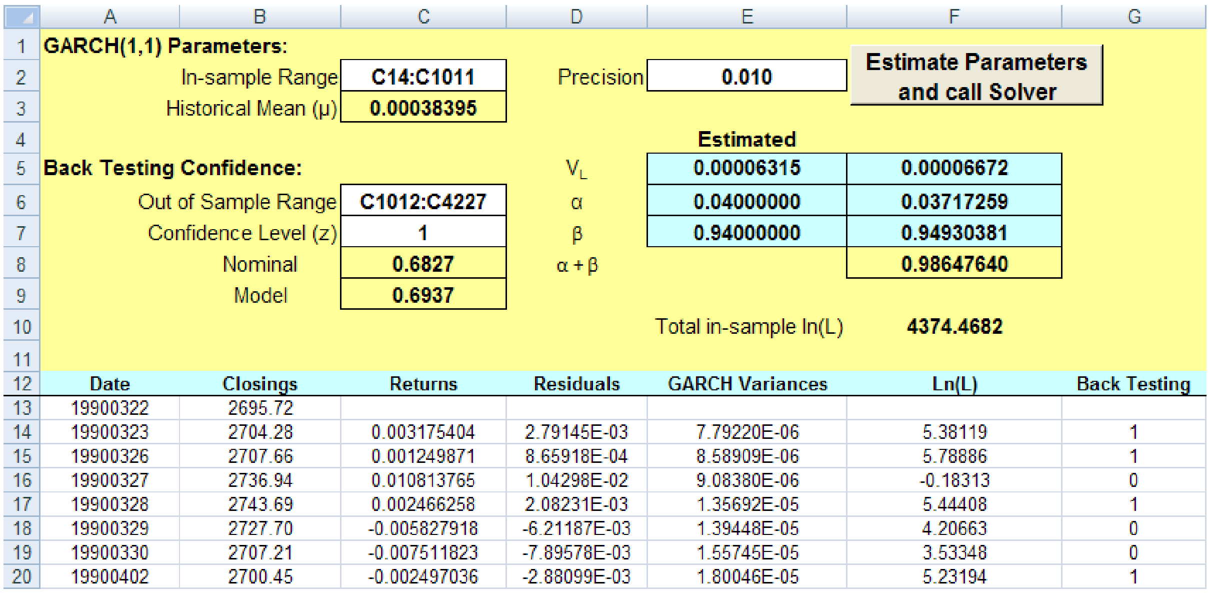
The best model parameters should therefore generate the volatilities { $\sigma_1, \sigma_2, \ldots, \sigma_n$ } that maximize the likelihood in this expression or equivalently the logarithm of likelihood given by

$$ln(L) = -\tfrac{1}{2} \textstyle\sum_{t=1}^{n} \left[\; ln(\sigma_t^2) + \dfrac{(r_t - \mu)^2}{\sigma_t^2} \right] \qquad (2.5)$$

where all constant terms irrelevant to the maximization are ignored in the equation.

2.2. EXCEL Implementation

Figure 2.1 illustrates how the calculation could be organized in EXCEL spreadsheet [2]. The table analyzes daily returns of Dow Jones Industrial Average (DJI) between 19900322 and 20061206. The leading segment from 19900322 to 19940302 will be taken as in-sample data for the determination of model parameters. The rest will be used as out of sample data to back test the accuracy of the model. From row 13 onward, column A in the table records the date, column B shows the closings of DJI in each of these dates, while column C calculates the corresponding daily returns. For example, the formula adopted in C14 = (B14 – B13)/B13. The cell C2 defines the range “C14:C1011” of the entire in-sample historical returns { $r_1, r_2, \ldots, r_n$ }. The cell C3 = **AVERAGE**(**INDIRECT**(C2)) calculates the corresponding constant mean μ in the model. Trial values of the model parameters (V_L, α, β) are input through cells F5, F6, and F7, respectively. We may define several named cells to enhance the readability of the formulae: C3(mu), F5(longvar), F6(alpha), F7(beta), and C7(zvalue).



