The Effectiveness of Structural Bond Pricing Models with Maximum Likelihood Estimation of Asset Dynamics Parameters

Dale Holborow

A thesis presented in partial fulfillment of the requirements for the Masters of Science degree program majoring in Financial Mathematics

THE UNIVERSITY OF QUEENSLAND
AUSTRALIA

November 2008

Department of Mathematics

School of Physical Sciences

Faculty of Engineering, Physical Sciences and Architecture

University of Queensland

Australia

Dedicated to Kirsty

The Effectiveness of Structural Bond Pricing Models with Maximum Likelihood Estimation of Asset Dynamics Parameters

Dale Holborow

Submitted for the degree of Masters of Science
November 2008

Abstract

This thesis empirically tests the performance of two structural bond pricing models when the input parameters are estimated via the pure proxy and maximum likelihood estimation (MLE) techniques. We test the models over a sample of fourteen bonds taken from firms with a simple capital structure and selected from the period 2002-2007. We aim to establish the validity of recent claims suggesting that MLE improves the performance of the structural models considerably, and that past failures of the models may relate more to errors introduced by parameter choices than by the form of the models themselves.

We find that the maximum likelihood approach results in consistently higher estimates of asset volatility, and that this in turn appears to increase the predicted yields. We also find strong evidence that a linear relationship exists between the pure proxy and MLE values, such that simple rules of thumb can be applied to quickly transform a pure proxy estimate into an equivalent MLE value without having to perform computationally intensive optimisation routines.

Ultimately however, we find that the models remain too inaccurate to be of practical use, regardless of which parameter estimation technique is used. In particular, when we examine the model performance on bonds aggregated by investment rating, we find that the models appear consistently unable to capture latent risk factors that are being priced by the market.

Declaration

Please find attached the following work, submitted as partial fulfillment of the requirements towards completion of the Masters of Science (majoring in Financial Mathematics) degree as hosted by the University of Queensland.

I declare that the following work contains material and ideas that have not, to the best of my knowledge, been previously written nor published by any other person, except where indicated and acknowledged at the appropriate locations throughout the text.

I submit this body of work with the knowledge and permission that it may be made publicly available and/or reproduced by the University of Queensland, now and in the future. Please note that any and all included programming source code shall remain the intellectual property of the author, and may be used only with permission.

Dale Holborow

Acknowledgements

I would like to take this opportunity to thank all those people who assisted me while I worked on this research paper.

First and foremost, I thank Dr Jamie Alcock for suggesting potential topics and agreeing to supervise my research project. I've enjoyed your honest opinions on university research, mathematical finance and careers in general.

Thanks must also go to Dr Karen Alpert for taking me on at short notice and helping out with the early stages of my research. Your feedback and advice has been very much appreciated.

Thanks to M. Imran for your stylish LyX template, and to those people who contribute free library functions to the Matlab FileExchange community. You prevented me from having to write a lot of LaTeX markup and CSV-parsers, and the time saved came in very handy.

A big thank you to the various friends and family who for the past year have listened to me talk constantly about how much work I had left to do, and how little time left to do it. You probably didn't understand my topic, and probably didn't care, but thanks for humouring me anyway.

Finally, and most importantly. Thank you Kirsty, for your continued patience and support. Sorry it took so long...

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background

Companies typically have a number of options available when choosing how to best to finance their operations. One of the most popular methods to borrow money for periods of more than one year is the issuance of corporate bonds. A bond is an agreement between two parties, whereby the issuer of the bond receives an amount of money from the debt holder at the commencement of the agreement. In return, the issuer promises to pay the debt holder periodic interest payments (coupons) and the face value, or principal amount, when the bond matures. Corporate bonds are attractive to the issuing firm as a method of financing long-term projects because they can be configured to have a lifetime equal to the project in question, may include tax incentives, and feature regular, predictable cash flows, amongst other benefits. Whenever a firm issues a bond, there exists the possibility that they will be unable to repay some or all of that debt. This is known as default risk.

Despite the widespread use of corporate bonds, the question of how to correctly price default risk is still subject to intense scrutiny and research. As yet, no single unified approach to pricing risky corporate bonds exists.

A large number of mathematical models have been proposed as solutions. The riskiness of a bond is influenced by a multitude of factors including the value and liquidity of the firm's assets, leverage ratios, interest rates and the time to maturity. Each model attempts to incorporate some subset of these influencing factors to predict bond prices as witnessed in real markets. Depending on the specific nature and input parameters of the model, these attempts to mathematically predict market prices of corporate bonds have met with varying levels of success.

1.2 Motivating problem

Much academic research is focused on trying to price risky corporate bonds using a structural modelling approach. Structural models incorporate various mathematical frameworks to model credit risk in a process known as contingent claims analysis (CCA). CCA is based on the option pricing works of Black and Scholes [1973] and Merton [1973], in which these authors discuss the possibility of treating equity and debt as opposing claims on the assets of a firm.

The first structural model of Merton [1974] offers an easy and computationally efficient method of pricing corporate bonds, when several simplifying and restrictive assumptions are applied. Subsequent generations of structural models have been presented in attempts to relax these original restrictions, in order to make the models more applicable to real world market conditions.

Until relatively recently, there has been little in the way of published findings discussing the ability of structural models to accurately price individual bond issues. Empirical tests [Eom et al., 2004, Li and Wong, 2008] suggest that despite the many modifications and alleged improvements to the original Merton model, structural models remain unable to accurately predict bond prices as actually witnessed in the marketplace.

Some recent suggestions to improve model performance have focused less on the nature of the models themselves, and more on the process of accurately estimating the model input parameters [Duan, 1995, Duan et al., 2003, Ericsson and Reneby, 2005]. These parameters often represent properties whose values are not directly observable, such as the instantaneous risk-free interest rate, the market value of debt, and the firm asset volatility. Depending on the particular input parameter, various techniques to estimate the values have been presented. The effects of several of these estimation techniques on the prediction power of the bond models has been studied in detail [Ericsson and Reneby, 2005, Li and Wong, 2008].

This thesis focuses on testing the performance of two of the more popular structural bond pricing models, those of Merton [1974] and Longstaff and Schwartz [1995]. In particular, we test the models using two different techniques to estimate the input parameters relating to firm asset dynamics and value. We then compare the relative performance of each structural model with each set of parameter estimates, and see whether resulting performance is significantly improved.

1.3 Goals and objectives

We set out the following points defining the aims and intentions of this thesis. We will:

- Provide an introductory overview of the underlying concepts behind the structural modelling approach.
- Discuss the simplifying assumptions of several structural models and their likely effect on the relevance and applicability of those models to practical scenarios.
- Summarise the subsequent generations of structural models, including the particular shortcomings of the original Merton [1974] model that they target.
- Summarise the findings from previous studies that empirically test and compare the performance of several popular structural models.
- Describe and implement the maximum likelihood estimation (MLE) method to arrive at improved estimates of input parameters related to asset dynamics and value.
- Compare and contrast the difference in the output from the MLE method as opposed to a pure proxy method of parameter estimation.
- Implement two simple structural models over a small homogeneous sample of bonds taken from the United States corporate bond market, using both methods of parameter estimation.
- Make quantitative statements about the relative performance of each structural model and each estimation technique.
- Offer suggestions about what we believe are the key shortcomings in the models, if indeed there are any.
- Finally, state whether the structural modelling approach appears suitable for predicting market bond prices, or if further effort is required to make this theoretically appealing approach more practical and accurate enough to price individual bond issues.

1.4 Outline of paper

The remainder of this thesis adheres to the following outline:

In Chapter (2) we review existing literature, beginning with a description of the various approaches to mathematically model the credit risk of corporate bonds. This section continues with an explanation of option pricing theory and how it is applied as contingent claims analysis to price corporate bonds, and includes descriptions of a number of the most popular approaches starting with the original Merton [1974] model.

Chapter (3) outlines the selection of two structural models for testing and also discusses the selection of an interest rate model to describe the dynamics of the US Treasury risk-free rate. Chapter (3) also describes the process of maximum likelihood estimation and how it is applied to the problem of estimation input parameters for the bond pricing models.

Chapter (4) includes sections on the collection and collation of all the data required to perform our tests. We outline the various sources and any difficulties in matching data sets, and describe the the overall characteristics of our end sample. We demonstrate the process of fitting the Vasicek [1977] model to an observed interest rate term structure. Finally, we describe the various simplifying assumptions made, including any assumed parameter values based on prior research.

Chapter (5) presents the results of the parameter estimation techniques and the trickle-down effect on the accuracy of the structural models.

We conclude with Chapter (6), and offer a summary of the project, an overview of the results and the contributions this study makes to the existing literature.

Chapter 2

Literature review

2.1 Introduction

The following chapter outlines the history of attempts to estimate market prices of corporate bonds using a variety of mathematical models. It includes an overview of several studies that aim to quantitatively establish the relative success, or lack thereof, of these models. We discuss trends in research, and the incremental revisions of the models designed to address issues with, and improve upon, earlier promising works.

We finish with an overview of recent developments and suggest interesting topics for potential future research.

2.2 Classes of corporate bond pricing models

Pricing a risk-free bond with repayments are guaranteed and interest rates are constant is easy. Simply use Equation (2.1) to discount the principal amount and any intermediate n coupon payments back to a present value,

$$B^{RF} = \frac{X}{(1+y)^n} + cX\left[\frac{1-(1+y)^{-n}}{y}\right] \tag{2.1}$$

where $c \geq 0\%$ is the coupon interest rate, X the face value of the bond, and y the yield, i.e. the rate at which the investment is discounted.

However, in practice bond repayments are rarely guaranteed. As the risk of default increases, the price of the bond decreases and hence the yield to investors increases. The difference in yield between a risky and equivalent risk-free investment is called credit spread. To compare investment opportunities investors typically refer to yields as opposed to the bond price, with a relationship described by

$$y = -\frac{1}{T-t}\ln\left(B(t,T)\right) \tag{2.2}$$

where t, T are times of observation and to maturity respectively, $B(\cdot)$ is the price of the bond under examination, and y is the yield an investor will make if they hold that bond until maturity.

Attempts to predict the price of risky corporate bonds have resulted in the development of three main classes of model: empirical, reduced and structural models. Each approach differs considerably in the underlying assumptions and implementation.

2.2.1 Empirical pricing models

Empirical models assume that it is difficult, if not impossible, to use probabilistic mathematical techniques to model the risk of debt. Instead, empirical models assign broadly indicative scores to various aspects of the issuing company's operations and financial health. These scores are then used to compare companies that have defaulted on their debts against those that have not. One of the most widely used empirical models is the Altman Z-Score [Chacko et al., 2006].

The Altman [1968] model incorporates five simple financial ratios, including working capital versus total assets and market value of equity versus book value of debt, to assign a score describing the likelihood that a firm will default within two years. Altman claims an impressively accurate bankruptcy prediction rate of between 70-90% over a sample of 66 firms, suggesting the approach may be suitable for certain practical applications.

An empirical modeller compiles scores for a large sample of companies across different industries and economic scenarios. By comparing the Z-score of similar companies operating under similar economic circumstances, one attempts to estimate the relative credit risk of the issuing firm. Similar comparisons are then used to select an appropriate yield with which to discount the firm's bond.

The empirical method is arguably the least technical method of pricing bonds. It requires a large amount of reference scenarios and historic financial data from similar companies in order to generate convincing price estimates and accurately estimate risk. Despite this, empirical bond models are popular amongst accountants and auditors for their ability to quickly and clearly compare and contrast the risk of corporate debt [Eidleman, 1995].

2.2.2 Reduced form pricing models

Reduced form models avoid referring to financial reports to analyse the financial strength of an issuing company. Rather, these models assume that default is triggered by some unexpected event, occurring at unpredictable times and as a result of a sudden decrease in the issuing company's market value of assets [Chacko et al., 2006]. Reduced form models appeal to proponents of the efficient market hypothesis, who argue that default is an unexpected event triggered exogenously by circumstances unforeseen by the issuing firm. If default were foreseeable the firm would act to prevent the default, possibly by restructuring.

Reduced models utilise market price observations of existing products in order to value new debt, believing that market participants actively factor all available information into the market price. Those observed prices can therefore be used as a foundation upon which to base all subsequent pricing decisions [Jarrow and Protter, 2004], under the assumption that past performance is representative of future outcomes.

The frequency of default in any company is typically modelled using Poisson processes, with the challenge being to select an appropriate value for the intensity of the default process. The default distribution function is typically rearranged to back out an implied default frequency. A modeller uses observed market prices of bonds previously issued by the target company as well as companies of similar size and perceived health, operating in similar industries [Jarrow et al., 1997]. Despite being theoretically appealing, the ability of reduced form models to accurately price risky bonds is still unproven and remains the subject of much ongoing study.

2.2.3 Structural pricing models

The final class of model, and the focus of this paper henceforth, is the structural approach. Structural models examine the capital structure of the underlying company by considering values of existing assets and debt, interest rates, debt service priorities and other such factors. By factoring in these various elements, the models predict the credit quality of a firm [Elizalde, 2005], and as a result, the market value of risky debt issues.

Structural models use a probabilistic approach to mathematically model dynamic processes such as changes in firm asset value and interest rates. They calculate the probability that the firm's asset value will drop below some default threshold and trigger bankruptcy before the bond is repaid. Based on the expectation of the payout, structural models predict a bond price according to its level of credit risk.

The remainder of this chapter focuses on examining structural models in detail.

2.3 The original structural model

In this section we outline the history of the structural modelling approach, and discuss the including underlying assumptions and theory.

2.3.1 The option pricing theory of Black, Scholes and Merton

The origins of structural bond price modelling lie in the collective realisations of Black and Scholes [1973] and Merton [1973], who present a consistent theoretical framework for option pricing in the celebrated papers, The Pricing of Options and Corporate Liabilities [Black and Scholes, 1973], and Theory of Rational Option Pricing [Merton, 1973]. An option grants the holder the right, but not the obligation, to purchase some underlying asset at an agreed time for an agreed price. The option price depends on a variety of conditions including the length of time until maturity, the volatility of the underlying asset value, and the particular payoff structure of the option.

Option pricing theory typically requires arbitrage-free market conditions (the idea that it is impossible in a mature market to make a risk-free profit). This suggests that all investors act rationally in their best interests and that market prices accurately reflect all information available to all investors, and therefore indicate fair value. This last point is commonly expressed as the efficient market hypothesis (EMH) [Fama, 1965]. The EMH is a recurring idea in financial literature that suggests, among other things, that all information relevant to valuing a traded security is already incorporated into its market price, and so an investor is unable to use past price movements to predict future trends. This suggests also that analysts can use existing market prices for a fully observable asset to value suitably similar assets in the absence of complete information [Derman, 2001, 2002].

Black and Scholes use this framework and consider bonds as options on a firm's assets. In return for an initial investment in a bond the bond holder has the right, but not the obligation, to take ownership of a company's assets should that company fail to repay their debts. Similarly, equity in the firm can be considered as a call option on the assets of the firm. Equity holders have a residual claim on any assets remaining after all liabilities to debt holders are met, that is, the equity has a strike equal to the value of liabilities.

Black and Scholes assume that the value of an underlying asset, for example a stock, fluctuates continuously in time according to a log-normal distribution, described by

$$dS_t = \mu S_t dt + \sigma S_t dW_t \tag{2.3}$$

where S_t is the share price at time t. μ and σ are the drift and volatility of the asset value and are constant through time. W_t is a standard Brownian motion process. Black and Scholes imagine a risk-neutral pricing environment in which investors do not demand extra return for holding increased risk, and so all assets return the instantaneous risk-free rate, i.e. $\mu = r$ in Equation (2.3), where r represents the instantaneous risk-free interest rate. In this risk-neutral environment, Black and Scholes establish a price for European call and put options on some asset as given by the formulas

$$\begin{aligned} C^{BS}\left(V_0, X, T\right) &= V_0 \Phi\left(d_1\right) - X e^{-rT} \Phi\left(d_2\right) \qquad (2.4)\\ P^{BS}\left(V_0, X, T\right) &= X e^{-rT} \Phi\left(-d_2\right) - V_0 \Phi\left(-d_1\right) \\ d_1 &= \frac{\ln\left(V_0/X\right) + \left(r + \frac{1}{2}\sigma^2\right) T}{\sigma\sqrt{T}} \\ d_2 &= d_1 - \sigma\sqrt{T} \end{aligned}$$

where Φ is the cumulative Normal distribution, X is the strike price of the option, V_0 is the value of the underlying asset at time $t = 0$ and T is time remaining until the option matures. The drift of the underlying asset value is not required in the option pricing formula. Therefore the same formula can be used in a risk-averse world to price options fairly, regardless of required rates of return for individual investors.

Building on this foundation, Merton proposes the first structural model in the landmark paper On the Pricing of Corporate Debt [Merton, 1974]. Using a combination of analysis of financial statements and option pricing theory, it should be possible to calculate an appropriate value for corporate bonds. The value of a firm's assets must be given by

$$V_t = E_t + D_t \tag{2.5}$$

where V_t is the total market value of the firm's assets, E_t is the market value of shareholder equity, and D_t is the market value of debt, true for all times t. By observing asset and equity prices and the asset dynamics, one can calculate the implied value of debt.

Applying the option pricing methodology to the problem of pricing corporate bonds, Merton assumes that the value process of the firm fluctuates according to geometric Brownian motion under a risk-neutral measure $\mathbb{Q}$.

$$dV_t = \mu V_t dt + \sigma V_t dW_t \tag{2.6}$$

where μ and σ are the time-constant drift and volatility of a geometric Brownian motion process, and W_t is taken to be a standard Brownian motion process. Under Merton's model, the payoff of a bond that pays no coupons, with maturity time T is $\min(X, V_T)$ where X is the face value of the bond and V_T is the value of assets at maturity. By considering a corporate bond as a risk-free bond less the cost of a put option on the firm assets, the price of a bond under the Merton [1974] model becomes

$$B^M(V_0, X, T) = Xe^{-rT}\left\{\Phi\left(h_2\left(d,\sigma^2,T\right)\right) + \frac{1}{d}\Phi\left(h_1\left(d,\sigma^2,T\right)\right)\right\} \tag{2.7}$$

where

$$\begin{aligned} d &\equiv \frac{Xe^{-rT}}{V_0} \\ h_1\left(d,\sigma^2,T\right) &= -\left[\frac{1}{2}\sigma^2 T - \ln(d)\right] / \left(\sigma\sqrt{T}\right) \\ h_2\left(d,\sigma^2,T\right) &= -\left[\frac{1}{2}\sigma^2 T + \ln(d)\right] / \left(\sigma\sqrt{T}\right) \end{aligned}$$

Recall that the price of the bond in the risk-neutral measure does not depend on the drift of the company assets, only the volatility. A structural bond price modeller prices risky bonds by simplifying and analysing the financial structure of the firm and calculating expected payoffs with regard to default probabilities under the risk-neutral measure $\mathbb{Q}$. Because the bond price does not depend on the asset drift μ, the bond price also holds true in the real probability measure $\mathbb{P}$.

2.3.2 Assumptions and requirements of Merton's structural model

The Merton [1974] structural model is appealing as it uses a minimal number of parameters to describe the financial health and behaviour of a company's assets. It is easily implemented with values retrieved directly from the firm's financial statements and observed market prices of equity.

The model's analytical solution and subsequent appeal hinge somewhat on its simplicity and implied ease of use. This simplicity is the result of several restrictive assumptions regarding investor behaviour and overall market operations. An outline of these assumptions and a brief evaluation of their validity follows.

A1 Company assets can be accurately valued by referring to financial performance statements that are regularly delivered to the market. This marked asset price represents the value which could be recouped in the event of forced liquidation, through sale of said assets.

A2 There are no transaction costs or taxes, and securities are available in perfectly divisible quantities, allowing an investor to purchase some fraction of an asset in order to pursue their desired investment strategy.

A3 The market allows short-selling of any and all assets with the seller able to reinvest the proceeds without restriction.

A4 The market is large and liquid enough that any investor is always able to buy and sell as many assets as they require without affecting the market price.

A5 Investors may borrow and lend at a single risk-free rate of interest r, specified as the rate of return for assets that are not subject to default risk.

A6 Trading takes place continuously in time.

A7 In the absence of taxes and default costs, firm value is invariant to its capital structure, as stated in the Modigliani-Miller theorem. That is, no particular ratio of company debt/equity dominates any other when structuring a company for maximum value.

A8 The value of a risk-less bond can be discounted through time at the constant instantaneous rate r.

A9 The firm asset value fluctuates through time according to a stochastic process described by equation (2.6).

A10 The asset structure of the firm consists of a single debt issue and equity. The company is unable to issue either new equity or senior debt until all existing claims are fulfilled.

A11 The bond to be issued is a non-callable zero coupon bond (i.e. early repayment is not allowed and and there are no interest payments made between the issue date and maturity date).

A12 A company can only default on its debt on the maturity date of the bond, never before, and does so if the value of the company is less than that of the due debt payment obligation. In the event of default, equity becomes worthless, company operations cease and bond holders take over immediate ownership of the company's assets after a costless bankruptcy process. This implies strict observance of the absolute priority of liabilities rule, which dictates that equity holders receive no payoff until the full liabilities to debt holders have been met.

Under these assumptions, Merton derives his model by extending the analytical option pricing formula of Black and Scholes [1973] to model the contingent claims on company assets.

Although attempting to model real-world phenomena, the Merton model forgoes absolute realism in order to gain some level of tractability and ease of implementation. The same simplifications that make the model appealing are a potential source of error when the model is applied to scenarios more complex than those originally specified. Before asking how successful Merton's model is in accurately predicting market prices of corporate debt, a detailed analysis of the aforementioned assumptions and limitations is required. We question whether these simplifications prevent the model from achieving realistic and/or useful results, and whether attempts to reduce the rigidity of such requirements has in turn produced perceptible benefits.

2.3.3 Limitations and criticisms of the Merton model

The Merton [1974] model allows for easy analytical calculation of fair value of risky corporate bonds subject to certain prerequisite assumptions, but these same assumptions call into question its validity. Jones et al. [1984] perform the first comprehensive empirical testing of Merton's model across a broad spectrum of debt grades. They find that the model tends to over-predict bond prices by over 4%, and as a result, implicitly under predict the risk-premium discount rate.

Numerous subsequent studies [Anderson and Sundaresan, 2000, Eom et al., 2004] further dissect each requirement and assumption. Empirical results typically suggest that the Merton structural model is oversimplified in its original form, and that the model does not predict market prices of corporate debt to a degree that would allow its use in practical applications. Two questions that must be immediately asked when validating a model's performance and trying to establish the source of any existing errors, are:

1. Is the model itself fundamentally flawed because certain assumptions or prerequisite conditions are in fact inappropriate, and
2. If the model is based on valid assumptions and conditions, is it the input parameters which lead to incorrect results?

As we examine the underlying assumptions of the Merton model, we see that indeed many are potential sources of inaccuracy, and are at least worthy of critical discussion.

2.3.3.1 Assumptions A1 and A9

Asset valuation as an input into the Merton [1974] model is historically established by reviewing the book value of company assets and debt as stated in a company's regular financial statements, added to market values of equity in what is called the pure proxy approach [Li and Wong, 2008]. Volatility of the firm assets are then directly calculated from these pure proxy asset values.

However, because the book values of debt are constant between statements, the pure proxy asset volatility is influenced only by movement in the share price. Li and Wong find that this approach results in an upwardly biased estimate of asset value, which in turn results in significant underpricing of corporate debt.

Structural bond models typically assume that firm asset volatility is constant throughout time. This assumption is found to be unrealistic by Black and Cox [1976], and will be shown to be demonstrably false in later chapters outlining our results.

2.3.3.2 Assumptions A2, A3, A4 and A7

Assumptions of no taxes and no transaction costs are demonstrably false in most financial markets, which in turn negates assumptions that firm value is strictly invariant to capital structure. The existence of taxes has a tangible effect on optimal capital structure as companies take on tax-deductible debt to finance operations. Elton et al. [2001] note that interest payments on US corporate bonds are sometimes taxed at the state and local levels while government bonds typically are not, and bond investors demand a greater discount rate to counteract the resulting losses through taxation. High risk, speculative-grade debt is subject to higher coupon rates as a result of the increased risk. This taxation effect is particularly evident for such debt.

The lack of absolute divisibility of securities and the existence of transaction costs mean that investors cannot always purchase securities in the precise numbers, nor at the continuous time increments, that would be desirable when financing and hedging market positions. Similarly, short-selling is neither free nor always available in practice. It may be impossible to find counterparties to take on the opposite side of the investor's intended position, or the counterparty may be willing to do so only at a prohibitively high cost.

2.3.3.3 Assumptions A5 and A6

The assumption of a single rate of borrowing and lending is unlikely to hold. The financial institutions that mediate transactions between two parties typically charge different rates depending on whether they operate on the buying or selling side, making their profit on the bid-ask spread applied to each position and any additional transaction fees charged.

2.3.3.4 Assumption A8

The constant interest rate assumption has been the subject of much discussion, with the significance of this simplification widely contested [Jones et al., 1984, Leland, 1994a, Longstaff and Schwartz, 1995, Leland and Toft, 1996]. The yield curve of any financial instrument is simply a relation between the discounting interest rates and time to maturity for that investment.

Whilst some bond pricing models assume constant interest rates, that is, a flat yield curve, in practice yield curves are dynamic in time, and are subject to both market forces and direct government intervention. Given that interest rates are not constant, and that discount yield rates are particularly critical for bond valuation as the bond maturity time increases, several bond models incorporate variable interest rates [Longstaff and Schwartz, 1995, Briys and de Varenne, 1997, Collin-Dufresne and Goldstein, 2001].

Unfortunately variable interest rates are difficult to model and may introduce new peculiarities depending on the interest rate model selected. Chan et al. [1992] suggest that there is still considerable scope for further research in modelling interest rate dynamics, which will turn affect those structural bond pricing models that implement stochastic interest rates.

2.3.3.5 Assumptions A10, A11 and A12

Lastly, we mention the Merton [1974] model's assumptions relating to capital structure, debt issuance provisions, and default boundaries and proceedings. These are arguably some of the most important and yet simultaneously most over-simplified of all the assumptions outlined.

In order to facilitate the options-on-assets reasoning which is the critical focus of the Merton structural model, a most simple capital structure is assumed. The target company consists solely of two contingent claims on company assets. These assets are a single seniority level of shareholder equity and a single seniority level of debt, as described by Equation (2.5).

In practice however, companies often operate with a complex capital structure, including different levels of seniority in both equity and debt and a variety of types of debt of varying maturities. Payments to senior debt holders must be made in their entirety before more junior debts can be serviced. A firm with both senior and subordinated debt must consider that servicing junior debt in any payment interval removes funds from the firm and may result in difficulties paying more senior debt at some later date.

Black and Cox [1976] note that for bonds of similar face value and maturity period but differing seniorities, the seniority-of-debt safety covenants make senior debt clearly more valuable than otherwise equivalent junior issues. Subordinated bond prices tend to exhibit different sensitivity to model inputs compared to senior and single-level issues. Because of this additional complexity many empirical bond model test samples exclude firms with different seniorities of debt. The question of how to properly price bonds from firms with complex capital structures remains unanswered.

Another modelling issue related to corporate structure is that of indenture agreements or covenants. Bond covenants are terms and conditions specified by investors in risky debt that dictate how the issuing company is allowed to operate in future. Such covenants grant debt holders additional powers so that they are more likely to recoup their investment should the issuing firm default.

Examples of typical conditions include:

- forbidding the issuance of debt that is more senior or equivalent to that of the debt issue in question,
- restricting raising of new funds to issuing only new equity and not new debt,
- forbidding dividend payments to equity holders until debt interest payments have been paid in full,
- allowing the bond to be called, (i.e. demanding immediate payment of face value) should the firm hit pre-agreed levels of financial stress, or fail to meet some scheduled interest payment.

It logically follows that any bond issued with such conditions must be more valuable than an equivalent debt without them [Black and Cox, 1976]. Although Merton notes that covenants imposed in an indenture, such as the callability and seniority of debt, will all have an effect on the bond price, these effects are not explicitly captured by his model.

Finally, various studies question Merton's assumptions regarding default boundaries and bankruptcy proceedings. Merton's model suggests that risky debt survives as a single lump sum payable at maturity, at which time debt holders receive the payoff

$$B(V_T, T) = \min(V_T, X) \tag{2.8}$$

where X is the face value of the bond, V_T is the value of the firm at maturity time T. By comparison, equity holders receive the payoff

$$E(V_T, T) = \max(V_T - X, 0) \tag{2.9}$$

That is to say, at maturity time $t = T$, equity holders pay the value of outstanding debt to debt holders if possible, and retain any residual assets. If a firm is unable to meet its debt obligations, default occurs and immediate costless bankruptcy proceedings ensue. In the event of bankruptcy equity holders receive nothing. Debt holders assume immediate ownership of the company, cease operations, and liquidate any remaining assets in an attempt to recoup as much of their initial investment as possible.

Merton assumes that debt issuers face default only at maturity and when the value of the firm's assets is strictly below the face value of debt X. This implies that investors are not witness to the evolution of the firm value through time and that default occurs as a surprise, with no attempt made by either debt or equity holders to restructure the firm should it encounter financial stress. In practice a firm may default at any time as market participants watch for signs that the issuer may exhaust its finances and cease operations before the bond maturity date is reached.

Fan and Sundaresan [2000] note that in practice, approximately half of long term bond defaults actually lead to bankruptcy while the remainder lead to the restructure of the defaulting firm. Various alternative frameworks have been proposed which, for example, incorporate a minimum time spent below the default threshold. Such models separate default and bankruptcy into distinct events occurring at different times [Acharya et al., 2000]. Bankruptcy proceedings often feature lengthy bargaining-game scenarios between debt and equity holders, as highlighted in research by Longstaff and Schwartz [1995], Anderson and Sundaresan [1996], and Fan and Sundaresan [2000]. In response, models that incorporate elements of game theory to capture the conflicting interests of equity and debt holders have been suggested [Anderson et al., 1996], and explore the idea that absolute priority of debt is often violated in practice [Franks and Torous, 1989].

In summary, the Merton model is subject to many potential sources of error, including the problem of correctly calculating the various input parameters and the restrictive assumptions underpinning it. These assumptions, their relevance, and their effects are the subject of much subsequent research. Despite the apparent flaws, structural models interest both researchers and practitioners because they suggest a direct and observable relationship between the market factors that influence the riskiness of a company, and the amount and price of the default risk that occurs as a result. Should structural models prove able to quantify the effects that factors such as interest rates and leverage ratios have on the market prices of risky bonds, then a market analyst is able to not only predict fair prices, but also use observed market prices to back out market expectations for different securities. In response to the flawed but theoretically appealing framework set out by Merton, numerous more complex structural models have been suggested, each designed to more completely capture realistic market conditions. The next section outlines a number of well known adaptations that attempt to address the shortcomings of the Merton model, and provides an overview of various studies detailing the relative successes and failures of these alternative models empirically.

2.4 Extensions and empirical comparisons of structural model performance

The Merton [1974] model is hailed as a ground-breaking work and despite being over thirty years old structural modelling continues to be a topic of ongoing research and discussion. Much of this research aims to verify the validity of the contingent claims analysis approach, and where possible, suggest improvements. Starting with modifications addressing Merton's most obviously restrictive and easily addressed assumptions, a number of models have since gained popularity in academic literature. By including additional input parameters and more complex logic, each model aims to be a more powerful and more accurate method of pricing corporate bonds. Certain studies suggest that these efforts have been at least moderately successful in achieving this goal.

2.4.1 Considering coupon bonds: Geske's compound options and the Merton portfolio of zeros

Geske [1977] presents the first structural model to offer an analytical formula to price coupon bonds of finite maturity, that is, bonds that pay n regularly scheduled interest deposits as well as a principal amount upon maturity. Geske represents each coupon payment as an option, whereby equity holders pay a coupon scheduled at time t_i to effectively buy an option on the assets of the company. Each coupon/option matures when the next coupon becomes due at time t_{i+1}, with this process continuing until the bond matures.

The model allows equity holders to default at any discrete coupon payment time t_i. Geske assumes that new equity is issued to make coupon payments so as to limit changes to capital structure. Default occurs when firm value has dropped so low that investors are not attracted to new equity issues.

Should equity holders default on a coupon payment, they forfeit any further claims to the firm's assets and these become the property of the creditors. Geske assumes an immediate and costless bankruptcy process - something already discussed as being unlikely in practice.

Geske develops this framework under similar assumptions to Merton [1974]. However, the Geske model relaxes the strict requirement of simple capital structure, presenting methods to price multiple classes of debt on issue within a single corporation. By reworking the Geske pricing formula it is possible to price a senior and a junior tranche of debt, assuming that strict absolute priority of debt obligations holds and equity holders receive nothing upon default.

Thus the price of a coupon bond can be calculated by considering the expected payoffs of the bond upon maturity starting at time $t_{i=n} = T$, and iteratively stepping backwards in time through each time t_{i-1} to price the option on the remaining company assets until reaching the issue date at time t_0. The price of a risky bond becomes the solution to this system of equations. The length and complexity of the Geske model precludes it from inclusion here, but the interested reader is referred to Geske [1977] for further details.

Eom et al. [2004] present an alternative approach to modelling coupon bonds, treating a coupon bond as a simple portfolio of zero coupon bonds, of face values and maturities matching the coupon payments. For any given zero coupon bond pricing formula $B(\cdot)$, simply price each coupon as a unique bond with face value based on the coupon rate c, and sum the values to approximate the price of an equivalent coupon bond, written (assuming biannual payments) as

$$B^{\text{coupon}}(T) = \sum_{t_i=1/2}^{t=T} \frac{c}{2} B^{\text{zero}}(t_i) + B^{\text{zero}}(T) \tag{2.10}$$

Merton suggests this approach when detailing his zero coupon model, however he finds it conceptually unsatisfactory. In contrast to the compound option approach of Geske, the portfolio of zeros (PoZ) approach suggests zero default correlation between each coupon payment. That is, should a firm be unable to make a coupon payment due to financial distress, the PoZ model fails to capture the increased likelihood that future coupons will also default, and so likely understates the risk of coupon bonds.

2.4.2 Early empirical analysis of structural models

Despite being widely regarded as an elegant, tractable method to price risky corporate debt, the Merton [1974] model remains largely untested until the study of Jones et al. [1984]. Jones et al. set out to establish whether the Merton model is able to accurately predict the value of corporate bonds, and test the model on bonds from a broad spectrum of investment grades.

Jones et al. ultimately select 305 bonds from 27 American firms between 1975 and 1981. They price these bonds according to the Merton model and simultaneously using a naive risk-less model that assumes no possibility of default. The comparison against the naive model in turn allows Jones et al. to examine which assumptions of Merton's model have the most impact on the predicted prices, and which of those assumptions may lead to significant errors.

Jones et al. select a bond sample from firms with simple capital structure and low debt-to-total capital ratios. By limiting the complexity of the capital structures of firms in their sample set, Jones et al. reduce the effects of any factors that might influence the market price of debt when those factors are not being explicitly reflected in the bond valuation formula. They aim to price bonds from firms that operate, as closely as reasonably possible, within the restrictions of the Merton model. They are thus able to make quantitative statements regarding the ability of Merton's model to generate accurate predictions of price, and comment on which requirements of the Merton model may be overly restrictive and/or cause errors in price predictions.

Jones et al. suggest that Merton's model suffers from a fundamental flaw in assuming a constant interest rate r. They find that this assumption leads to incorrect prices for risk-less bonds, and so is expected to be similarly flawed when applied to the more complex task of pricing risky debt.

Secondly, they find that the estimation of the volatility of the firm's assets is highly influential on the prediction accuracy as a result of the option-pricing-based methodology of the contingent claims analysis approach. Different methods of estimating asset volatility in turn affect the estimated price of the risky bonds, with no clear best approach. Finally, they find evidence that the inability of the model to incorporate taxes causes significant errors in the price estimations.

In summary, Jones et al. find that the Merton model overestimates the prices of risky corporate bonds and subsequently underestimates credit spreads. They find that the Merton and their naive model can both be configured to price investment grade debt, however the Merton model does not significantly improve on the predictions of the naive model. Merton's model appears to more accurately predict credit spreads of non-investment grade debt but the question of precisely why this occurs remains unresolved. They suggest that perhaps the increased spread predictions are the result of the higher volatility of asset values observed in riskier, lower-grade debt. Jones et al. conclude by suggesting that future research into contingent claims models include stochastic interest rates, the effects of taxation, and also consider the question of how to best estimate unknown parameters such as the volatility of the company asset value.

2.4.3 Black and Cox, and Longstaff and Schwartz consider early default and stochastic interest rates

Black and Cox [1976] study the effects of bond indenture provisions on corporate bond prices whilst also making one of the first significant attempts to model the possibility of early default. Black and Cox assume that the value of a firm's assets evolves according to the stochastic differential equation given in Equation (2.6). They propose a new condition on the bonds, the concept of 'first passage' of asset value. They assume default is triggered the first time that the firm's asset value descends below some constant threshold H.

While Black and Cox do allow for bonds to default prior to maturity, their study focuses more specifically on bond safety covenants, subordinated levels of debt and restrictions on changes to capital structure. They ignore issues like bankruptcy costs and taxation by assuming that upon default the issuing company passes into immediate and costless ownership of the bondholders.

Despite these assumptions, Black and Cox concede that costly bankruptcy and taxation are likely to be critical issues when attempting to price risky bonds, leaving their solution open to further improvement. Indeed, Eom et al. [2004] cite multiple studies that find bankruptcy is typically a lengthy, costly process, and that bond recovery rates may range greatly with reported values as low as 40% and as high as 85%.

Longstaff and Schwartz [1995] propose a model that expands on that of Black and Cox. They include the possibility of early default, incorporate interest rate risk and multiple debt seniorities, and also allow deviation from absolute priority rules. Longstaff and Schwartz model these factors based on a belief that the constant interest rates assumption is inappropriate given real market conditions, in which interest rates may vary widely depending on general economic activity. They also refer to earlier empirical studies discussing the high likelihood of deviation from absolute priority rules in the event of bankruptcy [Franks and Torous, 1989], as evidence that this is another important consideration when calculating market prices of risky bonds.

In establishing their model, Longstaff and Schwartz make several assumptions regarding market dynamics, including assuming the evolution of the company asset value as proposed by the Merton model. Earlier models assume constant rates making derivation of bond valuation formulas simpler and less computationally-intensive, yet this assumption is demonstrably false in real markets. To capture the variability of interest rates, Longstaff and Schwartz incorporate the Vasicek [1977] interest rate model into their bond pricing model. Thus, theirs is a two-factor model, which depends on both the asset and interest rate process dynamics to influence default probabilities.

The Vasicek interest rate model describes the evolution of interest rates as a mean-reverting stochastic process modelling the movement of the instantaneous risk-free interest rate r. It is defined as

$$dr_t = (\alpha - \beta r_t)\, dt + \eta dW_t^{(2)} \tag{2.11}$$

or equivalently

$$dr_t = \kappa (\theta - r_t)\, dt + \eta dW_t^{(2)} \tag{2.12}$$

where κ, θ, η are the rate of mean reversion, long-term average and volatility respectively. $W_t^{(2)}$ is a standard Brownian motion process. Based on his model of interest rate dynamics, Vasicek is able to price a zero coupon risk-free bond at time t with maturity T according to

$$B^{\mathrm{V}}(t,T) = \mathbb{E}\left[\exp\left(-\int_t^T r_s ds\right) | \mathcal{F}_t\right] \tag{2.13}$$

where $\mathcal{F}_t$ is a filtration for the interest rate process. That is, $\mathcal{F}_t$ is an increasing series defining all measurable events as the Vasicek process evolves throughout time. Longstaff and Schwartz restate this equation in the more accessible form

$$B^{\mathrm{V}}(r,T) = \exp\left(A(T) - B(T)\, r\right) \tag{2.14}$$

where

$$\begin{aligned} A(T) &= \left(\frac{\eta^2}{2\beta^2} - \frac{\alpha}{\beta}\right) T \\ &+ \left(\frac{\eta^2}{\beta^3} - \frac{\alpha}{\beta^2}\right)(\exp(-\beta T) - 1) \\ &- \left(\frac{\eta^2}{4\beta^3}\right)(\exp(-2\beta T) - 1) \\ B(T) &= \frac{1 - \exp(-\beta T)}{\beta} \end{aligned}$$

In contrast to the Black and Cox assumption that transition from default to bankruptcy is an immediate and costless event, Longstaff and Schwartz suggest that the first time t that the company assets hits the default boundary, $V_t = H$, costly corporate restructuring occurs. They avoid modelling the potentially complex bankruptcy bargaining process that occurs between equity and bond holders. They instead represent the outcome with a single parameter representing value lost due to bankruptcy proceedings. Upon default, debt holders receive some payout ωX, where ω is the percentage of the bond face value X recovered after costly bankruptcy. Furthermore, Longstaff and Schwartz note that different bond issues may attract different values of ω depending on the seniority of debt that is defaulting (senior, junior, etc), and cite prior empirical studies as evidence of this.

The Longstaff and Schwartz model thus depends on calculating the expected payouts based on costly bankruptcy and factoring the probability of default in a risk-neutral measure $\mathbb{Q}$, that is, calculating the first-passage time density of the asset process V_t through the constant default boundary level H. They suggest a solution based on an equation modelling one-factor Markov processes, presented by Fortet [1947]. Because the Longstaff and Schwartz model is actually a two-factor process in which returns depend on both stochastic interest rates and changes in firm value, the accuracy of their solution is called into question by subsequent research [Collin-Dufresne and Goldstein, 2001].

The Longstaff and Schwartz zero coupon bond pricing model is defined the recursive equation

$$B^{\mathrm{LS}}(X, r, T) = B^{\mathrm{V}}(r, T) - (1 - \omega) B^{\mathrm{V}}(r, T) Q(X, r, T) \tag{2.15}$$

where

$$
\begin{aligned}
Q(X,r,T,n) &= \sum_{i=1}^{n} q_i \\
q_1 &= \Phi(a_1) \\
q_i &= \Phi(a_i) - \sum_{j=1}^{i-1} q_j \Phi(b_{i,j}), \qquad i = 2,3...n \\
a_i &= \frac{-\ln(X) - M\left(\frac{iT}{n}, T\right)}{\sqrt{S\left(\frac{iT}{n}\right)}} \\
b_{i,j} &= \frac{M\left(\frac{jT}{n}, T\right) - M\left(\frac{iT}{n}, T\right)}{\sqrt{S\left(\frac{iT}{n}\right) - S\left(\frac{jT}{n}\right)}} \\
M(t,T) &= \left(\frac{\alpha - \rho\sigma\eta}{\beta} - \frac{\eta^2}{\beta^2} - \frac{\sigma^2}{2}\right) t \\
&+ \left(\frac{\rho\sigma\eta}{\beta^2} + \frac{\eta^2}{2\beta^3}\right) \exp(-\beta T)(\exp(\beta T) - 1) \\
&+ \left(\frac{r}{\beta} - \frac{\alpha}{\beta^2} + \frac{\eta^2}{\beta^3}\right)(1 - \exp(-\beta T)) \\
&- \left(\frac{\eta^2}{2\beta^3}\right) \exp(-\beta T)(1 - \exp(-\beta T)) \\
S(t) &= \left(\frac{\rho\sigma\eta}{\beta} + \frac{\eta^2}{\beta^2} + \sigma^2\right) t \\
&- \left(\frac{\rho\sigma\eta}{\beta^2} + \frac{2\eta^2}{\beta^3}\right)(1 - \exp(-\beta t)) \\
&+ \left(\frac{\eta^2}{2\beta^3}\right)(1 - \exp(-2\beta t))
\end{aligned}
$$

The value $\rho dt = dW_t dW_t^{(2)}$ represents the instantaneous correlation between dW_t of equation (2.6) and $dW_t^{(2)}$ of Equation (2.11). Longstaff and Schwartz note that $Q(X,r,T,n) \rightarrow Q(X,r,T)$ as $n \rightarrow \infty$ where $Q(X,r,T)$ represents the probability under a risk-neutral measure $\mathbb{Q}$ that default occurs, and that convergence representative of the limit typically occurs with $n \approx 200$.

To test their model, Longstaff and Schwartz examine industry averages for industrial, utility and railroad corporate bond yields for a range of investment grades over a 15 year period. They compare the credit spreads implied by their model to the industry averages observed in the market data. They find that the price of corporate bonds is in fact heavily influenced by changes in interest rates, and offer this as evidence that structural models should incorporate variable interest rates if they are to accurately predict the value of risky debt. Furthermore, they suggest that changes to interest rates are more responsible for credit spreads of risky bonds than changes to company asset value, at least for investment-grade issues. They find also that the strength of the effect on credit spreads caused by interest rate risk varies across different industries, most likely as a result of the difference in each industry's correlation between changes in the asset value and changes in interest rates.

Ultimately, Longstaff and Schwartz consider their model to be a successful benchmark against which future studies can be compared and contrasted. The model offers an easily implemented and tractable solution to pricing zero coupon and also coupon bonds, using the idea from Equation (2.10). It appears to capture more realistic marketplace conditions, relaxes strict enforcement of absolute priority assumptions by capturing more complex capital structures, includes variable interest rates, and also allows for early default. They note that their model is able to generate a wide variety of credit spread shapes to match observed average values, and suggest that future work should aim to test the model's ability to price individual bonds.

Finally, it is worth noting that subsequent attempts to test the Longstaff and Schwartz model have noted that its supposed strengths may in fact also be a source of weakness. Guo and Wei [1997] find that in trying to capture more realistic market conditions, the Longstaff and Schwartz model introduces several additional input parameters over the Merton model. They find that using the Longstaff and Schwartz model invokes the considerable problem of trying to accurately estimate these extra parameters, and that this estimation process is itself a potential source of significant error. Indeed, in their particular study, Guo and Wei find that the Merton model actually outperforms that of Longstaff and Schwartz in several of their tests. Thus, researchers face the challenge of not only deriving an accurate model for the pricing of risky debt, but to ensure that their process of calculating the input parameters is valid, lest an otherwise appealing model be rendered useless by an inability to accurately calibrate it.

2.4.4 Leland and Toft analyse optimal capital structure and endogenous bankruptcy

Another structural model to price risky corporate debt is given by Leland and Toft [1996] in their paper Optimal Capital Structure, Endogenous Bankruptcy, and the Term Structure of Credit Spreads. As the name suggests, this work sets out not only a method for pricing risky debt, but also studies the effects that various capital structures have on the price of that debt. Leland and Toft state their main purpose is to find both the optimal amount and maturity of debt that a firm should aim for, differing from earlier studies attempting to merely price the debt itself. If a firm has a more complete understanding of how their credit spread, and subsequently credit risk profile, changes with differing debt maturities and risk-free interest rates, they should be better able to hedge their positions in the face of varying market conditions, as well as take maximum advantage of tax deductions on interest payments.

The Leland and Toft model includes parameters to capture taxation and costly bankruptcy, and in contrast to earlier models such as Merton [1974] and Black and Cox [1976], Leland and Toft consider default to be an endogenously decided event triggered by equity holders surrendering the firm to debt holders in an attempt to maximise their own wealth. This default process occurs when the firm's asset value reaches some boundary value H, and may lead to costly restructuring or outright bankruptcy. As with Longstaff and Schwartz [1995], Leland and Toft avoid trying to capture the intermediate bargaining process, and represent the amount of firm value recovered after costly restructuring/bankruptcy proceedings as a ratio ω applied to the default boundary, such that debt holders receive an amount ωH after default is triggered.

Leland and Toft differ from the Longstaff and Schwartz model in that they return to an assumption of a constant interest rate r, in the belief that stochastic interest rates add little to the effectiveness of structural pricing models while adding largely unjustifiable complexity to the calculations. They also make the assumption that the issuing firm chooses to reissue debt into perpetuity with a constant coupon rate and principal amount each time old debt matures. This is a simplifying assumption based on earlier works by Leland [1994a] in which he presents a model for pricing perpetual debt. It is also more consistent with conventional financial theory suggesting that firms target an optimal level of debt, trading off the tax benefits of debt against the cost of debt in order to maximise firm value.

Leland and Toft price risky debt according to the equation

$$B^{\mathrm{LT}}(0,T) = \frac{c}{r} + \left(1 - \frac{c}{r}\right)\left(\frac{1 - e^{-rT}}{rT} - I(T)\right) + \left(\omega H - \frac{c}{r}\right) J(T) \tag{2.16}$$

where

$$\begin{aligned}
I(T) &= \frac{1}{rT}\left(G(T) - e^{-rT}F(T)\right)\\
J(T) &= \frac{1}{z\sigma\sqrt{T}}\left[-e^{(z-a)b}\Phi\left(q_-(T)\right)q_-(T) + e^{-(z+a)b}\Phi\left(q_+(T)\right)q_+(T)\right]\\
G(T) &= e^{(z-a)b}\Phi\left(q_-(T)\right) + e^{-(z+a)b}\Phi\left(q_+(T)\right)\\
F(T) &= G(T)\,|_{z=a}\\
a &= \frac{r-\delta}{\sigma^2} - \frac{1}{2}\\
b &= \ln\left(\frac{V_0}{H}\right)\\
z &= \left(a^2 + \frac{2r}{\sigma^2}\right)^{1/2}\\
q_{\mp}(t) &= \frac{-b \mp z\sigma^2 t}{\sigma\sqrt{t}}
\end{aligned}$$

The lengthy derivation of H and an explanation of the different forms it takes under various conditions can be found in equations (11) and (13) of Leland and Toft [1996].

As stated above, Leland and Toft focus less on empirical testing of the validity of their model, and instead devote their efforts to analysing the outcomes predicted by their model in terms of credit spread shapes and shifts. They find that the model is able to generate a wide variety of credit spread shapes, manage to derive an optimal bankruptcy trigger H, and also make suggestions on the types of debt that different firms should optimally issue depending on their circumstances. However, while they cite some empirical studies that generally support their results, they present no specific data testing their model at either an aggregate or individual bond issue level. As presented, the paper offers an interesting starting point for several further areas of discussion, but the question of the model's applicability in practical circumstances is left unanswered.

2.4.5 Briys and de Varenne remedy default barrier defects

Shortly after the work of Leland and Toft [1996], Briys and de Varenne [1997] propose another structural model with the intent to study generated credit spread patterns for comparison with market data. Similarly to Leland and Toft, they avoid extensive testing, offering only cursory evidence that the model is supported by actual market observations of bond prices. Instead they focus on discussion of the shapes of the credit spread term structures that the model is able to generate.

In contrast to the Leland and Toft assumption of permanent re-issuance of debt however, Briys and de Varenne opt to return to the approach favoured by Longstaff and Schwartz [1995], with some minor adjustments. The Briys and de Varenne model incorporates early default, variable cost of bankruptcy and stochastic interest rates, and also includes parameters reflecting the write-down on assets paid out to debt holders in the event of default. That is, the model specifically captures the possibility of deviation from the absolute-priority rules, and also differentiates between cases where the default occurs before maturity as well as specifically at maturity. Briys and de Varenne make the realisation that the pricing formulas in earlier works, such as that of Longstaff and Schwartz, often contain a defect that allows the calculated bankruptcy payout to bondholders to potentially exceed the actual value of the company's assets at default. As a solution they propose an evolving default barrier $H(t)$, defined by discounting some ratio γ of the face value X, where $0 \leq \gamma \leq 1$. Should the company assets sink below this level, default is triggered and immediate costly bankruptcy proceedings commence.

While the assumptions are similar, the mathematics behind the Briys and de Varenne model differs slightly to the earlier Longstaff and Schwartz model, with the evolution of company assets modelled by the equation

$$dV_t = r_t V_t dt + \sigma V_t \left[\rho dW_t + \sqrt{1-\rho^2} dW_t^{(2)} \right] \tag{2.17}$$

where the variable definitions hold from previous sections. The model allows stochastic interest rates described by

$$dr_t = a(t) \left[b(t) - r_t \right] dt + \sigma(t) dW_t^{(2)} \tag{2.18}$$

where $a(t)$, $b(t)$ and $\sigma(t)$ are deterministic functions of time, and results in the Vasicek

[1977] model when each function is constant. Briys and de Varenne model risky debt prices according to

$$\begin{aligned} B^{\mathrm{BdV}}(0,T) = X \cdot B^{\mathrm{V}}(r,T) \cdot \Bigg[1 - P_E(l_0,1) + P_E\left(q_0, \frac{l_0}{q_0}\right) \\ - (1-f_1)\, l_0 \left(\Phi(-d_3) + \frac{\Phi(-d_4)}{q_0} \right) \\ - (1-f_2)\, l_0 \left(\Phi(d_3) - \Phi(d_1) + \frac{\Phi(d_4) - \Phi(d_6)}{q_0} \right) \Bigg] \end{aligned} \tag{2.19}$$

where

$$\begin{aligned} d_1 &= \frac{\ln(l_0) + \Sigma(T)/2}{\sqrt{\Sigma(T)}} = d_2 + \sqrt{\Sigma(T)} \\ d_3 &= \frac{\ln(q_0) + \Sigma(T)/2}{\sqrt{\Sigma(T)}} = d_4 + \sqrt{\Sigma(T)} \\ d_5 &= \frac{\ln(q_0^2/l_0) + \Sigma(T)/2}{\sqrt{\Sigma(T)}} = d_6 + \sqrt{\Sigma(T)} \\ \Sigma(T) &= \int_0^T \left[(\rho\sigma + \varphi(t,T))^2 + \left(1-\rho^2\right)\sigma^2 \right] dt \\ \varphi(t,T) &= \sigma(t) \int_t^T \exp\left(-\int_t^u a(s)\, ds \right) du \\ l_0 &= \frac{V_0}{X \cdot B^{\mathrm{V}}(r,T)} \\ q_0 &= \frac{V_0}{\gamma X \cdot B^{\mathrm{V}}(r,T)} \\ P_E(l_0,1) &= -l_0 \Phi(-d_1) + \Phi(-d_2) \\ P_E\left(q_0, \frac{l_0}{q_0}\right) &= -q_0 \Phi(-d_5) + \frac{l_0}{q_0} \Phi(-d_6) \\ H(t) &= \gamma X \cdot B^{\mathrm{V}}(r,T) \end{aligned}$$

where q_0 can be considered a debt-assets ratio, with $q_0 = 1$ triggering bankruptcy [Briys and de Varenne, 1997].

Also of note is the fact that the Briys and de Varenne model encapsulates the original Merton [1974] model. When $\gamma = 0$ Merton's assumptions of no early default apply, and with constant interest rates and strict observance of the absolute priority rules of asset allocation the last two terms of the bond pricing formula disappear. Finally, the original Briys and de Varenne model prices zero coupon bonds only, but can readily be adapted to price coupon bonds using the portfolio-of-zeros approach of Equation (2.10).

As mentioned previously, the Briys and de Varenne paper offers little in the way of empirical testing and instead merely examines the effects that changes to the various input parameters have on the credit spread structure. They find that by assigning parameter values roughly in line with averages witnessed in the marketplace, the model can generate a range of credit spread shapes that match those witnessed in empirical studies. For example they generate various graphs of firms with different leverage after having chosen $r = 0.05$, $\sigma = 0.2$ and so on. Briys and de Varenne conclude by offering suggestions for future research including adapting to price coupon-paying bonds, capturing more complex capital structures, and of course establishing how well the bond prices as predicted by the model actually correspond to those observed in the marketplace.

2.4.6 Collin-Dufresne and Goldstein model target leverage ratios

Another model that extends the work of Longstaff and Schwartz [1995] is that of Collin-Dufresne and Goldstein [2001]. Collin-Dufresne and Goldstein assume that firm asset value evolves according to Equation (2.6) and stochastic interest rates evolve according to Equation (2.12), and allow for costly early bankruptcy triggered immediately by default. However, instead of setting the Longstaff and Schwartz value H as a constant default barrier, they implement a dynamic default boundary and leverage ratio which are mean-reverting. This central concept of the Collin-Dufresne and Goldstein model runs contrary to the condition often applied to earlier structural models that assumes firms do not alter their capital structure. Instead they suggest that in response to changes in the value of their assets, firms routinely alter their level of debt to maintain a target leverage ratio. They cite studies supporting both the claim that firms across a given industry seem to share common target leverage ratios, and that firms aim to keep this ratio relatively stationary in order to maximise firm value by taking advantage of tax incentives on debt.

With this in mind, Collin-Dufresne and Goldstein specify a model to capture this mean-reverting leverage ratio, and compare its performance against that of its predecessor, the Longstaff and Schwartz model. They discover that assumptions made by Longstaff and Schwartz to calculate the probability of default are inappropriate. The formula presented by Fortet [1947] and used by Longstaff and Schwartz to calculate probability of default only correctly calculates first-passage probability for one-factor systems. The Longstaff and Schwartz model is a two-factor model affected by both interest rate and asset dynamics, and the use of Fortet's formula in their debt pricing Equation (2.15) results in merely an approximation of their model, not the explicit solution that they believe.

To implement their own two-factor model Collin-Dufresne and Goldstein derive an exact equation to calculate the probability two-factor first-passage, offering this as an improvement on the original Fortet version. Indeed, Collin-Dufresne and Goldstein redevelop the Longstaff and Schwartz model with the improved values and use this as a benchmark against which they test the performance of their own model.

For brevity neither the Collin-Dufresne and Goldstein structural model nor their improved version of the Longstaff and Schwartz model is included here. Suffice to say the added features increase the model complexity considerably. The interested reader is referred to the original paper [Collin-Dufresne and Goldstein, 2001] for further details.

Collin-Dufresne and Goldstein provide no empirical test results comparing output from their pricing model to market bond prices. Instead they focus on examining the credit spread shapes generated by setting input parameters to what they consider 'reasonable parameter choice' values based on unreported research. For example, they consider different leverage ratios for investment and speculative grade issues to be approximately 0.36 and 0.65 respectively. Furthermore, they set $r = 0.06$, a recovery rate of bond face value $\omega = 0.44$, and coupon recovery rate $\omega_c = 0$ based on evidence that outstanding coupon payments are rarely recovered in the event of bankruptcy [Helwege and Turner, 1999]. Applying these generic values, Collin-Dufresne and Goldstein report that spreads generated by their model more accurately match those shapes observed in the market, particularly for longer maturity debt, when compared with even their improved Longstaff and Schwartz model.

Ultimately however, Collin-Dufresne and Goldstein do not conclusively establish whether either the improved Longstaff and Schwartz or their own mean-reverting leverage model can accurately predict market prices for risky corporate bonds.

2.4.7 An empirical comparison of popular models

Eom et al. [2004] present a detailed study of five structural bond pricing models, testing their accuracy by pricing 182 bond issues and comparing the predicted values against the observed market prices. Their study tests a portfolio-of-zeros version of the Merton [1974] model, the Geske [1977] option-on-options model, the Longstaff and Schwartz [1995] early default model, the Leland and Toft [1996] optimal capital structure model, and the Collin-Dufresne and Goldstein [2001] mean-reverting leverage model. These models are representative of the various approaches taken to structural modelling since Merton's original implementation. By testing and comparing each model across the same data set, Eom et al. offer fresh insight into the effectiveness of each approach and make suggestions for promising future research. They also set out to verify previous suggestions that structural models predict bond yields considerably lower than those actually observed in markets [Jones et al., 1984, Franks and Torous, 1989].

Eom et al. choose a sample of bonds issued between 1986 and 1997, starting with 8700 potential issues of medium to long term coupon bonds issued by publicly listed firms with simple capital structures. They reject firms with complicating characteristics including those with multiple bonds on issue, multiple debt seniority levels, and bonds with covenants such as callability and sinking fund provisions, amongst others. They restrict the complexity of the sample firms' capital structure so as to first test whether each model can accurately predict bond prices operating under the original simplifying assumptions of the models, before tackling the problem of more complex scenarios.

Eom et al. discuss the methods they use to implement each model and highlight the aforementioned issue of input parameter estimation. For example, they perform their experiments with no less than seven different estimates of asset volatility and generate parameters for two different interest rate models [Nelson and Siegel, 1987, Vasicek, 1977]. They discuss multiple methods of estimating leverage values, and cite several studies suggesting various investment recovery rates before settling on a value of $\omega = 51.31\%$ of the bond face value for those models that include a recovery rate parameter [Longstaff and Schwartz, 1995, Leland and Toft, 1996, Collin-Dufresne and Goldstein, 2001]. That is, they assume that investors lose 48.69% of the face value of their investment due to costly bankruptcy proceedings should default occur.

Eom et al. set the default boundary H based on the work of Choi and Wong [2006], who find that the median default barrier of a firm is triggered as asset value drops to 73.8% of outstanding liabilities. Depending on the particular model, the pricing formulas may state the default boundary threshold as an explicit cutoff amount [Black and Cox, 1976, Longstaff and Schwartz, 1995], or as a ratio of the evolving asset value [Leland and Toft, 1996, Briys and de Varenne, 1997].

Eom et al. note that the models are particularly sensitive to variations when estimating their input parameters. Previous studies [Jones et al., 1984, Franks and Torous, 1989] find that structural models implemented with realistic parameter values are often incapable of generating credit spreads as wide as those witnessed in the market. Contrary to these earlier studies, Eom et al. do not observe this to be the case, with some large credit spreads occurring in their results. Unfortunately though, they find the accuracy of the model predictions varies wildly with a tendency for wide dispersion (both under and over estimation) of predicted credit spreads when taken across the entire sample set. For example, the Leland and Toft model predicts a credit spread of zero and 5096 basis points for two bonds whose actual values were 488 and 465 basis points respectively. They credit the tendency of Leland and Toft to both radically under and over-estimate credit spreads to the importance this model places on the coupon payments. As a result when the coupon rate is unusually high or low it can disproportionately affect the predictions. Eom et al. find that the other models tend to suffer similar inaccuracies, particularly when trying to price bonds of shorter maturities.

The extreme errors in the prices predicted by the various models prompt Eom et al. to look for systemic sources of error. They perform statistical analysis on a range of potential influences to establish the degree to which each factor affects the accuracy of the pricing models. Factors tested include firm size, coupon rate, credit rating, number of bonds on issue and correlation to interest rates. They are able to reject a small number of potential factors, such as firm size, as being relatively inconsequential across all models. At the same time they suggest certain factors such as maturity and asset volatility as being significant contributors to model accuracy. The specific results of their study are too lengthy for a full discussion here, the interested reader is referred to Eom et al. [2004] for further details.

Eom et al.'s study yields several interesting results. They note that the structural models tested do not predict individual bond prices with any significant degree of accuracy. This is despite the considerable amount of effort invested in improving their versatility and expanding beyond the simplifying assumptions of the Merton model. Additionally, they note that pricing errors vary greatly in sign and magnitude so that while the average performance of any given model might be encouraging, they are typically unsuitable for pricing individual bond issues. Finally, and perhaps most interestingly, Eom et al. note that as the number of input parameters increases so does the complexity, difficulty and importance of correctly estimating these parameters. Further research is needed to ensure that both the assumptions of the models and the methods of estimating their parameters are appropriately accounted for when developing tractable solutions to structural modelling of risky bond prices.

2.4.8 Better model performance through improved parameter estimation

Correctly estimating the parameter values when implementing structural models remains a serious and unresolved problem. Several inputs into the pricing formulas are not directly observable and so their values must be estimated. For example, the instantaneous risk-free interest rate parameter represents the rate of interest for a loan for an infinitesimally short period of time, which is obviously impossible to observe in practise. Similarly, the market value of debt, key to knowing the 'true' value of firm assets and consequently asset dynamics, is typically unknown since not all debts are fully traded in the market. The market value is therefore often proxied by book value of debt, which remains constant between reporting periods and results in a distorted perception of debt value [Sweeney et al., 1997].

As noted previously, Eom et al. [2004] test the prediction performance of several structural bond pricing models on individual bonds. In presenting their results, Eom et al. note that a major source of error when implementing structural models is the difficulty in appropriately estimating the value and volatility of a firm's assets. Asset value and dynamics parameters are particularly important for many structural bond models, and so the problem of how to correctly estimate them is of critical importance. Indeed, in one study by Ericsson and Reneby [2005], the authors find that pricing errors observed with structural models may occur largely as a result of using overly simple asset valuation techniques to establish asset volatility.

