Professional Financial Computing Using Excel and VBA
Contents

Preface ix

CHAPTER 1  
Financial Engineering and Computing 1  
1.1 Financial Engineering and Spreadsheet Modeling 1  
1.2 Lehman Brothers’ Products for Retail Investors 3  
1.3 Risk Management and Basel II 4  
1.4 About the Book 4  
1.5 Chapter Highlights 6  
1.6 Other Remarks 7

CHAPTER 2  
The GARCH(1,1) Model 9  
2.1 The Model 9  
2.2 Excel Implementation 10  
2.3 Excel Plus VBA Implementation 15

CHAPTER 3  
Finite Difference Methods 21  
3.1 Difference Equations 21  
3.2 Excel Implementation 24  
3.3 VBA Implementation 28  
3.4 Crank–Nicholson Scheme 33

CHAPTER 4  
Portfolio Mean-Variance Optimization 37  
4.1 Portfolio Selection 37  
4.2 Excel Implementation 42  
4.3 Excel Plus VBA Implementation 48
Preface

This book is a good company to Master degree programs in Financial Engineering, Financial Risk Management, Quantitative Investment, Computational Finance, or Mathematical Finance. Also, risk managers, traders, IT analysts, quantitative analysts working in investment banks and hedge fund will find it to be a good reference.

The book provides VBA examples on some widely-used finance and risk models. We expect that readers have prior training on these models because some of them require strong mathematical foundation. Through the examples, readers can easily build their implementable analytics and apply similar skills to other complex models.

Feedbacks from professors, students, analysts, and risk professionals are warmly welcome.

Humphrey Tung
Donny Lai
Michael Wong
Stephen Ng
Email: efmcw103@gmail.com
CHAPTER 1

Financial Engineering and Computing

1.1 FINANCIAL ENGINEERING AND SPREADSHEET MODELING

“Spreadsheet Modeling for Finance” has long been a popular course in the MSc Financial Engineering program at the university we served in Hong Kong. The course is different from introductory Excel courses in financial management. It is an advanced course offered mainly to students with solid training in mathematical finance, option pricing, and risk modeling. Most of the students in the course have been designated a chartered financial analyst (CFA) or certified as a financial risk manager (FRM). The financial engineering program mainly recruits part-time students working in various financial institutions. There are around 40 to 60 new recruits each year. Many of them are derivatives traders, bank risk managers, bank IT specialists, fund managers, product structurers, bank regulators, and product auditors. In 1997–2008, the program trained more than 500 graduates. Most of them successfully applied the knowledge gained to their daily work.

Some may ask why no “quantitative analysts” are mentioned. Loosely speaking, these financial engineering graduates are quantitative analysts in nature. Strictly speaking, none of them carries the job title “quantitative analyst.” A global investment bank may have one or two quantitative analysts and/or financial engineers in Hong Kong. Given the presence of 15 global institutions, there are a maximum of 10 quantitative analyst job vacancies a year. This number cannot satisfy the continuous supply of financial engineering graduates every year. Although our graduates are not called quantitative analysts, their training in financial engineering did help their fast career development. Also, their quantitative skills have enabled Hong Kong to excel in financial services.
When we planned this book in 2007, the financial market in Hong Kong was very bullish. Many China initial public offering (IPO) deals were completed in Hong Kong. The Hang Seng Index reached over 30,000 points. Structured products and hedge funds were prevalent in corporate banking and private banking. Equity-linked notes, minibonds, and currency-linked products were common in retail banking.

In addition to sizable financial institutions, Hong Kong is a hub of boutique hedge funds. It is believed that there are 600 or more. These hedge funds employ few people, but their asset under management (AUM) can be over US$100 million each. In these hedge funds, financial and risk analysis is mostly based on Excel and Visual Basic for Applications (VBA) programming. This is a reason why the course “Spreadsheet Modeling” is very popular.

Our progress in writing this book was hindered by the financial tsunami in 2008. High market volatility, depreciation of wealth, and massive layoffs in the banking sector brought a lot of frustration to financial practitioners and financial educators. When we completed this book in June 2009, the market remained very weak. Many wealthy individuals suffered huge losses in the past 12 months; financial institutions cut their manpower seriously; selling complex products became difficult; and new regulations were enacted relating to structured products. In 2009, students in the course “Spreadsheet Modeling” still enjoyed the class but were slightly worried outside of the class. This is because the next round, which would be the fourth or fifth round, of massive layoffs would affect them. Investment banking follows obvious business cycles. This applies to study programs in financial engineering as well.

