SOA Research Institute

Calibrating Interest Rate Models

October 2023

Calibrating Interest Rate Models

AUTHORS Rohana Ambagaspitiya, FSA, FCIA, PhD

Charles Ford, FSA, MAAA, CFA

SPONSOR Quantitative Finance and Investment Curriculum Committee

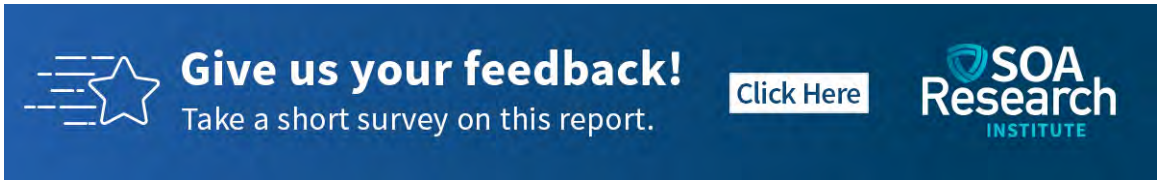


TABLE OF CONTENTS

Calibrating Interest Rate Models

Executive Summary

Investment Actuaries and now Valuation Actuaries need a mastery of stochastic interest rate models. This includes selecting a model appropriate for a given application, the correct use of real world versus risk neutral scenario sets, and how to calibrate and validate them properly.

Models like Vasicek and Cox-Ingersoll-Ross (CIR) are *stochastic* models that, when the parameters are set, determine the yield curve. They are appropriate when a stochastic model is needed and it is less important whether or not the yield curve implied by the model actually matches the "actual" observed yield curve. This setup is mainly used for US Statutory Valuation VM20, VM21 and Economic Capital calculations.

If the requirement is to price interest sensitive cash flows in a market consistent way, something else is needed. Both the CIR and Vasicek can be modified to get market consistent versions and obviously there are many other models beyond these. Examples of market-consistent applications are IFRS 17 valuation, measuring the cost of guarantees for universal life minimum interest rates, and measuring the cost of guarantees for participating insurance products. The following table summarizes what type of interest rate models should be used under different circumstances and to what they should be calibrated.

Table 1
TABLE TITLE OR DESCRIPTION

Purpose of the Interest Rate Generator	Interest Rate Model	Items Calibrated to
US Statutory Valuation Required Capital	Vasicek CIR	Historical rates, current yield curve, expert opinion, and/or regulatory criteria
Market Consistent Valuation Pricing of Options IFRS 17 valuation	2-factor Hull-White Lognormal Forward Model Extended CIR	Current Yield curve Current option prices

SOA QFI Curriculum

The existing literature presents the theory, but it can be challenging for the practitioner new to this specialty to take that theory, write code to implement it, use historical data to calibrate the model, and assess whether the resulting set of scenarios is reasonable.

This paper addresses this need by introducing practitioners to the selection and calibration of stochastic interest rate models. Six continuous time interest rate models and their calibration are presented. Actual code and data examples are added to help the practitioner implement them. The emphasis is on hands-on calibration with RStudio. The reader is assumed to have a working knowledge of RStudio, including writing R scripts.

The focus of this paper is narrow: calibrating and validating an interest rate model for default-free bond yields in a single economy. Out of scope are inflation and equity index variables, credit spreads, and factors for multiple economies such as foreign exchange.

With this focus the practitioner may begin their journey in stochastic interest rate modeling.

1 Introduction

In this paper we present calibration of six continuous time interest rate models. We use the notation in [24] and cite relevant chapters throughout the report. The emphasis is on hands-on calibration with RStudio. We assume the reader has some working knowledge of RStudio, including writing R scripts.

The focus of this paper is narrow: calibrating and validating an interest rate model for default-free bond yields in a single economy. These are outlined at a higher level in [1]. This paper adds actual code and data examples to help the practitioner implement these ideas. Out of scope are inflation and equity index variables, credit spreads, and factors for multiple economies such as foreign exchange.

In this paper the acronym ESG refers to economic scenario generators and not to environmental, social, and governance aspects of investing.

1.1 R SETUP

R is used for examples. The RStudio environment for working with R will be necessary for the reader to follow along and experiment with the code presented in this paper. If RStudio has not been downloaded and installed, now would be a good time to do so.

The R functions and examples given in this paper are available as a bundled package on the SOA website at https://www.soa.org/resources/research-reports/2023/interest-rate-model-calibration-study/ You may install "InterestCalibrationv1_1.2.0.tar.gz" in RStudio using "Tools → Install packages → Install from: → Package Archive". When you extract the package archive, its subfolders "R" and "examples" contain all the functions and examples respectively. However, readers are encouraged to key in the code chunks instead of copy/pasting them.

R is both a software environment and a scripting language, and it may interpret certain manipulations in an unintended way. When a new session is begun it is good idea to either restart R or clean the workspace using:

```
#example1
rm(list=ls()) # clear the work space and functions
graphics.off() # clear the plots

library(InterestCalibrationv1) # load the developed library
library(matrixStats) # load the library to calculate summary statistics
library(nlsr) # load the newest non-linear-least saure package
library(YieldCurve) # load the yield curve package.
options(digits=5) # this for printing
```

1.2 INTRODUCTION TO SCENARIOS

This paper explores the nuances of calibrating continuous-time interest rate models. It is not an introduction to economic scenario generators (ESGs). In actual practice there are several steps to the process of setting up a model to run multiple, stochastically generated interest rate scenarios:

- The application for the modeling work should be clear to practitioners
  - Derivative valuation, valuing financial guarantees such as insurance business pricing, financial planning, and economic capital or tail risk analysis
- Specify "stylized facts"
  - Clarify an understanding of how economic variables are expected to evolve over time and in what circumstances the relationships between them change, such as during a recession
- Source and analyze historical data
- Calibrate the model
- Run the generator
- Confirm that the resulting scenarios are fit for purpose. How well do they conform to the pre-determined stylized facts? Are changes necessary?
- Run the model of the security or business, iterating through each scenario
- Validate the model results
- Document each step so that:
  - choices made can be explained in management presentations and regulatory filings
  - the whole process can be run in an efficient, consistent manner when next required

This broader context is examined in Economic Scenario Generators: A Practical Guide, (SOA, 2016) (the "Practical Guide")[1]. This paper will only comment briefly on these points and suggest initial choices to provide context and a pathway through this process. Experienced practitioners will have reasons to make different judgements on these issues.

1.3 INTEREST RATE MARKETS

This paper assumes a context of US fixed income markets. Some other sovereign bond markets have the trading liquidity and full term structure seen in the US Treasury bond market. Practitioners in other markets are encouraged to consult with a bond analyst or trader for local market characteristics, such as liquidity and yield quoting conventions.

A word about U.S. Treasury yields. The U.S. Department of the Treasury issues intermediate and longer maturity notes (to 10 years) and bonds (longer than 10 years) every few months. Loosely speaking they are all often called "bonds", which we will do here. Consider a newly issued 10-year bond, which is said to be "on-the-run". A 30-year bond issued 20 years ago also has 10 years until maturity, but it is said to be "off-the-run". Trading in it is less liquid and it may trade for a couple of basis points higher yield.

As time passes that newly issued 10-year bond will no longer have 10 years until maturity, but rather 9 years 11 months, 10 months, etc. In an upwardly sloping yield curve environment these trade at a slightly lower yield than a true 10-year note. The Treasury Department does a curve fitting exercise every trading day to calculate that day's closing yield for a true 10-year bond, and other key maturities as well. These are

said to be "constant maturity treasury" (CMT) yields[1] and are often used for modeling, even though they are not actually tradable securities. CMT yields and discussion can be found by doing an internet search for "US Treasury CMT rates".

Six months after issue a new 10-year note is issued and becomes the on-the-run 10-year bond. The first bond now has just 9.5 years until maturity, and is said to be off-the-run. Trading liquidity will shift to the new on-the-run note.

1.4 MARKET-CONSISTENT MODELS

Before a model can be calibrated it must be selected. To select an appropriate model the practitioner must be clear on the uses to which the model will be applied.

One class is for realistic simulation of how something – an asset, a financial liability, or a business segment with both – evolves over time. By "realistic" we mean plausible given an accepted set of stylized facts about how market participants make decisions and how interest rates "should" evolve over time. These are discussed in detail in [1], chapter 6.

A different problem is to determine a value for a security that is not frequently traded. A model can be calibrated to observed market prices of similar securities and then used to calculate (estimate) values for illiquid securities.

Either problem can be solved with a model and an economic scenario generator. If equity index variables and economic variables such as inflation need not be modeled then those factors may be set to zero, one, or otherwise not used, leaving the interest rate model the only active part of the ESG. The generator is then set up with initial conditions such as observed yields and market prices as of the desired model start date. The generator has an interest rate model component which evolves yield curves forward one time step at a time to produce one interest rate scenario. This step is repeated 1,000, 10,000, or more times to generate a set of interest rate scenarios. For each scenario, a value is calculated for each security.

These values do not automatically reproduce observed market prices at the model start date. The model must be "calibrated", that is, its parameters must be adjusted to match specific criteria.

If the task at hand is to calculate prices for infrequently traded securities that are reasonably consistent with the observed market prices of frequently traded securities, then the calibration adjusts the parameters to reproduce those prices. Such a calibration is "market-consistent". The adjusted, or calibrated, scenario set may then be used to calculate estimated market values for similar non-traded securities. It should not be used for dissimilar securities.

Note that so far nothing has been said about risk neutral or real world.

1.5 THE MARKET PRICE OF RISK

One stylized fact is that market participants are risk averse. Market prices embed a factor for this called the "market price of risk".

---

[1] https://home.treasury.gov/policy-issues/_nancing-the-government/interest-rate-statistics/interest-rates-frequently-asked-questions

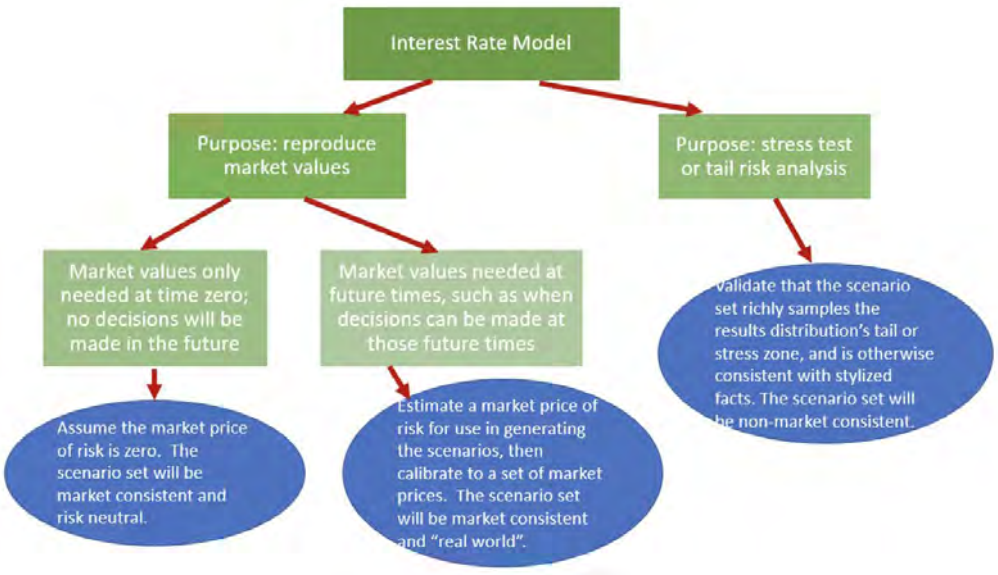
If the scenario set is only being used to value securities at the model start date, a useful mathematical tool is to assume the market price of risk is zero. The calibration step can still find parameters to match the observed securities prices, but it does so by adjusting the probability of each scenario occurring.

This is useful because the market price of risk can be estimated but not directly observed in the market. With the assumption of a zero market price of risk, the calibrated scenario set is said to be "risk-neutral." It is also market-consistent because it reproduces the observed market prices of the selected securities at the specified date. The trade-off is that this scenario set can no longer provide reliable information about future time steps or period; it is not fit for such a purpose.

It is also possible to choose a market price of risk from historical data. As with the other parameters this would need to fit the stylized facts and interact with other parameters in a reasonable way. The model calibration can then either fit the scenarios to observed market prices so as to be market consistent, or left in an equilibrium state. The latter approach might be desirable for stress testing or tail risk analysis. The former approach is necessary for estimating values of similar securities ("valuation").

Why might the former approach of setting a historically based market price of risk and calibrating to observed market prices be preferred to assuming a zero market price of risk (the risk-neutral approach)? When the item being modeled has complex future interactions, such as modeling a hedging program or embedded options in new business in a financial plan model, having scenarios and model outcomes that are intuitively understandable and can be explained to stakeholders beyond the modeling team may be important for credibility and influencing decisions. Silly-looking scenarios come with risk-neutral; they're part of the deal.

More complete discussions can be found in [23] for the latest research as well as the classic chapter 11 of [3].



1.6 PARAMETER UNCERTAINTY

Not only is the choice of the most appropriate model a matter of judgement, the parameters of the chosen model are uncertain as well. [4] studies the Wilkie model, which models inflation, dividend yields, and an equity index return, as well as a long-term interest rate. Parameter uncertainty is found to have a significant impact on the dispersion of these four parameters. The paper applies its findings to a block of annuity contracts, finding that the distribution of outcomes is significantly wider when the parameters are recognized as instances of random variables. Developing methods to apply this insight would be an avenue for further research.

2 Three Continuous-Time Interest Rate Models

In this section we provide a brief description of the three models that we consider. We consider three models: the Vasicek (one and two factor) models, Cox-Ingersoll-Ross (CIR) model, and the Hull-White models (one and two factor). The Vasicek and CIR models are called equilibrium models or real-world models. The equivalent risk-neutral models can be obtained by changing the drift of those models. We assume that a one factor continuous-time interest rate model for short rate r_t is an Itô process, i.e. r_t follows the stochastic differential equation (SDE)

$$dr_t = a(r_t)dt + b(r_t)dX_t$$

where $a(r_t)$ and $b(r_t)$ are functions of r_t and X_t is a standard Wiener process. For a formal definition of an Itô process and an SDE one may refer to [8]. Similarly, the two-factor models satisfy the following SDE

$$dr_t = a(r_t)dt + b_1(r_t)dX_{1,t} + b_2(r_t)dX_{2,t}\ ,$$

with $dX_{1,t}dX_{2,t} = \rho dt$

In statistical calibrations, the underlying assumption is that data follows the model that we try to fit. Therefore, to evaluate calibration techniques we need sample paths from the interest rate model. The only way to get sample paths that follow the underlying distribution with known parameters is Monte Carlo simulation. Therefore, it is important to have an accurate method of simulating sample paths from the given SDE with known parameters. There are number of techniques to generate sample paths from a given SDE and Chapter 2 of [17] is a good introduction. However, we present two methods: Euler-Maruyama discretization and the transition density method introduced in [11]. Note that Chapter 17 of [24] only discusses implementation of Euler-Maruyama method for simulating sample paths, however we demonstrate that this method is not suitable for practical applications.

2.1 THE VASICEK MODEL

In the Vasicek model the risk free rate of interest r_t is based on an SDE:

$$dr_t = \gamma(\bar{r} - r_t)dt + \sigma dX_t\ , \tag{1}$$

where X_t is the standard Wiener process and γ, $\bar{r}$, and σ are strictly positive parameters. The three parameters γ, $\bar{r}$, and σ have the following interpretation:

- $\bar{r}$ is the long term mean level; when the rate r_t drifts too much from $\bar{r}$, it will be pulled back closer to that level. This is called mean reversion.
- γ is the speed of mean reversion.
- σ is the instantaneous volatility.

The solution of the above SDE takes the following form:

$$r_{t+s} = r_t e^{-\gamma s} + \bar{r}(1 - e^{-\gamma s}) + \sigma e^{-\gamma s} \int_0^s e^{\gamma u} dX_u \,,$$

where r_t is the initial point (starting point) of the process and where s > 0.

Since $\int_0^s e^{\gamma u} dX_u$ is a normal random variable with mean zero and variance $\int_0^s e^{2\gamma u} du$, we see that $r_{t+s}|r_t$,where s > 0 (i.e. the conditional random variable of r_{t+s} conditioned on r_t), is normally distributed with mean $\mu(r_t, s)$ and variance $\sigma(s)^2$ which are given as below:

$$\mu(r_t, s) = \bar{r} + (r_t - \bar{r})e^{-\gamma s} \tag{2}$$

$$\sigma(s)^2 = \frac{\sigma^2}{2\gamma}(1 - e^{-2\gamma s}) \tag{3}$$

As the conditional distribution or_{t+s}f is normal, the probability of $r_{t+s} < 0$ can be calculated as below:

$$\begin{aligned} P[r_{t+s} < 0 | r_t] &= P\left(\frac{r_{t+s} - E[r_{t+s}|r_t]}{\sigma(s)} < -\frac{E[r_{t+s}]}{\sigma(s)}\right) \\ &= \Phi\left(-\frac{E[r_{t+s}]}{\sigma(s)}\right) \\ &= \Phi\left(-\frac{\bar{r} + (r_t - \bar{r})e^{-\gamma s}}{\sigma\sqrt{\frac{1-e^{-2\gamma s}}{2\gamma}}}\right) \end{aligned}$$

For example the set of values $(r_t = 0.01, \bar{r} = 0.05,\ \gamma = 0.5, \sigma = 0.02, s = 0.1)$ will yield $P[r_{t+s} < 0|r_t] = 0.02637$. This implies that there is a positive probability of negative interest rates occurring, which is generally seen as a drawback of this model (Japanese and European experience notwithstanding). However, we can minimize the chance of negative rates in simulation studies by choosing parameters appropriately.

2.1.1 SIMULATING PATHS OF THE VASICEK MODEL: EULER-MARUYAMA DISCRETIZATION

In this method we discretize the Vasicek model SDE in the following manner:

$$\begin{aligned} dr_t &= \gamma(\bar{r} - r_t)dt + \sigma dX_t \\ r_{t+\Delta} - r_t &= \gamma(\bar{r} - r_t)\Delta + \epsilon_{t+\Delta} \\ r_{t+\Delta} &= r_t + \gamma(\bar{r} - r_t)\Delta + \epsilon_{t+\Delta} \end{aligned}$$

where $\epsilon_{t+\Delta}$ is a normal random variable with mean 0 and variance $\sigma^2\Delta$. As dX_t is a standard Brownian motion we can write from the above:

$$r_{i\Delta} = r_{(i-1)\Delta} + \gamma\big(\bar{r} - r_{(i-1)\Delta}\big)\Delta + \epsilon_{i\Delta}, \qquad i = 1, 2, \ldots$$

From the properties of Brownian motion we know that $\epsilon_{i\Delta}, i = 1, 2, \ldots$ are independent identically distributed (i.i.d.) normal with mean zero and variance $\sigma^2\Delta$. For convenience let us write:

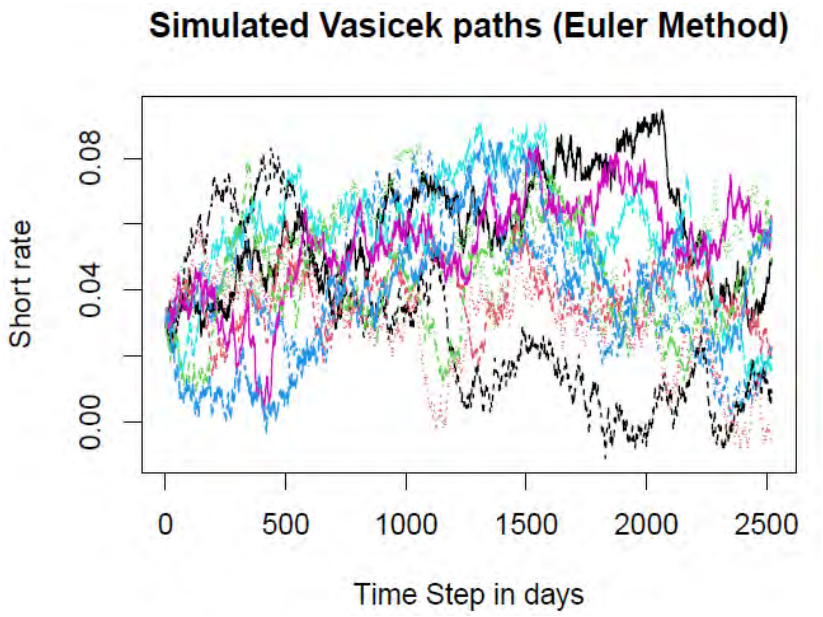
$$\begin{aligned} r(0) &= r_0 \\ r(i) &= r_{i\Delta}, \quad i = 1, 2, \ldots \\ \alpha &= \gamma \bar{r} \Delta \\ \beta &= 1 - \gamma \Delta \\ \sigma^* &= \sigma \sqrt{\Delta} \\ \epsilon_i &= \epsilon_{i\Delta} \end{aligned}$$

Then:

$$r(i) = \alpha + \beta r(i-1) + \epsilon_i, \qquad i = 1, 2, \ldots$$

with ϵ_i's that are i.i.d. normal with mean 0 and standard deviation σ^*. The R function "Vasicek.Euler.sim" in the "InterestCalibration" library implements this method. The following code snippet demonstrates use of "Vasicek.Euler.sim."

```
#example2
rm(list=ls()) # clear the workspace and functions
set.seed(123)
X=Vasicek.Euler.sim(Delta=1/252,M=10)
matplot(X,type="l",main="Simulated Vasicek paths (Euler Method)",
ylab="Short rate",xlab="Time Step in days")
```



2.1.2 SIMULATING PATHS OF THE VASICEK MODEL: TRANSITION DENSITY METHOD

The transition density method of simulating interest rate paths relies on the fact that the simulation can be carried out using the distribution of next observation r_{t+s} at *t+s* given observation r_t at time *t*. This method is introduced by [11]. This is an exact method of simulation as opposed to the Euler-Maruyama scheme which relies on discretizing an SDE. Since $r_{t+s}|r_t$ is normal with mean and variance as given in (2) and (3) respectively, the realized value of r_{t+s} is calculated from:

$$r_{t+s} = \bar{r} + (r_t - \bar{r})e^{-\gamma s}\left(\frac{\sigma^2}{2\gamma}(1 - e^{-2\gamma s})\right)^{\frac{1}{2}} Z$$

where Z is a standard normal random number. By writing:

$$\begin{aligned} r(0) &= r_0, \\ r(i) &= r_{i\Delta}, \quad i = 1, 2, \ldots, \end{aligned}$$

We can obtain:

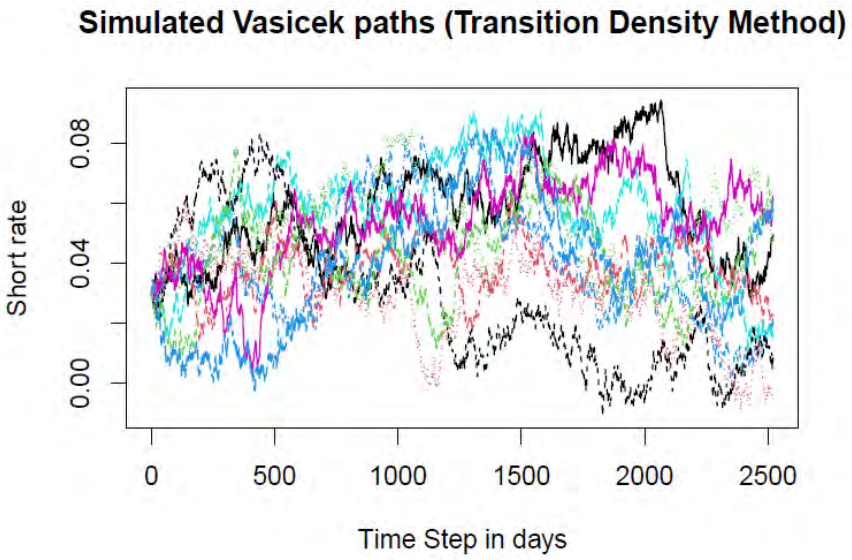
$$r(i) = \alpha^* + \beta^* r(i-1) + \epsilon_i, \qquad i = 1, 2, \ldots,$$

where:

$$\begin{aligned} \alpha^* &= (1 - \exp(-\gamma\Delta))\bar{r}, \\ \beta^* &= \exp(-\gamma\Delta), \\ \sigma^{**} &= \sqrt{\frac{\sigma^2}{2\gamma}(1 - e^{-2\gamma\Delta})}, \end{aligned}$$

with ϵ_i's that are i.i.d. normal with mean 0 and standard deviation σ^{**}. The R function "Vasicek.Trans.sim" in the "InterestCalibration" library implements this method. The following code snippet demonstrates use of "Vasicek.Trans.sim."

```
#example3
rm(list=ls()) # clear the workspace and functions
set.seed(123)
X =Vasicek.Trans.sim(Delta=1/252,M=10)
matplot(X,type="l",main="Simulated Vasicek paths (Transition Density Method)",
ylab="Short rate",xlab="Time Step in days")
```



2.2 COX-INGERSOLL-ROSS (CIR) MODEL

The SDE describing the CIR model is as follows:

$$dr_t = \gamma(\bar{r} - r_t)dt + \sqrt{\alpha r_t}dX_t \tag{4}$$

This model is also a mean reverting model and the parameters γ and $\bar{r}$ have the same interpretation as in the Vasicek model. One advantage of this model over the Vasicek model is that when $\gamma\bar{r} > \frac{\alpha}{2}$ the rates are always non-negative. This model was introduced in [13] and [10] used it for modelling interest rates.

With some difficult mathematics one can show that the transition density function of r_{t+s} conditioned on r_t is given by:

$$f(r_{t+s}|r_t) = c_s\chi^2(c_s r_{t+s}, \nu, \lambda_{t+s}) \tag{5}$$

where $\chi^2(.,\nu,\lambda_{t+s})$ is a non-central χ^2 density function with ν degrees of freedom and non-centrality parameter λ_{t+s} , with:

$$\begin{aligned} c_s &= \frac{4\gamma}{\alpha(1-\exp(-\gamma s))} \\ \nu &= \frac{4\gamma}{\alpha}\bar{r} \\ \lambda_{t+s} &= c_s r_t \exp(-\gamma s) \end{aligned}$$

A relatively easy way to visualize a characterization of the non-central χ^2 random variable with an integer-valued degrees of freedom parameter (i.e. ν is an integer) is that it can be obtained by summing up squares of ν independent normal random variables with non-zero means and unit variances. However, in our case ν may not be an integer.

Another characterization of the non-central χ^2 distributions is that they can be represented as a mixture of Poisson and central χ^2 distributions. As given on page 436 in [18], we can write the cdf, $F(x;\nu,\lambda)$, of a non-central χ^2 with degrees of freedom ν and non-central parameter λ as:

$$F(x;\nu,\lambda) = \sum_{j=0}^{\infty}\left[\frac{\left(\frac{\lambda}{2}\right)^j}{j!}\exp(-\lambda/2)\right]F(x;\nu+2j,0) \tag{6}$$

Although the above expression involves an infinite sum, it leads to easy generation of random numbers from a non-central χ^2. It involves first generating a random number J from a Poisson distribution with mean $\lambda/2$ and then generating a random number from a central χ^2 with degrees of freedom $\nu + 2J$; this random number will be from the non-central χ^2.

2.2.1 SIMULATING PATHS OF THE CIR MODEL: EULER-MARUYAMA DISCRETIZATION

As in the Vasicek case we can discretize the SDE in the following manner:

$$dr_t = \gamma(\bar{r} - r_t)dt + \sqrt{\alpha r_t}dX_t$$

$$r_{t+\Delta} - r_t = \gamma(\bar{r} - r_t)\Delta + \sqrt{r_t}\epsilon_{t+\Delta}$$

where $\epsilon_{t+\Delta}$ is a normal random variable with mean 0 and variance $\alpha\Delta$.

As in the Vasicek model we can further simplify and obtain:

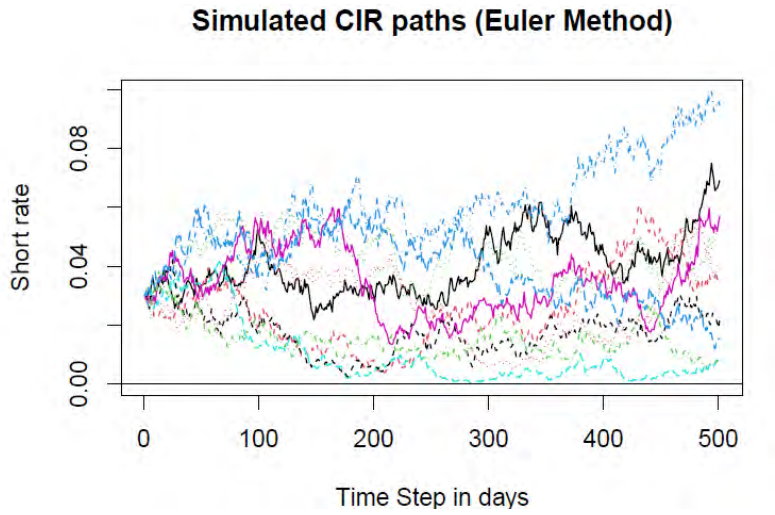
$$r(i) = \alpha_1 + \beta_1 r(i-1) + \sqrt{r(i-1)}\epsilon_i, i = 1, 2, \ldots,$$

where:

$$\begin{aligned} r(0) &= r_0 \\ r(i) &= r_{i\Delta}, \quad \epsilon_i = \epsilon_{i\Delta}, \quad i = 1, 2, \ldots \\ \alpha_1 &= \gamma\bar{r}\Delta \\ \beta_1 &= 1 - \gamma\Delta \\ \sigma &= \sqrt{\alpha\Delta} \end{aligned}$$

with ϵ_i's that are i.i.d. normal with mean 0 and standard deviation σ. The R function "CIR.Euler.sim" in the "InterestCalibration" library implements this method. The following R-code snippet illustrates the usage of "CIR.Euler.sim."

```
#example4
rm(list=ls()) # clear the workspace and functions
set.seed(123)
X =CIR.Euler.sim(Delta=1/500,M=10)
matplot(X,type="l",main="Simulated CIR paths (Euler Method)",
        ylab="Short rate",xlab="Time Step in days")
abline(0,0)
```



As we can see from the graph, the Euler-Maruyama discretization method leads to negative values for r_t which is theoretically impossible. Note that we can calculate the probability of $r(i)$ becoming negative conditioned on $r(i-1)$ for $i = 1, 2, \ldots$ as given below:

$$\begin{aligned} E[r(i)|r(i-1)] &= \alpha_1 + \beta_1 r(i-1) \\ Var[r(i)|r(i-1)] &= Var[\sqrt{r(i-1)}\epsilon_i] \\ &= r(i-1)\sigma^2 = r(i-1)\alpha\Delta \end{aligned}$$

Under the Euler-Maryuma scheme the conditional distribution of $r(i)$ conditioned on $r(i-1)$ is normal with mean $E[r(i)|r(i-1)]$ and variance $Var[r(i)|r(i-1)]$, the probability of $r(i)$ becoming negative conditioned on $r(i-1)$ is:

$$\begin{aligned} P[r(i)<0|r(i-1)] &= \Phi\left(-\frac{E[r(i)|(r(i-1))]}{\sqrt{Var[r(i)|(i-1)]}}\right) \\ &= \Phi\left(-\frac{\alpha_1+\beta_1 r(i-1)}{\sqrt{r(i-1)\sigma^2}}\right) \end{aligned}$$

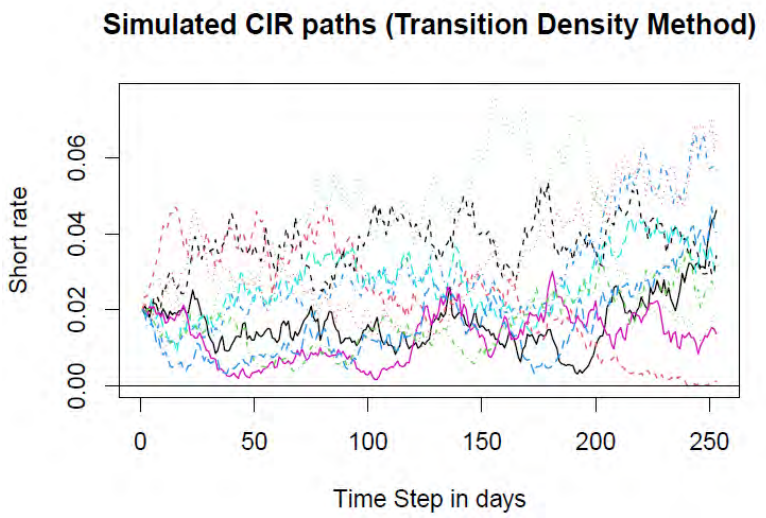
For example, the set of values $(r(i)=0.2\%, \bar{r}=7\%, \gamma=0.3262, \alpha=0.0221, \Delta=1/252)$ will yield $P[r(i)<0|r(i-1)]=0.00108$. Therefore, the Euler-Maruyama method should be used with caution, though it does have the appeal of simplicity.