Li and Wong [2008] perform research that logically extends that of Eom et al.. They select the Merton [1974], Longstaff and Schwartz [1995], Briys and de Varenne [1997] and Leland and Toft [1996] models, and apply them to a selection of bonds from firms using similar selection criteria as that applied by Eom et al.. Li and Wong select firms based on criteria including simple capital structure and having few bonds on issue. Their sample data set ultimately contains 807 bonds from a total of 171 different firms, with a wide range of maturities, coupon rates, and credit ratings.

Ericsson and Reneby suggest maximum likelihood estimation (MLE) as an alternative method with which practitioners may more accurately estimate various model parameters and thus improve the performance of those models. In response to this suggestion Li and Wong study the performance of each model using several different methods to estimate the dynamics of the firm asset value process. They test a pure proxy of asset value, the volatility restriction (VR) method of Ronn and Verma [1986], the mixed proxy approach of Eom et al., and an MLE method such as that favoured by Ericsson and Reneby.

The pure proxy approach involves summing the market value of equity and the book value of debt, and volatility is affected solely by fluctuations in equity values. The VR method involves solving simultaneous equations to calculate asset value and volatility, based on observations of equity prices. The mixed proxy approach as applied by Eom et al. takes the proxy value of a firm to be the market firm value, and calculates the volatility of the firm by extending elements of the VR approach.

Finally, the maximum likelihood approach takes market prices of equity and debt and, as opposed to the pure proxy approach of letting equity explicitly represent the firm asset value, assumes instead that equity is an option on firm assets. The implied asset value is then calculated depending on the particular type of option used to model the equity. Given an option pricing model and related transitional density function of the asset evolution, it is sometimes possible to calculate an analytical likelihood function to match. It is the task of the researcher to find the drift and volatility values that would most likely lead to the observed asset value transitions, by maximising the value of the likelihood function. Once the most likely values for each parameter are established they are fed into the various bond pricing models.

After establishing the asset dynamics using each of the aforementioned methods, Li and Wong find the MLE approach to be by far the superior method of parameter estimation. In comparison, they find the proxy methods to be upwardly biased and inappropriate, consistently resulting in underestimation of credit spreads with each of the models implemented. While in some instances the VR method produces satisfactory results, it has the potential to produce either no solution or, perhaps more troublingly, multiple solutions for each parameter value, leaving open the question of which value is then optimal. By contrast the MLE approach should always produce a unique and optimal estimate for each parameter estimated. Importantly, Li and Wong find that the MLE technique appears to significantly improve the accuracy of each structural model in their study. They suggest that the MLE parameter estimation technique is more theoretically consistent than other methods, and produces better results when applied to bond price modelling.

2.5 Summary

The preceding chapters outline the concepts of contingent claims analysis, and detail the earliest attempts to use this approach to systematically price risky corporate debt, as outlined by Merton [1974]. In reviewing that pricing model, several of its simplifying assumptions are examined and critiqued, and examples given where those same assumptions are do not hold or appear overly restrictive. The results of several studies [Franks and Torous, 1989, Jones et al., 1984] suggest that Merton's initial structural pricing model requires an unrealistic set of assumptions, and that in order to accurately price risky debt, several of the more restrictive assumptions should be relaxed.

Each subsequent model presented extends upon the ideas of the original Merton model in order to better capture the conditions of real markets, and tries to generate credit spreads more in line with those witnessed in those markets.

Structural models have evolved to include the modelling of coupon bonds [Geske, 1977, Eom et al., 2004], early default [Geske, 1977, Black and Cox, 1976, Longstaff and Schwartz, 1995], the inclusion of stochastic interest rates [Longstaff and Schwartz, 1995, Briys and de Varenne, 1997, Collin-Dufresne and Goldstein, 2001], and optimal capital structure [Collin-Dufresne and Goldstein, 2001, Leland and Toft, 1996], amongst other modifications. While each of these subsequent generations of models have claimed to generate credit spreads more representative of those observed in actual markets, we highlight their tendency to avoid presenting precise empirical data to support such claims.

Finally, we present the outcomes of several recent studies which suggest that the task of pricing risky debt may be significantly more complex than originally proposed. Not only do structural models need to incorporate sufficient parameters to capture both the broader economic conditions as well as individual financial circumstances of a firm as it issues risky debt, but there remains the task of accurately estimating these same parameters before predicting prices. Increasing evidence suggests that the problem of estimating model parameters remains a serious impediment to the design and implementation of a practical, tractable, and above all, accurate model to predict the price of risky corporate bond prices and credit yield spreads. Future research must ultimately resolve the question of whether the previous problems with structural models can be overcome by improved parameter estimation techniques, or whether they are still subject to some fundamental flaw which prevents them from being applicable to the task of pricing risky corporate bonds.

Chapter 3

Modelling and theory

Thi study aims to determine whether the performance of two simple bond pricing models is improved by the use of the MLE technique as compared to a pure proxy approach. We test whether the resulting model performance is sufficiently accurate to allow their use in real markets.

3.1 Selection of bond pricing models

We follow the examples of Li and Wong [2008], Guo and Wei [1997], wherein the authors note that the Merton [1974] and Longstaff and Schwartz [1995] models are two of the simplest and most commonly studied structural bond pricing models. For simplicity, we choose to price coupon bonds with the portfolio-of-zeros approach as per Equation (2.10) for each of our selected bond pricing models.

3.1.1 Modified Merton model

To price coupon bonds using an extended Merton [1974] model we combine Equations (2.10) and (2.7) based on the description given by Eom et al. [2004],

$$\begin{aligned} B^{MC}\left(V_0, X, T\right) = & \sum_{i=1}^{2T} B^{\mathrm{V}}\left(r, T_i\right) \mathbb{E}^{\mathbb{Q}}\left[\left(\frac{c}{2}\right) I_{\left\{V_{T_i} \geq H\right\}} + \min\left(\frac{(1-\omega)\,c}{2}, V_{T_i}\right) I_{\left\{V_{T_i} < H\right\}}\right] \qquad (3.1) \\ & + B^{\mathrm{V}}\left(r, T\right) \mathbb{E}^{\mathbb{Q}}\left[I_{\{V_T \geq H\}} + \min\left((1-\omega), V_T\right) I_{\{V_T < H\}}\right], \end{aligned}$$

where

$$
\begin{aligned}
\mathbb{E}^{\mathbb{Q}}\left[I_{\left\{V_{T_i} \geq H\right\}}\right] &= \Phi\left(d_2\left(H,t\right)\right) \\
\mathbb{E}^{\mathbb{Q}}\left[I_{\{V_t < H\}} \min\left(\psi, V_t\right)\right] &= \frac{V_0}{B^{\mathrm{V}}\left(r,t\right)}\Phi\left(-d_1\left(\psi,t\right)\right) + \psi\left[\Phi\left(d_2\left(\psi,t\right)\right) - \Phi\left(d_2\left(H,t\right)\right)\right] \\
d_1\left(x,t\right) &= \frac{\ln\left(V_0/\left(x \cdot B^{\mathrm{V}}\left(r,t\right)\right)\right) + \frac{1}{2}\sigma^2 t}{\sigma\sqrt{t}} \\
d_2\left(x,t\right) &= d_1\left(x,t\right) - \sigma\sqrt{t}
\end{aligned}
$$

All previous definitions of the input parameters apply.

3.1.2 Longstaff and Schwartz

We use the portfolio-of-zeros approach of Equation (2.10) with the Longstaff and Schwartz [1995] bond pricing formula to price coupon bonds,

$$
\begin{aligned}
B^{\mathrm{LSC}}\left(X,r,T\right) = \frac{c}{2}\sum_{i=1}^{2T}\left[B^{\mathrm{V}}\left(r,T_i\right) - \left(1-\omega\right)B^{\mathrm{V}}\left(r,T\right)Q\left(X,r,T_i\right)\right] \\
+ B^{\mathrm{V}}\left(r,T\right) - \left(1-\omega\right)B^{\mathrm{V}}\left(r,T\right)Q\left(X,r,T\right)
\end{aligned}
\tag{3.2}
$$

where all parameters are defined previously in Equation (2.15).

3.2 Selection of interest rate models

The two selected bond pricing models make very different assumptions about the behaviour of the instantaneous risk-free interest rate r. Longstaff and Schwartz [1995] assume that r is a stochastic process described by the Vasicek [1977] mean-reverting model.

By comparison Merton [1974] assumes a constant interest rate but offers frustratingly little advice in regards to how to best derive an appropriate value, nor does Merton suggest any particular interest rate model to use.

3.2.1 Vasicek's model of stochastic interest rates

The bond pricing models of Longstaff and Schwartz [1995], Briys and de Varenne [1997], Collin-Dufresne and Goldstein [2001] all assume the Vasicek [1977] interest rate model. Given that we implement and test the Longstaff and Schwartz model, modelling the interest rate according to Vasicek is an integral part of this study.

The Vasicek model as defined in Equation (2.12) is a simple and easy to implement model of the dynamics of the risk-free interest rate. Despite its popularity, the model contains several theoretically unappealing features. For example, the model assumes that the values of κ, θ and η are constant throughout time, and also allows interest rate values to become negative. However, Longstaff and Schwartz note that for realistic parameter values the probability that r will ever become negative is small. It is also important to consider that whilst more complex models may more accurately capture interest rate dynamics, their increased complexity makes it difficult to find analytical solutions when incorporated into bond pricing models.

Previous empirical tests of the Merton [1974] model [Eom et al., 2004, Li and Wong, 2008] implement the interest rate models of both Nelson and Siegel [1987] and Vasicek and compare the effect on the bond price predictions.

Unreported test results conducted for this study suggested little difference in the performance of the Merton model whether using the interest rate model of Nelson and Siegel or Vasicek. The decision was made to use a single interest rate model for all testing, in this case that of Vasicek. It is our intension to keep conditions as similar as possible between the bond pricing models under examination so as to offer the clearest possible comparison of their relative performance.

We implement the Vasicek model and use the parameters κ, θ, r and η as inputs into the Longstaff and Schwartz model. For the Merton model only the value of the instantaneous rate r is used, the other Vasicek parameters serve no purpose and are disregarded.

3.3 Maximum likelihood estimation and application to parameter estimation

Early empirical tests of structural bond pricing models typically relied on simple proxy methods to estimate the input parameters [Jones et al., 1984, Eom et al., 2004]. More recent studies suggest that the weak performance of structural models may not be due to some critical flaw in the structural approach. Instead the observed errors be largely the result of the methods used to estimate the model input parameters, in particular the volatility of firm asset value [Ericsson and Reneby, 2005, Li and Wong, 2008]. It has been suggested that the parameter estimates, and subsequently bond model performance, can be improved using the method of maximum likelihood estimation.

3.3.1 An introduction to maximum likelihood estimation

A common problem in statistical analysis is to estimate the probability of some experimental outcome $x = (x_1, x_2, ..., x_n)$ given an underlying probability distribution $\mathbb{X}$ and known parameter vector $\theta = (\theta_1, \theta_2, ..., \theta_m)$. We let $f(x|\theta)$ represent the probability density function that describes the probability that observations x occur under this probability distribution [Myung, 2003].

If all observations x_i are independent and identically distributed according to $\mathbb{X}$ then standard probability theory states that the probability of observing the vector x is simply the cumulative multiplicative probability of each observation, also called the *joint probability density function*. Expressed more precisely:

$$\begin{aligned} \mathbb{P}\left(x = (x_1, ..., x_n) \,|\, \theta = (\theta_1, ..., \theta_m)\right) &= f(x_1|\theta) \times f(x_2|\theta) \times ... \times f(x_n|\theta) \\ &= \prod_{i=1}^{n} f(x_i|\theta) \end{aligned} \tag{3.3}$$

Maximum likelihood estimation is a statistical technique that works to estimate probability distribution parameters based on observed outcomes. Start with a given sample of experimental observations $x = (x_1, x_2, ..., x_n)$ that are believed to follow some known underlying probability distribution $\mathbb{X}$ but with unknown parameter vector θ. The method of maximum likelihood estimation produces values $\hat{\theta}$ for the vector θ that, for the particular probability distribution, are 'most likely' to have generated the observed outcomes. For any given values of θ, there will be an associated probability of having observed the values x, as stated in Equation (3.3).

We define a likelihood function L such that the roles of our data and parameter vectors are reversed

$$L(\theta|x) = f(x|\theta) \tag{3.4}$$

and choose $\theta = \hat{\theta}$ such that $\hat{\theta}$ makes the value of L maximal [Myung, 2003]. This vector $\hat{\theta}$ is the maximum likelihood estimate.

The maximum likelihood estimator values are often taken over $\ln(L(\theta|x))$ because the functions $L(\theta|x)$ and $\ln(L(\theta|x))$ are maximal for the same values of θ but the computation of the log-likelihood is often more computationally convenient.

Maximum likelihood estimation is theoretically appealing as it produces efficient parameter estimates. That is, MLE is able to generate good parameter estimates with smaller error than many alternative techniques. For large samples, the asymptotic properties of maximum likelihood estimates include Normal distribution, small variance and lack of bias, all desirable properties when performing statistical data analysis. It is important to note that the values of $\hat{\theta}$ generated by MLE are not necessarily those that *actually* generated the observed data, merely those *most likely* to have caused the observed outcomes.

While not all problems have analytical solutions to calculate their MLE estimates, in such cases numerical methods can often be used. Indeed, in future sections we discuss the iterative numerical routines implemented to search for solutions to the particular problem of estimating asset parameter dynamics.

3.3.2 Application to the Merton model

One of the first widely acknowledged applications of MLE to price financial instruments is the work of Duan [1995]. Duan estimates the values of unknown process parameters for the Vasicek [1977] interest rate model and also firm asset dynamics based on prices of derivative contracts. Duan notes that the asset values for a firm cannot be directly observed, but that equity values may be considered as options on assets and that their market value is directly observable.

The mapping between equity prices as an option on assets and asset value is one-to-one when all other parameters are known. Observed equity prices can therefore be considered as transformed asset pricing data, mapped via an option pricing model. By inverting the relevant option pricing formula, one can use observed market values to calculate an implied value of the unobservable underlying asset. By solving this system of equations for a series of equity price observations Duan produces a series of implied asset values at each observation time.

The key realisation of Duan relies on the assumption of log-normally distributed asset returns of Black and Scholes [1973]

$$\ln\left(\frac{V_{t+1}}{V_t}\right) \sim N\left(\mu, \sigma^2\right) \tag{3.5}$$

The probability of witnessing any change in value of $\ln(V_t) = v_t$ between times t and $t+1$ is given by the one-period transitional density function

$$g(v_i|v_{i-1}) = \frac{1}{\sigma\sqrt{2\pi(t_i - t_{i-1})}} \exp\left(-\frac{\left[v_i - v_{i-1} - \left(\mu - \frac{1}{2}\sigma^2\right)(t_i - t_{i-1})\right]^2}{2\sigma^2(t_i - t_{i-1})}\right) \tag{3.6}$$

Duan showed that the likelihood function for the observed equity returns S_t at time t, with $S_t \equiv S(t)$, is

$$L(\mu, \sigma) = \sum_{i=2}^{n} \left[\ln\left(f\left(S_i|S_{i-1}, \mu, \sigma\right)\right)\right] \tag{3.7}$$

where $f(\cdot)$ is the probability density function of the transitions of share price process S. Any asset-to-equity transformation via an option pricing formula $\Psi(t_i, V_t, \sigma)$ can be approximated by the Chain Rule. Duan applies the change of variables technique to integrate the derivative to show that

$$f(S_i|S_{i-1}, \mu, \sigma) = g(v_i|v_{i-1}, \mu, \sigma) \times \left[V_i \cdot \Delta(\Psi)\,|_{V=V_i}\right]^{-1} \tag{3.8}$$

where $\Delta(\Psi)$ is the first derivative of the option pricing formula Ψ with respect to the asset value V_t .

As a result, the earlier log-likelihood equity function described in equation (3.7) can be restated

$$\begin{aligned} L(\mu, \sigma) &= \sum_{i=2}^{n} \left[\ln\left(f\left(S_i|S_{i-1}, \mu, \sigma\right)\right)\right] \\ &= \sum_{i=2}^{n} \left[\ln\left(g\left(v_i|v_{i-1}\right)\right) - \ln\left(V_i \cdot \Delta(\Psi)\,|v = v_i\right)\right] \end{aligned} \tag{3.9}$$

Depending on the option used to model equity, calculate the equation for $\Delta(\Psi)$. For the Black and Scholes European call option formula which underpins the Merton [1974] structural bond pricing model it is well known that

$$\Delta(\Psi) = \frac{dC^{BS}}{dV_0} = \Phi(d_1) \tag{3.10}$$

with d_1 defined in Equation (2.4). A complete derivation of Δ follows. Recall that

$$\Psi = C^{BS}\left(V_0, X, T\right) = V_0 \Phi\left(d_1\right) - X e^{-rT} \Phi\left(d_2\right)$$

where for mathematical convenience we now define

$$d_1 = \frac{\ln\left(\frac{V_0}{X}\right) + rT}{\sigma\sqrt{T}} + \frac{\sigma\sqrt{T}}{2} \tag{3.11}$$

$$d_2 = \frac{\ln\left(\frac{V_0}{X}\right) + rT}{\sigma\sqrt{T}} - \frac{\sigma\sqrt{T}}{2} = d_1 - \sigma\sqrt{T} \tag{3.12}$$

Take the first derivative with respect to V_t and see that the delta of the call option Ψ is

$$\Delta\left(\Psi\right) = \Phi\left(d_1\right) + V_0 \frac{\partial}{\partial V_0}\Phi\left(d_1\right) - X e^{-rT}\frac{\partial}{\partial V_0}\Phi\left(d_2\right) \tag{3.13}$$

Apply the chain rule to see

$$\frac{\partial}{\partial V_0}\Phi\left(d_1\right) = \phi\left(d_1\right)\frac{\partial d_1}{\partial V_0}$$

and

$$\frac{\partial}{\partial V_0}\Phi\left(d_2\right) = \phi\left(d_2\right)\frac{\partial d_2}{\partial V_0}$$

We now set out to show that the last two terms of Equation (3.13) cancel each other out, that is,

$$\begin{aligned} V_0 \frac{\partial}{\partial V_0}\Phi\left(d_1\right) &= X e^{-rT}\frac{\partial}{\partial V_0}\Phi\left(d_2\right) \qquad (3.14) \\ \Rightarrow V_0 \phi\left(d_1\right)\frac{\partial d_1}{\partial V_0} &= X e^{-rT}\phi\left(d_2\right)\frac{\partial d_2}{\partial V_0} \\ \Rightarrow V_0 \phi\left(d_1\right) &= X e^{-rT}\phi\left(d_2\right) \end{aligned}$$

since we observe that

$$\begin{aligned}\frac{\partial}{\partial V_0}\left[\frac{\ln\left(\frac{V_0}{X}\right)+rT}{\sigma\sqrt{T}}+\frac{\sigma\sqrt{T}}{2}\right] &= \frac{\partial}{\partial V_0}\left[\frac{\ln\left(\frac{V_0}{X}\right)+rT}{\sigma\sqrt{T}}-\frac{\sigma\sqrt{T}}{2}\right]\\ \Rightarrow \frac{\partial d_1}{\partial V_0} &= \frac{\partial d_2}{\partial V_0} = \frac{1}{\sigma V_0\sqrt{T}}\end{aligned}$$

We recall that the cumulative standard Normal distribution $\Phi(z)$ is given by

$$\Phi(z) = \frac{1}{\sqrt{2\pi}}\int_{-\infty}^{z}\exp\left(-\frac{x^2}{2}\right)dx$$

It follows that

$$\phi(z) = \frac{1}{\sqrt{2\pi}}\exp\left(-\frac{x^2}{2}\right)$$

and so

$$\begin{aligned}\phi(d_1) &= \frac{1}{\sqrt{2\pi}}\exp\left(-\frac{d_1^2}{2}\right),\\ \phi(d_2) &= \frac{1}{\sqrt{2\pi}}\exp\left(-\frac{d_2^2}{2}\right)\end{aligned}$$

We must therefore show that

$$\begin{aligned}V_0\frac{1}{\sqrt{2\pi}}\exp\left(-\frac{d_1^2}{2}\right)\left(\frac{1}{\sigma V_0\sqrt{T}}\right) &= Xe^{-rT}\frac{1}{\sqrt{2\pi}}\exp\left(-\frac{d_2^2}{2}\right)\left(\frac{1}{\sigma V_0\sqrt{T}}\right)\\ \Rightarrow V_0\exp\left(-\frac{d_1^2}{2}\right) &= Xe^{-rT}\exp\left(-\frac{d_2^2}{2}\right)\end{aligned}$$

We now rearrange and cancel terms as follows

$$\begin{aligned}
V_0 \exp\left(-\frac{d_1^2}{2}\right) &= Xe^{-rT}\exp\left(-\frac{d_2^2}{2}\right) \qquad (3.15)\\
&= Xe^{-rT}\exp\left(-\frac{1}{2}\left[d_1-\sigma\sqrt{T}\right]^2\right)\\
&= Xe^{-rT}\exp\left(-\frac{1}{2}\left[d_1^2-2d_1\sigma\sqrt{T}+\sigma^2T\right]\right)\\
&= Xe^{-rT}\exp\left(-\frac{1}{2}d_1^2+d_1\sigma\sqrt{T}-\frac{1}{2}\sigma^2T\right)\\
\Rightarrow V_0 &= Xe^{-rT}\exp\left(-\frac{1}{2}d_1^2+d_1\sigma\sqrt{T}-\frac{1}{2}\sigma^2T\right)\exp\left(\frac{1}{2}d_1^2\right)\\
&= Xe^{-rT}e^{d_1\sigma\sqrt{T}-\frac{1}{2}\sigma^2T}
\end{aligned}$$

Recall Equation (3.11) and see that

$$d_1\sigma\sqrt{T} = \ln\left(\frac{V_0}{X}\right)+rT+\frac{1}{2}\sigma^2T$$

Continuing on with Equation (3.15),

$$\begin{aligned}
V_0 &= Xe^{-rT}e^{\ln(V_0/X)+rT+\frac{1}{2}\sigma^2T-\frac{1}{2}\sigma^2T}\\
&= Xe^{\ln(V_0/X)} = X\frac{V_0}{X}\\
&= V_0
\end{aligned}$$

and we see that Equation (3.14) is true and so the last two terms of Equation (3.13) cancel. We have thus shown the delta of the Black and Scholes [1973] option pricing equation is indeed defined as per Equation (3.10).

Finally, having established a likelihood function $L(\mu,\sigma)$ for the equity-as-an-option pricing model, we choose values of our parameter set $\hat{\theta}=(\mu,\sigma)$ to maximise L while adhering to the strict requirement that for each equity price observation $S(t_i)$ (and implied asset valuations V_t), $S(t_i)=\Psi(t_{i,}V(t_i),\sigma)\,, \quad \forall i=1,2,...,n$ [Li and Wong, 2008].

3.3.3 The barrier dependent model of LS

The early-default model of Longstaff and Schwartz [1995] models equity as a down-and-out barrier option that becomes worthless the first time that asset values dip below some threshold H.

$$\Psi = C^{DOC}\left(V_0, X, H, R\right) = \Psi_A - \Psi_B - \Psi_C + \Psi_D + \Psi_E + \Psi_F \tag{3.16}$$

where

$$\begin{aligned}
\Psi_A &= V_0 \Phi\left(a\right) \\
\Psi_B &= X e^{-rT} \Phi\left(a - \sigma\sqrt{T}\right) \\
\Psi_C &= V_0 \left(\frac{H}{V_0}\right)^{2\nu} \Phi\left(b\right) \\
\Psi_D &= X e^{-rT} \left(\frac{H}{V_0}\right)^{2\nu-2} \Phi\left(b - \sigma\sqrt{T}\right) \\
\Psi_E &= R \left(\frac{H}{V_0}\right)^{2\nu-1} \Phi\left(c\right) \\
\Psi_F &= R \left(\frac{V_0}{H}\right) \Phi\left(c - 2\nu\sigma\sqrt{T}\right)
\end{aligned}$$

and

$$\begin{aligned}
a &= \begin{cases} \frac{\ln(V_0/X)+\left(r+\frac{1}{2}\sigma^2\right)T}{\sigma\sqrt{T}} & \text{, for } X \geq H \\ \frac{\ln(V_0/H)+\left(r+\frac{1}{2}\sigma^2\right)T}{\sigma\sqrt{T}} & \text{, for } X < H \end{cases} \\
b &= \begin{cases} \frac{\ln\left(H^2/V_0X\right)+\left(r+\frac{1}{2}\sigma^2\right)T}{\sigma\sqrt{T}} & \text{, for } X \geq H \\ \frac{\ln(H/V_0)+\left(r+\frac{1}{2}\sigma^2\right)T}{\sigma\sqrt{T}} & \text{, for } X < H \end{cases} \\
c &= \frac{\ln\left(H/V_0\right) + \left(r + \frac{1}{2}\sigma^2\right)T}{\sigma\sqrt{T}} \\
\nu &= \frac{r}{\sigma^2} + \frac{1}{2}
\end{aligned}$$

where R is the rebate paid out to equity holders in the event of default, and ϕ is the probability density function of the standard Normal distribution.

The logic to formulate the likelihood function for the Longstaff and Schwartz framework is the same as that of the Merton approach, with the only difference being the value for $\Delta\left(\Psi\right)$. A straight forward but lengthy application of the Chain and Product rules to C^{DOC} to calculate $\Delta\left(\Psi\right)$ follows.

Note that the first two terms Ψ_A and Ψ_B are of a similar format to the Black and Scholes [1973] call option in Equation (2.4). We use the working from Section (3.3.2) to see immediately that

$$\frac{\partial}{\partial V_0}\left(\Psi_A - \Psi_B\right) = \Phi\left(a\right)$$

Before we solve the remaining sections, we outline some simplifying conditions. Applying the Chain rule to the cumulative standard Normal distribution function,

$$\begin{aligned}\frac{\partial}{\partial V_0}\Phi\left(x\right) &= \frac{\partial}{\partial u}\Phi\left(u\right) \times \frac{\partial}{\partial V_0}u\\ &= \phi\left(u\right) \times \frac{\partial}{\partial V_0}u\end{aligned}$$

Observe also that

$$\frac{\partial}{\partial V_0}b = \frac{\partial}{\partial V_0}\left(b - \sigma\sqrt{T}\right) = \frac{\partial}{\partial V_0}c = \frac{\partial}{\partial V_0}\left(c - 2\nu\sigma\sqrt{T}\right) = \frac{-1}{V_0\sigma\sqrt{T}}$$

and therefore, for all $x \in \left\{b,\ b - \sigma\sqrt{T},\ c,\ c - 2\nu\sigma\sqrt{T}\right\}$,

$$\frac{\partial}{\partial V_0}\Phi\left(x\right) = \left(\frac{-1}{V_0\sigma\sqrt{T}}\right) \times \phi\left(x\right)$$

We continue with the rest of the derivation, performing piecewise operations on the remaining elements of Equation (3.16).

$$\begin{aligned}
\frac{\partial}{\partial V_0}\Psi_C &= \frac{\partial}{\partial V_0}\left[V_0\left(\frac{H}{V_0}\right)^{2\nu}\right]\Phi\left(b\right) + \left[V_0\left(\frac{H}{V_0}\right)^{2\nu}\right]\frac{\partial}{\partial V_0}\Phi\left(b\right)\\
&= \left[\frac{\partial V_0}{\partial V_0}\times\left(\frac{H}{V_0}\right)^{2\nu} + V_0\frac{\partial}{\partial V_0}\left(\frac{H}{V_0}\right)^{2\nu}\right]\Phi\left(b\right) + \left[V_0\left(\frac{H}{V_0}\right)^{2\nu}\right]\frac{\partial}{\partial V_0}\Phi\left(b\right)\\
&= \left[\left(\frac{H}{V_0}\right)^{2\nu} + V_0\left(\frac{-2\nu}{V_0}\right)\left(\frac{H}{V_0}\right)^{2\nu}\right]\Phi\left(b\right) + \left[V_0\left(\frac{H}{V_0}\right)^{2\nu}\right]\frac{\partial}{\partial V_0}\Phi\left(b\right)\\
&= \left[\left(\frac{H}{V_0}\right)^{2\nu} - 2\nu\left(\frac{H}{V_0}\right)^{2\nu}\right]\Phi\left(b\right) + \left[V_0\left(\frac{H}{V_0}\right)^{2\nu}\right]\left(\frac{-1}{V_0\sigma\sqrt{T}}\right)\phi\left(b\right)\\
&= \left[\left(-2\nu+1\right)\left(\frac{H}{V_0}\right)^{2\nu}\Phi\left(b\right)\right] - \left[\frac{\phi\left(b\right)}{\sigma\sqrt{T}}\left(\frac{H}{V_0}\right)^{2\nu}\right]\\
&= \left(\frac{H}{V_0}\right)^{2\nu}\left[\left(-2\nu+1\right)\Phi\left(b\right) - \frac{\phi\left(b\right)}{\sigma\sqrt{T}}\right]
\end{aligned}$$

$$\begin{aligned}
\frac{\partial}{\partial V_0}\Psi_D &= Xe^{-rT}\Phi\left(b-\sigma\sqrt{T}\right)\left[\frac{\partial}{\partial V_0}\left(\frac{H}{V_0}\right)^{2\nu-2}\right] + Xe^{-rT}\left(\frac{H}{V_0}\right)^{2\nu-2}\frac{\partial}{\partial V_0}\Phi\left(b-\sigma\sqrt{T}\right)\\
&= Xe^{-rT}\Phi\left(b-\sigma\sqrt{T}\right)\left[\frac{\left(-2\nu+2\right)}{V_0}\left(\frac{H}{V_0}\right)^{2\nu-2}\right] + Xe^{-rT}\left(\frac{H}{V_0}\right)^{2\nu-2}\frac{\partial}{\partial V_0}\Phi\left(b-\sigma\sqrt{T}\right)\\
&= \left(-2\nu+2\right)\frac{Xe^{-rT}}{V_0}\Phi\left(b-\sigma\sqrt{T}\right)\left(\frac{H}{V_0}\right)^{2\nu-2} - \left(\frac{H}{V_0}\right)^{2\nu-2}\left(\frac{Xe^{-rT}}{V_0\sigma\sqrt{T}}\right)\phi\left(b-\sigma\sqrt{T}\right)\\
&= \frac{Xe^{-rT}}{V_0}\left(\frac{H}{V_0}\right)^{2\nu-2}\left[\left(-2\nu+2\right)\Phi\left(b-\sigma\sqrt{T}\right) - \frac{\phi\left(b-\sigma\sqrt{T}\right)}{\sigma\sqrt{T}}\right]
\end{aligned}$$

$$\begin{aligned}
\frac{\partial}{\partial V_0}\Psi_E &= R\frac{\partial}{\partial V_0}\left(\frac{H}{V_0}\right)^{2\nu-1}\Phi\left(c\right) + R\left(\frac{H}{V_0}\right)^{2\nu-1}\frac{\partial}{\partial V_0}\Phi\left(c\right)\\
&= \frac{R}{V_0}\left(-2\nu+1\right)\left(\frac{H}{V_0}\right)^{2\nu-1}\Phi\left(c\right) - R\left(\frac{H}{V_0}\right)^{2\nu-1}\left(\frac{\phi\left(c\right)}{V_0\sigma\sqrt{T}}\right)\\
&= \frac{R}{V_0}\left(\frac{H}{V_0}\right)^{2\nu-1}\left[\left(-2\nu+1\right)\Phi\left(c\right) - \frac{\phi\left(c\right)}{\sigma\sqrt{T}}\right]
\end{aligned}$$

$$
\begin{aligned}
\frac{\partial}{\partial V_0}\Psi_F &= \frac{R}{H}\Phi\left(c-2\nu\sigma\sqrt{T}\right)+\frac{RV_0}{H}\frac{\partial}{\partial V_0}\Phi\left(c-2\nu\sigma\sqrt{T}\right)\\
&= \frac{R}{H}\Phi\left(c-2\nu\sigma\sqrt{T}\right)-\frac{RV_0}{H}\frac{\phi\left(c-2\nu\sigma\sqrt{T}\right)}{V_0\sigma\sqrt{T}}\\
&= \frac{R}{H}\left[\Phi\left(c-2\nu\sigma\sqrt{T}\right)-\frac{\phi\left(c-2\nu\sigma\sqrt{T}\right)}{\sigma\sqrt{T}}\right]
\end{aligned}
$$

By combining all the above elements, we complete the derivation of Δ for a down-and-out call option:

$$
\begin{aligned}
\Delta\left(\Psi\right) = \frac{dC^{DOC}}{dV_0} &= \frac{\partial}{\partial V_0}\left(\Psi_A-\Psi_B-\Psi_C+\Psi_D+\Psi_E+\Psi_F\right) \qquad (3.17)\\
&= \Phi\left(a\right)\\
&- \left(\frac{H}{V_0}\right)^{2\nu}\left[\left(-2\nu+1\right)\Phi\left(b\right)-\frac{\phi\left(b\right)}{\sigma\sqrt{T}}\right]\\
&+ \frac{Xe^{-rT}}{V_0}\left(\frac{H}{V_0}\right)^{2\nu-2}\left[\left(-2\nu+2\right)\Phi\left(b-\sigma\sqrt{T}\right)-\frac{\phi\left(b-\sigma\sqrt{T}\right)}{\sigma\sqrt{T}}\right]\\
&+ \frac{R}{V_0}\left(\frac{H}{V_0}\right)^{2\nu-1}\left[\left(-2\nu+1\right)\Phi\left(c\right)-\frac{\phi\left(c\right)}{\sigma\sqrt{T}}\right]\\
&+ \frac{R}{H}\left[\Phi\left(c-2\nu\sigma\sqrt{T}\right)-\frac{\phi\left(c-2\nu\sigma\sqrt{T}\right)}{\sigma\sqrt{T}}\right]
\end{aligned}
$$

Chapter 4

Data and methodology

Jones et al. [1984] note that '[contingent claims analysis] requires three kinds of data in order to solve for prices of individual claims as functions of total firm value: (1) indenture data, (2) standard deviation data, and (3) interest rate data.'

With this in mind, the following chapter discusses the various data sources and selection criteria applied to collate the sample data sets. We outline the procedures used to select interest rates, company financial statements, market bond prices and equity valuations, and any processing required to clean the data before inclusion in our samples.

We include a detailed breakdown of the methods and techniques used to estimate input parameters for the bond models. Where relevant, we include an explanation of any assumptions made based on results from previous studies, such as values for expected recovery rates and typical default boundary ratios.

4.1 Choice of market

All research in this study is sourced from financial markets in the United States of America. While checking data availability, the UK and Germany markets are quickly ruled out by a combination of problems including limited numbers of bond issues and difficulty in obtaining the required financial statement data.

As an indication of their relative size, a precursory search for bonds that matched broad initial conditions yields over 8,000 US corporate bonds, approximately 200 German bonds, and a total of 5 from the United Kingdom. We examine US market data because US markets offer the best chance of getting a reasonable number of bonds once more rigorous selection criteria are applied.

4.2 Interest rate data

The ability to accurately model the interest rate dynamics under the Vasicek [1977] model is key to the MLE asset dynamics estimation technique and also to the bond price predictions that are the ultimate focus of this study.

4.2.1 Sample selection criteria

We first establish the date of the earliest bond issue in our sample. As part of the asset estimation process described in later sections, it is necessary to calculate interest rate values for the calendar year prior to the bond pricing date in question. Consequently, yield curve values are downloaded for each available business day between January 2001 and June 2008.

4.2.2 Sourcing risk-free yield curve interest rate data

Daily observations of the United States risk-free zero yield curve are retrieved from Thomson Datastream Advance, a popular software package containing a diverse collection of interest rate, bond price and company financial statement information from markets around the world.

Datastream contains observations of the zero yield curve term structure for each business day modelled off the returns from risk-free financial instruments issued by the US Treasury. For each daily term structure observation we collect yield curve data for monthly maturity intervals out to 12 months and then yearly observations out to a maximum of 10 years, and will later search for parameter values to best fit these daily observations.

4.3 Company financials, bond details and historical bond price data

Having described the collection of interest rate data, we now outline the selection process used to choose our sample of firms and bonds before we implement the various parameter estimation techniques and structural modelling formulae.

4.3.1 Bond sample selection criteria and data sourcing procedures

The Merton [1974] model, and indeed several of the subsequent revisions, imagines an issuing firm as operating under an extremely simplified capital structure, consisting of a single debt issue and equity. To fairly test the model performances, previous researchers [Longstaff and Schwartz, 1995, Eom et al., 2004, Li and Wong, 2008] typically limit their samples to firms with a simple capital structure incorporating few types of debt, single levels of seniority and so on, yet still allow some deviation from a strict observance of all the strict assumptions of Merton's model.

We take a more absolute stance, and restrict our sample to firms with strictly a single bond on issue within the time frame of analysis. By limiting the number of complicating factors related to capital structure as much as possible, we intend to make a more conclusive statement concerning the ability of each model to perform as originally described.

Apart from this more strict approach, we follow the selection process of Eom et al. [2004], Li and Wong [2008] and choose firms that meet the following criteria.

We search for bonds issued by non-financial, non-utility firms with no preferred stock on issue. The bonds must be non-callable, non-convertible, and have finite maturities of greater than one year. Over 8,000 bonds matched these initial requirements. We then trim out all firms with more than a single bond on issue, and choose only bonds with fixed interest coupons, issued in US dollars on a US exchange between January 2002 and June 2008. At this point, just over 350 potential bonds remain.

For each bond we require a full history of its daily close prices starting at the date of issue. For reasons unknown, Datastream does not always contain a complete price history for each bond. Any bond record whose daily price entries begin more than one month from date of issue is discarded from the sample.

Datastream lacks relational ties between bonds and the issuing firms themselves, making it impossible to easily match bond price observations with financial data for the relevant firm. In addition, the Datastream records of yearly balance sheet statements for each firm are found to often be incomplete or unavailable, and so we turn instead to Compustat to locate end-of-year financial statements for each potential firm. Compustat is another financial data software package, and features a more comprehensive collection of financial data for each firm than that available in Datastream.

We manually match Datastream bond issue data with Compustat company financial records based on company name. We reject any firm whose statements are not issued in December each year so that the MLE asset valuation technique is applied to each firm for the same relative time periods.

Finally, we require daily equity values, which, multiplied by the number of shares outstanding, gives the daily market capitalisation. Neither Datastream nor Compustat offer reliable amounts of share price data, so we retrieve daily market observations from the Yahoo! Finance website where possible. In order to avoid the problem of handling payout ratios and dividends, we take adjusted close prices, which are modified to factor in the effects of stock splits and dividend payments.

Due to the extensive amount of manual record-matching required to collect and collate data for each potential firm, the task of finding suitable bonds for the sample proved to be a laborious one. As a result, the number of bonds in our sample is roughly a tenth the size of those of Eom et al. [2004], Li and Wong [2008], and approximately half of the sample tested by Jones et al. [1984]. Our final sample consists of 14 bonds.

4.3.2 Bond sample summary

Table (4.1) briefly outlines the critical details for each bond included in the sample.

Despite the wide range of industry classifications, there are similarities between the different bond issues. All bonds included in the final sample have issue dates between 2002 and 2008, and an average maturity of approximately 9 years. The sample is split roughly equally into investment and non-investment grade debt, which results in a consistant range in the observed coupon rates, starting at a minimum of 4.25% and as high as 10%.

4.4 Experiment design and implementation

We now explicitly outline the steps involved to implement the selected bond pricing models using both pure proxy and maximum likelihood estimation of parameters. We include a description of all values that we explicitly estimate, as well as those additional parameters whose values

Table 4.1: Sample firm and bond description

Name	Issue Date	Maturity[‡]	Principal[⋆]	Coupon %	Rating[†]
Anixter Inc.	24/02/05	10	200	5.95	Ba1
Baldor Electric Comp.	25/01/07	10	550	8.625	B3
Berry Petroleum Comp.	19/10/06	10	200	8.25	B3
Blount Inc.	4/08/04	8	175	8.875	B2
Coca-Cola Comp.	29/10/07	10	1,750	5.35	Aa3
Comstock Resources Inc.	18/02/04	8	175	6.875	B2
Dean Foods Comp.	11/05/06	10	500	7	B3
Millicom Intl. Cellular SA	19/11/03	10	550	10	B1
Public Service Ent. Grp.	16/08/04	10	250	5	A3
Rockwell Collins Inc.	17/11/03	10	200	4.75	A1
Schlumberger Tech. Corp.	4/04/02	10	1,000	6.5	A2
SK Telecom Comp. Ltd.	23/03/04	7	300	4.25	A2
W&T Offshore Inc.	8/06/07	7	450	8.25	B3
WCA Waste Corp.	1/11/06	7.5	150	9.25	B3

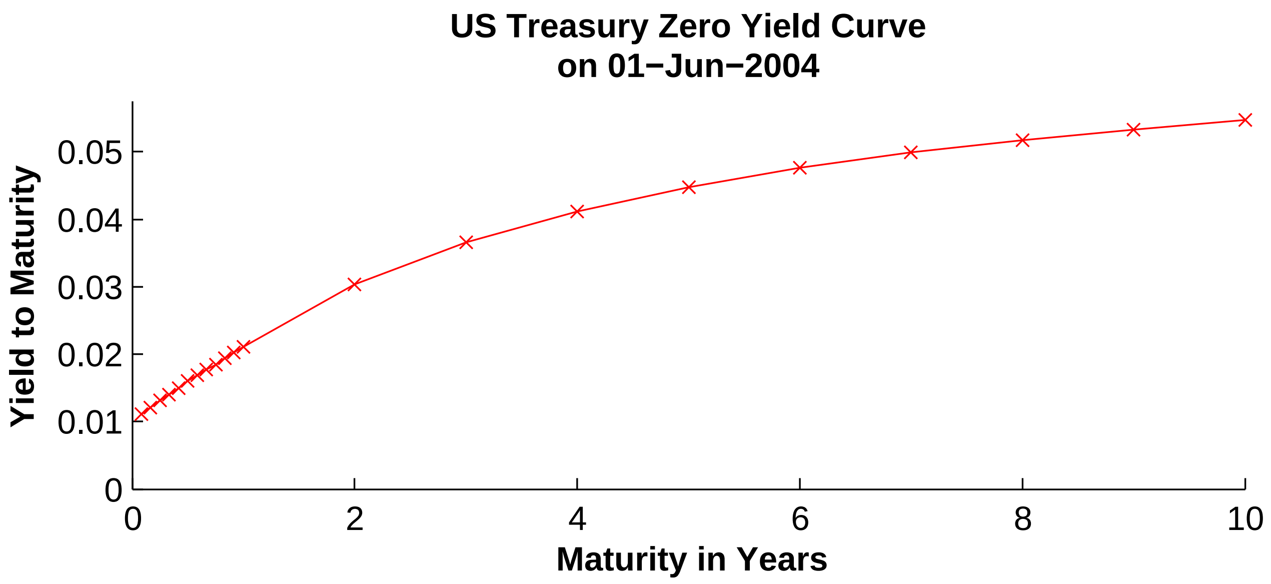
[‡] Maturity in years.[⋆] Face value in $millions. [†] Credit rating supplied by Moody's.

are assumed on the basis of prior research.

4.4.1 Estimating the Vasicek model instantaneous interest rate parameters

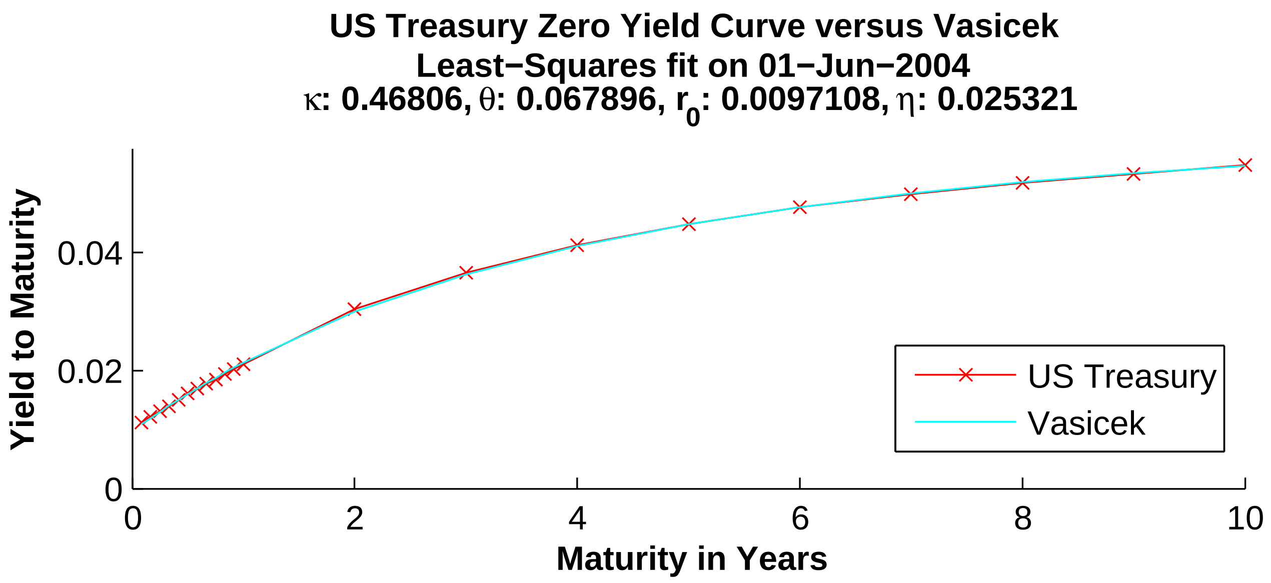
For any given day it is possible to observe the yield to maturity of several risk-free instruments over a range of discrete maturity values, as seen in Figure (4.1). This relationship between yield and maturity is commonly referred to as the term structure, and reflects market sentiments about future interest rate movements. The term structure describes an investment's yield versus time to maturity, typically based on a set number of actual instruments of varying maturities, interpolated to solve for any intermediate, unobservable maturities required. Depending on the broader economic conditions, the yield curve can take a range of shapes, as the relationship between short and long term investments shifts according to market expectations of future economic activity.

Figure 4.1: Illustrative observation of typical US Treasury yield curve shape



We follow the approach of Eom et al. [2004], Li and Wong [2008] to estimate the parameters of the Vasicek [1977] stochastic process that is assumed to capture the dynamics of the instantaneous risk-free interest rate. We choose values of κ, η, r, θ to minimise the least-squares difference between the resultant estimated yield curve and a set of observed term structure data points.

Figure 4.2: Vasicek [1977] model least-squares fit to observed yield curve



The least-squares approach is applied by Nelson and Siegel [1987] when fitting their own interest rate model to observed term structures [Bayazit, 2004]. They note the difficulties of getting any model to perfectly match observed data, especially at both short and long ends of the yield curve simultaneously. However, they also suggest that it may in fact be preferable to match the overall suggestive shape of the interest rate curve as opposed to trying to capture absolutely all subtle nuances. They suggest that minor deviations may often be artifacts of the data collection process, and so should not factor heavily into the parameter estimation process anyway.

Our average sample bond maturity is approximately nine years and the maximum is ten years. We therefore choose Vasicek parameter values to reflect the observed term structure behaviour at similar maturities. We model our estimates to match the longer end of the term structure, and note that this may require a sacrifice of model performance at the short end of the estimated curve.

4.4.2 Estimating asset value and volatility using the pure proxy method

The simplest method of estimating market value of assets and asset volatility is the pure proxy method. This method equates market value of assets to the sum of market value of debt and equity, as defined in Equation (2.5). The market value of a firm's debt is not directly observable and is typically proxied by the book value as stated in regular balance sheet statements released to the market by each firm at the end of each financial year. As a result, the value of debt is considered constant throughout the financial year.

Market value of equity is calculated by multiplying the number of outstanding shares by the daily value of equity. Having calculated implied asset values via the pure proxy approach, the drift and volatility of the log of asset returns is calculated directly from the results.

4.4.3 Estimating asset value and volatility using the MLE method

Li and Wong [2008] note that by substituting the observable but higher book value of debt for the unobservable market value, the pure proxy method overstates the value of assets. This in turn leads to an underestimation of risk. Based on the work of Duan [1995] they suggest the more theoretically consistent method of maximum likelihood estimation to calculate asset value and asset dynamics. Duan uses the observable value of derivative contracts to calculate the implied value of an underlying asset when market value of that asset is not directly observable.

Instead of taking market value of equity to directly represent the value of the company assets, Li and Wong consider equity as an option on those assets. The implied asset value will necessarily be below that of the simple sum of market value of equity and book value of debt since the value of an option must incorporate some opportunity premium. If this were false, investors could buy an option and exercise immediately to make a risk-free profit, an impossibility under the assumption of arbitrage-free markets.

A pivotal idea for our implementation of the MLE method of asset estimation is suggested in Duan et al. [2003]. Duan et al. suppose that equity holders hold an option whose maturity coincides with the release of annual financial reports. On the date of each report and depending on the financial health of the company, debt holders may force the company into bankruptcy.

We decide to estimate the asset values and dynamics by adhering to this approach. Suppose we wish to estimate the asset dynamics for some year Y_E. The MLE technique requires the following

- The market value of debt for year Y_E. As discussed previously, the Merton [1974] model assumes a very simple capital structure whereby the firm has only a single bond issue as debt. However, in practice we acknowledge that equity has no value until all liabilities are paid out. Thus, we substitute the balance sheet Total Liabilities entry for the value of debt. We retrieve the yearly Total Liabilities value from the end-of-financial year statements released in year $Y_F = Y_E - 1$.

- Total number of outstanding shares for year Y_E, again retrieved from Y_F financial statements.

- Adjusted equity prices for each business day of Y_E.

- The appropriate option pricing model for equity, dependent on the particular bond pricing model being tested.

- Interest rate data for each business day of Y_E.

The MLE approach requires the option pricing formula to be inverted so that it takes the price of the option (in this case, represented by the daily market capitalisation) as an input, and produces an implied firm asset value as output. Thus begins the iterative process to select optimal estimates for the most likely values of yearly asset drift μ and volatility σ to have produced the observed equity prices.

For each daily equity observation and some estimates of μ and σ, calculate the implied asset values. Set the maturity date of each option to be the end of year, that is, each option calculated will have a decreasing maturity as we work through the daily equity observations for year Y_E. Set the strike price of the inverted option formula to be $X =$ Total Liabilities, since equity has no value until all liabilities are paid out. Based on the above conditions, produce an implied asset path for the year for each drift and volatility estimate.

By repeatedly substituting the resulting years worth of implied asset values into an appropriate log-likelihood function L, we search for the values of μ and σ that maximise L while equating the equity observations to the value of the relevant option, stated more formally [Li and Wong, 2008] as

$$\max_{\mu,\sigma} L_{Y_E}(\mu,\sigma)\text{s.t. } S(t_i) = \Psi(t_i, V_{t_i}, \sigma, X, H), \ \forall i = 1, 2, ..., n, \tag{4.1}$$

where $S(t_i)$ are daily share price observations, V_{t_i} is the implied asset value, X is the total liabilities of the firm and H is the default barrier at which bond holders would force the company into bankruptcy. These are our maximum likelihood estimates of μ and σ.

In addition the Longstaff and Schwartz [1995] model requires the correlation ρ between asset returns and the risk-free interest rate. Having fitted Vasicek [1977] to estimate r and used MLE to estimate of μ and σ, both the pure proxy and most likely implied asset paths for the year are generated and ρ^{PP}, ρ^{MLE} are calculated directly from the results.

4.4.4 Additional assumptions while performing MLE of asset dynamics

When implementing the MLE technique we apply a few simplifying assumptions. In all cases, we set the default barrier of the equity-as-an-option to be the full face value of debt, $H = X$. We also follow the approach of Li and Wong [2008] in assuming that equity holders receive nothing if the issuing firm goes bankrupt. In the down-and-out call option model used to estimate asset dynamics for the Longstaff and Schwartz [1995] model, this means setting the rebate to equity holders $R = 0$.

4.4.5 Additional assumptions regarding bond model parameters

Finally, several parameters for the bond pricing models are assigned values based on previous empirical studies.

Eom et al. [2004] cite numerous studies related to recovery rates in the event of bankruptcy. While it is acknowledged that the recovery rate will depend on the industry and nature of the firm's operations and assets, we follow Eom et al. and assume a universal recovery rate $\omega = 51.31\%$ for the face value of a defaulting bond. Collin-Dufresne and Goldstein [2001] note that outstanding coupon payments are rarely recovered in the event of bankruptcy. We follow their example and set the recovery rate of coupons $\omega_c = 0\%$.

In specifying a default boundary H for the Longstaff and Schwartz [1995] model, we take the findings of Choi and Wong [2006] and set $H = 0.738X$.

4.4.6 Summary of model parameters, description and data sources

Table 4.2: Summary of model input parameters

Parameter	Description	Estimated as	Data source
Bond features			
c	Coupon rate (%)	Given	Datastream
T	Maturity	Given	Datastream
X	Face value	Total liabilities	Compustat & Datastream
ω	Face value recovery rate	Eom et al. [2004]	n/a
ω_c	Coupon recovery rate	Collin-Dufresne and Goldstein [2001]	n/a
H	Default boundary	Choi and Wong [2006]	n/a
Firm characteristics			
V_0	Firm value at issue	Total liabilities + market capitalisation	Compustat & Yahoo! Finance
μ	Asset returns	Yearly change in asset value V_t, (pure proxy and MLE)	n/a
σ	Asset volatility	Yearly standard deviation in asset returns, (pure proxy and MLE)	n/a
Interest rates			
r	Instantaneous risk-free interest rate	Vasicek [1977]	Datastream
κ	Strength of interest rate mean-reversion	Vasicek [1977]	Datastream
θ	Long term risk-free interest rate	Vasicek [1977]	Datastream
η	Volatility of risk-free interest rate	Vasicek [1977]	Datastream
ρ	Correlation between V_t and r	Correlation between pure proxy & MLE implied asset values and r	Compustat, Datastream & Yahoo! Finance

Chapter 5

Results

The results chapter is divided into three key topics. In Section (5.1) we discuss the results of the least-squares approach when fitting the Vasicek [1977] interest rate model to observed term structures. Section (5.2) discusses the differences in the bond model input parameter estimates using the pure proxy and maximum likelihood approaches. The final sections discuss in detail the effect that the parameter estimation techniques have on the performance of the Merton [1974] and Longstaff and Schwartz [1995] structural bond pricing models.

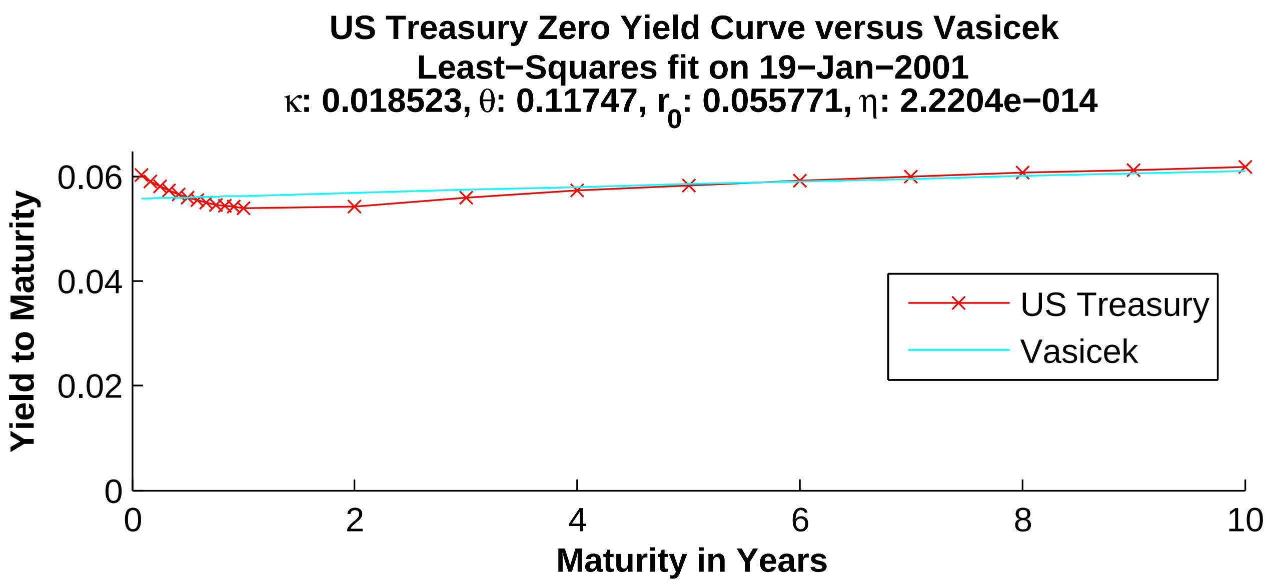
5.1 Fitting the yield curve and interest rate parameter estimation

The Vasicek [1977] model appears to perform quite well for most term structure shapes witnessed within our sample period. Appendix (A) shows plots of each estimated yield curve versus the observed market values on the closest day on or after the issue date of each bond in our sample.

The model struggles when the curve is sharply inverted at shorter maturities, such as is seen in Figure (5.1), which plots the yield curve as of January 2001. The model appears to have too few parameters to capture multiple changes of sign in the slope of the term structure.

While Appendix (A) includes only graphs of the fitted curves immediately after each date of issue, we actually calculate and store the Vasicek parameters for each daily observation in the sample period so that both asset value estimation and also bond pricing techniques can be quickly performed for any particular date.

Figure 5.1: Poor Vasicek [1977] model fit at short end of yield curve



It is interesting to note that the interest rate data in our sample incorporates a full range of term structure shapes. Early in the sample the yield curve slopes gently upward as is the norm in calm economic climates [Hull, 2006]. As time progresses however, the yield curve flattens and ultimately becomes inverted, which typically indicates that investors believe that economic activity will decline in future. This inverted curve period coincides with widely reported unrest in global financial markets, as global economies slow and investors experience large-scale defaults on complex derivative investment products.

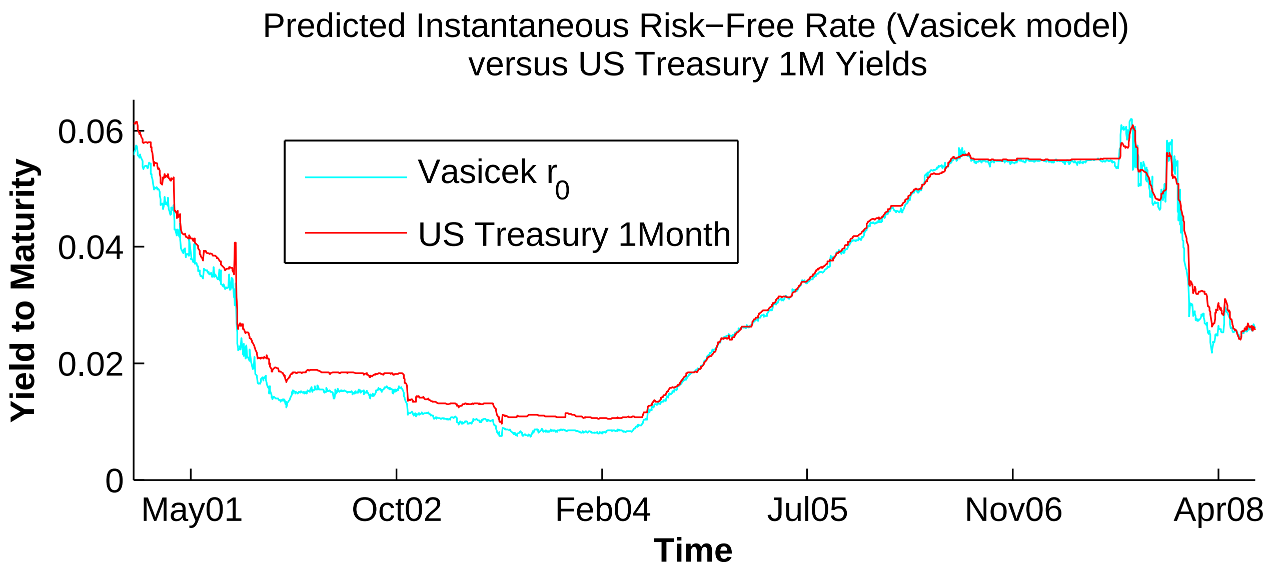
Finally, in Figure (5.2) we overlay the historic predictions of r according to Vasicek with the shortest observable market rate, a one month Treasury note.

The instantaneous rate predicted by the Vasicek model can be seen to closely match the one month rate, suggesting that the least-squares method adequately predicts the short term interest rate even while matching the behaviour at the long end of the term structure.

5.2 Estimates of asset dynamics using pure proxy and MLE techniques

The empirical results for asset dynamics estimates are summarised on a per-firm basis, as calculated for financial year intervals between 2002-2007 inclusive.

Figure 5.2: Least-squares fitted Vasicek [1977] instantaneous rate versus observed 1 Month US Treasury rate



5.2.1 Asset dynamics estimates

Table (5.1) lists the yearly estimates of μ, σ and ρ broken down by company and estimation technique, where μ, σ and ρ represent the asset drift, volatility and correlation with the estimated instantaneous risk-free interest rate respectively.