	A	B	C	D	E	F	G
1	**GARCH(1,1) Parameters:**						
2		In-sample Range	**C14:C1011**	Precision	**0.010**	**Estimate Parameters and call Solver**	
3		Historical Mean (μ)	**0.00038395**				
4					**Estimated**		
5	**Back Testing Confidence:**			V_L	**0.00006315**	**0.00006672**	
6		Out of Sample Range	**C1012:C4227**	α	**0.04000000**	**0.03717259**	
7		Confidence Level (z)	**1**	β	**0.94000000**	**0.94930381**	
8		Nominal	**0.6827**	α + β		**0.98647640**	
9		Model	**0.6937**				
10					Total in-sample ln(L)	**4374.4682**	
11							
12	**Date**	**Closings**	**Returns**	**Residuals**	**GARCH Variances**	**Ln(L)**	**Back Testing**
13	19900322	2695.72					
14	19900323	2704.28	0.003175404	2.79145E-03	7.79220E-06	5.38119	1
15	19900326	2707.66	0.001249871	8.65918E-04	8.58909E-06	5.78886	1
16	19900327	2736.94	0.010813765	1.04298E-02	9.08380E-06	-0.18313	0
17	19900328	2743.69	0.002466258	2.08231E-03	1.35692E-05	5.44408	1
18	19900329	2727.70	-0.005827918	-6.21187E-03	1.39448E-05	4.20663	0
19	19900330	2707.21	-0.007511823	-7.89578E-03	1.55745E-05	3.53348	0
20	19900402	2700.45	-0.002497036	-2.88099E-03	1.80046E-05	5.23194	1

Figure 2.1: EXCEL Implementation of GARCH(1,1) model.

The fourth column from D14 onward calculates the residuals ($r_t - \mu$) for each of these returns using the formula D14 = (C14 – mu), for example. GARCH variances σ_t^2 are records in the fifth column from E14. They are generated iteratively using the formula (see equation (2.2))

$$\text{E15} = (1 - \text{alpha} - \text{beta}) * \text{longvar} + \text{alpha} * \text{D14\^{}2} + \text{beta} * \text{E14}$$

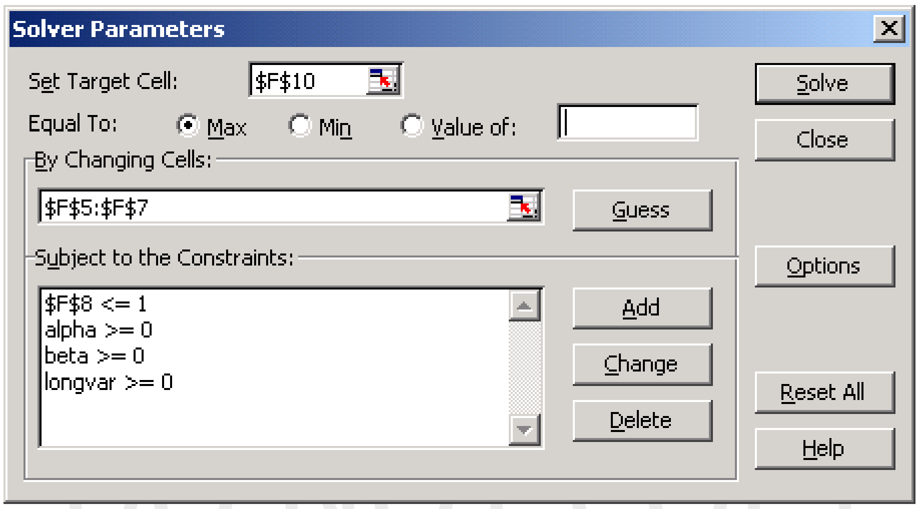
starting off with the value in E14 = D14^2. To determine the best model parameters, we need to first evaluate the likelihood value associated with each trial parameter set. Column F under the data caption implements term-by-term the expression for *ln*(*L*) in equation (2.5) using the formula

[2] Refer to garch11.xls.

$$F14 = (-0.5) * (\mathbf{LN}(E14) + D14\text{^}2/E14)$$

such that the total in-sample $ln(L)$ is given by cell F10 = **SUM**(**OFFSET**(**INDIRECT**(C2),0,3)). For example, consider the trial model parameters of ($V_L = 0.00005$, $\alpha = 0.02$, $\beta = 0.95$) that satisfy the constraints in (2.3), we have the likelihood value being $ln(L) = 4365.5993$.