2.2.2 SIMULATING PATHS OF THE CIR MODEL: TRANSITION DENSITY METHOD

From the discussion above the conditional distribution of $\frac{r_{t+s}}{c_s}$, conditioned on r_s, is distributed as a non-central χ^2 with degrees of freedom $\nu=\frac{4\gamma}{\alpha}\bar{r}$ and non-centrality parameter $\lambda_{t+s}=c_s r_t exp(-\gamma s)$. The algorithm uses r_t's value to generate r_{t+s}. Since we use $s=\Delta$, a constant value, at each step the non-centrality parameter $\lambda_{i\Delta}=c_\Delta r_{i\Delta} exp(-\Delta)$ has to be evaluated using $r_{(i-1)\Delta}$ before calculating $r_{i\Delta}$ for $i=1,2,\ldots$.

The R function "CIR.Trans.sim" in the "InterestCalibration" library implements this method. The following R-code snippet illustrates the usage of "CIR.Trans.sim."

```
#example5
rm(list=ls()) # clear the workspace and functions
set.seed(123)
X =CIR.Trans.sim(Delta=1/252,M=10)
matplot(X,type="l",main="Simulated CIR paths (Transition Density Method)",
        ylab="Short rate",xlab="Time Step in days")
abline(0,0)
```



In the implementation we assumed that the non-central χ^2 random number generation function "rchisq" works perfectly. However, "rchisq" uses the mixture of Poisson and central χ^2 characterization given in (6) to generate random numbers. The method is simple and straightforward but "Simulation of Square-root Processes" in [9] lists many drawbacks of this method and for practical applications one may use a method given in [12] and [2] to simulate non-central χ^2 random numbers.

2.3 THE TWO-FACTOR VASICEK MODEL WITH CORRELATED FACTORS

This model can be written as the short rate process as of the sum of two factors, a short rate factor and a long rate factor:

$$r_t = \phi_{1,t} + \phi_{2,t}$$

Where each factor follows the following SDEs:

$$\begin{aligned} d\phi_{i,t} &= \gamma_i(\overrightarrow{\phi}_i - \phi_{i,t})dt + \sigma_i dX_{i,t}, \quad i = 1, 2, \\ dX_{1,t}dX_{2,t} &= \rho dt \end{aligned}$$

With a few lines of algebra we can obtain the solution of these SDEs as follows:

$$\phi_{i,t+s} = \phi_{i,t}\exp(-\gamma_i s) + \overrightarrow{\phi}_i(1 - \exp(-\gamma_i s)) + \sigma_i \exp(-\gamma_i s)\int_0^s \exp(\gamma_i v)dX_{i,v}, \quad i = 1, 2.$$

The conditional means and variance of these two factors are given by:

$$\begin{aligned} E[\phi_{i,t+s}|\phi_{i,t}] &= \phi_{i,t}\exp(-\gamma_i s) + \overrightarrow{\phi}_i(1 - \exp(-\gamma_i s)), i =, 1, 2 \\ Var[\phi_{i,t+s}|\phi_{i,t}] &= Var\left[\sigma_i \exp(-\gamma_i s)\int_0^s \exp(\gamma_i v)dX_{i,v}\right] \\ &= \sigma_i^2 \exp(-2\gamma_i s)Var\left[\int_0^s \exp(\gamma_i v)dX_{i,v}\right] \\ &= \sigma_i^2 \exp(-2\gamma_i s)\int_0^s \exp(2\gamma_i v)dv \\ &= \frac{\sigma_i^2}{2\gamma_i}(1 - \exp(-2\gamma_i s)), i = 1, 2 \end{aligned}$$

Where the second to last equation follows from Itô's isometry. The conditional covariance between $\phi_{1,t+s}$ and $\phi_{2,t+s}$ conditioned on $\phi_{i,t}$, $i = 1, 2$ is given by:

$$\begin{aligned} Cov[\phi_{1,t+s}, \phi_{2,t+s}|\phi_{1,t}, \phi_{1,t}] &= E\left[\sigma_1 \exp(-\gamma_1 s)\int_0^s \exp(\gamma_1 v)dX_{1,v}\sigma_2 \exp(-\gamma_2 s)\int_0^s \exp(\gamma_2 u)dX_{2,u}\right] \\ &= \rho\sigma_1\sigma_2 \exp(-(\gamma_1 + \gamma_2)s)\int_0^s \exp(-(\gamma_1 + \gamma_2)v)dv \\ &= \frac{\rho\sigma_1\sigma_2}{\gamma_1 + \gamma_2}(1 - \exp(-(\gamma_1 + \gamma_2)s)) \end{aligned}$$

The conditional correlation coefficient between $\phi_{1,t+s}$ and $\phi_{2,t+s}$, which can be denoted as $\rho(s)$, is given by:

$$
\begin{aligned}
Corr[\phi_{1,t+s}, \phi_{2,t+s} | \phi_{1,t}, \phi_{1,t}] &= \frac{\frac{\rho\sigma_1\sigma_2}{\gamma_1+\gamma_2}(1-\exp(-(\gamma_1+\gamma_2)s))}{\sqrt{\frac{\sigma_1^2}{2\gamma_1}(1-\exp(-2\gamma_1 s))\frac{\sigma_2^2}{2\gamma_2}(1-\exp(-2\gamma_2 s))}} \\
\rho(s) &= \rho\left(\frac{4\gamma_1\gamma_2(1-\exp(-(\gamma_1+\gamma_2)s))^2}{(\gamma_1+\gamma_2)^2(1-\exp(-2\gamma_1 s))(1-\exp(-2\gamma_2 s))}\right)^{\frac{1}{2}}
\end{aligned}
$$

2.3.1 SIMULATING PATHS OF THE TWO-FACTOR VASICEK MODEL: TRANSITION DENSITY METHOD

When simulating paths we need to simulate each factor separately and add them together. We can use the transition density method for each factor. It works as follows:

- Choose initial short term rate r_0 and initial long term rate $r_0(\tau)$ for a suitable value of τ.
- Solve the following two equations for $\phi_{1,0}$ and $\phi_{2,0}$:

$$
\begin{aligned}
\phi_{1,0} + \phi_{2,0} &= r_0 \\
A(\tau) - B_1(\tau)\phi_{1,0} - B_2(\tau)\phi_{2,0} &= -r_0(\tau)\tau
\end{aligned}
$$

  The above two equations can be solved to obtain the following:

$$
\begin{aligned}
\phi_{1,0} &= \frac{B_2(\tau)r0 - \tau r_0(\tau) - A(\tau)}{B_2(\tau) - B_1(\tau)} \\
\phi_{2,0} &= \frac{\tau r_0(\tau) + A(\tau) - B_1(\tau)r_0}{B_2(\tau) - B_1(\tau)}
\end{aligned}
$$

- Simulate standard bivariate normal random number (Z_1, Z_2) with correlation coefficient $\rho(s)$. Note that this can be achieved by simulating two independent random numbers, x_1 and x_2, from a standard normal distribution and then setting:

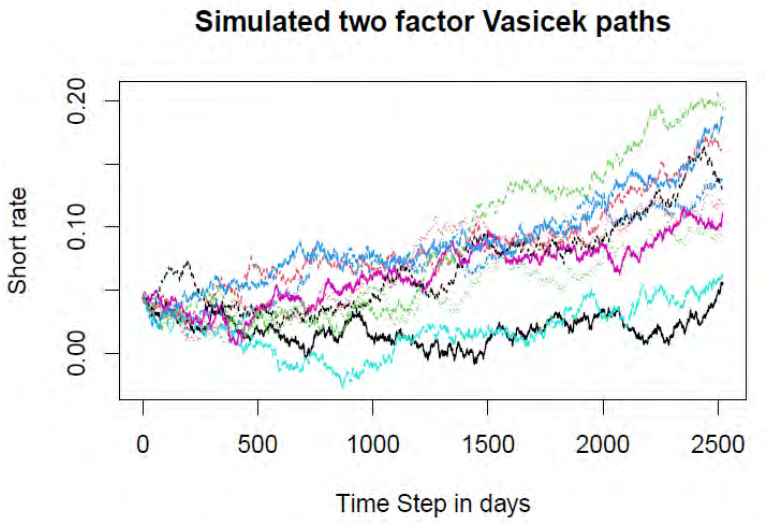
$$
\begin{aligned}
Z_1 &= x_1 \\
Z_2 &= \rho(s)x_1 + \sqrt{1-\rho(s)^2}x_2
\end{aligned}
$$

- Set:

$$
\begin{aligned}
\phi_{i,t+s} &= \phi_{i,t}\exp(-\gamma_i s) + \vec{\phi}_i(1-\exp(-\gamma_i s)) + \sqrt{\frac{\sigma_i^2}{2\gamma_i}(1-\exp(-2\gamma_i s))}Z_i, i = 1, 2, \\
r_{t+s} &= \phi_{1,t+s} + \phi_{2,t+s}
\end{aligned}
$$

The R function "Two.factor.Vasicek.Trans.sim" in the "InterestCalibration" library implements this method. The following code snippet demonstrates use of "Two.factor.Vasicek.Trans.sim".

```
#example6
rm(list=ls()) # clear the workspace and functions
set.seed(123)
X =Two.factor.Vasicek.Trans.sim(Delta=1/252,M=10)
matplot(X,type="l",main="Simulated two factor Vasicek paths",
        ylab="Short rate",xlab="Time Step in days")
```



3 Calibration Techniques

In this section we provide calibrations of the Vasicek models and the CIR model. We look at real-world calibration for the Vasicek model and CIR model followed by risk-neutral calibration for the Vasicek model, the CIR model and for the two-factor Vasicek model. For real-world calibration the maximum likelihood estimation (MLE) technique is frequently used. MLE solves for parameters that maximize a likelihood function. This function is the probability distribution function of the parameters. MLE finds the parameters with the highest likelihood of generating the observed data from a statistical distribution assumed to fit the underlying data. However, as we shall see the parameter estimates have some bias.

Since the MLE parameters are estimates of the underlying data distribution's parameters, the parameters found by MLE are random variables with distributions and moments such as a mean. This means the accuracy of the estimated parameters may be estimated and evaluated for goodness of fit.

The MLE method has several drawbacks that limit its usefulness for interest rate models. First, it fits parameters to a data sample from a statistical distribution chosen by the modeler but provides no information on whether that model is a reasonable fit to the "true" underlying distribution. This means the assumed statistical distribution may be statistically biased, with a mean significantly different than the underlying "true" distribution of the data. MLE relies on asymptotic properties of large datasets which may not hold for interest rate models. A small sample of short rates may not produce convergent parameters, while extending the data farther back in time brings in interest rates from different economic and policy environments (say, pre-global financial crisis) that may not be applicable for models projecting into the future. MLE assumes the data is stationary. Short rates may appear stationary for a period of time due to government monetary policy, but may exhibit a drift or jump when such policies change. This drift could be due to a set of complex macroeconomic interactions of economic cycles, fiscal and monetary policy, or shocks due to extreme weather, climate, pandemic, or demographic changes.

3.1 THE VASICEK MODEL: REAL-WORLD CALIBRATION

In this section we look at calibrating a Vasicek model with real-world (equilibrium model) data. We assume that a random sample of short rate data $r_0, r_\Delta, r_{2\Delta}, \ldots, r_{n\Delta}$, perhaps overnight rates observed over five years, is available to us. For simplicity we have assumed rates are observed at consecutive times with

equidistant periods. In this section we discuss two calibration techniques for the Vasicek model: maximum likelihood estimates and long-term quantile method.

3.1.1 MAXIMUM LIKELIHOOD ESTIMATOR METHOD

We have shown that the transition density function $f(r_{t+s}|r_t)$ is normal with the following means and variances:

$$\begin{aligned} E[r_{t+s}|r_t] &= \bar{r} + (r_t - \bar{r})e^{-\gamma s} \\ Var[r_{t+s}|r_t] &= \frac{\sigma^2}{2\gamma}(1 - e^{-2\gamma s}) \end{aligned}$$

For notational simplicity let us define the following variables as defined in the introductory section:

$$\begin{aligned} \alpha^* &= (1 - \exp(-\gamma\Delta))\bar{r} \\ \beta^* &= \exp(-\gamma\Delta) \\ \sigma^* &= \sqrt{\frac{\sigma^2}{2\gamma}(1 - e^{-2\gamma\Delta})} \end{aligned}$$

Then we can write:

$$\begin{aligned} E[r_{i\Delta}|r_{(i-1)\Delta}] &= \alpha^* + \beta^* r_{(i-1)\Delta}, \\ Var[r_{i\Delta}|r_{(i-1)\Delta}] &= \sigma^{*2}, \quad i = 1, 2, \ldots, n \end{aligned}$$

The density function, $f\left(r_{i\Delta}|r_{(i-1)\Delta}\right)$, of the conditional random variable $r_{i\Delta}|r_{(i-1)\Delta}$ for $i = 1, 2, \ldots, n$ can be written as:

$$f(r_{i\Delta}|r_{(i-1)\Delta}) = \frac{1}{\sqrt{2\pi}\sigma^*} \exp\left(-\frac{(r_{i\Delta} - \alpha^* - \beta^* r_{(i-1)\Delta})^2}{2\sigma^{*2}}\right)$$

Now it remains to specify the density function of the random variable r_0. The literature is not quite clear about how to specify it, therefore we denote it as $f_0(r_0|\alpha^*, \beta^*, \sigma^*)$ without specifying its functional form. With this notation we can write the joint likelihood function of the random sample $r_0, r_\Delta, r_{2\Delta}, \ldots, r_{n\Delta}$ as:

$$\begin{aligned} \mathscr{L} &= f_0(r_0|\alpha^*, \beta^*, \sigma^*) \prod_{i=1}^{n} f(r_{i\Delta}|r_{(i-1)\Delta}) \\ \mathscr{L} &= f_0(r_0|\alpha^*, \beta^*, \sigma^*) \prod_{i=1}^{n} \frac{1}{\sqrt{2\pi}\sigma^*} \exp\left(-\frac{(r_{i\Delta} - \alpha^* - \beta^* r_{(i-1)\Delta})^2}{2\sigma^{*2}}\right) \\ &= f_0(r_0|\alpha^*, \beta^*, \sigma^*) \frac{1}{\sqrt{(2\pi)^n}\sigma^{*n}} \exp\left(-\sum_{i=1}^{n} \frac{(r_{i\Delta} - \alpha^* - \beta^* r_{(i-1)\Delta})^2}{2\sigma^{*2}}\right) \end{aligned}$$

Then the log-likelihood function becomes:

$$\begin{aligned} \ln(\mathscr{L}) &= \ln(f_0(r_0|\alpha^*, \beta^*, \sigma^*)) - \\ &\quad \ln(\sqrt{(2\pi)^n}) - n\ln(\sigma^*) - \sum_{i=1}^{n} \frac{(r_{i\Delta} - \alpha^* - \beta^* r_{(i-1)\Delta})^2}{2\sigma^{*2}} \end{aligned} \tag{7}$$

Before we maximize $ln(\mathcal{L})$, we need to specify the function $f_0(r_0|\alpha^*, \beta^*, \sigma^*)$. If n is sufficiently large we could ignore the contribution of $f_0(r_0|\alpha^*, \beta^*, \sigma^*)$ into the $ln(\mathcal{L})$ and proceed. This method is called the quasi-maximum likelihood method. In this situation maximization is straightforward and we can obtain the following explicit expressions for estimates:

$$\begin{aligned}
\hat{\beta}^* &= \frac{\sum_{i=1}^{n} r_{i\Delta} r_{(i-1)\Delta} - \frac{1}{n}\sum_{i=0}^{n-1} r_{i\Delta} \sum_{i=1}^{n} r_{i\Delta}}{\sum_{i=0}^{n-1} r_{i\Delta}^2 - \frac{1}{n}\left(\sum_{i=0}^{n-1} r_{i\Delta}\right)^2} \\
\hat{\alpha}^* &= \frac{1}{n}\left(\sum_{i=1}^{n} r_{i\Delta} - \hat{\beta}\sum_{i=0}^{n-1} r_{i\Delta}\right) \\
\hat{\sigma^{*2}} &= \frac{1}{n-2}\sum_{i=1}^{n}\left(r_{i\Delta} - \hat{\alpha} - \hat{\beta} r_{(i-1)\Delta}\right)^2
\end{aligned}$$

Note that $\hat{\sigma}^{*2}$ is the unbiased estimator for σ^{*2}, not the MLE. Once we have these estimates we solve for $\gamma, \bar{r}$ and σ using the following formulas:

$$\begin{aligned}
\gamma &= -\frac{\ln(\hat{\beta}^*)}{\Delta} \\
\bar{r} &= \frac{\hat{\alpha}^*}{1-\hat{\beta}^*} \\
\sigma &= \sqrt{\frac{2\hat{\gamma}\hat{\sigma}^{*2}}{1-\hat{\beta}^{*2}}}
\end{aligned}$$

The implementation of this in R is straightforward as we can use R function "lm" for estimation. We illustrate the MLE calculation using the data in chapter 14, exercise Q5 of [24], as discussed in Section 5.2 and 5.3 below.

```
#example7
# Exercises Q5 Chapter 14 of Veronesi
#clear the workspace and functions
rm(list=ls())
rt =VeronesiTable14p7q5$rt
Delta = 1/252
N = length(rt)
y = rt[2:N]
x = rt[1:(N-1)]
model = lm(y~x)
mle.gamma= -log(as.numeric(model$coefficients)[2])/Delta
mle.rbar = as.numeric(model$coefficients)[1]/
        (1-as.numeric(model$coefficients)[2])
mle.sigma = sigma(model)*(2*mle.gamma/
        (1-as.numeric(model$coefficients)[2]^2))^.5
tb = anova(model)
```

The MLE's of $\gamma, \bar{r}$ and σ are 3.64532, 0.30769 and 1.81376, respectively. The ANOVA table related to this calibration is:

	Df	Sum Sq	Mean Sq	F value	p-value
x	1	376.303	376.303	29245	0
Residuals	541	6.961	0.013	NA	NA

To examine the performance of the MLE we carry out a simulation study using the R code snippet given below.

```
#example8
# clear the workspace and functions
rm(list=ls())
t0=0
T=10
r0=0.03
gamma=0.3
rbar=0.05
sigma=0.0221
paras = c(gamma,rbar,sigma)
# Initialize # of paths number of points in each path
Delta = 1/252
N = ceiling((T-t0)/Delta)
set.seed(123)
X=Vasicek.Trans.sim(t0,T,Delta,r0,gamma,rbar,sigma,M=1000)
M = ncol(X)
mle= matrix(0,3,M)
for (i in 1:M){
 y = X[2:(N+1),i]
 x = X[1:N,i]
 model=lm(y~x)
 mle.gamma= -log(as.numeric(model$coefficients)[2])/Delta
 mle.rbar = as.numeric(model$coefficients)[1]/
          (1-as.numeric(model$coefficients)[2])
 mle.sigma = sigma(model)*(2*mle.gamma/
          (1-as.numeric(model$coefficients)[2]^2))^.5
 mle[,i]=c(mle.gamma,mle.rbar,mle.sigma)
}
# Simulation performance criteria
bias =rowMeans(mle-paras)
Sd = rowSds(mle)
rmse = rowMeans((mle-paras)^2)^0.5
```

Simulation performance based on 1000 sample paths

	Parameter		
	γ	$\bar{r}$	σ
Bias	0.50238	-0.00267	1.58447×10^{-5}
Standard deviation	0.50194	0.05453	3.03235×10^{-4}
Root mean square	0.70999	0.05457	3.03497×10^{-4}

From the table we observe that MLE of σ performs very well, $\bar{r}$ performs somewhat better, and γ performs poorly. This is consistent with the results of [21] and readers are encouraged to explore this by changing the simulation specifications.

Instead of dropping the term $f_0(r_0|\alpha^*, \beta^*, \sigma^*)$ from the likelihood function, we could assume that r_0 is normally distributed with:

$$\begin{aligned} E[r_0] &= \frac{\alpha^*}{1-\beta^*} \\ Var[r_0] &= \frac{\sigma^{*2}}{1-\beta^{*2}} \end{aligned}$$

This choice seems reasonable and since $\gamma > 0$, $\beta^* < 1$ the pdf in this case becomes:

$$f_0(r_0|\alpha^*, \beta^*, \sigma^*) = \frac{1}{\sqrt{2\pi\sigma^{*2}/(1-\beta^{*2})}} \exp\left[-\frac{(r_0 - [\alpha^*/(1-\beta^*)])^2}{2\sigma^{*2}/(1-\beta^{*2})}\right]$$

Substituting this in (7) we obtain the following as the log-likelihood function:

$$\begin{aligned} \ln(\mathscr{L}) = &-\frac{1}{2}\ln(2\pi) - \frac{1}{2}\ln(2\sigma^{*2}/(1-\beta^{*2})) - \frac{(r_0 - [\alpha^*/(1-\beta^*)])^2}{2\sigma^{*2}/(1-\beta^{*2})} \\ &- \ln(\sqrt{(2\pi)^n}) - n\ln(\sigma^*) - \sum_{i=1}^{n} \frac{(r_{i\Delta} - \alpha^* - \beta^* r_{(i-1)\Delta})^2}{2\sigma^{*2}} \end{aligned} \tag{8}$$

When we compare this log-likelihood function with the log-likelihood function given in equation (5.2.9) on page 119 in [15], we see that (8) is in fact the log-likelihood function of an AR(1) model. Now it should become clear to the reader why this pdf for r_0 was chosen. For parameter estimation we could use R function ARIMA as illustrated in the code below.

```
#example9
# clear the workspace and functions
rm(list=ls())
t0=0
T=10
r0=0.03
gamma=0.3
rbar=0.05
sigma=0.0221
```

```
paras = c(gamma,rbar,sigma)
# Initialize # of paths number of points in each path
Delta = 1/252
N = ceiling((T-t0)/Delta)

set.seed(123)
X=Vasicek.Trans.sim(t0,T,Delta,r0,gamma,rbar,sigma,M=1000)

M = ncol(X)
mle= matrix(0,3,M)
for (i in 1:M){
  y = X[2:(N+1),i]
  x = X[1:N,i]

  model2 = arima(X[1:(N+1),i],order=c(1,0,0),include.mean = TRUE,
                 method="ML")
  mle.gamma1= -log(as.numeric(model2$coef)[1])/Delta
  mle.rbar1 = as.numeric(model2$coef)[2]
  mle.sigma1= (model2$sigma2*2*mle.gamma1/
  (1-as.numeric(model2$coef)[1]^2))^.5
  mle[,i]=c(mle.gamma1,mle.rbar1,mle.sigma1)
}
# Simulation performance criteria
bias =rowMeans(mle-paras)
sd = rowSds(mle)
rmse = rowMeans((mle-paras)^2)^0.5
```

Simuation performance based on 1000 sample paths

	Parameter		
	γ	$\bar{r}$	σ
Bias	0.49128	-0.00672	8.54594×10^{-6}
Standard deviation	0.47409	0.01674	3.03163×10^{-4}
Root mean square	0.68256	0.01803	3.03132×10^{-4}

We observe that the performance of MLEs are quite similar to that of MLEs based on a quasi-likelihood function.

3.1.2 LONG TERM QUANTILE METHOD

This method is proposed in [5]. The major underlying assumption in this method is that sample quantiles are representative of the future observed quantiles. The second mathematically justifiable assumption is that theoretical quantiles are calculated based on the process behaviour when $t \to \infty$. The calculation of the long-term mean and variance of r_t is as follows:

$$\begin{aligned} E[r_t|r_0] &= \bar{r} + (r_0 - \bar{r})e^{-\gamma t} \\ \lim_{t\to\infty} E[r_t|r_0] &= \bar{r} \\ Var[r_t|r_0] &= \frac{\sigma^2}{2\gamma}(1 - e^{-2\gamma t}) \\ \lim_{t\to\infty} Var[r_t|r_0] &= \frac{\sigma^2}{2\gamma} \end{aligned}$$

As r_t's are normally distributed random variables the 95% confidence interval for $lim_{t\to\infty} r_t$ is given by:

$$P\left[\bar{r} - \frac{1.96\sigma}{\sqrt{2\gamma}} \leq \lim_{t\to\infty} r_t \leq \bar{r} + \frac{1.96\sigma}{\sqrt{2\gamma}}\right] = 0.95$$

Let us denote $\hat{q}_{0.025}$ and $\hat{q}_{0.975}$ as the 2.5th and the 97.5th percentiles respectively, based on a sample of r_t where preferably $t > 10$. Then we have the following two equations:

$$\hat{q}_{0.025} = \bar{r} - \frac{1.96\sigma}{\sqrt{2\gamma}}$$

$$\hat{q}_{0.975} = \bar{r} + \frac{1.96\sigma}{\sqrt{2\gamma}}$$

We can solve these two equations to obtain the following expressions for γ and $\bar{r}$:

$$\gamma = 2\left(\frac{1.96\sigma}{\hat{q}_{0.975} - \hat{q}_{0.025}}\right)^2$$

$$\bar{r} = \frac{\hat{q}_{0.025} + \hat{q}_{0.975}}{2}$$

This procedure is easy to implement as given in the R code snippet below, but the drawback is obtaining a sample of r_t for a large value of t.

```
#example10
# clear the workspace and functions
rm(list=ls())
t0=0
T=20
r0=0.03
gamma=0.3
rbar=0.05
sigma=0.02
# Initialize # of paths number of points in each path
Delta = 1/252
N = ceiling((T-t0)/Delta)
set.seed(123)
# simulate M sample paths and retain the last row only.
X=Vasicek.Trans.sim(t0,T,Delta,r0,gamma,rbar,sigma,M=100)[N+1,]
qs = quantile(X,c(0.025,0.975))
est.gamma = 2*((1.96*sigma)/(as.numeric(qs[2])-as.numeric(qs[1])))^2
est.rbar = (as.numeric(qs[2])+as.numeric(qs[1]))/2
```

The resulting estimates of γ and $\bar{r}$ are 0.28199 and 0.05267 respectively.

3.2 THE VASICEK MODEL: RISK-NEUTRAL CALIBRATION

In this section we present calibration of a Vasicek model when we have a set of zero-coupon bond prices or a set of discount factors. The underlying assumption is that the real-world interest rate follows a Vasicek model and hence we can use a drift-adjusted Vasicek model that leads to the following closed-form formula for zero coupon bond prices.

$$[24](15.28) \qquad Z(r,t;T) = e^{A(t;T)-B(t;T)r}$$

$$[24](15.29) \qquad B(t;T) = \frac{1}{\gamma^*}\left(1 - e^{-\gamma^*(T-t)}\right)$$

$$[24](15.30) \qquad A(t;T) = (B(t;T) - (T-t))\left(\bar{r}^* - \frac{\sigma^2}{2(\gamma^*)^2}\right) - \frac{\sigma^2 B(t;T)^2}{4\gamma^*}$$

Note that we are using parameters γ^* and $\bar{r}^*$ to indicate the drift of the SDE for r_t, $\gamma^*(\bar{r}^* - r_t)$ in the risk-neutral world, as opposed to parameters γ and $\bar{r}$ with a drift of $\gamma(\bar{r} - r_t)$ in the real-world.

The calibration process involves minimizing the following quantity:

$$J(\gamma^*, \bar{r}^*) \quad = \quad \sum_{i=1}^{n} (Z^{Vasicek}(r_0, 0; T_i) - Z^{Data}(0, T_i))^2$$

We can use either the "nlm" function or the "optim" function for this purpose. The following example first calculates a bond price vector for given maturities with known parameter values and then calculates parameters using the minimization. It is used as a verification of our code for the minimization.

```
#example11
# clear the workspace and functions
rm(list=ls())
bond.maturities = seq(0.5,10,0.5) #Time_to_maturity
bond.prices = Vasicek.zcbp(r0=0.02,t=0,T=bond.maturities,
                           gamma=0.5,rbar=0.07,sigma=0.02)
model = nlm(f=Vasicek.J,p=c(0.4,0.01),r0=0.02,sigma=0.02,
            bond.prices=bond.prices,bond.maturities=bond.maturities)
model1 = optim(c(0.4,0.01),Vasicek.J,method=("BFGS")
               ,r0=0.02,sigma=0.02,bond.prices=bond.prices,
               bond.maturities=bond.maturities)
```

Risk-neutral Parameter Estimates

Method	Parameter	
	γ^*	$\bar{r}^*$
nlm	0.49999	0.07
optim	0.49951	0.07002

Readers are encouraged to try various values for parameters γ^* and $\bar{r}^*$ and various reasonable guesses for initial values of those parameters to test the accuracy of the "nlm" and the "optim" routines.

Another option we may consider is if we do not have a reliable estimate for σ we could also estimate it from observed bond price data as described in the following code snippets.

```
#example12
# clear the workspace and functions
rm(list=ls())
bond.maturities = seq(0.5,10,0.5) #Time_to_maturity
bond.prices = Vasicek.zcbp(r0=0.02,t=0,T=bond.maturities,
                           gamma=0.5,rbar=0.07,sigma=0.02)
model = nlm(f=Vasicek.J,p=c(0.4,0.02,0.04),r0=0.02,iterlim=1000,
            bond.prices=bond.prices,bond.maturities=bond.maturities)
model1 = optim(c(0.4,0.02,0.04),lower=c(0,0,0),Vasicek.J,
               method=("L-BFGS-B"),r0=0.02,
               bond.prices=bond.prices,
               bond.maturities=bond.maturities)
```

Risk-neutral Parameter Estimates

Method	Parameter		
	γ^*	$\bar{r}^*$	σ
nlm	0.43033	0.0802	0.06206
optim	0.41491	0.08264	0.06617

After trying out a few different initial values and increasing the maximum number of iterations in "nlm" we see that "nlm" produces values close to the actual values, but values produced by "optim" are not very close. This led to examination of other optimization methods available in R. Instead of writing our own function to minimize we can use the "nls" package for non-linear least squares minimization. Readers who may want to know more about non-linear regression with R may consult [22].