Table 5.1: Asset dynamics estimates

Name	Year	μ^{PP}	σ^{PP}	ρ^{PP}	μ^{M}	σ^{M}	ρ^{M}	μ^{LS}	μ^{LS}	ρ^{LS}
Anixter Inc.										
	2004	0.241	0.140	0.941	0.256	0.154	0.942	0.256	0.154	0.942
	2005	0.057	0.167	0.566	0.087	0.195	0.623	0.087	0.195	0.623
	2006	0.172	0.172	0.712	0.213	0.195	0.721	0.213	0.195	0.721
	2007	0.124	0.200	0.038	0.177	0.232	0.014	0.177	0.232	0.014
Baldor Electric Comp.										
	2006	0.224	0.234	0.274	0.270	0.264	0.297	0.270	0.264	0.297
	2007	0.045	0.286	0.246	0.107	0.325	0.239	0.107	0.325	0.239

Berry Petroleum Comp.									
2005	0.212	0.364	0.580	0.303	0.415	0.585	0.303	0.415	0.585
2006	0.048	0.307	-0.548	0.113	0.337	-0.534	0.113	0.337	-0.534
2007	0.304	0.235	-0.494	0.362	0.260	-0.488	0.362	0.260	-0.488
Blount Inc.									
2003	0.118	0.098	-0.167	0.139	0.109	-0.247	0.133	0.107	-0.226
2004	0.250	0.116	0.897	0.266	0.127	0.898	0.265	0.127	0.898
2005	-0.050	0.155	-0.281	-0.025	0.170	-0.168	-0.025	0.170	-0.168
2006	-0.116	0.167	-0.855	-0.080	0.188	-0.845	-0.080	0.188	-0.845
2007	-0.033	0.218	0.481	0.023	0.250	0.423	0.023	0.250	0.424
Coca-Cola Comp.									
2006	0.172	0.094	0.700	0.184	0.105	0.703	0.184	0.105	0.703
2007	0.248	0.123	-0.397	0.267	0.141	-0.396	0.267	0.141	-0.396
Comstock Resources Inc.									
2003	0.324	0.144	-0.711	0.345	0.161	-0.715	0.345	0.161	-0.715
2004	0.067	0.244	0.418	0.106	0.269	0.431	0.106	0.269	0.431
2005	0.234	0.229	0.772	0.281	0.263	0.785	0.281	0.263	0.785
2006	-0.032	0.254	-0.293	0.017	0.279	-0.247	0.017	0.279	-0.247
2007	0.089	0.198	-0.545	0.143	0.230	-0.535	0.143	0.230	-0.535
Dean Foods Comp.									
2005	0.124	0.085	0.941	0.145	0.098	0.952	0.145	0.098	0.952
2006	0.044	0.084	0.308	0.075	0.097	0.434	0.075	0.097	0.434
2007	-0.037	0.114	0.229	0.002	0.132	0.190	0.002	0.132	0.190
Millicom Intl. Cellular SA									

2002	-0.243	0.123	0.058	-0.210	0.194	-0.092	-0.225	0.134	0.045
2003	0.497	0.159	-0.737	0.631	0.200	-0.771	0.518	0.170	-0.737
2004	0.108	0.237	-0.094	0.148	0.263	-0.076	0.147	0.263	-0.079
2005	0.103	0.177	0.139	0.135	0.201	0.196	0.135	0.201	0.196
2006	0.511	0.337	0.287	0.598	0.349	0.310	0.597	0.350	0.310
2007	0.486	0.399	-0.438	0.613	0.456	-0.438	0.612	0.456	-0.438
Public Service Enterprise. Grp.									
2003	0.049	0.027	-0.810	0.060	0.030	-0.836	0.060	0.030	-0.836
2004	0.041	0.028	0.246	0.048	0.030	0.353	0.048	0.030	0.353
2005	0.059	0.048	0.915	0.081	0.053	0.951	0.081	0.053	0.951
2006	0.008	0.050	-0.130	0.043	0.057	0.428	0.043	0.057	0.428
2007	0.130	0.089	-0.261	0.177	0.100	-0.286	0.176	0.100	-0.286
Rockwell Collins Inc.									
2002	0.111	0.315	0.317	0.176	0.348	0.308	0.176	0.348	0.308
2003	0.186	0.182	-0.785	0.210	0.199	-0.787	0.210	0.199	-0.787
2004	0.238	0.176	0.937	0.261	0.199	0.938	0.261	0.199	0.938
2005	0.152	0.155	0.435	0.174	0.175	0.459	0.174	0.175	0.459
2006	0.250	0.151	0.614	0.276	0.171	0.626	0.276	0.171	0.626
2007	0.112	0.175	-0.509	0.143	0.201	-0.505	0.143	0.201	-0.505
Schlumberger Tech. Corp.									
2001	-0.307	0.337	0.843	-0.228	0.374	0.837	-0.228	0.374	0.837
2002	-0.150	0.262	0.215	-0.104	0.291	0.210	-0.104	0.291	0.210
2003	0.179	0.167	-0.727	0.202	0.188	-0.731	0.202	0.188	-0.731
2004	0.152	0.170	0.600	0.173	0.191	0.607	0.173	0.191	0.607

	2005	0.349	0.206	0.905	0.384	0.233	0.907	0.384	0.233	0.907
	2006	0.181	0.312	0.453	0.248	0.343	0.470	0.248	0.343	0.470
	2007	0.436	0.284	-0.200	0.501	0.321	-0.202	0.501	0.321	-0.202
SK Telecom Comp. Ltd.										
	2003	-0.079	0.239	-0.438	-0.040	0.264	-0.451	-0.040	0.264	-0.451
	2004	0.111	0.183	-0.121	0.134	0.203	-0.111	0.134	0.203	-0.111
	2005	-0.024	0.133	0.207	-0.004	0.152	0.291	-0.004	0.152	0.291
	2006	0.182	0.162	0.253	0.214	0.179	0.300	0.214	0.179	0.300
W&T Offshore Inc.										
	2006	0.012	0.354	-0.290	0.095	0.384	-0.277	0.095	0.384	-0.277
	2007	0.030	0.183	-0.259	0.075	0.212	-0.315	0.075	0.212	-0.315
WCA Waste Corp.										
	2005	-0.148	0.185	-0.906	-0.119	0.204	-0.892	-0.119	0.204	-0.892
	2006	0.017	0.143	-0.209	0.057	0.165	-0.069	0.057	0.165	-0.070
	2007	-0.097	0.163	0.350	-0.048	0.189	0.281	-0.048	0.189	0.282

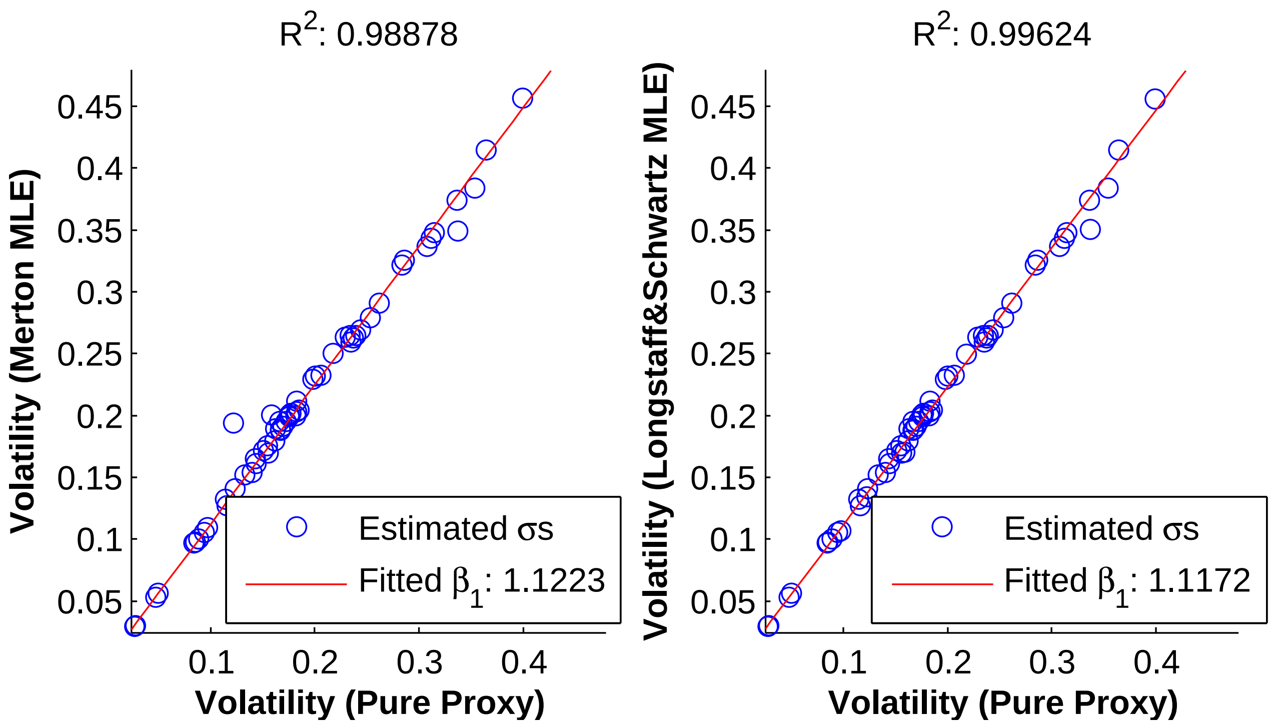
Appendices (B) and (D) plot the implied asset volatility values by year and the resulting implied firm asset valuations respectively. The plots show that implied asset value is always greater using the pure proxy approach, and estimates of asset volatility lower. The graphs in Appendix (D) also plot the yearly book value of total liabilities and market value of equity. We see that firms in our sample are typically well capitalised, with large equity-to-total-liabilities ratios. This is would suggest that our sample firms are generally low risk, consistent with the observation that the majority of bonds in our sample are of investment and upper-speculative grades.

5.2.2 Correlation between methods of asset dynamics estimation

A surprising trend in the asset dynamics estimation results is the apparent strength of correlation in the output of the pure proxy and maximum likelihood methods. We provide scatter plots of the yearly volatility estimates on a per-firm basis comparing pure proxy to MLE output of the European call and down-and-out call option approaches. The plots provide strong evidence of a linear relationship between the estimated volatility values with very high correlation values $R^2 \geq 0.97$ consistently reported, see Appendix (C).

The same correlations results hold when we compare values across the entire set of yearly firm volatility estimates, plotting each pure proxy measurement against the corresponding MLE value as seen in Table (5.3).

Figure 5.3: Relationship between pure proxy vs MLE volatility estimates



We perform a simple regression analysis over the data and see that the relationship between the pure proxy estimate and the MLE counterpart is described very well by a linear model representing a straight line,

$$y_i = \beta_0 + x_i\beta_1 + \epsilon_i \tag{5.1}$$

where x_i, y_i are the independent and dependent variables respectively, β_0 and β_1 are the intercept and slope of the regression line respectively, and ϵ_i represents an error term. For the aggregate sample the regression model parameters are predicted as $\beta_0 = 0$, $\beta_1 \approx 1.1223$ and $\beta_0 = 0$, $\beta_1 \approx 1.1172$ for the Merton [1974] and Longstaff and Schwartz [1995] implied option pricing models respectively. The 95% confidence intervals for each model's β_1 parameter are (1.1095, 1.1351) and (1.1097, 1.1247), with the minimal variance again suggesting a good fit.

Unreported results suggest that the estimates of drift tend to differ between estimation techniques slightly more so than the volatility estimates, but are still highly correlated. Because asset drift is not used as a parameter in bond pricing models, we do not discuss these results in any more detail than already provided in Table (5.1).

The high values of R^2 between pure proxy and MLE volatility estimates as visible at the level of both individual firms and also the aggregate sample suggest an interesting possibility. There is apparently a very strong correlation and highly linear relationship between output from the two estimation techniques. It may therefore be possible for analysts to forgo the lengthy and computationally taxing process of searching for MLE estimates of asset drift and volatility in future, and to instead simply scale the easily calculated pure proxy values by the fitted β_1 factor.

Further study would be required to test which bond features affect the scaling factor. Given that our study contains only a small number of bonds, of similar maturities and medium level risk ratings, more work is required to establish whether different scaling values are needed, or whether our observed ratios apply in a broader range of scenarios. Of course, this estimation shortcut may ultimately prove to be futile if the MLE estimates do not improve bond model performance.

5.3 Aggregate performance of structural models and parameter estimation techniques

Table (5.2) summarises the performance of both bond pricing models using each method to estimate asset volatility, averaged across all the predictions of sample bond prices taken as near to date of issue as possible. For brevity, all future references to predictions performed on the bond issue date will refer to the first available date on or after issue for which market price observations are available.

Table (5.3) provides data in the same format but instead uses the entire collection of end-of-month predictions to calculate the average performance. Each bond is therefore factored into the averaging process numerous times, once for each end-of-month prediction. Again, the end of month observations referred to in future will refer to the first available date on or after the end of each month for which a market price observation is available.

The results in Table (5.3) are included to offer insight into the performance of each bond model over a larger number of sample observation dates, however the calculated values ignore any correlation between the monthly values of a single firm. Consequently, the results should be considered as a qualitative indication of model performance only, and not considered in a strictly quantitative sense.

Sections (5.4) and (5.5) report on the performance of the Merton [1974] and Longstaff and Schwartz [1995] models broken down by individual firm.

Table 5.2: Aggregate performance of models and estimation techniques as predicted at date of issue

Model	Pure Proxy			MLE		
	% error in prices	% error in yields	Yield differences	% error in prices	% error in yields	Yield differences
Merton						
Mean	13.22%	-23.83%	-1.81%	13.15%	-23.72%	-1.80%
Std Dev	11.44%	16.43%	1.49%	11.35%	16.34%	1.48%
Longstaff and Schwartz						
Mean	3.61%	-4.26%	-0.42%	0.56%	2.52%	0.05%
Std Dev	13.00%	23.99%	1.85%	13.64%	27.30%	2.03%

Table of the average empirical performance of each bond pricing model with asset dynamics parameters estimated using the pure proxy and MLE techniques. Each bond price/yield is estimated on the day soonest on or after the date of issue and compared with the observed price. For each model and estimation technique, percentage error in prices is $\frac{\text{predicted price-actual price}}{\text{actual price}}$ and percentage error in yields in $\frac{\text{predicted yield - actual yield}}{\text{actual yield}}$. Yield differences are calculated as predicted yield $-$ actual yield.

Table 5.3: Aggregate performance of models and estimation techniques averaged across all available monthly predictions

Model	Pure Proxy			MLE		
	% error in prices	% error in yields	Yield differences	% error in prices	% error in yields	Yield differences
Merton						
Mean	11.44%	-22.08%	-1.82%	11.42%	-22.02%	-1.82%
Std Dev	11.49%	18.74%	1.79%	11.47%	18.71%	1.79%
Longstaff and Schwartz						
Mean	6.01%	-10.34%	-1.00%	4.23%	-6.20%	-0.71%
Std Dev	10.89%	22.24%	1.77%	10.91%	23.77%	1.81%

Table of the averaged empirical performance of each bond pricing model over all end-of-month estimation values. Results are broken down by asset dynamics parameters estimated using the pure proxy and MLE techniques. Each bond price/yield is estimated on the day soonest on or after each end of month interval in the sample period and compared with the observed price. For each model and estimation technique, percentage error in prices is $\frac{\text{predicted price-actual price}}{\text{actual price}}$ and percentage error in yields in $\frac{\text{predicted yield - actual yield}}{\text{actual yield}}$. Yield differences are calculated as actual yield-predicted yield.

5.4 Performance of the Merton [1974] model

Figure (5.4) graphically demonstrates the tendency of the Merton [1974] model to greatly underestimate the yield of risky bonds. We discuss errors in bond yields as opposed to bond prices as they provide a clearer comparison of performance. In almost all cases, the yield predicted by Merton almost exactly matches that of an otherwise equivalent risk-less bond priced off the yield curve using the Vasicek [1977] model. This suggests that the Merton model fails to adequately capture market valuations of risk, and is true of estimates generated by both the pure proxy and MLE methods.

Table (5.4) breaks down the results of the Merton model by estimation method as applied to the individual firms in our sample. Again, it appears to make little difference whether Merton is implemented using either the pure proxy or the MLE approach. The average yield error of the MLE approach is 1.44% at best and -51.71% at worst, which is of negligible difference to the 1.41% and -52.18% values achieved using the pure proxy approach. The net difference between the pure proxy predicted and actual yields is at best 0.07% and at worst -4.96%. This compares with the best and worst MLE predicted differences of 0.07% and -4.92% respectively.

In contrast to the study of Li and Wong [2008], we find little evidence to suggest that the MLE technique significantly improves the performance of the Merton model over the pure proxy approach. Our earlier results suggest using the strong correlations between the estimation technique values to bypass the complexity of MLE while still benefitting from any improvement to model performance. However, the Merton model remains inaccurate even when we use parameters estimated via maximum likelihood. This suggests that flawed input parameter values cannot fully explain the poor model performance, and that the model itself fails to capture some crucial factor in predicting market perceptions of bond risk.

The plots in Appendix (E) show that the Merton model does sometimes manage to predict yields close to the observed market values. For investment grade debt such as Rockwell Collins Inc. and Schlumberger Tech. Corp. the predicted yields very closely match the market yields. However, we note that this only occurs when the market yield of the corporate bond is already very close to the implied risk-free bond yield, as may be expected for investment grade debt, and so any failure of the model to capture credit risk is less apparent in the output.

Further evidence that the Merton model fails to capture some inherent market perception of risk is visible in results for sub-investment grade bonds, for which the model performs very poorly, see Table (5.5). We repeat the observations of Li and Wong and note that a small standard deviation coupled with a wrong mean value suggests a strong bias in the results. The Merton model average yield error is nearly five times greater for speculative grade debt than for investment grade. The standard deviation of estimates for speculative grade bonds is approximately double that of their investment grade counterparts.

On the basis of these observations, we suggest that neither asset parameter estimation technique produces results accurate enough to justify use of the Merton model in practical applications. The model severely overprices lower grade debt and for investment grade issues is barely differentiable from a naive approach pricing the bond directly from the risk-free yield curve.

Table 5.4: Performance of Merton [1974] model on individual bonds at date of issue

	Pricing Date	Actual		Pure Proxy				MLE			
		Price	Yield	Price	Yield	% yield error	yield diff %	Price	Yield	% yield error	yield diff
Anixter Inc.	28/02/05	1.008	5.76	1.097	4.66%	-19.15%	-1.10%	1.097	4.66%	-19.15%	-1.10%
Baldor Electric Comp.	29/01/07	1.032	7.94	1.236	5.41%	-31.90%	-2.53%	1.236	5.41%	-31.87%	-2.53%
Berry Petroleum Comp.	20/10/06	1.004	7.99	1.215	5.33%	-33.22%	-2.65%	1.215	5.34%	-33.18%	-2.65%
Blount Inc.	6/08/04	1.023	8.31	1.293	4.43%	-46.66%	-3.88%	1.292	4.44%	-46.57%	-3.87%
Coca-Cola Comp.	31/10/07	0.994	5.33	1.021	4.99%	-6.45%	-0.34%	1.021	4.99%	-6.45%	-0.34%
Comstock Resources Inc.	20/02/04	1.010	6.57	1.180	4.13%	-37.05%	-2.43%	1.180	4.13%	-37.04%	-2.43%
Dean Foods Comp.	15/05/06	0.990	6.96	1.087	5.71%	-17.97%	-1.25%	1.087	5.71%	-17.97%	-1.25%
Millicom Intl. Cellular SA	20/11/03	1.013	9.51	1.426	4.55%	-52.18%	-4.96%	1.422	4.59%	-51.71%	-4.92%
Public Service Ent. Grp.	20/08/04	1.000	4.95	1.019	4.71%	-4.87%	-0.24%	1.019	4.71%	-4.87%	-0.24%
Rockwell Collins Inc.	20/11/03	1.004	4.63	0.999	4.69%	1.41%	0.07%	0.998	4.69%	1.44%	0.07%
Schlumberger Tech. Corp.	16/04/02	1.006	6.31	1.034	5.95%	-5.74%	-0.36%	1.034	5.95%	-5.64%	-0.36%
SK Telecom Comp. Ltd.	26/03/04	0.987	4.40	1.034	3.65%	-17.05%	-0.75%	1.034	3.65%	-17.04%	-0.75%
W&T Offshore Inc.	20/06/07	1.000	8.11	1.137	5.77%	-28.88%	-2.34%	1.136	5.79%	-28.58%	-2.32%
WCA Waste Corp.	6/12/06	1.083	7.59	1.256	5.02%	-33.91%	-2.57%	1.253	5.05%	-33.50%	-2.54%
Average						-23.83%	-1.81%			-23.72%	-1.80%
Std. Deviation						16.43%	1.49%			16.34%	1.48%

Table 5.5: Percentage error in yields of models and estimation techniques at date of issue, grouped by investment grade

Model	Pure Proxy		MLE	
	Investment Grade	Speculative Grade	Investment Grade	Speculative Grade
Merton				
Mean	-6.54%	-33.44%	-6.51%	-33.29%
Std Dev	6.66%	11.21%	6.66%	11.12%
Longstaff and Schwartz				
Mean	7.90%	-11.02%	14.28%	-4.02%
Std Dev	18.34%	24.97%	23.86%	28.14%

Table of the average empirical performance of each bond pricing model with asset dynamics parameters estimated using the pure proxy and MLE techniques and grouped by investment grade. Bonds with a Moody's investment grade of Baa or higher are considered investment grade debt, anything below is speculative grade. Each bond yield is estimated on the day soonest on or after the date of issue and compared with the observed price. For each model and estimation technique percentage error in yields in $\frac{\text{predicted yield - actual yield}}{\text{actual yield}}$.

5.5 Performance of the Longstaff and Schwartz [1995] model

We again refer to Figure (5.4) and Tables (5.2), (5.3), (5.5) and (5.6) to demonstrate the performance of the Longstaff and Schwartz [1995] model. In contrast to the Merton [1974] model, Longstaff and Schwartz predicts yields noticeably higher than the equivalent risk-less rate, implemented under either parameter estimation technique. In all cases, the MLE method predicts a higher yield than the pure proxy method.

The Longstaff and Schwartz model manages to both over and under predict yields depending on the investment grade of the debt. Appendix (E) and Table (5.5) show that for speculative grade debt the model manages to outperform Merton, but still consistently under-predicts the yield. For speculative debt, the Longstaff and Schwartz model has average percentage errors of -11.02% and -4.01% for the pure proxy and MLE techniques. These averages are respectively three and eight times more accurate than the equivalent output from the Merton, however this does come at the cost of a larger standard deviation.

Somewhat worryingly however, the Longstaff and Schwartz model performs very erratically for investment grade bonds. For example, predicted yields for the Public Services Enterprise

Group and Schlumberger Technology Corp. at times closely match true yields, but the predicted values spike wildly and in other instances are more than twice the true yield. The Longstaff and Schwartz average yield error of 14.28% is more than twice the Merton value of -6.51% for MLE estimates of investment grade bonds, and has a much larger standard deviation at 23.86% versus 6.66% respectively.

For speculative grade debt, maximum likelihood estimation greatly improves the model performance over a pure proxy approach, and both parameter estimation methods result in better performance than the Merton model on average. However, examining only the average performance obscures the fact that the Longstaff and Schwartz model manages to both greatly under- and over-predict market yields in different instances. Indeed, in some cases we witness both extremes when reviewing the historic yield estimates plotted over time for a single bond, such as for Baldor Electric Company and Berry Petroleum Company.

As with the Merton model, we find little to suggest that the Longstaff and Schwartz model is suitable for pricing risky corporate bonds in practice, regardless of which parameter estimation technique is used.

5.6 Summary of model performance and discussion of possible influencing factors

Our findings suggest that the maximum likelihood method results in estimates of asset volatility that are consistently higher than those that occur using a simple pure proxy approach. However, the difference between the pure proxy and MLE implied asset valuations is typically small relative to the total asset value. That is, viewing the value of equity as an option on assets rather than as the asset directly does not typically result in a large difference for the implied asset values. This is consistent with behavior for call options that are far in the money, i.e. when the asset value is already well above the option strike price, the value of the option closely matches the value of the underlying asset (less the strike price) anyway. As noted previously, the sample firms typically have high equity to liabilities ratios, suggesting that the equity-as-an-option has little risk and will be well in the money. As a consequence, any difference between the bond price predictions using pure proxy and MLE approaches will likely occur because of the different volatility estimates more so than any differences in implied asset valuation.

Table 5.6: Performance of Longstaff and Schwartz [1995] model on individual bonds at date of issue

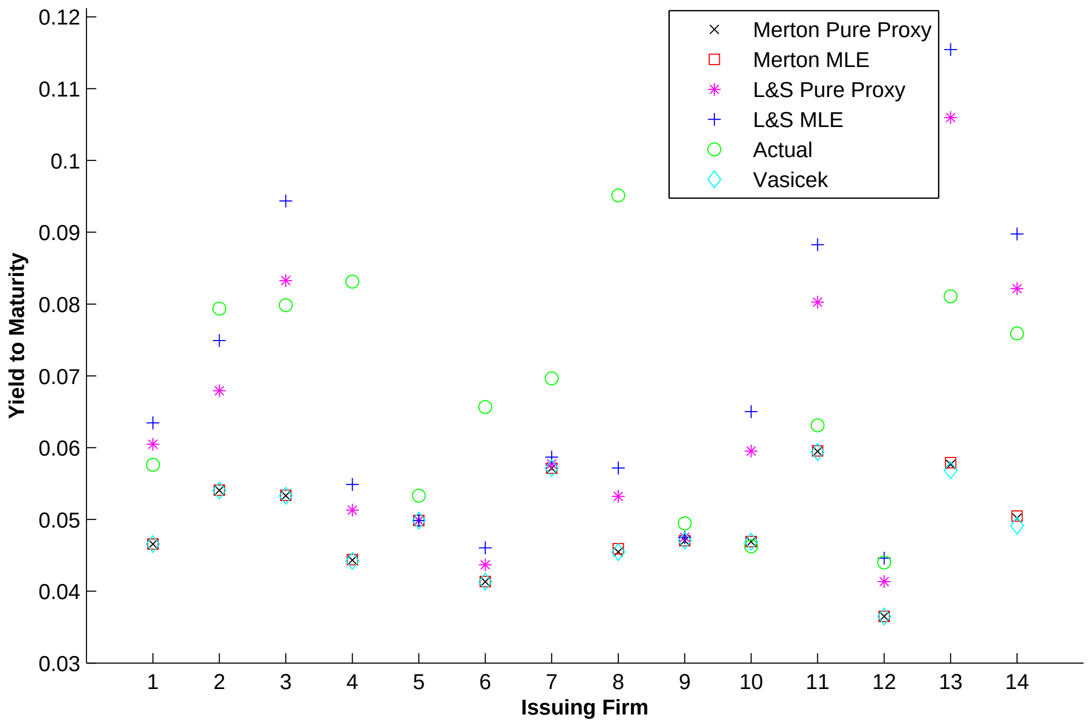
	Pricing Date	Actual		Pure Proxy				MLE			
		Price	Yield	Price	Yield	% yield error	yield diff %	Price	Yield	% yield error	yield diff %
Anixter Inc.	28/02/05	1.008	5.76%	0.986	6.05%	4.98%	0.29%	0.963	6.34%	10.13%	0.58%
Baldor Electric Comp.	29/01/07	1.032	7.94%	1.118	6.79%	-14.41%	-1.14%	1.064	7.49%	-5.61%	-0.45%
Berry Petroleum Comp.	20/10/06	1.004	7.99%	0.980	8.33%	4.30%	0.34%	0.907	9.44%	18.19%	1.45%
Blount Inc.	6/08/04	1.023	8.31%	1.239	5.13%	-38.29%	-3.18%	1.212	5.49%	-33.99%	-2.83%
Coca-Cola Comp.	31/10/07	0.994	5.33%	1.021	4.99%	-6.45%	-0.34%	1.021	4.99%	-6.45%	-0.34%
Comstock Resources Inc.	20/02/04	1.010	6.57%	1.162	4.37%	-33.46%	-2.20%	1.145	4.60%	-29.89%	-1.96%
Dean Foods Comp.	15/05/06	0.990	6.96%	1.082	5.77%	-17.10%	-1.19%	1.074	5.87%	-15.73%	-1.10%
Millicom Intl. Cellular SA	20/11/03	1.013	9.51%	1.350	5.32%	-44.07%	-4.19%	1.313	5.72%	-39.90%	-3.80%
Public Service Ent. Grp.	20/08/04	1.000	4.95%	1.016	4.75%	-3.95%	-0.20%	1.016	4.75%	-3.98%	-0.20%
Rockwell Collins Inc.	20/11/03	1.004	4.63%	0.902	5.95%	28.72%	1.33%	0.864	6.50%	40.56%	1.88%
Schlumberger Tech. Corp.	16/04/02	1.006	6.31%	0.886	8.03%	27.19%	1.72%	0.836	8.83%	39.90%	2.52%
SK Telecom Comp. Ltd.	26/03/04	0.987	4.40%	1.004	4.14%	-6.02%	-0.27%	0.984	4.46%	1.38%	0.06%
W&T Offshore Inc.	20/06/07	1.000	8.11%	0.875	10.60%	30.68%	2.49%	0.832	11.55%	42.39%	3.44%
WCA Waste Corp.	6/12/06	1.083	7.59%	1.045	8.22%	8.23%	0.62%	1.001	8.98%	18.24%	1.38%
Average						-4.26%	-0.42%			2.52%	0.05%
Std. Deviation						23.99%	1.85%			27.30%	2.03%

In contrast to the empirical results of Li and Wong [2008], we do not find that the MLE method achieves any significant improvement to the Merton [1974] model when considering average performance across the entire sample. We do, however, see that prediction accuracy varies greatly for individual bonds and also when comparing results aggregated by investment grade. The Merton model always drastically underestimates yields for speculative grade bonds, and in fact underestimates yield for any bond with an observed yield that is noticeably greater than an equivalent risk-free bond yield. The higher MLE volatility estimates do not appear overly critical to the Merton model, possibly because the model only considers default at maturity and so intermediate dips of asset value below the bond face value will not affect risk estimates.

Higher volatility estimates produced by the MLE technique do have considerable effect on the Longstaff and Schwartz [1995] model. Because this model allows default to occur any time that asset value fluctuates below liabilities value, default will be more likely than under the Merton model. Unfortunately, while the Longstaff and Schwartz model performance is improved on average by MLE, neither pure proxy nor MLE approach succeeds in making the model consistently accurate enough to price individual bonds. In fact, MLE appears to make the model considerably less accurate when pricing investment grade debt.

Finally, it is interesting to note that market interest rates were at historically low levels over the sample time period. Additionally, from mid 2007 onwards the market witnessed marked increases in yields. Widespread defaults, even on apparently high grade debt, suggests that the market had previously under-priced credit risk, and this would of course affect our perceptions of the models' performance.

Figure 5.4: Predicted yields by bond model and parameter estimation technique for sample firms priced at date of bond issue



1	Anixter Inc.	8	Millicom Intl. Cellular SA
2	Baldor Electric Comp.	9	Public Service Ent. Grp.
3	Berry Petroleum Comp.	10	Rockwell Collins Inc.
4	Blount Inc.	11	Schlumberger Tech. Corp.
5	Coca-Cola Comp.	12	SK Telecom Comp. Ltd.
6	Comstock Resources Inc.	13	W&T Offshore Inc.
7	Dean Foods Comp.	14	WCA Waste Corp.

Scatter plot of the actual and predicted yields on date of issue for each individual firm in the sample. The Merton [1974] and Longstaff and Schwartz [1995] models for both estimation techniques are plotted, as well as the implied yield for an otherwise equivalent risk-less bond priced off the fitted Vasicek [1977] yield curve.

Chapter 6

Conclusion

6.1 Summary of project

Structural models have been in existence since the early 1970s, and in that time have undergone extensive study and revision. Model input parameters representing asset value and volatility have often been estimated using a simple pure proxy approach, a method which has been noted to be upwardly biased and so underestimates risk. This paper implements the recently proposed maximum likelihood technique to better estimate these parameters. We then compare the difference between values produced by the MLE approach and the pure proxy approach. We discuss possible shortcuts that might reduce the need to perform the computationally intensive MLE technique for asset estimations in future.

We then perform empirical tests to quantitatively establish whether two structural models, those of Merton [1974] and Longstaff and Schwartz [1995], are improved by the use of MLE to the point where they may be of practical use in predicting market yields for risky corporate bonds.

6.2 Summary of results

Our results suggest a strong correlation between the asset volatility estimates produced by the pure proxy and MLE approaches. Results strongly indicate a linear relationship with $R^2 \geq 0.98$ and a scaling factor such that

$$\sigma^{MLE} \approx 1.12\sigma^{PP} \tag{6.1}$$

This suggests that we may be able to circumvent the need for lengthy and computationally intensive MLE calculations for each firm, and instead apply a scaling factor to the easily calculated pure proxy values, without sacrificing estimation performance. Further research is required to establish the validity of this approach and the conditions under which this potential shortcut might be applicable. The question then remains whether or not this is in fact worthwhile, if the bond models themselves still contain some critical flaw even when implemented with the theoretically appealing MLE technique.

We test the performance of the Merton [1974] and Longstaff and Schwartz [1995] structural bond models using both pure proxy and MLE parameter estimates, and find that overall, the average performance of each structural model is at least marginally improved by using MLE.

However, problems appear when we examine yield predictions at the individual bond level, with both radical under- and over-predictions evident for both bond models. Additionally, we witness a marked difference in the performance of each bond model depending on the investment rating of the bond. Large under- and over-predictions of bond yields appear when comparing aggregate performance for investment versus speculative grade debt.

The Merton model fails to predict bond yields in any instance where the market yield is significantly different to the implied risk-free yield. The Longstaff and Schwartz model sometimes closely predicts the market yield but in other instances is wildly inaccurate. The failure of both models to consistently and accurately predict market yields, regardless of which parameter estimation technique is used, suggests some deeper flaw that cannot be explained away solely by problems in the parameter estimation process. In a majority of instances, the two structural models appear unable to capture some perception of risk that is being priced in by the market. This is especially concerning on top of our earlier observations that for the sample period, the bond pricing models under-predict yields and so under-price credit risk. Given that this occurs during a period in which it is widely believed that the market itself was already underpricing risk, serious doubts are raised about the usefulness of the models under more typical market conditions.

We therefore conclude that despite their theoretically appealing nature and promising earlier studies suggesting broadly accurate performance, the Merton and Longstaff and Schwartz structural models remain unsuitable for pricing individual bonds in real world scenarios.

6.3 Contributions / Recommendations

This thesis makes several contributions to existing literature.

We provide further evidence that maximum likelihood estimation increases estimates of asset volatility and lowers estimates of asset value, thereby increasing the probability of default under each model. This helps increase the predicted yields, bringing them more in line with observed values. We also find strong evidence of a linear relationship between pure proxy and maximum likelihood asset parameter estimates, which may in turn lead to further computational shortcuts similar in nature to Equation (6.1).

We note the failure of the Merton [1974] and Longstaff and Schwartz [1995] structural models to accurately predict market yields for risky corporate bonds. This is true regardless of whether asset dynamics are estimated via the pure proxy approach or the maximum likelihood estimation technique as suggested by Duan [1995].

Finally, we present strong evidence that the error in the bond model output is in some way related to their failure to capture market perceptions of risk. This belief is supported by the observed differences in the average predictive power of the models when bonds are grouped by investment ratings.

Based on our various findings, we propose several topics that may provide interesting material for further study. Promising future research could include determining which factors (such as maturity, bond rating, coupon rate) influence the scaling factor for the conversion between pure proxy and MLE estimates. It may be beneficial to repeat this experiment to determine whether the average behaviours of the bond models as observed in this study are true for different time periods, different bond ratings, different markets and different market conditions.

We also suggest further attempts to implement newer structural bond pricing models, such as those of Briys and de Varenne [1997] and Collin-Dufresne and Goldstein [2001], and test their performance against observed market bond prices. A test of whether these newer models manage to more accurately predict market yields using the MLE technique over the time period and bonds included in this study might determine how well the weaknesses in earlier structural models have been addressed.

Finally, and perhaps most interestingly, we suggest a study to test whether the various structural bond models accurately predict default rates. That is, establishing whether errors in predicted yields are the result of the models under-predicting the likelihood of default, or alternatively, are the result of markets actually overpricing risk.

Bibliography

Viral Acharya, Jing zhi Huang, Marti Subrahmanyam, and Rangarajan Sundaram. Costly financing, optimal payout policies and the valuation of corporate debt. 2000.

Edward I. Altman. Financial ratios, discriminant analysis and the prediction of corporate bankruptcy. *The Journal of Finance*, 23(4):589–609, 1968.

Ronald Anderson and Suresh Sundaresan. A comparative study of structural models of corporate bond yields: An exploratory investigation. *Journal of Banking and Finance*, 24:255–269, 2000.

Ronald W. Anderson and Suresh Sundaresan. Design and valuation of debt contracts. *The Review of Financial Studies*, 9(1):37–68, 1996.

Ronald W. Anderson, Suresh Sundaresan, and Pierre Tychon. Strategic issues in financial economics: Strategic analysis of contingent claims. *European Economic Review*, 40:871–881, 1996.

Dervis Bayazit. Yield curve estimation and prediction with vasicek model. Master's thesis, Graduate School of Applied Mathematics The Middle East Technical University, 2004.

Fischer Black and John C. Cox. Valuing corporate securities: Some effects of bond indenture provisions. *The Journal of Finance*, 31(2):351–367, 1976.

Fischer Black and Myron Scholes. The pricing of options and corporate liabilities. *The Journal of Political Economy*, 81(3):637–654, 1973.

Eric Briys and Francois de Varenne. Valuing risky fixed rate debt: An extension. *Journal of Financial and Quantitative Analysis*, 32(2):239–248, 1997.

George C. Chacko, Anders Sjöman, Hideto Motohashi, and Vincent Dessain. *Credit Derivatives: A Primer on Credit Risk, Modeling, and Instruments.* Wharton School Publishing, 2006.

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

Tsz Wang Choi and Hoi Ying Wong. Estimating default barriers from market information. First draft: April 2004, 2006.

Pierre Collin-Dufresne and Robert S. Goldstein. Do credit spreads reflect stationary leverage ratios? *The Journal of Finance*, 56(5):1929–1957, 2001.

Emanuel Derman. The principles and practice of verifying derivatives prices. Working Paper Series. Available at SSRN: http://ssrn.com/abstract=267682 or DOI: 10.2139/ssrn.267682, 2001.

Emanuel Derman. The boy's guide to pricing and hedging. Working Paper Series. Available at SSRN: http://ssrn.com/abstract=364760 or DOI: 10.2139/ssrn.364760, 2002.

Jin-Chuan Duan. Maximum likelihood estimation using price data of the derivative contract. *Mathematical Finance*, 4(2):155–167, 1995.

Jin-Chuan Duan, Genevi'eve Gauthier, Jean-Guy Simonato, and Sophia Zaanoun. Estimating merton's model by maximum likelihood with survivorship consideration. 2003.

Gregory J Eidleman. Z scores - a guide to failure prediction. *The CPA Journal Online*, 1995.

Abel Elizalde. Credit risk models II: Structural models. Available: www.abelelizale.com, 2005.

Edwin J. Elton, Martin J. Gruber, Deepak Agrawal, and Christopher Mann. Explaining the rate spread on corporate bonds. *The Journal of Finance*, 56(1):247–277, 2001.

Young Ho Eom, Jean Helwege, and Jing-Zhi Huang. Structural models of corporate bond pricing: An empirical analysis. *The Review of Financial Studies*, 17(2):499–544, 2004.

Jan Ericsson and Joel Reneby. Estimating structural bond pricing models. *Journal of Business*, 78(2):707–735, 2005.

Eugene F. Fama. The behavior of stock-market prices. *The Journal of Business*, 38(1):34–105, 1965.

Hua Fan and Suresh M. Sundaresan. Debt valuation, renegotiation, and optimal dividend policy. *The Review of Financial Studies*, 13(4):1057–1099, 2000.

Robert Fortet. Les fonctions aleatoires du type de markoff associees a certainese equations lineaires aux derivees partielles du type parabolique. *Journal de Mathematiques Pures et Appliquees*, 22:177–243, 1947.

Julian R. Franks and Walter N. Torous. An empirical investigation of u.s. firms in reorganization. *The Journal of Finance*, 44(3):747–769, 1989.

Robert Geske. The valuation of corporate liabilities as compound options. *The Journal of Financial and Quantitative Analysis*, 12(4):541–552, 1977.

Dajiang Guo and David Guoming Wei. Pricing risky debt: An empirical comparison of the longstaff and schwartz and merton models. *The Journal of Fixed Income*, 7(2):8–28, 1997.

Jean Helwege and Christopher M. Turner. The slope of the credit yield curve for speculative-grade issuers. *Journal of Finance*, 54:1869–1884, 1999.

John C. Hull. *Options, Futures and Other Derivatives.* Pearson Education, 6th edition, 2006.

Robert A. Jarrow and Philip Protter. Structural versus reduced form models: A new information based perspective. *Journal of Investment Management*, 2(2):1–10, 2004.

Robert A. Jarrow, David Lando, and Stuart M. Turnbull. A markov model for the term structure of credit risk spreads. *The Review of Financial Studies*, 10(2):481–523, 1997.

E. Philip Jones, Scott P. Mason, and Eric Rosenfeld. Contingent claims analysis of corporate capital structures: An empirical investigation. *The Journal of Finance*, 39(3):611–625, 1984.

Hayne E. Leland. Corporate debt value, bond covenants, and optimal capital structure. *Journal of Finance*, 49(4):1213–1252, 1994a.

Hayne E. Leland and Klaus Bjerre Toft. Optimal capital structure, endogenous bankruptcy, and the term structure of credit spreads. *The Journal of Finance*, 51(3):987–1019, 1996.

Ka Leung Li and Hoi Ying Wong. Structural models of corporate bond pricing with maximum likelihood estimation. *Journal of Empirical Finance*, 15(15):751–777, 2008.

Francis A Longstaff and Eduardo S. Schwartz. A simple approach to valuing risky fixed and floating rate debt. *The Journal of Finance*, 50(3):789–819, 1995.

Robert C. Merton. Theory of rational option pricing. *The Bell Journal of Economics and Management Science*, 4(1):141–183, 1973.

Robert C. Merton. On the pricing of corporate debt: The risk structure of interest rates. *The Journal of Finance*, 29(2):449–470, 1974.

In Jae Myung. Tutorial on maximum likelihood estimation. *Journal of Mathematical Psychology*, 47:90–100, 2003.

Charles R. Nelson and Andrew F. Siegel. Parsimonious modeling of yield curves. *The Journal of Business*, 60(4):473–489, 1987.

Ehud I. Ronn and Avinash K. Verma. Pricing risk-adjusted deposit insurance: An option-based model. *Journal of Finance*, 41(4):871–895, 1986.

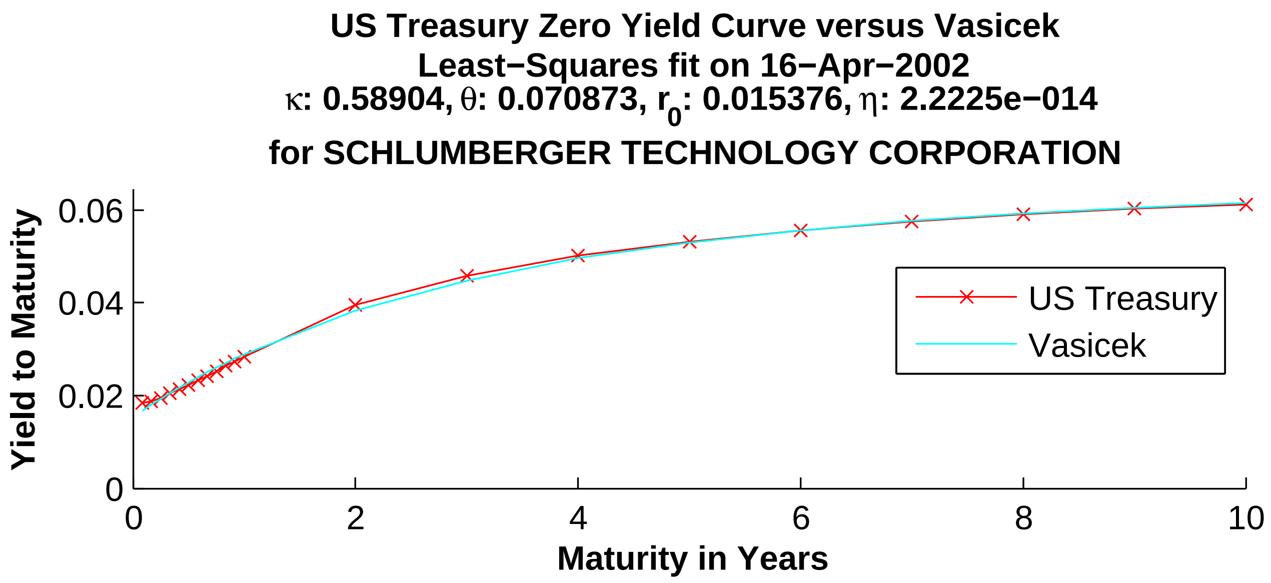
Richard Sweeney, Arthur Warga, and Drew Winters. The market value of debt, market versus book value of debt, and returns to assets. *Financial Management*, 26(1):5–21, 1997.

Oldrich Vasicek. An equilibrium characterization of the term structure. *Journal of Financial Economics*, 5(2):177–188, 1977.

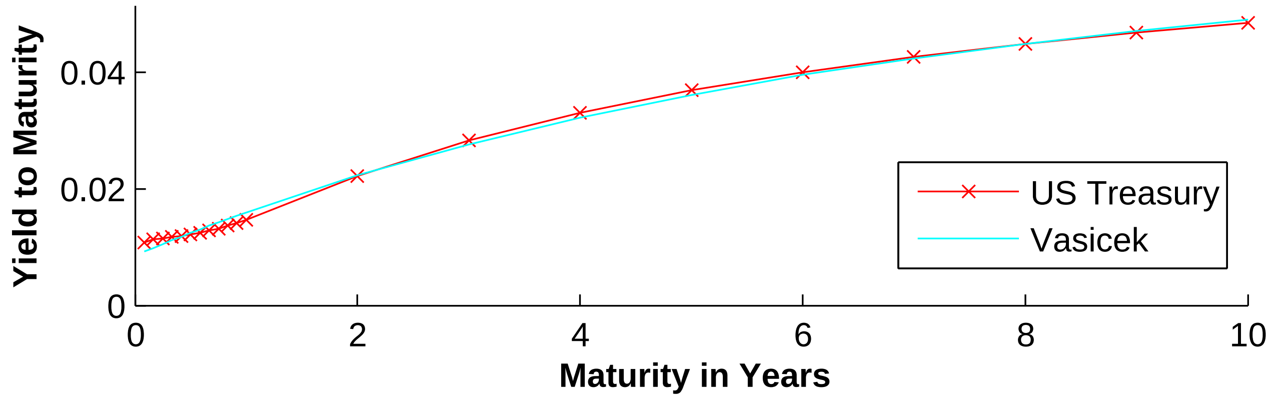
Appendix A

Plots of Vasicek interest rate curve fits for bond issue dates

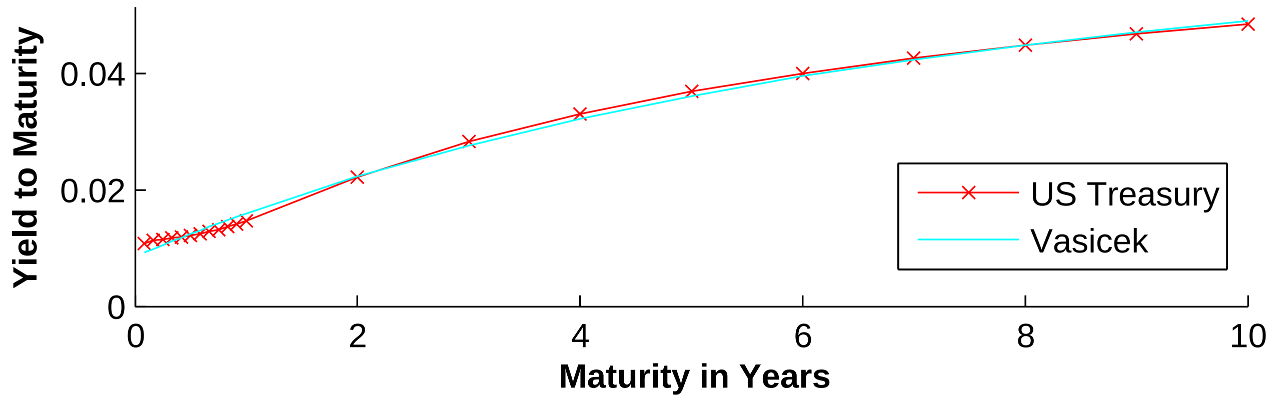
For each bond included in the sample, the Vasicek [1977] interest rate model is matched to the nearest day's term structure observations to predict the bond price on the date of issue. The graphs are arranged chronologically, and show the evolution of the term structure from a normal upward sloping yield curve that shows yields increasing with maturity, then slowly flattening with time, until it appears slightly inverted by the end of the sample period.



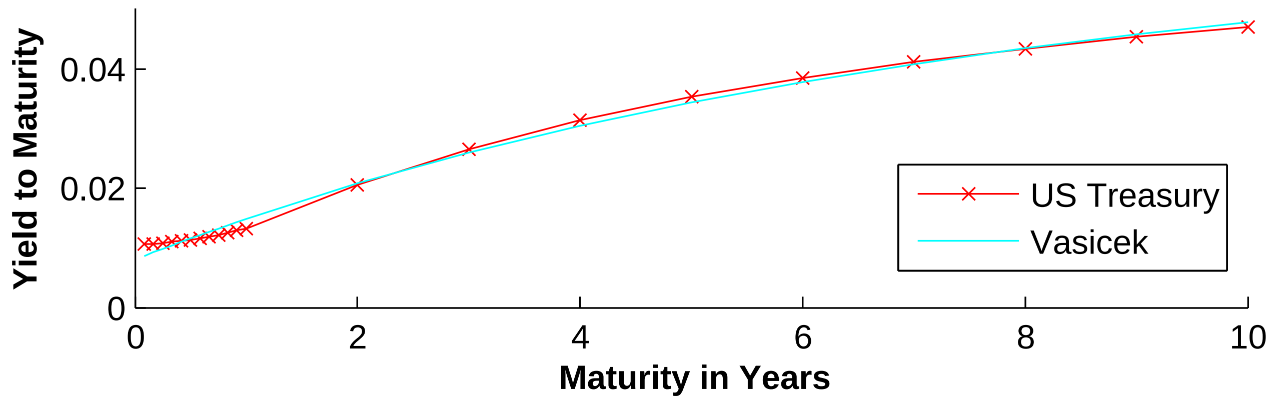
US Treasury Zero Yield Curve versus Vasicek
Least−Squares fit on 20−Nov−2003
κ: 0.25699, θ: 0.071686, r_0: 0.0085243, η: 6.7761e−010
for ROCKWELL COLLINS INCORPORATED



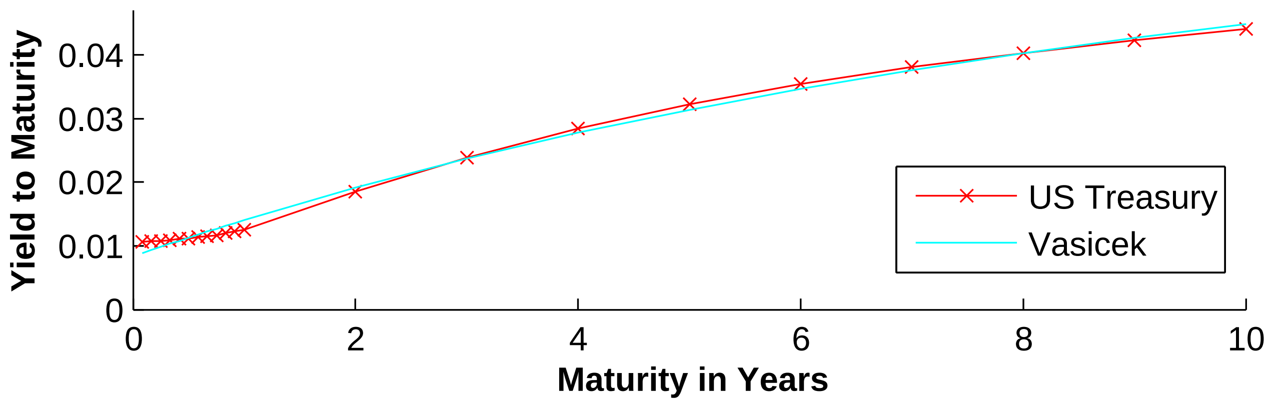
US Treasury Zero Yield Curve versus Vasicek
Least−Squares fit on 20−Nov−2003
κ: 0.25699, θ: 0.071686, r_0: 0.0085243, η: 6.7761e−010
for MILLICOM INTERNATIONAL CELLULAR SA



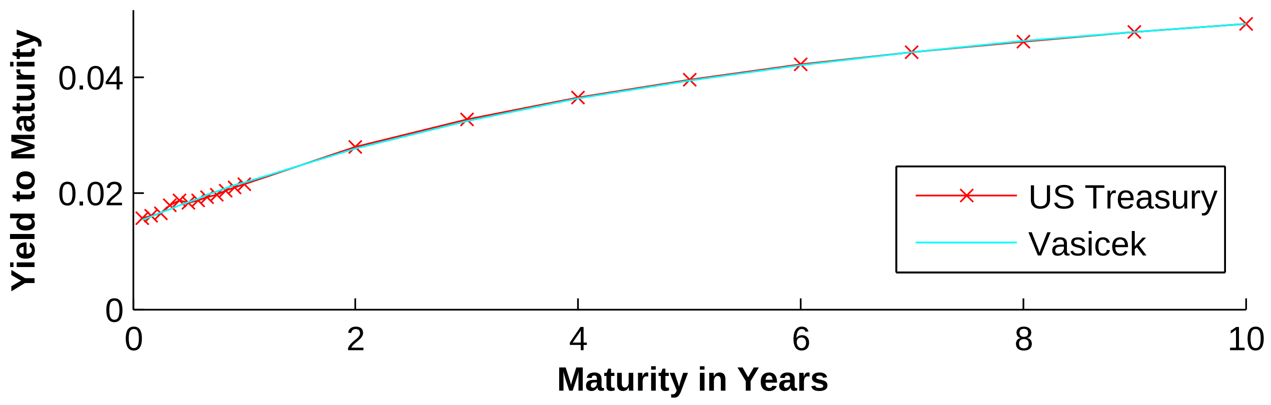
US Treasury Zero Yield Curve versus Vasicek
Least−Squares fit on 20−Feb−2004
κ: 0.2248, θ: 0.074163, r_0: 0.0080096, η: 3.0332e−007
for COMSTOCK RESOURCES INCORPORATED



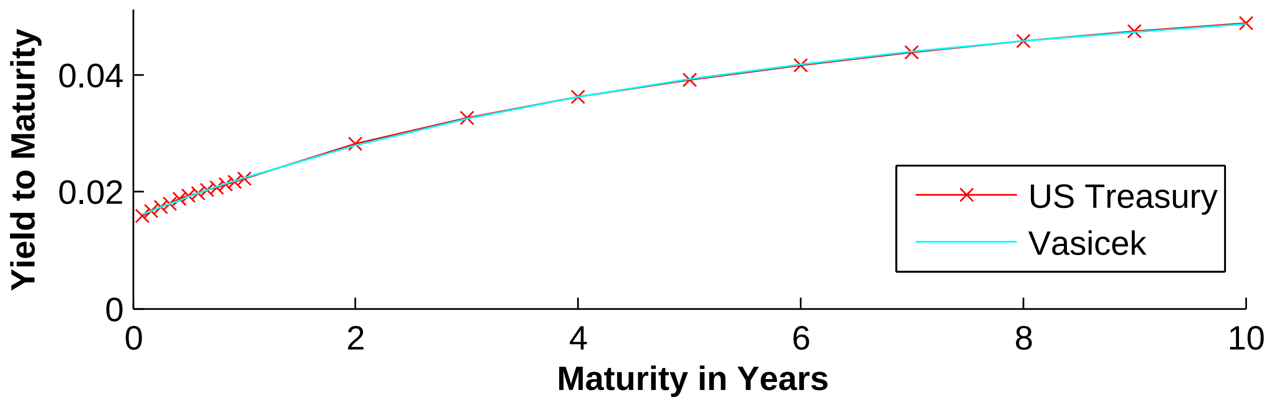
US Treasury Zero Yield Curve versus Vasicek
Least−Squares fit on 26−Mar−2004
κ: 0.17448, θ: 0.077633, r_0: 0.0083446, η: 3.1056e−014
for SK TELECOM COMPANY LIMITED

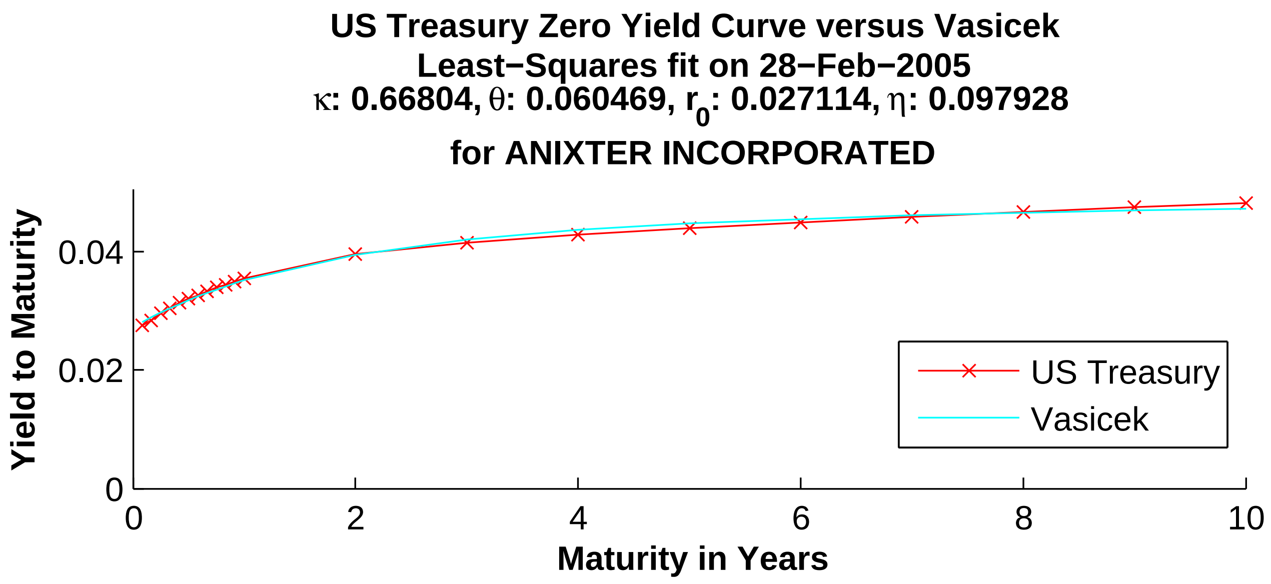


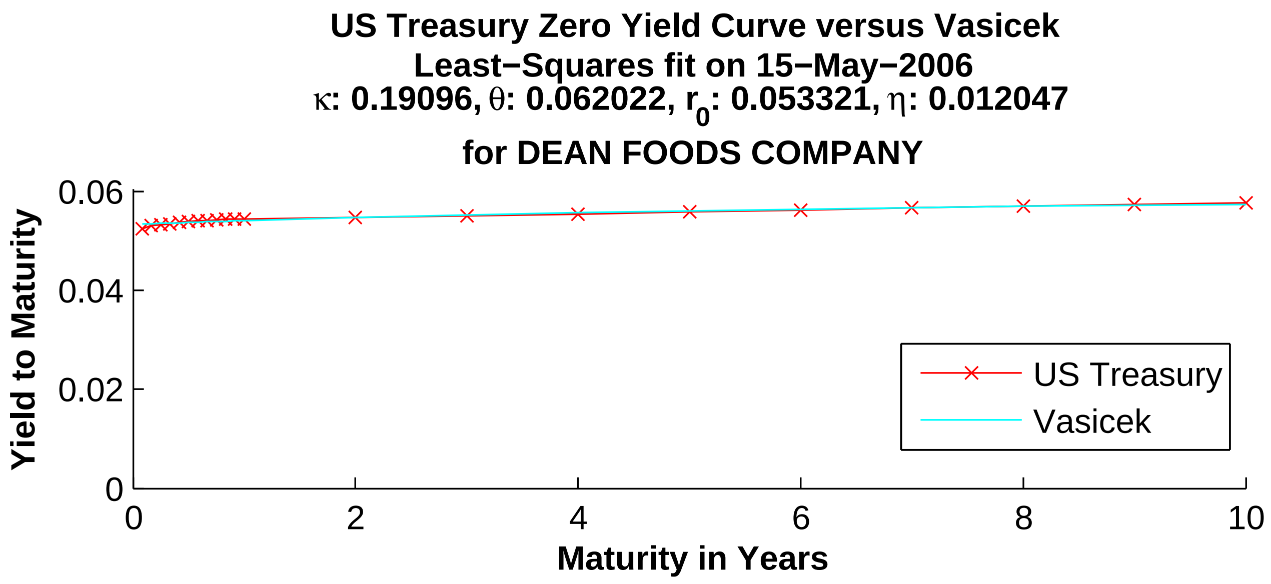
US Treasury Zero Yield Curve versus Vasicek
Least−Squares fit on 06−Aug−2004
κ: 0.32435, θ: 0.063958, r_0: 0.014774, η: 0.0064051
for BLOUNT INCORPORATED

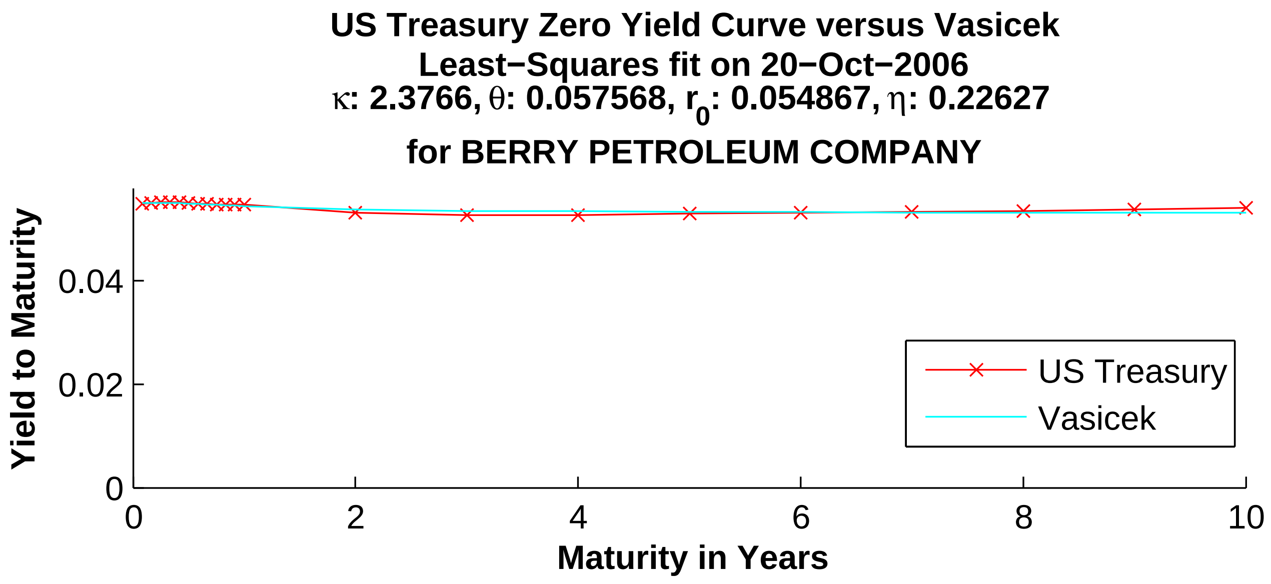


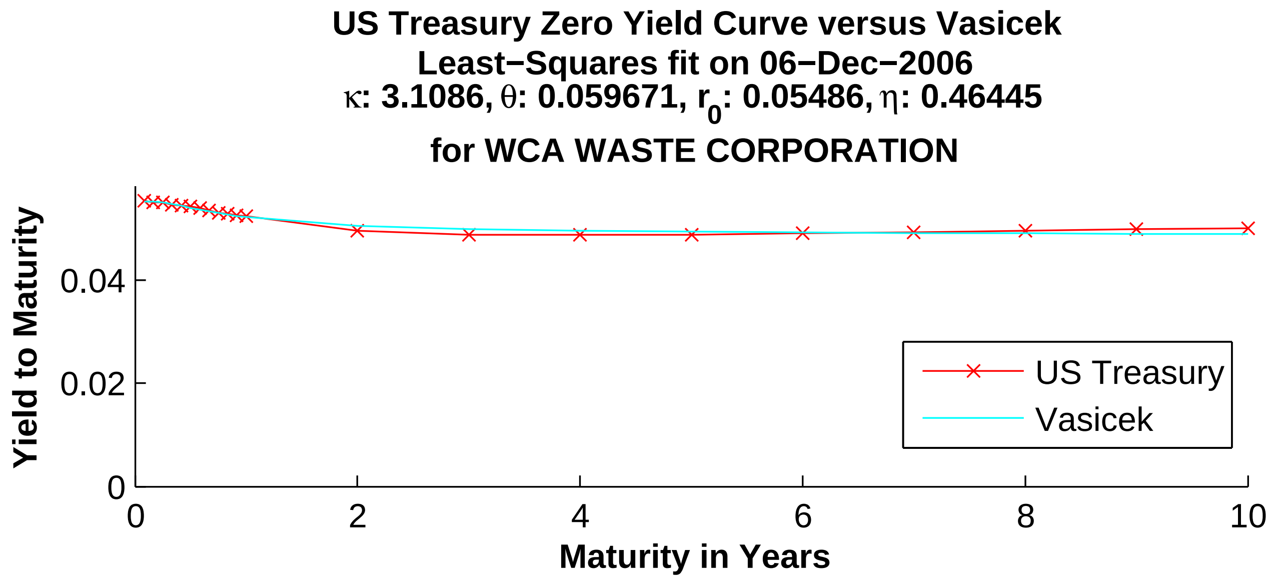
US Treasury Zero Yield Curve versus Vasicek
Least−Squares fit on 20−Aug−2004
κ: 0.29223, θ: 0.068399, r_0: 0.015482, η: 0.028736
for PUBLIC SERVICE ENTERPRISE GROUP INCORPORATED