Here, we are interested in choosing (V_L, α, β) that maximize $ln(L)$ under the constraints in (2.3). Such task can be achieved using the SOLVER algorithm in EXCEL. We can simply go to TOOLS, then SOLVER, and pop up the **Solver Parameters** screen as shown in Figure 2.2. Set **Target Cell** to be the cell F10 that is the likelihood value $ln(L)$, check **Equal To** as **Max** for maximizing, and include in **By Changing Cells** the cells F5:F7 for trial values of V_L, α, and β.


Solver Parameters

Set Target Cell: $F$10

Equal To: [✓ Max] Min Value of:

By Changing Cells: $F$5:$F$7

Subject to the Constraints:

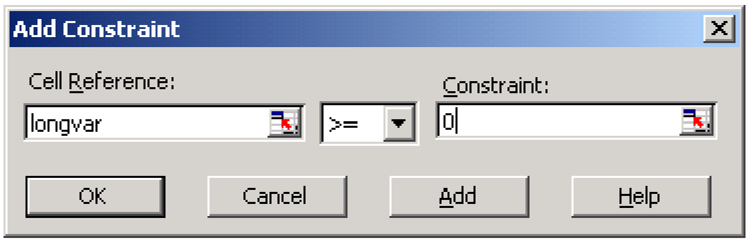
$F$8 <= 1
alpha >= 0
beta >= 0
longvar >= 0

Figure 2.2: Solver Parameters screen.

The constraints in (2.3) can easily be included in the SOLVER algorithm under the **Subject to the Constraints** field. Click **Add** and enter through the **Add Constraint** screen in Figure 2.3 statements :

longvar >= 0 , alpha >= 0 , beta >= 0 , and F8 <= 1

for the constraints $V_L > 0$, $\alpha > 0$, $\beta > 0$, and $\alpha + \beta < 1$, respectively. In the spreadsheet, we have defined the cell F8 = alpha + beta to be the sum of α and β. Note that SOLVER provides only the choices of “>=” and “<=” operators for our purpose. Under floating point environment, they work effectively in the same way as the strictly greater and strictly smaller operators “>” and “<”. The non-negative constraints can also be included through an alternative setup. We can click **Options** to open the **Solver Options** screen and check **Assume Non-Negative** that applies the constraints to the cells F5:F7 specified in **By Changing Cells**.


Add Constraint

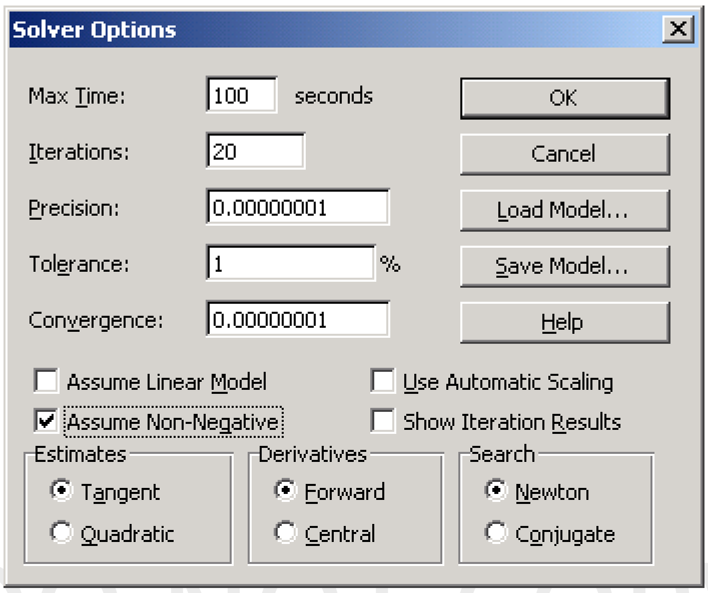
Cell Reference: longvar

>=

Constraint: 0

Figure 2.3: Add Constraint screen.