We realized using "nls" for yield rates instead of bond prices yields better results. The following code snippet illustrates usage of "nls" with bond yields.

```
#example13
rm(list=ls())
bond.maturities = seq(0.5,10,0.5) #Time_to_maturity
bond.yields = Vasicek.yield(r0=0.02,t=0,T=bond.maturities,
                            gamma=0.5,rbar=0.07,sigma=0.02)
bonddata = data.frame(cbind(bond.maturities,bond.yields))

Vasicek.fit =
   nls(bond.yields~Vasicek.yield(r0=0.02,t=0,T=bond.maturities,
                                 gamma,rbar,sigma),
       start=list(gamma=0.2,rbar=0.02,sigma=0.01),
       data=bonddata,algorithm = "port",
       lower=list(gamma=0.1,rbar=0.01,sigma=0.005),
       upper =list(gamma=3,rbar=1,sigma=1),
       nls.control(maxiter = 100, tol = 1e-3, minFactor = 1/1024,
                   printEval = FALSE, warnOnly = FALSE,
                   scaleOffset = 0, nDcentral = FALSE))
sum_mod= summary(Vasicek.fit)
```

The “nls” algorithm converged in 9 iterations and summary statistics are:

The Vasicek fit (simulated data):summary statistics

Parameter	Estimate	Std. Error	t-value	$Pr[>\|t\|]$
γ^*	0.5	3.53356×10^{-10}	1.41501×10^{9}	7.11346×10^{-155}
$\bar{r}^*$	0.07	1.4747×10^{-11}	4.74673×10^{9}	2.45829×10^{-164}

We can also try to estimate σ using bond yield data with “nls” as given below:

```
#example14
rm(list=ls())
bond.maturities = seq(0.5,10,0.5) #Time_to_maturity
bond.yields = Vasicek.yield(r0=0.02,t=0,T=bond.maturities,
                            gamma=0.5,rbar=0.07,sigma=0.02)
bonddata = data.frame(cbind(bond.maturities,bond.yields))
Vasicek.fit =
   nls(bond.yields~Vasicek.yield(r0=0.02,t=0,T=bond.maturities,
                                 gamma,rbar,sigma),
       start=list(gamma=0.2,rbar=0.02,sigma=0.01),
       data=bonddata,algorithm = "port",
       lower=list(gamma=0.1,rbar=0.01,sigma=0.005),
       upper =list(gamma=3,rbar=1,sigma=1),
       nls.control(maxiter = 100, tol = 1e-3, minFactor = 1/1024,
                   printEval = FALSE, warnOnly = FALSE,
                   scaleOffset = 0, nDcentral = FALSE))
sum_mod= summary(Vasicek.fit)
```

The “nls” algorithm converged in 35 iterations and summary statistics are:

The Vasicek fit (simulated data):summary statistics

Parameter	Estimate	Std. Error	t-value	$Pr[>\|t\|]$
γ^*	0.5	3.74357×10^{-9}	1.33562×10^{8}	4.00432×10^{-129}
$\bar{r}^*$	0.07	4.70173×10^{-10}	1.48881×10^{8}	6.32221×10^{-130}
σ	0.02	5.20598×10^{-9}	3.84174×10^{6}	6.34129×10^{-103}

We can also add a random error to simulated bond yields to evaluate the performance.

```
#example15
rm(list=ls())
bond.maturities = seq(0.5,10,0.5) #Time_to_maturity
bond.yields = Vasicek.yield(r0=0.02,t=0,T=bond.maturities,
                            gamma=0.5,rbar=0.07,sigma=0.02)+
                            rnorm(length(bond.maturities),0.00,0.001)
bonddata = data.frame(cbind(bond.maturities,bond.yields))
Vasicek.fit =
  nls(bond.yields~Vasicek.yield(r0=0.02,t=0,T=bond.maturities,
                                gamma,rbar,sigma=0.02),
      start=list(gamma=0.2,rbar=0.02),
      data=bonddata,algorithm = "port",
```

```
      lower=list(gamma=0.1,rbar=0.01),
      upper =list(gamma=3,rbar=1),
      nls.control(maxiter = 100, tol = 1e-3, minFactor = 1/1024,
                  printEval = FALSE, warnOnly = FALSE,
                  scaleOffset = 0, nDcentral = FALSE))
sum_mod= summary(Vasicek.fit)
```

The "nls" algorithm converged in 10 iterations and summary statistics are:

The Vasicek fit (simulated data):summary statistics

Parameter	Estimate	Std. Error	t-value	$Pr(>\|t\|)$
γ^*	0.47522	0.02492	19.06995	2.18791×10^{-13}
$\bar{r}^*$	0.07148	0.00116	61.55485	2.19487×10^{-22}

We can try estimating σ from the same data set with a random error added to the yields:

```
#example16
rm(list=ls())
bond.maturities = seq(0.5,10,0.5) #Time_to_maturity
bond.yields = Vasicek.yield(r0=0.02,t=0,T=bond.maturities,
                            gamma=0.5,rbar=0.07,sigma=0.02)+
              rnorm(length(bond.maturities),0.00,0.001)
bonddata = data.frame(cbind(bond.maturities,bond.yields))
Vasicek.fit =
  nls(bond.yields~Vasicek.yield(r0=0.02,t=0,T=bond.maturities,
                                gamma,rbar,sigma),
      start=list(gamma=0.2,rbar=0.02,sigma=0.01),
      data=bonddata,algorithm = "port",
      lower=list(gamma=0.1,rbar=0.01,sigma=0.005),
      upper =list(gamma=3,rbar=1,sigma=1),
      nls.control(maxiter = 100, tol = 1e-3, minFactor = 1/1024,
                  printEval = FALSE, warnOnly = FALSE,
                  scaleOffset = 0, nDcentral = FALSE))
sum_mod= summary(Vasicek.fit)
```

The "nls" algorithm converged in 44 iterations and summary statistics are:

The Vasicek fit (simulated data):summary statistics

Parameter	Estimate	Std. Error	t-value	$Pr(>\|t\|)$
γ^*	0.37994	2.10799	0.18024	0.8591
$\bar{r}^*$	0.08849	0.37455	0.23625	0.81606
σ	0.07293	0.31051	0.23488	0.81711

We see that when there is a random error, the estimate of σ is not close to the actual value. Irrespective of what the results are, it is not a good practice to estimate σ from the same data set; it has to be estimated from short rate volatility under a real life scenario.

We encourage readers to try different parameters and different initial values for all three methods using "optim", "nlm" and "nls". Based on our limited testing presented here for real applications, we recommend

using “nls” and trying out many different initial values and increasing number of iterations, before settling into one solution.

In the following code chunk we use data in Table 15.1 of [24].

```
#example17
# clear the workspace and functions
rm(list=ls())

bond.prices = VeronesiTable15p1$Strips/100
bond.maturities = VeronesiTable15p1$'Time to Maturity'
bond.yields = -log(bond.prices)/bond.maturities
bonddata = data.frame(cbind(bond.maturities,bond.yields))
model = nlm(f=Vasicek.J,p=c(0.4,0.06),r0=0.0168,sigma=0.0221,
            bond.prices=bond.prices,bond.maturities=bond.maturities)
model1 = optim(c(0.4,0.06),Vasicek.J,method=("CG"),sigma=0.0221
               ,bond.prices=bond.prices,bond.maturities=bond.maturities)
Vasicek.fit =
  nls(bond.yields~Vasicek.yield(r0=0.0168,t=0,T=bond.maturities,
                                gamma,rbar,sigma=0.0221),
      start=list(gamma=0.4,rbar=0.06),
      data=bonddata,algorithm = "port",
      lower=list(gamma=0.1,rbar=0.01),
      upper =list(gamma=10,rbar=10),
      nls.control(maxiter = 1000, tol = 1e-3, minFactor = 1/1024,
                  printEval = FALSE, warnOnly = FALSE,
                  scaleOffset = 0, nDcentral = FALSE))
sum_mod= summary(Vasicek.fit)
```

Risk-neutral Parameter Estimates

Method	Parameter	
	γ^*	$\bar{r}^*$
Text	0.46530	0.06340
nlm	0.46509	0.06344
optim	0.40785	0.06615
nls	0.36455	0.06899

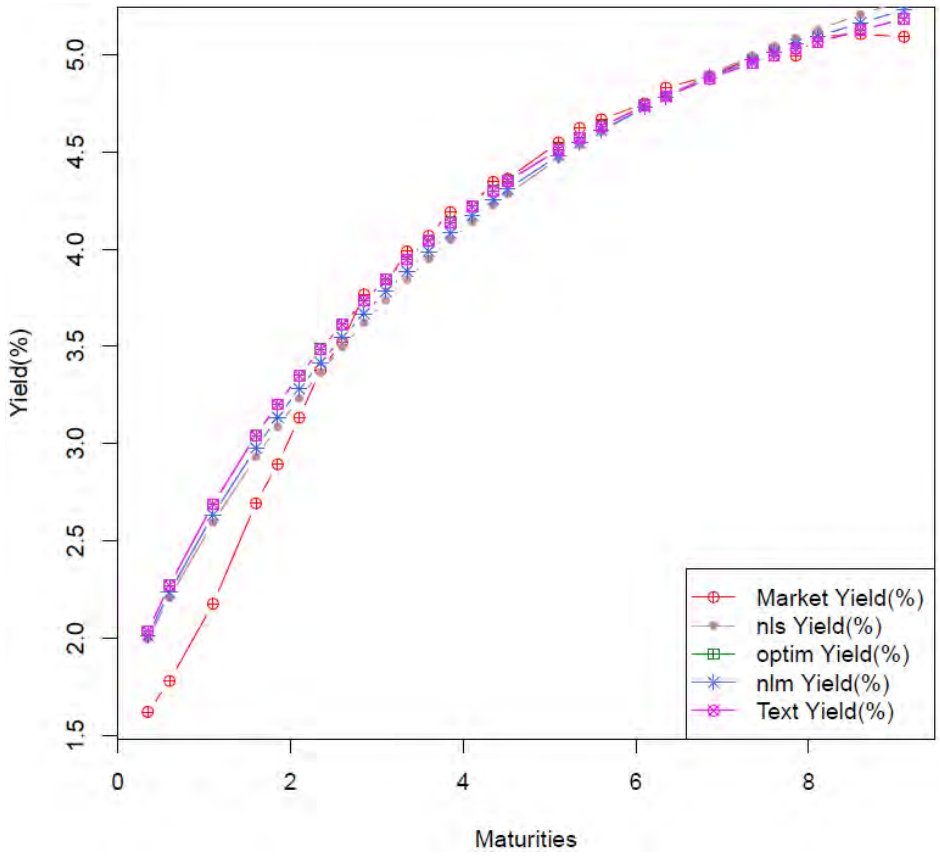
We see that values produced by “nlm” are close to the values given in Example 15.1 of [24]. The following is a summary statistics of the “nls” method, which converged after 7 iterations.

The Vasicek fit (simulated data):summary statistics

Parameter	Estimate	Std. Error	t-value	$Pr[>\|t\|]$
γ^*	0.36455	0.0406	8.97929	1.89865×10^{-9}
$\bar{r}^*$	0.06899	0.00318	21.67434	3.60621×10^{-18}

To understand which optimization routine performs better we plot the fitted data along with market data and textbook estimates:

```
#example18
optim.yield =Vasicek.yield(r0=0.0168,t=0,T=bond.maturities,
                           gamma=model$estimate[1],rbar=model$estimate[2],
                           sigma=0.0222)
nlm.yield =Vasicek.yield(r0=0.0168,t=0,T=bond.maturities,
                         gamma=model1$par[1],rbar=model1$par[2],
                         sigma=0.0222)
text.yield = Vasicek.yield(r0=0.0168,t=0,T=bond.maturities,
                           gamma=0.4653,rbar=0.0634,sigma=0.0222)
plot(bond.maturities,bond.yields*100,xlab="Maturities",ylab="Yield(%)",
     type="b",pch=10,lty="solid",col="red1")
lines(bond.maturities,fitted(Vasicek.fit)*100,type="b",pch=20,lty="solid",
      col="rosybrown")
lines(bond.maturities,optim.yield*100,type="b",pch=12,col="green4")
lines(bond.maturities,nlm.yield*100,type="b",pch=8,col="royalblue")
lines(bond.maturities,text.yield*100,type="b",pch=13,col="magenta")
legend("bottomright",legend=c("Market Yield(%)","nls Yield(%)",
                              "optim Yield(%)","nlm Yield(%)","Text Yield(%)"),
       col=c("red1","rosybrown","green4","royalblue","magenta"),
       lty=c(1,1,1),pch=c(10,20,12,8,13))
```



3.3 THE CIR MODEL: REAL-WORLD CALIBRATION

In this section we consider three calibration methods: the Euler method, exact likelihood estimates and the generalized method of moments (GMM). The Euler discretization method leads to explicit expressions for estimates, while the exact maximum likelihood estimate method and GMM require using numerical routines to obtain solutions.

3.3.1 EULER METHOD

As we saw earlier the Euler discretization of the CIR model leads to:

$$r(i) = \alpha_1 + \beta_1 r(i-1) + \sqrt{r(i-1)}\epsilon_i, \quad i = 1, 2, \ldots$$

with ϵ_i's that are i.i.d. normal with mean 0 standard deviation σ. We can rearrange this to get a linear regression as below:

$$\begin{aligned} r(i) &= \alpha_1 + \beta_1 r(i-1) + \sqrt{r(i-1)}\epsilon_i, \quad i = 1, 2, \ldots \\ \frac{r(i)}{\sqrt{r(i-1)}} &= \alpha_1 \left(\frac{1}{\sqrt{r(i-1)}} \right) + \beta_1 \sqrt{r(i-1)} + \epsilon_i \end{aligned}$$

Now write:

$$\begin{aligned} y_i &= \frac{r(i)}{\sqrt{r(i-1)}} \\ x_{1i} &= \frac{1}{\sqrt{r(i-1)}} \\ x_{2i} &= \sqrt{r(i-1)} \\ y_i &= \alpha_1 x_{1i} + \beta_1 x_{2i} + \epsilon_i, i = 1, 2, \ldots \end{aligned}$$

To obtain MLEs of α_1 and β_1 we can use the "lm" function in R. Once we obtain least square estimates (or MLE) of α_1 and β_1 we can obtain γ, $\bar{r}$ and α using:

$$\begin{aligned} \gamma &= \frac{1 - \beta_1}{\Delta} \\ \bar{r} &= \frac{\alpha_1}{1 - \beta_1} \\ \alpha &= \frac{\sigma^2}{\Delta} \end{aligned}$$

The following R code snippet illustrates this point along with a simple study of bias and mean square error of this estimate.

```
#example19
# The following code snippets calculate the bias of simulated parameters
#CIR Calibration
rm(list=ls()) # clear the workspace and functions
graphics.off() # clear the plots
# CIR Euler estimate
# Initialize parameters
library(matrixStats)
```

```
t0=0
T=10
r0=0.02
gamma=0.3807
rbar=0.072
alpha=0.0548
paras = c(gamma,rbar,alpha)
# Initialize # of paths number of points in each path
Delta = 1/252
N = ceiling((T-t0)/Delta)

set.seed(123)
X=CIR.Trans.sim(t0=0,T=10,Delta=1/252,r0=0.02,
                gamma=0.3807,rbar=0.072,alpha=0.0548,M=1000)

M = ncol(X)
euler.est= matrix(NA,3,M)
for (i in 1:M){
  x1 = X[1:N,i]^(-0.5)
  x2 = X[1:N,i]^(0.5)
  y = X[2:(N+1),i]*x1
  model=lm(y~0+x1+x2)
  euler.est[1,i]= (1-as.numeric(model$coefficients)[2])/Delta
  euler.est[2,i] = as.numeric(model$coefficients)[1]/
                   (1-as.numeric(model$coefficients)[2])
  euler.est[3,i] = sigma(model)^2/Delta
}

bias=rowMeans(euler.est-paras)
sd =rowSds(euler.est)
rmse =(rowMeans((euler.est-paras)^2))^0.5
```

Simulation performance based on 1000 sample paths

	Parameter		
	γ	$\bar{r}$	α
Bias	0.47566	0.02376	3.3307×10^{-4}
Standard deviation	0.5895	0.71757	0.0018
Root mean square	0.75724	0.7176	0.00183

3.3.2 MAXIMUM LIKELIHOOD ESTIMATE

As we saw earlier a short rate sample can be stated as $r(0), r(1), \ldots, r(n)$. Using (5) we can write the following expression for the full likelihood function:

$$\begin{aligned}\mathscr{L} &= f(r(0)|\gamma,\bar{r},\alpha)\prod_{i=0}^{n-1} f(r(i+1)|r(i),\gamma,\bar{r},\alpha)\\ &= f(r(0)|\gamma,\bar{r},\alpha)\prod_{i=1}^{n} c_\Delta \chi^2(c_\Delta r_i,\nu,\lambda_i)\end{aligned}$$

where $f(r(0)|\gamma,\bar{r},\alpha)$ is a pdf of $r(0)$, $f(r(0)|\gamma,\bar{r},\alpha)$ are the conditional pdfs of $r(i+1)|r(i)$ for $i = 0,1,2,\ldots,c_\Delta,\vartheta$ and λ_i are as given below:

$$\begin{aligned}c_\Delta &= \frac{4\gamma}{\alpha(1-\exp(-\gamma\Delta))}\\ \nu &= \frac{4\gamma}{\alpha}\bar{r}\\ \lambda_i &= c_\Delta r(i-1)\exp(-\gamma\Delta)\end{aligned}$$

Since $f(r(i+1)|r(i),\gamma,\bar{r},\alpha)$ is a function of a non-central χ^2 pdf, we will have the full likelihood function if we specify the pdf, $f(r(0)|\gamma,\bar{r},\alpha)$. However, in CIR calibration using MLE both [19] and [20] ignore the contribution of $f(r(0)|\gamma,\bar{r},\alpha)$ and call them as MLEs, even though it is actually based on quasi-likelihood functions.

The maximization of the quasi-likelihood function can be easily implemented in R as given below; however numerical solutions mostly depend on the initial guess. There are some concerns with this method, which uses R function "dchisq" as it is based on evaluation of a truncated form of the infinite sum in (6).

```
#example20
# The following code snippets calculate the bias of simulated parameters
# CIR Calibration
rm(list=ls()) # clear the workspace and functions
graphics.off() # clear the plots
# CIR Euler estimate
# Initialize parameters
library(matrixStats)
t0=0
T=10
r0=0.02
gamma=0.3807
rbar=0.072
alpha=0.0548
paras = c(gamma,rbar,alpha)
# Initialize # of paths number of points in each path
Delta = 1/252
N = ceiling((T-t0)/Delta)

set.seed(123)
X=CIR.Trans.sim(t0=0,T=10,Delta=1/252,r0=0.02,
                gamma=0.3807,rbar=0.072,alpha=0.0548,M=100)

M = ncol(X)
euler.est= matrix(NA,3,M)
mle.est = matrix(NA,3,M)
```

```
Wsample = NULL
for (i in 1:M){
  x1 = X[1:N,i]^(-0.5)
  x2 = X[1:N,i]^(0.5)
  y = X[2:(N+1),i]*x1
  model=lm(y~0+x1+x2)
  euler.est[1,i]= (1-as.numeric(model$coefficients)[2])/Delta
  euler.est[2,i] = as.numeric(model$coefficients)[1]/
                   (1-as.numeric(model$coefficients)[2])
  euler.est[3,i] = sigma(model)^2/Delta
  model1 = optim(euler.est[,i],CIR.log.lik,NULL,X[,i],method="BFGS")
  mle.est[,i]= model1$par
  if (!is.null(model1$message))
    Wsample = append(Wsample,i)
}
```

```
bias.euler= rowMeans(euler.est-paras)
sd.euler= rowSds(euler.est)
rmse.euler =(rowMeans((euler.est-paras)^2))^0.5
bias.mle =rowMeans(mle.est-paras)
sd.mle= rowSds(mle.est)
rmse.mle = (rowMeans((mle.est-paras)^2))^0.5
```

Simulation performance based on 100 sample paths

	Parameter		
	γ	$\bar{r}$	α
Bias (Euler method)	0.48285	−0.02065	9.20085×10^{-5}
Bias (MLE)	0.45736	−0.02114	-2.0002×10^{-4}
Standard deviation (Euler)	0.64624	0.21512	0.00156
Standard deviation (MLE)	0.56166	0.21401	0.00147
Root mean square (Euler)	0.80411	0.21504	0.00156
Root mean square (MLE)	0.72214	0.21398	0.00148

As you see in each of the sample calculations, the estimate of α has relatively smaller bias and the estimates of γ and $\bar{r}$ have a larger bias.

3.4 THE GENERALIZED METHOD OF MOMENTS

The idea of the generalized method of moments (GMM) first appeared in [16]. This method compares sample moments with their theoretical values. The parameters are estimated by minimizing the distance between sample moments and their theoretical values. Chan, Karolyi, Longstaff and Sanders (CKLS) [7] illustrate how to use GMM for the CKLS short rate model given by the following SDE:

$$dr_t = \gamma(\bar{r} - r_t)dt + (\alpha)^{1/2} r_t^{\tau} dX_t$$

Note that the above is a generalization of both the Vasicek and the CIR models as $\alpha = \sigma^2, \tau = 0$ and $\tau = 1/2$ yields the Vasicek and the CIR models respectively. As CIR contains only three parameters we can adopt GMM as follows:

First define:

$$
\begin{aligned}
f_1 &= \frac{1}{N}\sum_{i=1}^{N}(r(i) - \alpha_1 - \beta_1 r(i-1)) \\
f_2 &= \frac{1}{N}\sum_{i=1}^{N}[(r(i) - \alpha_1 - \beta_1 r(i-1))^2 - r(i-1)\sigma^2] \\
f_3 &= \frac{1}{N}\sum_{i=1}^{N}[r(i) - \alpha_1 - \beta_1 r(i-1)]\, r(i-1)
\end{aligned}
$$

We choose α_1, β_1 and σ to set f_1, f_2 and f_3 to zero, for instance by minimizing:

$$J(\alpha_1, \beta_1, \sigma) = f_1^2 + f_2^2 + f_3^2$$

The above equations are implemented in the code chunk:

```
CIR.gmm = function(param,X){
 alpha1 = param[1]
 beta1= param[2]
 sigma = param[3]
 N = length(X)
 f1=0
 f2=0
 f3=0
 for (i in (2:N))
 {
  f1 = f1 +  X[i]-alpha1-beta1*X[i-1]
  f2 = f2 + (X[i]-alpha1-beta1*X[i-1])^2 - X[i-1]*sigma^2
  f3 = f3 + (X[i]-alpha1-beta1*X[i-1]) * X[i-1]
 }
 return(1/(N-1)*(f1^2+f2^2+f3^2))
}
```

Once we obtain GMM estimates for α_1, β_1 and σ we obtain estimates for γ, $\bar{r}$ and α using the following as in the Euler method:

$$
\begin{aligned}
\gamma &= \frac{1-\beta_1}{\Delta} \\
\bar{r} &= \frac{\alpha_1}{1-\beta_1} \\
\alpha &= \frac{\sigma^2}{\Delta}
\end{aligned}
$$

The following R-code snippets implement this method:

```
#example21
rm(list=ls()) # clear the workspace and functions
graphics.off() # clear the plots
# CIR Euler estimate
# Initialize parameters
library(matrixStats)

t0=0
T=10
r0=0.02
gamma=0.3807
rbar=0.072
alpha=0.0548
paras = c(gamma,rbar,alpha)
# Initialize # of paths number of points in each path
Delta = 1/252
N = ceiling((T-t0)/Delta)

set.seed(123)
X=CIR.Trans.sim(t0=0,T=10,Delta=1/252,r0=0.02,
                gamma=0.3807,rbar=0.072,alpha=0.0548,M=10)

M = ncol(X)
euler.est= matrix(NA,3,M)
gmm.est = matrix(NA,3,M)
Wsample = NULL
for (i in 1:M){
 x1 = X[1:N,i]^(-0.5)
 x2 = X[1:N,i]^(0.5)
 y = X[2:(N+1),i]*x1
 model=lm(y~0+x1+x2)
 euler.est[1,i] = (1-as.numeric(model$coefficients)[2])/Delta
 euler.est[2,i] = as.numeric(model$coefficients)[1]/
                  (1-as.numeric(model$coefficients)[2])
 euler.est[3,i] = sigma(model)^2/Delta
 model2 = optim(c(as.numeric(model$coefficients)[1],
                  as.numeric(model$coefficients)[2],sigma(model))
                ,CIR.gmm,NULL,X[,i],method="L-BFGS-B")
 gmm.est[1,i] = (1- model2$par[2])/Delta
 gmm.est[2,i] = model2$par[1]/(1-model2$par[2])
 gmm.est[3,i] = model2$par[3]^2/Delta
}
# performance measurements
bias.euler = rowMeans(euler.est-paras)
sd.euler = rowSds(euler.est)
rmse.euler = (rowMeans((euler.est-paras)^2))^0.5
bias.gmm = rowMeans(gmm.est-paras)
sd.gmm = rowSds(gmm.est)
rmse.gmm = (rowMeans((gmm.est-paras)^2))^0.5
```

Simulation Performance Based
On 10 Sample Paths

	Parameter		
	γ	$\bar{r}$	α
Bias (Euler method)	0.38199	−0.00882	-2.74156×10^{-4}
Bias (GMM)	0.38199	−0.00887	-2.74157×10^{-4}
Standard deviation (Euler)	0.34194	0.01973	0.00157
Standard deviation (GMM)	0.34193	0.01971	0.00157
Root mean square (Euler)	0.50114	0.02069	0.00152
Root mean square (GMM)	0.50114	0.02069	0.00152

As we can see from the results, the difference between the Euler method and this method is minor. Mathematically, we can prove that the Euler method and GMM would produce identical estimates in the CIR model.

3.5 THE CIR MODEL: RISK-NEUTRAL CALIBRATION

This section is very similar to the Section 3.2 "Vasicek model: risk-neutral calibration". The main difference is, in this section we use CIR model zero coupon bond pricing formulas (15.70) to (15.72) of [24] which are listed below, instead of the formulas in Section 3.2.

[24](15.70) $$Z(r,t;T) = e^{A(t;T)-B(t;T)\times r}$$

[24](15.71) $$B(t;T) = \frac{2\left(e^{\psi_1(T-t)}-1\right)}{(\gamma^*+\psi_1)\left(e^{\psi_1(T-t)}-1\right)+2\psi_1}$$

[24](15.72) $$A(t;T) = 2\frac{\bar{r}^*\gamma^*}{\alpha}\log\left(\frac{2\psi_1 e^{(\psi_1+\gamma^*)\frac{(T-t)}{2}}}{(\gamma^*+\psi_1)\left(e^{\psi_1(T-t)}-1\right)+2\psi_1}\right), \text{ and } \psi_1 = \sqrt{(\gamma^*)^2+2\alpha}$$

The calibration process involves minimizing the following quantity for a given value of α:

$$J(\gamma^*,\bar{r}^*) = \sum_{i=1}^{n}\left(Z^{CIR}(r_0,0;T_i) - Z^{Data}(0,T_i)\right)^2$$

We can use either "nlm" or "optim" for this purpose as before. The following example first calculates a bond price vector for given maturities with known parameter values and then calculates parameters using the minimization. It is used as a verification of our code for the minimization as in the Vasicek case.

```
#example22
rm(list=ls()) # clear the workspace and functions
bond.maturities = seq(0.5,10,0.5)
bond.prices = CIR.zcbp(r0=0.02,t=0,T=bond.maturities,gamma=0.5,
                       rbar=0.07,alpha=0.05)
model = nlm(f=CIR.J,p=c(0.4,0.05),r0=0.02,alpha=0.05,
            bond.prices=bond.prices,
            bond.maturities=bond.maturities)
model1 = optim(c(0.4,0.05),CIR.J,method=("BFGS")
               ,r0=0.02,alpha=0.05,
               bond.prices=bond.prices,
               bond.maturities=bond.maturities)
```

CIR Risk-Neutral Parameter Estimates With $\alpha = 0.05$

Method	Parameter		Constraint
	γ^*	$\bar{r}^*$	$\gamma^* \times \bar{r}^* > \frac{1}{2}\alpha$
nlm	0.49999	0.07	TRUE
optim	0.49746	0.07011	TRUE

As in the Vasicek case we could try to estimate all three parameters from bond pricing data.

```
#example23
rm(list=ls()) # clear the workspace and functions
bond.maturities = seq(0.5,10,0.5)
bond.prices = CIR.zcbp(r0=0.02,t=0,T=bond.maturities,gamma=0.5,
                       rbar=0.07,alpha=0.05)
model = nlm(f=CIR.J,p=c(0.3,0.05,0.02),r0=0.02,
            bond.prices=bond.prices,
            bond.maturities=bond.maturities)
model1 = optim(c(0.3,0.05,0.02),lower = c(0,0,0),CIR.J,method=("L-BFGS-B")
               ,r0=0.02,bond.prices=bond.prices,
               bond.maturities=bond.maturities)
```

After a few initial guesses, we see that the results given in the table below are not satisfactory.

CIR Risk-neutral Parameter Estimates

Method	Parameter			Constraint
	γ^*	$\bar{r}^*$	α	$\gamma^* \times \bar{r}^* > \frac{1}{2}\alpha$
nlm	0.353	0.08847	0.13215	FALSE
optim	0.35041	0.08895	0.13359	FALSE

Therefore, we do not recommend using bond prices to estimate α.