Mature students are always pragmatic in acquiring knowledge. Complex mathematics is very fancy, but our mature students tend to take it for granted and focus mostly on the applications of the mathematics. The course “Spreadsheet Modeling” makes those fancy mathematical concepts more easily applicable. From the perspective of educators, this mindset of the students is not harmful. After using Excel and VBA to build their models, some students become more interested in complex mathematics. What we would like them to know is not simply building models for financial analysis. We wish that they could understand model risks and estimate when these risks are likely to occur. The increased curiosity of our students after the course made us feel satisfied about our educational efforts.

Many new financial products have no mathematical models. Due to the advancement of technology, an analyst can easily apply Monte Carlo simulation on related variables and find out an average value. Our students especially like this analytical approach because there is less of a mathematical foundation required. In fact, Excel and VBA can easily handle Monte Carlo simulation.
1.2 LEHMAN BROTHERS’ PRODUCTS FOR RETAIL INVESTORS

Since 2005, Lehman Brothers began actively distributing a wide range of structured products via retail banks in Hong Kong, as well as in Singapore. One of our former financial engineering students came from France. After graduation, he worked in Lehman Brothers (Tokyo). A major part of his job was to structure products, which were finally sold to Hong Kong retail investors via local retail banks.

These products included equity-linked notes, minibonds (collateralized debt obligation [CDO] with total return swaps), and index-linked guaranteed notes. The equity-linked notes could provide an annual yield of 30 percent. Obviously the distribution of stock returns at that time was asymmetric with high upside potential and limited downside risk. The minibonds offered yields much better than bank deposits and the principle was guaranteed by an AA/A-rated institution—Lehman Brothers. This rating is better than that of many local banks.

Unfortunately, Lehman Brothers collapsed in September 2008. More than 40,000 retail investors in Hong Kong became victims. Some lost almost all their wealth. These victims continuously demonstrated in the street, at the front doors of various banks, and at the entrance of the Hong Kong Monetary Authority. Regulators encouraged banks to buy back the Lehman products. Banks were unwilling to do so. The Hong Kong banking industry experienced unprecedented exposure to reputational risk. In fact, this risk has never been discussed seriously and measured properly.

The Lehman incident made financial regulators extremely busy. Many of our financial engineering students are working for the regulatory bodies in Hong Kong. They were under serious pressure in the six-month period after September 2008. To mitigate regulatory risk, the regulators in Hong Kong announced a series of measures to prevent ordinary citizens from mistakenly buying high-risk products. These measures included mystery shopper programs (that is somebody pretending to be a bank client in order to test the selling process of frontline people) and audio-recording all relevant transactions. At the same time, the legal risk of banks intensified. Misrepresentation and insufficient duty of care became the words surrounding all financial institutions in Hong Kong. As a result, one of our authors was appointed to be an expert witness in some legal disputes relating to complex products. Risk management in banks suddenly became crisis management. Quantitative risk measures seemed less appealing.
1.3 RISK MANAGEMENT AND BASEL II

This book does not cover much about Basel II, which is the standard of risk management for the banking sector. There is a chapter about value-at-risk (VaR) and a chapter about probability of default (PD). Both VaR and PD are fundamental to bank capital charge. This book intends to share how complex financial products can be priced properly with simple programming tools. Asset pricing is a cornerstone of risk management. If an asset does not have any pricing model, we find it hard to measure its risk and evaluate its fair value. A pricing model facilitates scenario analysis: how much the asset will gain or lose in different scenarios, including some stress scenarios.

After the financial tsunami, Basel II has lost its credibility. Regulators obviously underestimated the impact of pro-cyclicality on credit risk. In 2002–2006, our university worked closely with the Hong Kong Monetary Authority to promote Basel II discussion in the Hong Kong banking sector. One of our authors was also an architect of the first internal-ratings-based system in Hong Kong. Basel II did help banks save capital charge. This could be an incentive for banks to invest heavily in risk management systems. This is also a reason why banks were undercapitalized in the crisis.