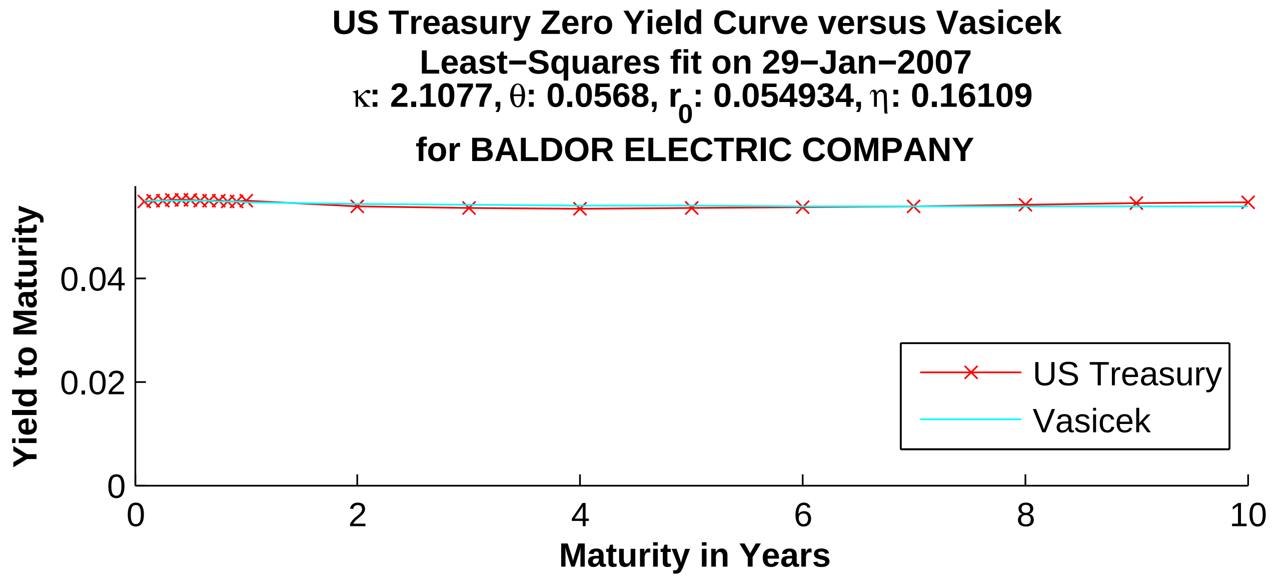


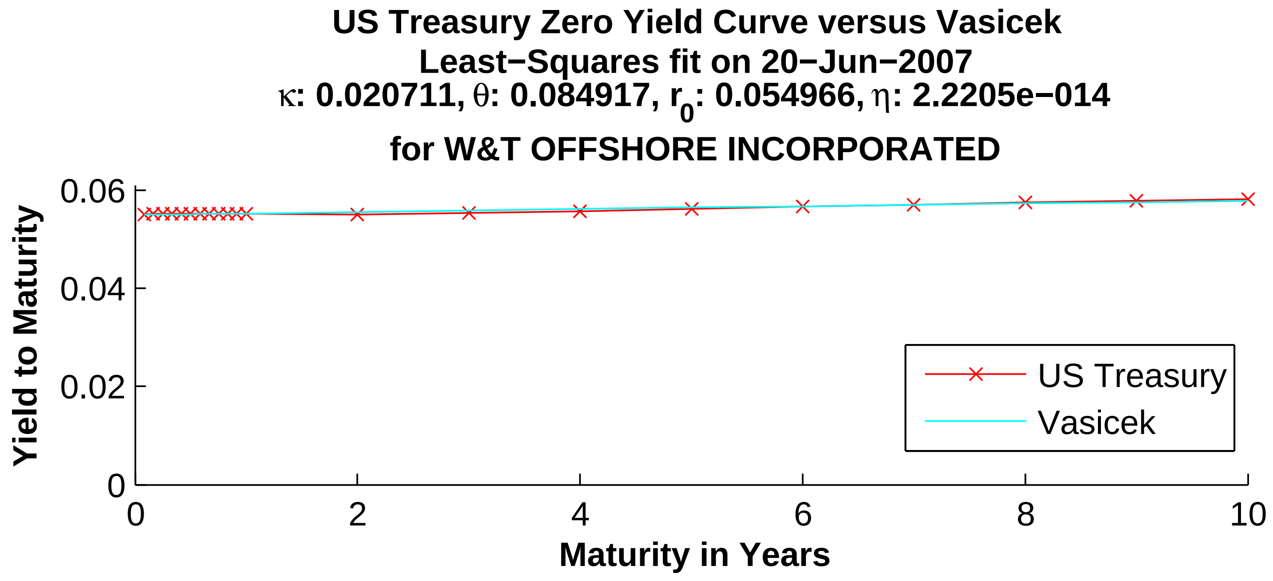

US Treasury Zero Yield Curve versus Vasicek
Least−Squares fit on 28−Feb−2005
κ: 0.66804, θ: 0.060469, r_0: 0.027114, η: 0.097928
for ANIXTER INCORPORATED
Yield to Maturity
0.04
0.02
0
Maturity in Years
0
2
4
6
8
10
US Treasury
Vasicek

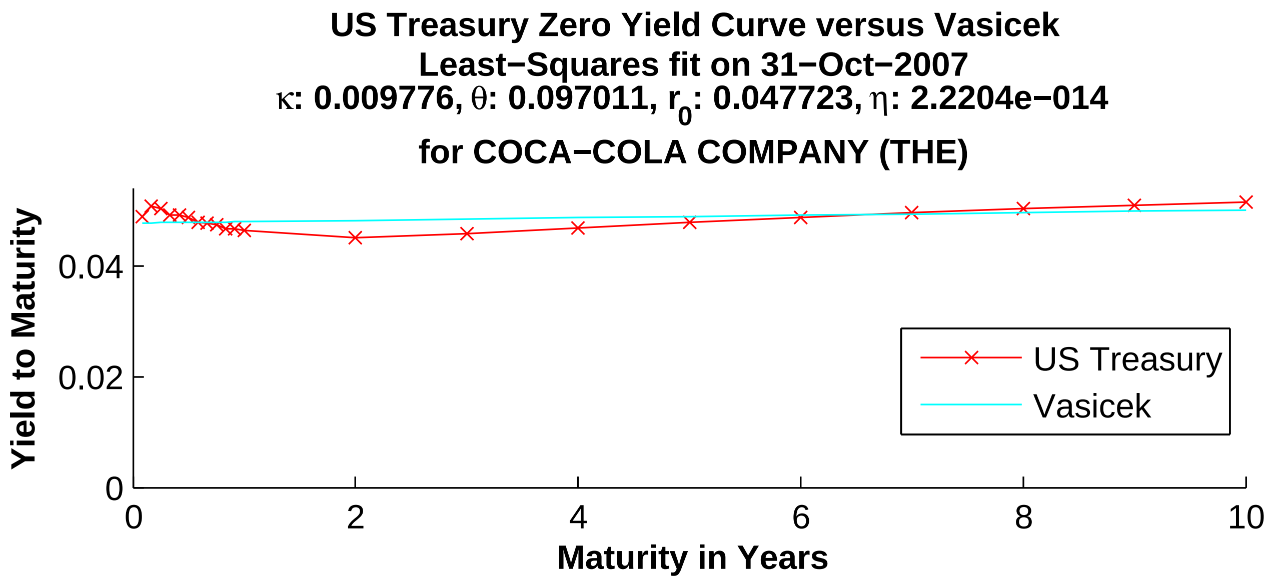

US Treasury Zero Yield Curve versus Vasicek
Least−Squares fit on 15−May−2006
κ: 0.19096, θ: 0.062022, r_0: 0.053321, η: 0.012047
for DEAN FOODS COMPANY
Yield to Maturity
0.06
0.04
0.02
0
Maturity in Years
0
2
4
6
8
10
US Treasury
Vasicek


US Treasury Zero Yield Curve versus Vasicek
Least−Squares fit on 20−Oct−2006
κ: 2.3766, θ: 0.057568, r_0: 0.054867, η: 0.22627
for BERRY PETROLEUM COMPANY
Yield to Maturity
0.04
0.02
0
Maturity in Years
0
2
4
6
8
10
US Treasury
Vasicek


US Treasury Zero Yield Curve versus Vasicek
Least−Squares fit on 06−Dec−2006
κ: 3.1086, θ: 0.059671, r0: 0.05486, η: 0.46445
for WCA WASTE CORPORATION
Yield to Maturity
0.04
0.02
0
0
2
4
6
8
10
Maturity in Years
US Treasury
Vasicek


US Treasury Zero Yield Curve versus Vasicek
Least−Squares fit on 29−Jan−2007
κ: 2.1077, θ: 0.0568, r0: 0.054934, η: 0.16109
for BALDOR ELECTRIC COMPANY
Yield to Maturity
0.04
0.02
0
0
2
4
6
8
10
Maturity in Years
US Treasury
Vasicek

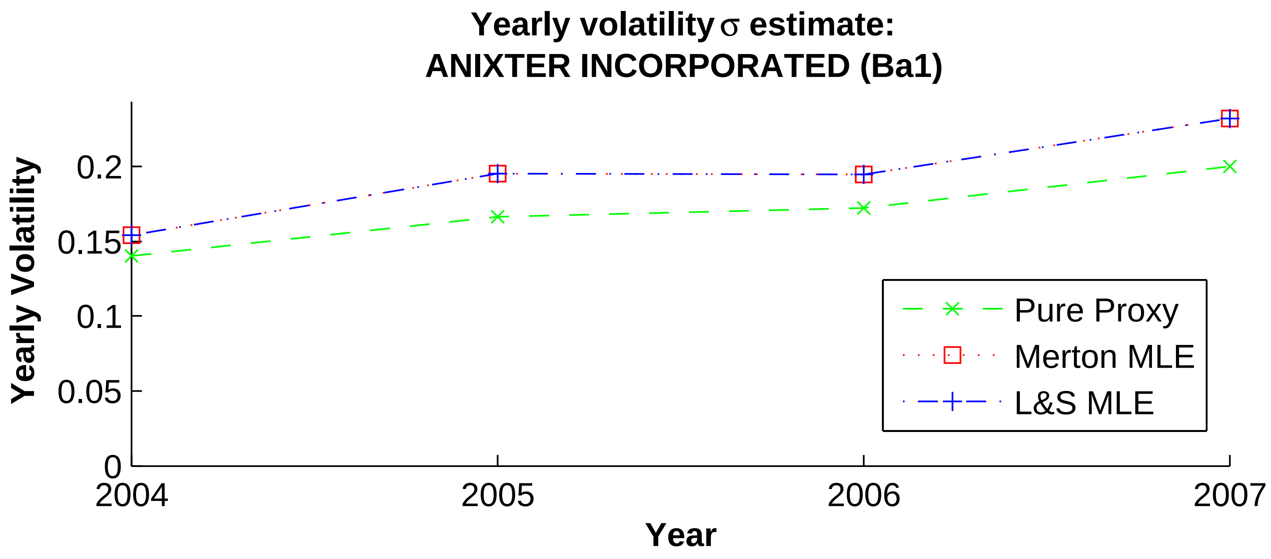

US Treasury Zero Yield Curve versus Vasicek
Least−Squares fit on 20−Jun−2007
κ: 0.020711, θ: 0.084917, r0: 0.054966, η: 2.2205e−014
for W&T OFFSHORE INCORPORATED
Yield to Maturity
0.06
0.04
0.02
0
0
2
4
6
8
10
Maturity in Years
US Treasury
Vasicek

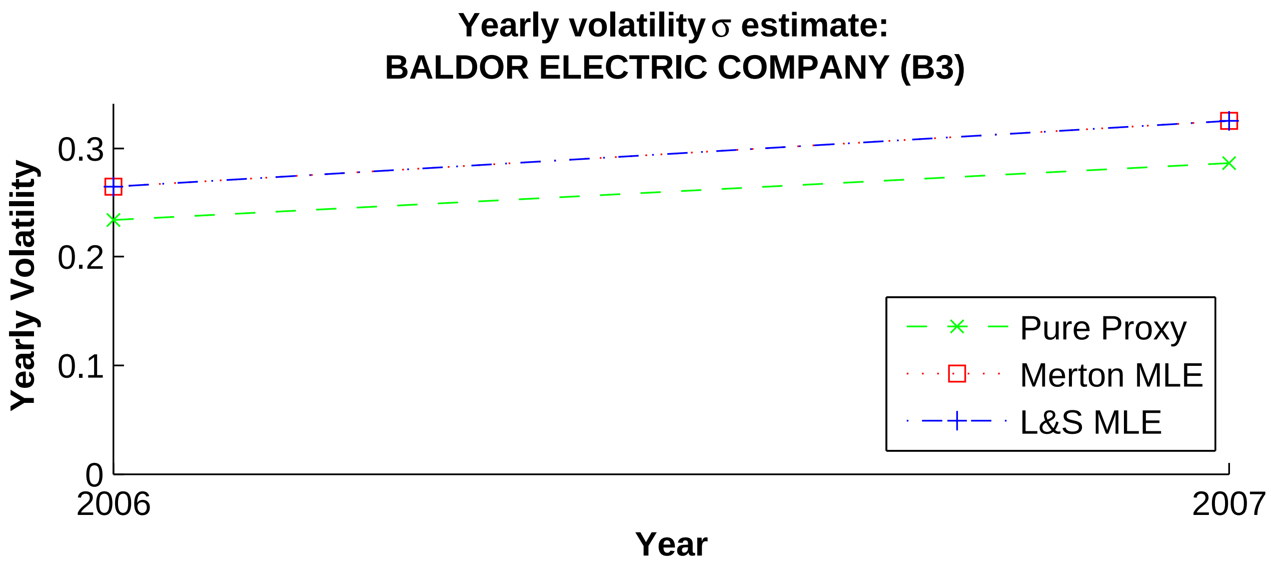

US Treasury Zero Yield Curve versus Vasicek
Least−Squares fit on 31−Oct−2007
κ: 0.009776, θ: 0.097011, r0: 0.047723, η: 2.2204e−014
for COCA−COLA COMPANY (THE)
Yield to Maturity
0
0.02
0.04
0
2
4
6
8
10
Maturity in Years
US Treasury
Vasicek

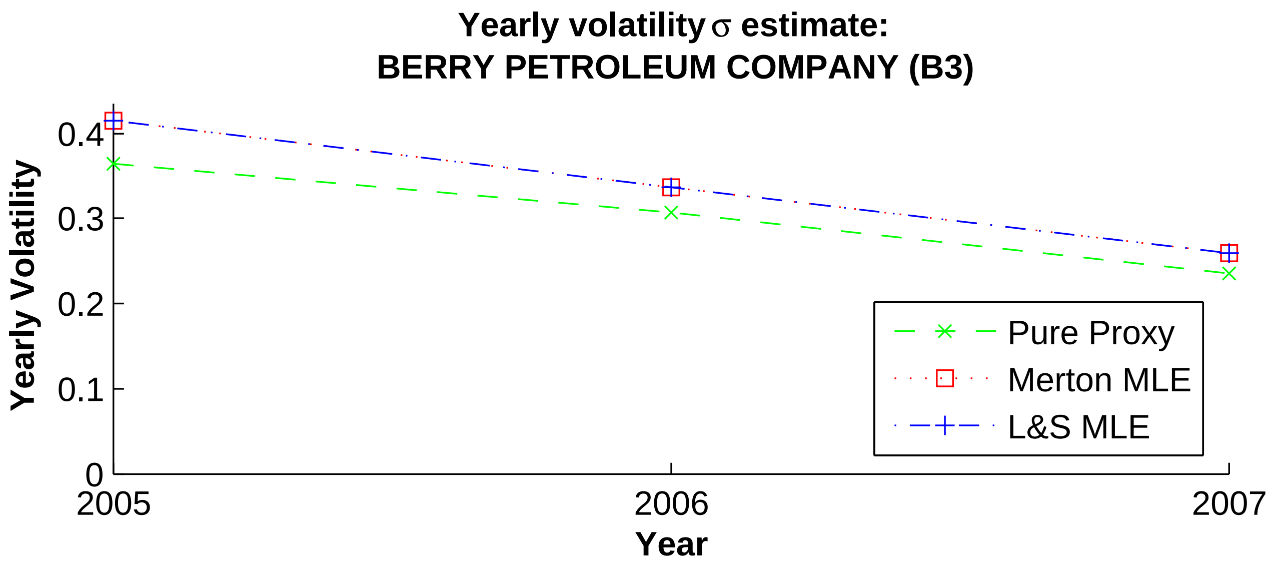
Appendix B

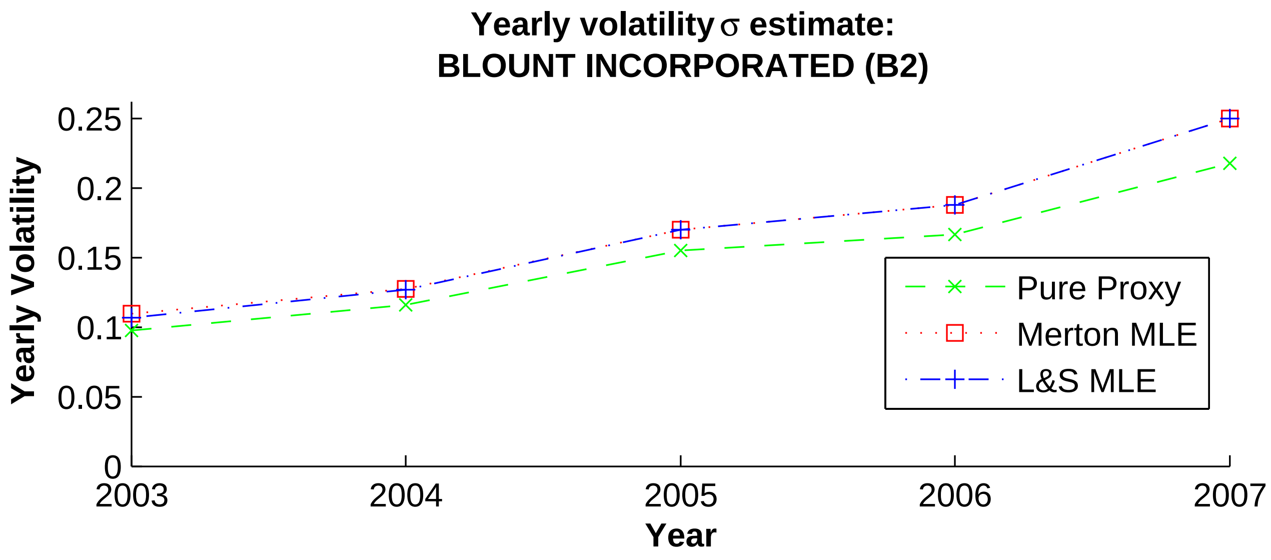
Plots of implied asset volatility estimations

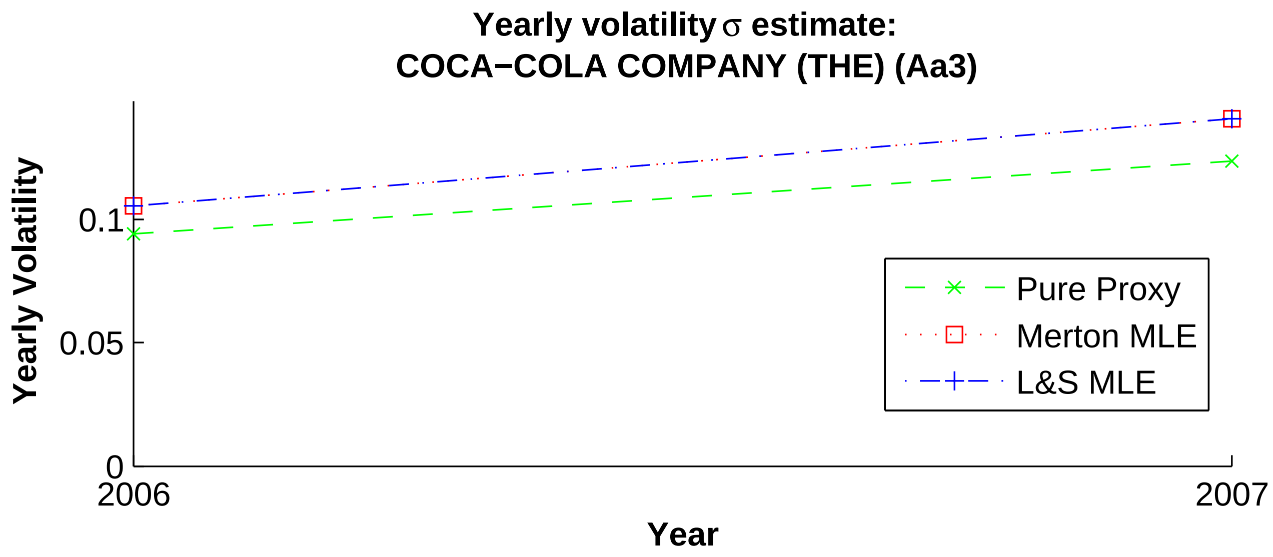
Graphs of each firm's yearly asset volatility as estimated using the pure proxy and maximum likelihood estimation methods. Firm asset dynamics, and in particular the volatility, are calculated on a yearly basis are assumed to be constant throughout the financial year. The firms in the sample each issue yearly balance sheet statements in December, so financial year is aligned with calendar year for this study.

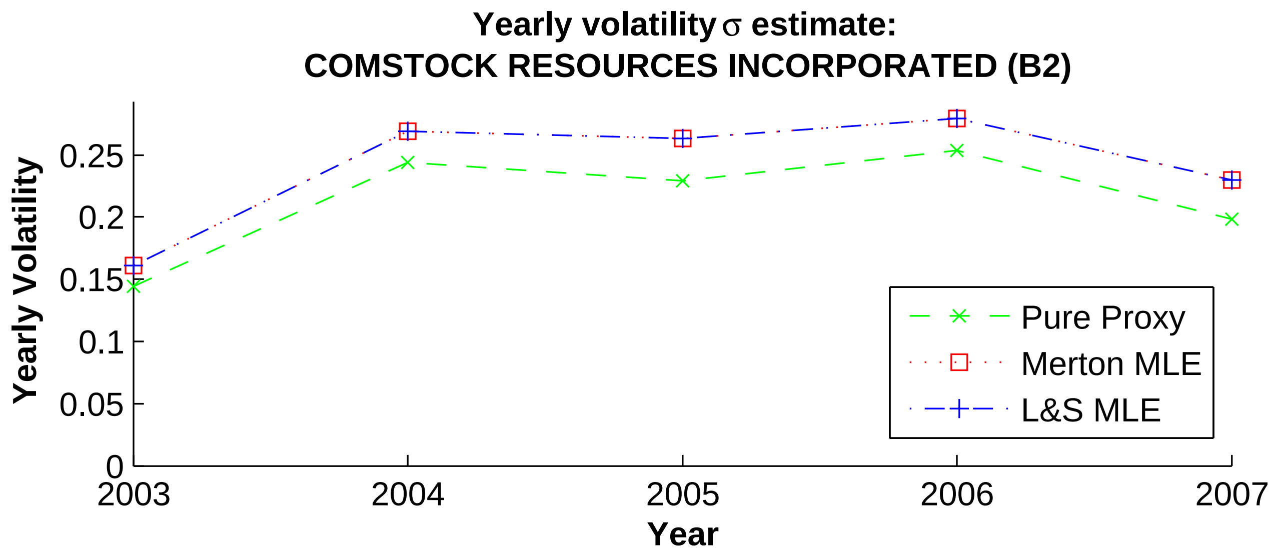


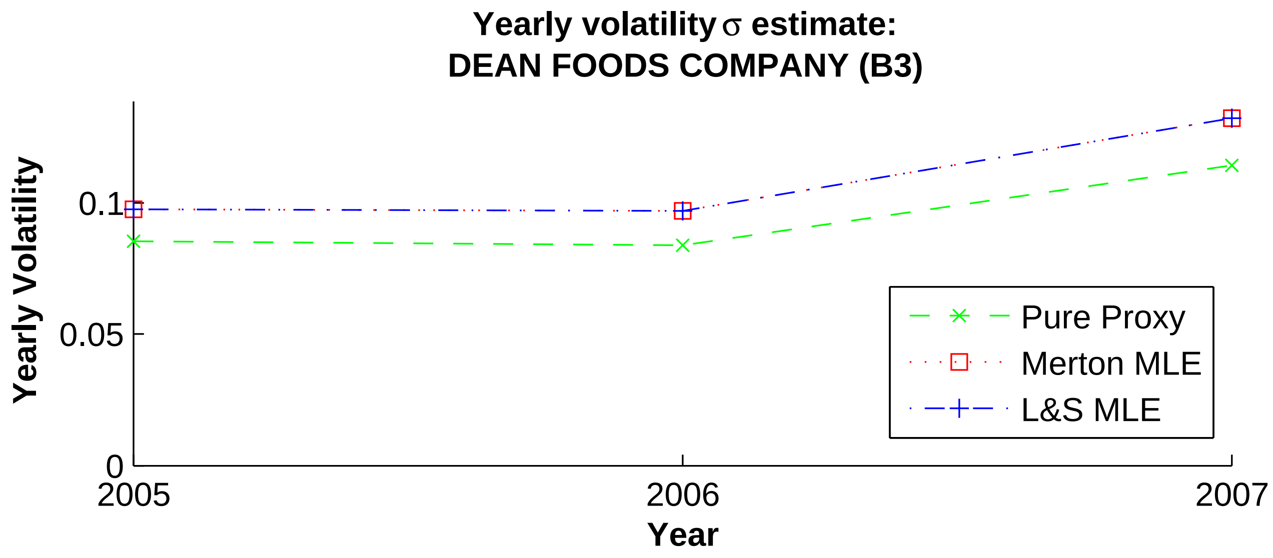

Yearly volatility σ estimate:
BALDOR ELECTRIC COMPANY (B3)
Yearly Volatility
0.3
0.2
0.1
0
2006
2007
Year
Pure Proxy
Merton MLE
L&S MLE

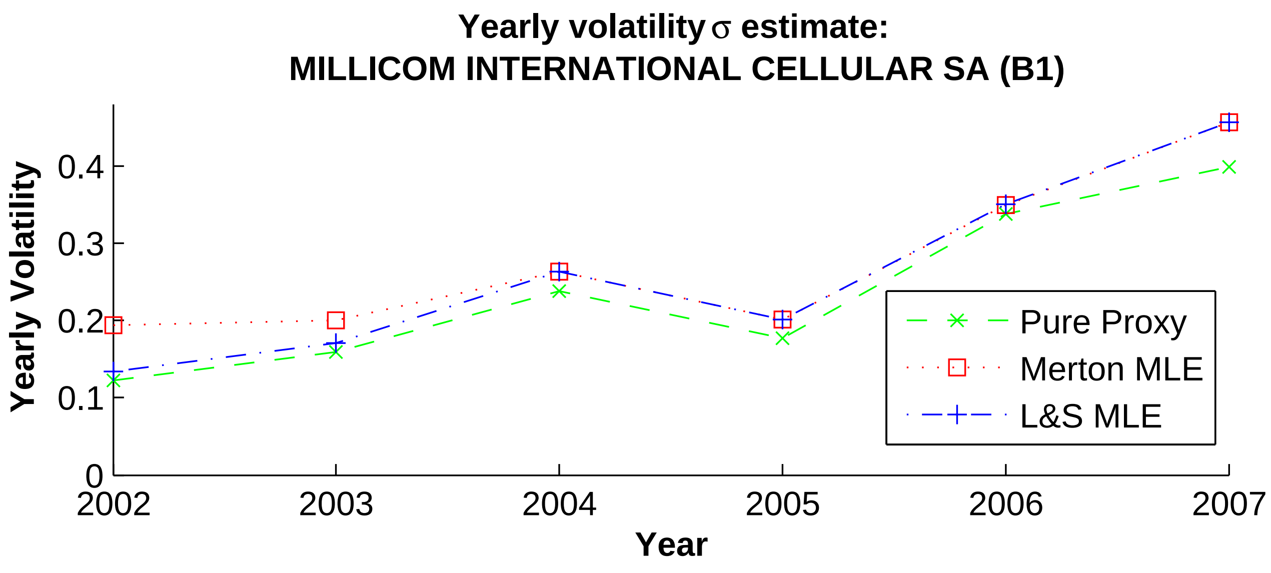

Yearly volatility σ estimate:
BERRY PETROLEUM COMPANY (B3)
Yearly Volatility
0.4
0.3
0.2
0.1
0
2005
2006
2007
Year
Pure Proxy
Merton MLE
L&S MLE

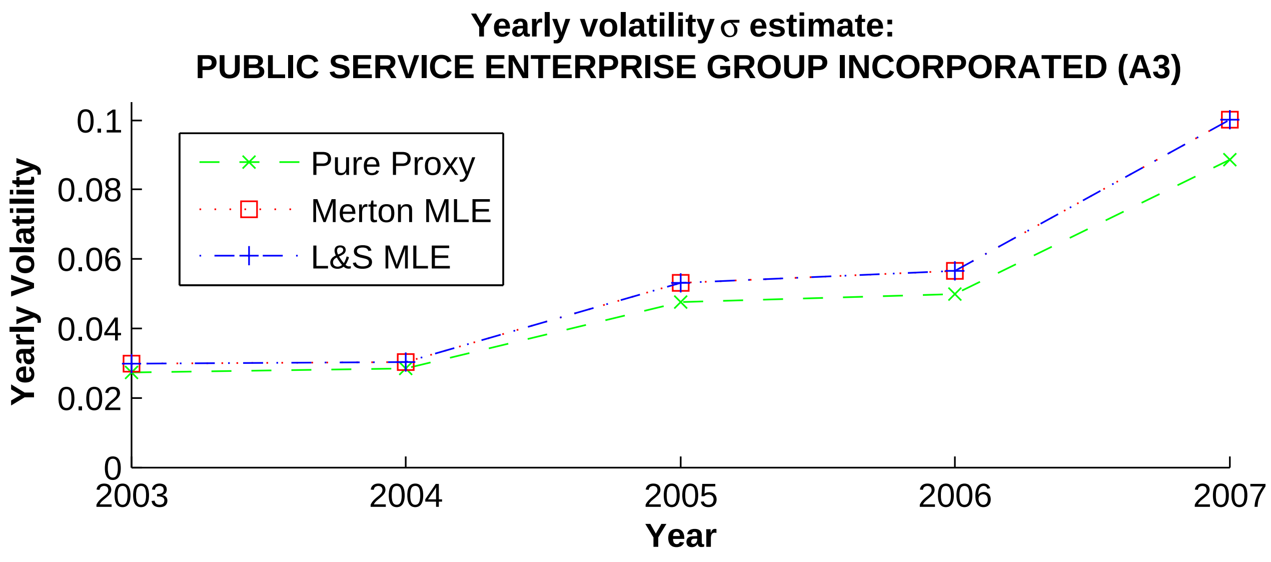

Yearly volatility σ estimate:
BLOUNT INCORPORATED (B2)
Yearly Volatility
0.25
0.2
0.15
0.1
0.05
0
2003
2004
2005
2006
2007
Year
Pure Proxy
Merton MLE
L&S MLE

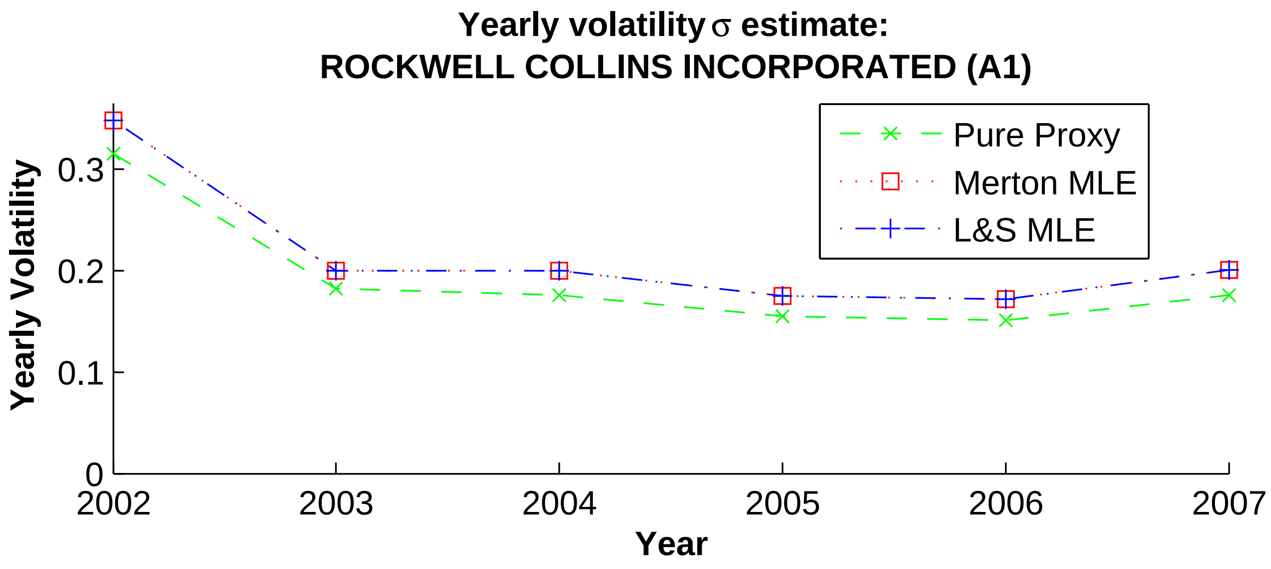

Yearly volatility σ estimate:
COCA−COLA COMPANY (THE) (Aa3)
Yearly Volatility
0.1
0.05
0
2006
2007
Year
Pure Proxy
Merton MLE
L&S MLE

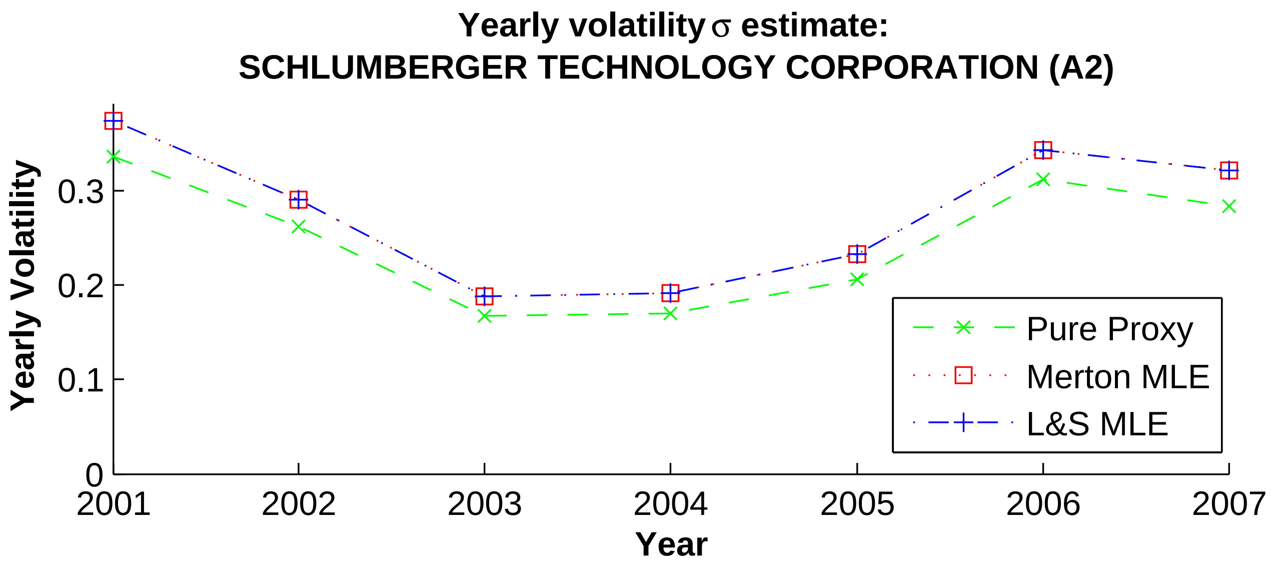

Yearly volatility σ estimate:
COMSTOCK RESOURCES INCORPORATED (B2)
Yearly Volatility
0.25
0.2
0.15
0.1
0.05
0
2003
2004
2005
2006
2007
Year
Pure Proxy
Merton MLE
L&S MLE

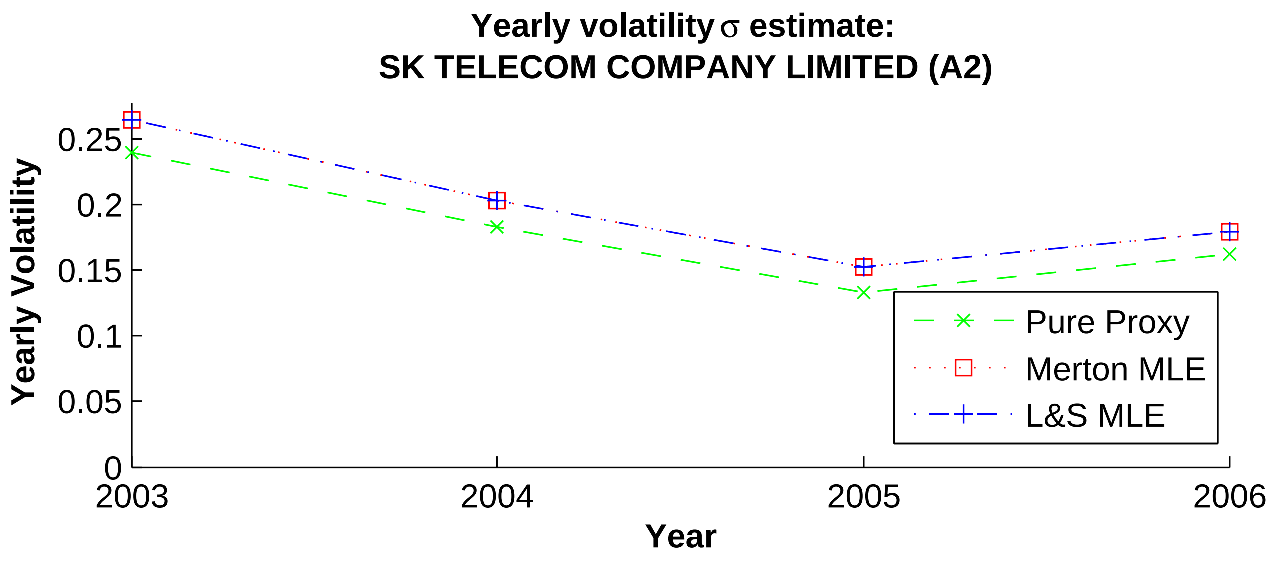

Yearly volatility σ estimate:
DEAN FOODS COMPANY (B3)
Yearly Volatility
0.1
0.05
0
2005
2006
2007
Year
Pure Proxy
Merton MLE
L&S MLE

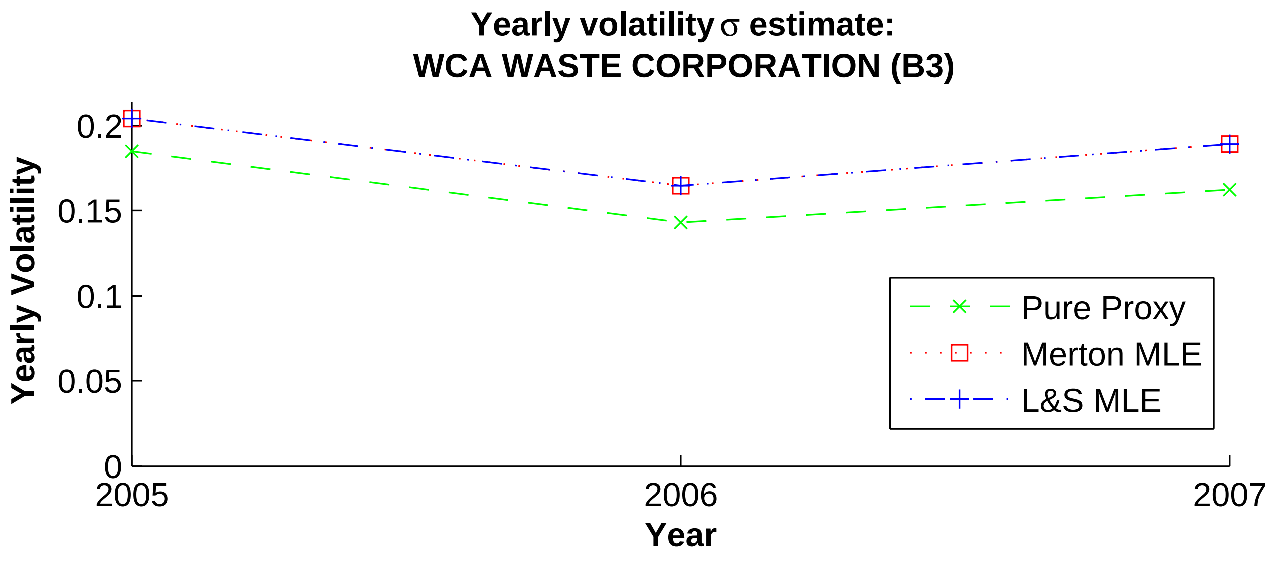

Yearly volatility σ estimate:
MILLICOM INTERNATIONAL CELLULAR SA (B1)
Yearly Volatility
0
0.1
0.2
0.3
0.4
2002
2003
2004
2005
2006
2007
Year
Pure Proxy
Merton MLE
L&S MLE

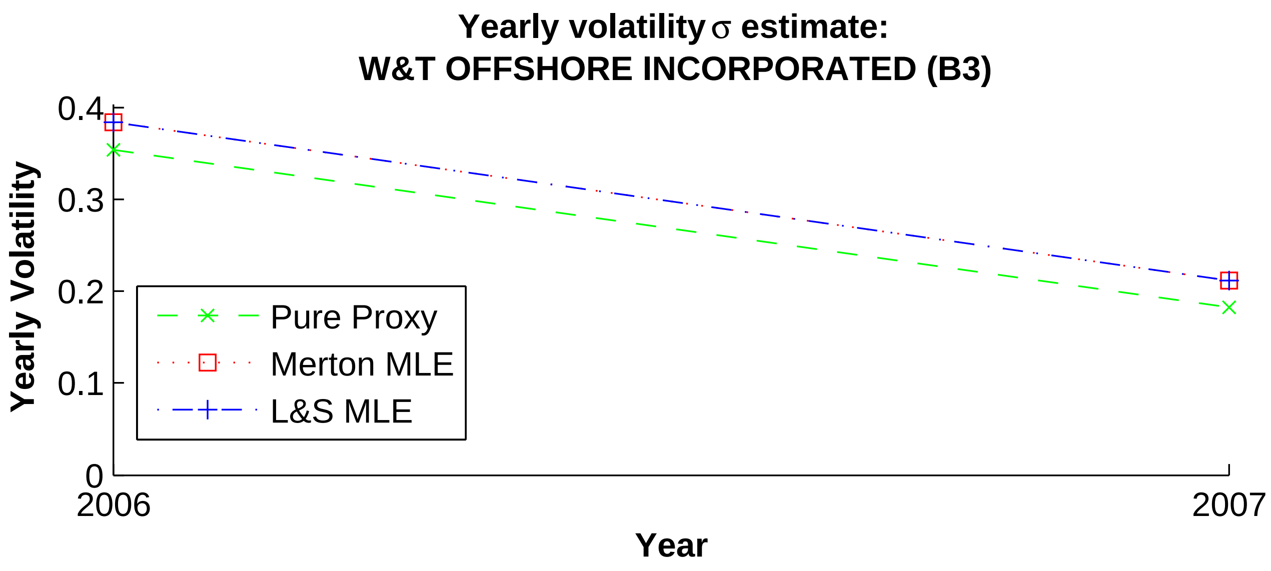

Yearly volatility σ estimate:
PUBLIC SERVICE ENTERPRISE GROUP INCORPORATED (A3)
Yearly Volatility
0
0.02
0.04
0.06
0.08
0.1
2003
2004
2005
2006
2007
Year
Pure Proxy
Merton MLE
L&S MLE


Yearly volatility σ estimate:
ROCKWELL COLLINS INCORPORATED (A1)
Yearly Volatility
0
0.1
0.2
0.3
2002
2003
2004
2005
2006
2007
Year
Pure Proxy
Merton MLE
L&S MLE


Yearly volatility σ estimate:
SCHLUMBERGER TECHNOLOGY CORPORATION (A2)
Yearly Volatility
0
0.1
0.2
0.3
2001
2002
2003
2004
2005
2006
2007
Year
Pure Proxy
Merton MLE
L&S MLE


Yearly volatility σ estimate:
SK TELECOM COMPANY LIMITED (A2)
Yearly Volatility
0
0.05
0.1
0.15
0.2
0.25
2003
2004
2005
2006
Year
Pure Proxy
Merton MLE
L&S MLE

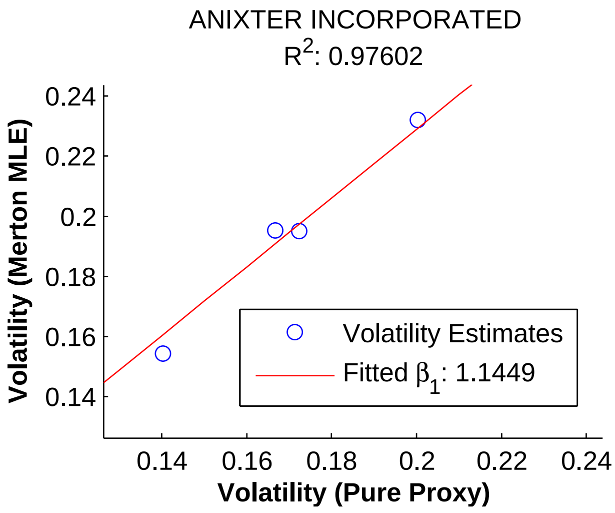

Yearly volatility σ estimate:
WCA WASTE CORPORATION (B3)
Yearly Volatility
0
0.05
0.1
0.15
0.2
2005
2006
2007
Year
Pure Proxy
Merton MLE
L&S MLE

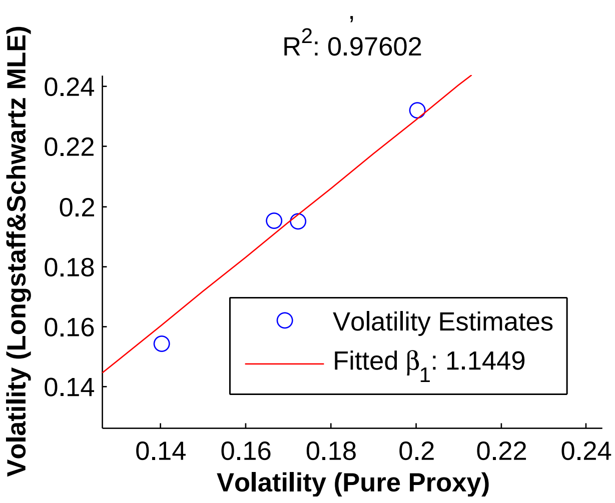

Yearly volatility σ estimate:
W&T OFFSHORE INCORPORATED (B3)
Yearly Volatility
0.4
0.3
0.2
0.1
0
2006
2007
Year
Pure Proxy
Merton MLE
L&S MLE

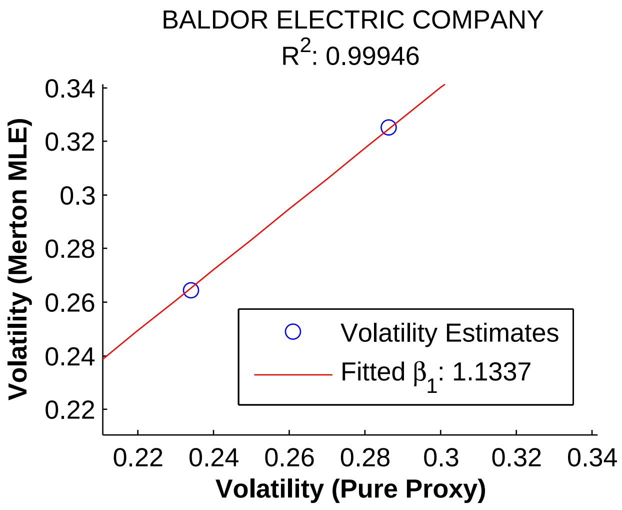
Appendix C

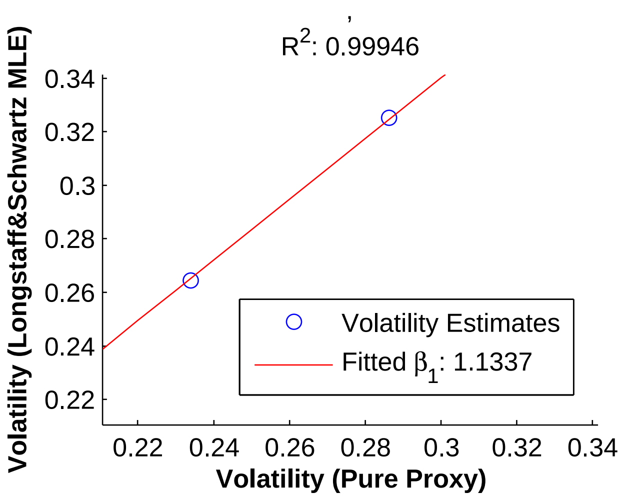
Plots of correlation between asset volatility estimates

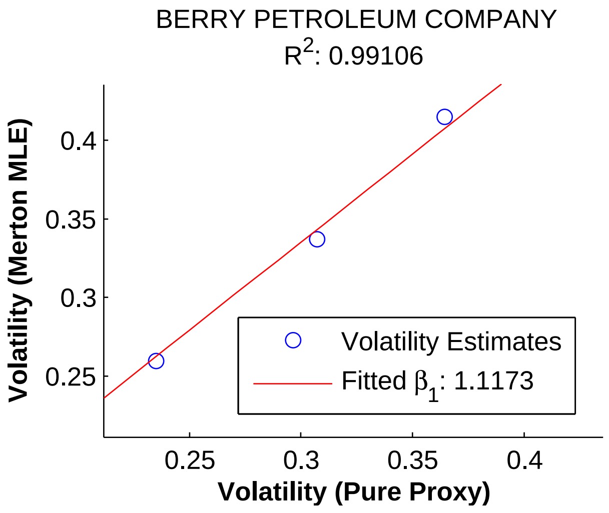
Plots showing the R^2 level of correlation between a simplistic pure proxy approach of asset volatility estimation, and the theoretically appealing and complex maximum likelihood estimation approach, broken down by firm.

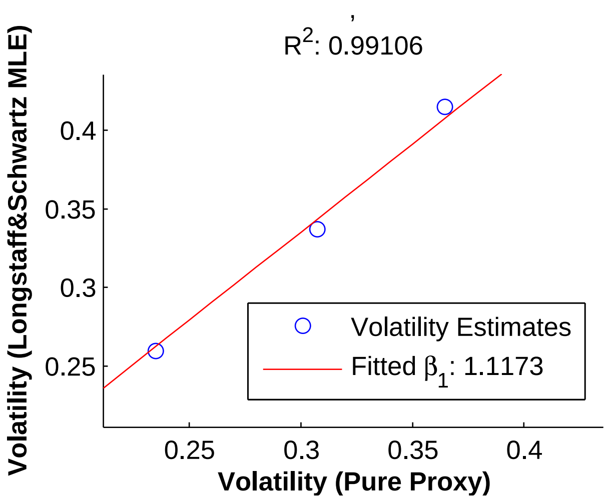


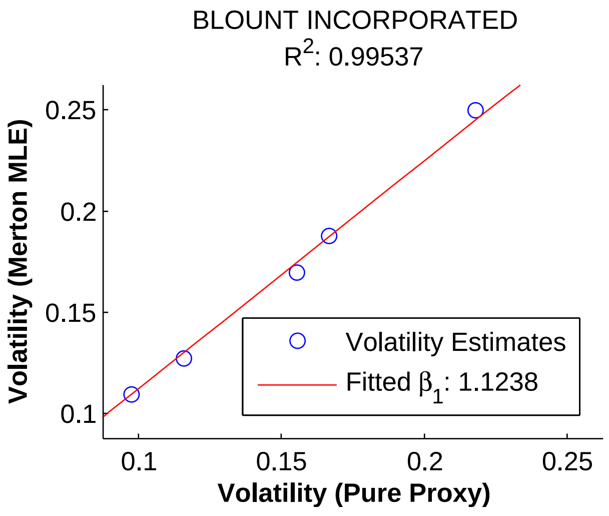


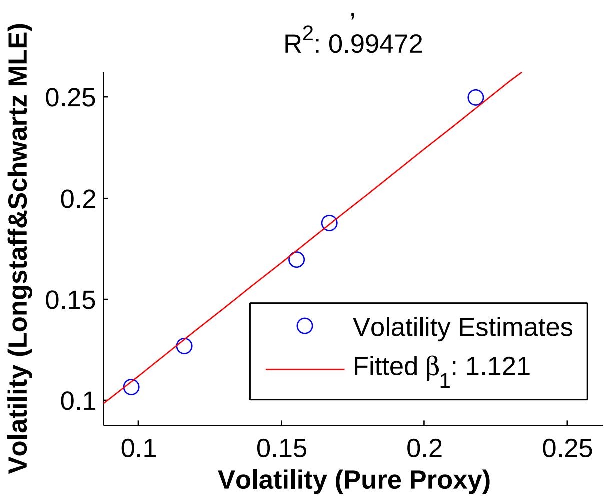

BALDOR ELECTRIC COMPANY
R^2: 0.99946
Volatility (Merton MLE)
Volatility (Pure Proxy)
Volatility Estimates
Fitted β_1: 1.1337

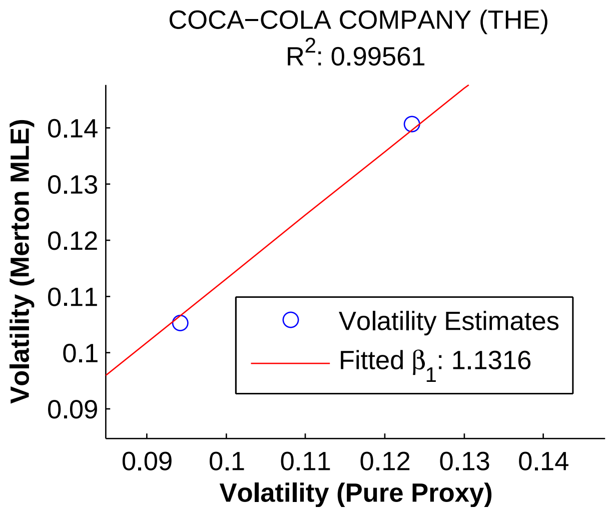

,
R^2: 0.99946
Volatility (Longstaff&Schwartz MLE)
Volatility (Pure Proxy)
Volatility Estimates
Fitted β_1: 1.1337

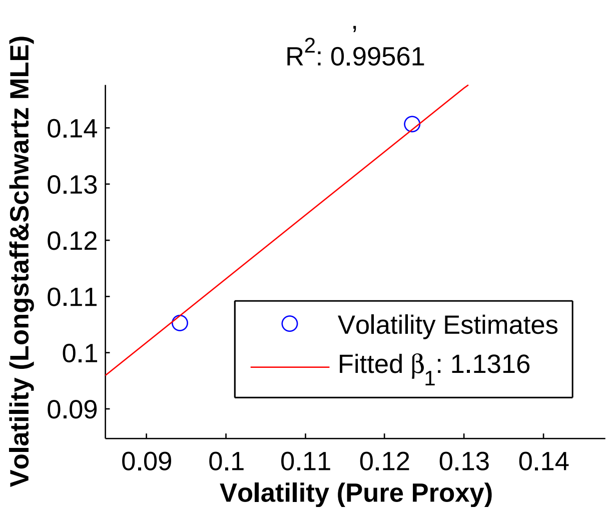

BERRY PETROLEUM COMPANY
R^2: 0.99106
Volatility (Merton MLE)
Volatility (Pure Proxy)
Volatility Estimates
Fitted β_1: 1.1173

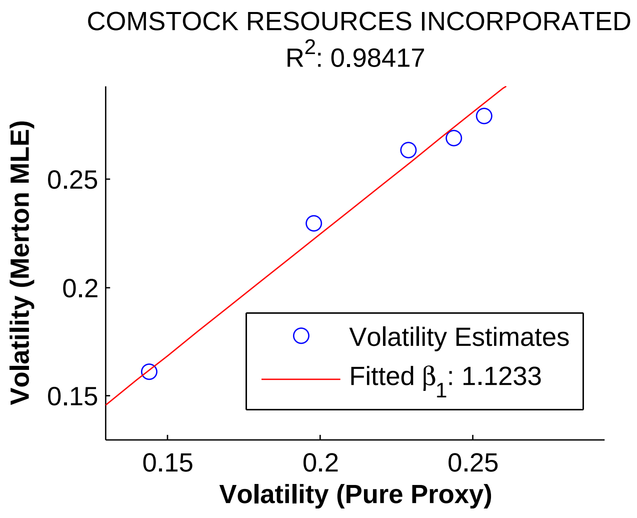

,
R^2: 0.99106
Volatility (Longstaff&Schwartz MLE)
Volatility (Pure Proxy)
Volatility Estimates
Fitted β_1: 1.1173

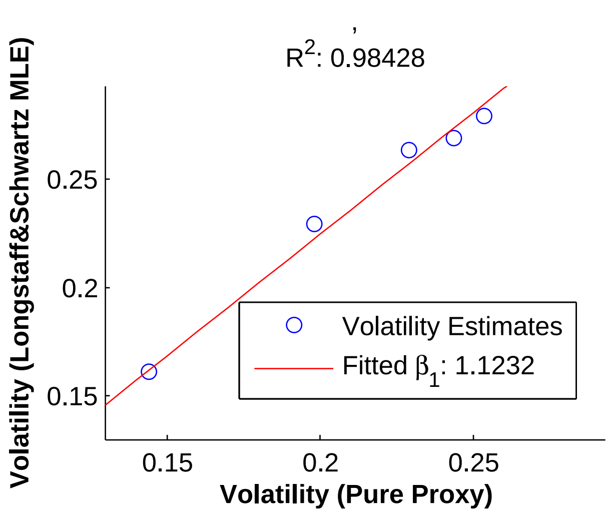

BLOUNT INCORPORATED
R^2: 0.99537
Volatility (Merton MLE)
Volatility (Pure Proxy)
Volatility Estimates
Fitted β_1: 1.1238

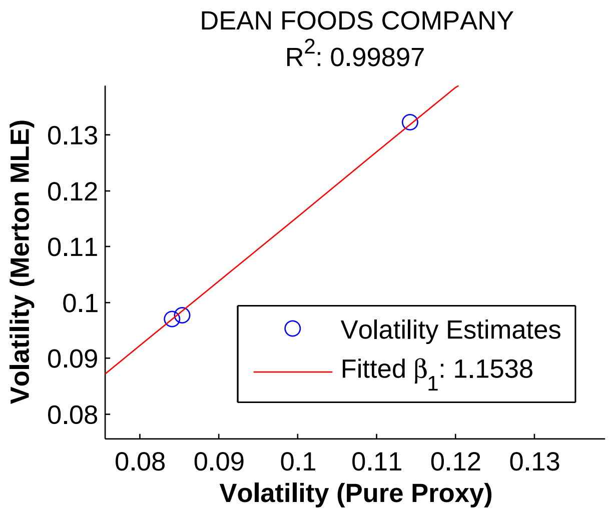

,
R^2: 0.99472
Volatility (Longstaff&Schwartz MLE)
Volatility (Pure Proxy)
Volatility Estimates
Fitted β_1: 1.121

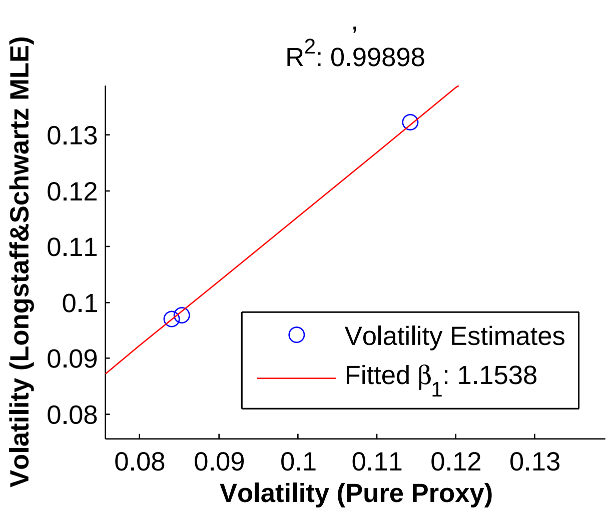

COCA−COLA COMPANY (THE)
R^2: 0.99561
Volatility (Merton MLE)
0.14
0.13
0.12
0.11
0.1
0.09
0.09 0.1 0.11 0.12 0.13 0.14
Volatility (Pure Proxy)
Volatility Estimates
Fitted β_1: 1.1316

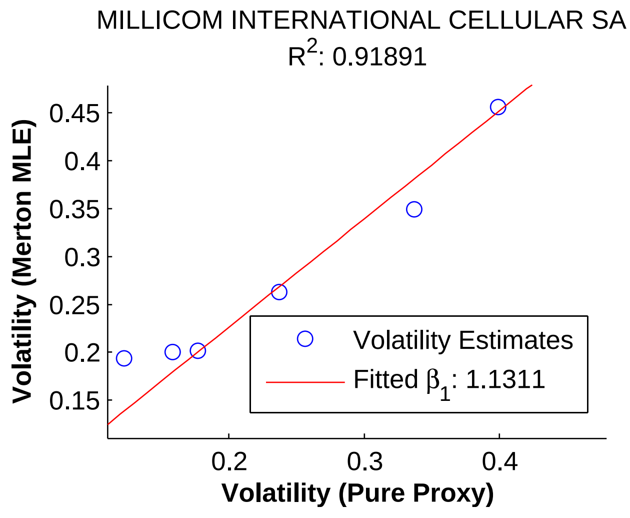

,
R^2: 0.99561
Volatility (Longstaff&Schwartz MLE)
0.14
0.13
0.12
0.11
0.1
0.09
0.09 0.1 0.11 0.12 0.13 0.14
Volatility (Pure Proxy)
Volatility Estimates
Fitted β_1: 1.1316

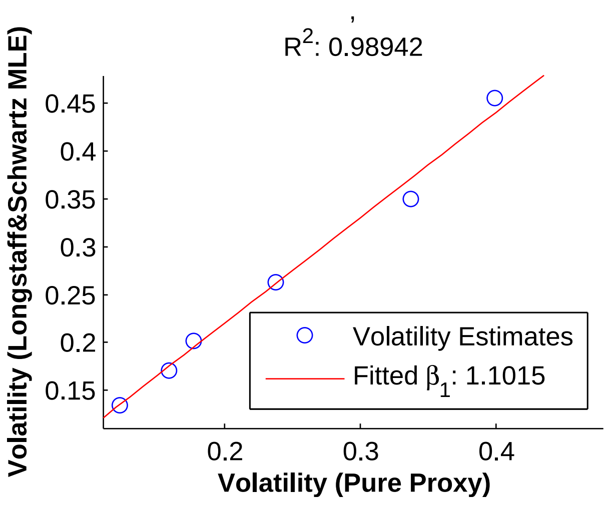

COMSTOCK RESOURCES INCORPORATED
R^2: 0.98417
Volatility (Merton MLE)
0.25
0.2
0.15
0.15 0.2 0.25
Volatility (Pure Proxy)
Volatility Estimates
Fitted β_1: 1.1233