As in the Vasicek case, instead of writing our own function to minimize we can use the "nls" package for non-linear least squares minimization. The following code snippet illustrates usage of "nls" with bond yields.

```
#example24
rm(list=ls())
CIR.yield = function(r0=0.02,t=0,T=1,gamma=0.3807,rbar=0.072,alpha=0.0548){
  psi1 = (gamma^2+2*alpha)^0.5
  Denom = (gamma+psi1)*(exp(psi1*(T-t))-1)+ 2*psi1
  B = 2*(exp(psi1*(T-t))-1)/Denom
  A = 2*rbar*gamma/alpha* log((2*psi1*exp((psi1+gamma)*(T-t)/2))/Denom)
  return((-A+B*r0)/(T-t))
}
bond.maturities = seq(0.5,10,0.5) #Time_to_maturity
bond.yields = CIR.yield(r0=0.02,t=0,T=bond.maturities,
                        gamma=0.5,rbar=0.07,alpha=0.05)
bonddata = data.frame(cbind(bond.maturities,bond.yields))
```

```
CIR.fit =
   nls(bond.yields~CIR.yield(r0=0.02,t=0,T=bond.maturities,
                                 gamma,rbar,alpha=0.05),
       start=list(gamma=0.2,rbar=0.02),
       data=bonddata,algorithm = "port",
       lower=list(gamma=0.1,rbar=0.01),
       upper=list(gamma=3,rbar=1),
       nls.control(maxiter = 100, tol = 1e-3, minFactor = 1/1024,
                   printEval = FALSE, warnOnly = FALSE,
                   scaleOffset = 0, nDcentral = FALSE))
sum_mod=summary(CIR.fit)
```

The "nls" algorithm converged in 9 iterations and summary statistics are:

The Vasicek Fit (Simulated Data): Summary Statistics

Parameter	Estimate	Std. Error	t-value	$Pr[>\|t\|]$
γ^*	0.5	2.80955×10^{-12}	1.77965×10^{11}	1.14742×10^{-192}
$\bar{r}^*$	0.07	1.34795×10^{-13}	5.19307×10^{11}	4.87627×10^{-201}

This illustrates that as in the Vasicek case "nls" performs better than both "optim" and "nlm" when the underlying distribution is CIR. As in the Vasicek case we may try to estimate the standard deviation parameter from the bond yield data as illustrated below:

```
#example25
rm(list=ls())
CIR.yield = function(r0=0.02,t=0,T=1,gamma=0.3807,rbar=0.072,alpha=0.0548){
   psi1 = (gamma^2+2*alpha)^0.5
   Denom = (gamma+psi1)*(exp(psi1*(T-t))-1)+ 2*psi1
   B = 2*(exp(psi1*(T-t))-1)/Denom
   A = 2*rbar*gamma/alpha* log((2*psi1*exp((psi1+gamma)*(T-t)/2))/Denom)
   return((-A+B*r0)/(T-t))
}
bond.maturities = seq(0.5,10,0.5) #Time_to_maturity
bond.yields = CIR.yield(r0=0.02,t=0,T=bond.maturities,
                            gamma=0.5,rbar=0.07,alpha=0.05)
bonddata = data.frame(cbind(bond.maturities,bond.yields))
CIR.fit =
   nls(bond.yields~CIR.yield(r0=0.02,t=0,T=bond.maturities,
       gamma,rbar,alpha),
       start=list(gamma=0.2,rbar=0.02,alpha=0.02),
       data=bonddata,algorithm = "port",
       lower=list(gamma=0.1,rbar=0.01,alpha=0.02),
       upper=list(gamma=3,rbar=1,alpha=1),
       nls.control(maxiter = 1000, tol = 1e-3, minFactor = 1/1024,
                   printEval = FALSE, warnOnly = FALSE,
                   scaleOffset = 0, nDcentral = FALSE))
```

The "nls" algorithm converged in 26 iterations and summary statistics are:

The Vasicek Fit (Simulated Data): Summary Statistics

Parameter	Estimate	Std. Error	t-value	$Pr[>\|t\|]$
γ^*	0.5	7.01055×10^{-9}	7.1321×10^{7}	1.71573×10^{-124}
$\bar{r}^*$	0.07	6.39192×10^{-10}	1.09513×10^{8}	1.17007×10^{-127}
σ	0.05	4.6444×10^{-9}	1.07657×10^{7}	1.56479×10^{-110}

For a practical example we could try using data from Table 15.1 of [24]:

```
#example26
rm(list=ls()) # clear the workspace and functions
CIR.yield = function(r0=0.02,t=0,T=1,gamma=0.3807,rbar=0.072,alpha=0.0548){
   psi1 = (gamma^2+2*alpha)^0.5
   Denom = (gamma+psi1)*(exp(psi1*(T-t))-1)+ 2*psi1
   B = 2*(exp(psi1*(T-t))-1)/Denom
   A = 2*rbar*gamma/alpha* log((2*psi1*exp((psi1+gamma)*(T-t)/2))/Denom)
   return((-A+B*r0)/(T-t))
}
bond.maturities = VeronesiTable15p1$'Time to Maturity'
bond.prices=VeronesiTable15p1$Strips/100
bond.yields = -log(bond.prices)/bond.maturities
bonddata = data.frame(cbind(bond.maturities,bond.yields))
model = nlm(f=CIR.J,p=c(0.4,0.05),r0=0.0168,alpha=0.0548,
                bond.prices=bond.prices,
                bond.maturities=bond.maturities)
model1 = optim(c(0.4,0.05),CIR.J,method=("BFGS")
                  ,r0=0.0168,alpha=0.0548,bond.prices=bond.prices,
                   bond.maturities=bond.maturities)
CIR.fit =
   nls(bond.yields~CIR.yield(r0=0.0168,t=0,T=bond.maturities,
                                gamma,rbar,alpha=0.0548),
       start=list(gamma=0.4,rbar=0.06),
       data=bonddata,algorithm = "port",
       lower=list(gamma=0.1,rbar=0.01),
       upper=list(gamma=10,rbar=10),
       nls.control(maxiter = 1000, tol = 1e-3, minFactor = 1/1024,
                   printEval = FALSE, warnOnly = FALSE,
                   scaleOffset = 0, nDcentral = FALSE))
sum_mod= summary(CIR.fit)
```

CIR Risk-Neutral Parameter Estimates With $\alpha = 0.0548$

Method	Parameter		Constraint
	γ^*	$\bar{r}^*$	$\gamma^* \times \bar{r}^* > \frac{1}{2}\alpha$
nlm	0.37982	0.07207	FALSE
optim	0.37876	0.07215	FALSE
nls	0.26401	0.08599	FALSE

We see that "optim" produced values that are close to the estimates of $\gamma^* = 0.3807$ and $\bar{r}^* = 7.2\%$ in footnote 10 of page 552 of [24]. However, the resulting estimates do not satisfy the required constraint of $\gamma \times \bar{r} > \alpha/2$.

The following are summary statistics of the "nls" method, which converged after 11 iterations:

The CIR Fit (Simulated Data): Summary Statistics

Parameter	Estimate	Std. Error	t-value	$Pr[>\|t\|]$
γ^*	0.26401	0.04652	5.67475	5.72022×10^{-6}
$\bar{r}^*$	0.08599	0.00847	10.15586	1.53483×10^{-10}

If α is also unknown, we need to modify the code as given below:

```
#example27
rm(list=ls()) # clear the workspace and functions
CIR.yield = function(r0=0.02,t=0,T=1,gamma=0.3807,rbar=0.072,alpha=0.0548){
   psi1 = (gamma^2+2*alpha)^0.5
   Denom = (gamma+psi1)*(exp(psi1*(T-t))-1)+ 2*psi1
   B = 2*(exp(psi1*(T-t))-1)/Denom
   A = 2*rbar*gamma/alpha* log((2*psi1*exp((psi1+gamma)*(T-t)/2))/Denom)
   return((-A+B*r0)/(T-t))
}
bond.maturities = VeronesiTable15p1$'Time to Maturity'
bond.prices = VeronesiTable15p1$Strips/100
bond.yields = -log(bond.prices)/bond.maturities
bonddata = data.frame(cbind(bond.maturities,bond.yields))
model = nlm(f=CIR.J,p=c(0.4,0.05,0.03),r0=0.0168,
                bond.prices=bond.prices,iterlim=2000,
                bond.maturities=bond.maturities)
model1 = optim(c(0.4,0.05,0.03),CIR.J,method=("BFGS")
                    ,r0=0.0168,bond.prices=bond.prices,
                    bond.maturities=bond.maturities)
CIR.fit =
   nls(bond.yields~CIR.yield(r0=0.0168,t=0,T=bond.maturities,
                                gamma,rbar,alpha),
       start=list(gamma=0.4,rbar=0.06,alpha=0.03),
       data=bonddata,algorithm = "port",
       lower=list(gamma=0.1,rbar=0.01,alpha=0.03),
       upper=list(gamma=10,rbar=10,alpha=1),
       nls.control(maxiter = 1000, tol = 1e-3, minFactor = 1/1024,
                    printEval = FALSE, warnOnly = FALSE,
                    scaleOffset = 0, nDcentral = FALSE))
sum_mod= summary(CIR.fit)
```

CIR Risk-Neutral Parameter Estimates

Method	Parameter			Constraint
	γ^*	$\bar{r}^*$	α	$\gamma^* \times \bar{r}^* > \frac{1}{2}\alpha$
nlm	0.02632	0.72611	0.18018	FALSE
optim	0.46538	0.06311	0.00727	TRUE
nls	0.1	0.19222	0.11924	FALSE

We see that only "optim" produces acceptable estimates but the resulting α value is not close to 0.0548. The following is a summary statistics of "nls" method which converged after 55 iterations:

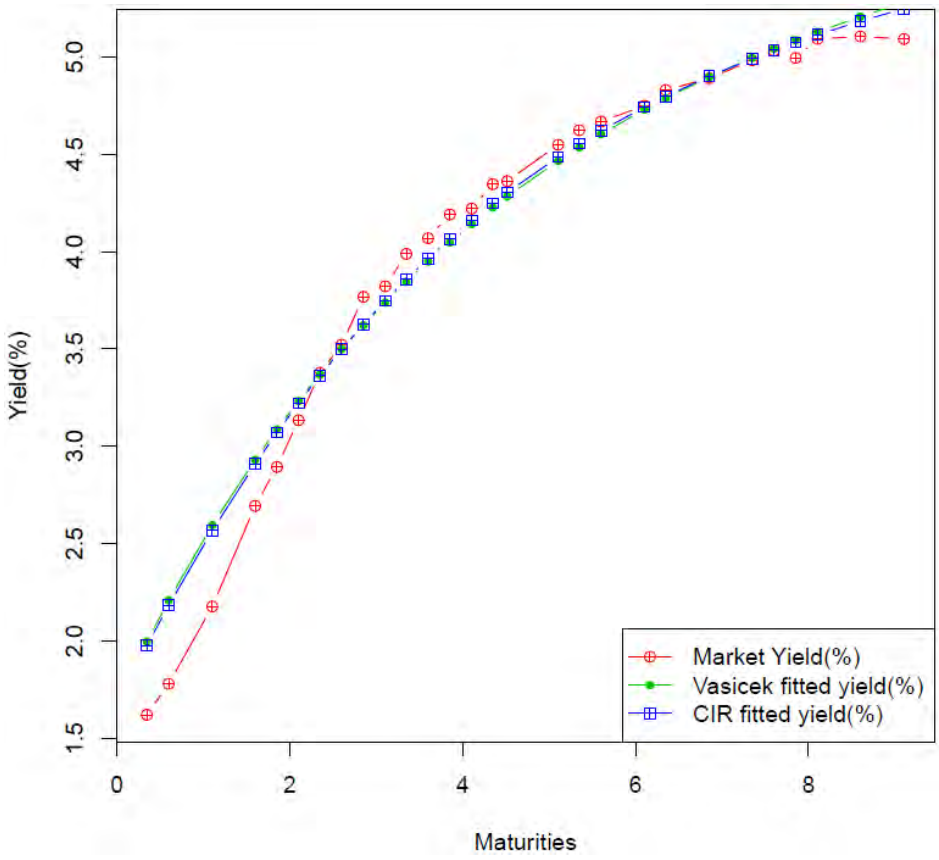
The CIR Fit (Simulated Data): Summary Statistics

Parameter	Estimate	Std. Error	t-value	$Pr(>\|t\|)$
γ^*	0.1	0.49444	0.20225	0.84136
$\bar{r}^*$	0.19222	0.837	0.22965	0.82023
α	0.11924	0.13725	0.86881	0.39322

Based on the summary statistics we can conclude that CIR is not suitable model for the data set. To evaluate this issue further we can look at the fit of the model with the Vasicek model.

```
#example28
rm(list=ls()) # clear the workspace and functions
CIR.yield = function(r0=0.02,t=0,T=1,gamma=0.3807,rbar=0.072,alpha=0.0548){
   psi1 = (gamma^2+2*alpha)^0.5
   Denom = (gamma+psi1)*(exp(psi1*(T-t))-1)+ 2*psi1
   B = 2*(exp(psi1*(T-t))-1)/Denom
   A = 2*rbar*gamma/alpha* log((2*psi1*exp((psi1+gamma)*(T-t)/2))/Denom)
   return((-A+B*r0)/(T-t))
}
bond.maturities = VeronesiTable15p1$'Time to Maturity'
bond.prices = VeronesiTable15p1$Strips/100
bond.yields = -log(bond.prices)/bond.maturities
bonddata = data.frame(cbind(bond.maturities,bond.yields))
Vasicek.fit =
   nls(bond.yields~Vasicek.yield(r0=0.0168,t=0,T=bond.maturities,
                                 gamma,rbar,sigma=0.0221),
       start=list(gamma=0.4,rbar=0.06),
       data=bonddata,algorithm = "port",
       lower=list(gamma=0.1,rbar=0.01),
       upper=list(gamma=10,rbar=10),
       nls.control(maxiter = 1000, tol = 1e-3, minFactor = 1/1024,
                   printEval = FALSE, warnOnly = FALSE,
                   scaleOffset = 0, nDcentral = FALSE))
```

```
CIR.fit =
   nls(bond.yields~CIR.yield(r0=0.0168,t=0,T=bond.maturities,
                                gamma,rbar,alpha),
       start=list(gamma=0.4,rbar=0.06,alpha=0.03),
       data=bonddata,algorithm = "port",
       lower=list(gamma=0.1,rbar=0.01,alpha=0.03),
       upper =list(gamma=10,rbar=10,alpha=1),
       nls.control(maxiter = 1000, tol = 1e-3, minFactor = 1/1024,
                   printEval = FALSE, warnOnly = FALSE,
                   scaleOffset = 0, nDcentral = FALSE))
plot(bond.maturities,bond.yields*100,xlab="Maturities",ylab="Yield(%)",
      type="b",pch=10,lty="solid",col="red1")
lines(bond.maturities,fitted(Vasicek.fit)*100,type="b",pch=20,lty="solid",
       col="green3")
lines(bond.maturities,fitted(CIR.fit)*100,type="b",pch=12,col="blue")
legend("bottomright",legend=c("Market Yield(%)","Vasicek fitted yield(%)",
                              "CIR fitted yield(%)"),col=c("red1","green3","blue"),
                              lty=c(1,1,1),pch=c(10,20,12))
```



From the plot we see that the CIR fit is marginally better than the Vasicek fit but the CIR fitted values may lead to negative interest rates; therefore we recommend using the Vasicek model for the given data.

3.6 THE TWO-FACTOR VASICEK MODEL CALIBRATION

The literature on real-world calibration of the two-factor Vasicek model is non-existent. Even the seminal text on interest rate models by Brigo (2009) focuses on calibrating in the risk-neutral environment. The difficulty in calibrating in the real world lies in selecting short term rates and then long term rates. However, short and long term rates are extracted from bond prices which in turn use the risk-neutral world to model and calculate yields. Essentially, to calibrate in the real world we need the observed values of $\phi_{1,t}$ and $\phi_{2,t}$ individually but the market data (say overnight rates) are r_t values. Therefore we have to use statistical mixture models to calibrate to real time data, but that is beyond the scope of this paper. As the appendix shows, zero-coupon bond prices under the two-factor Vasicek model are given by:

$$Z(\phi_{1,t}, \phi_{2,t}, t; T) = exp\left(A(t;T) - \phi_{1,t}B_1(t;T) - \phi_{2,t}B_2(t;T)\right) \tag{9}$$

where:

$$\begin{aligned} A(t;T) \;=\; & \overrightarrow{\phi}_1 (B_1(t;T) - (T-t)) - \frac{\sigma_1^2}{2}\left[\frac{1}{\gamma_1^2}(B_1(t;T) - (T-t)) + \frac{1}{2\gamma_1}B_1(t;T)^2\right] \\ & + \overrightarrow{\phi}_2 (B_2(t;T) - (T-t)) - \frac{\sigma_2^2}{2}\left[\frac{1}{\gamma_2^2}(B_2(t;T) - (T-t)) + \frac{1}{2\gamma_2}B_2(t;T)^2\right] \\ & - \frac{\rho\sigma_1\sigma_2}{\gamma_1\gamma_2}\left[B_1(t;T) + B_2(t;T) - B_3(t,T) - (T-t)\right] \end{aligned}$$

and

$$B_i(t;T) = \int_0^{T-t} exp(-\gamma_i s)\, ds, i = 1,2,3$$

With $\gamma_3 = \gamma_1 + \gamma_2$. We write $B_i(t;T)$ in an integral form as it is easy to evaluate it for $\gamma_i = 0$ and $\gamma_i \neq 0$. The calibration process involves minimizing the following quantity:

$$J(\gamma^*, \bar{r}^*) = \sum_{i=1}^{n}\left(Z^{Two-factor-Vasicek}(r_0, 0; T_i) - Z^{Data}(0, T_i)\right)^2 \tag{10}$$

where $Z^{Two-factor-Vasicek}(r_0, 0; T_i)$ is as given in (9). As we need to estimate three parameters, we use a nonlinear least square minimization algorithm for this task. The R package "nls" is suitable for this task. In some cases γ_1 or γ_2 may turn out to be negative and in iterative computations the parameter values may go through zero; therefore it is important to know the behavior of $A(t;T)$ and $B(t;T)$ in the neighborhood of zero. As the appendix points out we can write:

- When $\gamma_1 = 0, \gamma_2 \neq 0$

$$\begin{aligned} A(t;T) \;=\; & \overrightarrow{\phi}_2 (B_2(t;T) - (T-t)) + \sigma_1^2\frac{(T-t)^3}{6} - \frac{\sigma_2^2}{2}\left[\frac{1}{\gamma_2^2}(B_2(t;T) - (T-t)) + \frac{1}{2\gamma_2}B_2(t;T)^2\right] \\ & + \rho\sigma_1\sigma_2\left[\frac{(T-t)^2}{2} + \frac{\exp(-\gamma_2(T-t))(\gamma_2(T-t)+1) - 1}{\gamma_2^2}\right] \end{aligned}$$

- When $\gamma_1 \neq 0, \gamma_2 = 0$

$$\begin{aligned} A(t;T) \;=\; & \overrightarrow{\phi}_1 (B_1(t;T) - (T-t)) - \frac{\sigma_1^2}{2}\left[\frac{1}{\gamma_1^2}(B_1(t;T) - (T-t)) + \frac{1}{2\gamma_1}B_1(t;T)^2\right] + \sigma_2^2\frac{(T-t)^3}{6} \\ & + \rho\sigma_1\sigma_2\left[\frac{(T-t)^2}{2} + \frac{\exp(-\gamma_1(T-t))(\gamma_1(T-t)+1) - 1}{\gamma_1^2}\right] \end{aligned}$$

- When $\gamma_1 = \gamma_2 = 0$

$$A(t;T) = (\sigma_1^2 + \sigma_2^2 + 2\rho\sigma_1\sigma_2)\frac{(T-t)^3}{6}$$

The R function "Two.factor.Vasicek.A" implements these formulas. For the calibration we assume we have the following data after choosing a possible long-term rate (for example 5 years or 10 years):

- Volatility of short-term rate dr_t, σ_τ
- Volatility of long-term rate $dr_t(\tau), \sigma(\tau)$
- Correlation between short-term rate dr_t and long-term rate $dr_t(\tau), \rho(0,\tau)$
- Series of zero-coupon strip prices.

For each set of given values of $\gamma_1^*, \gamma_2^*, \vec{\phi}_1^*$ and $\vec{\phi}_2^*$ we solve following three simultaneous equations for σ_1, σ_2 and ρ:

$$(22.51) \quad \sigma_\tau = \sqrt{\sigma_1^2 + \sigma_2^2 + 2\sigma_1\sigma_2\rho}$$

$$(22.52) \quad \sigma(\tau) = \sqrt{\sigma_1^2\left(\frac{B_1(\tau)}{\tau}\right)^2 + \sigma_2^2\left(\frac{B_2(\tau)}{\tau}\right)^2 + 2\left(\frac{B_1(\tau)}{\tau}\right)\left(\frac{B_2(\tau)}{\tau}\right)\sigma_1\sigma_2\rho}$$

$$(22.53) \quad \rho(0,\tau) = \frac{\sigma_1^2\left(\frac{B_1(\tau)}{\tau}\right) + \sigma_2^2\left(\frac{B_2(\tau)}{\tau}\right) + \left(\frac{B_1(\tau)}{\tau} + \frac{B_2(\tau)}{\tau}\right)\sigma_1\sigma_2\rho}{\sigma_\tau\sigma(\tau)}$$

Although the above three equations appear to be non-linear, they can be converted into linear equations of σ_1, σ_2 and *cova*, where *cova* is defined as $cova = \rho\sigma_1\sigma_2$. The R function "Two.factor.Vasicek.Vols" given below implements this.

```
Two.factor.Vasicek.Vols = function(phi1=-0.0413,phi2=0,gamma1=0.8269,
                                   gamma2=-0.0288,Tau=5,short.term.vol=0.0221,
                                   long.term.vol=0.0125,correlation=0.4713){
B1Tau = Vasicek.B(gamma=gamma1,T=Tau)/Tau
B2Tau = Vasicek.B(gamma=gamma2,T=Tau)/Tau
if (correlation==0) {
  A = matrix(c(1,1,B1Tau^2,B2Tau^2),nrow=2,ncol=2,byrow=TRUE)
  B = matrix(c(short.term.vol^2,long.term.vol^2),nrow=2,ncol=1)
  est= solve(A,B)
  paras = c(est[1]^0.5,est[2]^0.5)
  return(paras)
}
else {
  cova = short.term.vol*long.term.vol*correlation
```

```
  A = matrix(c(1,1,2,B1Tau^2,B2Tau^2,2*B1Tau*B2Tau,B1Tau,B2Tau,
               (B1Tau+B2Tau)),nrow=3,ncol=3,byrow=TRUE)
  B = matrix(c(short.term.vol^2,long.term.vol^2,cova),nrow=3,ncol=1)
  est= solve(A,B)
  paras = c(est[1]^0.5,est[2]^0.5,est[3]/(est[1]*est[2])^0.5)
  return(paras)
  }
}
```

Use these values of σ_1, σ_2 and ρ in (9) along with strip prices to minimize (10) with respect to $\gamma_1^*, \gamma_2^*, \vec{\phi}_1^*$ and $\vec{\phi}_2^*$. In an iterative scheme with each set of new values of $\gamma_1^*, \gamma_2^*, \vec{\phi}_1^*$ and $\vec{\phi}_2^*$, the parameters σ_1, σ_2 and ρ have to be calculated. Therefore, to feed to non-linear regression analysis we use the following routine for log bond yield calculation under the two-factor Vasicek model.

```
Two.factor.Vasicek.yield = function(t,T,Tau,rt,rlTau,phi1,phi2,gamma1,gamma2,
                                    short.term.vol,long.term.vol,correlation)
{
  est = Two.factor.Vasicek.Vols(phi1,phi2,gamma1,gamma2,Tau,
                                short.term.vol,long.term.vol,correlation)
  sigma1= est[1]
  sigma2 = est[2]
  rho = est[3]
  B1Tau = Vasicek.B(gamma=gamma1,T=Tau)/Tau
  B2Tau = Vasicek.B(gamma=gamma2,T=Tau)/Tau
  ATau = Two.factor.Vasicek.A(phi1,phi2,gamma1,gamma2,sigma1,sigma2,rho,Tau)
  phi1t = (B2Tau*rt-Tau*rlTau-ATau)/(B2Tau-B1Tau)
  phi2t = (Tau*rlTau+ATau-rt*B1Tau)/(B2Tau-B1Tau)
  AtT = Two.factor.Vasicek.A(phi1,phi2,gamma1,gamma2,sigma1,sigma2,rho,(T-t))
  B1tT = Vasicek.B(gamma=gamma1,(T-t))
  B2tT = Vasicek.B(gamma=gamma2,(T-t))
  return ((-AtT+B1tT*phi1t+B2tT*phi2t)/(T-t))
}
```

The final call for the minimization is carried out as given below:

```
#example29
# Two factor Vasicek risk-neutral calibration with simulated data example
rm(list=ls()) # clear the workspace and functions
bond.maturities =seq(0.5,20,0.5)
set.seed(12345)
```

```
bond.prices = Two.factor.Vasicek.zcbpv2(t=0,T=bond.maturities,Tau=5,
                                        rt=0.0168,rlTau=0.0452,
                                        phi1=-0.0413,phi2=0,
                                        gamma1=0.8269,gamma2=-0.0288,
                                        short.term.vol = 0.0221,
                                        long.term.vol = 0.0125,
                                        correlation = 0.4713)
bonddata = data.frame(cbind(bond.maturities,bond.prices))

Two.factor.Vasicek.fit =
 nls(bond.prices~Two.factor.Vasicek.zcbpv2(t=0,T=bond.maturities,
                                           Tau=5,rt=0.0168,
                                           rlTau=0.0452,
                                           phi1,phi2=0,
                                           gamma1,gamma2,
                                           short.term.vol = 0.0221,
                                           long.term.vol = 0.0125,

                    start=list(phi1=-0.1,gamma1=0.08,gamma2=-0.1),
                    data=bonddata,algorithm = "port",
                    lower=list(phi1=-1,gamma1=0,gamma2=-0.5),
                    upper =list(phi1=1,gamma1=0.999,gamma2=-0.01),
                    nls.control(maxiter = 100, tol = 1e-3, minFactor = 1/1024,
                                printEval = FALSE, warnOnly = FALSE,
                                scaleOffset = 0, nDcentral = FALSE))
sum_mod = summary(Two.factor.Vasicek.fit)
est.nls=
Two.factor.Vasicek.Vols(phi1=as.numeric(coef(Two.factor.Vasicek.fit))[1],
                        phi2=0,
                        gamma1=as.numeric(coef(Two.factor.Vasicek.fit))[2],
                        gamma2=as.numeric(coef(Two.factor.Vasicek.fit))[3],
                        Tau=5,short.term.vol=0.0221,
                            long.term.vol=0.0125,correlation=0.4713)
```

The "nls" algorithm converged in 74 iterations and summary statistics are:

The Two-Factor Vasicek Fit (Simulated Data): Summary Statistics

Parameter	Estimate	Std. Error	t-value	$Pr[>\|t\|]$
$\vec{\phi}_1^*$	-0.0413	3.50551×10^{-14}	-1.17815×10^{12}	0
γ_1^*	0.8269	4.62426×10^{-14}	1.78818×10^{13}	0
γ_2^*	-0.0288	1.09881×10^{-14}	-2.62101×10^{12}	0

The following values are obtained using the estimated values of $\vec{\phi}_1^*, \gamma_1^*$ and γ_2^*:

Parameter	Estimate
σ_1	0.02508
σ_2	0.01318
ρ	-0.47532

From the summary statistics we see that the "nls" function converges to its true value quite easily. Now we can test the fit using an actual data set. For this purpose we use Table 15.1 of [24].

```
#example30
rm(list=ls()) # clear the workspace and functions
bond.maturities = VeronesiTable15p1$'Time to Maturity'
bond.yields = -log(VeronesiTable15p1$Strips/100)/bond.maturities
bonddata = data.frame(cbind(bond.maturities,bond.yields))
Two.factor.Vasicek.fit =
  nls(bond.yields~Two.factor.Vasicek.yield(t=0,T=bond.maturities,
                                           Tau=5,rt=0.0168,rlTau=0.0452,
                                           phi1,phi2=0,gamma1,gamma2,
                                           short.term.vol = 0.0221,
                                           long.term.vol = 0.0125,
                                           correlation = 0.4713),
                                  start=list(phi1=-0.08,gamma1=0.7,gamma2=-0.04),
                                  data=bonddata,algorithm = "port",
                                  lower=list(phi1=-0.9,gamma1=-2,gamma2=-2),
                                  upper=list(phi1=0.9,gamma1=2,gamma2=2),
                                  nls.control(maxiter = 10000, tol = 1e-8,
                                              minFactor = 1/10240,
                                              printEval = FALSE,
                                              warnOnly = TRUE, scaleOffset = 0,
                                              nDcentral = FALSE),trace=FALSE)

## Warning in nls(bond.yields ~ Two.factor.Vasicek.yield(t = 0, T = bond.maturities,
:   Convergence failure:  false convergence (8)

sum_mod = summary(Two.factor.Vasicek.fit)
est.nls = Two.factor.Vasicek.Vols(
    gamma1 = as.numeric(coef(Two.factor.Vasicek.fit))[2],
    gamma2 = as.numeric(coef(Two.factor.Vasicek.fit))[3],
                Tau=5,short.term.vol=0.0221,long.term.vol=0.0125,correlation=0.4713)
```

The "nls" algorithm converged in 12 iterations and summary statistics are:

The Two-Factor Vasicek Fit: Summary Statistics

Parameter	Estimate	Std. Error	t-value	$Pr[>\|t\|]$
$\vec{\phi}_1^*$	-0.9	21.90681	-0.04108	0.96756
γ_1^*	0.34742	0.15571	2.23117	0.03487
γ_2^*	1.45913×10^{-4}	3.27303×10^{-7}	445.80356	2.77736×10^{-50}

The following values are obtained using the estimated values of $\vec{\phi}_1^*, \gamma_1^*$ and γ_2^*:

Parameter	Estimate
σ_1	0.0373
σ_2	0.02273
ρ	-0.83721

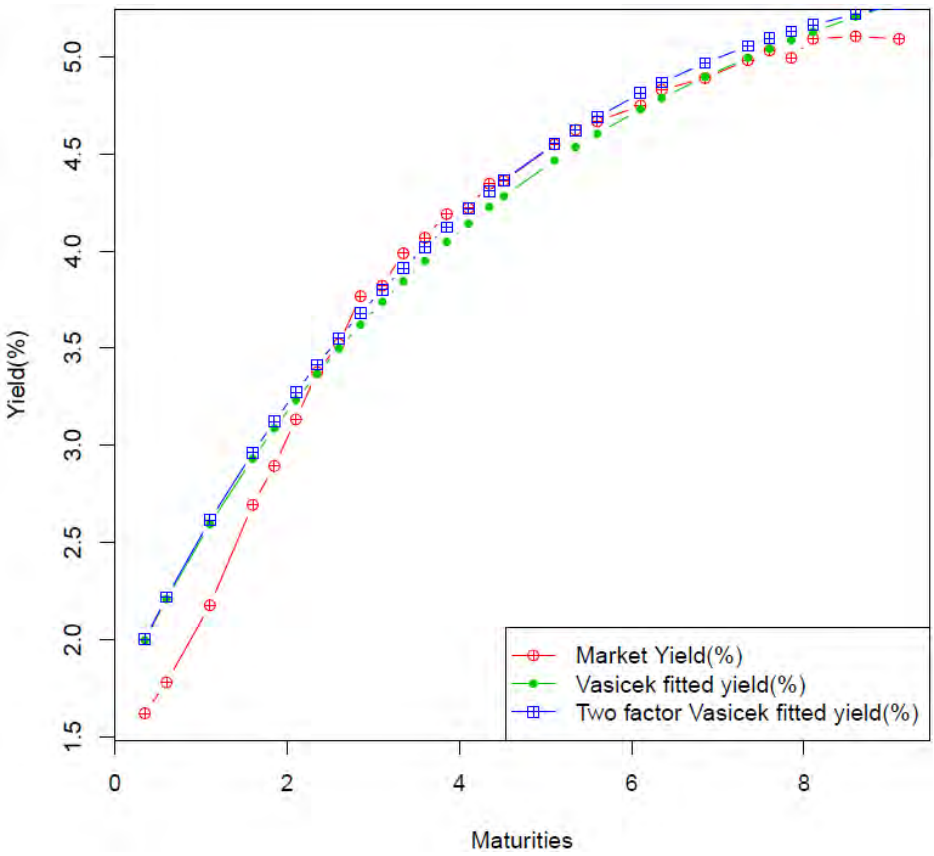
We tried many possible starting values but it converged only to these values. However the convergence was always accompanied a warning message "false convergence(8)". It may mean the yield rate function may not be continuous in the neighborhood of converging values. One may refer to [14] complete list of return codes[2].