Basel II imposes capital requirements on market risk, credit risk, and operational risk. However, the interrelationship of these three risks has not been considered seriously. The VaR methodology assumes normal distribution of asset returns. Many credit-linked products, such as CDOs, collateralized mortgage obligations (CMOs), and others, are marketable securities subject to both interest rate risk plus credit migration risk. Actual or expected increase in credit risk can substantially lower asset prices. It seems that the Basel II capital requirement does not adequately address this issue. How should the correlation of credit risk and market risk be modeled? That is beyond the scope of this book.

Liquidity risk and stress testing risk are key issues in the collapse of banks. These risks are covered in Pillar II of Basel II. How can liquidity risk be modeled? Excel and VBA may help, but there is no consensus on what assumptions should be adopted. Stress testing usually involves many assumptions and a subjective selection of scenarios. Stress tests can be easily done and regulators usually find it hard to challenge those test results.

1.4 ABOUT THE BOOK

The main topic of this book is the practical implementation of financial models using Excel and VBA programming. Too often, books on
spreadsheet modeling provide only quick-and-dirty implementations of financial models that have very little use in real-world applications. This book focuses on the programming practices and skills to perform real-world implementation of financial models that are robust, reusable, and flexible. It takes an in-depth look at how to implement financial models using both Excel and VBA, and discusses the essential programming practices and skills in structuring complex financial models through advanced VBA features. It provides comprehensive coverage of financial models in the areas of derivatives pricing, market and credit risk modeling, and advanced interest rate modeling. Each of the later chapters on model implementation starts with a review of all the necessary financial theory and concepts from a practitioner’s perspective. Step-by-step instructions on the implementation are then provided to explain the programming techniques involved for models with different complexities. Alternative approaches are also discussed to enable readers a comprehensive understanding of different techniques.

This book is suitable for those who have solid backgrounds in financial engineering, financial modeling, and financial risk management; a master’s degree in financial mathematics, financial engineering, or computational finance is preferable. CFA, FRM, or professional risk manager (PRM) qualifications will be helpful to readers, but these readers must have prior training in calculus and matrix algebra. When we wrote this book, we surveyed books with relevant titles. None of them were advanced enough for our MSc (Financial Engineering) students. Most books with titles such as *Financial Modeling, Excel Modeling in Finance*, or *Spreadsheet Modeling in Finance* are targeted at undergraduate students in Finance or MBA students. Our book is targeted at financial engineering or mathematical finance students at business schools or engineering schools.

The book title “Financial Computing” is modified from “Computational Finance.” When our MSc (Financial Engineering) program was first launched in the 1990s, a number of professors from Carnegie Mellon University (CMU) served as our program advisors and teaching fellows. CMU offers a well-known program—MSc (Computational Finance). Computational Finance focuses on financial models that are based on mathematical theories and computational intelligence. Our book places less emphasis on financial models although we provide brief summaries on the theories mentioned in the book. We place more emphasis on how to implement these advanced models with Excel and VBA programming. This helps quantitative analysts quickly develop some models for their analytical work. This is the reason we named the book “Financial Computing” instead of “Computational Finance.” Our book covers a small number of well-known models and illustrates how Excel and VBA programming can be applied to implement these models. Through these models, readers can
pick up Excel and VBA skills easily and apply these skills to other complex models. We believe that the book will be a good companion to any degree program in financial engineering or financial mathematics.

1.5 CHAPTER HIGHLIGHTS

Chapter 2 deals with the GARCH(1,1) model, which is used to predict the volatility of asset prices. Volatility estimates are critical for derivatives pricing and the volatility index can be traded. We introduce an effective way to use Solver in conjunction with VBA routines to enhance the functionality of Solver. Chapter 3 looks at the finite difference model, which is frequently used in derivatives pricing based on the Black–Scholes partial differential equation. We discuss the use of matrix manipulation under Excel as well as the VBA programming environment. A general framework that may be used to price a variety of options is formulated. Chapter 4 turns to portfolio mean-variance optimization. This is the base of modern investment theory and investment portfolio formation. We pay particular attention to the implementation of the Markowitz algorithm under short-selling restrictions. In all these chapters, we discuss the deficiency in taking a simple Excel implementation and demonstrate the necessity of using VBA programming in efficiently coping with complex conditions.