SOLVER adapts a gradient search algorithm specified by the **Estimates**, **Derivatives**, and **Search** fields in the **Solver Options** screen as shown in Figure 2.4. For the current problem, we need to insert in **Max Time** a maximum running time, in **Iterations** the number of iterations in the search, and in **Precision** the required precision on the cells F5:F7 in **By Changing Cells**. Depending on the problem, this represents only the maximum achievable precision if there is enough number of iterations. Here, SOLVER will normally require 10 to 20 iterations to achieve the precision of 10^{-8}. To start off the search algorithm, we need to provide initial values for the cells F5:F7. Suppose we initiate the search with F5 = 0.00007, F6 = 0.04, and F7 = 0.90. By clicking **Solve**, SOLVER returns the optimal solutions to be F5 = 0.00006663, F6 = 0.03714556, and F7 = 0.94929286 after the search with maximum likelihood of F10 = 4374.46820612.


Solver Options

Max Time: 100 seconds
Iterations: 20
Precision: 0.00000001
Tolerance: 1 %
Convergence: 0.00000001

[ ] Assume Linear Model
[✓] Assume Non-Negative
[ ] Use Automatic Scaling
[ ] Show Iteration Results

Estimates: [✓ Tangent] Quadratic
Derivatives: [✓ Forward] Central
Search: [✓ Newton] Conjugate

Figure 2.4: Solver Options screen.

How good is the GARCH(1,1) model with these optimal parameters ? To answer this question, we will back test the model with out of sample historical data. We have included in columns A and B historical closings of DJI up to the trading day 20061206 with about 3000 back testing points right after the in-sample data. The out of sample historical returns are located in C1012 to C4227 as defined in cell C6, and the GARCH variances are located in E1012 to E4227. Recall in the model, known values of r_t and σ_t^2 will allow us to forecast the new variance σ_{t+1}^2 in the next day when the actual r_{t+1} will be observed subsequently. According to the random normal assumption in equation (2.1), the confidence interval of r_{t+1} is given by [$\mu - z\sigma_{t+1}$, $\mu + z\sigma_{t+1}$] with z being the confidence level. In this respect, we can back test the accuracy of the model by checking the percentage that the forecasted interval has included the observation for the entire out of sample data. In Figure 2.1, the cell C7 defines the chosen value of z, while the cell C8 calculates the nominal confidence of the interval based on standard cumulative probability as

C8 = **NORMSDIST**(C7) – **NORMSDIST**(–C7)

Column G under the data caption records on each day the success (1) or failure (0) of whether the forecast confidence interval has included the observation using the formula

G14 = **IF**(**AND**(C14 <= mu + zvalue ∗ **SQRT**(E14) , C14 >= mu – zvalue ∗ **SQRT**(E14)) , 1 , 0)

The cell C9 = **AVERAGE**(**OFFSET**(**INDIRECT**(C6),0,4)) then accumulates the back testing confidence for the entire out of sample data. To check the accuracy of GARCH(1,1), we can compare this value with the nominal value as given by C8.

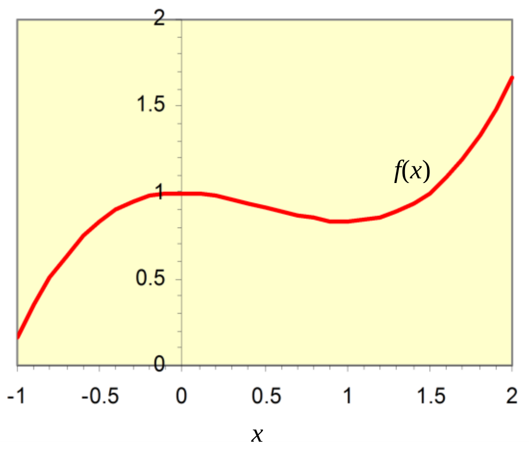
There are critical issues on the choice of initial values that are relevant to the current problem. Without going into detail discussion on its search algorithm, we consider the following example to simply demonstrate an important shortfall of SOLVER. For arbitrary choice of initial values, there is indeed no guarantee of finding the correct optimal solution by SOLVER. The initial values should be as close to the solution as possible.