The summary statistics suggests that the model is not appropriate. To examine further we plot the fitted value based on the two-factor Vasicek model along with the Vasicek model:

```
#example31
Vasicek.fit =
  nls(bond.yields~Vasicek.yield(r0=0.0168,t=0,T=bond.maturities,
                                gamma,rbar,sigma=0.0221),
      start=list(gamma=0.4,rbar=0.06),
      data=bonddata,algorithm = "port",
      lower=list(gamma=0.1,rbar=0.01),
      upper =list(gamma=10,rbar=10),
      nls.control(maxiter = 100, tol = 1e-3, minFactor = 1/1024,
                  printEval = FALSE, warnOnly = FALSE,
                  scaleOffset = 0, nDcentral = FALSE))

plot(bond.maturities,bond.yields*100,xlab="Maturities",ylab="Yield(%)",
     type="b",pch=10,lty="solid",col="red1")
lines(bond.maturities,fitted(Vasicek.fit)*100,type="b",pch=20,
      lty="solid",col="green3")
lines(bond.maturities,fitted(Two.factor.Vasicek.fit)*100,type="b",
      pch=12,col="blue")
legend("bottomright",legend=c("Market Yield(%)","Vasicek fitted yield(%)",
                              "Two factor Vasicek fitted yield(%)"),
       col=c("red1","green3","blue"),lty=c(1,1,1),pch=c(10,20,12))
```

[2] return code 8: false convergence: the gradient $\nabla f(x)$ may be computed incorrectly, the other stopping tolerances may be too tight, or either f or ∇f may be discontinuous near the current iterate x.



The diagnostic p-values indicate that the two-factor Vasicek model is not an appropriate model for this data set. This agrees with the conclusion drawn in [24].

4 No-Arbitrage Models

When we fit risk-neutral versions of Vasicek and CIR models to a term structure extracted from Treasury bond prices we see that the resulting models are not exact fits, they just minimize the total error between market values and model values. To eliminate this mismatch, Hull-White (1990, 1993) extend Vasicek models by including a time dependent drift parameter. The two-factor Hull-White model and the LIBOR Market models are two more examples of no arbitrage models.

4.1 HULL-WHITE MODELS

In this section we describe the single factor Hull-White model and the two factor Hull-White model. In theory arbitrage-free models reproduce the yield curve and prices for a set of interest rate derivatives such as caps, floors or swaptions. In the calibration process that we outline, we see that they do not precisely reproduce the yield curve and derivative prices, but the error is minimal.

4.1.1 ONE FACTOR HULL-WHITE MODEL

The one factor Hull-White model is an extension of the Vasicek model with a time-varying drift coefficient given below:

$$dr_t = (\theta_t - \gamma^* r_t)dt + \sigma dX_t$$

The solution of this is:

$$\begin{aligned} r_{t+s} &= r_t \exp(-\gamma^* s) + \exp(-\gamma^*(t+s)) \int_t^{t+s} \theta_u \exp(\gamma^* u) du + \sigma \exp(-\gamma^* s) \int_0^s \exp(\gamma^* u) dX_u \\ E[r_{t+s}|r_t] &= r_t \exp(-\gamma^* s) + \exp(-\gamma^*(t+s)) \int_t^{t+s} \theta_u \exp(\gamma^* u) du \\ Var[r_{t+s}|r_t] &= [\sigma \exp(-\gamma^* s)]^2 \int_0^s \exp(2\gamma^* u) du = \frac{\sigma^2}{2\gamma^*}(1 - e^{-2\gamma^* s}) \end{aligned}$$

As the Hull-White model is a Gaussian model (Ornstein-Uhlenbeck process[3]) the zero-coupon bond prices take the form:

$$\begin{aligned} &[24](19.25) \quad Z(r_0, 0; T) = e^{A(0;T) - B(0;T) r_0} \\ &[24](19.26) \quad B(0;T) = \frac{1}{\gamma^*}\left(1 - e^{-\gamma^* T}\right) \\ &[24](19.27) \quad A(0;T) = -\int_0^T B(0; T-t)\theta_t dt + \frac{\sigma^2}{2(\gamma^*)^2}\left(T + \frac{1 - e^{-2\gamma^* T}}{2\gamma^*} - 2B(0;T)\right) \end{aligned}$$

When the Hull-White model is calibrated to be arbitrage-free, market discount factors should match the model price. We can solve for function θ_t as illustrated below:

$$\begin{aligned} Z(r_0, 0; T) &= e^{A(0;T) - B(0;T) r_0} \\ \ln(Z(r_0, 0; T)) &= A(0;T) - B(0;T) r_0 \\ \frac{\partial \ln(Z(r_0, 0; T))}{\partial T} &= \frac{\partial A(0;T)}{\partial T} - r_0 \frac{\partial B(0;T)}{\partial T} \end{aligned}$$

But

$$\begin{aligned} f(0,T) &= -\frac{\partial \ln(Z(r_0, 0; T))}{\partial T} \\ \frac{\partial B(0;T)}{\partial T} &= \exp(-\gamma^* T) \\ \frac{\partial A(0;T)}{\partial T} &= -B(0; T-T)\theta_t - \int_0^T \frac{\partial B(0; T-t)}{\partial T}\theta_t dt + \frac{\sigma^2}{2(\gamma^*)^2}\left(1 + \exp(-2\gamma^* T) - \frac{\partial B(0;T)}{\partial T}\right) \\ \frac{\partial B(0; T-t)}{\partial T} &= \exp(-\gamma^*(T-t)) \\ \frac{\partial A(0;T)}{\partial T} &= -\int_0^T \exp(-\gamma^*(T-t))\theta_t dt + \frac{\sigma^2}{2(\gamma^*)^2}(1 - \exp(-\gamma^* T) + \exp(-2\gamma^* T)) \\ f(0,T) &= \int_0^T \exp(-\gamma^*(T-t))\theta_t dt - \frac{\sigma^2}{2(\gamma^*)^2}(1 - \exp(-\gamma^* T) + \exp(-2\gamma^* T)) + r_0 \exp(-\gamma^* T) \\ f(0,T) &= \int_0^T \exp(-\gamma^*(T-t))\theta_t dt - \frac{\sigma^2}{2(\gamma^*)^2}(1 - \exp(-\gamma^* T))^2 + r_0 \exp(-\gamma^* T) \\ f(0,T)\exp(\gamma^* T) &= \int_0^T \exp(\gamma^* t)\theta_t dt - \frac{\sigma^2}{2(\gamma^*)^2}(1 - \exp(-\gamma^* T))^2 \exp(\gamma^* T) + r_0 \end{aligned}$$

Differentiating both sides with respect to T we can obtain an expression for θ_T

[3] An Ornstein-Uhlenbeck process is a Weiner process (random walk) modified so that it mean-reverts.

$$\begin{aligned}
&\frac{\partial f(0,T)}{\partial T}\exp(\gamma^* T) + f(0,T)\gamma^*\exp(\gamma^* T) \\
&= \exp(\gamma^* T)\theta_T - \frac{\sigma^2}{2(\gamma^*)^2}2\left(1-\exp(-\gamma^* T)\right)\exp(-\gamma^* T)\exp(\gamma^* T) - \\
&\quad \frac{\sigma^2}{2(\gamma^*)^2}\left(1-\exp(-\gamma^* T)\right)^2\gamma^*\exp(\gamma^* T) \\
&= \exp(\gamma^* T)\theta_T - \frac{\sigma^2}{2\gamma^*}(1-\exp(-2\gamma^* T))\exp(\gamma^* T) \\
\theta_T &= \frac{\partial f(0,T)}{\partial T} + \gamma^* f(0,T) + \frac{\sigma^2}{2\gamma^*}(1-\exp(-2\gamma^* T))
\end{aligned}$$

We notice that to evaluate $E[r_{s+t}|r_t]$, we need $\int_t^{s+t}\theta_u exp(\gamma^* u)\,du$. However, this integral can be simplified as illustrated below.

We already have the following from the derivation of θ_t:

$$\int_0^T \exp(\gamma^* t)\theta_t dt = -r_0 + f(0,T)\exp(\gamma^* T) + \frac{\sigma^2}{2(\gamma^*)^2}\left(1-\exp(-\gamma^* T)\right)^2\exp(\gamma^* T)$$

From this we see:

$$\begin{aligned}
\int_t^{t+s}\exp(\gamma^* u)\theta_u du &= \int_0^{t+s}\exp(\gamma^* u)\theta_u du - \int_0^t \exp(\gamma^* u)\theta_u du \\
&= -r_0 + f(0,t+s)\exp(\gamma^*(t+s)) + \frac{\sigma^2}{2(\gamma^*)^2}\left(1-\exp(-\gamma^*(t+s))\right)^2\exp(\gamma^*(t+s)) \\
&\quad -\left[-r_0 + f(0,t)\exp(\gamma^* t) + \frac{\sigma^2}{2(\gamma^*)^2}\left(1-\exp(-\gamma^* t)\right)^2\exp(\gamma^* t)\right] \\
&= f(0,t+s)\exp(\gamma^*(t+s)) - f(0,t)\exp(\gamma^* t) \\
&\quad +\frac{\sigma^2}{2(\gamma^*)^2}\left(1-\exp(-\gamma^*(t+s))\right)^2\exp(\gamma^*(t+s)) \\
&\quad -\frac{\sigma^2}{2(\gamma^*)^2}\left(1-\exp(-\gamma^* t)\right)^2\exp(\gamma^* t) \\
&= f(0,t+s)\exp(\gamma^*(t+s)) - f(0,t)\exp(\gamma^* t) + \\
&\quad \frac{\sigma^2}{2(\gamma^*)^2}\left[\exp(\gamma^*(t+s)) - \exp(\gamma^* t) + \exp(-\gamma^*(t+s)) - \exp(-\gamma^* t)\right]
\end{aligned}$$

Therefore, we have the following expression for $E[r_{s+t}|r_t]$:

$$\begin{aligned}
E[r_{t+s}|r_t] &= r_t\exp(-\gamma^* s) + f(0,s+t) - f(0,t)\exp(-\gamma^* s) + \\
&\quad \frac{\sigma^2}{2(\gamma^*)^2}\left[1-\exp(-\gamma^* s) + \exp(-2\gamma^*(t+s)) - \exp(-\gamma^*(2t+s))\right]
\end{aligned}$$

This expression along with the expression for the $Var[r_{s+t}|r_t]$ can be used to simulate interest rate paths from the Hull-White model.

4.1.2 THE TWO-FACTOR HULL-WHITE MODEL

The two-factor Hull-White model is a generalization of the two-factor Vasicek model. As in the two-factor Vasicek model, short rates are the sum of two factors,

$$r_t = \phi_{1,t} + \phi_{2,t},$$

Where each factor follows the following SDEs:

$$\begin{aligned} d\phi_{1,t} &= (\theta_t - \gamma_1^*\phi_{1,t})dt + \sigma_1 dX_{1,t} \\ d\phi_{2,t} &= -\gamma_2^*\phi_{2,t}dt + \sigma_2 dX_{2,t} \\ dX_{1,t}dX_{2,t} &= \rho dt \end{aligned}$$

As given in ([24]) this is in fact the two-factor Vasicek model with $\gamma_1^*\vec{\phi}_1^*$ replaced by a time-varying function θ_t and setting $\vec{\phi}_2 = 0$. As in the two-factor Vasicek model, the SDE for each factor can be solved to obtain the following:

$$\begin{aligned} \phi_{1,t+s} &= \phi_{1,t}\exp(-\gamma_1^* s) + \exp(-\gamma_1^*(t+s))\int_t^{t+s}\theta_u \exp(\gamma_1^* u)du + \sigma_1\exp(-\gamma_1^* s)\int_0^s \exp(\gamma_1^* u)dX_{1,u} \\ \phi_{2,t+s} &= \phi_{2,t}\exp(-\gamma_2^* s) + \sigma_2\exp(-\gamma_2 s)\int_0^s \exp(\gamma_2 v)dX_{2,v} \end{aligned}$$

Hence we can obtain the following expressions for conditional expectations, variances and covariance:

$$\begin{aligned} \mathbb{E}[\phi_{1,t+s}|\phi_{1,t}] &= \phi_{1,t}\exp(-\gamma_1^* s) + \exp(-\gamma_1^*(t+s))\int_t^{t+s}\theta_u\exp(\gamma_1^* u)du \\ \mathbb{E}[\phi_{2,t+s}|\phi_{2,t}] &= \phi_{2,t}\exp(-\gamma_2^* s) \\ \mathbb{V}ar[\phi_{1,t+s}|\phi_{1,t}] &= = \sigma_1^2\frac{1-\exp(-2\gamma_1^* s)}{2\gamma_1} \\ \mathbb{V}ar[\phi_{2,t+s}|\phi_{2,t}] &= = \sigma_2^2\frac{1-\exp(-2\gamma_2^* s)}{2\gamma_2} \\ \mathbb{C}ov[\phi_{1,t+s},\phi_{2,t+s}|\phi_{1,t},\phi_{2,t}] &= \frac{\rho\sigma_1\sigma_2}{\gamma_1+\gamma_2}(1-\exp(-(\gamma_1+\gamma_2)s)) \end{aligned}$$

As we worked out for the one-factor Hull-White model we can obtain expressions for θ_t in terms of instantaneous forward rates, $f(0,t)$, and $\gamma_1^*, \gamma_2^*, \sigma_1^*$ and σ_2^* and hence an expression for $E[r_{s+t}|r_t]$ but it is a quite long and involved expression, so we avoid reproducing it here.

4.1.3 HULL-WHITE MODEL CALIBRATION

Both one-factor and two-factor Hull-White model calibrations involve two steps. In the first step the yield curve will be interpolated to obtain values for chosen derivative cash flow timings. Then the time independent parameters of the Hull-White model are estimated using the interpolated yield curve, derivative prices, and non-linear least square methods. Least square methods can be used as long as the number of parameters is smaller than the available derivative prices. Then, using these estimated parameters, the time dependent parameter θ_t will be calculated.

The yield curve interpolation stage is the same for both one-factor and two-factor Hull-White models. For the yield curve interpolation one may use cubic splines, higher degree (degree of six or ten) polynomial fit, or Nelson-Siegel curve.

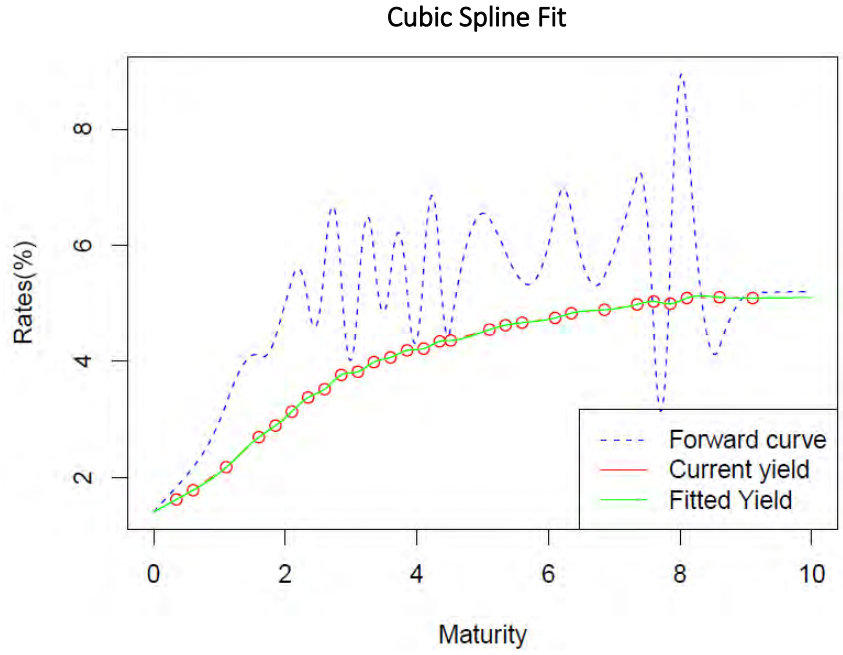
As it is difficult to cover all the possible option prices that can be used for calibration of time independent parameters in implementation, we concentrate on using cap prices. In the yield curve interpolation stage both one-factor and two-factor model use the following steps:

1. Obtain rates $r(0,t_i)$ using the discount factors $Z(r,t_i)$ from $r(0,t_i) = -\frac{1}{t_i}ln\big(Z(0,t_i)\big), i = 1,2,\dots,n$.
2. Fit a curve, $\hat{r}(0,t_i)$, for the points $\big(t_i, r(0,t_i)\big), i = 1,2,\dots,n$.
3. Use the fitted curve to get an estimate $\hat{f}(0,t)$ for $f(0,t)$ using $\hat{f}(0,t) = -\frac{\partial}{\partial t}[t\hat{r}(0,t)]$

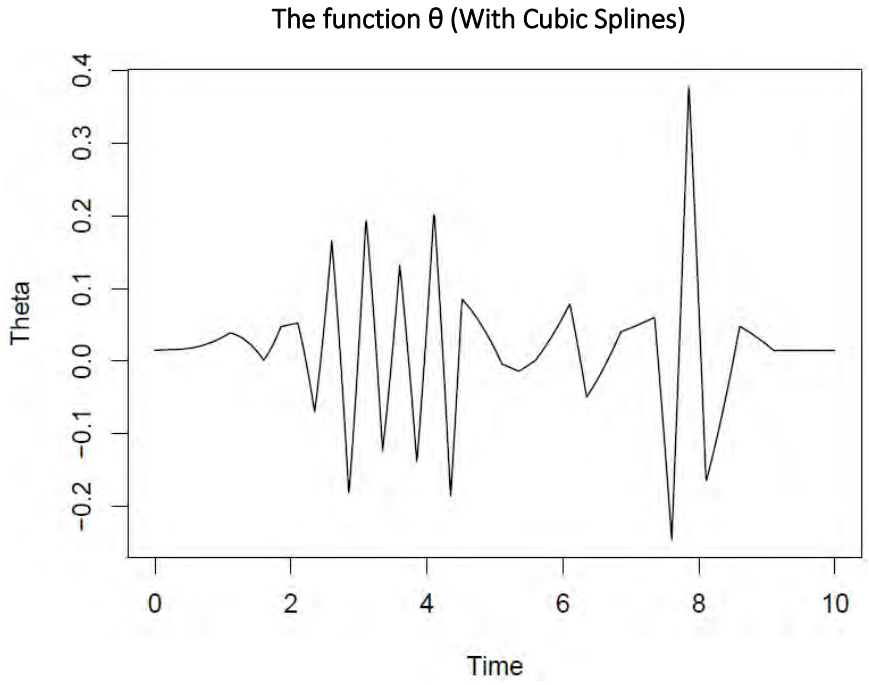
4.2 YIELD CURVE INTERPOLATION: ONE-FACTOR MODEL

In fitting a curve for $r(0,t)$ we found that cubic-spline interpolation works poorly as the resulting $f(0,t)$ was an oscillating function. [24] suggests using a polynomial of degree 6 or degree 10 for $r(0,t)$. As an alternative we can fit using a Nelson-Siegel curve.

```
#example 32
# Hull-White model
TM = VeronesiTable15p1$'Time to Maturity'
Yield = VeronesiTable15p1$'Yield %'*100
sigma=0.0196
gamma=0.19
xvals = seq(0,10,0.01)
rates= model1(xvals)
ratesdash = model1(xvals,1)
rates2dash = model1(xvals,2)
f0t = rates+ratesdash*xvals
thetat = 2*ratesdash/100+xvals*rates2dash/100+sigma^2*xvals+
 gamma*f0t/100+sigma^2/
 (2*gamma)*(1-exp(-2*gamma*xvals))
plot(xvals,f0t,type="l",lty=2,col="blue",main="Cubic Spline fit",
     xlab="Maturity",ylab="Rates(%)" )
points(TM,Yield,type="b",lty=1,col="red")
points(xvals,rates,type="l",lty=1,col="green")
legend("bottomright",legend=c("Forward curve","Current yield",
                              "Fitted Yield"),
       col=c("blue","red","green"),lty=c(2,1,1))
```



```
plot(xvals,thetat,type="l",
     main=expression(paste("The function ",
                           theta," (With Cubic Splines)")),
     xlab="Time",ylab="Theta")
```



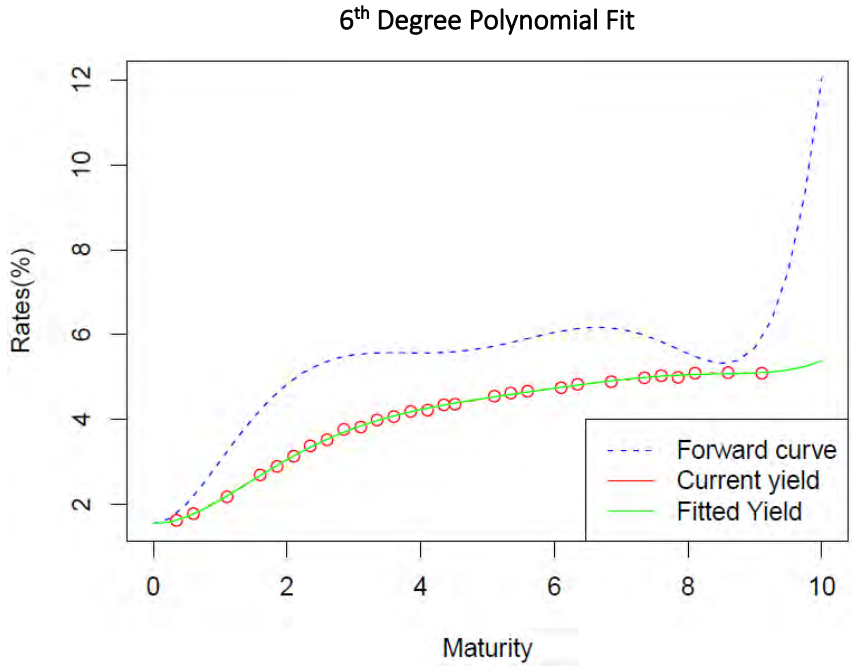
4.2.1 FITTING AN n DEGREE POLYNOMIAL FOR $r(0,t)$

R's "poly" function fits the polynomial model:

$$
\begin{aligned}
r(0,t) &= \sum_{i=0}^{n} a_i t^i \\
f(0,t) &= \sum_{i=0}^{n} a_i (i+1) t^i \\
\frac{\partial f(0,t)}{\partial t} &= \sum_{i=1}^{n} a_i i(i+1) t^{i-1} \\
\theta_t &= \sum_{i=1}^{n} a_i i(i+1) t^{i-1} + \sum_{i=0}^{n} \gamma^* a_i (i+1) t^i + \frac{\sigma^2}{2\gamma^*}\left(1 - e^{-2\gamma^* t}\right)
\end{aligned}
$$

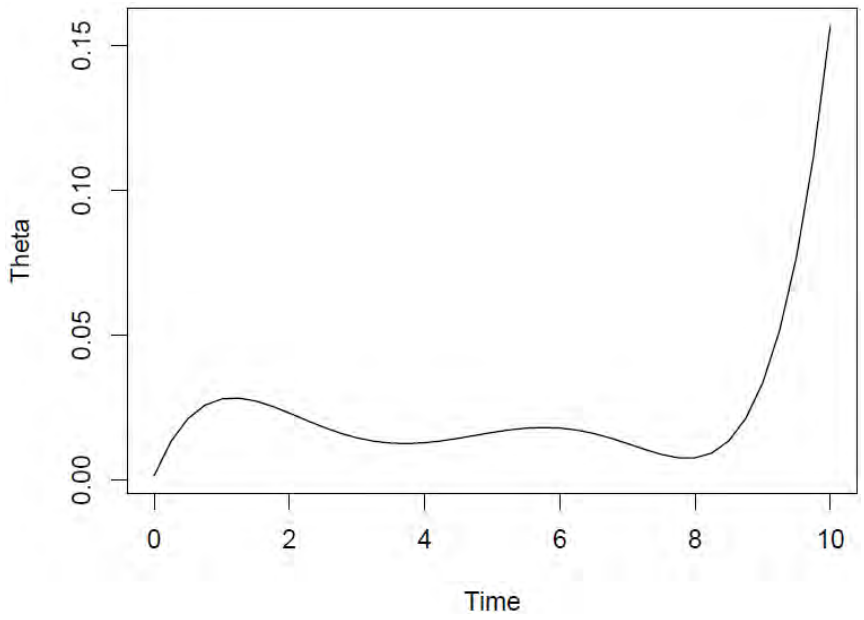
```
#example33
TM = VeronesiTable15p1$'Time to Maturity'
Yield = VeronesiTable15p1$'Yield %'*100
model1 = splinefun(TM,Yield,method="natural")
sigma=0.0196
gamma=0.19
model2 = lm(Yield~poly(TM,6,raw=TRUE))
coefs = as.numeric(coef(model2))
```

```
xvals = seq(0,10,0.25)
rates = sapply(xvals,polyfit,coeffs=coefs)
ratesdash = sapply(xvals,polyfit,coeffs=coefs,deriv=1)
rates2dash = sapply(xvals,polyfit,coeffs=coefs,deriv=2)
f0t = rates+ratesdash*xvals
thetat = 2*ratesdash/100+xvals*rates2dash/100+sigma^2*xvals+
   gamma*f0t/100+sigma^2/(2*gamma)*(1-exp(-2*gamma*xvals))
plot(xvals,f0t,type="l",lty=2,col="blue",
      main="6th degree polynomial fit",xlab="Maturity",ylab="Rates(%)" )
points(TM,Yield,type="b",lty=1,col="red")
points(xvals,rates,type="l",lty=1,col="green")
legend("bottomright",legend=c("Forward curve","Current yield",
                                        "Fitted Yield"),
          col=c("blue","red","green"),lty=c(2,1,1))
```




```
plot(xvals,thetat,type="l",
      main=expression(paste("The function ",
                            theta," (With 6 Degree Polynomial)")),
      xlab="Time",ylab="Theta")
```

The function θ (With 6th Degree Polynomial)



4.2.2 FITTING A NELSON-SIEGEL CURVE TO $r(0,t)$

The Nelson-Siegel model specifies $f(0,t)$ in the following form:

$$f(0,t) = \beta_0 + \beta_1 exp\left(-\frac{t}{\tau}\right) + \frac{\beta_2 t}{\tau} exp\left(-\frac{t}{\tau}\right)$$

However since we are fitting market yields we will get an expression for $r(0,t)$:

$$\begin{aligned} r(0,t) &= \frac{1}{t}\int_0^t f(0,s)ds \\ &= \frac{1}{t}\int_0^t \left(\beta_0 + \beta_1 \exp\left(-\frac{s}{\tau}\right) + \frac{\beta_2 s}{\tau}\exp\left(-\frac{s}{\tau}\right)\right) ds \\ &= \frac{1}{t}\left(\beta_0 t + \beta_1 \tau\left(1-\exp\left(-\frac{t}{\tau}\right)\right) + \beta_2\tau\left(1-\exp\left(-\frac{t}{\tau}\right)\right) - \beta_2 t\exp(-\frac{t}{\tau})\right) \\ &= \beta_0 + \beta_1\frac{1-\exp\left(-\frac{t}{\tau}\right)}{\frac{t}{\tau}} + \beta_2\frac{1-\exp\left(-\frac{t}{\tau}\right)-\frac{t}{\tau}\exp\left(-\frac{t}{\tau}\right)}{\frac{t}{\tau}} \end{aligned}$$

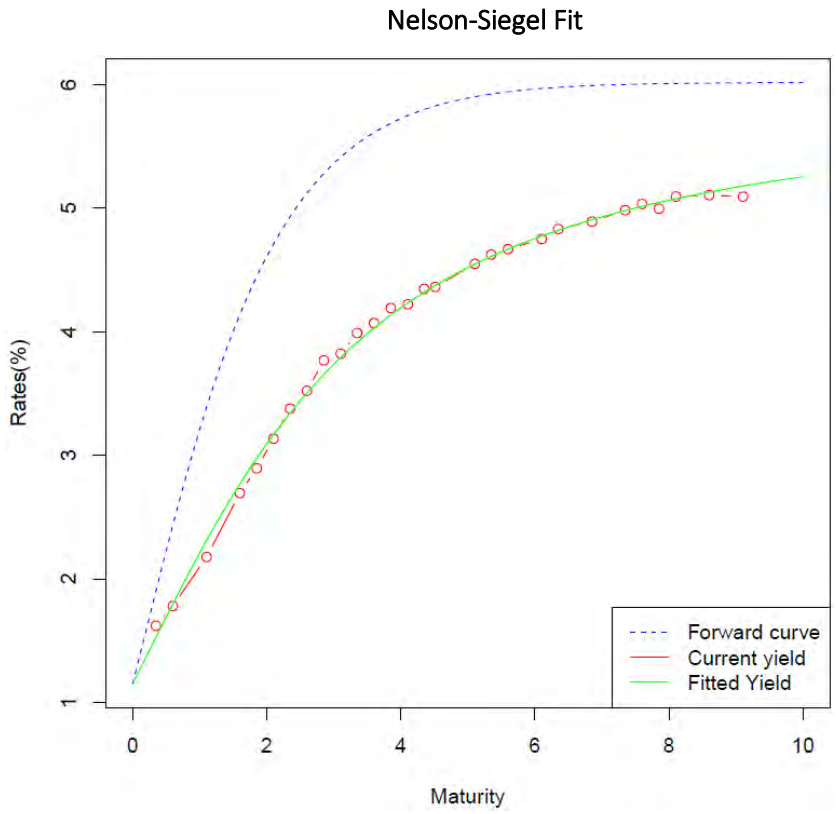
Also we need:

$$\frac{\partial}{\partial t}f(0,t) = -\frac{\beta_1}{\tau}exp\left(-\frac{t}{\tau}\right) + \frac{\beta_2}{\tau}exp\left(-\frac{t}{\tau}\right) - \frac{\beta_2 t}{\tau^2}exp\left(-\frac{t}{\tau}\right)$$

```
#example 34
# This example uses function "YieldCurve" package
# first it must be installed and loaded, before running this example.
#
TM = VeronesiTable15p1$'Time to Maturity'
Yield =-log(VeronesiTable15p1$Strips/100)/TM*100
xvals = seq(0,10,0.01)
```