Chapter 5 introduces the Newton–Raphson method. This numerical procedure is powerful in solving a system of equations, and the routine developed here will be useful throughout the book. Chapter 6 discusses yield curve construction with cubic spline interpolation. We describe a generalized bootstrapping method, a computer-intensive statistical method, in the construction of a smooth yield curve given any available data set of bond prices. This enables the construction of an interest rate tree discussed in later chapters.

Chapters 7 and 8 deal with two different tree models in option pricings: the binomial model and the Black–Derman–Toy model. The binomial model can be applied to a wide range of equity derivatives. It can be implemented very easily using VBA programming. The Black–Derman–Toy model is particularly useful for pricing interest rate derivatives. We introduce an effective way to implement this model in VBA taking bond options as our working example.

Chapter 9 discusses option pricing using the Monte Carlo simulation method, which is a powerful tool in the valuation of exotic options with complex payoff conditions. We discuss various important issues regarding this method and look at the implementation for a number of exotic options.
In particular, we take a closer look at the Monte Carlo pricing of American-style options with early exercising features.

Chapter 10 applies simulation techniques to determine portfolio value-at-risk. This chapter aims at providing the necessary programming skills to build a flexible and expandable risk engine for portfolio risk simulation.

Chapter 11 looks at the state-of-the-art Hull–White model of interest rates, which is commonly adopted by the industry for pricing interest rate derivatives. We discuss an effective way to implement the complex structure of this model taking bond options again as an example.

Chapters 12 and 13 discuss two well-known credit risk models: the CreditMetrics model and the KMV–Merton model. We start the discussion of the CreditMetrics model with a single issuer and then move to credit migration risk of credit portfolios. Chapter 12 focuses on the implementation of the credit RiskMetrics framework with the use of Monte Carlo simulation. In Chapter 13 we introduce the structural model developed by Robert C. Merton and extend our discussion to the KMV–Merton model. The KMV–Merton model is best applied to publicly traded firms and its underlying methodology predicts the probability of default of a firm within a given time horizon.

Appendices A to G provide a review of Excel and VBA programming. Many engineering school graduates may be familiar with Fortran, C, or Java and seldom touch Excel or VBA. The appendices will help these readers.

In all chapters, mathematical models are briefly mentioned. Our focus is to share with readers how to write relevant VBA programs. There is no standard programming route for a single problem. Readers may find faster programming methods to achieve the same outcome. These readers are welcome to contact us and share your better approaches with us. Practical exercises are provided at the end of each chapter that allow the readers to apply their technical skills acquired from the chapter. The solutions to these questions can be downloaded through the ftp link given by http://www.cs.cityu.edu.hk/~donny/humphrey/financial_computing.

1.6 OTHER REMARKS

We would like to thank our students in Hong Kong for asking us challenging questions in class. This helps improve our thinking and sharpen our teaching performance. Among all the authors, Dr. Humphrey Tung contributed the most. He carefully reviewed every equation in the book. The other three authors would like to thank him for his passion in this project.
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. The distinctive feature of this 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:

$$ r_t = \epsilon_t \mu, \sigma_t $$  \hspace{1cm} (2.1)

with dynamical volatility $\sigma_t$ and constant mean $\mu$. The model attempts to keep track and forecast the variations in the volatility through time. Applications of this so-called GARCH (generalized autoregressive conditional heteroscedasticity) volatility are widespread especially in the assessment of portfolio risk exposure over a short period of time.

In GARCH(1,1), future variance $\sigma_{t+1}^2$ is a weighted average of its immediate past estimation $\sigma_t^2$, 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:

$$ \sigma_{t+1}^2 = \gamma V_L + \alpha (r_t - \mu)^2 + \beta \sigma_t^2 $$  \hspace{1cm} (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 $\mu$ 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,

$$ V_L > 0, \alpha > 0, \beta > 0, \text{ and } \alpha + \beta < 1. $$  \hspace{1cm} (2.3)
They can be estimated under the notion of maximum likelihood of seeing the historical data. Given the historical time series of price returns \( \{r_1, r_2, \ldots, r_n\} \), we can first estimate the constant mean historically as:

\[
\mu \equiv \frac{1}{n}(r_1 + \ldots + r_n).
\]