***Example* 2.1** :

Consider the single-variable function given by $f(x) = \frac{1}{3} x^3 - \frac{1}{2} x^2 + 1$. The local maximum and minimum are located at $x = 0$ and $x = 1$, respectively, where $f(0) = 1$ and $f(1) = 0.8333$. The function is strictly increasing to the right of the local minimum, and it is strictly decreasing to the left of the local maximum.

Suppose we want to use SOLVER to determine the overall maximum and minimum points of the function under the constraints $x \geq -0.4$ and $x \leq 1.4$.

The values of the function at both edges are given by $f(-0.4) = 0.8987$ and $f(1.4) = 0.9347$. They are neither the maximum point nor the minimum point of our concern. The solution should clearly be the points $x = 0$ and $x = 1$, respectively.



Depending on the initial value of x, SOLVER determines only the nearest maximum or minimum points that are not necessary the overall solution. In particular, if the initial value is chosen to be $x = 1.2$, the nearest maximum is located at $x = 1.4$ (the right edge). Upon maximizing, SOLVER returns this point to be the solution rather than $x = 0$ as expected. Similarly, if the initial value is chosen to be $x = -0.2$ for minimizing, SOLVER returns the nearest minimum at $x = -0.4$ (the left edge) not $x = 1$.

Initial value	*Maximum* by SOLVER	*Minimum* by SOLVER
$x = 0.5$	$x = -0.00000001$	$x = 0.99999996$
$x = 1.2$	$x = 1.40000000$	$x = 1.00000000$
$x = -0.2$	$x = -0.00000000$	$x = -0.40000000$

Set Iterations = 20 and Precision = 10^{-8}.

2.3. EXCEL Plus VBA Implementation

In general, there is no a priori information on the model parameters in GARCH(1,1). A preliminary procedure for the purpose of determining proper initial values for SOLVER would definitely be required in view of the shortfall as demonstrated in the above example. An effective way to perform such integrated task is to write additional VBA routines underneath the spreadsheet in Figure 2.1 such that SOLVER can be initiated immediately after the preliminary procedure.

We first develop a function called TotalLikelihood( ) that calculates the likelihood value with trial model parameters. As input, the function reads in the historical squared residuals $\{ h_1, h_2, \ldots , h_n \}$ where $h_t = (r_t - \mu)^2$, the number of terms n, and the model parameters (V_L, α, β). Iteratively, it generates the GARCH variances $\{ \sigma_1^2, \sigma_2^2, \ldots , \sigma_n^2 \}$ and accumulates the likelihood value $ln(L)$ according to equations (2.2) and (2.5), respectively. The pseudo code of TotalLikelihood( ) is given by Code 2.1 as follows:

```
TotalLikelihood( $h(1{:}n)$ , $n$ , $\alpha$ , $\beta$ , $V_L$ )

# define $v_1 = \sigma_1^2$ to start off the iteration and accumulate $ln(L)$
    $v(1) = h(1)$

    $sum = ln(\, v(1)\, ) + h(1) / v(1)$

# generate $\{ v_2 = \sigma_2^2, \ldots , v_n = \sigma_n^2 \}$ by iteration and accumulate $ln(L)$
    For ( $i = 1$ to $n - 1$ ) {  $v(i+1) = ( 1 - \alpha - \beta )\, V_L + \alpha\, h(i) + \beta\, v(i)$

                                  $sum = sum + ln(\, v(i+1)\, ) + h(i+1) / v(i+1)$    }

    $TotalLikelihood = -½\, sum$
```