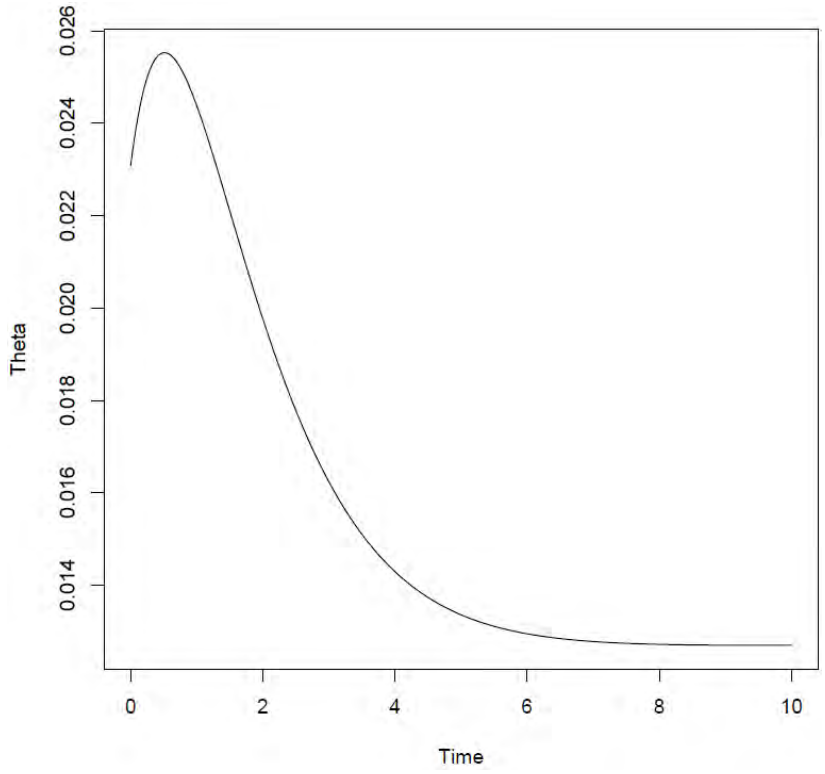
```
# The function "Nelson.Siegel" is from the Yield curve package.
NSP = Nelson.Siegel(Yield,TM)
# The following three functions are developed with this paper.
rates = NSRates(as.numeric(NSP),xvals)
f0t = NSForwards(as.numeric(NSP),xvals)
thetat = Theta(as.numeric(NSP),sigma=0.0221,gamma=0.19, xvals)

plot(xvals,f0t,type="l",lty=2,col="blue",main="Nelson-Siegel Fit",
     xlab="Maturity",ylab="Rates(%)" )
points(TM,Yield,type="b",lty=1,col="red")
points(xvals,rates,type="l",lty=1,col="green")
legend("bottomright",legend=c("Forward curve","Current yield",
                              "Fitted Yield"),col=c("blue","red","green"),lty=c(2,1,1))
```




```
plot(xvals,thetat,type="l",
     main=expression(paste("The function ",
                           theta," (with Nelson-Siegel)")),
     xlab="Time",ylab="Theta")
```

The Function θ (with Nelson-Siegel)



4.3 CALIBRATION OF THE ONE-FACTOR HULL-WHITE MODEL

For both one-factor and two-factor Hull-White model calibration we assume that we have a data set similar to Table 19.4:

Swap Rates and Cap Prices on November 3, 2008

Cap#	Maturity (T)	Swap Rate	Price (x100)	Discount Factors
1	0.25	0.028588	0	0.9929037
2	0.50	0.026486	0.0528	0.9868908
3	0.75	0.024929	0.1313	0.9815442
4	1.00	0.024320	0.2401	0.9760606
5	1.25	0.024491	0.3826	0.9699535
6	1.50	0.024938	0.5405	0.9633988
7	1.75	0.025561	0.7106	0.9563730
8	2.00	0.026260	0.8932	0.9489501
9	2.25	0.027252	1.1095	0.9406132

Cap#	Maturity (T)	Swap Rate	Price (x100)	Discount Factors
10	2.50	0.028630	1.3729	0.9309471
11	2.75	0.030108	1.6636	0.9204540
12	3.00	0.031400	1.9502	0.9098978
13	3.25	0.032471	2.2235	0.8995225
14	3.50	0.033474	2.4973	0.8889794
15	3.75	0.034408	2.7711	0.8783263
16	4.00	0.035270	3.0451	0.8676278
17	4.25	0.036076	3.3208	0.8568746
18	4.50	0.036835	3.5968	0.8460722
19	4.75	0.037531	3.8700	0.8353260
20	5.00	0.038150	4.1370	0.8247441

The above table contains one additional column, discount factors, which is derived from the Swap Rate column ([24]) of Table 19.4. The data set gives cap prices for 20 caps, each cap is quarterly spaced and the cap rate for each cap is the associated swap rate. For example, cap 12 is a 3-year cap with four payments per year and the cap rate is 3.14% with the first payment occurring at 0.5 years, if the observed rate is above the cap rate of 3.14%. The value of a caplet is given by the following formulas from [24]:

$$(19.33)\qquad S_Z(T_O;T_B)^2 = B(T_O;T_B)^2\frac{\sigma^2}{2\gamma^*}\left(1-e^{-2\gamma^* T_O}\right)$$

$$(19.41)\qquad A(t;T) = \log\left(\frac{Z(r_0,0;T)}{Z(r_0,0;t)}\right) + B(t;T)f(0,t) - \frac{\sigma^2}{4\gamma^*}B(t;T)^2\left(1-e^{-2\gamma^* t}\right)$$

$$(19.42)\qquad A(t;T) = \log\left(\frac{Z(r_0,0;T)}{Z(r_0,0;t)}\right) + (T-t)f(0,t) - \frac{\sigma^2}{2}(T-t)^2 t$$

$$\text{write}\quad K = \frac{1}{1+r_K\Delta}$$

$$(19.44)\qquad V(r_0,0) = M\times\left(KZ(r_0,0;T-\Delta)\mathcal{N}(-d_2) - Z(r_0,0;T)\mathcal{N}(-d_1)\right)$$

$$(19.45)\qquad d_1 = \frac{1}{S_Z(T-\Delta;T)}\log\left(\frac{Z(r_0,0;T)}{KZ(r_0,0;T-\Delta)}\right) + \frac{S_Z(T-\Delta;T)}{2}$$

$$(19.46)\qquad d_2 = d_1 - S_Z(T-\Delta;T)$$

while $B(t;T)$ remains as:

$$B(t;T) = \frac{1}{\gamma^*}\left(1-e^{-\gamma^*(T-t)}\right)$$

A careful observation reveals that this is the put option formula under the Vasicek model with modeled bond prices replaced by market observed bond prices. The cap prices are obtained by adding all the prices of corresponding caplets together. We first implement a caplet formula in the code block given below.

```
#Text (19.45)
Hull.White.caplet = function(rK= 0.03815,T=5,Delta=0.25,
                             ZT1=0.835326,ZT2=0.8247441,
                             gamma=0.054523,sigma=0.0149)
{
 BTOTB = Vasicek.B(gamma,Delta)
 SZ = (BTOTB^2*sigma^2*Vasicek.B((2*gamma),(T-Delta)))^0.5
 K = 1/(1+rK*Delta)
 d1 = 1/SZ*log(ZT2/(K*ZT1))+ SZ/2
 d2 = d1-SZ
 return(ZT1*pnorm(-d2)-ZT2/K*pnorm(-d1))
}
```

Note that the R function Vasicek.B is implemented as:

```
# Vasicek bond pricing formula B
Vasicek.B = function(gamma,T){
 if (gamma==0) {
  return(T)
 }
 else {
  return((1-exp(-gamma*T))/gamma)
 }
}
```

Once the caplet prices are calculated the cap price is calculated as the sum of these caplets as given below:

```
Hull.White.cap = function(rK=0.05,TK=5,Delta=0.25,discount_factors,
                          gamma,sigma)
{
 no.caplets = TK/Delta -1
 sum(Hull.White.caplet(rK=rK,T=seq(2*Delta,TK,Delta),Delta=Delta,
                       ZT1=discount_factors[1:no.caplets],
                       ZT2=discount_factors[2:(no.caplets+1)],gamma,
                       sigma))
}
```

Now to calculate a cap price vector as given in Table 19.4 of ([24]) we can use the following R function:

```
Hull.White.cap.price.vector = function(cap_rates,Maturity,Delta=0.25,
                                       discount_factors,gamma,sigma)
{
 n = length(Maturity)-1
 caps =rep(0,n+1)
 for (i in 2:(n+1)){
  caps[i] = Hull.White.cap(rK=cap_rates[i],TK=Maturity[i],
                           Delta=Delta,discount_factors =discount_factors,
                           gamma=gamma,sigma=sigma)
 }
 return(caps)
}
```

Now we can estimate γ^* and σ^* using the non-linear least squares function "nls" which can be downloaded from R's CRAN archive. To verify the nls function we first use a simulated data set assuming the prices are from a Hull-White model.

```
#example35
# Hull-White fit with simulated data example
Testdata2 = VeronesiTable19p4
Testdata2$'Cap Price (x100)' =
    Hull.White.cap.price.vector(
    Maturity=VeronesiTable19p4$Maturity,
    cap_rates=VeronesiTable19p4$'Swap Rate',
    discount_factors=VeronesiTable19p4$discount_factors,
    gamma=1,sigma=0.015)+rnorm(20,0,0.0001)

Hull.White.fit =
 nls('Cap Price (x100)'~
      Hull.White.cap.price.vector(Maturity=Testdata2$Maturity,
                cap_rates=Testdata2$'Swap Rate',
                discount_factors=Testdata2$discount_factors,
                gamma=gamma,sigma=sigma),
      data= Testdata2,start=list(gamma=.1,sigma=0.01),
      nls.control(maxiter = 100, tol = 1e-05, minFactor = 1/1024,
                printEval = FALSE, warnOnly = FALSE, scaleOffset = 0,
                nDcentral = FALSE))
sum_mod = summary(Hull.White.fit)
```

The "nls" algorithm converged in 4 iterations and summary statistics are:

The Hull-White Model Fit: Summary Statistics

Parameter	Estimate	Std. Error	t-value	$Pr[>\|t\|]$
γ^*	0.09707	0.00531	18.28544	4.49888×10^{-13}
σ	0.01496	9.61873×10^{-5}	155.50562	1.29281×10^{-29}

From the output it is clear that "nls" function estimates are very close to the actual values. Now we can do the analysis for the data in Table 19.4 of [24]. The code snippet for the analysis is:

```
#example36
Data_set =VeronesiTable19p4
Data_set$'Cap Price (x100)' = VeronesiTable19p4$'Cap Price (x100)'/100
Data_set$'Cap Price (x100)'[1]=0
Hull.White.fit =
nls(Data_set$'Cap Price (x100)'~ Hull.White.cap.price.vector(
                                        Maturity=Data_set$Maturity,
                                        cap_rates=Data_set$'Swap Rate',
                                        discount_factors=
                                        Data_set$discount_factors,
                                        gamma=gamma,
                                        sigma=sigma),

                        data=Data_set ,start=list(gamma=.1,sigma=0.1),
                        nls.control(maxiter = 100, tol = 1e-05,
                                    minFactor = 1/10240,
                                    printEval = FALSE, warnOnly = FALSE,
                                    scaleOffset = 0,nDcentral = FALSE))
sum_mod = summary(Hull.White.fit)
```

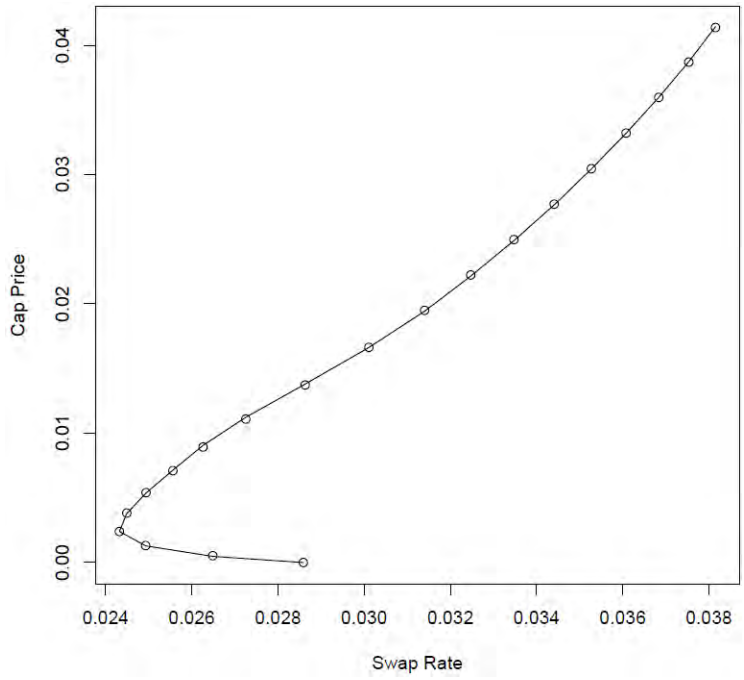
The "nls" algorithm converged in 5 iterations and summary statistics are:

The Hull-White Model Fit: Summary Statistics

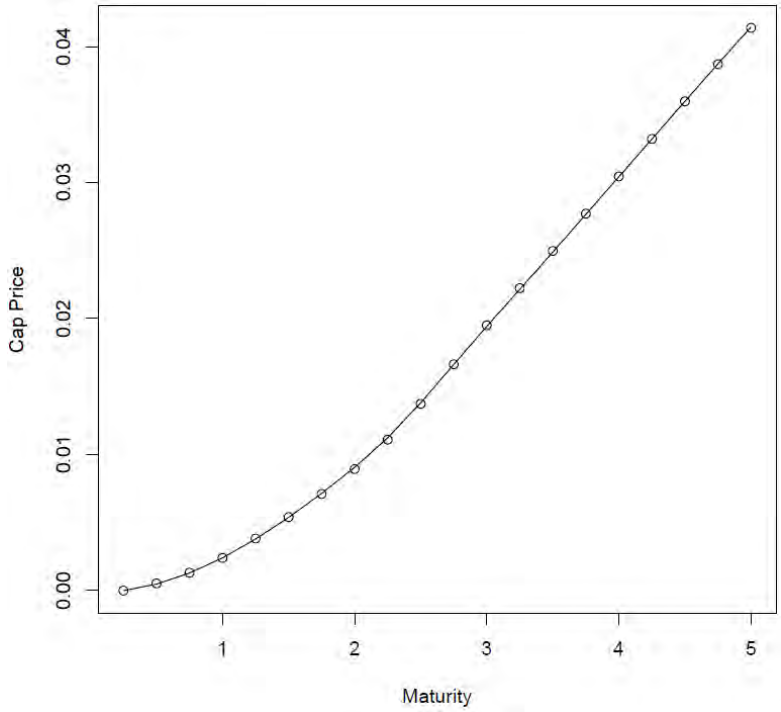
Parameter	Estimate	Std. Error	t-value	$Pr[>\|t\|]$
γ^*	0.06712	0.00376	17.84779	6.80885×10^{-13}
σ	0.01454	6.87588×10^{-5}	211.40969	5.1521×10^{-32}

From the summary statistics we see that the fit is very good. As further evidence of the appropriateness of the fit we consider some graphical analysis with code snippets:

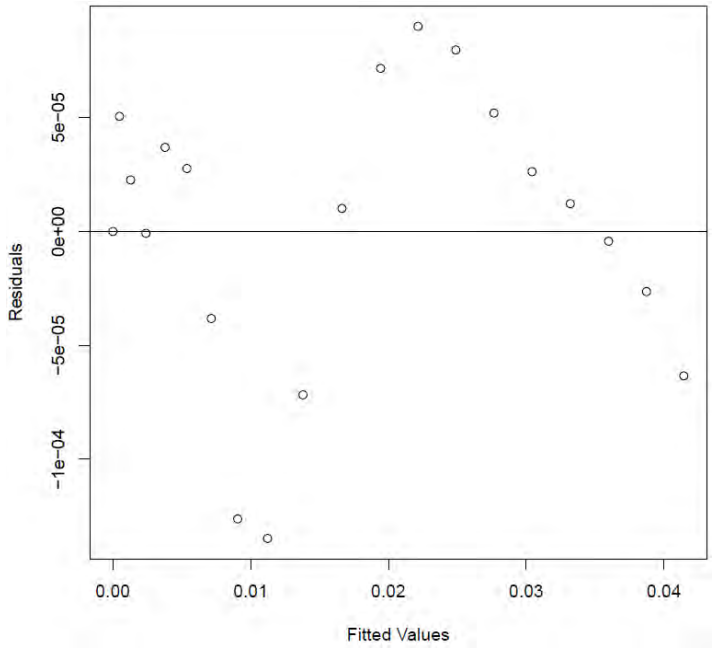
```
#example37
# Hull-White fit graphical illustration
plot('Cap Price (x100)'~ 'Swap Rate', data =Data_set,
     xlab = "Swap Rate", ylab = "Cap Price")
lines(Data_set$'Swap Rate', fitted(Hull.White.fit))
```




```
plot('Cap Price (x100)'~ 'Maturity', data =Data_set,
     xlab = "Maturity", ylab = "Cap Price")
lines(Data_set$'Maturity', fitted(Hull.White.fit))
```




```
plot(fitted(Hull.White.fit),residuals(Hull.White.fit),xlab="Fitted Values",
     ylab="Residuals")
abline(a=0,b=0)
```



As the graphs indicate the fitted model is an ideal one.

4.4 CALIBRATION OF THE TWO-FACTOR HULL-WHITE MODEL

The calibration of time-independent parameters will work as follows:

- First calculate the value of a caplet using the following formulas:

[23] (22.57)

$$\begin{aligned} S_Z(T_O;T_B)^2 &= B_1^2(TO;TB)\frac{\sigma_1^2}{2\gamma_1^*}(1-\exp(-2\gamma_1^* T_O) \\ &\quad + B_2^2(T_O;T_B)\frac{\sigma_2^2}{2\gamma_2^*}(1-\exp(-2\gamma_2^* T_O) \\ &\quad + B_1(T_O;T_B)B_2(T_O;T_B)\sigma_1\sigma_2\rho\frac{1-\exp(-(\gamma_1^*+\gamma_2^*)T_O)}{\gamma_1^*+\gamma_2^*} \\ K &= \frac{1}{1+r_K\Delta} \\ V(r_0,0) &= M\times(KZ(r_0,0;T-\Delta)\mathcal{N}(-d_2)-Z(r_0,0;T)\mathcal{N}(-d_1)) \\ d_1 &= \frac{1}{S_Z(T-\Delta;T)}\log\left(\frac{Z(r_0,0;T)}{KZ(r_0,0;T-\Delta)}\right)+\frac{S_Z(T-\Delta;T)}{2} \\ d_2 &= d_1 - S_Z(T-\Delta;T) \end{aligned}$$

In the above formula bond prices are market or interpolated bond prices. The above formulas are implemented in the following code chunk:

```
Two.factor.Hull.White.caplet = function(rK,T,Delta=0.25,ZTO,ZTB,
                                        gamma1,gamma2,sigma1,sigma2,rho){
 TB =T
 TO = T-Delta
 K = 1/(1+rK*Delta)
 B1TOTB = Vasicek.B(gamma=gamma1,T=(TB-TO))
 B2TOTB = Vasicek.B(gamma=gamma2,T=(TB-TO))
 SZTO2 = B1TOTB^2 * sigma1^2*Vasicek.B(gamma=2*gamma1,T=TO) +
          B2TOTB^2 * sigma2^2*Vasicek.B(gamma=2*gamma2,T=TO) +
          B1TOTB*B2TOTB* sigma1*sigma2*rho*
          Vasicek.B(gamma=(gamma1+gamma2),T=TO)
 SZTO = SZTO2^0.5
 d1 = 1/SZTO * log(ZTB/(K*ZTO))+SZTO/2
 d2 = d1 - SZTO
 V0 = -ZTB*pnorm(-d1)+K*ZTO*pnorm(-d2)
 return (V0)
}
```

- In the R functions for cap price, the cap price vector is similar to functions in the one-factor Vasicek model.

  The following example illustrates calibration using a simulated data set.

```
#example 38
# Hull-White fit with simulated data example
Testdata2 = VeronesiTable19p4
Testdata2$'Cap Price (x100)' =
 Two.factor.Hull.White.cap.price.vector(cap_rates=VeronesiTable19p4$'Swap Rate',
                                        Maturity=VeronesiTable19p4$Maturity,
                        discount_factors=VeronesiTable19p4$discount_factors,
                        gamma1=0.1,gamma2=-0.2,sigma1=0.2,sigma2=0.3,rho=0.5)

Testdata2$'Cap Price (x100)'[1]=0

Two.factor.Hull.White.fit =nls('Cap Price (x100)'~
          Two.factor.Hull.White.cap.price.vector(cap_rates=Testdata2$'Swap Rate',
          Maturity=Testdata2$Maturity,
          discount_factors=Testdata2$discount_factors,
          gamma1=gamma1,gamma2=gamma2,sigma1=sigma1,sigma2=sigma2,rho=rho),
          data= Testdata2,
          start=list(gamma1=.3,gamma2=-0.3,sigma1=0.1,sigma2=0.4,rho=0.6),
          nls.control(maxiter = 100, tol = 1e-05, minFactor = 1/1024,
          printEval = FALSE, warnOnly = FALSE, scaleOffset = 0,
          nDcentral = FALSE))
```

```
## Error in nls(`Cap Price (x100)` ~ Two.factor.Hull.White.cap.price.vector(cap rates
= Testdata2$`Swap Rate`, : step factor 0.000488281 reduced below 'minFactor'
of 0.000976562

summary(Two.factor.Hull.White.fit)

## Error in summary(Two.factor.Hull.White.fit): object 'Two.factor.Hull.White.fit'
not found
```

After trying out several initial values we failed to find results. If we look at the cap price vector formula closely we see that each element is a function similar to $\sum_i a_i \, \Phi\left(x_i, \underline{\mu_i}, \underline{\sigma_i}\right)$ where $a_i, i = 1,2, \ldots$ are known constants and $\Phi\left(x_i, \underline{\mu_i}, \underline{\sigma_i}\right)$, cdf of a normal distribution, is as given below:

$$\Phi\left(x_i, \underline{\mu_i}, \underline{\sigma_i}\right) = \frac{1}{2\pi\underline{\sigma_i}} \int_{-\infty}^{x_i} exp\left(-\frac{\left(t - \underline{\mu_i}\right)}{\underline{\sigma_i}^2}\right) dt,$$

In the above expression, $\underline{\mu_i}$ and $\underline{\sigma_i}$ are functions of the five parameters $\gamma_1, \gamma_2, \sigma_1, \sigma_2$, and ρ. In the non-linear least squares minimization, we minimize the distance between $\sum_i a_i \, \Phi\left(x_i, \underline{\mu_i}, \underline{\sigma_i}\right)$ and its observed values. It may be very sensitive to initial guesses and it may also have local values that minimize the objective function. So, we need an alternative method to compute parameters.

When we look at (22.57) of [24] carefully we see that caplet with a pay-off at time T is a function of $S_Z(T - \Delta; T)$, where it is given by:

$$\begin{aligned}
S_Z(T-\Delta;T)^2 &= B_1(T-\Delta;T)^2 \frac{\sigma_1^2}{2\gamma_1^*}(1-\exp(-2\gamma_1^*(T-\Delta)) \\
&\quad + B_2(T-\Delta;T)^2 \frac{\sigma_2^2}{2\gamma_2^*}(1-\exp(-2\gamma_2^*(T-\Delta)) \\
&\quad + B_1(T-\Delta;T)B_2(T-\Delta;T)\sigma_1\sigma_2\rho\frac{1-\exp(-(\gamma_1^*+\gamma_2^*)(T-\Delta))}{\gamma_1^*+\gamma_2^*} \\
&= B_1(0;\Delta)^2 \frac{\sigma_1^2}{2\gamma_1^*}(1-\exp(-2\gamma_1^*(T-\Delta) \\
&\quad + B_2(0;\Delta)^2 \frac{\sigma_2^2}{2\gamma_2^*}(1-\exp(-2\gamma_2^*(T-\Delta) \\
&\quad + B_1(0;\Delta)B_2(0;\Delta)\sigma_1\sigma_2\rho\frac{1-\exp(-(\gamma_1^*+\gamma_2^*)(T-\Delta))}{\gamma_1^*+\gamma_2^*}
\end{aligned}$$

The above formula can be coded as:

```
Two.factor.Vasicek.SZ = function(T0,TB,gamma1,gamma2,sigma1,sigma2,rho){
  B1T0TB = Vasicek.B(gamma=gamma1,T=(TB-T0))
  B2T0TB = Vasicek.B(gamma=gamma2,T=(TB-T0))
  SZT02 = B1T0TB^2 * sigma1^2*Vasicek.B(gamma=2*gamma1,T=T0) +
    B2T0TB^2 * sigma2^2*Vasicek.B(gamma=2*gamma2,T=T0) +
    B1T0TB*B2T0TB* sigma1*sigma2*rho*Vasicek.B(gamma=(gamma1+gamma2),T=T0)
  SZT0 = SZT02^0.5
  return(SZT0)
}
```

We see that $S_z(T-\Delta;T)$ characterizes the particular caplet, independent of which caplet it belongs to. We can exploit this fact first to obtain $S_Z((i-1)\Delta;i\Delta)$ for $i = 1,2,\ldots$ and then applying the nonlinear least squares method to estimate parameters$\gamma_1^*, \gamma_2^*, \sigma_1$ and σ2. Extracting $S_Z\big((i-1)\Delta;i\Delta\big)$ i = 1,2,... is similar to extracting forward volatilities from flat volatilities as given in section 20.1.2 of [24]. It works as follows:

- Use the caplet with payment at time 0.5 to obtain $S_z(0;0.5)$ This is similar to computing Black implied volatility.
- Calculate the price of a caplet with a payout at time 0.5 with cap rate applicable to a cap maturing at time 0.75. Subtract this caplet value from the cap maturing at time 0.75 to obtain caplet with a payoff at time 0.75. Use this to compute $S_Z(0.5;0.75)$.
- Proceed similarly to calculate caplet prices and then compute $S_Z(0.25(i-1),0.25i), i = 4,5,\ldots$

Note that the above procedure of extracting $S_Z((i-1)\Delta;i\Delta)$ for $i = 1,2,\ldots$values are identical for both one factor and two factor Hull-White model, except for the one factor model $S_Z(T-\Delta,;T)$ given as:

$$S_Z(T-\Delta;T)^2 \quad = \quad B(0;\Delta)^2\frac{\sigma^2}{2\gamma^*}(1-\exp(-2\gamma^*(T-\Delta))$$

The above formula can be coded as:

```
Vasicek.SZ= function(TO=0.5,TB=1,gamma=0.4653,sigma=0.0221){
  SZ = sigma*Vasicek.B(gamma=gamma,T=(TB-TO))*
       (Vasicek.B(gamma=(2*gamma),T=TO))^0.5
  return(SZ)
}
```

We can implement the above algorithm as given below:

```
HW.forward.vol = function(Maturity,discount_factors, cap_rates,cap_prices)f
   Nsize = length(Maturity)
   sigma.forward = rep(0,Nsize)
   sigma.forward[2]=uniroot(HW.caplet.implied.vol,c(0,1),
                            price=cap_prices[2],
                            rK= cap_rates[2],
                            T=Maturity[2],
                            ZTO=discount_factors[1],
                            ZTB=discount_factors[2])$root
   for ( i in (3:Nsize)){
      price.caplet = cap_prices[i]-
         Hull.White.capv2(rK= cap_rates[i],
                          TK= Maturity[i-1],
                          discount_factors = discount_factors[1:i],
                          sigmaf = sigma.forward[2:(i-1)])
```

```
   sigma.forward[i] =
      uniroot(HW.caplet.implied.vol,c(0,1),
                price=price.caplet,
                rK= cap_rates[i],
                T=Maturity[i],
                ZTO=discount_factors[i-1],
                ZTB=discount_factors[i])$root
   }
   return(sigma.forward)
}
```

The auxiliary functions needed for the above functions are given below:

```
Hull.White.capletv2 = function(rK,T,Delta=0.25,ZTO,ZTB,SZTO)f
   TB =T
   TO = T-Delta
   K = 1/(1+rK*Delta)
   d1 = 1/SZTO * log(ZTB/(K*ZTO))+SZTO/2
   d2 = d1 - SZTO
   V0 = ZTO*pnorm(-d2)-ZTB/K*pnorm(-d1)
   return (V0)
}

Hull.White.capv2 = function(rK=0.05,TK=5,Delta=0.25,discount_factors,sigmaf){
   no.caplets = TK/Delta -1
   sum(Hull.White.capletv2(rK=rK,T=seq(2*Delta,TK,Delta),Delta=Delta,
   ZTO=discount_factors[1:no.caplets],
   ZTB=discount_factors[2:(no.caplets+1)],SZTO=sigmaf))
}

Hull.White.cap.price.vectorv2 = function(cap_rates,Maturity,Delta=0.25,
                                         discount_factors,sigma){
   n = length(Maturity)-1
   caps =rep(0,n+1)
   for (i in 2:(n+1)){
      caps[i] = Hull.White.capv2(rK=cap_rates[i],TK=Maturity[i],
                                 Delta=Delta,
                                 discount_factors =discount_factors,
                                 sigmaf=sigma[i])
   }
   return(caps)
}
```

```
HW.caplet.implied.vol = function(sigmaf, price=0.0786/100,
                                 rK= 0.02442,T=0.75,Delta=0.25,
                                 ZTO=0.988510,ZTB=0.981899){
  TB =T
  TO = T-Delta
  K = 1/(1+rK*Delta)
  d1 = 1/sigmaf * log(ZTB/(K*ZTO))+sigmaf/2
  d2 = d1 - sigmaf
  V0 = ZTO*pnorm(-d2)-ZTB/K*pnorm(-d1)-price
  return(V0)
}
```

Even though the Hull-White model performance with cap prices and "nls" was excellent, we can try it with the method proposed in this section. The following code block illustrates this method with a simulated data set.

```
#example39
# simulate a data set as in example 36
# Hull-White fit with simulated data example
rm(list=ls()) # clear the workspace and functions
Testdata =subset(VeronesiTable19p4,select=-c('Cap Price (x100)'))
Testdata$'Cap Prices' =
  Hull.White.cap.price.vector(
  Maturity=Testdata$Maturity,
  cap_rates=Testdata$'Swap Rate',
  discount_factors=Testdata$discount_factors,
  gamma=.1,sigma=0.015)+rnorm(length(Testdata$Maturity),0,0.00001)
Testdata$'Cap Prices'[1]=0
fvol.dat = HW.forward.vol(Maturity=Testdata$Maturity,
                          discount_factors =Testdata$discount_factors,
                          cap_rates=Testdata$'Swap Rate',
                          cap_prices =Testdata$'Cap Prices')
Hull.White.fit =nls('Forward Vol.'~ Vasicek.SZ(
                                TB=Maturity,
                                TO = Maturity-0.25,
                                gamma=gamma,sigma=sigma),
                    data= fvol.dat,
                    start=list(gamma=.3,sigma=0.1),
                    algorithm = "port",
                    upper=list(gamma=5,sigma=1),
                    lower=list(gamma=-5,sigma=0.001),
                    nls.control(maxiter = 1000000,
                                tol = 1e-05,
                                minFactor = 1/10240,
                                printEval = FALSE,
                                warnOnly = FALSE,
                                scaleOffset = 0,
                                nDcentral = FALSE))
sum_mod = summary(Hull.White.fit)
```

We see that in this case also the "nls" algorithm converged in 6 iterations. Summary statistics are:

The Hull-White Model Fit: Summary Statistics

Parameter	Estimate	Std. Error	t-value	$Pr[>\|t\|]$
γ^*	0.10091	0.0049	20.61337	1.82562×10^{-13}
σ	0.01502	1.15879×10^{-4}	129.59386	6.63247×10^{-27}

Now we can attempt the same analysis with the data set in Table 19.4 of [24]:

```
#example40
rm(list=ls()) # clear the workspace and functions
fvol.dat = HW.forward.vol(Maturity=VeronesiTable19p4$Maturity,
                          discount_factors =VeronesiTable19p4$discount_factors,
                          cap_rates=VeronesiTable19p4$'Swap Rate',
                          cap_prices =VeronesiTable19p4$'Cap Price (x100)'/100 )
Hull.White.fit =nls('Forward Vol.'~ Vasicek.SZ(TB=Maturity,
                                               TO = Maturity-0.25,
                                               gamma=gamma,
                                               sigma=sigma),
                    data= fvol.dat,
                    start=list(gamma=.3,sigma=0.1),
                    algorithm = "port",
                    upper=list(gamma=5,sigma=1),
                    lower=list(gamma=-5,sigma=0.001),
                    nls.control(maxiter = 1000000,
                                tol = 1e-05,
                                minFactor = 1/10240,
                                printEval = FALSE,
                                warnOnly = FALSE,
                                scaleOffset = 0,
                                nDcentral = FALSE))
sum_mod = summary(Hull.White.fit)
```

We see that in this case the "nls" algorithm converged in 7 iterations and summary statistics are:

The Hull-White Model Fit: Summary Statistics

Parameter	Estimate	Std. Error	t-value	$Pr[>\|t\|]$
γ^*	0.07803	0.01253	6.22849	9.17305×10^{-6}
σ	0.01474	3.01536×10^{-4}	48.87517	9.9965×10^{-20}

From the summary statistics we see that the estimate of γ^* is a little different but the estimate of σ is quite close to that in example 36.