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 \( \sigma_1^2 \equiv (r_1 - \mu)^2 \). According to the random normal assumption in equation (2.1), the likelihood or chance of the entire historical data set being observed is proportional to:

\[
L \propto \exp\left\{ -\frac{1}{2}(r_1 - \mu)^2 / \sigma_1^2 \right\} \times \ldots \times \exp\left\{ -\frac{1}{2}(r_n - \mu)^2 / \sigma_n^2 \right\}.
\] (2.4)

The best model parameters should therefore generate the volatilities \( \{\sigma_1, \sigma_2, \ldots, \sigma_n\} \) that maximize the likelihood \( L \) in (2.4) or equivalently the logarithm of likelihood \( \ln(L) \) given by:

\[
\ln(L) = -\frac{1}{2} \sum_{t=1}^{n} \left[ \ln(\sigma_t^2) + \frac{(r_t - \mu)^2}{\sigma_t^2} \right].
\] (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 above calculation could be organized in an Excel spreadsheet.\(^2\) The table analyzes daily returns of the Dow Jones Industrial Average (DJI) between March 22, 1990 and December 6, 2006. 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 closing of the DJI on each of these dates, while column C calculates the corresponding daily returns. For example, the formula adopted in C14 = (B14 / B13) - 1. 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 \( \mu \) in the model. Trial values of the model parameters (\( V_L, \alpha, \beta \)) 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).

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 \(\sigma_t^2\) are recorded in the fifth column from E14. They are generated iteratively using the formula (see equation \([2.2]\)):

\[
E15 = \left(1 - \alpha - \beta\right) \times \text{longvar} + \alpha \times D14^2 + \beta \times E14
\]

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:

\[
F14 = (\ln(E14) + D14^2/E14)
\]

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

Here, we are interested in choosing \((\nuL, \alpha, \beta)\) that maximize \(\ln(L)\) under the constraints in (2.3). Such a task can be achieved by using the Solver algorithm in Excel. We can simply go to Tools, then Solver, and the Solver Parameters screen will pop up as shown in Figure 2.2. Set Target Cell is the cell F10 that is the likelihood value \(\ln(L)\), check Equal To as Max for maximizing, and input the cells F5:F7 in By Changing Cells for the trial values of \(\nuL, \alpha, \) and \(\beta\).
The constraints in (2.3) can easily be included in the Solver algorithm under the Subject to the Constraints field. Click Add and, as shown in Figure 2.3, enter the following through the Add Constraint screen:

\[
\text{longvar} \geq 0, \quad \alpha \geq 0, \quad \beta \geq 0, \quad \text{and } F8 < 1
\]

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 \(\alpha\) and \(\beta.\) Note that Solver provides only the choices "\(\geq\)" and "\(<\)" for our purpose. Under a 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.

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 a maximum running time in Max Time, the number of iterations in the search in Iterations, and in Precision the required precision in the cells F5:F7 in By Changing Cells.

**Figure 2.2** Solver Parameters screen.

**Figure 2.3** Add Constraint screen.
Depending on the problem, this represents only the maximum achievable precision if there is enough 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 a maximum likelihood of $F10 = 4374.46820612$.

How good is the GARCH(1,1) model with these optimal parameters? To answer this question, we will backtest the model with out-of-sample historical data. In columns A and B we have included historical closings of the DJI up to the trading day 20061206 with about 3,000 backtesting 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 $\sigma^2_t$ will allow us to forecast the new variance $\sigma^2_{t+1}$ 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 backtest 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:

\[ C_8 = \text{NORMSDIST}(C7) - \text{NORMSDIST}(-C7). \]

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:

\[ G_{14} = \text{IF}(\text{AND}(C_{14} <= \text{mu} + \text{zvalue} * \text{SQRT}(E_{14}), C_{14} >= \text{mu} - \text{zvalue} * \text{SQRT}(E_{14})), 1, 0). \]

The cell C9 = \text{AVERAGE}((\text{OFFSET}(\text{INDIRECT}(C6),0,4))) then accumulates the backtesting 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 a detailed discussion of 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 (see Figure 2.5).