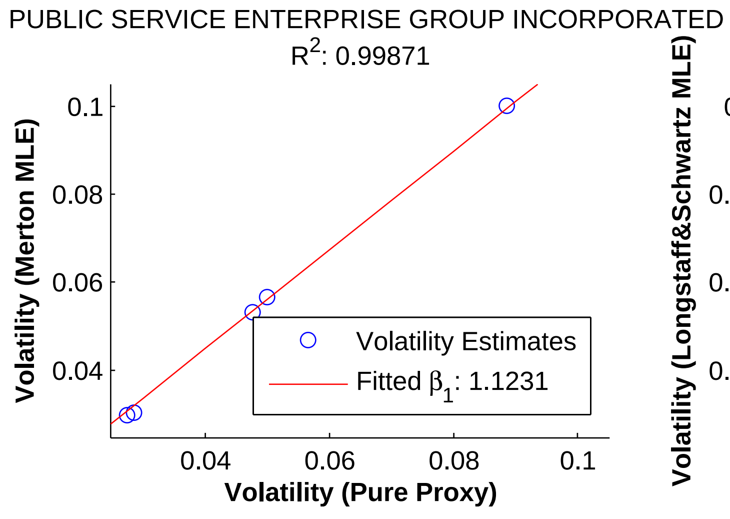

,
R^2: 0.98428
Volatility (Longstaff&Schwartz MLE)
0.25
0.2
0.15
0.15 0.2 0.25
Volatility (Pure Proxy)
Volatility Estimates
Fitted β_1: 1.1232

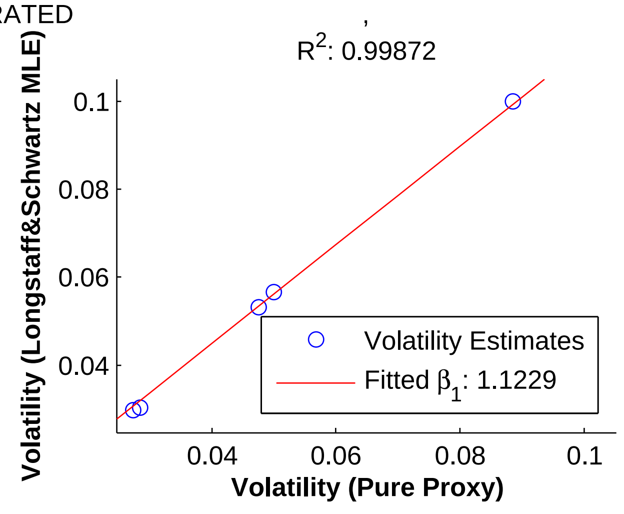

DEAN FOODS COMPANY
R^2: 0.99897
Volatility (Merton MLE)
0.13
0.12
0.11
0.1
0.09
0.08
0.08 0.09 0.1 0.11 0.12 0.13
Volatility (Pure Proxy)
Volatility Estimates
Fitted β_1: 1.1538

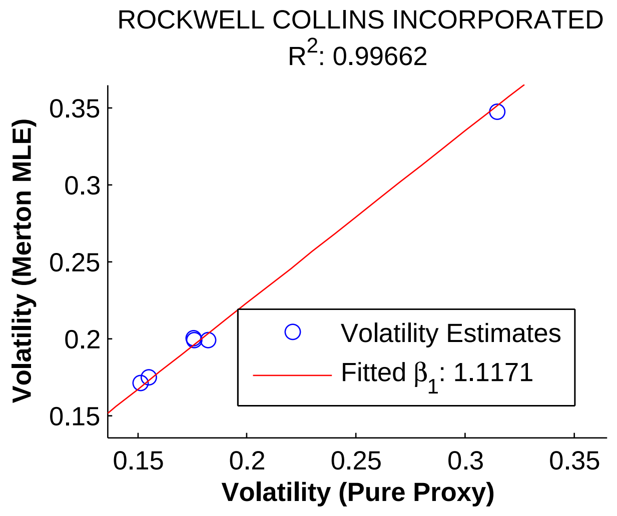

,
R^2: 0.99898
Volatility (Longstaff&Schwartz MLE)
0.13
0.12
0.11
0.1
0.09
0.08
0.08 0.09 0.1 0.11 0.12 0.13
Volatility (Pure Proxy)
Volatility Estimates
Fitted β_1: 1.1538

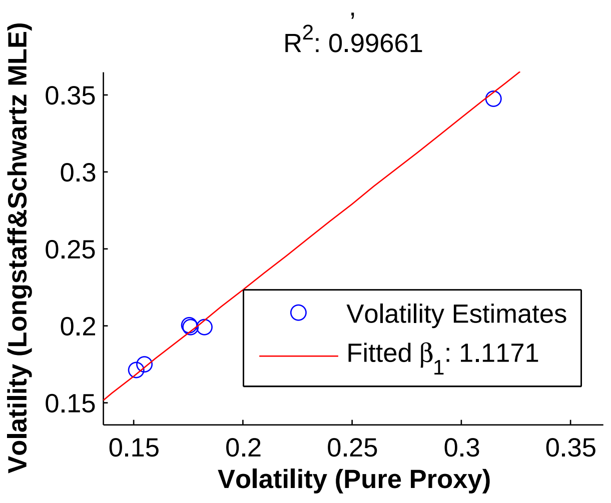

MILLICOM INTERNATIONAL CELLULAR SA
R^2: 0.91891
Volatility (Merton MLE)
Volatility (Pure Proxy)
Volatility Estimates
Fitted β_1: 1.1311

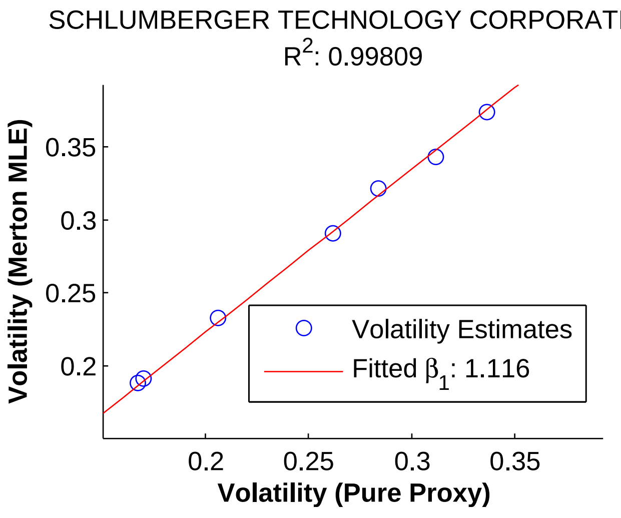

,
R^2: 0.98942
Volatility (Longstaff&Schwartz MLE)
Volatility (Pure Proxy)
Volatility Estimates
Fitted β_1: 1.1015

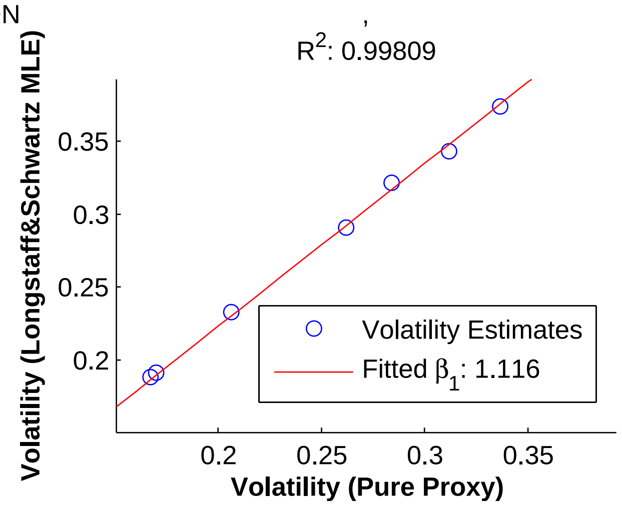

PUBLIC SERVICE ENTERPRISE GROUP INCORPORATED
R^2: 0.99871
Volatility (Merton MLE)
Volatility (Pure Proxy)
Volatility Estimates
Fitted β_1: 1.1231

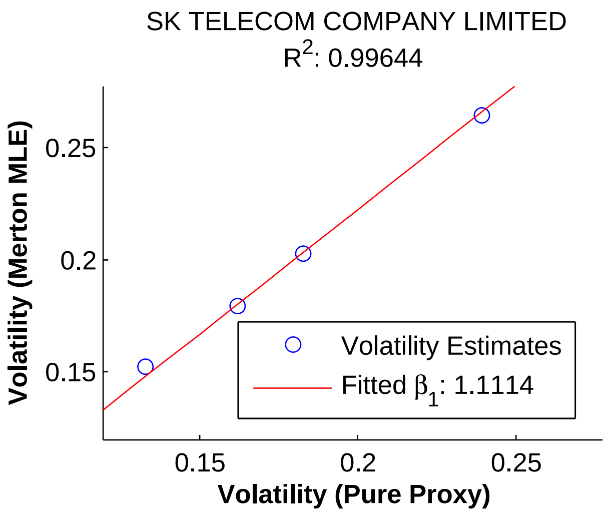

,
R^2: 0.99872
Volatility (Longstaff&Schwartz MLE)
Volatility (Pure Proxy)
Volatility Estimates
Fitted β_1: 1.1229

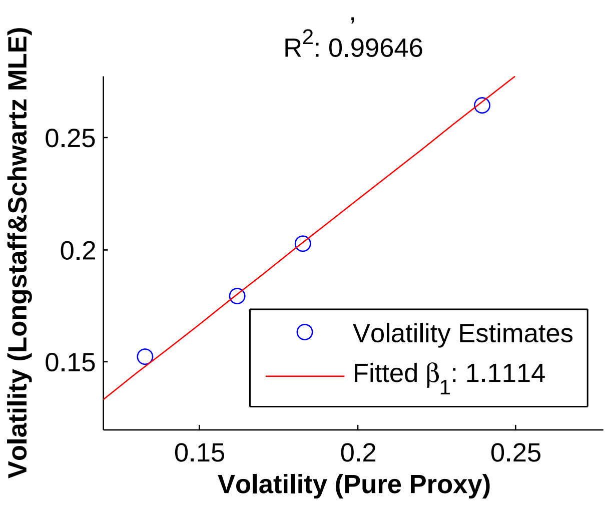

ROCKWELL COLLINS INCORPORATED
R^2: 0.99662
Volatility (Merton MLE)
Volatility (Pure Proxy)
Volatility Estimates
Fitted β_1: 1.1171

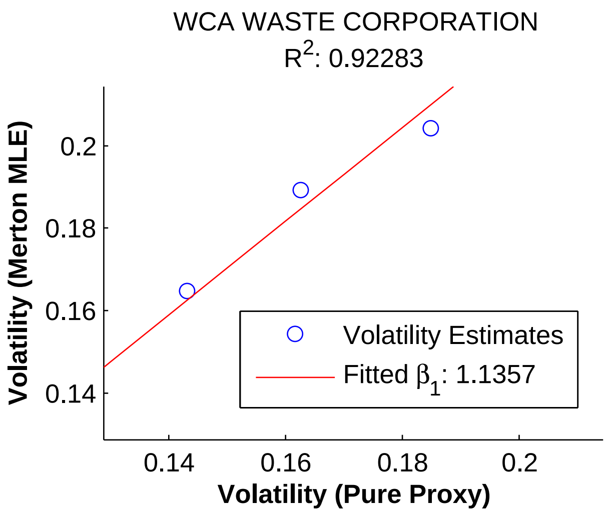

,
R^2: 0.99661
Volatility (Longstaff&Schwartz MLE)
Volatility (Pure Proxy)
Volatility Estimates
Fitted β_1: 1.1171

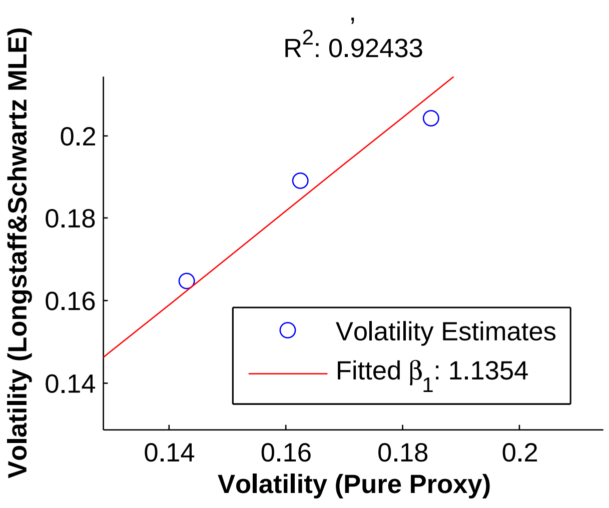

SCHLUMBERGER TECHNOLOGY CORPORATION
R^2: 0.99809
Volatility (Merton MLE)
Volatility (Pure Proxy)
Volatility Estimates
Fitted β_1: 1.116
0.35
0.3
0.25
0.2
0.2
0.25
0.3
0.35

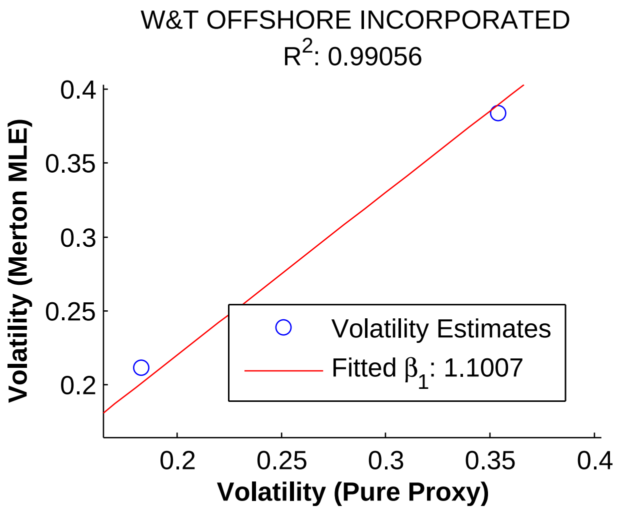

,
R^2: 0.99809
Volatility (Longstaff&Schwartz MLE)
Volatility (Pure Proxy)
Volatility Estimates
Fitted β_1: 1.116
0.35
0.3
0.25
0.2
0.2
0.25
0.3
0.35

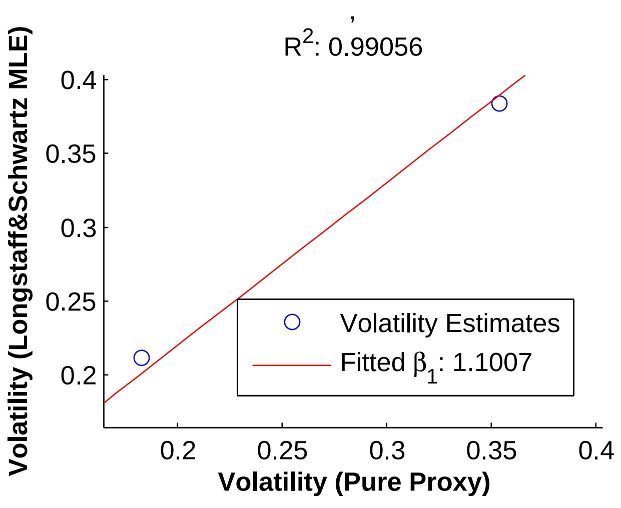

SK TELECOM COMPANY LIMITED
R^2: 0.99644
Volatility (Merton MLE)
Volatility (Pure Proxy)
Volatility Estimates
Fitted β_1: 1.1114
0.25
0.2
0.15
0.15
0.2
0.25


,
R^2: 0.99646
Volatility (Longstaff&Schwartz MLE)
Volatility (Pure Proxy)
Volatility Estimates
Fitted β_1: 1.1114
0.25
0.2
0.15
0.15
0.2
0.25


WCA WASTE CORPORATION
R^2: 0.92283
Volatility (Merton MLE)
Volatility (Pure Proxy)
Volatility Estimates
Fitted β_1: 1.1357
0.2
0.18
0.16
0.14
0.14
0.16
0.18
0.2


,
R^2: 0.92433
Volatility (Longstaff&Schwartz MLE)
Volatility (Pure Proxy)
Volatility Estimates
Fitted β_1: 1.1354
0.2
0.18
0.16
0.14
0.14
0.16
0.18
0.2

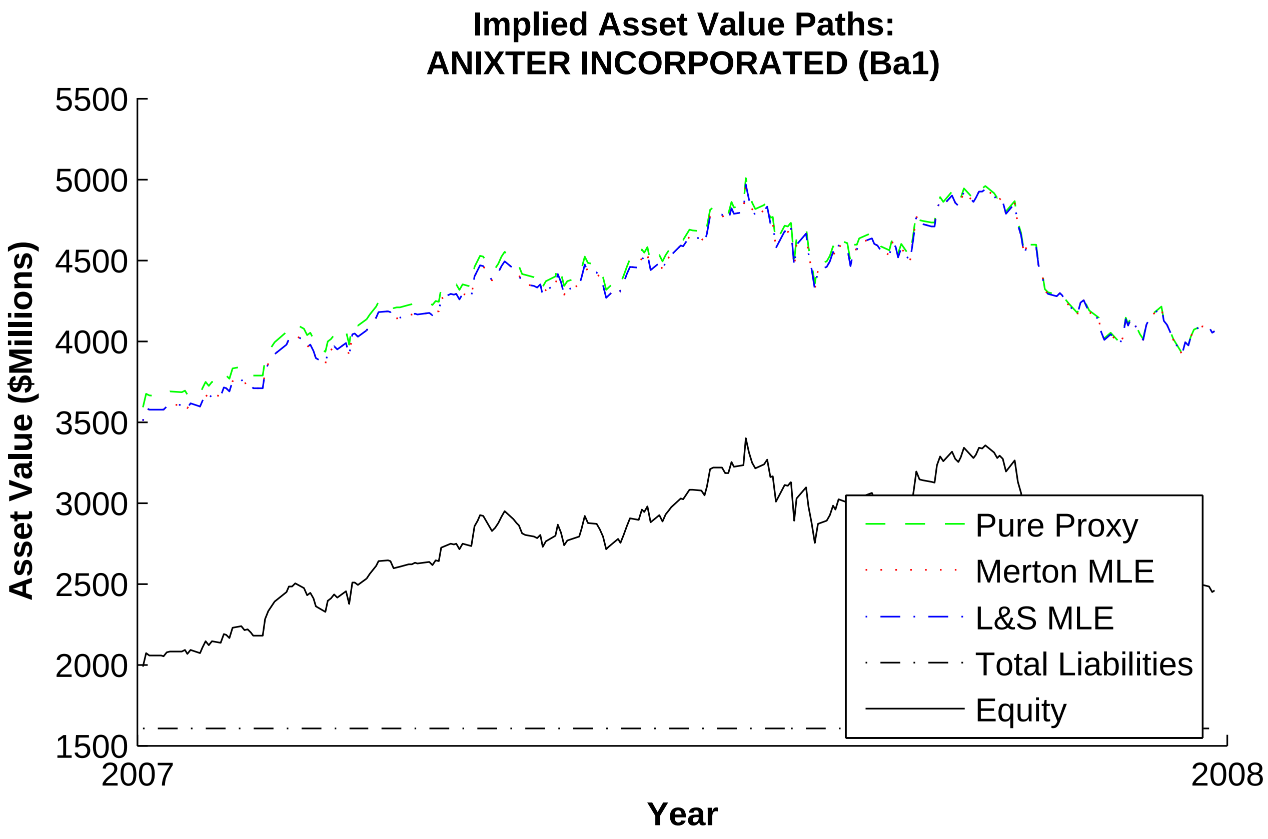

W&T OFFSHORE INCORPORATED
R²: 0.99056
Volatility (Merton MLE)
Volatility (Pure Proxy)
Volatility Estimates
Fitted β1: 1.1007
0.4
0.35
0.3
0.25
0.2
0.2
0.25
0.3
0.35
0.4

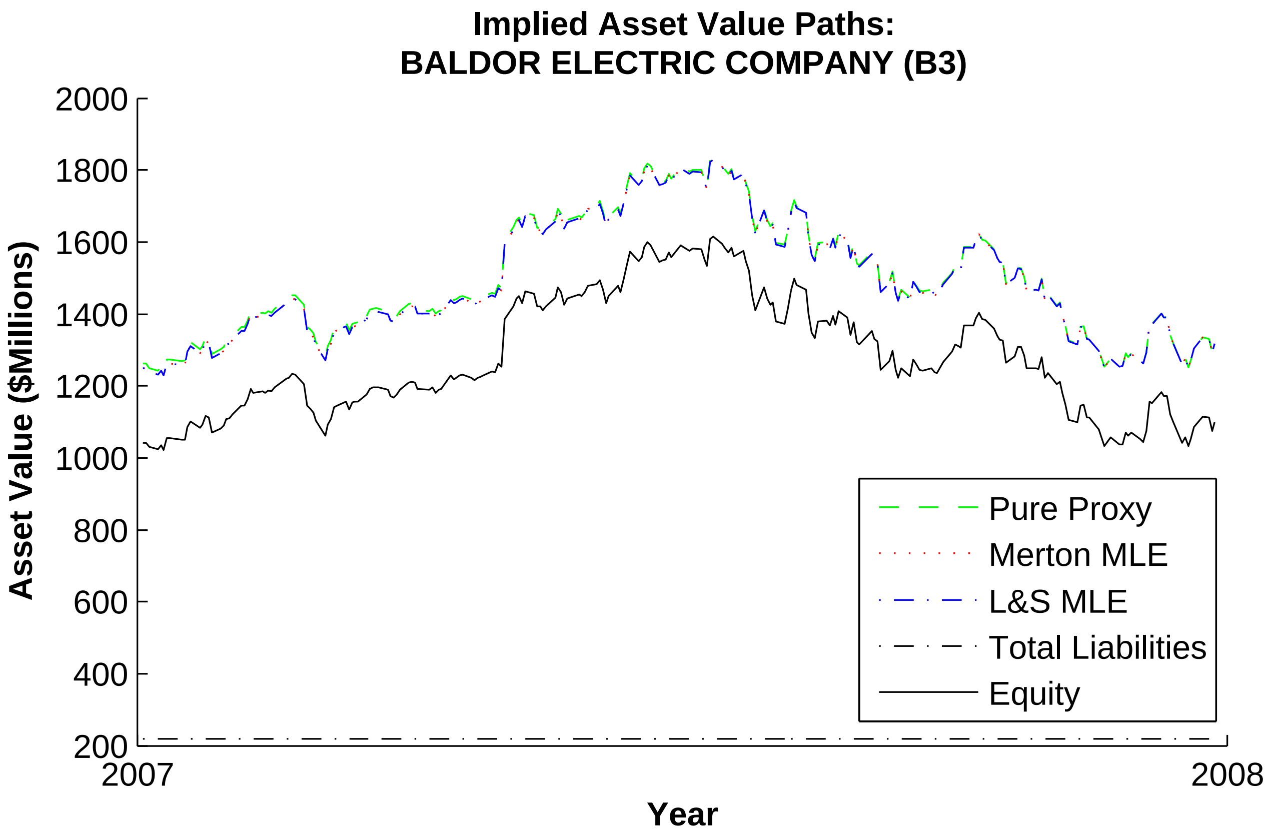

,
R²: 0.99056
Volatility (Longstaff&Schwartz MLE)
Volatility (Pure Proxy)
Volatility Estimates
Fitted β1: 1.1007
0.4
0.35
0.3
0.25
0.2
0.2
0.25
0.3
0.35
0.4

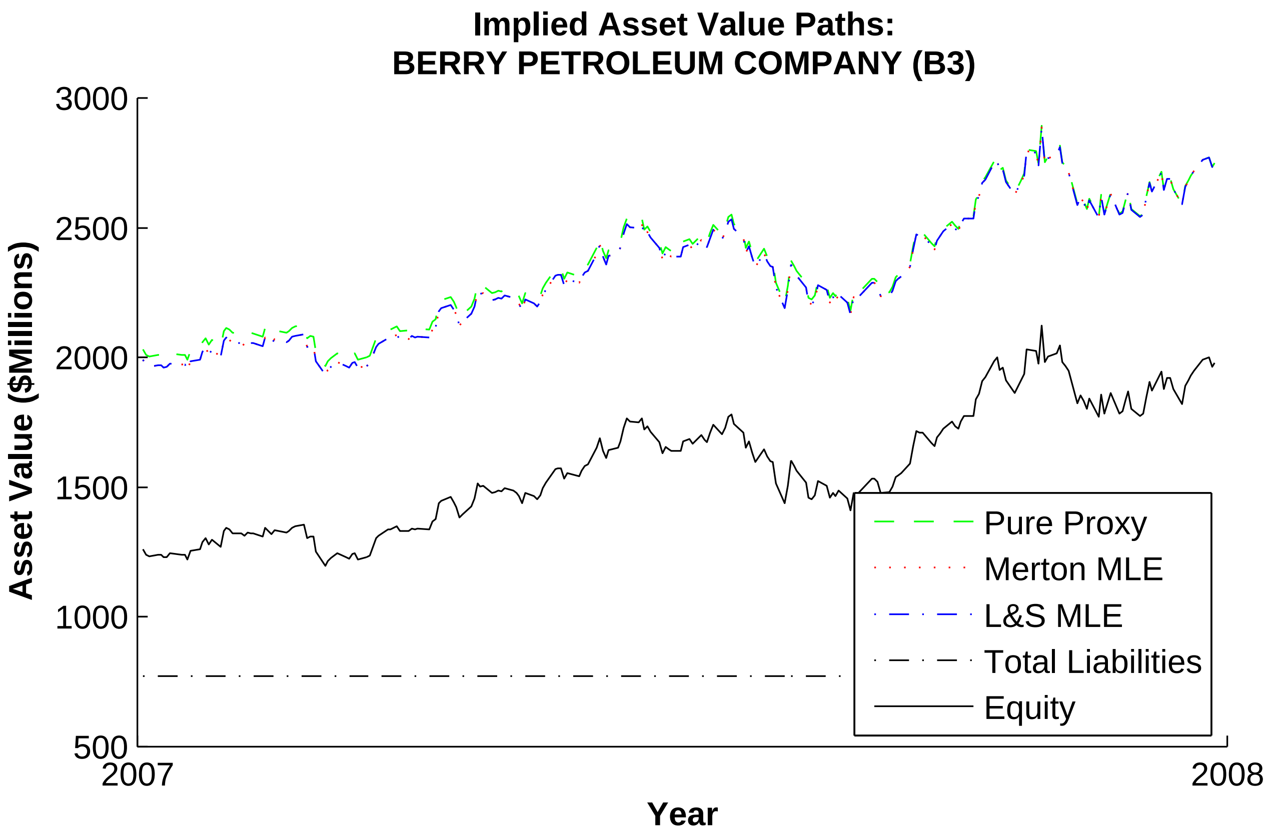
Appendix D

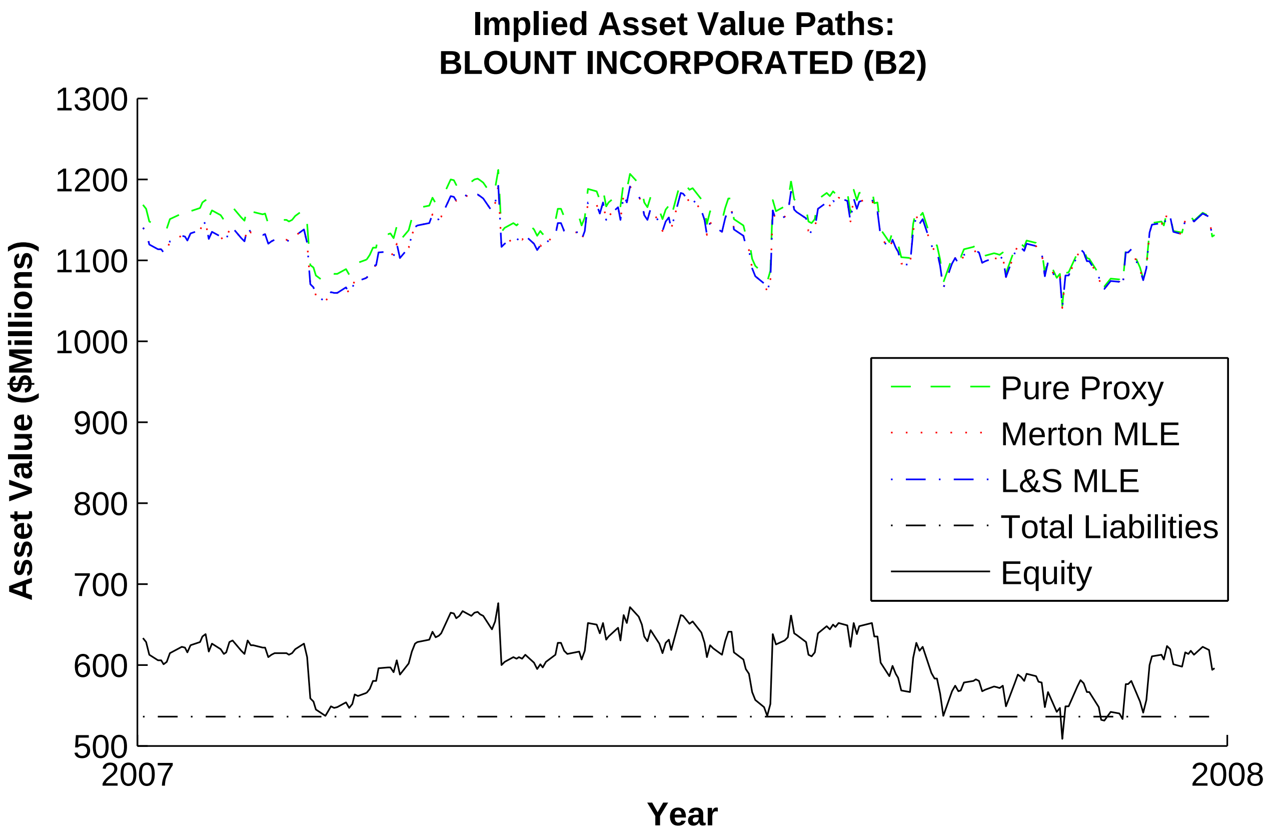
Plots of yearly implied asset value estimates

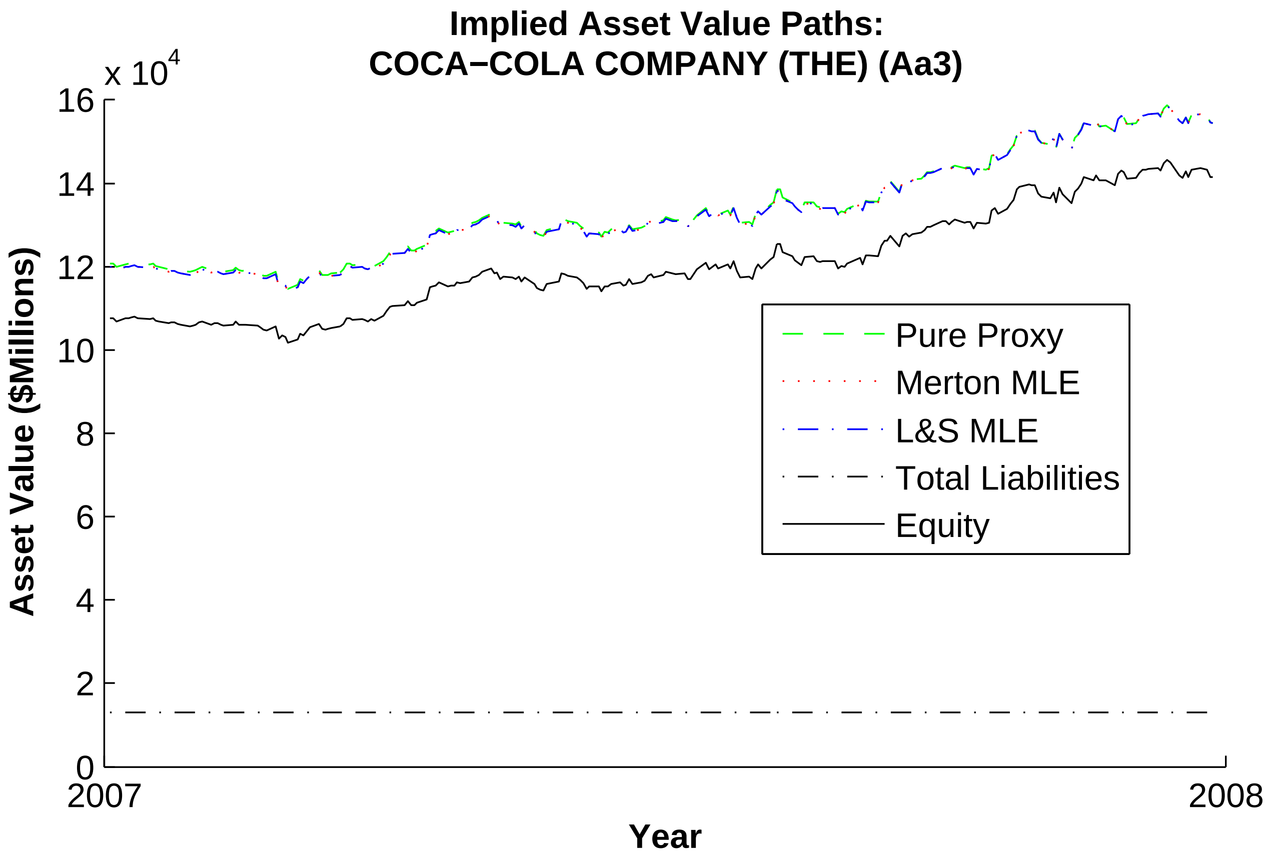
Graphs of the implied asset values for each firm included in the study. Pure proxy asset valuations are the sum of debt and equity from which drift and volatility can be directly calculated. Merton [1974], Longstaff and Schwartz [1995] implied asset valuations are calculated using the maximum likelihood estimation approach to calculate implied asset volatility. The volatility estimates are fed into the relevant inverted option pricing formulas to calculate the implied asset value for each day within a calendar year.

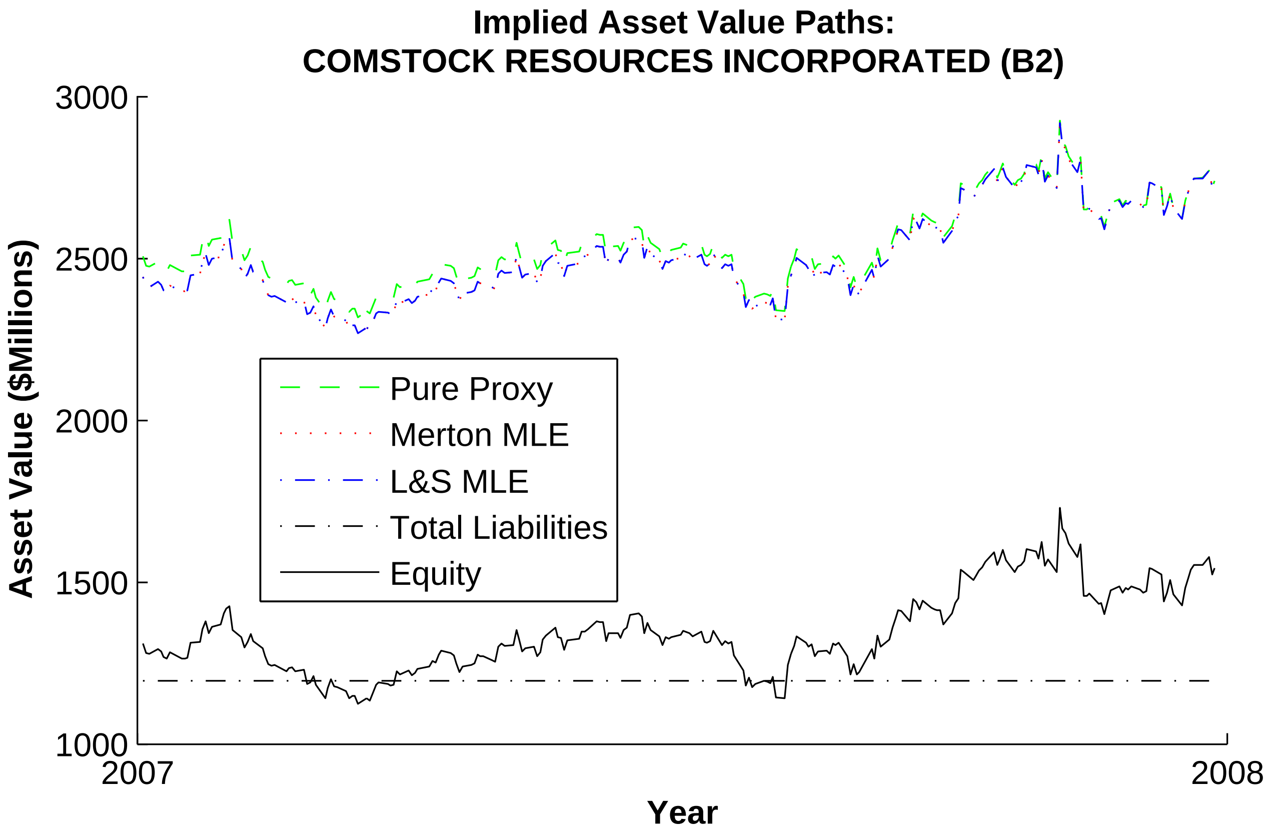

Implied Asset Value Paths:
ANIXTER INCORPORATED (Ba1)
Asset Value ($Millions)
5500
5000
4500
4000
3500
3000
2500
2000
1500
2007
2008
Year
Pure Proxy
Merton MLE
L&S MLE
Total Liabilities
Equity

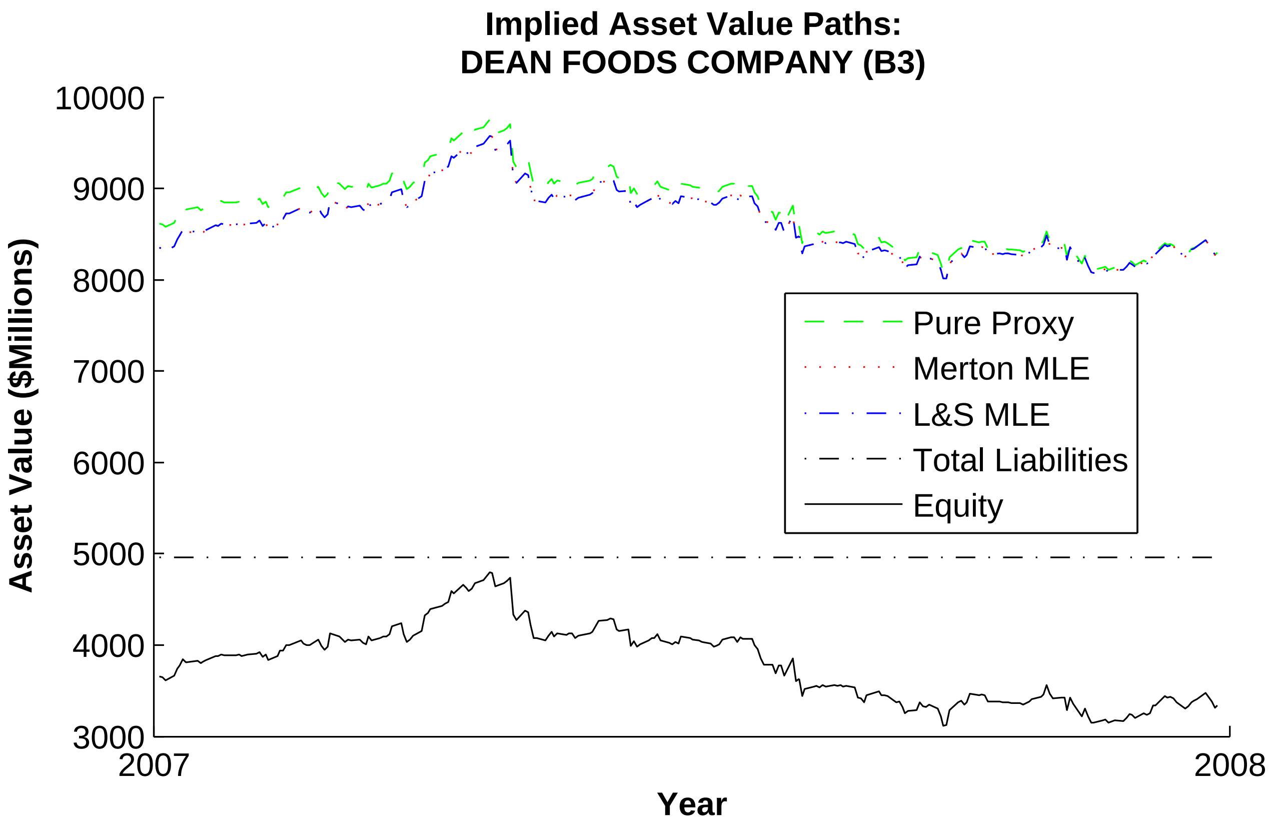

Implied Asset Value Paths:
BALDOR ELECTRIC COMPANY (B3)
Asset Value ($Millions)
2000
1800
1600
1400
1200
1000
800
600
400
200
2007
2008
Year
Pure Proxy
Merton MLE
L&S MLE
Total Liabilities
Equity

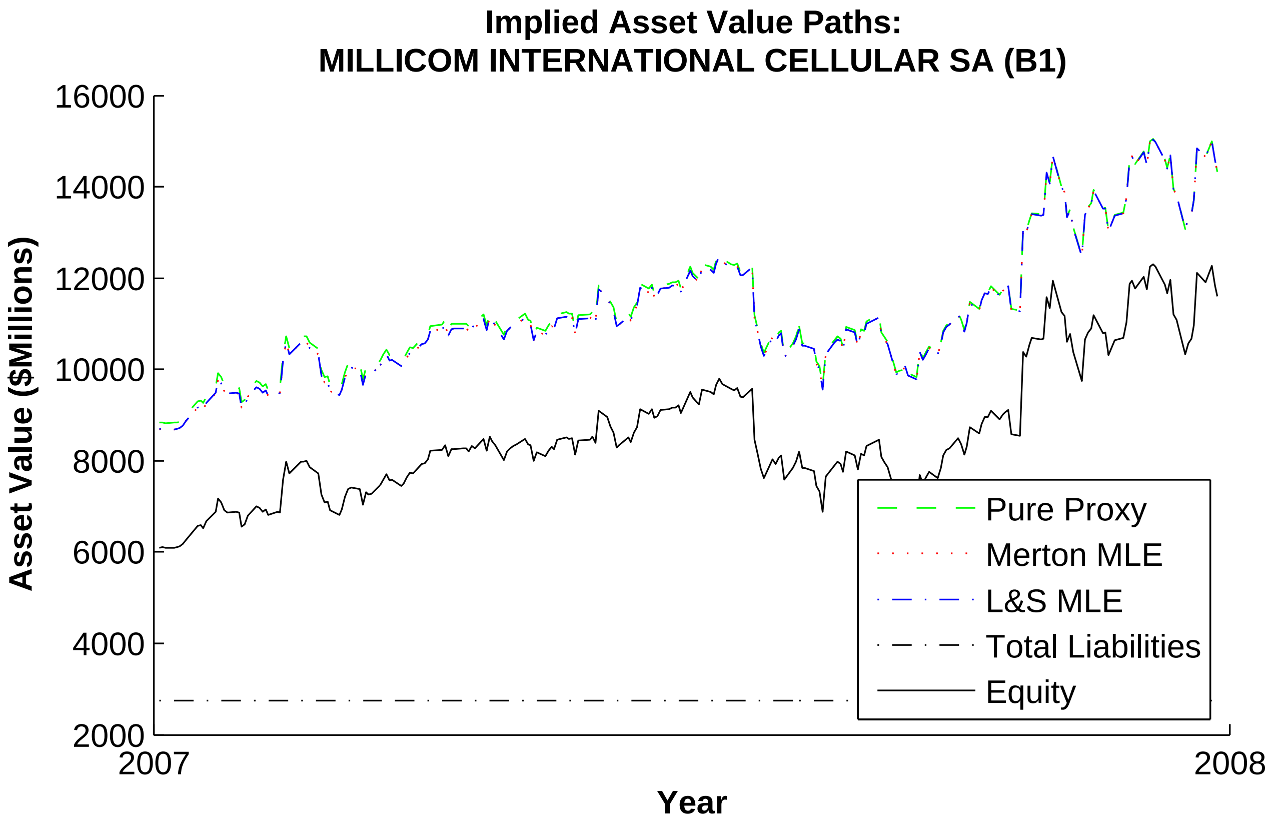

Implied Asset Value Paths:
BERRY PETROLEUM COMPANY (B3)
Asset Value ($Millions)
3000
2500
2000
1500
1000
500
2007
2008
Year
Pure Proxy
Merton MLE
L&S MLE
Total Liabilities
Equity

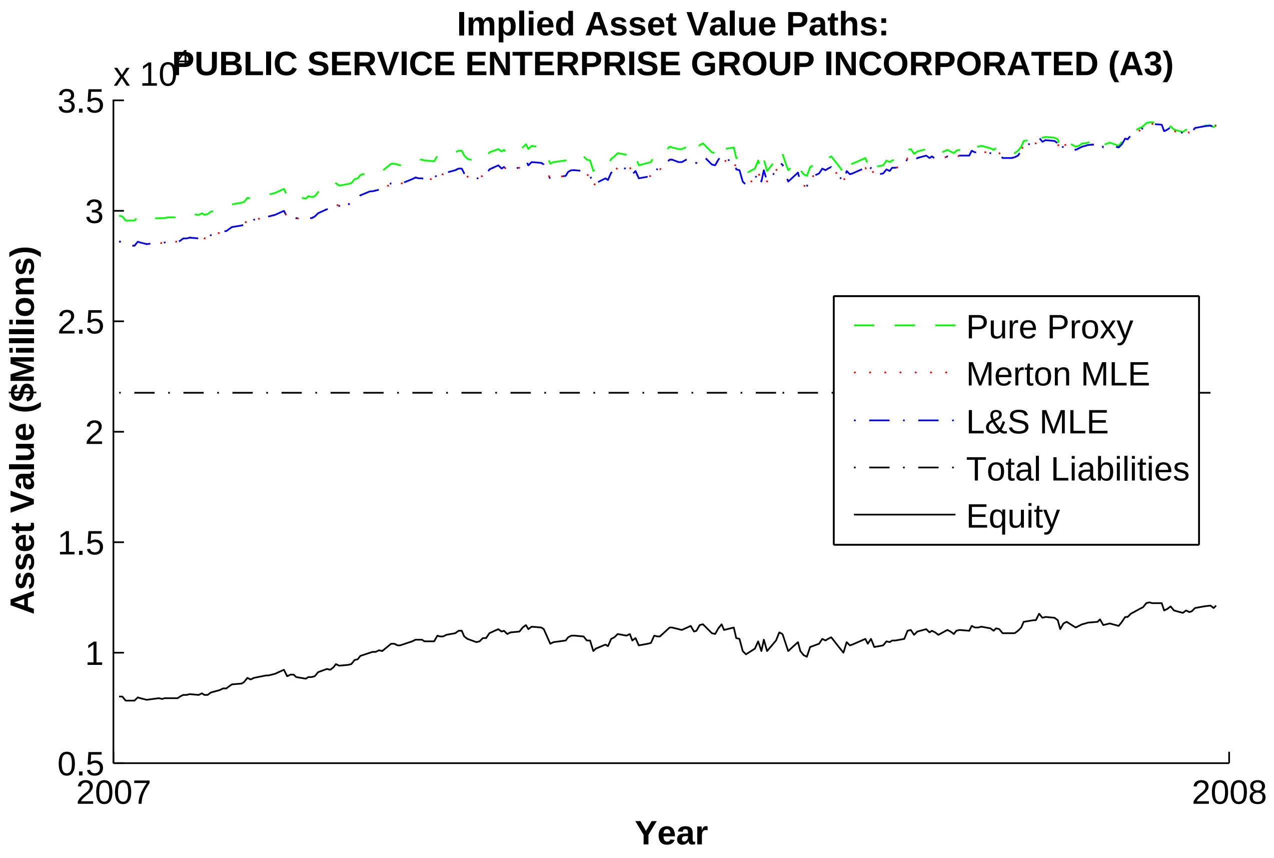

Implied Asset Value Paths:
BLOUNT INCORPORATED (B2)
Asset Value ($Millions)
1300
1200
1100
1000
900
800
700
600
500
2007
2008
Year
Pure Proxy
Merton MLE
L&S MLE
Total Liabilities
Equity

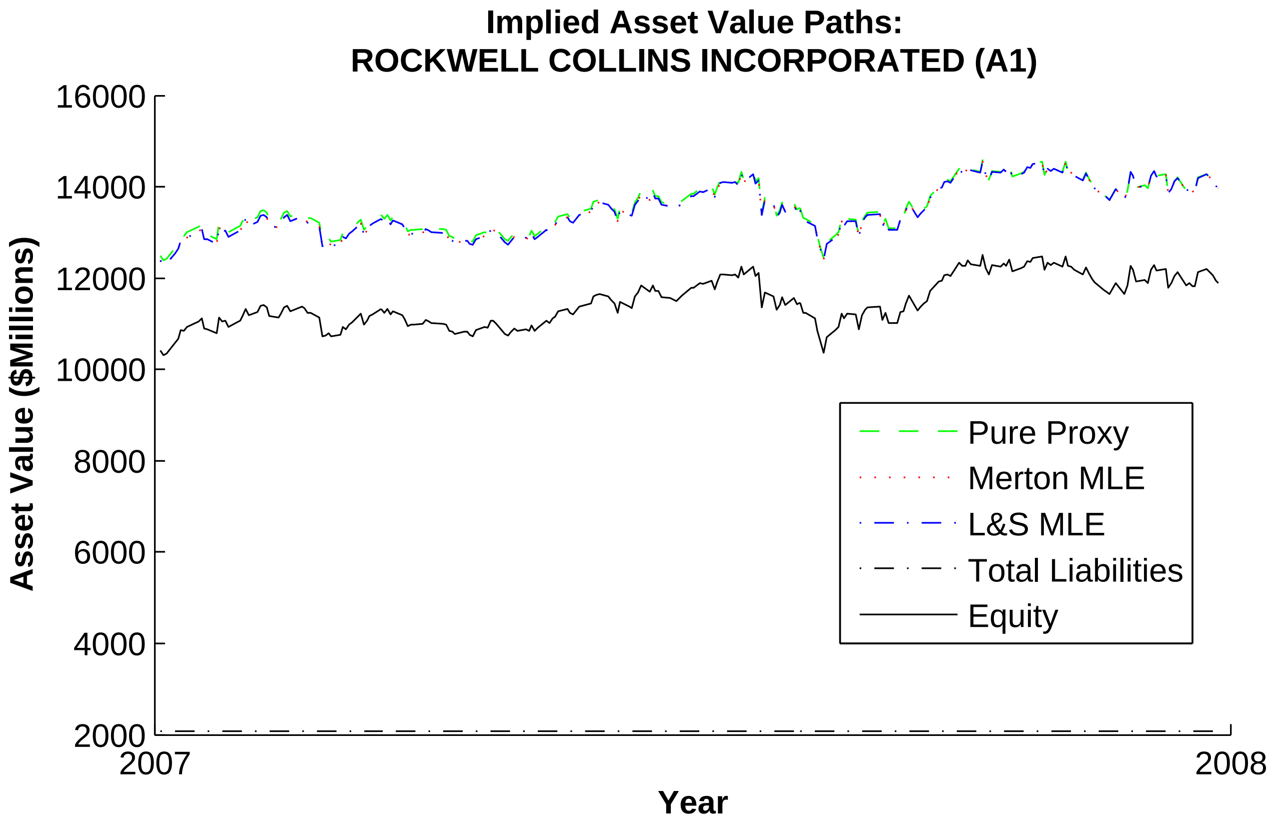

Implied Asset Value Paths:
COCA−COLA COMPANY (THE) (Aa3)
x 10^4
16
14
12
10
8
6
4
2
0
Asset Value ($Millions)
2007
2008
Year
Pure Proxy
Merton MLE
L&S MLE
Total Liabilities
Equity

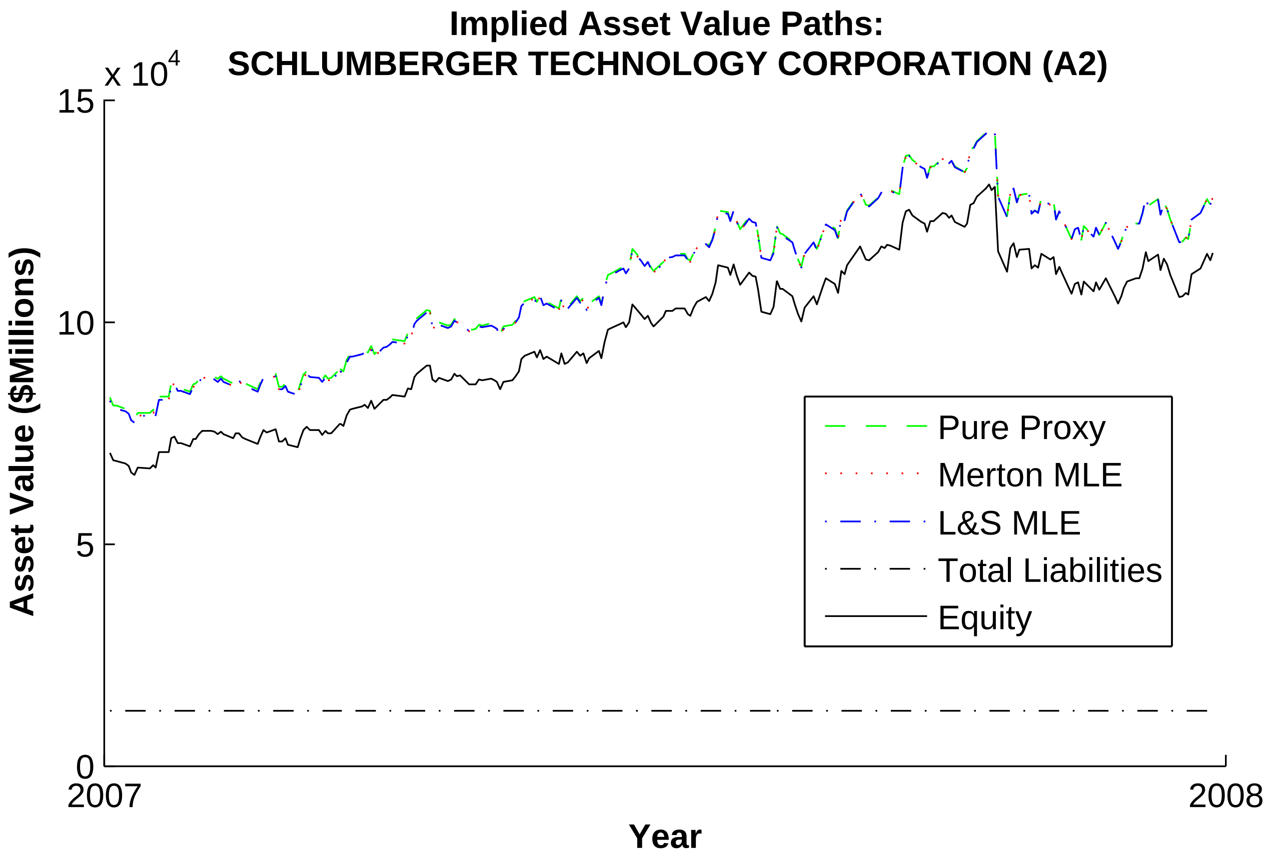

Implied Asset Value Paths:
COMSTOCK RESOURCES INCORPORATED (B2)
3000
2500
2000
1500
1000
Asset Value ($Millions)
2007
2008
Year
Pure Proxy
Merton MLE
L&S MLE
Total Liabilities
Equity

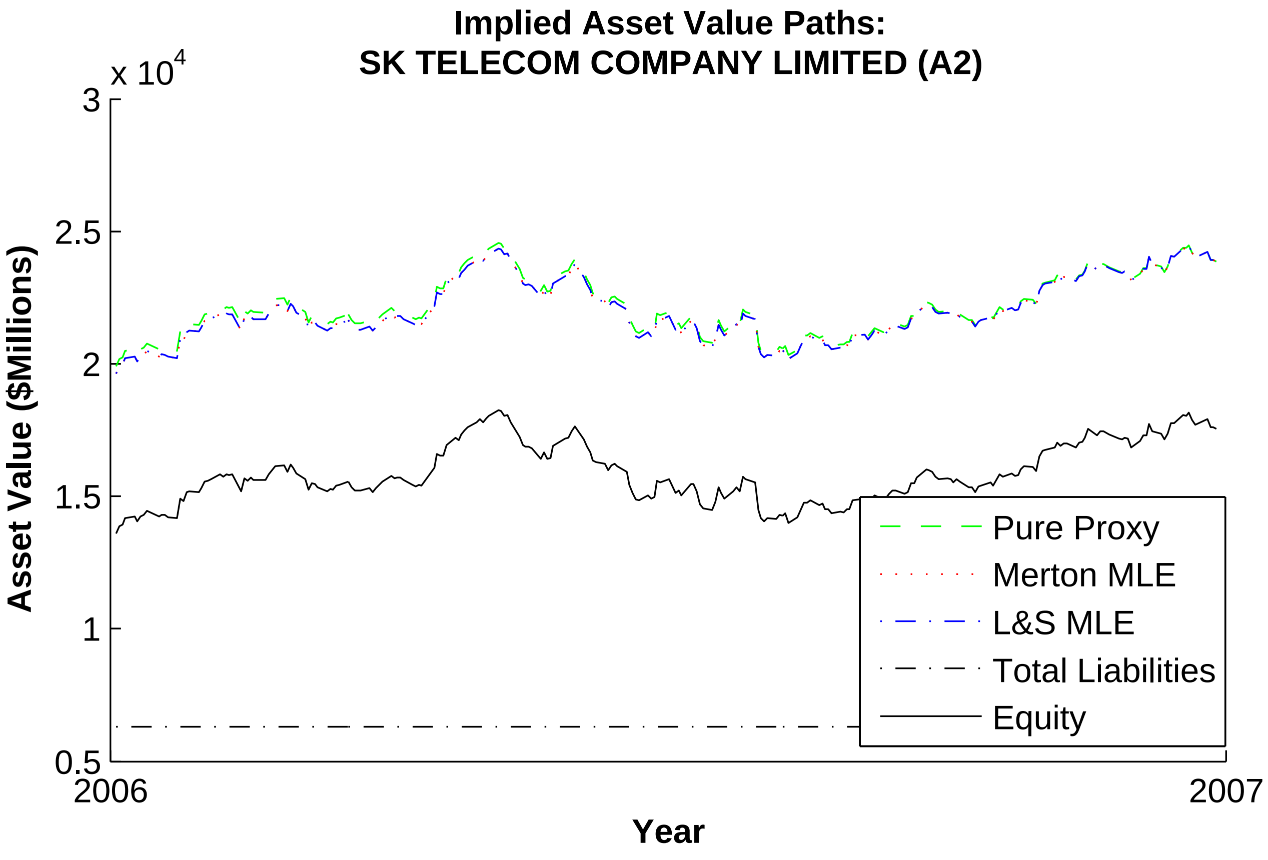

Implied Asset Value Paths:
DEAN FOODS COMPANY (B3)
Asset Value ($Millions)
10000
9000
8000
7000
6000
5000
4000
3000
2007
2008
Year
Pure Proxy
Merton MLE
L&S MLE
Total Liabilities
Equity

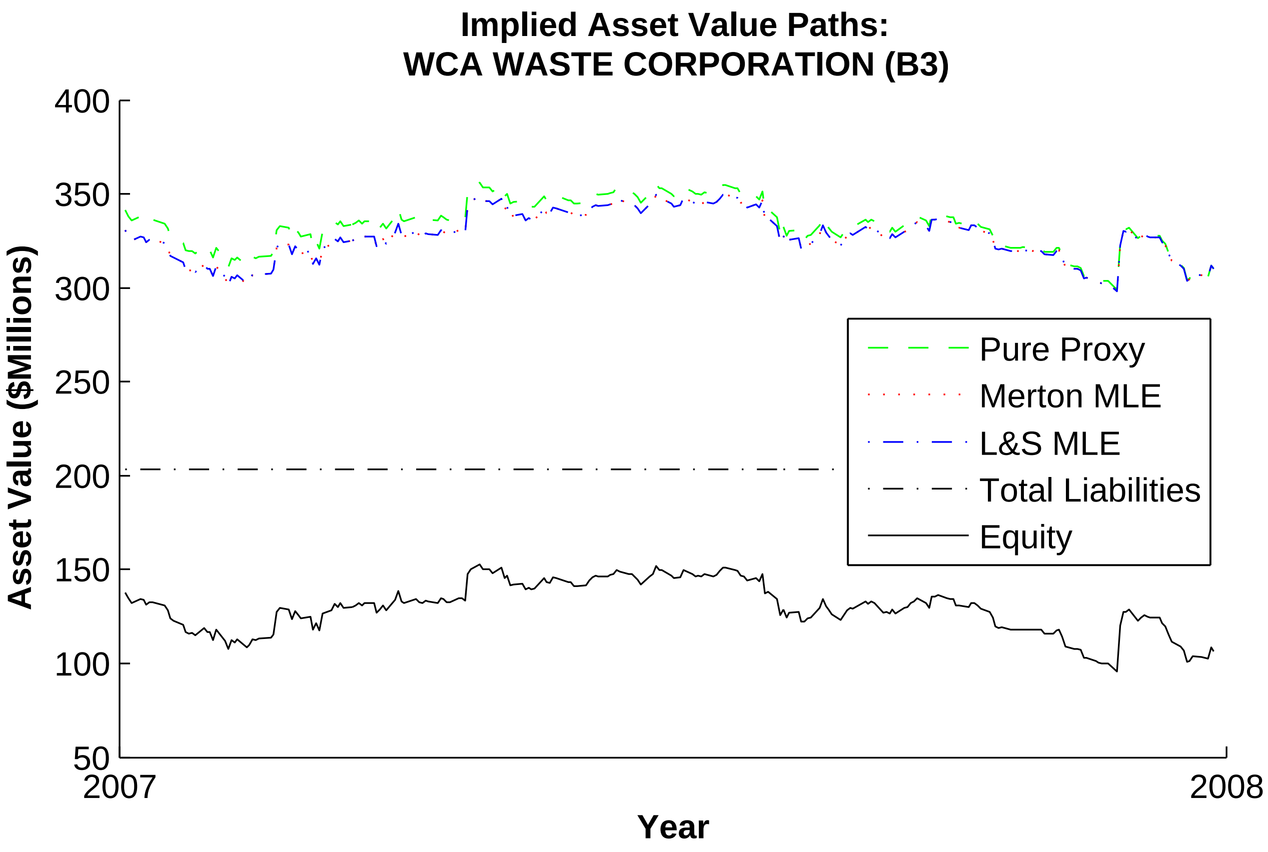

Implied Asset Value Paths:
MILLICOM INTERNATIONAL CELLULAR SA (B1)
Asset Value ($Millions)
16000
14000
12000
10000
8000
6000
4000
2000
2007
2008
Year
Pure Proxy
Merton MLE
L&S MLE
Total Liabilities
Equity