Since we were successful with the new method in the Hull-White case, we try the calibration of the two-factor Hull-White model with the new method:

```
# example41
# Two factor Hull-White fit with simulated data example
rm(list=ls()) # clear the workspace and functions
Testdata =subset(VeronesiTable19p4,select=-c('Cap Price (x100)'))
Testdata$'Cap Prices' =
   Two.factor.Hull.White.cap.price.vector(cap_rates=Testdata$'Swap Rate',
                                          Maturity=Testdata$Maturity,
                                discount_factors=Testdata$discount_factors,
                                gamma1=0.1,gamma2=-0.2,sigma1=0.02,sigma2=0.03,
                                rho=-0.4)
Testdata$'Cap Prices'[1]=0

fvol.dat = HW.forward.vol(Maturity=Testdata$Maturity,
                                discount_factors =Testdata$discount_factors,
                                cap_rates=Testdata$'Swap Rate',
                                cap_prices =Testdata$'Cap Prices')+
            rnorm(length(Testdata$Maturity),0,0.001)
sim.SZ = Two.factor.Vasicek.SZ(TB=fvol.dat$Maturity,
                                TO = fvol.dat$Maturity-0.25,
                                gamma1=0.1,gamma2=-0.2,
                                sigma1=0.02,sigma2=0.03,
                                rho=-0.4)
fvol.dat$sim.SZ = sim.SZ
Two.factor.Hull.White.fit =nls('Forward Vol.'~
                                Two.factor.Vasicek.SZ(TB=Maturity,
                                                      TO = Maturity-0.25,
                                                      gamma1=gamma1,
                                                      gamma2=gamma2,
                                                      sigma1=sigma1,
                                                      sigma2=sigma2,
                                                      rho=rho),
                                data= fvol.dat,
                                start=list(gamma1=.3,gamma2=-0.3,sigma1=0.1,
                                           sigma2=0.4,rho=0.6),
                                algorithm = "port",
                                upper=list(gamma1=1,gamma2=1,sigma1=1,
                                           sigma2=1,rho=0.9),
                                lower=list(gamma1=-1,gamma2=-1,sigma1=0.001,
                                           sigma2=0.001,rho=-0.9),
                                nls.control(maxiter = 1000,
                                            tol = 1e-05,
                                            minFactor = 1/10240,
                                            printEval = FALSE,
                                            warnOnly = FALSE,
                                            scaleOffset = 0,
                                            nDcentral = FALSE))
```

```
## Error in nls(`Forward Vol.` ~ Two.factor.Vasicek.SZ(TB = Maturity, TO = Maturity
- : Convergence failure: singular convergence (7)

sum_mod = summary(Two.factor.Hull.White.fit)

## Error in eval(expr, envir, enclos): object 'Two.factor.Hull.White.fit' not found
```

With many different initial values for the parameters, this method did not converge to a value close to the true value. Therefore we decided to use a nonlinear least squares estimation package "nlrs" which has been developed recently. In the package "nlrs" the function "nlfb" performs very well. However, the input for "nlfb" is different from that of "nls". We need to input the residual function and the gradient of the residual function.

The following code chunk implements the residual function to be used with "nflb":

```
Two.factor.Vasicek.SZ.res = function(paras,forward_vol.dat){
   gamma1=paras[1]
   gamma2=paras[2]
   sigma1 = paras[3]
   sigma2 = paras[4]
   rho = paras[5]
   Delta = 0.25
   TO = forward_vol.dat$Maturity-0.25
   B1 = Vasicek.B(gamma=gamma1,T=Delta)
   B2 = Vasicek.B(gamma=gamma2,T=Delta)
   B1G1T = Vasicek.B(gamma=(2*gamma1),T=TO)
   B2G2T = Vasicek.B(gamma=(2*gamma2),T=TO)
   BG1G2T = Vasicek.B(gamma=(gamma1+gamma2),T=TO)
   Res = ((B1*sigma1)^2*B1G1T+(B2*sigma2)^2*B2G2T+
            B1*B2*BG1G2T*sigma1*sigma2*rho)/TO-
   forward_vol.dat$'Forward Vol.sq'
   return(Res)
}
```

We also developed a function that calculates the Jacobian of the residual function with respect to parameters that need to be estimated; it is given in the following code chunk:

```
Two.factor.Vasicek.SZ.gradient = function(paras,forward_vol.dat){
  gamma1=paras[1]
  gamma2=paras[2]
  sigma1 = paras[3]
  sigma2 = paras[4]
  rho = paras[5]
  Delta = 0.25
  TO = forward_vol.dat$Maturity-0.25
  Nsize = length(fvol.dat$Maturity)
  B1 = Vasicek.B(gamma=gamma1,T=Delta)
  B2 = Vasicek.B(gamma=gamma2,T=Delta)
  B1G1T = Vasicek.B(gamma=(2*gamma1),T=TO)
  B2G2T = Vasicek.B(gamma=(2*gamma2),T=TO)
  BG1G2T = Vasicek.B(gamma=(gamma1+gamma2),T=TO)
  B1D = Vasicek.DB(gamma=gamma1,T=Delta)
  B2D = Vasicek.DB(gamma=gamma2,T=Delta)
  B1G1TD = Vasicek.DB(gamma=(2*gamma1),T=TO)*2
  B2G2TD = Vasicek.DB(gamma=(2*gamma2),T=TO)*2
  BG1G2TD = Vasicek.DB(gamma=(gamma1+gamma2),T=TO)
  Jacob = matrix(0.0,Nsize,5)
  Jacob[1:Nsize,1] = (2*B1*B1D*B1G1T + B1^2*B1G1TD)*sigma1^2+
    (B1D*B2*BG1G2T+B1*B2*BG1G2TD)*sigma1*sigma2*rho
  Jacob[1:Nsize,2] = (2*B2*B2D*B2G2T + B2^2*B2G2TD)*sigma2^2+
    (B2D*B1*BG1G2T+B1*B2*BG1G2TD)*sigma1*sigma2*rho
  Jacob[1:Nsize,3] = B1^2*B1G1T*2*sigma1+ B1*B2*BG1G2T*sigma2*rho
  Jacob[1:Nsize,4] = B2^2*B2G2T*2*sigma2+ B1*B2*BG1G2T*sigma1*rho
  Jacob[1:Nsize,5] = B1*B2*BG1G2T*sigma1*sigma2
  Jacob = Jacob/TO
  attr(Jacob,"gradient") =Jacob
  return(Jacob)
}
```

The function "Two.factor.Vasicek.SZ.gradient" calls another function, "Vasicek.DB", which calculates the derivative of $B(t;T)$ with respect to γ and it is:

```
Vasicek.DB = function(gamma,T){
  if (abs(gamma)<1e-8) {
    return(-(T^2)/2)
  }
  else {
    return(-(1-exp(-gamma*T))/gamma^2+T*exp(-gamma*T)/gamma)
  }
}
```

We illustrate the use of these functions in the following code chunk:

```
#example42
rm(list=ls()) # clear the workspace and functions
# first calculate cap price vector similar to Veronesi Table 19.4
Testdata =subset(VeronesiTable19p4,select=-c('Cap Price (x100)'))
Testdata$'Cap Prices' =
   Two.factor.Hull.White.cap.price.vector(cap_rates=Testdata$'Swap Rate',
                                          Maturity=Testdata$Maturity,
                                          discount_factors=Testdata$discount_factors,
                                          gamma1=0.1,gamma2=-0.2,sigma1=0.2,sigma2=0.3,rho=-0.2)

Testdata$'Cap Prices'[1]=0
# Now calculate forward volatility from the cap prices.
fvol.dat = HW.forward.vol(Maturity=Testdata$Maturity,
                          discount_factors =Testdata$discount_factors,
                          cap_rates=Testdata$'Swap Rate',
                          cap_prices =Testdata$'Cap Prices')

fvol.dat$'Forward Vol.sq' = fvol.dat$'Forward Vol.'^2/(fvol.dat$Maturity-0.25)

st = c(gamma1=0.4,gamma2=0.4,sigma1=0.4,sigma2=0.5,rho=-0.1)
paras = st
Two.factor.Hull.White.fit=nlfb(start=st, resfn=Two.factor.Vasicek.SZ.res,
                jacfn=Two.factor.Vasicek.SZ.gradient,data=fvol.dat,
                trace=FALSE, control=list(prtlvl=1),forward_vol.dat=fvol.dat)
sum_mod = summary(Two.factor.Hull.White.fit)
```

We see that in this case the "nlfb" algorithm converged in 22 iterations and summary statistics are:

The Hull-White Model Fit: Summary Statistics

Parameter	Exact value	Estimate	Std. Error	t-value	$Pr[>\|t\|]$
γ_1^*	0.1	0.10762	0.00221	48.72325	4.99986×10^{-17}
γ_2^*	-0.2	-0.20095	2.64635×10^{-4}	-759.33669	1.04201×10^{-33}
σ_1^*	0.2	0.1934	0.00187	103.5146	1.34983×10^{-21}
σ_2^*	0.3	0.2974	7.092×10^{-4}	419.34879	4.24367×10^{-30}
ρ	-0.2	-0.13636	0.01829	-7.45701	3.07326×10^{-6}

We tried many different starting values and the values presented here were the best. We see that estimates of $\gamma_1^*, \gamma_2^*, \sigma_1$ and σ_2 are very close to the actual values and the estimate of ρ is a little away from the actual value. To evaluate the performance of the function "nlfb" for our situation, we need to carry out an extensive Monte Carlo simulation study which is beyond the scope of this paper. We would like to conclude this section by stating that non-linear least square minimization to estimate the five parameters in the two-factor Hull White model is a challenging problem even with most recently developed R functions.

5 Model Validation

At a high level, model validation is about the process around actual modeling work. It specifies the business purposes for using the model and assesses and confirms whether:

- The model is “fit for purpose”
- The methods used are accepted practice and compliant with standards and regulation.

In the U.S. applicable guidance is in Actuarial Standard of Practice 56, Modeling, and American Academy of Actuaries Model Governance Practice Note, April 2017, pages 11-14.

Canadian practice is evolving from CALM to IFRS. CALM guidance is in “Calibration of Stochastic Risk-Free Interest Rate Models for Use in CALM Valuation”, from the Canadian Institute of Actuaries' Committee for Life Insurance Financial Reporting (CIA, CLIFR), June 2021.

ASOP 56 Section 3.6 covers model risk, and outlines the process of model validation as a means of dealing with model risk. Section 3.6 breaks model validation into model testing (3.6.1) and model output validation (3.6.2).

The AAA practice note lays out review and model testing procedures. It begins with Design Use/Fit, about the business uses for the model and whether it is appropriate to use the particular model in those situations – whether it is “fit for purpose”. Next is Design Methods/Processing – whether the methods used are accepted practice and compliant with standards and regulation. Explicitly stating the modeling work's purposes – stress testing, or setting prices as of a specific date, or showing realistic balance sheets and income statements into the future – will clarify certain design decisions. These issues were discussed in Sections 1.4 and 1.5.

5.1 DATA AND ASSUMPTIONS

Whether input data is accurate, consistent, complete, and correctly loaded is a simple matter for the initial yield curve. In contrast, the mean reversion strength and target are assumptions that must be developed. There isn't one right answer; we will have to experiment, use judgement, and document our reasoning.

In Section 2.1.1 the Vasicek model was run. Opening the R function we see time and time step parameters (daily), plus the starting value, mean reversion speed, long term mean reversion target, and instantaneous volatility.

```
# Default values of r0, gamma, rbar, and alpha are the ones given
# in Veronesi (2010) Table 15.3 real-world parameters.
#
Vasicek.Euler.sim= function(t0=0,T=10,Delta=1/252,
                            r0=0.03,gamma=0.3262,rbar=0.0509,sigma=0.0221,M=10)
{}
```

Above these were given, but in practice they must be estimated. This is done in section 3.1.1, where Example 6 works [24] Chapter 14 Question 5:

```
# Example6
# Exercises Q5 Chapter 14 of Veronesi
# clear the workspace and functions
rm(list=ls())
summary(VeronesiTable14p7q5)
rt = VeronesiTable14p7q5$rt
Delta = 1/252
N = length(rt)
y = rt[2:N]
x = rt[1:N-1]

model = lm(y~x)
mle.gamma= -log(as.numeric(model$coefficients)[2])/Delta
mle.rbar = as.numeric(model$coefficients)[1]/
          (1-as.numeric(model$coefficients)[2])
mle.sigma = sigma(model)*(2*mle.gamma/
          (1-as.numeric(model$coefficients)[2]^2))^.5
```

Notice how the gamma, rbar, and sigma code directly implement the formulas immediately preceding Example 6 in 3.1.1 above.

```
rm(list=ls())
summary(VeronesiTable14p7q5)

##              DATE                          TCMNOMM1          c.c.
##   Min.      :2008-01-02 00:00:00.00   Min.   :0.0000   Min.   :0.0000
##   1st Qu.   :2008-07-16 18:00:00.00   1st Qu.  :0.0575   1st Qu.  :0.0575
##   Median  :2009-02-02 12:00:00.00   Median  :0.1400   Median  :0.1400
##   Mean    :2009-01-31 11:36:10.58   Mean    :0.6446   Mean    :0.6418
##   3rd Qu.  :2009-08-17 06:00:00.00   3rd Qu.  :1.4725   3rd Qu. :1.4671
##   Max.      :2010-03-05 00:00:00.00   Max.    :3.3700   Max.    :3.3419
##             rt
##   Min.     :0.0000
##   1st Qu.  :0.0575
##   Median :0.1400
##   Mean   :0.6466
##   3rd Qu. :1.4770
##   Max.     :3.3419

rt = VeronesiTable14p7q5$rt
Delta = 1/252
N = length(rt)
y = rt[2:N]
x = rt[1:N-1]
```

```
model = lm(y~x)
mle.gamma= -log(as.numeric(model$coefficients)[2])/Delta
mle.gamma

## [1] 3.6453

mle.rbar = as.numeric(model$coefficients)[1]/
                        (1-as.numeric(model$coefficients)[2])
mle.rbar

## [1] 0.30769

mle.sigma = sigma(model)*(2*mle.gamma/
                    (1-as.numeric(model$coefficients)[2]^2))^.5
mle.sigma

## [1] 1.8138
```

The practice notes advise us that the values of these parameters need to be accurate, consistent, in line with accepted practice and compliant with standards and regulation.

The short rate r0 will most likely be taken from the model start date's actual yield curve, with the same tenor (time to maturity) as the short rate used in calibration. The data used here is the one-month U.S. Treasury Bill rate.

The other parameters – short rate volatility (sigma), mean reversion strength (gamma), and long term mean reversion target (rbar) – are typically estimated from historical data. Rbar is estimated at 0.31% (rounded), which is low because the data is from the global financial crisis (GFC) period.

This model can be tested by using these values of gamma, rbar, and sigma to generate scenarios:

```
#example43
rm(list=ls())
t0=0
T=10
r0=0.03
gamma=3.645321
rbar = 0.3076885
sigma=1.813762
paras=c(gamma,rbar,sigma)
# Initaliaaze # of paths number of points in each path
Delta = 1/252
N = ceiling((T-t0)/Delta)
set.seed(123)
X = Vasicek.Trans.sim(t0,T,Delta,r0,gamma,rbar,sigma,M=1000)
M = ncol(X)
mle = matrix(0,3,M)
```

```
for (i in 1:M){
  y = X[2:(N+1),i]
  x= X[1:N,i]
  model = lm(y~x)
  mle.gamma = -log(as.numeric(model$coefficients)[2])/Delta
  mle.gamma= -log(as.numeric(model$coefficients)[2])/Delta
  mle.rbar = as.numeric(model$coefficients)[1]/
                              (1-as.numeric(model$coefficients)[2])
  mle.sigma = sigma(model)*(2*mle.gamma/
                          (1-as.numeric(model$coefficients)[2]^2))^.5
  mle[,i]=c(mle.gamma,mle.rbar,mle.sigma)
}

# Simulation performance criteria
bias =rowMeans(mle-paras)
Sd = rowSds(mle)
rmse = rowMeans((mle-paras)^2)^0.5
rrmse =((rowMeans((mle-paras)^2))/(paras^2))^0.5
```

Simulation performance based on 1000 sample paths

	Parameter		
	γ	$\bar{r}$	σ
Bias	0.44992	-0.00406	0.00138
Standard deviation	0.98044	0.15718	0.02501
Root mean square	1.0783	0.15715	0.02503
Relative Root mean square	0.2958	0.51074	0.0138

In the code, the matrix "mle" is the gamma, rbar, and sigma of each scenario. "bias" is the difference between the average gamma, rbar, and sigma, versus the gamma, rbar, and sigma estimated in Example 6, meaning this set of scenarios is "off" by that much. A bias very close to zero is desired. rbar and sigma appear to satisfy that, while gamma's bias is about 0.45. Is that tolerable?

To assess this we consider the root mean squared error (RMSE), which is the average residual, or the average across all scenarios of how far the three parameters are from the line of best fit. If we found another model or another calibration with a lower RMSE it would be a better fit to the underlying data.

The RMSE for gamma also looks large at 1.078. However, RMSE scales with the data so again it is hard to know. Relative RMSE (RRMSE) normalizes RMSE against the actual value. An RRMSE above 30% is considered a poor fit. The value for gamma is borderline, but now rbar is assessed as a very poor fit. We need to adjust the model calibration.

5.2 INVESTIGATE THE DATA

The example being studied here is from Veronesi [24]. We see the data is a daily short rate series from January 2, 2008 to March 5, 2010. This is through the beginning of the GFC. The series begins with the rate drifting down from 3.09% to 1.56%. On March 14, 2008, the rate jumps to 1.20%, on March 18 to 0.71%,

and on March 19 to 0.26%. This is when Bear Stearns collapsed and was taken over by J.P. Morgan. On March 25 the rate jumped +0.80% to 1.47%. Later, on September 15, 2008, the rate jumps down from 1.37% to 0.36% when Lehman Brothers collapsed.

The Vasicek model is not designed to handle jumps like this. We will need to try fitting a different model, such as a regime switching model or a jump-diffusion version of CIR.

Continuing with this example and the Vasicek model for a moment, the data series ends on March 5, 2010, with a short rate of 0.11%. The modeled interest rates are then projected forward from that point in time (cell J9 references cell C552, the continuously compounded version of the observed "nominal" short rate in B552). However, the data used to calibrate the model include 3%+ rates from before the two major financial system shocks.

The modeler may make some design choices here. If a local model of very low rates is desired, the Vasicek model could be recalibrated using data beginning September 15, 2008. However, this series is unlikely to generate scenarios with rates in the 3% to 6% range that were the norm before the GFC.

Instead of daily data the modeler could choose to use monthly data from a much longer time span, say 1980 to present. This could be desirable if the model for which the scenarios are being generated will run with a monthly time step. However, that time span had generally declining rates, so the long-run mean reversion target $\bar{r}$ is unclear.

There is no one clear correct choice. This is the art of modeling.

5.3 RECALIBRATE

Download the rate data from the US Treasury Department's website[4]. Open it in Excel. Filter the data to eliminate rows with a blank or an ND in the one-month column. Convert semi-annual rates to effective annual rates. Create a new tab and copy the date and one-month rate column for the dates needed into it. Insert a simple header row, say Date and OneMonthTBill. Save as a .xlsx file.

In RStudio's top right window, Environment tab, select Import Dataset. R may want to install an updated readxl package. Browse for and select the downloaded rate data file. In Import Options (lower left), select the new sheet. Import.

```
#example44
# refit to CIR example13
# The following code snippets calculate the bias of simulated parameters
# CIR Calibration
rm(list=ls()) # clear the workspace and functions
graphics.off() # clear the plots
# CIR Euler estimate
# Initialize parameters
```

[4]Market yield on U.S. Treasury securities at 1-month constant maturity, quoted on investment basis, series H15/H15/RIFLGFCM01_N.B can be selected from www.federalreserve.gov/DataDownload. Filtered data can be obtained directly at https://home.treasury.gov/resource-center/data-chart-center/interest-rates/TextView?type=daily_treasury_yield_curve&field_tdr_date_value=all&data=yieldAll.

```
library(matrixStats)
rt = VeronesiTable14p7q5$rt
N = length(rt)
r0 =rt[N]
t0=0
Delta = 1/252
X=rt
euler.est= matrix(NA,3)
X[X<0.01]=0.01 # floor rates at 1 bp
x1 = X[1:N]^(-0.5)
x2 = X[1:N]^(0.5)
y = X[2:(N+1)]*x1
model=lm(y~0+x1+x2)

gamma = (1-as.numeric(model$coefficients)[2])/Delta
rbar = as.numeric(model$coefficients)[1]/
     (1-as.numeric(model$coefficients)[2])
alpha= sigma(model)^2/Delta
gamma

## [1] 4.4757

rbar

## [1] 0.37285

alpha

## [1] 7.6638

paras = c(gamma,rbar,alpha)

# Now generate a scenario set with these parameters
# Initialize # of paths and number of points on each path
t0=0
T=5
Delta = 1/252
M =1000
N = ceiling((T-t0)/Delta)
set.seed(123)
# A version of CIR.Trans.Sim without locked parameters
scenarioSet = CIR.Trans.sim(t0,T,Delta,r0,gamma,rbar,alpha,M)
```

We now have 1000 scenarios and want to evaluate whether they are a good fit relative to our calibrated parameters. To do this we'll calculate the CIR parameters implied by each scenario, and see whether on average they are close to our calibrated parameters. If so, then the set is valid and the calibration "worked".

```
#example44 continues
M = ncol(scenarioSet)
X = scenarioSet
X[X<0.01]=0.01
euler.est= matrix(NA,3,M)

for (i in 1:M){
 x1 = X[1:(N-1),i]^(-0.5)
 x2 = X[1:(N-1),i]^(0.5)
 y = X[2:N,i]*x1
 model=lm(y~0+x1+x2)
 euler.est[1,i]= (1-as.numeric(model$coefficients)[2])/Delta
 euler.est[2,i] = as.numeric(model$coefficients)[1]/
                  (1-as.numeric(model$coefficients)[2])
 euler.est[3,i] = sigma(model)^2/Delta
}

bias=rowMeans(euler.est-paras)
sd =rowSds(euler.est)
rmse =(rowMeans((euler.est-paras)^2))^0.5
rrmse =((rowMeans((euler.est-paras)^2))/(paras))^0.5
```

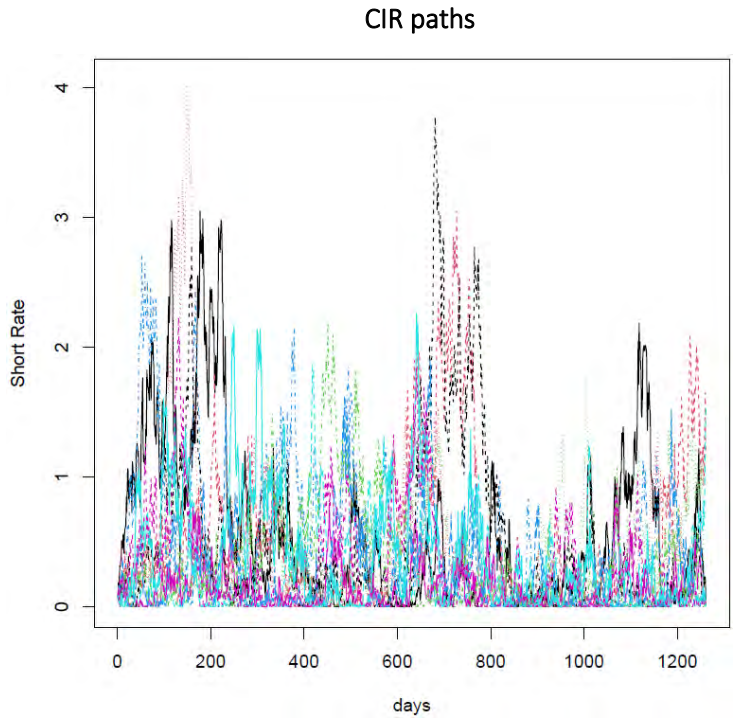
The code above was adapted from Example 13 code. We started with r0 set to 2.741129, the first value in the data series, so we produce a set of scenarios "around" the historical data used for calibration. Across the 1000 scenarios, the gamma, rbar, and alpha compare to the calibration gamma, rbar, and alpha as follows:

Simulation performance based on 1000 sample paths

	Parameter		
	γ	$\bar{r}$	α
Bias	0.65112	0.00504	−0.76063
Standard deviation	2.36859	0.19081	0.48787
Root mean square	2.45531	0.19078	0.90352
Relative Root mean square	1.16059	0.31244	0.32637

All three parameters show little bias, with rbar very close. The standard deviation across the 1000 scenarios is broader, which also shows up in the RMSE and relative RMSE metrics. Indeed, gamma and rbar have RRMSEs greater than 30%, indicating a poor fit. Looking at the scenarios produced,

```
# example 44 continues
# plot the first year of the 5%ile, 15%ile,.. 95n%ile scenarios sorted at the one year point
o = order(scenarioSet[252,])
matplot(scenarioSet[,c(1,o[-(M/20)+(M/10)*1:10],M)], type="l", main="CIR paths",
        ylab="Short Rate", xlab="days")
```



The scenarios show material dispersion during the first year, but they then settle down to extended low rates clustered in the 0% to 1.5% range. This is roughly the pattern of the historical data used for calibration, so the CIR model appears to produce scenarios in line with the calibration data, in this case. Calibrating to a historical dataset with higher or more variable rates might produce more dispersion in the scenario set. Alternatively, we notice the mean reversion strength gamma produced by our calibration is greater than 1.0. Reviewing the CIR model in equation (4), this could amplify volatility, which could be offset by the $\sqrt{\alpha r_t}$ volatility term. This emphasizes the need for assessing the scenario set produced by the model. Alternatively it is usable with the caveat that the scenario set only spans scenarios in a regime like 2017 to 2019 U.S. rates, but not scenarios from a different regime.

5.4 VALIDATE

Assuming that the scenario set is appropriate for the modeling work to be undertaken, we proceed to complete the validation process.

The Practical Guide [1] Section 6.3 explores desirable properties for a set of economic scenarios. Focusing on government yield curve projections, we assess the properties of our scenario set:

1. Yields for longer maturities are usually greater than yields for shorter maturities, i.e. the yield curve is “upward sloping”.
2. The volatility of short maturity yields tends to be greater than long maturity yields – but not during periods of central bank intervention.
3. Short and long maturity yields are highly correlated.
4. When short rates are low, long rates tend to be greater than short rates.
5. When short rates are high, long rates tend to be lower than short rates (i.e., the yield curve is said to be “inverted”). The above properties are not applicable for a one-factor short rate model, and point to the need to consider two-factor and other models.

6. Negative short rates are possible. The scenarios generated here do not span negative rates but they do produce many scenarios with rates close to zero.
7. Higher rates occur but do not persist for long. The scenarios generated here do not probe higher rates. The highest rate scenario at the one year point gets to about 6% and then reverts toward the mean reversion target of 0.37%.

Further stylized facts can be considered for corporate bond yields or spreads, equity market scenarios, and inflation and other economic variables. These are beyond the scope of models considered in this paper.

The following considerations may also apply:

1. Actual Results – is this model similar to others? We know that Vasicek and CIR scenario sets don't model long rates, inverted yield curves, or very low or high rate paths.
2. Evaluate with benchmarking and replication (static validation, parallel testing, spreadsheet replica). The Veronesi spreadsheet plots a zero-volatility trend developed in column J.
3. Evaluate with outcome analysis (dynamic replication, back-testing, out-of-sample testing). By about day 80 of the historical data, actual short rates went to zero as the central bank responded to the COVID-19 pandemic. The Vasicek scenarios did not encompass such an outcome, while the CIR scenarios seem to do so.
4. Sound/Stable Results – are results sound and stable across a range of use and scenarios? The scenarios are quite stable.
5. Stress, sensitivity, or extreme value testing. A test scenario set with gamma reduced or alpha increased may conform better to stylized facts, and would be worth exploring.
6. Dependencies and correlations – how is inflation correlated with the short rate? Should the inflation assumption in the broader model be linked to the interest rate scenario set, perhaps with a lag?

5.5 MODEL GOVERNANCE

As of this writing, standard practice is to submit model assumptions and design choices to a Model Governance Committee. The choices of model, data series, time steps, and the implications should be listed and described in accordance with applicable standards of practice. They then are submitted to the Model Governance Committee or the Principal sponsoring the work assignment for discussion and acceptance or further investigation. This aligns with standards of practice that a) results should be communicated in a useful and understandable way, and b) the model is appropriately documented and governed. Generally the interest rate or economic scenario model would be run as part of a larger modeling process, so documentation and review and acceptance may be part of the larger modeling effort.

A word of encouragement on documentation. We know we should do it, and standards of practice such as ASOP 56 Modeling, section 3.7 in the U.S. recommend it. Yet as a new procedure is developed it is unclear what to write, and as deadlines approach documentation falls by the wayside. Don't let the perfect be the enemy of the good. As the work progresses write down the project objectives, then the data sources, then the data transforms and loads, then the run procedure and results interpretation, then why the model is plausible. Do this as work proceeds. You will have a solid start at documentation! Then the next time the process is run, refine the documentation. Last, check back to the standards of practice and add an item or two to ensure compliance.

6 Conclusion

Each model has issues which become apparent when fit to an actual historical data series or a specific yield curve. In Section 5 we saw that the one-factor models have difficulty fitting historical data where exogenous shocks caused discontinuities, or “jumps”, in the data series. Adding a random jump term can be done, at the cost of additional complexity. The practitioner can also test the two-factor Vasicek model with correlated factors and the two-factor Hull-White model, as explored in Section 4. Both of these models are in the class of G2++ models, which can be explored further in [6] section 4.2. Ultimately the choice of model and calibration data comes down to art and professional judgement. Real world data is messy, with exogenous shocks from political shifts, central bank intervention, geopolitical events, and more. There's no one right answer, but rather informed choices and trade-offs. That's what makes it interesting. Best wishes.

7 Acknowledgments

The researchers' deepest gratitude goes to those without whose efforts this project could not have come to fruition: the Project Oversight Group and others for their diligent work in reviewing and editing this report for accuracy and relevance. Project Oversight Group members:

Ted Chang, FSA, MAAA, PhD

Gary Hatfield, FSA, MAAA, CFA, PhD

Jason Kehrberg, FSA, MAAA, PhD

Paul Ngai, FSA, FCIA

At the Society of Actuaries:

Doug Chandler, FSA FCIA

The volunteers who generously shared their wisdom, insights, advice, guidance, and arm's-length review of this study prior to publication. Any opinions expressed may not reflect their opinions nor those of their employers. Any errors belong to the authors alone.