Depending on the initial value of \( x, \) Solver determines only the nearest maximum or minimum point that is not necessarily the overall solution. In particular, if the initial value is chosen to be \( x = 1.2, \) the
In general, there is no \textit{a priori} information on the model parameters in GARCH(1,1). A preliminary procedure for the purpose of determining the 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 an 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.

\textbf{2.3 EXCEL PLUS VBA IMPLEMENTATION}

In general, there is no \textit{a priori} information on the model parameters in GARCH(1,1). A preliminary procedure for the purpose of determining the 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 an 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 \((V_L, \alpha, \beta)\). 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:

\[
\text{TotalLikelihood}(h(1:n), n, \alpha, \beta, V_L)
\]

```plaintext
# define \( \nu_1 = \sigma_1^2 \) to start off the iteration and accumulate \( \ln(L) \)
\( \nu(1) = h(1) \)
\( \text{sum} = \ln(\nu(1)) + h(1)/\nu(1) \)

# generate \( \{\nu_2 = \sigma_2^2, \ldots, \nu_n = \sigma_n^2\} \) by iteration and accumulate \( \ln(L) \)
For (\( i = 1 \) to \( n - 1 \)) { 
    \( \nu(i+1) = (1 - \alpha - \beta)V_L + \alpha h(i) + \beta \nu(i) \)
    \( \text{sum} = \text{sum} + \ln(\nu(i+1)) + h(i+1)/\nu(i+1) \)
}

TotalLikelihood = -\frac{1}{2} \text{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 the largest \( \ln(L) \), which utilizes the above likelihood function. Define in cell E2 the required precision \( \text{prec} \) of the parameters in this preliminary procedure. According to the constraints in (2.3), the search for a proper \( \alpha \) and \( \beta \) should run through all points given by the double-loop as:

\[
\begin{align*}
\alpha &= i \times \text{prec}, \quad i = 1, 2, \ldots, N - 1 \\
\beta &= j \times \text{prec}, \quad j = 1, 2, \ldots, N - i - 1
\end{align*}
\]

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

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

In practice, the search for a proper \( V_L \) can be confined within the region

\[
0.8 V_{\text{historical}} \leq V_L \leq 1.2 V_{\text{historical}}
\]

and run through the loop as:

\[
V_L = (k \times \text{prec}) V_{\text{historical}}, \quad k = k_{\text{low}}, \ldots, k_{\text{high}}
\]

\(2.7\)
where \( k_{low} \) and \( k_{high} \) are the nearest integers to 0.8\( N \) and 1.2\( N \), 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 \((V_L, \alpha, \beta)\) taken to be the initial values for Solver. We can set the precision to 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()  

\[
\text{# input historical price returns and precision parameter}
\text{Read } n, r(1:n), \text{ and } prec
\]

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

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

\[
\text{# determine the number of grids given precision}
N = \text{Int}(1/prec)
\]

\[
\text{# scan through the valid region of the parameters for the largest } \ln(L)
maxlnL = -10^8
\]

\[
\text{For ( } i = 1 \text{ to } N - 1 \{ }
\text{\quad For ( } j = 1 \text{ to } N - i - 1 \{ }
\text{\quad\quad For ( } k = \text{Int}(0.8N) \text{ to } \text{Int}(1.2N) \{ }
\quad\alpha = i \text{ prec } , \quad \beta = j \text{ prec } , \quad V_L = k \text{ prec } V_{historical}
\lnL = \text{TotalLikelihood}( h(1:n) , n , \alpha , \beta , V_L )
\text{\quad If ( } \lnL \geq maxlnL \{ }
\text{\quad\quad maxlnL = } \lnL
\text{\quad\quad bestL = } \alpha , \quad best\beta = \beta , \quad bestV_L = V_L
\text{\quad\quad \}} \}
\}
\]

\[
\text{# output best model parameters}
\text{Output } bestV_L , \text{ best} \alpha , \text{ and } best\beta
\]

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 so that it can be maintained easily. The first statement will display a message box asking for confirmation to start the calculation. It is always a good practice to include a message box to avoid mis-invoking a 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 \((V_L, \alpha, \beta)\) 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, as well as 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 configured 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 on References in the Tools menu and checking Solver in the Reference dialogue. The TRUE parameter of the SolverSolve function suspends the display of the resulting dialogue 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(Round(1/prec,15))
    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
End Sub