Code 2.1: Pseudo code of the TotalLikelihood( ) function

We want to develop a search routine called EstimateBestParameters( ) that scans through the valid region of the model parameters and identifies the spot with largest $ln(L)$ which utilizes the above likelihood function. Define in cell E2 the required precision *prec* on the parameters in this preliminary procedure. According to the constraints in (2.3), the search for proper α and β should run through all points given by the double-loop as

$$\alpha = i \times prec \ , \ i = 1, 2, \ldots, N - 1 \qquad (2.6)$$

$$\beta = j \times prec \ , \ j = 1, 2, \ldots, N - i - 1$$

where $N = (1/prec)$ is defined to be the number of grids between zero and one with precision *prec*. We should always choose *prec* such that $(1/prec)$ is an integer. It can be shown[3] that V_L is simply the unconditional variance $E[\, (r_t - \mu)^2 \,]$ of price returns. Numerically, it should be close to the historical variance given by

$$V_{historical} = (1/n)[\, (r_1 - \mu)^2 + \ldots + (r_n - \mu)^2 \,]$$

In practice, the search for proper V_L can be confined within the region $0.8V_{historical} \leq V_L \leq 1.2V_{historical}$ and run through the loop as

$$V_L = (k \times prec)\, V_{historical} \quad , \quad k = k_{low}, \ldots , k_{high} \qquad (2.7)$$

where k_{low} and k_{high} are the nearest integers to $0.8N$ and $1.2N$, respectively. The pseudo code of EstimateBestParameters( ) is given by Code 2.2. As input, the routine reads in historical price returns $\{ r_1, r_2, \ldots , r_n \}$, the number of terms n, and the search precision *prec*. As output, it returns the best model parameters (V_L, α, β) taken to be the initial values for SOLVER. We can set the precision to

[3] See Chapter 9 of Stephen J. Taylor, "Asset Price Dynamics, Volatility, and Prediction", Princeton University Press (2005).

0.01 in EstimateBestParameters( ) and then use SOLVER to fine tune the model parameters. Alternatively, we can set the precision to be very small and estimate the model parameters directly from EstimateBestParameters( ), but this will be numerically intensive.

```
EstimateBestParameters( )

# input historical price returns and precision parameter
  Read $n$, $r(1:n)$, and $prec$

# estimate the historical mean and variance
   $\mu = \textbf{AVERAGE}(\ r(1:n)\ )$
   $V_{historical} = \textbf{VAR}(\ r(1:n)\ )$

# construct the squared residuals
   For ( $i = 1$ to $n$ ) { $h(i) = (\ r(i) - \mu\ )^2$ }

# determine the number of grids given precision
   $N = Int(1/prec)$

# scan through the valid region of the parameters for the largest ln(L)
   $maxlnL = -10^8$

   For ( $i = 1$ to $N - 1$ ) {
       For ( $j = 1$ to $N - i - 1$ ) {
           For ( $k = Int(0.8N)$ to $Int(1.2N)$ ) {

               $\alpha = i\ prec$ ,  $\beta = j\ prec$ ,  $V_L = k\ prec\ V_{historical}$

               $lnL = Totallikelihood(\ h(1:n)\ ,\ n\ ,\ \alpha\ ,\ \beta\ ,\ V_L\ )$

               If( $lnL \geq maxlnL$ ){ $maxlnL = lnL$
                                        $best\alpha = \alpha$ , $best\beta = \beta$ , $bestV_L = V_L$ }

           } } }

# output best model parameters
  Output $bestV_L$, $best\alpha$, and $best\beta$
```