Steve Strommen, FSA, MAAA

References

[1] *Economic Scenario Generators: A Practical Guide*. Society of Actuaries, 2016.

[2] Leif BG Andersen. Efficient simulation of the Heston stochastic volatility model. Available at SSRN 946405, 2007.

[3] David F Babbel and Frank J Fabozzi. Investment management for insurers, volume 43. John Wiley & Sons, 1999.

[4] Jean-François Bégin. Economic scenario generator and parameter uncertainty: A Bayesian approach. ASTIN Bulletin: The Journal of the IAA, 49(2):335-372, 2019.

[5] Victor Bernal. Calibration of the Vasicek model: a step by step guide. 2016.

[6] Damiano Brigo and Fabio Mercurio. Interest rate models - theory and practice. Springer Finance. Springer, Berlin, Germany, 2 edition, Aug 2007.

[7] Kalok C Chan, G Andrew Karolyi, Francis A Longstaff, and Anthony B Sanders. An empirical comparison of alternative models of the short-term interest rate. The Journal of Finance, 47(3):1209-1227, 1992.

[8] Eric Chin, Sverrir Olafsson, and Dian Nel. *Problems and Solutions in Mathematical Finance, Volume 1: Stochastic Calculus*. John Wiley & Sons, 2014.

[9] Rama Cont. *Encyclopedia of quantitative finance*. Wiley, 2010.

[10] John Cox, Jonathan Ingersoll, and Stephen Ross. A theory of the term structure of interest rates. Econometrica, 53:385-407, 02 1985.

[11] J. L. Doob. The Brownian Movement and Stochastic Equations, volume 43. Annals of Mathematics, 1942.

[12] Daniel Dufresne, Felisa Vázquez-Abad, and Stephen Chin. Change of measure for the square-root process. In Proceedings of the Winter Simulation Conference 2014, pages 465-475, 2014.

[13] William Feller. Two singular diffusion problems. Annals of Mathematics, 54(1):173-182, 1951.

[14] David M Gay. Usage summary for selected optimization routines. Computing Science Technical Report 153, AT&T Bell Laboratories, Murray Hill, NJ 07974, October 1990.

[15] James Douglas Hamilton. *Time Series analysis*. Princeton University Press, 1994.

[16] Lars Peter Hansen. Large sample properties of generalized method of moments estimators. Econometrica: Journal of the econometric society, pages 1029-1054, 1982.

[17] Stefano M Iacus. *Simulation and inference for stochastic differential equations: with R examples*, volume 486. Springer, 2008.

[18] Norman L Johnson, Samuel Kotz, and Narayanaswamy Balakrishnan. *Continuous univariate distributions, volume 2*, volume 289. John Wiley & Sons, 1995.

[19] Giuseppe Orlando, Rosa Mininni, and Michele Bufalo. Interest rates calibration with a CIR model. *The Journal of Risk Finance*, 20(4):370-387, 2019.

[20] Giuseppe Orlando, Rosa Maria Mininni, and Michele Bufalo. Forecasting interest rates through Vasicek and CIR models: A partitioning approach. *Journal of Forecasting*, 39(4):569-579, July 2020.

[21] Hansen Pei. *Mean-Reverting Spread Modeling: Caveats in Calibrating the OU Process*. Hudson and Thames Research, August 2021.

[22] Christian Ritz and Jens Carl Streibig. *Nonlinear regression with R*. Springer, 2008.

[23] Stephen J. Strommen. *Understanding the Connection Between Real-World and Risk-Neutral Scenario Generators*. Society of Actuaries, 2022.

[24] Pietro Veronesi. *Fixed income securities: Valuation, risk, and risk management*. John Wiley & Sons, 2010.

Appendix A: Zero coupon bond prices under one-factor Vasicek model

Under one factor Vasicek model zero-coupon bond prices are given by:

$$[24](15.28) \qquad Z(r,t;T) = e^{A(t;T)-B(t;T)r}$$

$$[24](15.29) \qquad B(t;T) = \frac{1}{\gamma^*}\left(1-e^{-\gamma^*(T-t)}\right)$$

$$[24](15.30) \qquad A(t;T) = (B(t;T)-(T-t))\left(\bar{r}^* - \frac{\sigma^2}{2(\gamma^*)^2}\right) - \frac{\sigma^2 B(t;T)^2}{4\gamma^*}$$

Note that we are using parameters γ^* and $\bar{r}^*$ to indicate the drift of the SDE for r_t, $\gamma^*(\bar{r}^* - r_t)$ in the risk-neutral world, as opposed to parameters γ and $\bar{r}$ with the drift of $\gamma(\bar{r} - r_t)$ in the real world.

When we fit these to market bond prices we need to use a numerical search algorithm. As [24] points out in many places it is possible to get negative values for the parameter γ^* so we must look at the behaviour of the bond price function in the neighbourhood of $\gamma^* = 0$. We see that $B(t;T)$ is a continuous function with the limit of $\gamma \to 0$ being $(T-t)$. However, we see that calculating the limiting value of $A(t;T)$ as $\gamma^* \to 0$ is zero is not an easy task based on [24] (15.30), but with some calculus we can prove that $A(t;T)$ is discontinuous at $\gamma^* = 0$. So if we do not address this properly in the implementation of numerical optimizations, we may end up getting wrong values. To calculate the bond price formula when $\gamma^* = 0$ takes some effort. For this we can either use the fact that when $\gamma^* = 0$ the Vasicek model reduces to a zero-drift Ho-Lee model and then using [24] (19.8)-(19.9), or we can use the following approach.

First notice that bond prices are given by:

$$Z(r_t,t;T) = E^{\mathbb{Q}}\left[\exp\left(-\int_t^T r_s ds\right) \mid r_t\right]$$

In the risk-neutral world the SDE of r_t is given as:

$$dr_t = \gamma^*(\bar{r}^* - r_t)dt + \sigma dX_t$$

The solution of the above SDE takes the following form:

$$r_{t+s} = r_t \exp(-\gamma^* s) + \bar{r}^*(1-\exp(-\gamma^* s)) + \sigma \exp(-\gamma^* s)\int_0^s \exp(\gamma^* u) dX_u$$

From this we can obtain:

$$\begin{aligned}\int_0^{T-t} r_{t+s} ds &= r_t \int_0^{T-t} \exp(-\gamma^* s) ds + \int_0^{T-t} \bar{r}^*(1-\exp(-\gamma^* s)) ds + \\ &\int_0^{T-t} \sigma \exp(-\gamma^* s)\left(\int_{u=0}^{s} \exp(\gamma^* u) dX_u\right) ds,\end{aligned}$$

Using Stochastic Fubini's theorem,

$$\begin{aligned}\int_0^{T-t} r_{t+s} ds &= r_t \int_0^{T-t} \exp(-\gamma^* s) ds + \int_0^{T-t} \bar{r}^*(1-\exp(-\gamma^* s)) ds + \\ &\int_{u=0}^{T-t} \sigma \exp(\gamma^* u)\left(\int_{s=u}^{T-t} \exp(-\gamma^* s) ds\right) dX_u, \\ \int_t^T r_s ds &= r_t \int_0^{T-t} \exp(-\gamma^* s) ds + \int_0^{T-t} \bar{r}^*(1-\exp(-\gamma^* s)) ds + \\ &\int_{u=0}^{T-t} \sigma \exp(\gamma^* u)\left(\int_{s=u}^{T-t} \exp(-\gamma^* s) ds\right) dX_u,\end{aligned}$$

Since $\int_t^T r_s ds \mid r_t$ is a normal random variable the bond pricing formula reduces to:

$$Z(r_t, t; T) = \exp\left[\mathbb{E}^{\mathbb{Q}}\left(-\int_t^T r_s ds | r_t\right) + \frac{1}{2}\mathbb{V}ar^{\mathbb{Q}}\left(-\int_t^T r_s ds | r_t\right)\right] \tag{12}$$

But:

$$\begin{aligned}
\mathbb{E}^{\mathbb{Q}}\left(-\int_t^T r_s ds | r_t\right) &= -r_t \int_0^{T-t} \exp(-\gamma^* s) ds - \int_0^{T-t} \bar{r}^*(1 - \exp(-\gamma^* s)) ds \\
\mathbb{V}ar^{\mathbb{Q}}\left(-\int_t^T r_s ds | r_t\right) &= \sigma^2 \int_{u=0}^{T-t} \exp(2\gamma^* u) \left(\int_{s=u}^{T-t} \exp(-\gamma^* s) ds\right)^2 du,
\end{aligned}$$

where the first equation follows from the fact that expectation of the Itô integral is zero and the second equation follows from Itô isometry. By comparing (12) with the [24] (15.28)-(15.30) we see that:

$$\begin{aligned}
B(t;T) &= \int_0^{T-t} \exp(-\gamma^* s) ds \\
A(t;T) &= -\int_0^{T-t} \bar{r}^*(1 - \exp(-\gamma^* s)) ds + \frac{\sigma^2}{2} \int_{u=0}^{T-t} \exp(2\gamma^* u) \left(\int_{s=u}^{T-t} \exp(-\gamma^* s) ds\right)^2 du
\end{aligned}$$

Readers are encouraged to verify that when $\gamma^* \neq 0$ the above integrals simplify to [24] (15.29) and (15.30). Now we can easily calculate values of $B(t;T)$ and $A(t;T)$ when $\gamma^* = 0$ by substituting $\gamma^* = 0$ in the above:

$$\begin{aligned}
B(t;T) &= \int_0^{T-t} ds \\
&= (T-t) \\
A(t;T) &= \frac{\sigma^2}{2} \int_{u=0}^{T-t} \left(\int_{s=u}^{T-t} ds\right)^2 du \\
&= \frac{\sigma^2 (T-t)^3}{6}
\end{aligned}$$

Note that when $\gamma^* = 0$ the model is a Ho-Lee model with zero drift and our result agrees with [24] (19.9).

Appendix B: Zero coupon bond prices under the two-factor Vasicek model

This model can be written as the short rate process written as the sum of two factors, a short rate factor and a long rate factor:

$$r_t = \phi_{1,t} + \phi_{2,t}$$

where each factor follows the following SDEs:

$$d\phi_{i,t} = \gamma_i\left(\vec{\phi}_i - \phi_{i,t}\right)dt + \sigma_i dX_{i,t}, \qquad i = 1,2.$$

With a few lines of algebra we can obtain the solution to these SDEs as follows:

$$\phi_{i,t+s} = \phi_{i,t}exp(-\gamma_i s) + \vec{\phi}_i\left(1 - exp(-\gamma_i s)\right) + \sigma_i exp(-\gamma_i s)\int_0^s exp(\gamma_i v)\, dX_{i,v}, \qquad i = 1,2.$$

The conditional means and variance of these two factors are given by:

$$\begin{aligned}
E[\phi_{i,t+s}|\phi_{i,t}] &= \phi_{i,t}\exp(-\gamma_i s) + \vec{\phi}_i(1 - \exp(-\gamma_i s)) \\
Var[\phi_{i,t+s}|\phi_{i,t}] &= Var[\sigma_i \exp(-\gamma_i s)\int_0^s \exp(\gamma_i v)dX_{i,v}, i = 1,2. \\
&= \sigma_i^2 \exp(-2\gamma_i s)\int_0^s \exp(2\gamma_i v)dv \\
&= \frac{\sigma_i^2}{2\gamma_i}(1 - \exp(-2\gamma_i s))
\end{aligned}$$

where the second to last equation follows from the Itô isometry. The conditional covariance between $\phi_{1,t+s}$ and $\phi_{2,t+s}$ conditioned on $\phi_{i,t}, i = 1,2$ is given by:

$$\begin{aligned}
Cov[\phi_{1,t+s}, \phi_{2,t+s}|\phi_{1,t}, \phi_{1,t}] &= E\left[\sigma_1 \exp(-\gamma_1 s)\int_0^s \exp(\gamma_1 v)dX_{1,v}\sigma_2 \exp(-\gamma_2 s)\int_0^s \exp(\gamma_2 u)dX_{2,u}\right] \\
&= \rho\sigma_1\sigma_2 \exp(-(\gamma_1 + \gamma_2)s)\int_0^s \exp(-(\gamma_1 + \gamma_2)v)dv \\
&= \frac{\rho\sigma_1\sigma_2}{\gamma_1 + \gamma_2}(1 - \exp(-(\gamma_1 + \gamma_2)s))
\end{aligned}$$

The conditional correlation coefficient between $\phi_{1,t+s}$ and $\phi_{2,t+s}$, which can be denoted as $\rho(s)$, is given by:

$$\begin{aligned}
Corr[\phi_{1,t+s}, \phi_{2,t+s}|\phi_{1,t}, \phi_{1,t}] &= \frac{\frac{\rho\sigma_1\sigma_2}{\gamma_1+\gamma_2}(1 - \exp(-(\gamma_1 + \gamma_2)s))}{\sqrt{\frac{\sigma_1^2}{2\gamma_1}(1 - \exp(-2\gamma_1 s))\frac{\sigma_2^2}{2\gamma_2}(1 - \exp(-2\gamma_2 s))}} \\
\rho(s) &= \rho\left(\frac{4\gamma_1\gamma_2}{(\gamma_1 + \gamma_2)^2}\frac{(1 - \exp(-(\gamma_1 + \gamma_2)s))^2}{(1 - \exp(-2\gamma_1 s))(1 - \exp(-2\gamma_2 s))}\right) \\
&= \rho\left(\frac{4\gamma_1\gamma_2(1 - \exp(-(\gamma_1 + \gamma_2)s))^2}{(\gamma_1 + \gamma_2)^2(1 - \exp(-2\gamma_1 s))(1 - \exp(-2\gamma_2 s))}\right)
\end{aligned}$$

The zero-coupon bond prices can be calculated by evaluating:

$$Z(r_t, t; T) = E^{\mathbb{Q}}\left[\exp\left(-\int_t^T r_s ds\right) \mid r_t\right]$$

Since $r_t = \phi_{1,t} + \phi_{2,t}$ let us look at obtaining $\int_t^s \phi_{i,u}du$ first:

$$\begin{aligned}
\phi_{i,t+s} &= \phi_{i,t}\exp(-\gamma_i s) + \overrightarrow{\phi}_i(1-\exp(-\gamma_i s)) + \sigma_i \exp(-\gamma_i s)\int_0^s \exp(\gamma_i v)dX_{i,v}, \quad i=1,2.\\
\int_0^{T-t}\phi_{i,t+s}ds &= \phi_{i,t}\int_0^{T-t}\exp(-\gamma_i s)ds + \overrightarrow{\phi}_i\int_0^{T-t}(1-\exp(-\gamma_i s))ds\\
&\quad + \int_0^{T-t}\sigma_i\exp(-\gamma_i s)\int_0^s \exp(\gamma_i v)dX_{i,v}ds
\end{aligned}$$

Using the stochastic Fubini's lemma we can exchange the order of the last integral:

$$\int_{s=0}^{T-t}\sigma_i\exp(-\gamma_i s)\left(\int_{v=0}^{s}\exp(\gamma_i v)dX_{i,v}\right)ds = \sigma_i\int_{v=0}^{T-t}\exp(\gamma_i v)\left(\int_{s=v}^{T-t}\exp(-\gamma_i s)ds\right)dX_{i,v}$$

Now we see that all three integrals can be simplified when $\gamma_i \neq 0$ and when $\gamma_i = 0$. However we leave them as it is for now and observe the following. Each integral is a normally distributed random variable and the means, variances and covariances are given as below:

$$\begin{aligned}
E\left[\int_t^T \phi_{i,s}ds|r_t\right] &= \phi_{i,t}\int_0^{T-t}\exp(-\gamma_i s)ds + \overrightarrow{\phi}_i\int_0^{T-t}(1-\exp(-\gamma_i s))ds, \quad i=1,2,\\
Var\left[\int_t^T \phi_{i,s}ds|r_t\right] &= \sigma_i^2\int_{v=0}^{T-t}\exp(2\gamma_i v)\left(\int_{s=v}^{T-t}\exp(-\gamma_i s)ds\right)^2 dv, \quad i=1,2,
\end{aligned}$$

When $\gamma_i \neq 0$ the above integrals simplify to:

$$\begin{aligned}
E\left[\int_t^T \phi_{i,s}ds|r_t\right] &= \phi_{i,t}\frac{1}{\gamma_i}(1-\exp(-\gamma_i(T-t))) - \overrightarrow{\phi}_i\left(\frac{1-\exp(-\gamma_i(T-t))}{\gamma_i} - (T-t)\right)\\
Var\left[\int_t^T \phi_{i,s}ds|r_t\right] &= \sigma_i^2\int_{v=0}^{T-t}\exp(2\gamma_i v)\left(\int_{s=v}^{T-t}\exp(-\gamma_i s)ds\right)^2 dv\\
&= \sigma_i^2\int_{v=0}^{T-t}\exp(2\gamma_i v)\left(\frac{\exp(-\gamma_i v)-\exp(-\gamma_i(T-t))}{\gamma_i}\right)^2 dv\\
&= -\sigma_i^2\left[\frac{1}{\gamma_i^2}\left(\frac{1-\exp(-\gamma_i(T-t))}{\gamma_i} - (T-t)\right) + \frac{1}{2\gamma_i}\left(\frac{1-\exp(-\gamma_i(T-t))}{\gamma_i}\right)^2\right]
\end{aligned}$$

By defining $B_i(t;T)$ as in [24],

$$B_i(t;T) = \frac{1}{\gamma_i}\Big(1-exp\big(-\gamma_i(T-t)\big)\Big), \quad i=1,2$$

we can write:

$$\begin{aligned}
E\left[\int_t^T \phi_{i,s}ds|r_t\right] &= \phi_{i,t}B_i(t,T) - \overrightarrow{\phi}_i\left(B_i(t,T)-(T-t)\right)\\
Var\left[\int_t^T \phi_{i,s}ds|r_t\right] &= -\sigma_i^2\left[\frac{1}{\gamma_i^2}\left(B_i(t,T)-(T-t)\right) + \frac{1}{2\gamma_i}B_i(t,T)^2\right]
\end{aligned}$$

Let us simplify the conditional covariance between $\int_t^T \phi_{1,s}ds$ and $\int_t^T \phi_{2,s}ds$ conditioned on r_t.

$$\begin{aligned}
&Cov\left[\int_t^T \phi_{1,s}ds, \int_t^T \phi_{2,s}ds|r_t\right]\\
&= \rho\sigma_1\sigma_2\int_{v=0}^{T-t}\exp((\gamma_1+\gamma_2)v)\left(\int_{s=v}^{T-t}\exp(-\gamma_1 s)ds\right)\left(\int_{s=v}^{T-t}\exp(-\gamma_2 s)ds\right)dv
\end{aligned}$$

when $\gamma_i \neq 0$ for $i = 1,2$ the above integral can be evaluated as:

$$
\begin{aligned}
&Cov\left[\int_t^T \phi_{1,s}ds, \int_t^T \phi_{2,s}ds|r_t\right] \\
&= \rho\sigma_1\sigma_2 \int_{v=0}^{T-t} \exp((\gamma_1+\gamma_2)v)\left(\frac{\exp(-\gamma_1 v)-\exp(-\gamma_1(T-t))}{\gamma_1}\right)\left(\frac{\exp(-\gamma_2 v)-\exp(-\gamma_2(T-t))}{\gamma_2}\right)dv \\
&= -\frac{\rho\sigma_1\sigma_2}{\gamma_1\gamma_2}\left[\left(\frac{1-\exp(-\gamma_1(T-t))}{\gamma_1}-(T-t)\right)+\left(\frac{1-\exp(-\gamma_2(T-t))}{\gamma_2}-(T-t)\right)\right. \\
&\quad \left.-\left(\frac{1-\exp(-(\gamma_1+\gamma_2)(T-t))}{\gamma_1+\gamma_2}-(T-t)\right)\right]
\end{aligned}
$$

By writing:

$$B_3(t;T) = \frac{1}{\gamma_1+\gamma_2}\Big(1-exp\big(-(\gamma_1+\gamma_2)(T-t)\big)\Big)$$

we can rewrite the above expression for covariance as below:

$$
\begin{aligned}
&Cov\left[\int_t^T \phi_{1,s}ds, \int_t^T \phi_{2,s}ds|r_t\right] \\
&= -\frac{\rho\sigma_1\sigma_2}{\gamma_1\gamma_2}\left[B_1(t,T)+B_2(t,T)-B_3(t,T)-(T-t)\right]
\end{aligned}
$$

As [6] explains the

$$Z(r_t,t;T) = exp\left(E^{\mathbb{Q}}\left[-\int_t^T r_s ds \mid r_t\right]+\frac{1}{2}VAR^{\mathbb{Q}}\left[-\int_t^T r_s ds \mid r_t\right]\right)$$

But

$$
\begin{aligned}
\mathbb{E}^{\mathbb{Q}}\left[-\int_t^T r_s ds|r_t\right] &= \mathbb{E}^{\mathbb{Q}}\left[-\int_t^T(\phi_{1,s}+\phi_{2,s})ds|r_t\right] \\
&= \mathbb{E}^{\mathbb{Q}}\left[-\int_t^T(\phi_{1,s}+\phi_{2,s})ds|r_t\right] \\
&= -\mathbb{E}^{\mathbb{Q}}\left[\int_t^T \phi_{1,s}ds|r_t\right]-\mathbb{E}^{\mathbb{Q}}\left[\int_t^T \phi_{2,s}ds|r_t\right] \\
&= -\phi_{1,t}B_1(t,T)-\overrightarrow{\phi}_1(B_1(t,T)-(T-t)) \\
&\quad -\phi_{2,t}B_2(t,T)-\overrightarrow{\phi}_2(B_2(t,T)-(T-t))
\end{aligned}
$$

$$
\begin{aligned}
Var^{\mathbb{Q}}\left[-\int_t^T r_s ds|r_t\right] &= Var^{\mathbb{Q}}\left[-\int_t^T(\phi_{1,s}+\phi_{2,s})ds|r_t\right] \\
&= Var^{\mathbb{Q}}\left[\int_t^T \phi_{1,s}ds|r_t\right]+Var^{\mathbb{Q}}\left[\int_t^T \phi_{2,s}ds|r_t\right]+2Cov\left[\int_t^T \phi_{1,s}ds, \int_t^T \phi_{2,s}ds|r_t\right] \\
&= -\sigma_1^2\left[\frac{1}{\gamma_1^2}(B_1(t,T)-(T-t))+\frac{1}{2\gamma_1}B_1(t,T)^2\right] \\
&\quad -\sigma_2^2\left[\frac{1}{\gamma_2^2}(B_2(t,T)-(T-t))+\frac{1}{2\gamma_2}B_2(t,T)^2\right] \\
&\quad -\frac{2\rho\sigma_1\sigma_2}{\gamma_1\gamma_2}\left[B_1(t,T)+B_2(t,T)-B_3(t,T)-(T-t)\right]
\end{aligned}
$$

Therefore:

$$
\begin{aligned}
&\mathbb{E}^{\mathbb{Q}}\left[-\int_t^T r_s ds|r_t\right]+\frac{1}{2}Var^{\mathbb{Q}}\left[-\int_t^T r_s ds|r_t\right] \\
&= -\phi_{1,t}B_1(t,T)-\phi_{2,t}B_2(t,T) \\
&\quad +\overrightarrow{\phi}_1\left(B_1(t,T)-(T-t)\right)-\frac{\sigma_1^2}{2}\left[\frac{1}{\gamma_1^2}\left(B_1(t,T)-(T-t)\right)+\frac{1}{2\gamma_1}B_1(t,T)^2\right] \\
&\quad +\overrightarrow{\phi}_2\left(B_2(t,T)-(T-t)\right)-\frac{\sigma_2^2}{2}\left[\frac{1}{\gamma_2^2}\left(B_2(t,T)-(T-t)\right)+\frac{1}{2\gamma_2}B_2(t,T)^2\right] \\
&\quad -\frac{\rho\sigma_1\sigma_2}{\gamma_1\gamma_2}\left[B_1(t,T)+B_2(t,T)-B_3(t,T)-(T-t)\right]
\end{aligned}
$$

However [24] writes zero-coupon bond prices as:

$$Z(\phi_{1,t},\phi_{2,t},t;T)=exp\left(A(t,T)-\phi_{1,t}B_1(t,T)-\phi_{2,t}B_2(t,T)\right)$$

Therefore:

$$
\begin{aligned}
A(t;T) &= \overrightarrow{\phi}_1\left(B_1(t,T)-(T-t)\right)-\frac{\sigma_1^2}{2}\left[\frac{1}{\gamma_1^2}\left(B_1(t,T)-(T-t)\right)+\frac{1}{2\gamma_1}B_1(t,T)^2\right] \\
&\quad +\overrightarrow{\phi}_2\left(B_2(t,T)-(T-t)\right)-\frac{\sigma_2^2}{2}\left[\frac{1}{\gamma_2^2}\left(B_2(t,T)-(T-t)\right)+\frac{1}{2\gamma_2}B_2(t,T)^2\right] \\
&\quad -\frac{\rho\sigma_1\sigma_2}{\gamma_1\gamma_2}\left[B_1(t,T)+B_2(t,T)-B_3(t,T)-(T-t)\right]
\end{aligned}
$$

Since:

$$B_i(t;T)=\int_0^{T-t} exp(-\gamma_i s)\,ds,\ \ i=1,2,3$$

it is obvious that $B_i(t;T), i=1,2,3$ are continuous in the neighbourhood of $\gamma_i=0$ and its value at $\gamma_i=0$ is $T-t$. However, $A(t;T)$ is not continuous around $\gamma_i=0$ for $i=1$ or $i=2$ and the function has to be evaluated using its integral expression.

$$
\begin{aligned}
A(t;T) &= -\overrightarrow{\phi}_1\int_0^{T-t}(1-\exp(-\gamma_1 s))ds-\overrightarrow{\phi}_2\int_0^{T-t}(1-\exp(-\gamma_2 s))ds \\
&\quad +\frac{1}{2}\left[\sigma_1^2\int_{v=0}^{T-t}\exp(2\gamma_1 v)\left(\int_{s=v}^{T-t}\exp(-\gamma_1 s)ds\right)^2 dv\right] \\
&\quad +\frac{1}{2}\left[\sigma_2^2\int_{v=0}^{T-t}\exp(2\gamma_2 v)\left(\int_{s=v}^{T-t}\exp(-\gamma_2 s)ds\right)^2 dv\right] \\
&\quad +\rho\sigma_1\sigma_2\int_{v=0}^{T-t}\exp((\gamma_1+\gamma_2)v)\left(\int_{s=v}^{T-t}\exp(-\gamma_1 s)ds\right)\left(\int_{s=v}^{T-t}\exp(-\gamma_2 s)ds\right)dv
\end{aligned}
$$

Therefore if $\gamma_1=0$ and $\gamma_2\neq 0$ $A(t;T)$ becomes:

$$
\begin{aligned}
A(t;T) &= -\overrightarrow{\phi}_2\int_0^{T-t}(1-\exp(-\gamma_2 s))ds \\
&\quad +\frac{1}{2}\left[\sigma_1^2\int_{v=0}^{T-t}\left(\int_{s=v}^{T-t}ds\right)^2 dv\right] \\
&\quad +\frac{1}{2}\left[\sigma_2^2\int_{v=0}^{T-t}\exp(2\gamma_2 v)\left(\int_{s=v}^{T-t}\exp(-\gamma_2 s)ds\right)^2 dv\right] \\
&\quad +\rho\sigma_1\sigma_2\int_{v=0}^{T-t}\exp(\gamma_2 v)\left(\int_{s=v}^{T-t}ds\right)\left(\int_{s=v}^{T-t}\exp(-\gamma_2 s)ds\right)dv
\end{aligned}
$$

Upon simplification we obtain:

$$\begin{aligned} A(t;T) &= \overrightarrow{\phi}_2(B_2(t;T)-(T-t))+\sigma_1^2\frac{(T-t)^3}{6}-\frac{\sigma_2^2}{2}\left[\frac{1}{\gamma_2^2}\left(B_2(t;T)-(T-t)\right)+\frac{1}{2\gamma_2}B_2(t;T)^2\right] \\ &\quad +\rho\sigma_1\sigma_2\left[\frac{(T-t)^2}{2}+\frac{\exp(-\gamma_2(T-t))(\gamma_2(T-t)+1)-1}{\gamma_2^2}\right] \end{aligned}$$

By symmetry when $\gamma_1 \neq 0$ and $\gamma_2 = 0$, the expression for $A(t;T)$ simplifies to:

$$\begin{aligned} A(t;T) &= \overrightarrow{\phi}_2(B_2(t;T)-(T-t))+\sigma_1^2\frac{(T-t)^3}{6}-\frac{\sigma_2^2}{2}\left[\frac{1}{\gamma_2^2}\left(B_2(t;T)-(T-t)\right)+\frac{1}{2\gamma_2}B_2(t;T)^2\right] \\ &\quad +\rho\sigma_1\sigma_2\left[\frac{(T-t)^2}{2}+\frac{\exp(-\gamma_2(T-t))(\gamma_2(T-t)+1)-1}{\gamma_2^2}\right] \end{aligned}$$

When both $\gamma_1 = 0$ and $\gamma_2 = 0$ the integral for $A(t;T)$ becomes:

$$\begin{aligned} A(t;T) &= \frac{1}{2}\left[\sigma_1^2\int_{v=0}^{T-t}\left(\int_{s=v}^{T-t}ds\right)^2dv\right]+\frac{1}{2}\left[\sigma_2^2\int_{v=0}^{T-t}\left(\int_{s=v}^{T-t}ds\right)^2dv\right] \\ &\quad +\rho\sigma_1\sigma_2\int_{v=0}^{T-t}\left(\int_{s=v}^{T-t}ds\right)\left(\int_{s=v}^{T-t}ds\right)dv \end{aligned}$$

Which simplifies to:

$$A(t;T) = (\sigma_1^2+\sigma_2^2+2\rho\sigma_1\sigma_2)\frac{(T-t)^3}{6}$$

About The Society of Actuaries Research Institute

Serving as the research arm of the Society of Actuaries (SOA), the SOA Research Institute provides objective, data-driven research bringing together tried and true practices and future-focused approaches to address societal challenges and your business needs. The Institute provides trusted knowledge, extensive experience and new technologies to help effectively identify, predict and manage risks.

Representing the thousands of actuaries who help conduct critical research, the SOA Research Institute provides clarity and solutions on risks and societal challenges. The Institute connects actuaries, academics, employers, the insurance industry, regulators, research partners, foundations and research institutions, sponsors and non-governmental organizations, building an effective network which provides support, knowledge and expertise regarding the management of risk to benefit the industry and the public.

Managed by experienced actuaries and research experts from a broad range of industries, the SOA Research Institute creates, funds, develops and distributes research to elevate actuaries as leaders in measuring and managing risk. These efforts include studies, essay collections, webcasts, research papers, survey reports, and original research on topics impacting society.

Harnessing its peer-reviewed research, leading-edge technologies, new data tools and innovative practices, the Institute seeks to understand the underlying causes of risk and the possible outcomes. The Institute develops objective research spanning a variety of topics with its strategic research programs: aging and retirement; actuarial innovation and technology; mortality and longevity; diversity, equity and inclusion; health care cost trends; and catastrophe and climate risk. The Institute has a large volume of topical research available, including an expanding collection of international and market-specific research, experience studies, models and timely research.

Society of Actuaries Research Institute
475 N. Martingale Road, Suite 600
Schaumburg, Illinois 60173
www.SOA.org