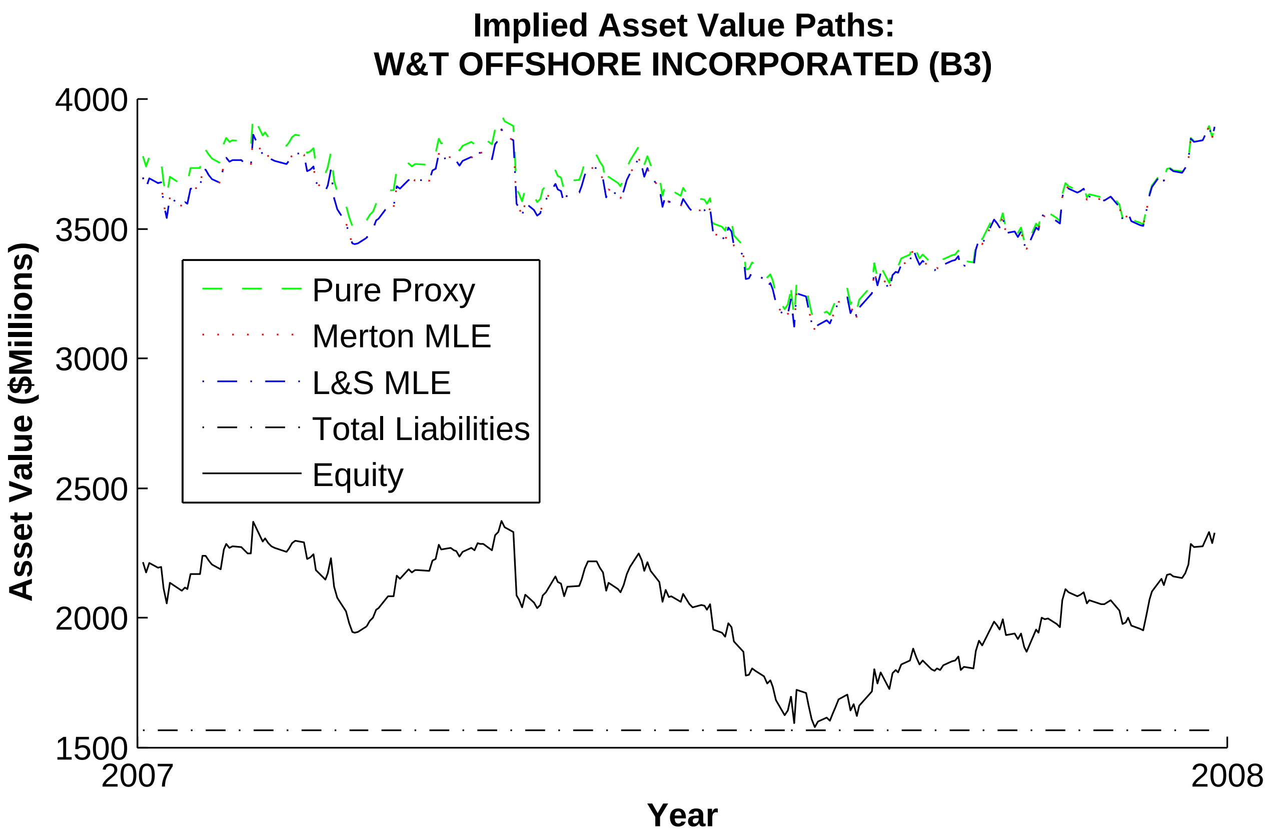

Implied Asset Value Paths:
PUBLIC SERVICE ENTERPRISE GROUP INCORPORATED (A3)
x 10^4
3.5
3
2.5
2
1.5
1
0.5
Asset Value ($Millions)
2007
2008
Year
Pure Proxy
Merton MLE
L&S MLE
Total Liabilities
Equity


Implied Asset Value Paths:
ROCKWELL COLLINS INCORPORATED (A1)
16000
14000
12000
10000
8000
6000
4000
2000
Asset Value ($Millions)
2007
2008
Year
Pure Proxy
Merton MLE
L&S MLE
Total Liabilities
Equity


Implied Asset Value Paths:
SCHLUMBERGER TECHNOLOGY CORPORATION (A2)
x 10^4
15
10
5
0
2007
2008
Year
Asset Value ($Millions)
Pure Proxy
Merton MLE
L&S MLE
Total Liabilities
Equity


Implied Asset Value Paths:
SK TELECOM COMPANY LIMITED (A2)
x 10^4
3
2.5
2
1.5
1
0.5
2006
2007
Year
Asset Value ($Millions)
Pure Proxy
Merton MLE
L&S MLE
Total Liabilities
Equity


Implied Asset Value Paths:
WCA WASTE CORPORATION (B3)
Asset Value ($Millions)
400
350
300
250
200
150
100
50
2007
2008
Year
Pure Proxy
Merton MLE
L&S MLE
Total Liabilities
Equity

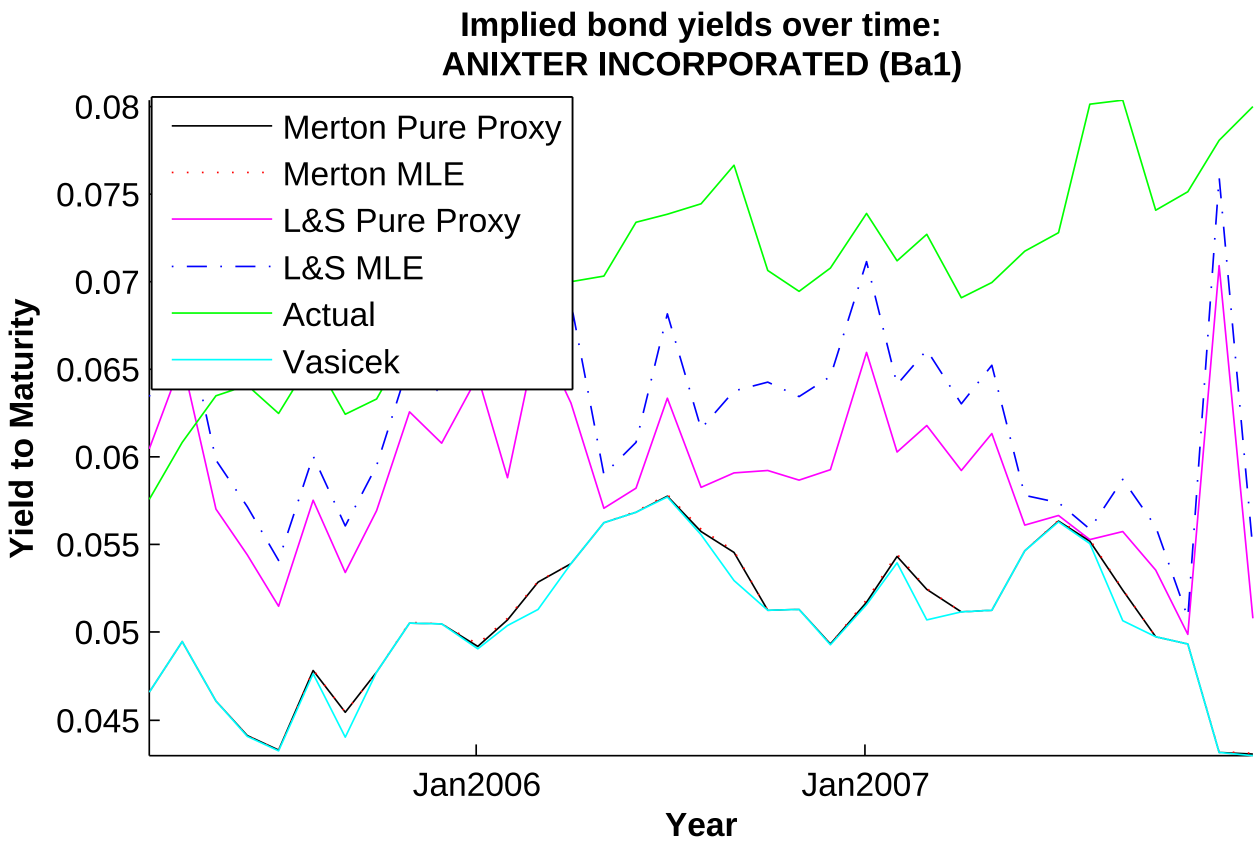

Implied Asset Value Paths:
W&T OFFSHORE INCORPORATED (B3)
Asset Value ($Millions)
4000
3500
3000
2500
2000
1500
2007
2008
Year
Pure Proxy
Merton MLE
L&S MLE
Total Liabilities
Equity

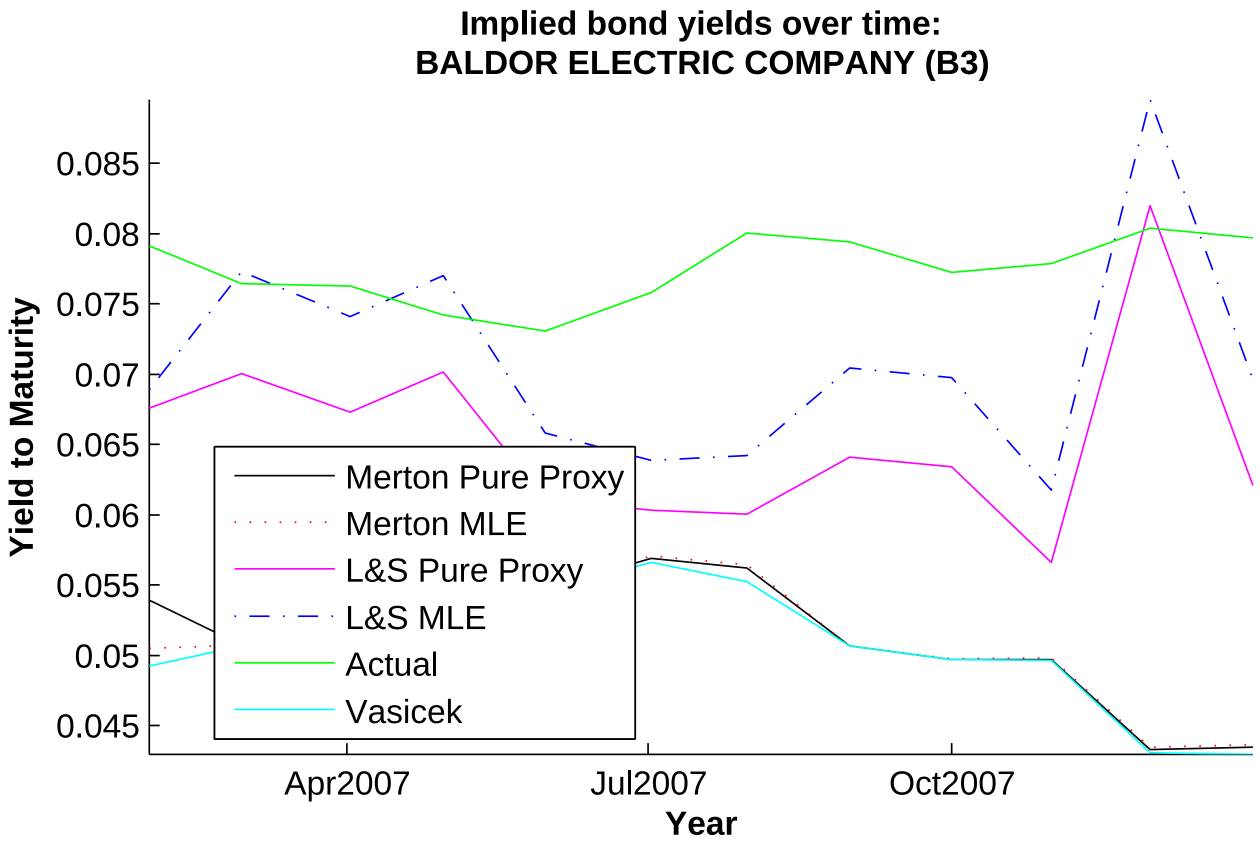
Appendix E

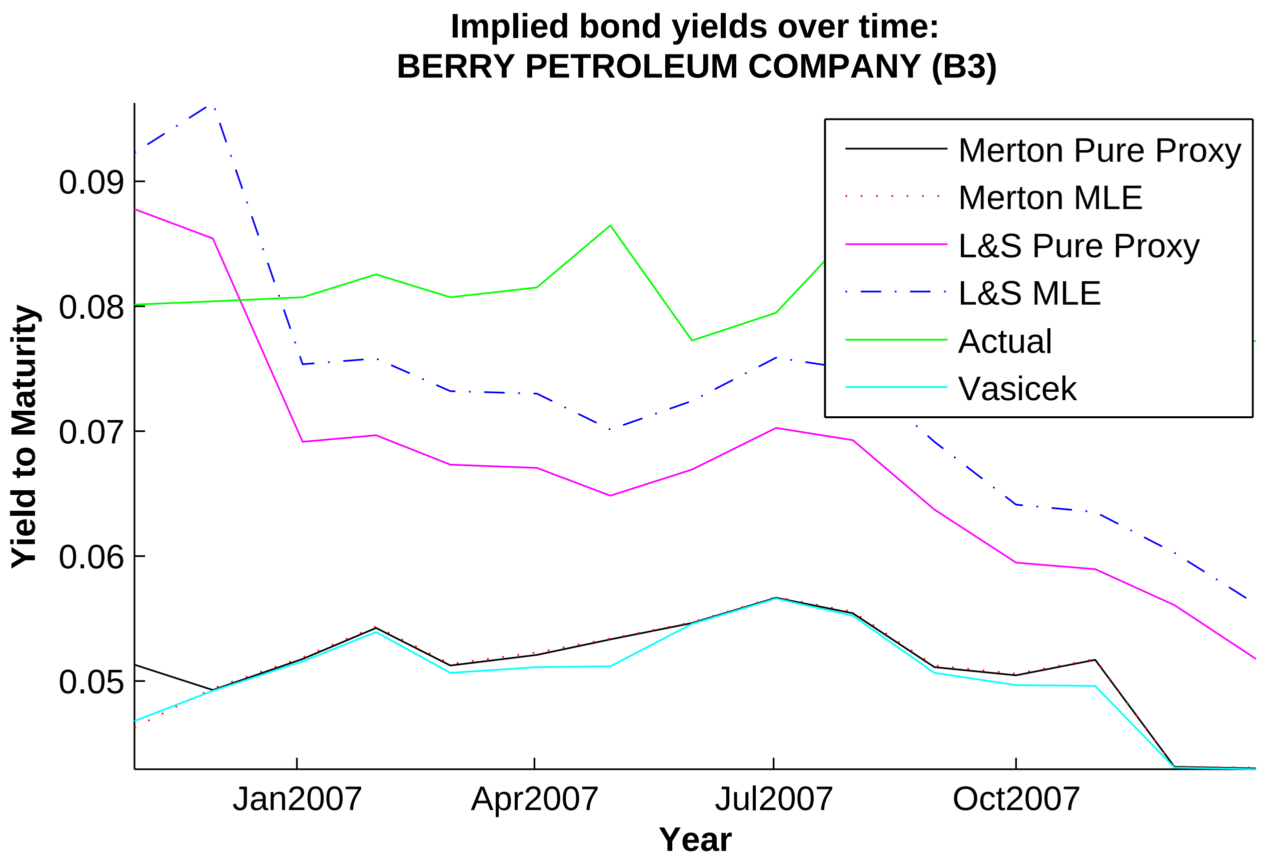
Plots of historic bond yield versus predicted yields

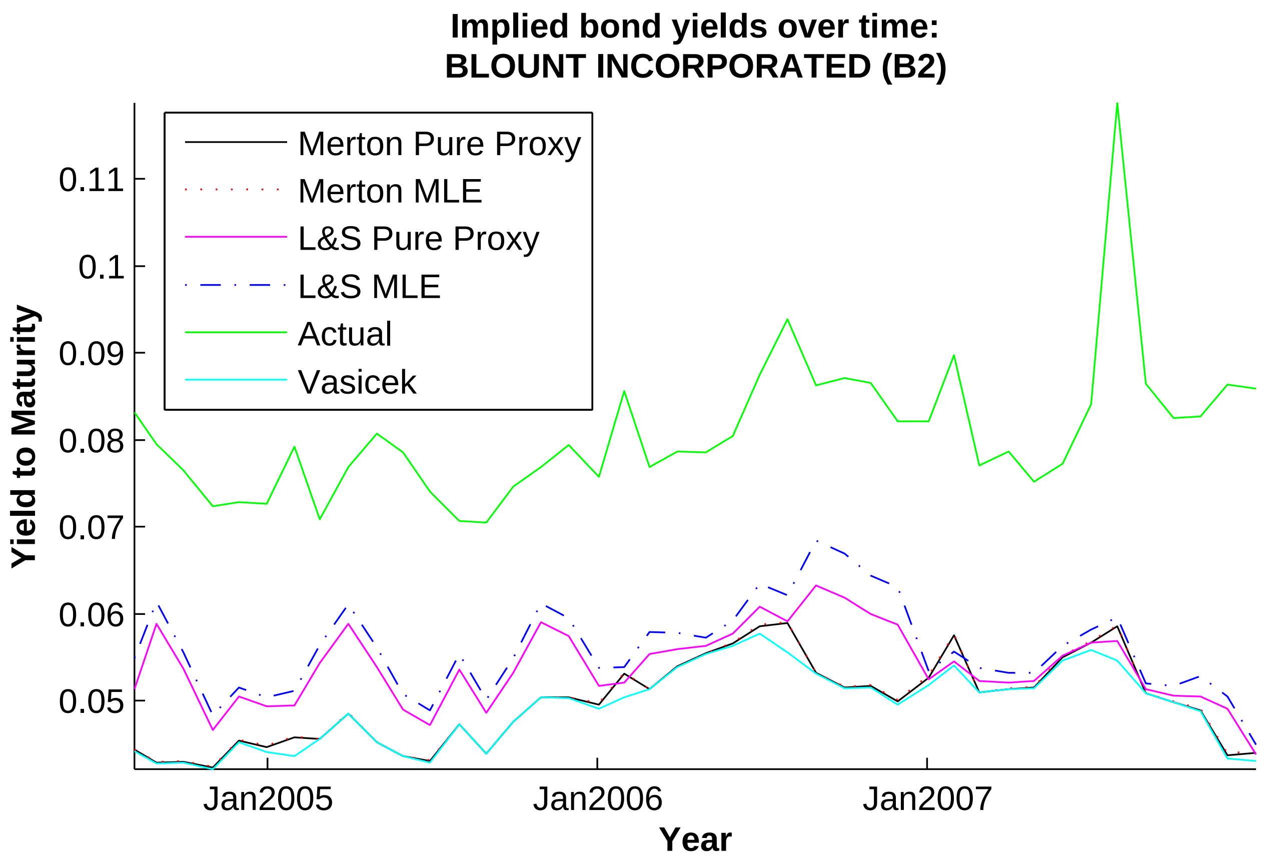
Graphs of the yield to maturity of each bond measured on the first available business day at the start of the month, starting at month of issue and extending as far as into 2008 as there are available market observations for the bond price. Each graph includes plots of the actual yield to maturity if purchased at the observed price, as well as yield estimates for the Merton [1974], Longstaff and Schwartz [1995] models when implemented with pure proxy and MLE asset parameter estimates.

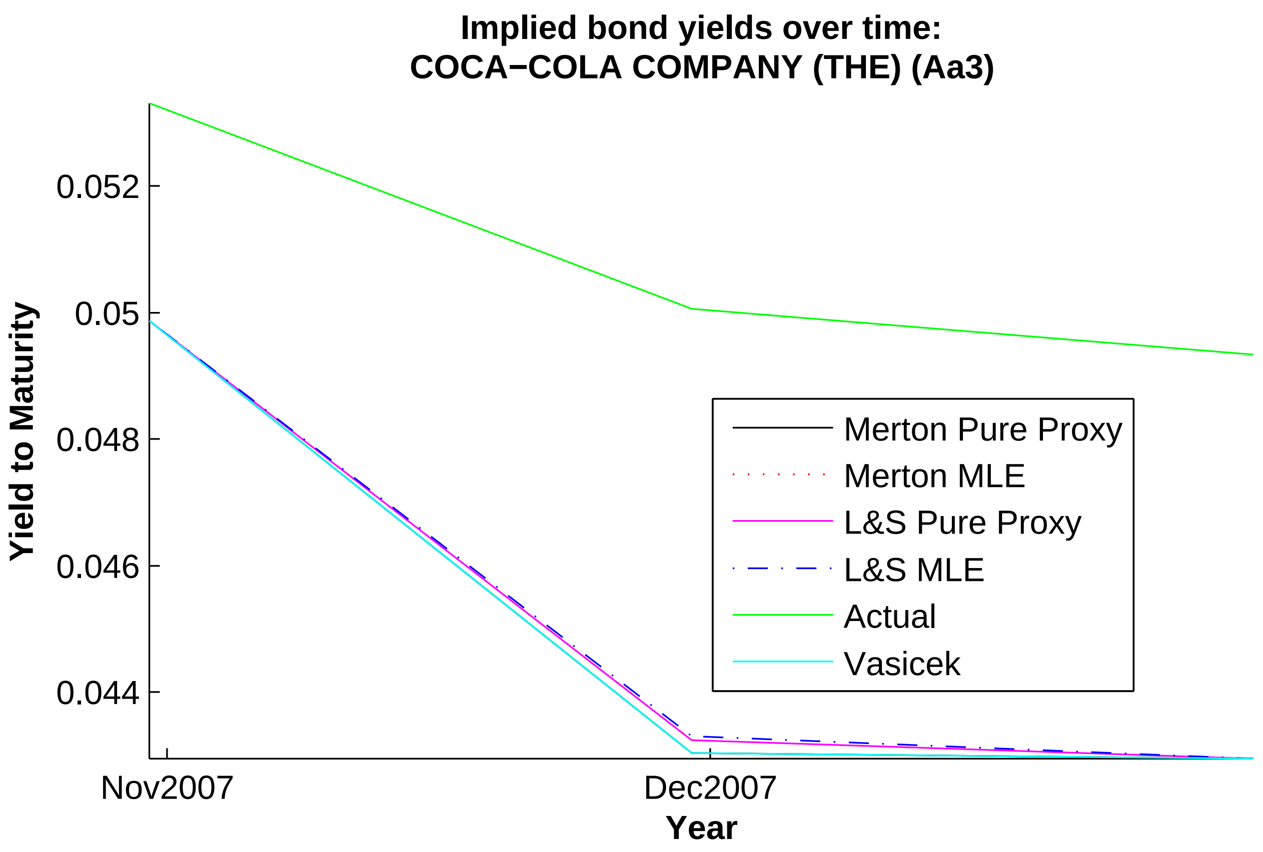
Included for reference is the implied yield for an otherwise equivalent risk-less bond priced off the Vasicek [1977] interest rate model fitted to the observed US Treasury yield curve from the date of the bond price observation.

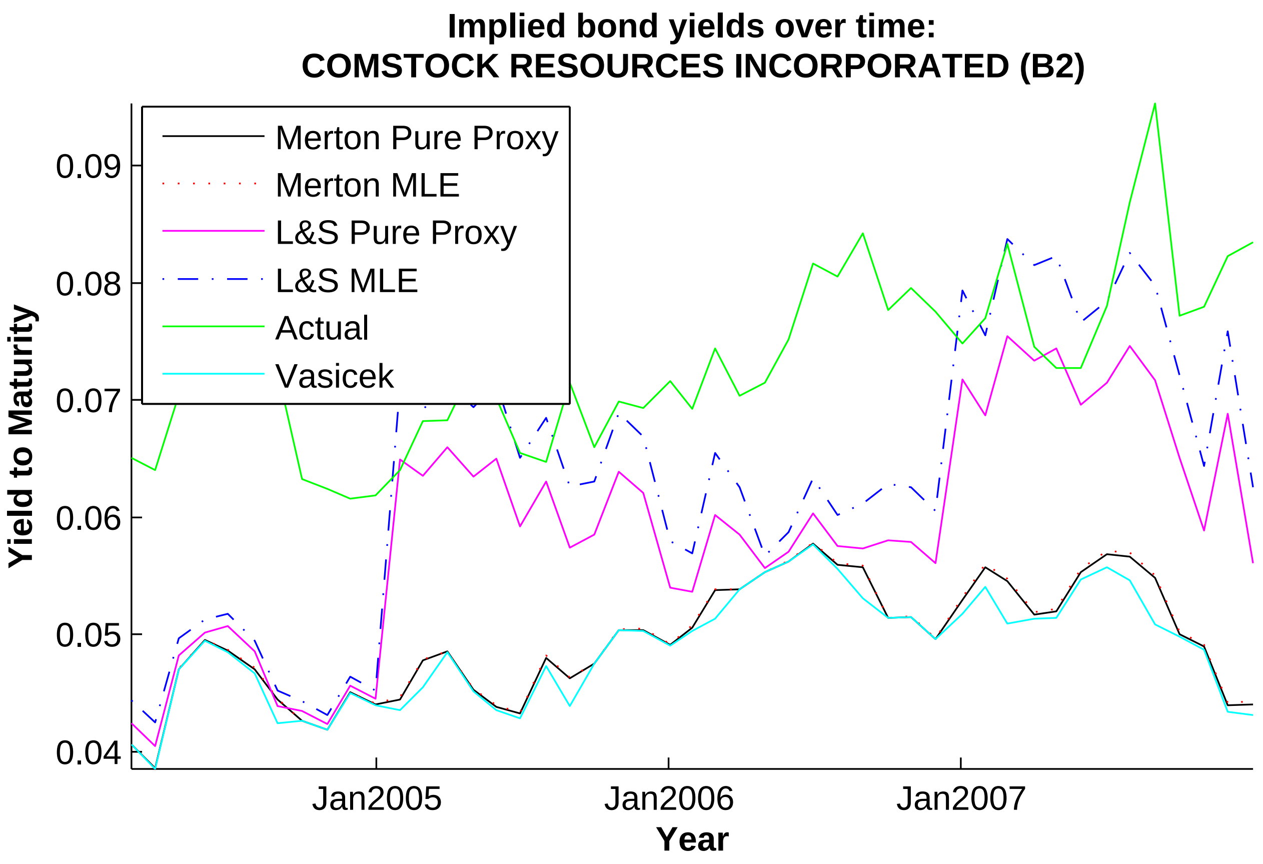

Implied bond yields over time:
ANIXTER INCORPORATED (Ba1)
Yield to Maturity
0.08
0.075
0.07
0.065
0.06
0.055
0.05
0.045
Jan2006
Jan2007
Year
Merton Pure Proxy
Merton MLE
L&S Pure Proxy
L&S MLE
Actual
Vasicek

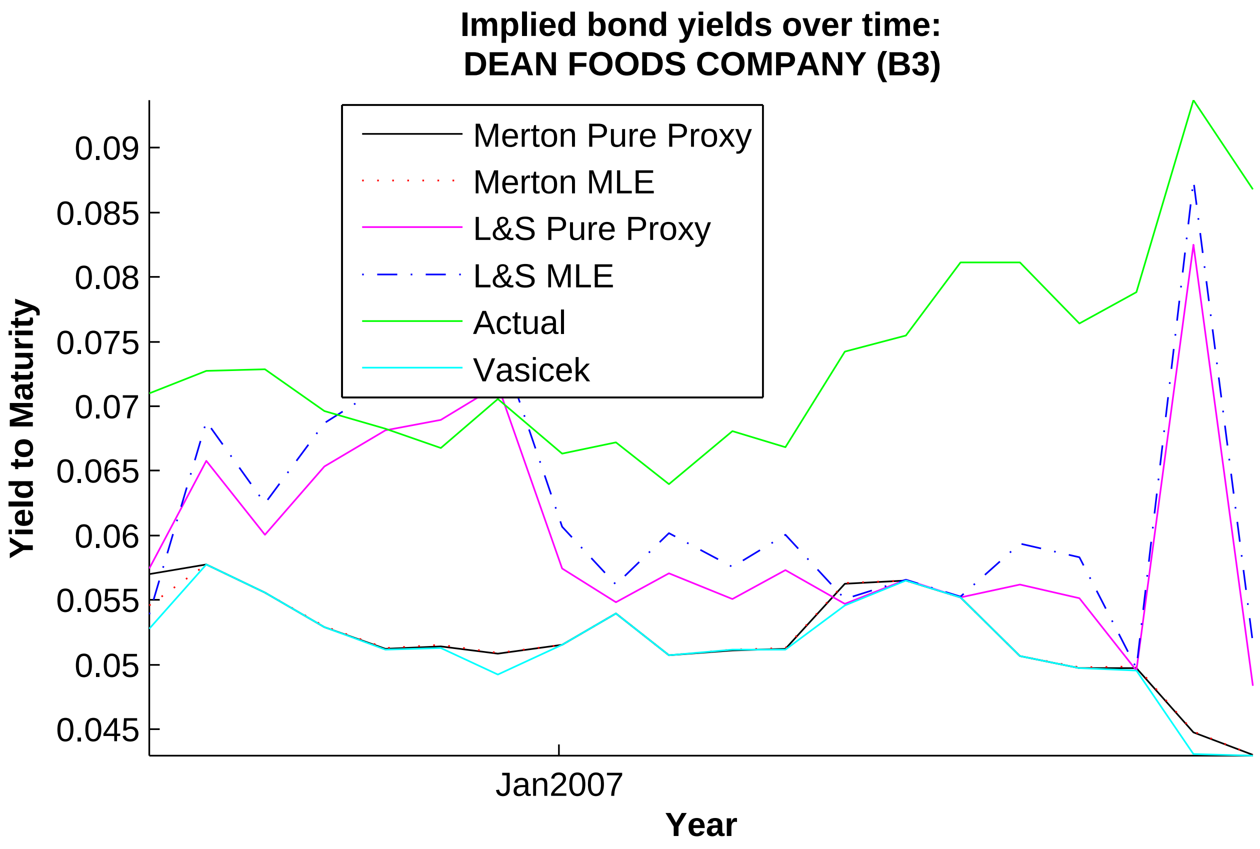

Implied bond yields over time:
BALDOR ELECTRIC COMPANY (B3)
Yield to Maturity
0.085
0.08
0.075
0.07
0.065
0.06
0.055
0.05
0.045
Apr2007
Jul2007
Oct2007
Year
Merton Pure Proxy
Merton MLE
L&S Pure Proxy
L&S MLE
Actual
Vasicek

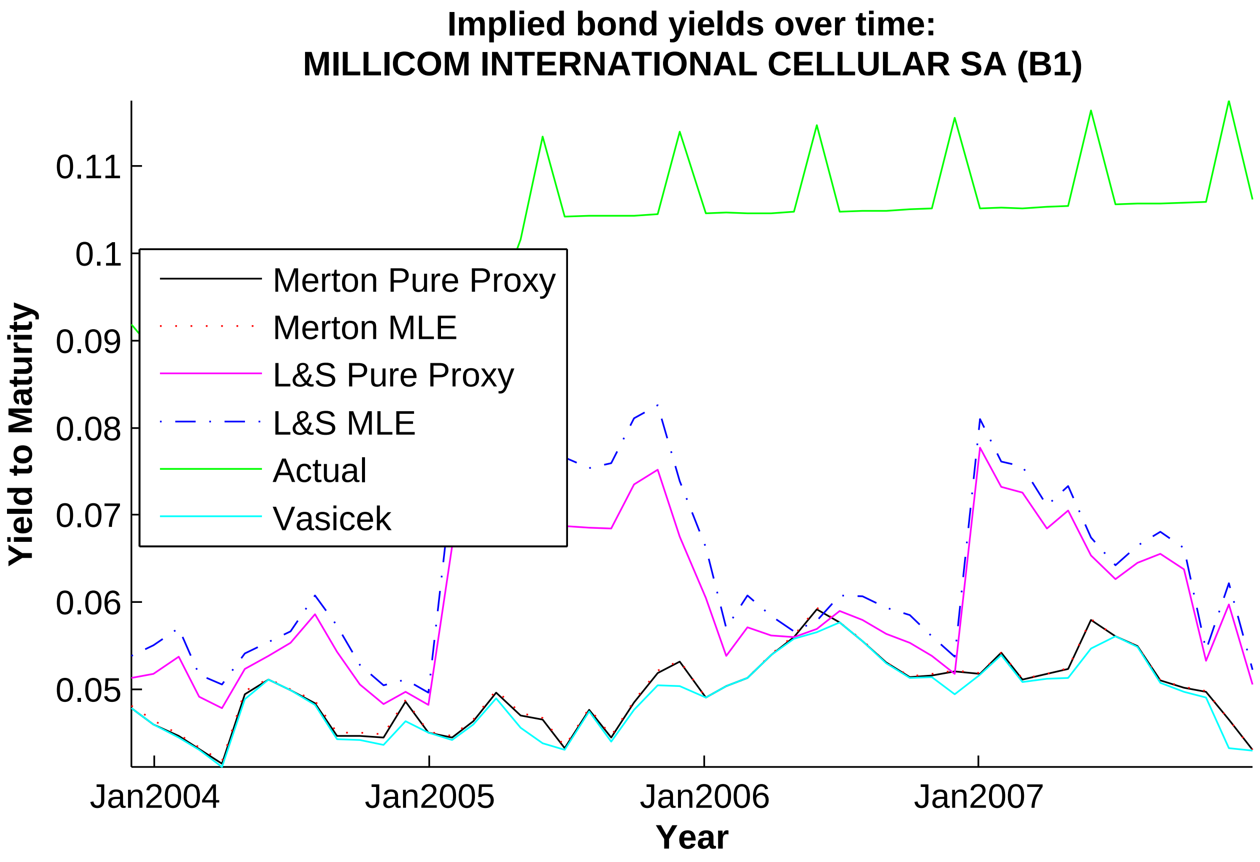

Implied bond yields over time:
BERRY PETROLEUM COMPANY (B3)
Yield to Maturity
0.09
0.08
0.07
0.06
0.05
Jan2007
Apr2007
Jul2007
Oct2007
Year
Merton Pure Proxy
Merton MLE
L&S Pure Proxy
L&S MLE
Actual
Vasicek

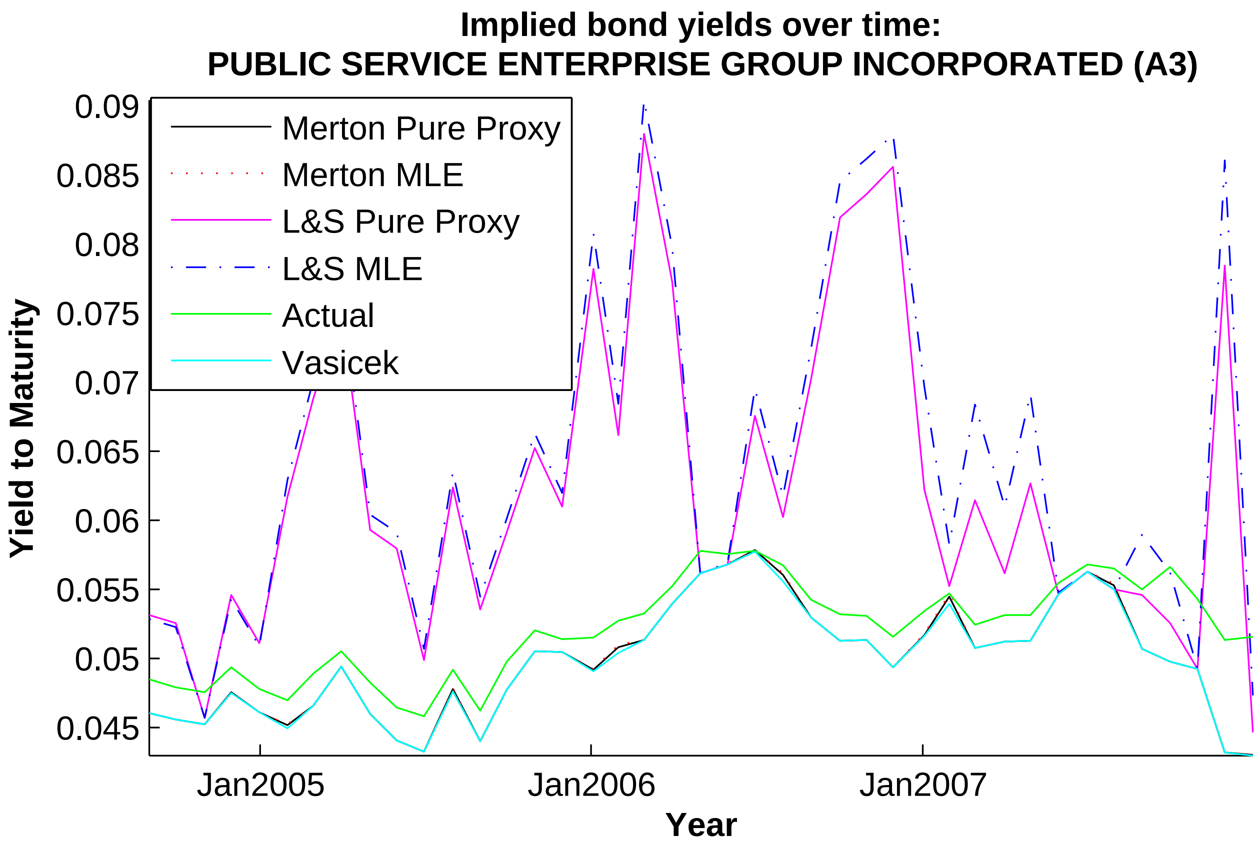

Implied bond yields over time:
BLOUNT INCORPORATED (B2)
Yield to Maturity
0.11
0.1
0.09
0.08
0.07
0.06
0.05
Jan2005
Jan2006
Jan2007
Year
Merton Pure Proxy
Merton MLE
L&S Pure Proxy
L&S MLE
Actual
Vasicek

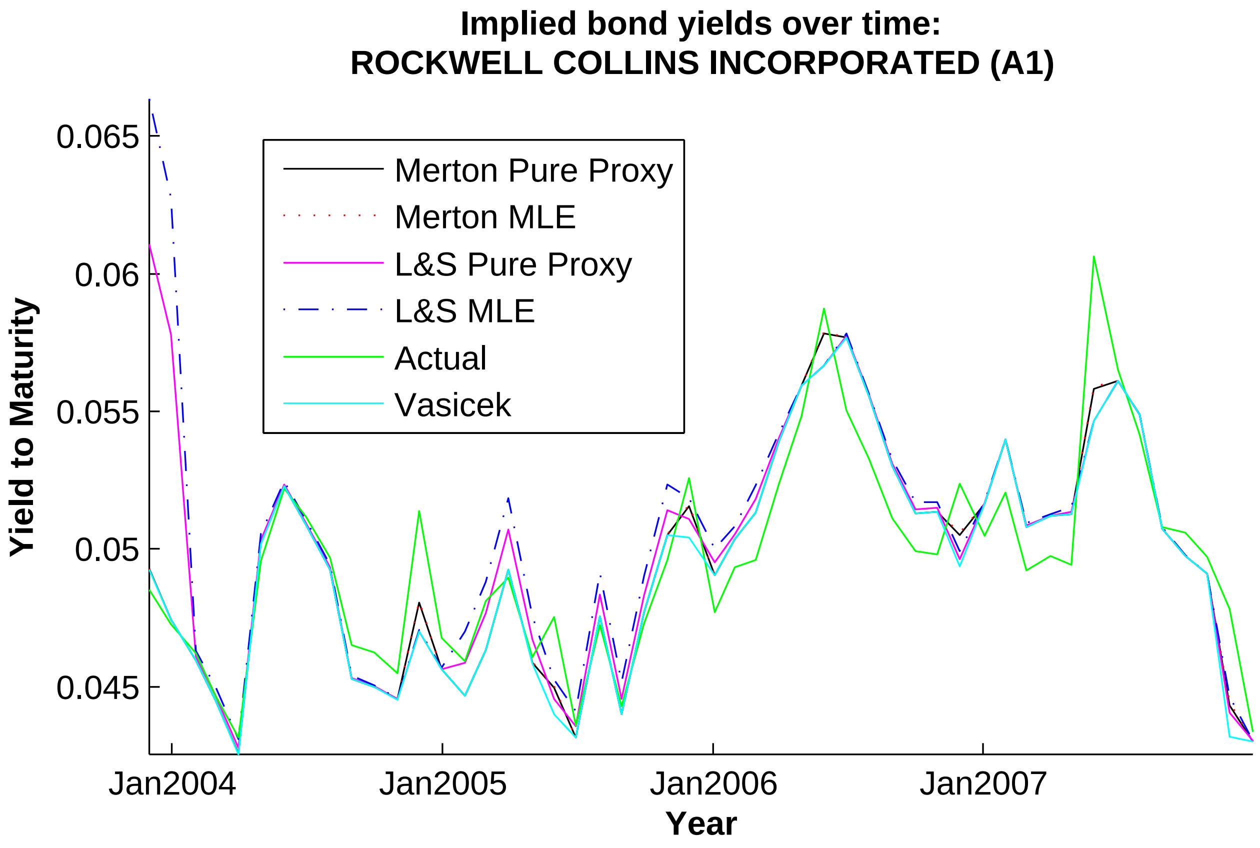

Implied bond yields over time:
COCA−COLA COMPANY (THE) (Aa3)
Yield to Maturity
0.052
0.05
0.048
0.046
0.044
Nov2007
Dec2007
Year
Merton Pure Proxy
Merton MLE
L&S Pure Proxy
L&S MLE
Actual
Vasicek

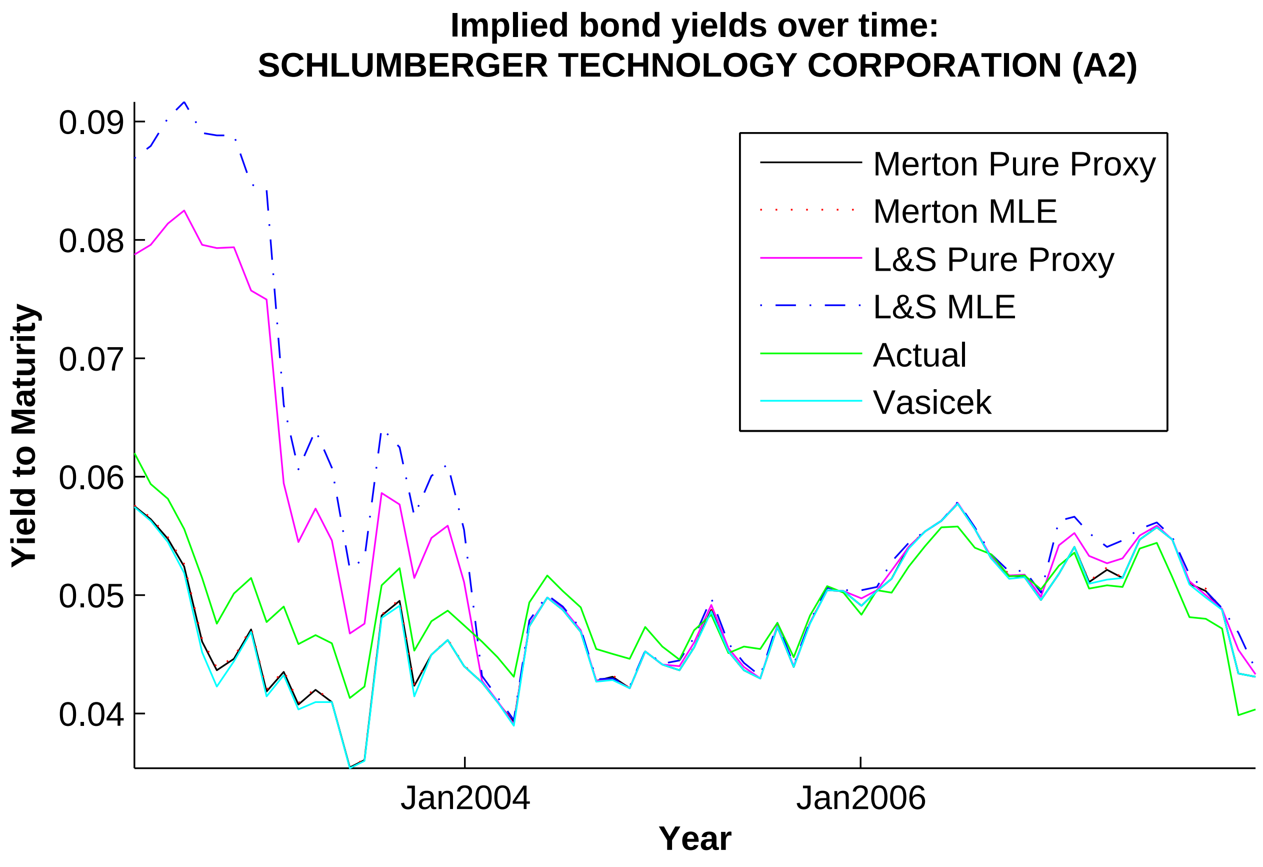

Implied bond yields over time:
COMSTOCK RESOURCES INCORPORATED (B2)
Yield to Maturity
0.09
0.08
0.07
0.06
0.05
0.04
Jan2005
Jan2006
Jan2007
Year
Merton Pure Proxy
Merton MLE
L&S Pure Proxy
L&S MLE
Actual
Vasicek

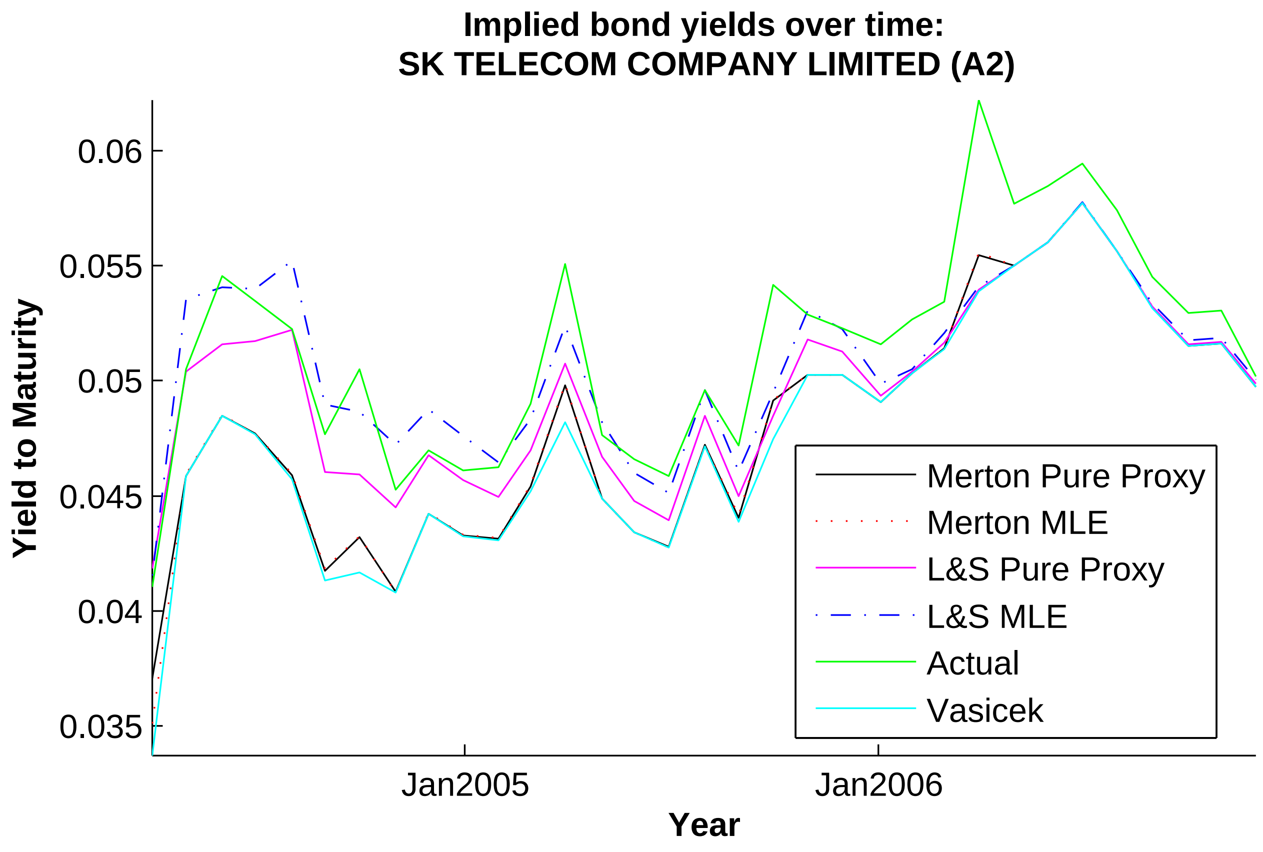

Implied bond yields over time:
DEAN FOODS COMPANY (B3)
Merton Pure Proxy
Merton MLE
L&S Pure Proxy
L&S MLE
Actual
Vasicek
Yield to Maturity
0.09
0.085
0.08
0.075
0.07
0.065
0.06
0.055
0.05
0.045
Jan2007
Year

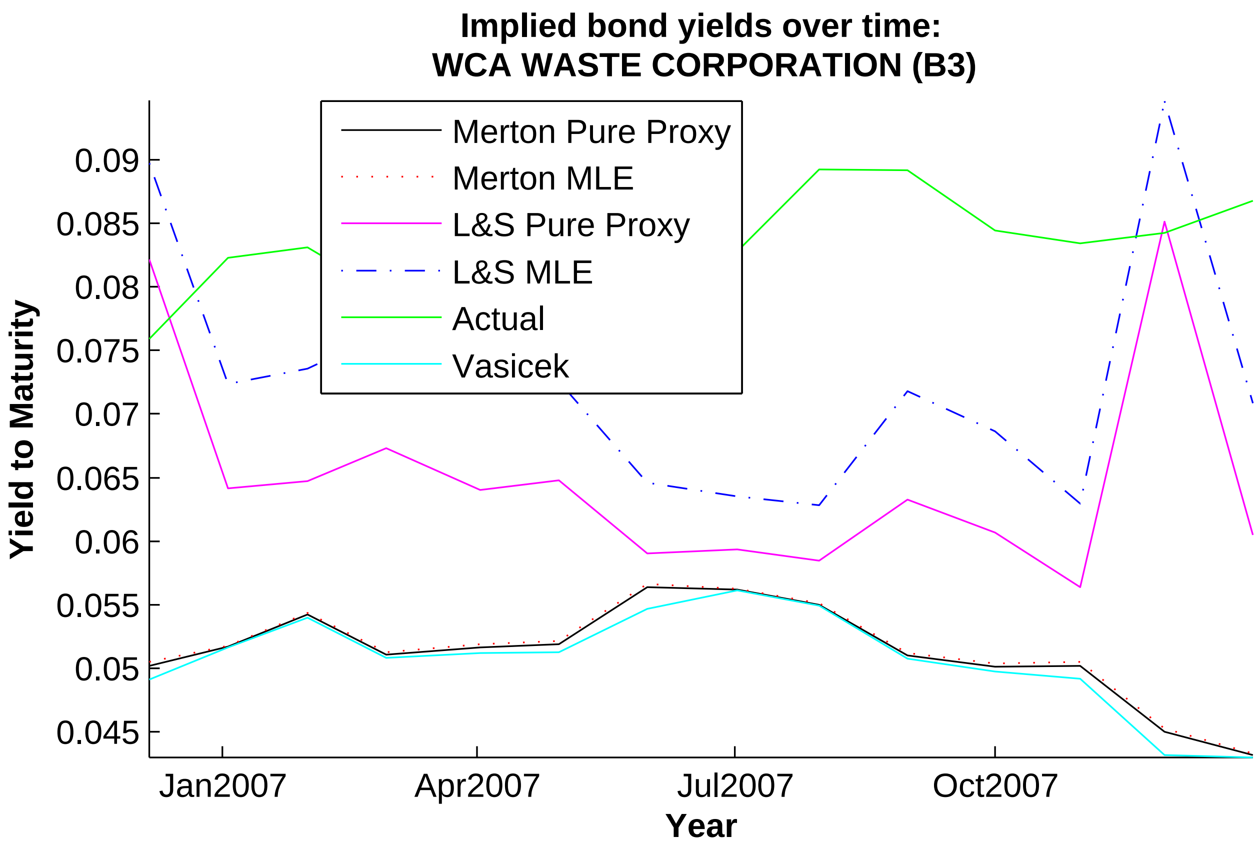

Implied bond yields over time:
MILLICOM INTERNATIONAL CELLULAR SA (B1)
Merton Pure Proxy
Merton MLE
L&S Pure Proxy
L&S MLE
Actual
Vasicek
Yield to Maturity
0.11
0.1
0.09
0.08
0.07
0.06
0.05
Jan2004
Jan2005
Jan2006
Jan2007
Year

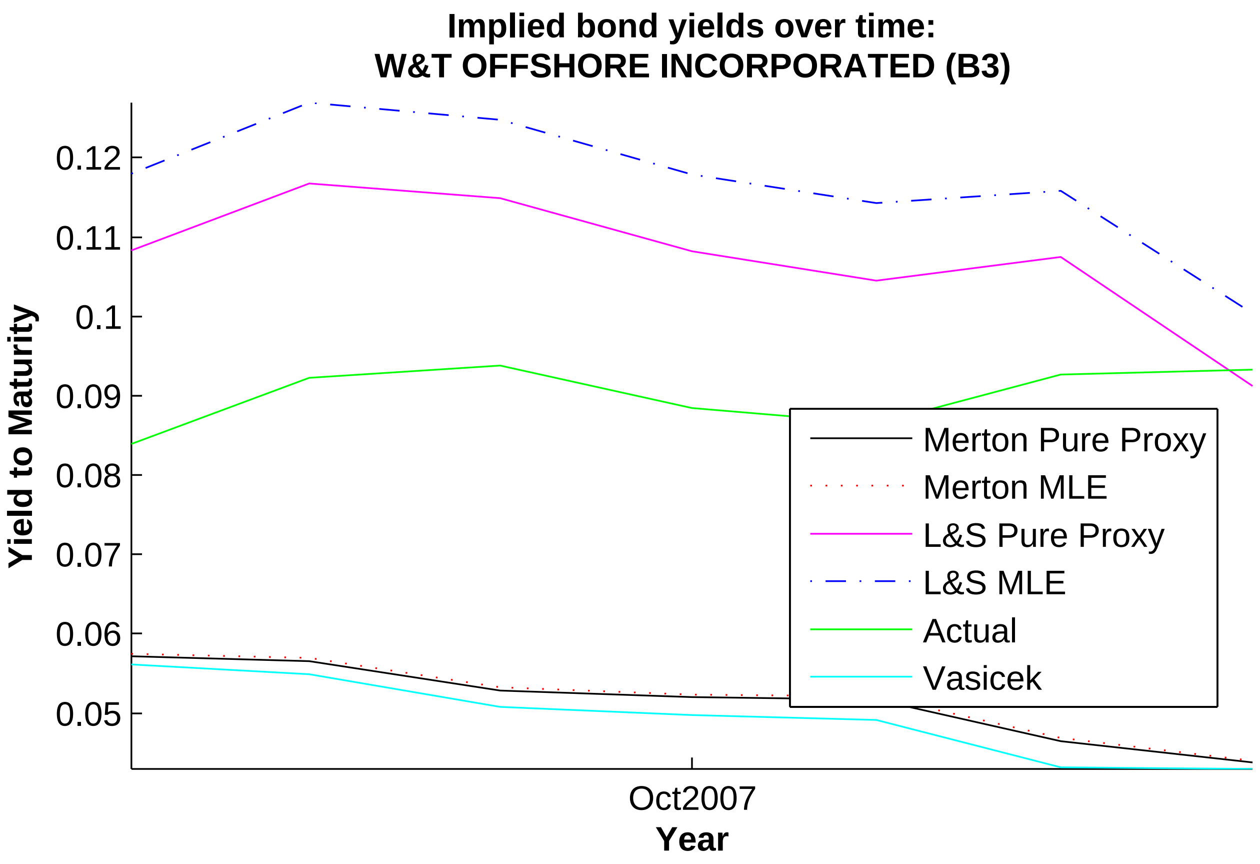

Implied bond yields over time:
PUBLIC SERVICE ENTERPRISE GROUP INCORPORATED (A3)
Merton Pure Proxy
Merton MLE
L&S Pure Proxy
L&S MLE
Actual
Vasicek
Yield to Maturity
0.09
0.085
0.08
0.075
0.07
0.065
0.06
0.055
0.05
0.045
Jan2005
Jan2006
Jan2007
Year


Implied bond yields over time:
ROCKWELL COLLINS INCORPORATED (A1)
Merton Pure Proxy
Merton MLE
L&S Pure Proxy
L&S MLE
Actual
Vasicek
Yield to Maturity
0.065
0.06
0.055
0.05
0.045
Jan2004
Jan2005
Jan2006
Jan2007
Year


Implied bond yields over time:
SCHLUMBERGER TECHNOLOGY CORPORATION (A2)
Yield to Maturity
0.09
0.08
0.07
0.06
0.05
0.04
Jan2004
Jan2006
Year
Merton Pure Proxy
Merton MLE
L&S Pure Proxy
L&S MLE
Actual
Vasicek


Implied bond yields over time:
SK TELECOM COMPANY LIMITED (A2)
Yield to Maturity
0.06
0.055
0.05
0.045
0.04
0.035
Jan2005
Jan2006
Year
Merton Pure Proxy
Merton MLE
L&S Pure Proxy
L&S MLE
Actual
Vasicek


Implied bond yields over time:
WCA WASTE CORPORATION (B3)
Merton Pure Proxy
Merton MLE
L&S Pure Proxy
L&S MLE
Actual
Vasicek
Yield to Maturity
0.09
0.085
0.08
0.075
0.07
0.065
0.06
0.055
0.05
0.045
Jan2007
Apr2007
Jul2007
Oct2007
Year


Implied bond yields over time:
W&T OFFSHORE INCORPORATED (B3)
Merton Pure Proxy
Merton MLE
L&S Pure Proxy
L&S MLE
Actual
Vasicek
Yield to Maturity
0.12
0.11
0.1
0.09
0.08
0.07
0.06
0.05
Oct2007
Year