Code 2.2: Pseudo code of the EstimateBestParameters( ) routine

The main routine EstimateBestParameters( ) can be invoked through the button in the spreadsheet. In the VBA coding as shown in Code 2.3, it contains three subroutines and one function designed to tackle specific tasks such that it can be maintained easily. The first statement will display a message box asking for confirmation of starting the calculation. It is always a good practice to include a message box to avoid mis-invoking of long running procedure. The GetInputs( ) routine will read in historical price returns and the precision parameter from the EXCEL spreadsheet and generate both the squared residuals and historical variance for the evaluation of likelihood values below. In the GetInputs( ) routine, the ByRef declaration denotes that the particular variable will be evaluated internally and taken as output of the routine. The first statement in GetInputs( ) inputs historical price returns from the range of in-sample data defined in C2. The triple-loop will scan through the valid region of the model parameters (V_L, α, β) and identify the best spot with maximum likelihood utilizing the TotalLikelihood( ) function. The PutBestValues( ) routine will then return the estimated values of the parameters to the cells E5:E7 for display and as well to the cells F5:F7 for SOLVER input. The final statement triggers SOLVER to perform further optimization based on these initial values. Certainly, it must be configurated properly ahead of time as described in section 2.2. As it will be used in this implementation, we need to add SOLVER in the VBA reference section by clicking **References** of the **Tools** menu item and check SOLVER in the Reference dialog. The TRUE parameter of the SolverSolve function suspends the display of the result dialog at the end of SOLVER execution.

```
Sub EstimateBestParameters()
    If MsgBox("Start calculation?", vbYesNo + vbInformation) = vbNo Then Exit Sub
    'Read inputs
    Dim residualSq() As Double, hVar As Double, prec As Double
    Call GetInputs(residualSq, hVar, prec)
    'initialize values
    Dim bestAlpha As Double, bestBeta As Double, bestLongVar As Double
    Dim i As Integer, j As Integer, k As Integer
    Dim nFrac As Integer: nFrac = Int(1 / prec)
    Dim maxlnL As Double: maxlnL = -100000000#
    'Iterate by the increment of alpha, beta, and longVar
    Dim alpha As Double, beta As Double, longVar As Double, lnL As Double
    For i = 1 To (nFrac - 1)
        alpha = i * prec
        For j = 1 To (nFrac - i - 1)
            beta = j * prec
            For k = Int(0.8 * nFrac) To Int(1.2 * nFrac)
                longVar = k * prec * hVar
                lnL = TotalLikelihood(residualSq, alpha, beta, longVar)
                If lnL > maxlnL Then
                    maxlnL = lnL
                    bestAlpha = alpha
                    bestBeta = beta
                    bestLongVar = longVar
                End If
            Next k
        Next j
    Next i
    'Write outputs
    Call PutBestValues(bestAlpha, bestBeta, bestLongVar)
    'Call solver and turn off the final Solver Results dialog
    'Solver must be configurated ahead of time
    SolverSolve (True)
End Sub

'Read inputs from excel
Sub GetInputs(ByRef residualSq() As Double, ByRef hVar As Double, ByRef prec As Double)
    Dim priceReturn As Range: Set priceReturn = Range(Range("C2").Text)
    Dim mu As Double
    With WorksheetFunction
        mu = .Average(priceReturn)
        hVar = .Var(priceReturn)
    End With
    ReDim residualSq(1 To priceReturn.Count)
    Dim i As Integer
    For i = 1 To priceReturn.Count
        residualSq(i) = (priceReturn(i) - mu) ^ 2
    Next
    prec = Range("E2").Value
End Sub

'Write outputs to excel
Sub PutBestValues(alpha As Double, beta As Double, longVar As Double)
    Range("E5:F5").Value = longVar
    Range("E6:F6").Value = alpha
    Range("E7:F7").Value = beta
End Sub

'Calculate the total log of likelihood
Function TotalLikelihood(residualSq() As Double, alpha As Double, beta As Double, longVar As Double) As Double
    Dim garchVar() As Double: ReDim garchVar(1 To UBound(residualSq))
    garchVar(1) = residualSq(1)
    Dim sum As Double: sum = Log(garchVar(1)) + residualSq(1) / garchVar(1)
    Dim i As Integer
    For i = 1 To (UBound(residualSq) - 1)
        garchVar(i + 1) = (1 - alpha - beta) * longVar + alpha * residualSq(i) + beta * garchVar(i)
        sum = sum + Log(garchVar(i + 1)) + residualSq(i + 1) / garchVar(i + 1)
    Next
    TotalLikelihood = -0.5 * sum
End Function
```

Code 2.3: VBA codes of the EstimateBestParameters( ) routine, GetInputs( ) routine, PutBestValues( ) routine, and TotalLikelihood ( ) function.