Appendix F

Matlab code for bond pricing models, interest rates, yields

```
function [BSCallEuro] = BlackScholesEuroCall(X,r,sigma,tau,V)
%-----------------------------------------------------------------------
% @description: BSCallEuro
%        Black-Scholes european call option.
% @params:
%  X      - Exercise/Strike price of option.
%  r      - Short-term risk-free interest rate
%  sigma    - Standard deviation of company asset value. Cannot be exactly
%        zero lest we get Divide-By-Zero errors.. use ZeroClean()
%        function to catch this.
%  tau      - Time to maturity of call option
%  V      - Current value of some traded asset, be it a share price, full
%        firm asset value, whatever...
%-----------------------------------------------------------------------

  % Default the value of the european call to be as low as it can
  % possibly be, i.e. zero.
  BSCallEuro    = 0;
```

```
% Perform data cleaning. For example, many parameters cannot be set to
% be exactly zero lest we get Divide-By-Zero errors, so before we begin
% any calculations, clean those values now:
X      = ZeroClean(X);
sigma    = ZeroClean(sigma);
tau      = ZeroClean(tau);
V      = ZeroClean(V);

% Calculate d1 and d2 placeholder variable values
d1      = BSCallEuroD1(X,r,sigma,tau,V);
d2      = BSCallEuroD2(X,r,sigma,tau,V);

% Calculate price of option
BSCallEuro  = V.*N(d1) - X*exp(-r*tau).*N(d2);
%%% End european call option pricing logic %%%

%%% Begin private methods %%%
%------------------------------------------------------------------------
% @description: Corresponds to LiWong2008 Appendix B definition of
%        Black-Scholes standard call option model parameter d1
%------------------------------------------------------------------------
function outBSCED1 = BSCallEuroD1(X,r,sigma,tau,V)
  outBSCED1 = (log(V/X) + tau.*(r+0.5*sigma^2))./(sqrt(tau).*sigma);
end

%------------------------------------------------------------------------
% @description: Corresponds to LiWong2008 Appendix B definition of
%        Black-Scholes standard call option model parameter d2.
%        N(d2) = Risk neutral probability of achieving strike price.
% @see:      http://www.bionicturtle.com/forum/viewthread/464/
%------------------------------------------------------------------------
```

```
  function outBSCED2 = BSCallEuroD2(X,r,sigma,tau,V)
    outBSCED2 = (log(V/X) + (r-0.5*sigma^2)*tau)./(sigma*sqrt(tau));
  end
end

function [ImpliedYield] = CalcImpliedYield(faceVal,coupon,taus,bondPrice)
%--------------------------------------------------------------------------
% @description:  CalculateImpliedYield
%        Based on some observed or predicted bond price, and values
%        for the remaining coupon and face value payment rates and
%        maturities, calculate the implied bond yield which would
%        produce that bond price.
% @params:
%
% @example:
%        coupon  = 0.05;
%        taus  = [1/2 : 1/2 : 10];
%        faceVal  = 100;
%        bondPrice  = 97;
%        implYield = CalcImpliedYield(faceVal,coupon,taus,bondPrice)
%--------------------------------------------------------------------------

  % Min and max boundaries of our search, expect realistic values to be
  % between 0 and 0.15...
  minYield  = 0;
  maxYield  = 0.5;
  ImpliedYield = fminbnd(@(yieldGuess) SeekForYield(yieldGuess,coupon,...
      taus,bondPrice,faceVal),minYield,maxYield);

  %-----------------------------------------------------------------------
  % @description:  SeekForYield
  %        Method to use to search for optimum implied yield given
  %        an observed bond price, coupon rate, and maturity
  %        dates.
  %---------------------------------------------------------------------
  function [yieldError] = SeekForYield(yieldGuess,coupon,...
          taus,targetPrice,faceVal)

    % Calculate implied present value of all coupons using current yield
    % guess:
    implPresVal = 0.5 * coupon * faceVal * exp(-yieldGuess*taus);

    % Calculate also the present value of the fave value payment using the
    % current yield guess:
    matTau    = taus(end);
    implPresVal(end+1) = faceVal * exp(-yieldGuess*matTau);

    % Implied price given a particular yield is the sum of the present
    % value of all coupons and the face value payment.
    implPrice  = sum(implPresVal);

    % Calculate the difference between the implied price and the observed
    % price, to find the optimal implied yield.
    yieldError = abs(implPrice - targetPrice);
  end
end

function [downOutCall] = DownOutCall(H,r0,R,sigma,tau,V,X)
%----------------------------------------------------------------------
% @description: Down and Out Call option pricing formula as presented by
%    Li&Wong in their 2008 paper "Structural models of corporate bond
%    pricing with maximum likelihood estimation".
%        Specifically, see that paper, Appendix C.
% @params:
%  H      - The barrier level.
%  r0      - The instantaneous risk-free interest rate.
%  R      -Rebate paid to equity holders in the event of default (i.e.
%        asset value falls below the default barrier level H).
%  sigma    - Volatility (as std. deviation) of asset process. Constant
%        throughout time.
%  tau      - The time to maturity
%  V      - The (market) value of the assets upon which the call is
%        being written.
%  X      - The future promised payment (e.g.: face value of bond)
%-------------------------------------------------------------------------

  % Perform data cleaning. For example, many parameters cannot be set to
  % be exactly zero lest we get Divide-By-Zero errors, so before we begin
  % any calculations, clean those values now:
  H    = ZeroClean(H);
  sigma  = ZeroClean(sigma);
  V    = ZeroClean(V);
  X    = ZeroClean(X);
  tau    = ZeroClean(tau);

  % Calculate model placeholder parameters once here for slight increase
  % in efficiency and clarity of notation.
  docA  = DownOutCallA(H,r0,sigma,tau,V,X);
  docB  = DownOutCallB(H,r0,sigma,tau,V,X);
  docC  = DownOutCallC(H,r0,sigma,tau,V);
  docEta  = DownOutCallEta(r0, sigma);

  % Calculate price of down-and-out call option
  downOutCall = V*N(docA) - X*exp(-r0*tau)*N(docA-sigma*sqrt(tau)) - ...
    V*(H/V)^(2*docEta)*N(docB) + ...
    X*exp(-r0*tau)*(H/V)^(2*docEta - 2)*N(docB - sigma*sqrt(tau)) + ...
    R*(H/V)^(2*docEta - 1)*N(docC) + ...
    R*(V/H)*N(docC - 2*docEta*sigma*sqrt(tau));

  % Finally, catch the instances where the above formula would generate a
  % negative price, when in fact, it should have a floor of zero:
  downOutCall = max(downOutCall,0);

  %%% End down-out-call main logic %%%
end

function outDOCA = DownOutCallA(H,r0,sigma,tau,V,X)
%-------------------------------------------------------------------------
% @description:  Corresponds to Li&Wong2008 Appendix C definition of model
%        parameter a
%-------------------------------------------------------------------------
  if X >= H
    outDOCA = DownOutCallGeneric((V/X),r0,sigma,tau);
  else
    outDOCA = DownOutCallGeneric((V/H),r0,sigma,tau);
  end
end

function outDOCB = DownOutCallB(H,r0,sig,tau,V,X)
%-------------------------------------------------------------------------
% @description:  Corresponds to Li&Wong2008 Appendix C definition of model
%        parameter b
%-------------------------------------------------------------------------
  if X >= H
```

```
    outDOCB = DownOutCallGeneric((H^2/(V*X)),r0,sig,tau);
  else
    outDOCB = DownOutCallGeneric((H/V),r0,sig,tau);
  end
end

function outDOCC = DownOutCallC(H,r0,sigma,tau,V)
%------------------------------------------------------------------------
% @description:  Corresponds to Li&Wong2008 Appendix C definition of model
%        parameter c
%------------------------------------------------------------------------
  outDOCC = DownOutCallGeneric((H/V),r0,sigma,tau);
end

function [docDelta] = DownOutCallDelta(H,r0,R,sigma,tau,V,X)
%-------------------------------------------------------------------------
% @description: DownOutCallDelta
%                               Calculate the delta of a Down and Out Call option using
%                               pricing formula as presented by Li
%                               and Wong in their 2008 paper "Structural models of corpo
%                               bond pricing with maximum likelihood estimation".
%                               Specifically, see that paper, Appendix C.
% @params:
%       H                       - The barrier level at which default occurs.
%       r0                      - The instantaneous risk-free interest rate.
%       R                       -Rebate paid to equity holders in the event of default (
%                               asset value falls below the default barrier level H).
%       sigma           - Volatility (as std. deviation) of asset process. Constant
%                               throughout time.
%       tau                     - The time to maturity
%       V                       - The (market) value of the assets upon which the call i
%                                being written.
%       X                        - The future promised payment (e.g.: face value of bond)
%---------------------------------------------------------------------------


        % Perform data cleaning. For example, many parameters cannot be set to
        % be exactly zero lest we get Divide-By-Zero errors, so before we begin
        % any calculations, clean those values now:
        H               = ZeroClean(H);
        sigma   = ZeroClean(sigma);
        V               = ZeroClean(V);
        X               = ZeroClean(X);

        % Precalculate model placeholder parameters once here for increase
        % in efficiency and clarity of notation.
        docA    = DownOutCallA(H,r0,sigma,tau,V,X);
        docB    = DownOutCallB(H,r0,sigma,tau,V,X);
        docC    = DownOutCallC(H,r0,sigma,tau,V);
        nu              = DownOutCallNu(r0, sigma);

        sst     = sigma*sqrt(tau);
        ptA     = N(docA);
        ptB     = 0;
        ptC     = (H/V)^(2*nu)*((-2*nu+1)*N(docB)-normpdf(docB)/sst);
        ptD     = X*exp(-r0*tau)/V*(H/V)^(2*nu-2)*((-2*nu+2)*...
                N(docB-sst)-normpdf(docB-sst)/sst);
        ptE     = R/V*(H/V)^(2*nu-1)*((-2*nu+1)*N(docC)-normpdf(docC)/sst);
        ptF     = R/H*(N(docC-2*nu*sst)-normpdf(docC-2*nu*sst)/sst);

        docDelta = ptA - ptB - ptC + ptD + ptE + ptF;
end
```

```
function outDOCEta = DownOutCallEta(drift, stddev)
%-------------------------------------------------------------------------
% @description:  Corresponds to Li&Wong2008 Appendix C definition of model
%         parameter eta.
% @alert:    LiWong2008 paper contains a typo, their formula is missing
%         the 'equals' sign that indicates the definition of their
%         parameter 'eta'. I referred to an older version of their
%         paper to clarify this.
%----------------------------------------------------------------------
  outDOCEta = drift / stddev^2 + 1/2;
end

function outDOCG = DownOutCallGeneric(logInput,r0,sigma,tau)
%----------------------------------------------------------------------
% @description:  DownOutCallGeneric
%         Generic function to generate Li&Wong2008 Appendix C DOC
%         option pricing formula internal parameters a, b and c.
%----------------------------------------------------------------------
  outDOCG = (log(logInput) + (r0 + 0.5*sigma^2)*tau) / (sigma*sqrt(tau));
end

function outDOCNu = DownOutCallNu(drift, stddev)
%-------------------------------------------------------------------------
% @description:  Corresponds to Li&Wong2008 Appendix C definition of model
%         parameter eta.
% @alert:    LiWong2008 paper contains a typo, their formula is missing
%         the 'equals' sign that indicates the definition of their
%         parameter 'nu'. I referred to an older version of their
%         paper to clarify this.
%----------------------------------------------------------------------
  outDOCNu = drift / stddev^2 + 1/2;
```

```
end


function [ImpliedValue] = ImpliedAssetValueBlackScholes(...
    K,mu,r0,sigma,S,tau)
%-----------------------------------------------------------------------
% @description:  ImpliedAssetValueBlackScholes
%        Calculate the implied asset value given all other
%        relevant option pricing paramters for a standard
%        Black-Scholes European call option.
%-----------------------------------------------------------------------
  % We assume that the asset value is strictly non-negative when
  % searching for its implied value. Make the (not too bold) assumption
  % that asset value is no more than 100 times the strikes price, when
  %  setting a test range.
  range    = [0, K*100];

  % Specify some optimisation options:
  options        = optimset('fzero');
  options.TolFun    = 10^-5;
  options.MaxIter    = 1*200;
  options.MaxFunEvals  = 1*200;
  options.Display    = 'notify';  % Notify only if it did NOT converge.
%   options.Display    = 'iter';
  options.FunValCheck  = 'on';



  % Search for the implied value of the asset, based on the current value
  % of the option and its defining parameters:
  ImpliedValue = fzero(@(V) ZeroBSCallEuro(S,K,r0,sigma,tau,V),...
      range,options);
  %%% End logic to find optimal bond price
  %%% Begin private methods
  %-----------------------------------------------------------------------
  % @description: Implied asset value objective function.
  %        The objective function simply calculates the difference
  %        between observed market value, or price, of the option
  %        and the theoretical value derived from the
  %        Black-Scholes model with some estimated asset value...
  %        We aim to minimise the difference!
  % @notes:    Based on the bsimplv.m functionality included by
  %        default in MatLab.
  %-----------------------------------------------------------------------
  function zero = ZeroBSCallEuro(S,K,r0,sigma,tau,V)

    % Inbuilt Matlab library for BS options... is SLOW!! :(
     %[call put]  = blsprice(V, K, r0, tau, sigma, 0);
      %zero  = S - call;

    % Use my Black-Scholes call function, NOT the default Matlab one...
    % The default has some nice error checking etc, but it is VASTLY
    % slower than my version when we are forced to do the checks one by
    % one, such as when searching for optimum parameters.
    zero = S - BlackScholesEuroCall(K,r0,sigma,tau,V);
  end
end

function [ImpliedValue] = ImpliedAssetValueDownAndOutCall(...
    K,mu,r0,sigma,S,tau)
%-----------------------------------------------------------------------
% @description:  ImpliedAssetValueDownAndOutCall
%        Calculate the implied asset value given all other
%         relevant option pricing paramters for a down-and-out call
%         option (also known as a barrier option).
%----------------------------------------------------------------------
  % We assume that the asset value is strictly non-negative when
  % searching for its implied value. Make the (not too bold) assumption
  % that asset value is no more than 75 times the strike price, when set
  % a test range. Can't evaluate at zero, because NaN errors occur.
  % Note that we can also use the knowledge that for our implied pricing,
  % the firm obviously hasn't gone broke, so asset value guess should be
  % above the strike price. We set equity strike to a minimum of 3/4 of
  % total liabilities, just for the sake of being accomodating.
  range    = [K*0.75, K*75];

  % Specify some optimisation options:
  options        = optimset('fzero');
  options.TolFun    = 10^-5;
  options.MaxIter    = 1*200;
  options.MaxFunEvals  = 1*200;
  options.Display    = 'notify';  % Notify only if it did NOT converge.
%   options.Display    = 'iter';
  options.FunValCheck  = 'on';



  % Search for the implied value of the asset, based on the current value
  % of the option and its defining parameters:
  ImpliedValue = fzero(@(V) ZeroDOCCall(V,K,r0,S,sigma,tau),...
      range,options);
  %%% End logic to find optimal bond price

  %%% Begin private methods
  %----------------------------------------------------------------------
  % @description: Implied asset value objective function.
  %        The objective function simply calculates the difference
  %        between observed market value, or price, of the option
  %        and the theoretical value derived from the
  %        Down-and-Out call option model with some estimated
  %        asset value...
  %        We aim to minimise the difference!
  %-----------------------------------------------------------------------
  function zero = ZeroDOCCall(V,K,r0,S,sigma,tau)
    % In the event of default, we assume equity holders receive zero
    % rebate:
    R    = 0;

    % When calculating implied asset values, we take the default
    % boundary to equal the total liabilities. Declare this for
    % explicitness!
    H    = K;

    zero = S - DownOutCall(H,r0,R,sigma,tau,V,K);
  end
end

function [lsUDBP] = UnitDiscBondLongSchwartz(tau,V,K,vParams,rr,rho,sigma)
%-----------------------------------------------------------------------
% @description:  UnitDiscBondLongSchwartz
%        Longstaff and Schwartz 1995 model of Risky Discount Bond
%        Pricing.
%        Calculate the price of a risky discount bond according
%        to the LS model as outlined in LS's 1996 paper, 'A Simple
%        Approach to Valuing Risky Fixed and Floating Rate Debt,
%        The Journal of Finance, 1996.
%        Bond is priced as though the face value of the bond at
%        maturity = 1.
% @notes:    Assumes the short-term risk-free interest rate r dynamics
%        are described by the alternate definition:
%        [dr = (alpha - beta*r)*dt + eta*dZ]
%        We still pass in the values as kappa and theta, and do the
%        equivalent calculations internally.
% @params:
%  tau      - Time until maturity. (i.e. bond has tau=T-t life
%        remaining).
%  V       - Firm asset value.
%  K       - Bankruptcy threshold. Financial distress occurs if V falls
%        below K before the bond matures.
%  vParams    - Structure containing vasicek interest rate parameters.
%  rr      - Recovery rate applied in the event of default. Multiplied
%        by the face-value payment to get refund received by
%        debt-holder.
%  rho      - Correlation between asset value and short-term risk free
%        interest rate r.
%  sigma    - Volatility (as std. deviation) of asset process. Constant
%        throughout time.
%-------------------------------------------------------------------------
  % For now, we set the number of integration slivers to be a constant
  % regardless of the time til maturity. Possibly this should be a factor
  % of maturity time, for example: [n = tau*100] ???
  n    = 200;
  % Calculate our liability ratio X
  X    = V/K;

  % Get our interest rate parameters in the variation required for the
  % particular format of the Vasicek model as used in LS1996
```

```
alpha  = vParams.kappa*vParams.theta;
beta  = vParams.kappa;

% The price of a riskless discount 0-coupon bond of equivalent maturity
% according to Vasicek
rfDBP = UnitDiscBondVasicek(tau,vParams);

% The default probability under the risk-neutral measure
defProbQ = Q_Sum(X,tau,n,vParams.r0,alpha,rho,sigma,vParams.eta,beta);

% Bond price as predicted by LS
lsUDBP = rfDBP*(1 - (1-rr)*defProbQ);

%-----------------------------------------------------------------------
% @note:The original LS formula calculates each element Q_i as a
%    recursive function. I do not know what sort of processing power
%    they were throwing at this problem to have had a useable solution
%    in 1997, but with my relatively new machine in 2008, using a
%    solution coded exactly as per their equation using n=20 took
%    upwards of 14 seconds for a single discount bond, and n > 20 became
%    ridiculous. As a result, I instead calculate each value in a more
%    linear method, and store it for future use. This makes my
%    code with n=200 almost instanteous, while values of n out
%    to about 800 still only take a couple of seconds.
%    Note also that I DO observe some slight changes in the
%    predicted bond price using n < 500 or so, in the order of
%    1/10,000th. Eg, predicted bond price might move from 0.5411 to
%    0.5410 as n goes from 200 to 400 for example.
%-----------------------------------------------------------------------
function q_Sum = Q_Sum(X,tau,n,r,alpha,rho,sigma,eta,beta)
  q_Sum = 0;    %Probability of default under Q-Measure
```

```
% Realised I can do this HUNDREDS of times faster by not performing
% the calculations for each recursive iteration... so instead of
% n-factorial-factorial-etc calculations, we just do n calculations
% of M() and S() and retrieve them in the recursive loops whenever we
% need to!!! Oh Excitement!! Oh Fast Code!!
M_vals = zeros(1,n);
S_vals = zeros(1,n);
for i_n = 1 : 1 : n
  M_vals(i_n) = M((i_n*tau/n),tau,r,alpha,rho,sigma,eta,beta);
  S_vals(i_n) = S((i_n*tau/n),rho,sigma,eta,beta);
end

% Now, we prepopulate a grid of all the a_i and b_ij values, since
% these get used time and time again... Note: we store them after
% running them through the N() function!!!
%
NA_vals = zeros(1,n);
NB_vals = zeros(n,n);

for i_i = 1 : 1 : n
  NA_vals(i_i) = N(A_I(X,i_i,M_vals,S_vals));
  for i_j = 1 : 1 : n
    % Only need to know halpha of the matrix of values, since
    % calculations never request values where i <= j
    if i_i > i_j
      NB_vals(i_i,i_j) = N(B_IJ(i_i,i_j,M_vals,S_vals));
    end
  end
end
```

```
% Now, we prepopulate a vector of Q_i elements so that we do NOT have
% to recursively generate the same value literally hundreds of
% thousands of times! This step is the one which above all saves us.
% Earlier optimisations, while significant in their own right, pale
% in comparison to this final optimisation!

% Store all the Q_i elements, start them with value 0
Q_vals = zeros(1,n);

for i_i = 1 : 1 : n
  % start by loading each Q_i with the N(a_i) as stored in
  % NA_vals(i), see Equation 6 in LS1996.
  Q_vals(i_i) = NA_vals(i_i);

  if i_i > 1
    % Now, subtract all the lesser indexed Q_i * N() elements.
    % This is effectively performing the recursive functionality,
    % but at a zillionth of the processing cost!!

    % Total amount to subtract...
    toSub = 0;
    for i_x = 1 : 1 : (i_i-1)
      toSub = toSub + Q_vals(i_x)*NB_vals(i_i,i_x);
    end
    Q_vals(i_i) = Q_vals(i_i) - toSub;
  end
end

% Finally, to calculate the total probability of default, we merely
%  sum all the Q_i elements for i=1:n.
q_Sum = sum(Q_vals);
```

```
end

%----------------------------------------------------------------------
% @description:  Corresponds to the equation a_i in LS1995, Equation 6.
% @params:
%  X    - Ratio of V/K
%  i_i    - Index value to retrieve ith value from the prepopulated
%      collection of M() and S() calculations
%  M_vals  - Collection of prepopulated M() calculations
%  S_vals  - Collection of prepopulated S() calculations
%---------------------------------------------------------------------
function a_out = A_I(X,i_i,M_vals,S_vals)
  a_out = (- log(X) - M_vals(i_i)) / sqrt(S_vals(i_i));
end

%---------------------------------------------------------------------
% @description:  Corresponds to the equation b_ij in LS1995, Equation 6.
% @params:
%  i_i    - Index value to retrieve ith value from the prepopulated
%      collection of M() and S() calculations
%  i_j    - Index value to retrieve jth value from the prepopulated
%      collection of M() and S() calculations
%  M_vals  - Collection of prepopulated M() calculations
%  S_vals  - Collection of prepopulated S() calculations
%---------------------------------------------------------------------
function b_out = B_IJ(i_i,i_j,M_vals,S_vals)
  b_out = (M_vals(i_j) - M_vals(i_i)) / sqrt(S_vals(i_i) - S_vals(i_j));
end

%----------------------------------------------------------------------
% @description:  Corresponds to equation M(t,tau) in LS1995, Equation 6.
```

```
%----------------------------------------------------------------------
function m_out = M(t,tau,r,alpha,rho,sigma,eta,beta)
  m_out = t*((alpha-rho*sigma*eta)/beta - eta^2/beta^2 - sigma^2/2) ...
      + exp(-beta*tau)*(exp(beta*t)-1)*(rho*sigma*eta/beta^2 + ...
      eta^2/(2*beta^3)) + (1-exp(-beta*t))*(r/beta - alpha/beta^2 + ...
      eta^2/beta^3) -(1-exp(-beta*t))*exp(-beta*tau)*(eta^2/(2*beta^3));
end


%-----------------------------------------------------------------------
% @description:  Corresponds to the equation S(t) in LS1995, Equation 6.
% @alert:    The Li and Wong 2006/07 equations differ to those of LS's
%         original, in that they multiple some fractions in this
%         equation by a factor of two. I have no idea why, only that
%         it caused me much grief trying to find out why my model
%         predictions did not match those from an Excel spreadsheet
%         with the LS logic implemented as a demonstration!!
%         As a result, I returned to original LS paper for the 'real'
%         version to implement.
%-----------------------------------------------------------------------
function s_out = S(t,rho,sigma,eta,beta)
%  LI AND WONG 2008 HAVE EXTRA MULTIPLIERS OF 2 -- _WHY_??!!
%   s_out = t*(2*rho*sigma*eta/beta + eta^2/beta^2 + sigma^2) - ...
%     (1-exp(-beta*t))*(2*rho*sigma*eta/beta^2 + 2*eta^2/beta^3) + ...
%     (1-exp(-2*beta*t))*(eta^2/(2*beta^3));

  s_out = t*(rho*sigma*eta/beta + eta^2/beta^2 + sigma^2) - ...
    (1-exp(-beta*t))*(rho*sigma*eta/beta^2 + 2*eta^2/beta^3) + ...
    (1-exp(-2*beta*t))*(eta^2/(2*beta^3));
end
end
```

```
function out = UnitDiscBondMerton(tau,V,K,vParams,sigma,rr)
%-------------------------------------------------------------------------
% @description:  Merton1974 model for risky zero-coupon bond prices.
%        Calculate the price of a risky discount bond according
%        to the extended Merton model as outlined in EHH2004
%        paper.
%        Assumes that asset value fluctuates (under a risk-neutral
%        measure) according to:
%        [dV = (r-delta)Vt*dt + sigma*Vt*dZ]
%
%        NOTE: Assumes that the face value of the bond is equal
%        to 1. Thus, to ensure that the pricing calculations are
%        correct, value of assets MUST be passed in as a ratio of
%        the unit face value. For example, for a company with
%        $120 of assets issuing a $5 bond, we should input V=120/5!!
% @params:
%  tau      - Time until bond matures. (i.e. bond has tau=T-t life
%        remaining).
%  V       - Firm asset value. (Expressed as a ratio compared to the
%        face-value of the bond, where we scale FV to equal a unit
%        bond.
%  K       - Default boundary, K in [0,1] (expressed as a percentage of
%        the face value of the bond.
%  vParams    - Structure containing vasicek interest rate parameters.
%  sigma    - Volatility (as std. deviation) of asset process. Constant
%        throughout time.
%  rr      -  Recovery rate after default, eg: (rr*ndp = 0.5131*1 = 0.5131)
%        suggests that 51% of face value is recovered after costly
%        default. It is implied that rr <= K always - the validity of
%        this statement been debated.
%-------------------------------------------------------------------------
```

```
% Perform data cleaning. For example, many parameters cannot be set to
% be exactly zero lest we get Divide-By-Zero errors, so before we begin
% any calculations, clean those values now:
kappa    = ZeroClean(vParams.kappa);
rr      = ZeroClean(rr);
sigma    = ZeroClean(sigma);

% Payout ratio: weighted average of c and the share repurchase-adjusted
% divident yield. Always set to 0, since should be incorporated by use
% of adjusted close prices etc.
delta    = 0;

% The price of a riskless discount 0-coupon bond of equivalent maturity
% according to Vasicek
 rfBP = UnitDiscBondVasicek(tau,vParams);

% Important idea behind Merton model: Calculate the weighted average of
% the value of the payment received if the bond defaults multiplied by
% the probabilty that it defaults, and sum that with the payout
% received if the bond does NOT default multiplied by the probability
% that it does NOT default. We do this now, in two parts:

% Part A: NO DEFAULT OCCURS
% Calculate the probability that no default occurs:
noDefProb = N(d2(K,tau,V,rfBP,delta,sigma));
% Calculate expected payout contributed by probability of no default
weightedNoDefPayout = 1 * noDefProb;

% Part B: DEFAULT OCCURS AND IS COSTLY
% Calculate the expected payout contributed by probability that
% costly default occurs:
  % (See EHH equation 6 for reference)
  weightedDefPayout = V/rfBP *exp(-delta*tau) * ...
    N(-d1(rr,tau,V,rfBP,delta,sigma)) + ...
    rr * (N(d2(rr,tau,V,rfBP,delta,sigma)) - ...
    N(d2(K,tau,V,rfBP,delta,sigma)));

  % Part C: Finally, return the price for the risky Merton discount bond
  out = rfBP * (weightedNoDefPayout + weightedDefPayout);

  %-----------------------------------------------------------------------
  % @description:  See EHH2004 Equation 7 for original definition.
  %-----------------------------------------------------------------------
  function out = d1(x,T,V,rfdb,delta,sigma)
    out = (log(V/(x*rfdb)) + (-delta+sigma^2/2)*T) / sigma*sqrt(T);
  end

  %-----------------------------------------------------------------------
  % @description:  See EHH2004 Equation 7 for original definition.
  %-----------------------------------------------------------------------
  function out = d2(x,T,V,rfdb,delta,sigma)
    out = d1(x,T,V,rfdb,delta,sigma) - sigma*sqrt(T);
  end
end

function bondPrice = UnitDiscBondVasicek(taus,params)
%-------------------------------------------------------------------------
% @description:  Vasicek 1977 model for Default-Free (Riskless) Zero-Coupon
%       Bond Pricing.
%       Calculates the price of a riskless zero-coupon bond
%       according to the model of Vasicek (1977), using the
%       particular formula specified in Hulls 'Options, Futures and
%        Other Derivatives', for a practical definition of the formula
%        to implement.
% @notes:    Considers the short-term risk-free interest rate dynamics
%        (in the risk-neutral measure!!)
%        to be described by the following stochastic equation:
%        [dr = kappa*(theta - r0)*dt + eta*dZ].
%        See Hull's Options Futures and Other Derivatives v6
%        Equation 23.6
%        Some bond pricing formulas use the alternative notation:
%        [dr = (alpha - beta*r0)*dt + eta*dZ]
%        We must be aware of these instances and work around them when
%        calculating our prices!! Transformation is:
%        alpha  = kappa*theta
%        beta  = kappa
% @params:
%  taus    - Matrix of times to maturity for which we are calculating the
%        yield-to-maturity under Vasicek model. Must be a 1*n
%        or an n*1 matrix... m*n will cause irregular results.
%  params    - Structure containing the properties required to calculate
%        yield-to-maturity. "params" has the following properties:
%        "params.r0": Instantaneous risk-free interest rate as observed
%        at time t. Typically used as the risk-free instantaneous
%        short-term rate.
%        "params.kappa": 'Pull-back' factor. Rate of mean reversion
%        (rate at which r0 is pulled back to long-term mean theta).
%        Constant through time.
%        "params.eta": Volatility of r0 (as std. deviation).
%        "params.theta": Constant. Long term average value of r0.
% @example:
%        taus = [0 : 0.1 : 10];
%        params.r0    = 0.09;
%         params.eta    = 0.03;
%         params.theta  = 0.06;
%         params.kappa  = 0.2;
%        bondPrices    = UnitDiscBondVasicek(taus,params);
% @author:    Dale Holborow, daleholborow@hotmail.com, Aug 7th, 2008
%--------------------------------------------------------------------------
  % Perform data cleaning. For example, many parameters cannot be set to
  % be exactly zero lest we get Divide-By-Zero errors, so before we begin
  % any calculations, clean those values now:
  params.kappa = ZeroClean(params.kappa);

  vasB = VasicekB();
  vasA = VasicekA(vasB);
  bondPrice = vasA.*exp(-params.r0*vasB);
  %%% End bond pricing logic  %%%

  %%% Begin private methods %%%
  %------------------------------------------------------------------------
  % @notes:    See Hull's Options Futures and Other Derivatives v6
  %        Equation 23.8
  %------------------------------------------------------------------------
  function a = VasicekA(vasB)
    a = exp(((vasB - taus).*(params.kappa^2*params.theta - ...
        0.5*params.eta^2)/(params.kappa^2)) - ...
        (params.eta^2*vasB.^2/(4*params.kappa)));
  end

  %------------------------------------------------------------------------
  % @notes:    See Hull's Options Futures and Other Derivatives v6
  %        Equation 23.7
  %------------------------------------------------------------------------
  function b = VasicekB()

    b = (1-exp(-params.kappa*taus))/params.kappa;

  end

end
```

Appendix G

Matlab code for maximum likelihood estimation of asset parameters, library functions and all additional logic

```
function [ActualBondPrice ActualYield] = CalcActualPriceAndYield(...
  actualPriceDtNum,firm)
%----------------------------------------------------------------------
% @description:  CalcActualPriceAndYield
%        Calculate the actual price/yields for a particular bond
%        based on market observations of its price on a particular
%        date. That way, we can calculate the implied yield to
%        maturity also.
%----------------------------------------------------------------------
  % Retrieve the actual observed bond price and the implied
  % yield

  % Since for testing purposes, all
  % bonds are priced as though they have a unit payoff, and our close
  % closes are stored similarly, we set the face value of the bond to
  % equal 1 when calculating yields.
  faceVal        = 1;
  % Calculate the remaining coupon payment times, so we can solve for
  % yield
  remMrktTaus  = CalcRemainingCouponTausInYrs(actualPriceDtNum,...
    firm.Bond.CouponDateNums);

  % Calculate actual bond price and subsequent yield to compare with
  % our above predictions.
  bondPriceObs    = get(firm.Bond.Prices, actualPriceDtNum);
  ActualYield      = CalcImpliedYield(...
    faceVal,...
    firm.Bond.CouponRate,...
    remMrktTaus,...
    bondPriceObs.ClosePrice);
  ActualBondPrice = bondPriceObs.ClosePrice;
end

function [drift volatility] = CalcActualYrlyDriftAndVolByObservations(...
  assetObs)
  numDailyObsForYr  = length(assetObs);
  dailyAssValsLog    = log(assetObs);
  dailyAssValsLogDiff  = diff(dailyAssValsLog);
  drift        = mean(dailyAssValsLogDiff)*numDailyObsForYr;
  volatility      = std(dailyAssValsLogDiff)*sqrt(numDailyObsForYr);
end

function [ImplAssetVals] = CalcCalendarYearImpliedAssetValues(...
  estimMode,vasParams,firm,estimYr,mu,sigma)

  paths    = PathInfo();
  const    = Constants();
```

```
% For a given parameter set, and a collection of observed
% market prices of equity, calculate the implied asset value at
% each point in time, treating the end of the year being estimated as
% time of maturity of the equity-as-an-option.

% We MUST loop through and calculate each implied asset value for each
% daily observation individually, because the fzero optimisation can
% only  receive scalar values as input!

% Establish the start and end dates of the year we intend to analyse
estimYrStartNum  = datenum(['01/01/' num2str(estimYr)], ...
  const.DateStringAU);
estimYrEndNum  = datenum(['31/12/' num2str(estimYr)],...
  const.DateStringAU);

% To perform asset estimation, we need to know how many shares there
% are, and also the total liabilities. For these, we need to turn to
% the end-of-year report from the year PRIOR to our estimation year:
priorYr      = estimYr-1;
priorYrFinObs  = get(firm.Financials, priorYr);

% Store all the calculated values in a hashtable. That way we never
% have to worry about remembering whether values are indexed in chrono
% order etc, as we would if we used arrays
ImplAssetVals = hashtable;

% MUST calculate maturities IN YEARS, because all our volatility and
% drift params are specified on a yearly basis!! Decide how many
% days are in this year.
daysInEstimYear = DaysInYear(estimYr);
```

```
% Loop through each day within the year period specified, and based on
% the asset dynamics parameters (mu and sigma) specified, calculate
% the implied asset values under the Merton model:

% Don’t estimate the assets on the final day, since the time to expiry
% of the equity-as-an-option is zero and thus worthless.
for dayInd = estimYrStartNum : 1 : (estimYrEndNum-1)

  % Only estimate the asset values if there are share
  % price and interest rate observations:
  if has_key(firm.Equity, dayInd) && has_key(vasParams, dayInd)
    % Time to maturity (i.e. time till end of year)
    tau      = (estimYrEndNum - dayInd)/daysInEstimYear;

    % Get interest rate params estimated based on daily term
    % structure
    dailyVasParams    = get(vasParams,dayInd);

    % Get the share price observation data for that day
    dailyEqtyObs    = get(firm.Equity, dayInd);
    equityVal      = dailyEqtyObs.AdjClose * ...
      priorYrFinObs.OutStShares;

    % Calculate implied asset values according to merton model
    if strcmp(const.ModeMerton, estimMode)
      dailyImplAssetVal  = ImpliedAssetValueBlackScholes(...
        priorYrFinObs.TotLiab, ...
        mu, ...
        dailyVasParams.r0, ...
        sigma, ...
```

```
          equityVal, ...
          tau);
      % Calculate implied asset values according to L&S model
      elseif strcmp(const.ModeLS, estimMode)
        dailyImplAssetVal  = ImpliedAssetValueDownAndOutCall(...
          priorYrFinObs.TotLiab, ...
          mu, ...
          dailyVasParams.r0, ...
          sigma, ...
          equityVal, ...
          tau);
      else
        error(['Died: Unknown asset estimation method detected']);
      end
      % Add the implied asset valuation into a hash, ready to be
      % returned
      ImplAssetVals = put(ImplAssetVals, dayInd, dailyImplAssetVal);
    end
  end
end

function [actualPriceDtNum] = CalcClosestPossibleValuationDate(firm, ...
    vasParams, tryToPriceDtNum, increment, maxDist)
%-----------------------------------------------------------------------
% @description:  CalcClosestPossibleValuationDate
%  We wish to price the bond as early as possible on or after its date of
%  issue. To test whether pricing is possible for a particular date, we
%  test that we have:
%  a) Observed equity prices for that date
%  b) Calculated instantaneous interest rate (and additional params) on
%  that date
%  c) An observed bond price on that date
%  d) Asset dynamics parameter estimates for the financial year prior
%  to pricing:
% @params:
%  firm    -
%  vasParams  -
%  tryToPriceDtNum- Initial target date that we want to try to perform
%    some pricing functionality. Search around this date for the
%    first day where we have all the necessary interest rate,
%    bond price, equity price etc, data, to be able to perform
%    pricing logic.
%  increment  - The number of days to move when trying to search for
%    dates. Typically, a date will be specified which we will
%    try to price, and if we can't price on that intented date
%    because of a lack of data, increment=+1 means we will try
%    the FOLLOWING day. increment=-1 means try the PREVIOUS day.
%-----------------------------------------------------------------------
  found    = false;
  tryDtNum  = tryToPriceDtNum;

  % To perform asset estimation, we need to know how many shares there
  % are, and also the total liabilities. For these, we need to turn to
  % the end-of-year report from the year PRIOR to our estimation year:
  prevYr      = year(tryToPriceDtNum)-1;

  while ~found & abs(tryDtNum-tryToPriceDtNum) < maxDist
    if has_key(vasParams, tryDtNum) & ...
        has_key(firm.Equity, tryDtNum) & ...
        has_key(firm.Bond.Prices, tryDtNum) & ...
        has_key(firm.Financials, prevYr)
```

```
      % The date being tested has all the information we need to
      % perform a bond price test, so use that date.
      actualPriceDtNum = tryDtNum;
      found = true;
    end
    tryDtNum = tryDtNum+increment;
  end

  if ~found
    error(['No suitable pricing date within range of : '...
      datestr(tryToPriceDtNum)]);
  end
end

function [mrktTaus] = CalcCouponTausInYears(startDtNum, couponDtNums)
%-------------------------------------------------------------------------
% @description:  CalcCouponTausInYears
%        Calculate the time remaining until maturity for each coupon
%        in the bond. The face value will also occur at the longest
%        maturity, but is not stored as an additional element in the
%        vector.
%-------------------------------------------------------------------------
  [mrktTaus] = [couponDtNums - startDtNum] / 365;
end

function [discBondYields] = CalcDiscountBondYield(taus,bondPrices)
%-------------------------------------------------------------------------
% @description:  Calculate the yield-to-maturity of zero-coupon discount
%        bond(s). Pricing coupon bonds is a more complex procedure.
%        Can price yield SPREADS by calling function along the lines
%        of:
```

```
%        yieldSpread = DiscBondYield(taus,(riskyBP/riskFreeBP)).
%        This allows us to derive the term structure of default
%        spreads. Refer to Briys
%        and de Varenne's 1997 paper, Section 4, Equation 12.
%        Calculates the difference between the yield of a risky
%        corporate bond and a risk-free bond of equivalent maturity.
%        This calculation is only valid for zero-coupon discount
%        bonds, the presence of coupons means yield calculations are
%        more complex.
% @params:
%  taus    - Matrix of times to maturity for which we are calculating
%        the yield-to-maturity under some interest rate model.
%        Must be a 1*n or an n*1 matrix... m*n will cause irregular
%        results.
%  bondPrices  - The n*1 or 1*n matrix of bond prices of zero-coupon bonds
%        for which we determining the yield-to-maturity.
% @example:
%        taus = [0 : 0.1 : 10];
%        params.r    = 0.09;
%         params.nu    = 0.03;
%         params.theta  = 0.06;
%         params.kappa  = 0.2;
%        bondPrices    = VasicekUnitDiscBond(taus,params);
%        yields      = DiscBondYield(taus,bondPrices)
% @author:
%-------------------------------------------------------------------------
  % Perform data cleaning. For example, many parameters cannot be set to
  % be exactly zero lest we get Divide-By-Zero errors, so before we begin
  % any calculations, clean those values now:
  bondPrices  = ZeroClean(bondPrices);
  taus    = ZeroClean(taus);
  % Note: As per Hull, Chapter 23, P538, if R(t,T) is the continuously
  % compounded interest rate at time t for a term T-t, (i.e. R(t,T) is
  % the yield-to-maturity, NOT the instantaneous rate) then:
  % R(t,T) = - (1 / (T-t)) * log(BondPrice(t,T)).
  discBondYields = (-taus.^(-1)).*log(bondPrices);
end

function [ImpliedYield] = CalcImpliedYield(faceVal,coupon,taus,bondPrice)
%--------------------------------------------------------------------------
% @description:  CalculateImpliedYield
%        Based on some observed or predicted bond price, and values
%        for the remaining coupon and face value payment rates and
%        maturities, calculate the implied bond yield which would
%        produce that bond price.
% @example:
%        coupon  = 0.05;
%        taus  = [1/2 : 1/2 : 10];
%        faceVal  = 100;
%        bondPrice  = 97;
%        implYield = CalcImpliedYield(faceVal,coupon,taus,bondPrice)
%--------------------------------------------------------------------------
  % Min and max boundaries of our search, expect realistic values to be
  % between 0 and 0.15...
  minYield  = 0;
  maxYield  = 0.5;
  ImpliedYield = fminbnd(@(yieldGuess) SeekForYield(yieldGuess,coupon,...
    taus,bondPrice,faceVal),minYield,maxYield);

  %------------------------------------------------------------------------
  % @description:  SeekForYield
  %        Method to use to search for optimum implied yield given
  %        an observed bond price, coupon rate, and maturity
  %        dates.
  %----------------------------------------------------------------------
  function [yieldError] = SeekForYield(yieldGuess,coupon,...
      taus,targetPrice,faceVal)

    % Calculate implied present value of all coupons using current yield
    % guess:
    implPresVal = 0.5 * coupon * faceVal * exp(-yieldGuess*taus);

    % Calculate also the present value of the fave value payment using the
    % current yield guess:
    matTau    = taus(end);
    implPresVal(end+1) = faceVal * exp(-yieldGuess*matTau);

    % Implied price given a particular yield is the sum of the present
    % value of all coupons and the face value payment.
    implPrice  = sum(implPresVal);

    % Calculate the difference between the implied price and the observed
    % price, to find the optimal implied yield.
    yieldError = abs(implPrice - targetPrice);
  end
end

function [remMrktTaus] = CalcRemainingCouponTausInYrs(startDtNum,...
  couponDtNums)
  mrktTaus = CalcCouponTausInYears(startDtNum, couponDtNums);

  % Return only coupon dates which are still outstanding after the
  % observation start date at which time we are trying to price the
  % bond
  remMrktTaus  = mrktTaus(find(mrktTaus > 0));
end


function CalculateAndStorePureProxyDynamics()
%----------------------------------------
% Patch, to fix bug in the calculation of M and LS rho values, and to also
% calculate Pure Proxy mean, std and rho values, which wasnt done earlier.
%----------------------------------------
  const    = Constants();
  paths    = PathInfo();
  vasParams    = ParseInterestRateParamsVasicek();
  firms = ParseCompanyList();
  for firm_i = 1 : 1 : length(firms)
    tmpFirm  = firms(firm_i);
    load([paths.PreCalcFirmHistory tmpFirm.Bond.DSBondCode], 'firm');
    clear tmpFirm;
    disp(['Begin processing firm ' firm.CompName]);

    [yrsKeys yrsVals] = dump(firm.Assets.MertonAssetParams);
    yrsKeys  = cell2mat(yrsKeys)

    % Create somewhere to store pure proxy info:
    firm.Assets.PureProxyAssetParams = hashtable;

    for estimYr = yrsKeys(1) : 1 : yrsKeys(end)
      prevYr    =  estimYr-1;
      yrlyParamsM = get(firm.Assets.MertonAssetParams, estimYr);
      yrlyParamsLS= get(firm.Assets.LSAssetParams, estimYr);
```

```
estimMode    = const.ModeMerton;
implAssetValsM  = CalcCalendarYearImpliedAssetValues(estimMode, ...
  vasParams,firm,estimYr,yrlyParamsM.mu,yrlyParamsM.sigma);
estimMode    = const.ModeLS;
implAssetValsLS  = CalcCalendarYearImpliedAssetValues(estimMode, ...
  vasParams,firm,estimYr,yrlyParamsLS.mu,yrlyParamsLS.sigma);

% Our num shares outstanding and total liabs must come from the
% previous year end of year balance sheet
prevYrFinObs  = get(firm.Financials, prevYr);
yrStartNum    = datenum(['01/01/' num2str(estimYr)], ...
  const.DateStringAU);
yrEndNum    = datenum(['31/12/' num2str(estimYr)], ...
  const.DateStringAU);

% Calculate the pure proxy asset values, and hence the yearly
% drift, volatility and correlation to interest rate changes.
dailyVasR0s    = [];
dailyAssValsPP  = [];
dailyAssValsM  = [];
dailyAssValsLS  = [];
for dayInd = yrStartNum : 1 : yrEndNum

  if has_key(vasParams, dayInd) & has_key(firm.Equity, dayInd) & ...
    has_key(implAssetValsM, dayInd) & has_key(implAssetValsLS, ...
    dayInd)

    % store the instantaneous interest rate
    dailyVasObs      = get(vasParams, dayInd);
    dailyVasR0s(end+1)  = dailyVasObs.r0;
    % Store the relevant asset estimation in matching index
    dailyEqtyObs      = get(firm.Equity, dayInd);
    dailyEqtyVal  = dailyEqtyObs.AdjClose*prevYrFinObs.OutStShares;
    dailyAssValsPP(end+1)  = dailyEqtyVal + prevYrFinObs.TotLiab;

    dailyAssValsM(end+1)  = get(implAssetValsM, dayInd);
    dailyAssValsLS(end+1)  = get(implAssetValsLS, dayInd);
  end
end

% Now that we have found all the relevant asset and interest
% rate values, calculate their drift, volatility and
% correlation:
dailyAssValsPPLog      = log(dailyAssValsPP);
dailyAssValsPPLogDiff    = diff(dailyAssValsPPLog);

[yrlyParamsPP.mu yrlyParamsPP.sigma] = ...
  CalcActualYrlyDriftAndVolByObservations(dailyAssValsPP)
yrlyParamsPP.rho  = corr2(dailyAssValsPP, dailyVasR0s);

% Now store it all back into the object, ready for saving
yrlyParamsM.rho    = corr2(dailyAssValsM, dailyVasR0s);
yrlyParamsLS.rho  = corr2(dailyAssValsLS, dailyVasR0s);

% Store all or calced values
firm.Assets.PureProxyAssetParams = put(...
  firm.Assets.PureProxyAssetParams, estimYr, yrlyParamsPP);
firm.Assets.MertonAssetParams  = put(...
  firm.Assets.MertonAssetParams, estimYr, yrlyParamsM);
firm.Assets.LSAssetParams    = put(...
  firm.Assets.LSAssetParams, estimYr, yrlyParamsLS);
```

```
    end


    % Finally, save the individual firm objects as matlab binary files
    % so they can be retrieved very quickly in future.
    save([paths.PreCalcFirmHistory firm.Bond.DSBondCode], 'firm');
  end
end


function [CouponDates] = CalculateCouponDates(...
  issDateNum, matDateNum, couponDateStr)
%----------------------------------------------------------------------
% @description:  Based on a string containing two embedded coupon date
%        day/month pairs, parse that string and extract the
%        values so that we can calculate on which days the
%        coupons land. Store a collection of all the dates that
%        the bond will pay coupons.
%----------------------------------------------------------------------
  paths    = PathInfo();
  const    = Constants();
  issDateVec    = datevec(issDateNum);
  matDateVec    = datevec(matDateNum);

  % Store all coupons calculated between the issue and maturity date
  % range.
  CouponDates    = [];

  coup2MStr    = couponDateStr(1,end-1:end);
  coup2DStr    = couponDateStr(1,end-3:end-2);
  coup1MStr    = couponDateStr(1,end-5:end-4);
  % Some date strings are passed in as a 7 digit string instead of 8
  if 8 == length(couponDateStr)
```

```
    spacer = 2;
  else
    spacer = 1;
  end
  coup1DStr    = couponDateStr(1,(1:spacer));

  % For each year in the range of the bond life, calculate a first
  % and second coupon date, and if they are within the issue and
  % maturity date, add them to the collection.
  for yrInd = issDateVec(1) : 1 : matDateVec(1)
    yrStr  = num2str(yrInd);
    coupon1Num  = datenum(datestr([coup1DStr '/' coup1MStr '/' yrStr],...
      const.DateStringAU));
    coupon2Num  = datenum(datestr([coup2DStr '/' coup2MStr '/' yrStr],...
      const.DateStringAU));

    % If the calculated coupon dates are valid, add them
    if coupon1Num > issDateNum && coupon1Num <= matDateNum
      CouponDates(end+1) = coupon1Num;
    end
    if coupon2Num > issDateNum && coupon2Num <= matDateNum
      CouponDates(end+1) = coupon2Num;
    end
  end
end

function EstimateAssetDynamics()
  paths    = PathInfo();
  const    = Constants();
  vasParams  = ParseInterestRateParamsVasicek();
  firms = ParseCompanyList();
```

```
for firm_i = 1 : 1 : length(firms)
  firm = firms(firm_i);

  % Load up as much historical bond price information as possible.
  firm.Bond.Prices = ParseBondPricesByDSBondCode(firm.Bond.DSBondCode);

  % Load up historical equity information (adjusted closing prices) so
  % we can calculate the implied asset dynamics
  firm.Equity    = ParseEquityByDSBondCode(firm.Bond.DSBondCode);

  % Load up historical financial statement information about the company
  firm.Financials  = ParseFinancialsByDSBondCode(firm.Bond.DSBondCode);

  % Preallocate firm assets params storage space for each pricing model
  firm.Assets.MertonAssetParams  = hashtable;
  firm.Assets.LSAssetParams    = hashtable;

  % Based on the year in which we see a bond issued, estimate the asset
  % dynamics for the previous year.
  estimYr      = year(firm.Bond.IssueDateNum)-1;
  for yrInd = estimYr : 1 : 2007
    try
      % For each year of interest, and each bond pricing model,calculate
      % and store the implied asset parameters.
      % Store the results of the different methods of parameter estim,
      % keyed on the year for which they are applicable.

      % First, estimate the yearly asset dynamics using the Merton model
      estimMode    = const.ModeMerton;
      yrlyParamsMert  = EstimateAssetDynamicsByFirmAndYear(...
        estimMode,vasParams,firm,yrInd);
        firm.Assets.MertonAssetParams = put(...
          firm.Assets.MertonAssetParams, yrInd, yrlyParamsMert);
        disp(['Calculated Merton MLE of ' num2str(yrInd) ...
          ' asset params-> mu:' num2str(yrlyParamsMert.mu) ...
          ', sigma: ' num2str(yrlyParamsMert.sigma)]);

        % Second, estimate the yearly asset dynamics using the LS model
        estimMode    = const.ModeLS;
        yrlyParamsLS  = EstimateAssetDynamicsByFirmAndYear(...
          estimMode,vasParams,firm,yrInd);
        firm.Assets.LSAssetParams = put(firm.Assets.LSAssetParams, ...
          yrInd, yrlyParamsLS);
        disp(['Calculated LS MLE of ' num2str(yrInd) ...
          ' asset params-> mu:' num2str(yrlyParamsLS.mu) ...
          ', sigma: ' num2str(yrlyParamsLS.sigma)]);
        disp(['Estimated asset values for firm: ' firm.CompName ...
          ' for year: ' num2str(yrInd)]);
      catch
        disp(['Tried to estimate ' firm.CompName ' for the year ' ...
          num2str(yrInd) ' but failed']);
      end
    end

    % Finally, save the individual firm objects as matlab binary files
    % so they can be retrieved very quickly in future.
    save([paths.PreCalcFirmHistory firm.Bond.DSBondCode], 'firm');
  end
  % Now patch our records by calculating the pure proxy, and also Merton
  % and LS correlation coefficients between assets and interest rates
  CalculateAndStorePureProxyDynamics();
```

```
%%% Begin logic for private methods
function logLikeSum = GenericLogLikelihood(...
    pars,estimMode,vasParams,firm,estimYr)

  % Sometimes the search algorithm passes in negative and/or zero values
  % for mu
  % and sigma, but of course, we can’t have a negative sigma value.
  % Catch this now by letting the optimiser know that negative sigmas
  % are BAD, we set the "minimal" value to something massive so that
  % it doesn’t continue processing, and also doesn’t appear to be a
  % valid path to continue with when searching for optimal
  % parameters.

  % Tiny sigma values fail in MLE models, because they result in
  % transition probabilities of zero, which in turn trigger infinite
  % values in our log-likelihood calculations. We have to manually
  % test for these, and to assist the optimiser search, we set a
  % minimum ’realistic’ volatility value of 0.01.
  sigmaMin  = 0.01;
  if pars(2) <= sigmaMin
    % Invalid sigma, exit early and let search algorithm know it
    % tried something naughty... set to artificially massive
    % positive number to tell our fMINsearch function to head the
    % other direction:
    disp([’Guessed small val of sig, turn around!:’ num2str(pars(2))]);
    logLikeSum = 99999999;
  else
    % Retrieve the guessed values of the drift and volatility, price
    % the implied asset value, and solve for the maximum of the
    % log-likelihood.
    muGuess      = pars(1);
    sigmaGuess    = pars(2);

    % For a specified pricing model, and a given parameter set, and a
    % collection of observed
    % market prices of equity, calculate the implied asset value at
    % each point in time, so we can then perform MLE to estimate
    % the driving asset dynamics:
    implAssetVals  = CalcCalendarYearImpliedAssetValues(estimMode, ...
      vasParams,firm,estimYr,muGuess,sigmaGuess);

    % Now calculate the loglikelihood of the MLE function for these
    % parameter estimates, so we can establish which param values are
    % most likely:
    logLikeSum = LogLikelihoodSummation(estimMode,implAssetVals,...
      firm,estimYr,vasParams,muGuess,sigmaGuess);

    % Since we want to MAXIMISE the log-likelihood function, but we
    % are using the FMINSEARCH function, we need to negate our
    % results which are coming back from the loglikelihood
    logLikeSum = -logLikeSum;
  end
end

function [yrlyParams] = EstimateAssetDynamicsByFirmAndYear(...
    estimMode,vasParams,firm,estimYr)

  % Guess some starting parameters for the asset price, to initialise
  % optimisation process. These aren't especially important but since we
  % have to declare the variables anyway, set them to something random
  % in the rough vacinity of 'real world' values.
  % For some stupid reason, randomising these values seems to cause
% problems, but hardcoded guesses dont? Even though our initial
% hard coded guesses are typically miles off anyway??
guessPars(1)      = 0.2;    % mu
guessPars(2)      = 0.4;    % sigma

% Configure search options to make sure that our optimal asset dynamic
% values are a good estimate. Increase the number of trials, just in
% case. With 250 observations over a 1 year time period, the number of
% calculations used is typically well below our limits, but have CPU
% to spare, so lets aim for EXTREME attempts to get estimate accuracy.
% We go for such extremely large estimation numbers because sometimes,
% the parameter estimates are problematic, they run thru maximum
% searches without finding good values, without much consistency to
% this performance. In an attempt to never encounter this problem, we
% get brutal....
options        = optimset('fminsearch');
options.TolFun    = 10^-5;
options.MaxIter    = 2*400;
options.MaxFunEvals  = 2*800;  % 2*700 wasn't solving all scenarios!
options.Display    = 'notify';  % Notify only if it did NOT converge.
options.FunValCheck  = 'on';

% Search for the optimum mu and sigma for the dynamics of the asset
% value evolution process, using whichever implied asset pricing mode
% was requested:
[results,fval,exitflag,output] = fminsearch(@(pars) ...
  GenericLogLikelihood(pars,estimMode,vasParams,firm,estimYr),...
  guessPars,options);

% Retrieve the estimates for the asset parameters and store them ready
% to be returned by the function.
```

```
    yrlyParams.mu      = results(1);
    yrlyParams.sigma    = results(2);
  end
end


function [LogLikelihoodSum] = LogLikelihoodSummation(estimMode,...
    implAssetVals,firm,estimYr,vasParams,mu,sigma,H)
  % Retrieve constants and path information
  const    = Constants();
  paths    = PathInfo();

  % Store each element required to calculate the total of the MLE
  % functions
  LogLikelihoodSums    = [];

  % Get start and end dates of the year for which we are estimating
  % parameters
  estimYrStartNum=datenum(['01/01/' num2str(estimYr)],const.DateStringAU);
  estimYrEndNum  =datenum(['31/12/' num2str(estimYr)], const.DateStringAU);

  % To perform asset estimation, we need to know how many shares there
  % are, and also the total liabilities. For these, we need to turn to
  % the end-of-year report from the year PRIOR to our estimation year:
  priorYr      = estimYr-1;
  priorYrFinObs  = get(firm.Financials, priorYr);
  defBoundary    = priorYrFinObs.TotLiab;

  % MUST calculate maturities IN YEARS, because all our volatility and
  % drift params are specified on a yearly basis!! Decide how many
  % days are in this year.
  daysInEstimYear = DaysInYear(estimYr);
```

```
% Retrieve all the asset valuations by examining the day on which they
% were made.
[assetDtKeys assetVals] = dump(implAssetVals);
assetDtKeys        = cell2mat(assetDtKeys);
assetVals         = cell2mat(assetVals);
% The log of all the observation dates, for the transitional density
assetValLogs       = log(assetVals);



% Loop through each day in the year, calculating transitional densities
% for the implied asset valuations.
for dayIndex = 2 : 1 : length(assetDtKeys)
  currDayNum    = assetDtKeys(dayIndex);
  prevDayNum    = assetDtKeys(dayIndex-1);
  tDiffInYrs    = (currDayNum-prevDayNum)/365;

  currDayVal    = assetVals(dayIndex);
  currDayLogVal  = assetValLogs(dayIndex);
  prevDayLogVal  = assetValLogs(dayIndex-1);

  % Time to maturity (i.e. time till end of year)
  tau          = (estimYrEndNum - currDayNum)/daysInEstimYear;

  % Get interest rate params estimated based on daily term
  % structure
  dailyVasParams    = get(vasParams,currDayNum);

  % Calc the loglikelihood values for each transitional date etc
  % For some parameter estimates, we get very small values, and
  % subsequently get LogOfZero warnings. Matlab seems to be
% capable of continuing on and getting valid results, so suppress
% the warning messages temporarily, since they just slow our
% calculations down.
warning off MATLAB:log:logOfZero;



% Log likelihood asset dynamics parameter estimation via Merton model:
if strcmp(const.ModeMerton,estimMode)
  % Transitional density of Merton model
  gIndex  = MertonG(mu,sigma,tDiffInYrs,currDayLogVal,prevDayLogVal);
  % Delta of Merton model
  md1  = MertonD1(defBoundary,dailyVasParams.r0,sigma,tau,currDayVal);
  % Store loglikelihood estimate for this transition
  optionDelta  = N(md1);

% Log likelihood asset dynamics parameter estimation via the LS model:
elseif strcmp(const.ModeLS,estimMode)
  bb      = BarrierB(defBoundary);
  bEta    = DownOutCallEta(mu, sigma);
  eqtyRebate  = 0;  % Equity holders receive nothing on default

  % Transitional density of LS model
  gIndex    = BarrierG(defBoundary,mu,sigma,tDiffInYrs,...
    currDayLogVal,prevDayLogVal,bb,bEta);
  % Delta of LS model, according to DOC option pricing
  % methodology
  optionDelta  = DownOutCallDelta(...
    defBoundary,...
    dailyVasParams.r0, ...
    eqtyRebate, ...
    sigma,...
```

```
        tau,...
        currDayVal,...
        defBoundary);
    else
      error(['Error : Undetected asset estimation method detected']);
    end
    % Have to perform some error checking, since sometimes the log of
    % zero causes errors because it returns infinite values etc. We
    % catch this scenario, and instead, just set the likelihood to
    % zero...is the best we can do to work around the situation.
    elementalSum = log(gIndex) - log(currDayVal*optionDelta);
    if isinf(elementalSum) || isnan(elementalSum)
      elementalSum = 0;
    end
    LogLikelihoodSums(end+1) = elementalSum;

    % Don't forget to reenable the warnings
    warning on MATLAB:log:logOfZero;
  end
  % Sum all the daily transitional values ready for return
  LogLikelihoodSum  = sum(LogLikelihoodSums);
  %%% End log-likelihood summation main logic

  %%% Begin private method logic
  %---------------------------------------------------------------------
  % @description:  Li&Wong2008 Section 2.4.2 Equation 3
  % @params:
  %  H      - default boundary
  %---------------------------------------------------------------------
  function [bg] = BarrierG(H,mu,sigma,tDiffInYrs,vi,viM1,bb,bnu)
    % By default, the transitional density value is zero, unless the
  % value of the underlying asset is larger than the barrier. In
  % that case we must explicitly calculate the density.
  bg = 0;
  if viM1 > log(H)
    bg = BarrierVarPhi(mu,sigma,tDiffInYrs,(vi-viM1)) - ...
      exp(2*(bnu-1)*(bb-viM1))*...
      BarrierVarPhi(mu,sigma,tDiffInYrs,(vi+viM1-2*bb));
  end
end



%-----------------------------------------------------------------------
% @description:  Li&Wong2008 Section 2.4.2 Equation 3
%-----------------------------------------------------------------------
function [bVarPhi] = BarrierVarPhi(mu,sigma,tDiffInYrs,x)
  bVarPhi = 1/(sigma*sqrt(2*pi*(tDiffInYrs))) * ...
    exp(-((x-((mu-0.5*sigma^2)*(tDiffInYrs)))^2)/...
    (2*sigma^2*(tDiffInYrs)));
end

%-----------------------------------------------------------------------
% @description:  Li&Wong2008 Section 2.4.2 Equation 3
%-----------------------------------------------------------------------
function [bb] = BarrierB(H)
  bb = log(H);
end

%-----------------------------------------------------------------------
% @description:  The density function of log(Vt) given under the
%         physical probability measure. See Li&Wong2008, Appendix
%         B for notation.
```

```
  %------------------------------------------------------------------------
  function [mg] = MertonG(mu,sigma,tDiffInYrs,vi,viM1)
    mg = 1/(sigma*sqrt(2*pi*(tDiffInYrs))) * ...
      exp(-((vi-viM1-((mu-0.5*sigma^2)*(tDiffInYrs)))^2)/...
      (2*sigma^2*(tDiffInYrs)));
  end



  %-------------------------------------------------------------------------
  % @description: Corresponds to LiWong2008 Appendix B definition of
  %         Black-Scholes standard call option model parameter d1
  % @params:
  %  K       - Book value of corporate liabilities
  %-------------------------------------------------------------------------
  function [md1] = MertonD1(K,r0,sigma,tau,V)
    md1 = (log(V/K) + (r0+0.5*sigma^2)*tau)/(sigma*sqrt(tau));
  end

end


function [PredictedBondPrice ImpliedYield] = ...
    PredictCouponBondPriceAndYield(estimMode,priceDtNum,...
    firm,vasParams,rr_c,rr_p)
%---------------------------------------------------------------------------
% @description:  Predicts bond as though it were issued with a unit face
%        value
%---------------------------------------------------------------------------
  const    = Constants();

  % Make sure we aren't trying to price a bond before it was issued, or
  % after it matures!!
```

```
if firm.Bond.IssueDateNum > priceDtNum || ...
    firm.Bond.MatDateNum <= priceDtNum
  error(['Processing halted: Cannot price bond outside its lifetime!']);
end

% Variables to store total price of coupons and price of face value
% payment
cPrices  = [];
fvPrice  = [];

% Get total liabilities as best we know them for the year when we are
% pricing the bond. Note: When we are pricing the bond on
% issue, we take the book value from financial
% statement of previous year and add the liability of the new bond,
% otherwise, bond is already absorbed into previous year liability
% value.
prevYr      = year(priceDtNum)-1;
prevYrFinObs  = get(firm.Financials, prevYr);

% MUST calculate maturities IN YEARS, because all our volatility and
% drift params are specified on a yearly basis!! Decide how many
% days are in this year that we are using to price the bond:
daysInEstimYear = DaysInYear(year(priceDtNum));

% Establish the start and end dates of the year we intend to analyse
estimYrStartNum  = datenum(['01/01/' num2str(year(priceDtNum))], ...
  const.DateStringAU);
estimYrEndNum  = datenum(['31/12/' num2str(year(priceDtNum))], ..
const.DateStringAU);

% Calculate the implied value of the assets on the date that we are
% pricing the bond. To do this, use the asset dynamics estimated as of
% end of last year, the share price observations, the interest rate
% observations, etc:
% Time to maturity (i.e. time till end of year)
tau          = (estimYrEndNum - priceDtNum)/daysInEstimYear;

% Get the params for the Vasicek interest rate on that same date as the
% asset observation
vParamsAtObsDt    = get(vasParams, priceDtNum);

% Get the share price observation data for that day
priceDtEqtyObs    = get(firm.Equity, priceDtNum);
priceDtEquityVal  = priceDtEqtyObs.AdjClose *  prevYrFinObs.OutStShares;

% Depending on which pricing model we wish to test, retrieve the
% appropriate asset parameters before performing the pricing now.
% Parameters of asset dynamics for the previous year before when we
% price the
% bond, since those params are as good and as close as we can get,
% given that we don't know the current year's parameters because the
% year has not finished!
% Calculate the implied asset value on the date that we are pricing our
% bond:
if strcmp(estimMode, const.ModeVasicek)

  % We are pricing a risk free bond, don't need to calculate any
  % implied asset valuations etc...
  priceDtImplAssetVal = NaN;

elseif strcmp(estimMode, const.ModePureProxyM)
  priceDtPrevYrAssetParams=get(firm.Assets.PureProxyAssetParams,prevYr);
```

```
    priceDtImplAssetVal  = ImpliedAssetValueBlackScholes(...
      prevYrFinObs.TotLiab, ...
      priceDtPrevYrAssetParams.mu, ...
      vParamsAtObsDt.r0, ...
      priceDtPrevYrAssetParams.sigma, ...
      priceDtEquityVal, ...
      tau);
  elseif strcmp(estimMode, const.ModeMerton)
    priceDtPrevYrAssetParams = get(firm.Assets.MertonAssetParams, prevYr);
    priceDtImplAssetVal  = ImpliedAssetValueBlackScholes(...
      prevYrFinObs.TotLiab, ...
      priceDtPrevYrAssetParams.mu, ...
      vParamsAtObsDt.r0, ...
      priceDtPrevYrAssetParams.sigma, ...
      priceDtEquityVal, ...
      tau);
  elseif strcmp(estimMode, const.ModePureProxyLS)
    priceDtPrevYrAssetParams=get(firm.Assets.PureProxyAssetParams,prevYr);
    priceDtImplAssetVal  = ImpliedAssetValueDownAndOutCall(...
      prevYrFinObs.TotLiab, ...
      priceDtPrevYrAssetParams.mu, ...
      vParamsAtObsDt.r0, ...
      priceDtPrevYrAssetParams.sigma, ...
      priceDtEquityVal, ...
      tau);
  elseif strcmp(estimMode, const.ModeLS)
    priceDtPrevYrAssetParams = get(firm.Assets.LSAssetParams, prevYr);
    priceDtImplAssetVal  = ImpliedAssetValueDownAndOutCall(...
      prevYrFinObs.TotLiab, ...
      priceDtPrevYrAssetParams.mu, ...
      vParamsAtObsDt.r0, ...
    priceDtPrevYrAssetParams.sigma, ...
    priceDtEquityVal, ...
    tau);
else
  die(['Died: Invalid pricing mode specified: ' estimMode]);
end

% Now, we MUST test to see if the bond pricing is occuring within the
% same year that the bond was issued. If this IS the case, then the
% face value of the bond IS NOT included in the firm's balance sheet
% yet, so we must manually add the face value of the bond to the debt
% and asset values.
% If the bond is being priced some year AFTER it was issued, then the
% book value will have absorbed the bond issue, so we do NOT have to
% add it in manually!
if year(firm.Bond.IssueDateNum) < year(priceDtNum)
  defBoundValue  = prevYrFinObs.TotLiab;
  assetValue    = priceDtImplAssetVal;
else
  % When calculating the default boundary K, we assume that the bond
  % defaults when the assets are less than the total liabilities. For a
  % firm that is issuing its first bond, we can take this to be total
  % liabilities reported at the end of last year, and the face value of
  % the total amount of bond being issued, that is:
  % [Default boundary = totalLiabsBeforeIssue + FaceValueOfIssue]
  defBoundValue  = prevYrFinObs.TotLiab + firm.Bond.FaceValue;

  % In a similar vein to the default boundary, the issuance of new debt
  % also increases the market value of the assets. We add the face value
  % of the debt to the market value of assets. This has the effect of
  % increasing the value of both debt and total asset valuations by the
  % same amount, but of course, will actually increase the leverage
  % ratio. E.g.:
  % At time t=0, V=1, K=0.5. We issue a bond with FV=0.1 at time t=1.
  % Leverage is then 0.5.
  % At time t=1, V=1+0.1, K=0.5+0.1, but leverage has increased to
  % 0.5455, thus representing increased risk of the firm.
  assetValue  = priceDtImplAssetVal + firm.Bond.FaceValue;
end

% For any coupons which were still outstanding after the
% observation start date at which time we are trying to price the
% bond, price them according to Merton now:
remMrktTaus  = CalcRemainingCouponTausInYrs(priceDtNum, ...
  firm.Bond.CouponDateNums);

% Predict the price of all the coupons
for coup_i = 1 : 1 : length(remMrktTaus)
  tau    = remMrktTaus(coup_i);

  % Only bother processing coupons if there were coupons to be
  % processed at some payout rate above zero
  if firm.Bond.CouponRate ~= 0
    if strcmp(estimMode, const.ModeVasicek)
      predictedFullCouponPrice =UnitDiscBondVasicek(tau,vParamsAtObsDt);
    elseif strcmp(estimMode, const.ModeMerton) || ...
        strcmp(estimMode, const.ModePureProxyM)
      predictedFullCouponPrice = UnitDiscBondMerton(...
        tau, ...
        assetValue,...
        defBoundValue,...
        vParamsAtObsDt,...
```

```
          priceDtPrevYrAssetParams.sigma,...
          rr_c);
      elseif strcmp(estimMode, const.ModeLS) || ...
          strcmp(estimMode, const.ModePureProxyLS)
        predictedFullCouponPrice = UnitDiscBondLongSchwartz(...
          tau,...
          assetValue,...
          defBoundValue,...
          vParamsAtObsDt,...
          rr_c,...
          priceDtPrevYrAssetParams.rho,...
          priceDtPrevYrAssetParams.sigma);
      end
      % Assuming semiannual coupon payments)
      cPrices(length(cPrices)+1) = (0.5 * firm.Bond.CouponRate *...
        predictedFullCouponPrice);
    end
  end

  % Predict the price of the face value payment
  faceTau  = remMrktTaus(end);
  if strcmp(estimMode, const.ModeVasicek)
    fvPrice = UnitDiscBondVasicek(faceTau,vParamsAtObsDt);
  elseif strcmp(estimMode, const.ModeMerton) || ...
      strcmp(estimMode, const.ModePureProxyM)
    fvPrice = UnitDiscBondMerton(...
      faceTau, ...
      assetValue,...
      defBoundValue,...
      vParamsAtObsDt,...
      priceDtPrevYrAssetParams.sigma,...
      rr_p);
  elseif strcmp(estimMode, const.ModeLS) || ...
      strcmp(estimMode, const.ModePureProxyLS)
    fvPrice = UnitDiscBondLongSchwartz(...
      faceTau,...
      assetValue,...
      defBoundValue,...
      vParamsAtObsDt,...
      rr_p,...
      priceDtPrevYrAssetParams.rho,...
      priceDtPrevYrAssetParams.sigma);
  else
    die(['Died: Invalid pricing mode specified: ' estimMode]);
  end

  % Tally the total predicted price of the bond (as a portfolio-of-zeros)
  PredictedBondPrice = sum(cPrices) + fvPrice;

  % Calculate the implied yield that would generate a bond with this
  % price
  % Since for testing purposes, all
  % bonds are priced as though they have a unit payoff, and our close
  % closes are stored similarly, we set the face value of the bond to
  % equal 1 when calculating yields.
  faceVal = 1;
  ImpliedYield = CalcImpliedYield(faceVal,firm.Bond.CouponRate,...
    remMrktTaus,PredictedBondPrice);
end

function PredictCouponBondPricesAtIssue()
%-------------------------------------------------------------------------
```

```
% @description:  PredictCouponBondPricesAtIssue
%         For the bond as close as possible to the issue date (within
%         a month, or it fails), perform pricing estimations and
%         save to a firm-specific file.
%--------------------------------------------------------------------------
  % Predict the bond prices at issue
  PredictCouponBondPricesByDates();
  % Make sure that our csv copy is up to date
  TabulateBondYieldsAtIssue();
end

function PredictCouponBondPricesByDates(targetPriceDtNums,maxDist)
%------------------------------------------------------------------------
% @description:
%     Given an array of dates that we wish to price all our bonds on,
%     loop through all firms and calculate our predictions using each
%     different model and save the updated firm data binaries.
% @note:
%     If no parameters are passed in, the logic defaults to pricing the
%     bonds as close as possible to the date of issue.
%------------------------------------------------------------------------
  const    = Constants();
  paths    = PathInfo();

  % Load all precalculated Vasicek interest rate model parameters so we
  % can use the instantaneous spot rates in our estimation of asset
  % dynamics.
  vasParams  = ParseInterestRateParamsVasicek();

  % Set up the recovery rates on the coupons and face values. In this
  % way, we add/remove risk from the bond pricing....
```

```
rr_c     = const.RecoveryRateCoupons;
rr_p     = const.RecoveryRateFaceValue;

firms = ParseCompanyList();
for firm_i = 1 : 1 : length(firms)
  tmpFirm  = firms(firm_i);
  load([paths.PreCalcFirmHistory tmpFirm.Bond.DSBondCode], 'firm');
  clear tmpFirm;

  if nargin == 0
    disp('No datenums specified, default price bond at to issue date!');
    targetPriceDtNums = [firm.Bond.IssueDateNum];
    maxDist = 40;
  end

  % Now, loop through all the intended targeting dates and predict
  % bond prices for each date, for each model.
  for targetDate_i = 1 : 1 : length(targetPriceDtNums)
    tryToPriceDtNum = targetPriceDtNums(targetDate_i);
    trytopriceon = datestr(tryToPriceDtNum)
    try
      % Search into the future, one day at a time
      moveDaysBy       = 1;
      actualPriceDtNum  = CalcClosestPossibleValuationDate(firm,...
        vasParams, tryToPriceDtNum, moveDaysBy, maxDist);
      actualPriceDtStr  = datestr(actualPriceDtNum, const.DateStringAU);

      if ~isfield(firm.Bond, 'PredBondPricesPPM')
        firm.Bond.PredBondPricesPPM = hashtable;
      end
      estimMode = const.ModePureProxyM;
```

```
disp(['Estimating: ' estimMode]);
[PredictedBondPrice  PredictedYield] = ...
  PredictCouponBondPriceAndYield(estimMode,actualPriceDtNum,...
  firm,vasParams,rr_c,rr_p)
firm.Bond.PredBondPricesPPM = put(firm.Bond.PredBondPricesPPM,...
  actualPriceDtNum,PredictedBondPrice);

if ~isfield(firm.Bond, 'PredBondPricesM')
  firm.Bond.PredBondPricesM = hashtable;
end
estimMode = const.ModeMerton;
[PredictedBondPrice  PredictedYield] = ...
  PredictCouponBondPriceAndYield(estimMode,actualPriceDtNum,...
  firm,vasParams,rr_c,rr_p)
firm.Bond.PredBondPricesM = put(firm.Bond.PredBondPricesM,...
  actualPriceDtNum,PredictedBondPrice);

if ~isfield(firm.Bond, 'PredBondPricesPPLS')
  firm.Bond.PredBondPricesPPLS = hashtable;
end
estimMode = const.ModePureProxyLS;
disp(['Estimating: ' estimMode]);
[PredictedBondPrice  PredictedYield] = ...
  PredictCouponBondPriceAndYield(estimMode,actualPriceDtNum,...
  firm,vasParams,rr_c,rr_p)
firm.Bond.PredBondPricesPPLS = put(...
  firm.Bond.PredBondPricesPPLS,actualPriceDtNum,...
  PredictedBondPrice);

if ~isfield(firm.Bond, 'PredBondPricesLS')
  firm.Bond.PredBondPricesLS = hashtable;
```

```
        end
        estimMode = const.ModeLS;
        [PredictedBondPrice  PredictedYield] = ...
          PredictCouponBondPriceAndYield(estimMode,actualPriceDtNum,...
          firm,vasParams,rr_c,rr_p)
        firm.Bond.PredBondPricesLS = put(firm.Bond.PredBondPricesLS,...
          actualPriceDtNum,PredictedBondPrice);

        if ~isfield(firm.Bond, 'PredBondPricesRF')
          firm.Bond.PredBondPricesRF = hashtable;
        end
        estimMode = const.ModeVasicek;
        [PredictedBondPrice  PredictedYield] = ...
          PredictCouponBondPriceAndYield(estimMode,actualPriceDtNum,...
          firm,vasParams,rr_c,rr_p);
        firm.Bond.PredBondPricesRF = put(firm.Bond.PredBondPricesRF,...
          actualPriceDtNum,PredictedBondPrice);
      catch
        disp('Error in pricing attempt, ignore and continue.');
      end
    end
    % Finally, save the individual firm objects as matlab binary files
    % so they can be retrieved very quickly in future.
    save([paths.PreCalcFirmHistory firm.Bond.DSBondCode], 'firm');
  end
end

function PredictCouponBondPricesByMonthlyDates()
%----------------------------------------------------------------------
% @description:  Specify a broad band of end-of-month dates, over several
%        years, and trigger
%        bond price estimate for each firm accordingly.
%-----------------------------------------------------------------------
  const    = Constants();
  maxDist = 14;
  tryToPriceDtNums = [];
  for yrInd = 2002 : 1 : 2008
    for mnthInd = 1 : 1 : 12
      % We wish to loop through several years, pricing each
      % firm's bond as close to the end of each month that we
      % can do so. Calculate the end day of each month:
      eom    = eomday(yrInd, mnthInd);
      tryToPriceDtNum  = datenum([num2str(eom) '/' num2str(mnthInd) '/'...
        num2str(yrInd)], const.DateStringAU);
      tryToPriceDtNums(end+1) = tryToPriceDtNum;
    end
  end
  PredictCouponBondPricesByDates(tryToPriceDtNums,maxDist);
end


function [PredictedBondPrice PredictedYield ActualBondPrice ...
    ActualYield] = PriceAndComparePredictedToActual(estimMode,...
    actualPriceDtNum,firm,vasParams,rr_c,rr_p)
%-----------------------------------------------------------------------
% @description:  PriceAndComparePredictedToActual
%        For any particular pricing methodology, calculate the
%        implied and actual price/yields, for comparison
%-----------------------------------------------------------------------
  [PredictedBondPrice  PredictedYield] = PredictCouponBondPrice(...
    estimMode,actualPriceDtNum,firm,vasParams,rr_c,rr_p);

  % Now, retrieve the actual observed bond price and the implied
  % yield, and we can compare the two.

  % Since for testing purposes, all
  % bonds are priced as though they have a unit payoff, and our close
  % closes are stored similarly, we set the face value of the bond to
  % equal 1 when calculating yields.
  faceVal        = 1;

  % Calculate the remaining coupon payment times, so we can solve for
  % yield
  remMrktTaus  = CalcRemainingCouponTausInYrs(actualPriceDtNum, ...
    firm.Bond.CouponDateNums);

  % Calculate actual bond price and subsequent yield to compare with
  % our above predictions.
  bondPriceObs    = get(firm.Bond.Prices, actualPriceDtNum);
  ActualYield      = CalcImpliedYield(...
    faceVal,...
    firm.Bond.CouponRate,...
    remMrktTaus,...
    bondPriceObs.ClosePrice);
  ActualBondPrice = bondPriceObs.ClosePrice;
end

function const = Constants()
%------------------------------------------------------------------------
% @description:  CONSTANTS
%         Declare all the "constant", (i.e. lookup) values used
%         throughout the processing of our bond records.
% @note:    Due to the daftness of Matlab, these are not constant in
%         the true immutable sense of the word, but they will NEVER
%         be overwritten by my code. Could have effectively called
%         this section "lookUpKeys" for example...
%------------------------------------------------------------------------
  %%%
  % Specify which method of parameter estimation is to be
  % used when generating asset dynamics estimates.
  const.ModeMerton    = 'merton';
  const.ModeLS      = 'ls';
  const.ModePureProxyM  = 'ppm';
  const.ModePureProxyLS  = 'ppls';
  const.ModeVasicek    = 'vas';

  % Constants for plotting consistancy
  const.ColourPP      = 'g';
  const.PointPP      = 'x';
  const.LinePP      = '--';
  const.PlotLegendPP    = 'Pure Proxy';

  const.ColourPPM      = 'k';
  const.PointPPM      = 'x';
  const.LinePPM      = '-';
  const.PlotLegendPPM    = 'Merton Pure Proxy';

  const.ColourM      = 'r';
  const.PointM      = 's';
  const.LineM       = ':';
  const.PlotLegendM    = 'Merton MLE';

  const.ColourPPLS    = 'm';
  const.PointPPLS      = '*';
  const.LinePPLS      = '-';
const.PlotLegendPPLS  = 'L&S Pure Proxy';

const.ColourLS      = 'b';
const.PointLS      = '+';
const.LineLS      = '-.';
const.PlotLegendLS    = 'L&S MLE';

const.ColourAct      = 'g';
const.PointAct      = 'o';
const.LineAct      = '-';
const.PlotLegendAct    = 'Actual';

% For the VASICEK predictions, NOT actual treasury yields
const.ColourRF      = 'c';
const.PointRF      = 'd';
const.LineRF      = '-';
const.PlotLegendRF    = 'Vasicek';

% For actual released US Treasury yield curve values
const.ColourTY      = 'r';
const.PointTY      = 'x';
const.LineTY      = '-';
const.PlotLegendTY    = 'US Treasury';

const.ColourEquity    = 'k';
const.LineEquity    = '-';
const.PointEquity    = '';
const.PlotLegendEquity  = 'Equity';

const.ColourTotLiab    = 'k';
const.LineTotLiab    = '-.';
const.PointTotLiab    = '';

const.PlotLegendTotLiab  = 'Total Liabilities';


%%%

% Wong&Choi2007 show empirically that bond default boundaries tend to be

% less than book value of total liabilities and median default boundary

% is 73.8% of total liability.

const.WongChoiDefBoundRatio    = 0.738;


%%%

% Know how to handle date strings when reading in from csv files

const.DateStringUSA        = 'mm/dd/yyyy';

const.DateStringAU        = 'dd/mm/yyyy';

const.DateStringAUCompressed  = 'ddmmyyyy';


%%%

% Eom recovery rate on coupons and face value

const.RecoveryRateCoupons    = 0;

const.RecoveryRateFaceValue    = 0.5131;


%%%

% ComputStat-specific codes and constants:

%

% String which appears in CompuStat export files when some particular

% data entry is unavailable. We test for this to see when to quit

% searches across files etc.

const.CompuStatDataNA      = '@NA';

% Total liabilities row string in historical company finances files

const.CsTotalLiabs        = 'Total Liabilities';

% Total Liabilities listed in millions in CompuStat?

const.CsTotalLiabMult      = 1000000;
```

```
  % Total shares outstanding
  const.CsSharesOutSt        = 'Common Shares O/S';
  % # Shares listed in millions in CompuStat?
  const.CsSharesOutStMult      = 1000000;
  % Cash row... use this to know where the CompuStat files store the
  % date information, relative to the cash column.
  const.CsCash          = 'Cash';
  % Compustat historic monthly date format
  const.CsMnthlyDtFormat      = 'mmm-yy';

  %%%
  % Datastream-specific codes and constants
  %
  % Amount of bond on issue listed in 1000's ?
  const.DSAmountOutStMult      = 1000;
end

function dayOfYear = DayOfYear(theDateNum)
%-------------------------------------------------------------------------
% @description:
% @acknowledgement:  Peter J. Acklam, http://home.online.no/~pjacklam
%-------------------------------------------------------------------------
  % Get the specific values of the date passed in for processing
  theDateVec  = datevec(theDateNum);
  day      = theDateVec(3);
  month    = theDateVec(2);
  year    = theDateVec(1);

  days_in_prev_months = [0 31 59 90 120 151 181 212 243 273 304 334];

  dayOfYear = days_in_prev_months(month) ...    % days in prev. months
```

```
    + ( isleapyear(year) & ( month > 2 ) ) ...  % leap day
    + day;                    % day in month
end

function [daysInYear] = DaysInYear(year)
%----------------------------------------------------------------------
% @description: DaysInYear
%         Calculate how many days there are in any given year.
%----------------------------------------------------------------------
  daysInYear = 365;
  if isleapyear(year)
    daysInYear = 366;
  end
end

function out = N(x)
%-------------------------------------------------------------------------
% @description:  Generic library function to shorten the notation for
%         cumulative standard normal distribution function
%-------------------------------------------------------------------------
  out = normcdf(x,0,1);
end

function paths = PathInfo()
%-------------------------------------------------------------------------
% @description:  PathInfo
%         Return a structure containing all the paths for the
%         relevant files and directories of interest while
%         implementing my project. Store these is one convenient
%         location.
%-------------------------------------------------------------------------
```

```
c_driveLappy      = 'D:\';
c_driveBeast      = 'F:\';

c_projectHomeDir    = ['Documents\UQ Study\MastScience\2008 ' ...
  'Sem02\MATH7021 Project\ThesisData\';
c_bondPricesDir      = 'Bond Prices\';
c_bondInfoDir      = 'Bond Info\';
c_financialsDir      = 'Financials\';
c_interestRatesDir    = 'Interest Rates\';
c_sharePricesDir    = 'Share Prices\';
c_processedDir      = 'Processed\';
c_sourcedDir      = 'Sourced\';
c_firmData        = 'Firm Data\';
c_imagesDir        = 'Images\';

if exist([c_driveLappy c_projectHomeDir])
  c_drive        = c_driveLappy;
else
  c_drive        = c_driveBeast;
end

%%%
% Sourced information, from a combination of CompuStat, Datastream, and
% Yahoo share price (bond price?) information.
paths.SharePricesDir  = [c_drive c_projectHomeDir ...
  c_sourcedDir c_sharePricesDir];
paths.BondPricesDir    = [c_drive c_projectHomeDir ...
  c_sourcedDir c_bondPricesDir];
paths.BondInfoDir    = [c_drive c_projectHomeDir ...
  c_sourcedDir c_bondInfoDir];
paths.FinancialsDir    = [c_drive c_projectHomeDir ...
```

```
  c_sourcedDir c_financialsDir];
paths.InterestRatesDir  = [c_drive c_projectHomeDir ...
  c_sourcedDir c_interestRatesDir];

%%%
% Prefixes used for each file related to a particular bond issue. Used
% largely as a method to make the files differently named so we can
% open up all files related to a single bond issue at the same time in
% Excel, which is fiddly about files of the same name *sigh*

% Prefix for files related to bond issuance data - issue date, coupon
% rate etc
paths.BondInfoPre    = 'bi_';
% Prefix for files related to daily bond price observations.
paths.BondPricePre    = 'bp_';
% Prefix for files related to daily share price observations
paths.EqtyPricePre    = 'ep_';
% Prefix for files related to company historical finance statements
paths.CompFinPre    = 'cf_';

%%%
% All the specific lookup files that we might need
%
% The source of historic interest rate information, contains values
% from the USA Treasury zero-curve out to maturities of 10 years.
paths.HistoricInterestRateFile  = [paths.InterestRatesDir ...
  'US Treasury Zero1-10.csv'];

% The joining table containing a company name and DS bond code that
% all a given company's bond, share price, and financial statements
% information is subsequently identified by.
% Contains the bond-specific information about all the bonds that we
% are analysing for this project (coupon, issue date, maturity etc)
paths.BondInfoFile        = [c_drive c_projectHomeDir ...
  c_processedDir c_bondInfoDir 'Bond Info.csv'];

% Contains parameter estimates for each daily term structure
% observation matched to the Nelson-Siegel1987 interest rate model
paths.NelsonSiegelPredictions  = [c_drive c_projectHomeDir ...
  c_processedDir c_interestRatesDir 'NelsonSiegelParams.csv'];

% Contains parameter estimates for each daily term structure
% observation matched to the Vasicek1977 interest rate model
paths.VasicekPredictions    = [c_drive c_projectHomeDir ...
  c_processedDir c_interestRatesDir 'VasicekParams.csv'];

%%%
% For a massive speed increase, we can save precalculated values, such
% as the Vasicek interest rate parameter predictions, into a matlab
% binary file for very quick reload.
%
% Filename of the precalculated vasicek interest rate parameters
paths.PreCalVasicekPredictFile  = [c_drive c_projectHomeDir ...
  c_processedDir c_interestRatesDir 'vasicekParams.mat'];
paths.PreCalVasicekPredictVar  = 'vasicekParams';

% Directory of the individual firm data files, which include all the
% bond issue data, the equity price history, and financial data
% history. Firms stored as <dsBondCode>.mat e.g. '1234F3.mat'
paths.PreCalcFirmHistory    = [c_drive c_projectHomeDir ...
  c_processedDir c_firmData];
```

```
% Where to store our tables of precalculated values for asset dynamics
% and predictions of bond prices
paths.TabularAssetDynamicsFile  = [c_drive c_projectHomeDir ...
  c_processedDir c_firmData 'Firm Asset Dynamics.csv'];
paths.TabularBondIssuePricesFile= [c_drive c_projectHomeDir ...
  c_processedDir c_firmData 'Bond Issue Price Predictions.csv'];
paths.TabularBondHistoricalPricesFile= [c_drive c_projectHomeDir ...
  c_processedDir c_firmData 'Bond Historical Price Predictions.csv'];

paths.ThesisImages        = [c_drive c_projectHomeDir ...
  c_processedDir c_imagesDir];
% Prefix for plots of yearly asset dynamics, asset paths, implied
% yields etc:

% Plots of asset dynamics estimates by year
paths.YrlyAssDynPlotsPre    = 'yadp_';
% Plots of implied asset valuations
paths.AssPathPlotsPre      = 'app_';
%
paths.YieldAtIssuePlot      = 'yields_at_issue';
% Plots of historic yield calculations
paths.YieldHistoryPlotPre    = 'yhp_';
paths.VasR0VsTreasuryPlot    = 'vas_r0_vs_treasury';
% Fitted Vasicek images
paths.VasLeastSqrFitPre      = 'vas_fit_';
% Asset dynamics scatter plots per-firm prefix
paths.AssDynScatPre        = 'ads_';
% Asset dynamics scatter plot full sample
paths.AssDynAllFirmsScat    = 'scat_plot_all_vols';
% Scatter of bond yields at issue
paths.YieldsAllFirmsScat    = 'scat_all_issue_yld';
```

```
end


function [x] = ZeroClean(x)
%--------------------------------------------------------------------------
% @description:  Generic library function to offer us a standard way of
%         checking critical values. Should some value HAVE to be
%         NON-ZERO to avoid Divide-By-Zero errors, we can clean it
%         using this function.
%         Note that we set the alternative value to be 'EXTREMELY'
%         small to try to avoid DBZ errors, but have the outcome of our
%         modelling significantly affected either!
% @params:
%  x       - Matrix of values to perform our cleansing of zero values
%         on. [n*m or m*n vector]
% @example:
%         dirtyVals  = [0,1,2,3,4,0]
%         cleanVals  = ZeroClean(dirtyVals)
%--------------------------------------------------------------------------
  x(find (x == 0)) = 0.000000000001;
end


function [BondPrices] = ParseBondPricesByDSBondCode(dsBondCode)
%--------------------------------------------------------------------------
% @description:  ParseBondPricesByDSBondCode
%         Based on a company's DataStream bond code, find the file
%         with all the historical adjusted equity closing prices.
%         Load up each observation and store it in a collection
%         indexed by datenum() values.
%--------------------------------------------------------------------------
  tic
  disp(['Retrieving historical bond price obs for:' num2str(dsBondCode)]);
```

```
constants    = Constants();
paths      = PathInfo();
sourceFile= [paths.BondPricesDir paths.BondPricePre dsBondCode '.csv'];
[dataIn, result]= readtext(sourceFile, '[,]', '', '"', 'textual');
[dataInRows dataInCols]  = size(dataIn);

% Create an initial hash to store daily bond price observations. Start
% with approx 3 years of spaces reserved, for speed...
BondPrices = hashtable('size',1000);

% Which columns the various items of interest are to be found in
dateCol       = find(strcmpi(dataIn(2,:), 'Code'));
priceCol     = find(strcmpi(dataIn(2,:), [dsBondCode '(GP)']));

% For each end of day bond price observation, record it against the
% date now. We know that our data might be dirty and might include dates
% against which there are no bond price observations. We test that both
% values exist before adding the date as a key in our hash. This is
% unfortunately slower than need be, but it handles dirty data more
% easily.
for index = 3 : 1 : dataInRows
  dataInDtStr    = cell2mat(dataIn(index,dateCol));
  dataInObsClose  = cell2mat(dataIn(index,priceCol));

  % Only add the row of data if all required values exist
  if length(dataInDtStr) ~= 0 && length(dataInObsClose) ~= 0
    dailyObs.ObsDateNum  = datenum(dataInDtStr,constants.DateStringAU);
    dailyObs.ClosePrice  = str2num(dataInObsClose)/100;

    % Add observation values to the series, keyed on observation
    % date
```

```
      BondPrices = put(BondPrices, dailyObs.ObsDateNum, dailyObs);
    end
  end
  disp(['Successfully loaded historic bond price data for: ' dsBondCode]);
  disp(['Total # observations found: ' num2str(count(BondPrices))]);
  disp(['Total processing time: ' num2str(toc)]);
end


function [firms] = ParseCompanyList()
%--------------------------------------------------------------------------
% @description:  Load up a collection of all the companies whose debt issue
%    we will be examining.
% @history:
%  2008/08/22  Cannot seem to find company records in Datastream to match
%        the bond issues we find. Instead, we will have to manually
%        trawl through CompuStat looking for matching companies.
%        What this means is that we no longer retrieve financial
%        data of any kind about a company from Datastream. Compustat
%        becomes the source of our power...
%--------------------------------------------------------------------------
  disp([' ']);
  disp(['Loading firm-bond pairs:']);
  paths  = PathInfo();
  const  = Constants();
  [dataIn, result]= readtext(paths.BondInfoFile,'[,]', '', '"','textual');
  [dataInRows dataInCols]  = size(dataIn);

  % For our collection of bonds that we will be performing experiments
  % on, find and store enough information so we can find their relevant
  % details in linked files later on.
  % Which columns the various items of interest are to be found in:
```

```
bondDsCodeCol  = find(strcmpi(dataIn(1,:), 'Type'));
issuerNameCol  = find(strcmpi(dataIn(1,:), 'CompName'));
coupCol      = find(strcmpi(dataIn(1,:), 'C'));
issDtCol    = find(strcmpi(dataIn(1,:), 'ID'));
matDtCol    = find(strcmpi(dataIn(1,:), 'RD'));
coupDtCol    = find(strcmpi(dataIn(1,:), 'CD'));
amountOutCol  = find(strcmpi(dataIn(1,:), 'AOS'));
moodyRateCol  = find(strcmpi(dataIn(1,:), 'MRT'));

% Store all the firms and their bond details now
firms = [];
for rowInd = 2 : 1 : dataInRows

  placeAt  = length(firms)+1;

  % Load up the basic information about each bond, enough that we can
  % then go and process their various linked files later on one by
  % one.
  firms(placeAt).CompName  = cell2mat(dataIn(rowInd,issuerNameCol));
  firms(placeAt).Bond.DSBondCode    = ...
      cell2mat(dataIn(rowInd,bondDsCodeCol));
  firms(placeAt).Bond.IssueDateNum  = ...
      datenum(cell2mat(dataIn(rowInd,issDtCol)), const.DateStringAU);
  firms(placeAt).Bond.MatDateNum    = datenum(cell2mat(dataIn(rowInd,...
      matDtCol)), const.DateStringAUCompressed);
  firms(placeAt).Bond.CouponRate    = ...
      str2num(cell2mat(dataIn(rowInd,coupCol)))/100;

  % Total amount to be repaid at maturity, in effect, book value of
  % the liability that the firm takes on as a result of this debt:
  firms(placeAt).Bond.FaceValue    = str2num(cell2mat(dataIn(rowInd,...
        amountOutCol)))*const.DSAmountOutStMult;
    firms(placeAt).Bond.MoodysRating = ...
        cell2mat(dataIn(rowInd,moodyRateCol));

    % Calculates all the coupon dates
    firms(placeAt).Bond.CouponDateNums  = CalculateCouponDates(...
      firms(placeAt).Bond.IssueDateNum, ...
      firms(placeAt).Bond.MatDateNum, ...
      cell2mat(dataIn(rowInd,coupDtCol)));
  end

  disp(['Loaded collection of companies for param estimation.']);
  disp(['Total # companies found: ' num2str(length(firms))]);
  %%% End parsing of company list logic

  %%% Begin private methods
  %----------------------------------------------------------------------
  % @description:  Based on a string containing two embedded coupon date
  %        day/month pairs, parse that string and extract the
  %        values so that we can calculate on which days the
  %        coupons land. Store a collection of all the dates that
  %        the bond will pay coupons.
  %----------------------------------------------------------------------
  function [couponDates] = CalculateCouponDates(issDateNum, ...
          matDateNum, couponDateStr)
    issDateVec    = datevec(issDateNum);
    matDateVec    = datevec(matDateNum);

    % Store all coupons calculated between the issue and maturity date
    % range.
    couponDates    = [];
```

```
len        = length(couponDateStr);
coup2MStr    = couponDateStr(1,len-1:len);
coup2DStr    = couponDateStr(1,len-3:len-2);
coup1MStr    = couponDateStr(1,len-5:len-4);
% Some date strings are passed in as a 7 digit string instead of 8
if 8 == len
  spacer = 2;
else
  spacer = 1;
end
coup1DStr    = couponDateStr(1,(1:spacer));

% For each year in the range of the bond life, calculate a first
% and second coupon date, and if they are within the issue and
% maturity date, add them to the collection.
for yrInd = issDateVec(1) : 1 : matDateVec(1)
  yrStr  = num2str(yrInd);
  coupon1Num  = datenum([coup1DStr '/' coup1MStr '/' yrStr], ...
      const.DateStringAU);
  coupon2Num  = datenum([coup2DStr '/' coup2MStr '/' yrStr],...
      const.DateStringAU);

  % If the calculated coupon dates are valid, (between issue date
  % and maturity date, and making sure that first coupon is paid no
  % less than 6 months from issue), add them:
  if coupon1Num > issDateNum && coupon1Num <= matDateNum && ...
      (coupon1Num - issDateNum >= 180)
    couponDates(end+1) = coupon1Num;
  end
  if coupon2Num > issDateNum && coupon2Num <= matDateNum && ...
```

```
          (coupon2Num - issDateNum >= 180)
        couponDates(end+1) = coupon2Num;
      end
    end
  end
end

function [Equity] = ParseEquityByDSEquityCode(dsBondCode)
%--------------------------------------------------------------------------
% @description:  ParseEquityByDSEquityCode
%         Based on a company's DataStream bond code, find the file
%         with all the historical adjusted equity closing prices.
%         Load up each observation and store it in a collection
%         indexed by datenum() values.
%--------------------------------------------------------------------------
  tic; disp([' ']);
  disp(['Retrieving share price observations for: ' num2str(dsBondCode)]);

  constants    = Constants();
  paths      = PathInfo();
  sourceFile= [paths.SharePricesDir paths.EqtyPricePre dsBondCode '.csv'];
  [dataIn, result]= readtext(sourceFile, ...
    '[,]', '', '"', 'textual');
  [dataInRows dataInCols]  = size(dataIn);

  % Create an initial hash to store daily share price observations. Start
  % with approx 3 years of spaces reserved, for speed...
  Equity = hashtable('size',1000);

  % Which columns the various items of interest are to be found in
  dateCol      = find(strcmpi(dataIn(1,:), 'Date'));
  adjCloseCol    = find(strcmpi(dataIn(1,:), 'Adj Close'));

  % For each end of day adjusted price observation, record it against the
  % date now. We know that our data might be dirty and might include dates
  % against which there are no share price observations. We test that both
  % values exist before adding the date as a key in our hash. This is
  % unfortunately slower than need be, but it handles dirty data more
  % easily.
  for index = 2 : 1 : dataInRows
    dataInDtStr    = cell2mat(dataIn(index,dateCol));
    dataInObsClose  = cell2mat(dataIn(index,adjCloseCol));

    % Only add the row of data if all required values exist
    if length(dataInDtStr) ~= 0 && length(dataInObsClose) ~= 0
      dailyObs.ObsDateNum  = datenum(dataInDtStr,constants.DateStringAU);
      dailyObs.AdjClose  = str2num(dataInObsClose);

      % Add observation values to the series, keyed on observation date
      Equity = put(Equity, dailyObs.ObsDateNum, dailyObs);
    end
  end

  disp(['Successfully loaded share price data for: ' dsBondCode]);
  disp(['Total # observations found: ' num2str(count(Equity))]);
  disp(['Total processing time: ' num2str(toc)]);
end

function [Financials] = ParseFinancialsByDSBondCode(dsBondCode)
%-------------------------------------------------------------------
% @description:  ParseFinancialsByDSBondCode
%        Uses exports of historical financial data from CompuStat in
%       CSV format!!
%--------------------------------------------------------------------------
  tic; disp([' ']);
  disp(['Loading historic firm financials for: ' dsBondCode]);

  const      = Constants();
  paths      = PathInfo();
  sourceFile= [paths.FinancialsDir paths.CompFinPre dsBondCode '.csv'];

  [dataIn, result]= readtext(sourceFile, ...
    '[,]', '', '"', 'textual');
  [dataInRows dataInCols]  = size(dataIn);

  % Store historical financial information in a hash
  Financials  = hashtable('size', dataInCols);

  % Find the appropriate row for each time-series value we are interested
  % in, and then iterate through the rows recording the values as
  % appropriate.
  totLiabRow    = find(strcmpi(dataIn(:,1), const.CsTotalLiabs));
  numShOutstRow  = find(strcmpi(dataIn(:,1), const.CsSharesOutSt));
  datesRow    = (find(strcmpi(dataIn(:,1), const.CsCash)) - 1);

  % Loop through as many columns of data as are available, testing each
  % one to make sure that its a valid entry and not just appearing as a
  % result of our broad search range terms:
  for colInd = 2 : 1 : dataInCols
    totalLiabsStr  = cell2mat(dataIn(totLiabRow,colInd));
    numOutSharesStr  = cell2mat(dataIn(numShOutstRow,colInd));
    dateStr      = cell2mat(dataIn(datesRow,colInd));
    % If is a a valid column, all the fields we require will exist and
    % not equal const.CompuStatDataNA:
    if ~strcmp(totalLiabsStr, const.CompuStatDataNA) && ...
        ~strcmp(numOutSharesStr, const.CompuStatDataNA) && ...
        ~strcmp(dateStr, const.CompuStatDataNA)

      obsDateVec      = datevec(dateStr,const.CsMnthlyDtFormat);
      obsDateVec(3)    = 31; % Set to the end of the month
      dailyObs.ObsDateNum  = datenum(obsDateVec);
      dailyObs.TotLiab  = str2num(totalLiabsStr) * const.CsTotalLiabMult;
      dailyObs.OutStShares= str2num(numOutSharesStr) * ...
          const.CsSharesOutStMult;

      % Store the yearly observations in the hash, keyed by year
      Financials = put(Financials, obsDateVec(1), dailyObs);
    end
  end

  disp(['Loaded historical financial data for: ' dsBondCode]);
   disp(['Total # yrly observations found: ' num2str(count(Financials))]);
  disp(['Total processing time: ' num2str(toc)]);
end

function [vasicekParams] = ParseInterestRateParamsVasicek(forceReload)
%-------------------------------------------------------------------------
% @description:  ParseInterestRateParamsVasicek
%        Read in the csv of historic yield curve zero spot rates, and
%        store an estimation of the Vasicek1977 parameters which
%        would generate a curve to most closely match those observed
%        values.
%        Takes in a csv file like:
```

```
%         Date,1,2,3,4,5,6,7,8,9,10,11,12,24,36,48... <- months
%         1/1/1997,6.9,7.0,7.2,6.8 <- yield curve values
%         For each set of daily observations, estimate the parameters
%         in the Vasicek1977 model which would generate that yield
%         curve, and store them in in a binary file for quick future
%         retrieval.
%-------------------------------------------------------------------------
  if nargin == 0
    forceReload = false;
  end

  constants        = Constants();
  paths          = PathInfo();

  % If values already exist stored as a matlab binary file
  if 2 == exist(paths.PreCalVasicekPredictFile) && ~forceReload
    load(paths.PreCalVasicekPredictFile, paths.PreCalVasicekPredictVar);
    disp('Retrieved Vasicek parameter data from binary file');
  % Else load up the values from precalculated csv source files.
  else
    [dataIn, result]    = readtext(paths.HistoricInterestRateFile, ...
      '[,]', '', '"', 'empty2NaN');
    [dataInRows dataInCols]  = size(dataIn);

    % Store a copy of the original data values, so that we can save them
    % csv file for handy future reference. We will append our calculated
    % values to the end columns.
    dataOut    = dataIn;
    dataOut(1,dataInCols+1)  = {'r0'};
    dataOut(1,dataInCols+2)  = {'theta'};
    dataOut(1,dataInCols+3)  = {'kappa'};
dataOut(1,dataInCols+4)  = {'eta'};

% Pull out the maturity, observation date and market interest rate
% observation values from the source CSV file:
mrktTausMths  = cell2mat(dataIn(1,2:dataInCols));
obsDates    = dataIn(2:dataInRows,1);
mrktYields    = cell2mat(dataIn(2:dataInRows, 2:dataInCols));
[rateObsRows rateObsCols] = size(mrktYields);

% Convert cells to matrices so we can pass them into our
% parameterisation functions, we store the mrktTaus in YEARS for yrly
% parameter estimates
mrktTaus    = mrktTausMths(1,:)/12;

% Store each daily observation of interest rate values and our
% estimated parameters in a hash
vasicekParams  = hashtable;

% Day by day, go through the interest rate observations and try to
% predict the parameters that would have generated such a term
% structure, storing them in a cell so we can save the processed vals
% out to a CSV file:
for obsIndex = 1 : 1 : rateObsRows
  if mod(obsIndex,1) == 0
    disp(['Processing record #' num2str(obsIndex)]);

    dailyRateObs  = mrktYields(obsIndex,:)/100;
    dailyParams    = YieldCurveFitVasicek(mrktTaus, dailyRateObs);
    dailyParams.ObsDateNum  = datenum(obsDates(obsIndex,1), ...
        constants.DateStringAU);
```

```
for mrktTausMths_i = 1 : 1 : length(mrktTausMths)
  varnameMt  = ['m' num2str(mrktTausMths(mrktTausMths_i))];
  dailyParams.(varnameMt)  = dailyRateObs(mrktTausMths_i);
end

observedOn = datestr(dailyParams.ObsDateNum)
guessR0    = dailyParams.r0
guesstheta  = dailyParams.theta
guesskappa  = dailyParams.kappa
guesseta  = dailyParams.eta
vasicekParams = put(vasicekParams, ...
    dailyParams.ObsDateNum, dailyParams);

% Now, given a set of estimated paramteres, plot the observed yld
% curve for this, and compare that with the curve generated by our
% estimated values:
guessPrices  = UnitDiscBondVasicek(mrktTaus,dailyParams);
guessYields  = CalcDiscountBondYield(mrktTaus, guessPrices);

close all;
hold on;
plot(mrktTaus, dailyRateObs, 'g-s');
plot(mrktTaus, guessYields, 'r-x');
axis([0,max(mrktTaus),0.01,0.11]);
pause(0.55)

% Finally, add our calculated values to the dataOut object
dataOut(obsIndex+1,dataInCols+1)  = {dailyParams.r0};
dataOut(obsIndex+1,dataInCols+2)  = {dailyParams.theta};
dataOut(obsIndex+1,dataInCols+3)  = {dailyParams.kappa};
dataOut(obsIndex+1,dataInCols+4)  = {dailyParams.eta};
```

```
      end
    end

    % Now, we want to save the dataOut values to csv so we can view them
    % easily for future reference.
    destination  = paths.VasicekPredictions;
    WriteCellToCsv(destination, dataOut);

    % Also save the binary file, so we don’t have to go through this
    % process again next time:
    save(paths.PreCalVasicekPredictFile, paths.PreCalVasicekPredictVar);
  end
end

function [params] = YieldCurveFitVasicek(mrktTaus, mrktYields)
%--------------------------------------------------------------------------
% @description:  Attempt to estimate the parameters of the Vasicek
%        1977 short-term risk-free interest rate
%        term structure model of yields. We do this by
%        using least-squares for a cross-sectional data set, that
%        is, a structural yield-to-maturity curve for zero-coupon
%         treasury instruments.
% @params:
%  mrktTaus  - Matrix of times to maturity for which we are calculating
%        the yield-to-maturity under Vasicek model. Must be a
%        1*n or an n*1 matrix... m*n will cause irregular results.
%  mrktYields  - Matrix of observed yields-to-maturity that correspond to
%        the matching index in mrktTaus.
% @example:
%        mrktTaus      = [.125,.25,.5,1,2,3,5,7,10,20,30];
%        mrktYields  =
%        [2.57,3.18,3.45,3.34,3.12,3.13,3.52,3.77,4.11,4.56,4.51];
%        mrktYields = mrktYields / 100;
%        params = YieldCurveFitVasicek(mrktTaus, mrktYields)
% @author:    Dale Holborow, daleholborow@hotmail.com, August 7, 2008
%-------------------------------------------------------------------------
  % Set up some initial values for parameter guesses. Had initial
  % troubles trying to use 'sensible' initial guesses with low number of
  % function eval/iterations. Resolved by greatly increasing number of
  % function eval/iterations in order to be sure end result was precice,
  % so initial param estimates become far less important, now we just use
  % some random approximations in the rough vacinity of the expected
  % answers:
  pars(1)    = mrktYields(1);    % interest rate r0
  pars(2)    = mrktYields(end);  % interest rate long term mean theta
  pars(3)    = 0.40*rand;    % interest rate mean reversion rate kappa
  pars(4)    = 0.05*rand;    % interest rate volatility eta


  % Specify some lower/upper bounds for each parameter we will be
  % estimating (r,theta,kappa,eta):
  lowBound      = [0, 0, -10, 0];
  upBound        = [0.15, 0.25, 10, 10];

  % Make optimisation routine VERY precice by performing the routine LOTS
  % of times. Sacrifice a bit of speed for a highly precise and accurate
  % answer (hopefully).
  options        = optimset('lsqnonlin');
  options.TolFun    = 10^-20;
  options.MaxFunEvals = 4*1000;
  options.MaxIter    = 4*1000;
  options.TolX    = 10^-10;
options.Display    = 'off';

% Now, estimate our optimal parameters to match the observed market yld
% curve values.
[pars,resNorm,residual,exitFlag,output] = lsqnonlin(@(pars) ...
  VasicekFit(pars),pars,lowBound,upBound,options);

% Retrieve the optimal values and place them into a structure so they
% can be returned by the function.
params.r0    = pars(1);
params.theta  = pars(2);
params.kappa  = pars(3);
params.eta    = pars(4);
%%% End optimal parameter estimation logic %%%

%-----------------------------------------------------------------------
% @description:  Return the array of differences between predicted and
%        observed yields-to-maturity based on an estimated
%        parameter set, so we can implement our least-squared
%        error algorithm.
function [F] = VasicekFit(tmpPars)
  tmpParams.r0  = tmpPars(1);
  tmpParams.theta  = tmpPars(2);
  tmpParams.kappa  = tmpPars(3);
  tmpParams.eta  = tmpPars(4);

  prices  = UnitDiscBondVasicek(mrktTaus,tmpParams);
  yields  = CalcDiscountBondYield(mrktTaus,prices);

  F = mrktYields - yields;
end
```

```
end


function params = NelsonSiegelCurveFit(mrktTaus, mrktYields)
%-----------------------------------------------------------------------
% @description:  Attempt to estimate the parameters of the Nelson-Spiegel
%         1987 short-term risk-free interest rate
%         term structure model of yields, from their paper
%         'Parsimonious modelling of Yield Curves'. We do this by
%         using least-squares for a cross-sectional data set, that
%         is, a structural yield-to-maturity curve for zero-coupon
%          treasury instruments.
% @note:    This optimisation hinges on the choice of zeta, because
%         the subsequent values of the betaX's can be explicitly
%         backed out of the market observations. Hence, this should
%         be a very precise method of parameter estimation, and is
%         in fact about 8 times faster than when estimating all four
%         parameters using "lsqnonlin", for example.
%         I made slight tweaks to the code and prevented it from
%         erroring when encountering a zero tau, but full credit to
%         Dimitri.
% @params:
%  mrktTaus  - Matrix of times to maturity for which we are calculating
%         the yield-to-maturity under Nelson-Siegel model. Must be a
%         1*n or an n*1 matrix... m*n will cause irregular results.
%  mrktYields  - Matrix of observed yields-to-maturity that correspond to
%         the matching index in mrktTaus. Yields should be passed in
%         represented as percentage values, e.g. 4.5% should be
%         passed in as 0.045, NOT 4.5, so that initial guesses
%         for the optimisation for pars(2) and pars(4) are in
%         proportion to the rest.
% @example:
```

```
%        mrktTaus    = [.125,.25,.5,1,2,3,5,7,10,20,30];
%        mrktYields  =
%        [2.57,3.18,3.45,3.34,3.12,3.13,3.52,3.77,4.11,4.56,4.51];
%        mrktYields  = mrktYields/100;
%        params = NelsonSiegelCurveFit(mrktTaus, mrktYields)
% @author:    Dale Holborow, daleholborow@hotmail.com, August 7, 2008
% @acknowledge:  Dimitri Shvorob, dimitri.shvorob@vanderbilt.edu, 12/30/07
%        http://www.mathworks.com/matlabcentral/fileexchange/loadFil
%        e.do?objectId=18160&objectType=file
%-------------------------------------------------------------------------

  % Perform data cleaning. For example, many parameters cannot be set to
  % be exactly zero lest we get Divide-By-Zero errors, so before we begin
  % any calculations, clean those values now:
  mrktTaus   = ZeroClean(mrktTaus);
  mrktYields  = ZeroClean(mrktYields);

  params.zeta  = fminbnd(@(tmpZeta) NelsonSiegelSumErrors(tmpZeta),0,15);
  tmpBetas   = LeastSqrBetas(params.zeta);
  params.beta1= tmpBetas(1);
  params.beta2= tmpBetas(2);
  params.beta3= tmpBetas(3);

  %-----------------------------------------------------------------------
  % @description:  Function to get optimisation value for the fminbnd
  %        procedure when trying to predict optimum zeta value.
  function[f] = NelsonSiegelSumErrors(tmpZeta)
    [b,f] = LeastSqrBetas(tmpZeta);
  end

  %-----------------------------------------------------------------------
  % @description:  For some potential optimal zeta value, calculate the
  %         corresponding values of the betas, given that they can
  %         be calculated explicitly as a set of linear factors.
  %         This method subsequently appears to be far more precice
  %         than output from 'lsqcurvefit' and other such
  %         optimisation procedures, and vastly quicker as well!
  function[b,varargout] = LeastSqrBetas(tmpZeta)
    i = mrktTaus(:)/tmpZeta;
    j = 1-exp(-i);
    n = length(mrktTaus);
    z = [ones(n,1) j./i (j./i)+j-1];
    b = (z'*z)\(z'*mrktYields(:));
    e = mrktYields(:) - z*b;
    varargout(1) = {e'*e};
  end
end

function PlotAssetDynamics()
  firms = ParseCompanyList();
  for firm_i = 1 : 1 : length(firms)
    % For each firm, we need to get the bond name as it was stored in
    % csv, but then we will load up the ENTIRE firm/bond/financials
    % data which was saved by a previous precalculation process.
    tmpFirm  = firms(firm_i);
    PlotAssetDynamicsByFirm(tmpFirm.Bond.DSBondCode);
    clear tmpFirm;
  end
end

function PlotAssetDynamicsByFirm(dSBondCode)
  const = Constants(); paths    = PathInfo();
```

```
load([paths.PreCalcFirmHistory dSBondCode], 'firm');


disp(['Begin processing firm ' firm.CompName]);
[yrsKeys yrsVals] = dump(firm.Assets.MertonAssetParams);
yrsKeys  = cell2mat(yrsKeys);


xVals      = [];
yrlySigmaPP    = [];
yrlySigmaM    = [];
yrlySigmaLS    = [];


for yrInd = yrsKeys(1) : 1 : yrsKeys(end)
  xVals(end+1)  = datenum(['01/01/' num2str(yrInd)], ...
    const.DateStringAU);


  yrAssetParamsM  = get(firm.Assets.MertonAssetParams, yrInd);
  yrAssetParamsPP  = get(firm.Assets.PureProxyAssetParams, yrInd);
  yrAssetParamsLS  = get(firm.Assets.LSAssetParams, yrInd);


  yrlySigmaPP(end+1)  = yrAssetParamsPP.sigma;
  yrlySigmaM(end+1)  = yrAssetParamsM.sigma;
  yrlySigmaLS(end+1)  = yrAssetParamsLS.sigma;
end


close all;
finalW  = 15;  % Inches?
finalH  = 6;  % Inches?
rect = [0,0,finalW,finalH];
myPlot1 = figure('PaperPosition',rect);
hold on;
plot(xVals, yrlySigmaPP, ...
    [const.ColourPP const.LinePP const.PointPP]);
  plot(xVals, yrlySigmaM, [const.ColourM const.LineM const.PointM]);
  plot(xVals, yrlySigmaLS, [const.ColourLS const.LineLS const.PointLS]);
  axis([min(xVals),max(xVals),0,...
    1.05*max([yrlySigmaPP yrlySigmaM yrlySigmaLS])]);
  datetick('x','yyyy');
  titleText(1) = {'Yearly volatility \sigma estimate:'};
  titleText(2) = {[firm.CompName ' (' firm.Bond.MoodysRating ')']};
  legend(const.PlotLegendPP, const.PlotLegendM, ...
    const.PlotLegendLS,'Location','Best');
  title(titleText,'FontWeight','Bold');
  xlabel('Year','FontWeight','Bold');
  ylabel('Yearly Volatility','FontWeight','Bold');
  displayW  = 600; % Pixels?
  displayH  = displayW*finalH/finalW;
  set(myPlot1,'Position',[100,100,displayW,displayH]);
  destinationFile  = [paths.ThesisImages paths.YrlyAssDynPlotsPre ...
    firm.Bond.DSBondCode];

  % Pause so we can adjust the legend if need be
  pause;
  % Print at eps files for final work
  print('-depsc','-r300', destinationFile);
end

function PlotAssetPathEstimates()
  vasParams    = ParseInterestRateParamsVasicek();
  firms = ParseCompanyList();
  for firm_i = 1 : 1 : length(firms)
      tmpFirm  = firms(firm_i);
      PlotAssetPathEstimatesByFirmByYearRange(...
          vasParams,tmpFirm.Bond.DSBondCode,2002,2008);
      clear tmpFirm;
  end
end


function PlotAssetPathEstimatesByFirmByYearRange(...
    vasParams,dSBondCode,minYear,maxYear)
  close all
  const = Constants(); paths    = PathInfo();
  load([paths.PreCalcFirmHistory dSBondCode], 'firm');

  disp(['Begin processing firm ' firm.CompName]);
  [yrsKeys yrsVals] = dump(firm.Assets.MertonAssetParams);
  yrsKeys  = cell2mat(yrsKeys);

  for estimYr = yrsKeys(1) : 1 : yrsKeys(end)
    if estimYr >= minYear && estimYr <= maxYear
      % Our num shares outstanding and total liabs must come from the
      % previous year end of year balance sheet
      prevYr      = estimYr - 1;
      prevYrFinObs  = get(firm.Financials, prevYr);
      yrStartNum=datenum(['01/01/' num2str(estimYr)], const.DateStringAU);
      yrEndNum  =datenum(['31/12/' num2str(estimYr)], const.DateStringAU);

      disp(['Processing year: ' num2str(estimYr)]);
      % Parameters of asset dynamics for the year we are plotting
      estimYrAssetParamsM    =get(firm.Assets.MertonAssetParams, estimYr);
      ImplAssetValsMerton    = CalcCalendarYearImpliedAssetValues(...
          const.ModeMerton,vasParams,firm,estimYr,...
          estimYrAssetParamsM.mu,estimYrAssetParamsM.sigma);
      estimYrAssetParamsLS  = get(firm.Assets.LSAssetParams, estimYr);
```

```
ImplAssetValsLS      = CalcCalendarYearImpliedAssetValues(...
    const.ModeLS,vasParams,firm,estimYr,estimYrAssetParamsLS.mu,...
    estimYrAssetParamsLS.sigma);


xValsYrGen  = [];
yValsYrPP    = [];
yValsYrM    = [];
yValsYrLS    = [];
yValsYrE    = [];
yValsYrTL    = [];


for day_i = yrStartNum : 1 : yrEndNum
  if has_key(ImplAssetValsMerton, day_i)
    xValsYrGen(end+1)  = day_i;


    yValsYrM(end+1)    = get(ImplAssetValsMerton, day_i);
    yValsYrLS(end+1)  = get(ImplAssetValsLS, day_i);
    dailyEqtyObs    = get(firm.Equity,day_i);
    yValsYrE(end+1)=dailyEqtyObs.AdjClose*prevYrFinObs.OutStShares;
    yValsYrTL(end+1)  = prevYrFinObs.TotLiab;
    yValsYrPP(end+1)  = yValsYrE(end) + yValsYrTL(end);
  end
end


yValsYrM = yValsYrM/1000000;
yValsYrLS = yValsYrLS/1000000;
yValsYrE = yValsYrE/1000000;
yValsYrTL = yValsYrTL/1000000;
yValsYrPP = yValsYrPP/1000000;


close all;
```

```
      finalW  = 15;  % Inches?
      finalH  = 9;  % Inches?
      rect = [0,0,finalW,finalH];
      myPlot1 = figure('PaperPosition',rect);
      hold on;
      plot(xValsYrGen, yValsYrPP, [const.ColourPP const.LinePP]);
      plot(xValsYrGen, yValsYrM, [const.ColourM const.LineM]);
      plot(xValsYrGen, yValsYrLS, [const.ColourLS const.LineLS]);
      plot(xValsYrGen, yValsYrTL,[const.ColourTotLiab const.LineTotLiab]);
      plot(xValsYrGen, yValsYrE, [const.ColourEquity const.LineEquity]);
      titleText(1) = {'Implied Asset Value Paths:'};
      titleText(2) = {[firm.CompName ' (' firm.Bond.MoodysRating ')']};
      title(titleText,'FontWeight','Bold');
      xlabel('Year','FontWeight','Bold');
      ylabel('Asset Value ($Millions)','FontWeight','Bold');
      legend(const.PlotLegendPP, const.PlotLegendM, ...
        const.PlotLegendLS,const.PlotLegendTotLiab,...
        const.PlotLegendEquity,'Location','Best');
      datetick('x','yyyy');
      displayW  = 600; % Pixels?
      displayH  = displayW*finalH/finalW;
      set(myPlot1,'Position',[100,100,displayW,displayH]);
    end
  end

  destinationFile  = [paths.ThesisImages paths.AssPathPlotsPre ...
        firm.Bond.DSBondCode];
  pause;
  % Print at eps files for final work
  print('-depsc','-r300', destinationFile);
end
```

```
function PlotAssetVolatilityScatter()
  firms = ParseCompanyList();
  for firm_i = 1 : 1 : length(firms)
      tmpFirm  = firms(firm_i);
      PlotAssetVolatilityScatterByFirm(tmpFirm.Bond.DSBondCode);
      clear tmpFirm;
  end
end

function PlotAssetVolatilityScatterByFirm(dSBondCode)
  const = Constants(); paths    = PathInfo();
  load([paths.PreCalcFirmHistory dSBondCode], 'firm');

  disp(['Begin processing firm ' firm.CompName]);
  [yrsKeys yrsVals] = dump(firm.Assets.MertonAssetParams);
  yrsKeys  = cell2mat(yrsKeys);

  yrlySigmaPP    = [];
  yrlySigmaM    = [];
  yrlySigmaLS    = [];

  for yrInd = yrsKeys(1) : 1 : yrsKeys(end)
    yrAssetParamsM  = get(firm.Assets.MertonAssetParams, yrInd);
    yrAssetParamsPP  = get(firm.Assets.PureProxyAssetParams, yrInd);
    yrAssetParamsLS  = get(firm.Assets.LSAssetParams, yrInd);

    yrlySigmaPP(end+1)  = yrAssetParamsPP.sigma;
    yrlySigmaM(end+1)  = yrAssetParamsM.sigma;
    yrlySigmaLS(end+1)  = yrAssetParamsLS.sigma;
  end
```

```
[betas_PPM, betas_Int_PPM, R_PPM, R_Int_PPM, stats_PPM] = ...
  regress(yrlySigmaM',yrlySigmaPP')
R_sqr_PPM = stats_PPM(1);

[betas_PPLS, betas_Int_PPLS, R_PPLS, R_Int_PPLS, stats_PPLS] = ...
  regress(yrlySigmaLS',yrlySigmaPP')
R_sqr_PPLS = stats_PPLS(1);

close all;
finalW  = 20;  % Inches?
finalH  = 6.75;  % Inches?
rect = [0,0,finalW,finalH];
myPlot1 = figure('PaperPosition',rect);

% Calc min and max region to show in axis
axisMin = 0.9*min([yrlySigmaPP yrlySigmaM yrlySigmaLS])
axisMax = 1.05*max([yrlySigmaPP yrlySigmaM yrlySigmaLS])
% Calc xaxis values to plot fitted line
fitPP = [0 : 0.01 : axisMax];

% Plot Pure Proxy vs Merton Volatility estimates and a Linear fit
subplot(1,2,1);
hold on;
titleText(1) = {firm.CompName};
titleText(2) = {['R^2: ' num2str(R_sqr_PPM)]};
title(titleText);
scatter(yrlySigmaPP, yrlySigmaM);

plot(fitPP, fitPP*betas_PPM, 'r');
axis([axisMin axisMax axisMin axisMax]);
```

```
  xlabel('Volatility (Pure Proxy)','FontWeight','Bold');
  ylabel('Volatility (Merton MLE)','FontWeight','Bold');
  legend('Volatility Estimates', ...
    ['Fitted \beta_1: ' num2str(betas_PPM)],'Location','Best');

  subplot(1,2,2);
  hold on;
  titleText(1) = {','};
  titleText(2) = {['R^2: ' num2str(R_sqr_PPLS)]};
  title(titleText);
  scatter(yrlySigmaPP, yrlySigmaLS);
  plot(fitPP, fitPP*betas_PPLS, 'r');
  axis([axisMin axisMax axisMin axisMax]);
  xlabel('Volatility (Pure Proxy)','FontWeight','Bold');
  ylabel('Volatility (Longstaff&Schwartz MLE)','FontWeight','Bold');
  legend('Volatility Estimates', ...
    ['Fitted \beta_1: ' num2str(betas_PPLS)],'Location','Best');

  displayW  = 700; % Pixels?
  displayH  = displayW*finalH/finalW;
  set(myPlot1,'Position',[100,100,displayW,displayH]);

  pause;
  destinationFile  = [paths.ThesisImages paths.AssDynScatPre ...
    firm.Bond.DSBondCode];
  % Print at eps files for final work
  print('-depsc','-r300', destinationFile);
end

function PlotAssetVolatilityScatterAllData()
  const = Constants(); paths    = PathInfo();
```

```
allSigmaPP    = [];
allSigmaM    = [];
allSigmaLS    = [];

firms = ParseCompanyList();
for firm_i = 1 : 1 : length(firms)
  tmpFirm  = firms(firm_i);
  load([paths.PreCalcFirmHistory tmpFirm.Bond.DSBondCode], 'firm');

  disp(['Begin processing firm ' firm.CompName]);
  [yrsKeys yrsVals] = dump(firm.Assets.MertonAssetParams);
  yrsKeys  = cell2mat(yrsKeys);

  for yrInd = yrsKeys(1) : 1 : yrsKeys(end)
    yrAssetParamsM  = get(firm.Assets.MertonAssetParams, yrInd);
    yrAssetParamsPP  = get(firm.Assets.PureProxyAssetParams, yrInd);
    yrAssetParamsLS  = get(firm.Assets.LSAssetParams, yrInd);

    allSigmaPP(end+1)  = yrAssetParamsPP.sigma;
    allSigmaM(end+1)  = yrAssetParamsM.sigma;
    allSigmaLS(end+1)  = yrAssetParamsLS.sigma;
  end
end

[betas_PPM, betas_Int_PPM, R_PPM, R_Int_PPM, stats_PPM] = ...
  regress(allSigmaM', allSigmaPP');
betas_Int_PPM = betas_Int_PPM
R_sqr_PPM = stats_PPM(1);

[betas_PPLS, betas_Int_PPLS, R_PPLS, R_Int_PPLS, stats_PPLS] = ...
```

```
  regress(allSigmaLS', allSigmaPP');
betas_Int_PPLS = betas_Int_PPLS
R_sqr_PPLS = stats_PPLS(1);


close all;
finalW  = 15;  % Inches?
finalH  = 7.5;  % Inches?
rect = [0,0,finalW,finalH];
myPlot1 = figure('PaperPosition',rect);

% Calc min and max region to show in axis
axisMin = 0.9*min([allSigmaPP allSigmaM allSigmaLS])
axisMax = 1.05*max([allSigmaPP allSigmaM allSigmaLS])
% Calc xaxis values to plot fitted line
fitPP = [0 : 0.01 : axisMax];

% Plot Pure Proxy vs Merton Volatility estimates and a Linear fit
subplot(1,2,1);
hold on;
titleText(1) = {['R^2: ' num2str(R_sqr_PPM)]};
title(titleText);
scatter(allSigmaPP, allSigmaM);

plot(fitPP, fitPP*betas_PPM, 'r');
axis([axisMin axisMax axisMin axisMax]);
xlabel('Volatility (Pure Proxy)','FontWeight','Bold');
ylabel('Volatility (Merton MLE)','FontWeight','Bold');
legend('Estimated \sigmas', ...
  ['Fitted \beta_1: ' num2str(betas_PPM)],'Location','Best');

subplot(1,2,2);
```

```
  hold on;
  titleText(1) = {['R^2: ' num2str(R_sqr_PPLS)]};
  title(titleText);
  scatter(allSigmaPP, allSigmaLS);
  plot(fitPP, fitPP*betas_PPLS, 'r');
  axis([axisMin axisMax axisMin axisMax]);
  xlabel('Volatility (Pure Proxy)','FontWeight','Bold');
  ylabel('Volatility (Longstaff&Schwartz MLE)','FontWeight','Bold');
  legend('Estimated \sigmas', ...
    ['Fitted \beta_1: ' num2str(betas_PPLS)],'Location','Best');

  displayW  = 700; % Pixels?
  displayH  = displayW*finalH/finalW;
  set(myPlot1,'Position',[100,100,displayW,displayH]);

  pause;
  destinationFile  = [paths.ThesisImages paths.AssDynAllFirmsScat];
  % Print at eps files for final work
  print('-depsc','-r300', destinationFile);
end

function PlotBondYieldsAtIssue()
  const = Constants(); paths    = PathInfo();
  vasParams  = ParseInterestRateParamsVasicek();
  firms    = ParseCompanyList();

  compNames    = {};
  bondYieldsPPM  = [];
  bondYieldsM    = [];
  bondYieldsPPLS  = [];
  bondYieldsLS  = [];
```

```
bondYieldsAct  = [];
bondYieldsRF  = [];

for firm_i = 1 : 1 : length(firms)
  tmpFirm  = firms(firm_i);
  load([paths.PreCalcFirmHistory tmpFirm.Bond.DSBondCode], 'firm');
  clear tmpFirm;

  % Search into the future, one day at a time
  moveDaysBy      = 1;
  maxDist        = 40;
  actualPriceDtNum  = CalcClosestPossibleValuationDate(firm, ...
    vasParams, firm.Bond.IssueDateNum, moveDaysBy, maxDist);
  actualPriceDtStr  = datestr(actualPriceDtNum,const.DateStringAU);

  [remMrktTaus] = CalcRemainingCouponTausInYrs(actualPriceDtNum,...
    firm.Bond.CouponDateNums);
  faceVal    = 1;
  coupon    = firm.Bond.CouponRate;

  bondPricePPM = get(firm.Bond.PredBondPricesPPM, actualPriceDtNum);
  impliedYieldPPM = CalcImpliedYield(faceVal,...
    coupon,remMrktTaus,bondPricePPM);
  bondPriceM  = get(firm.Bond.PredBondPricesM, actualPriceDtNum);
  impliedYieldM = CalcImpliedYield(faceVal,coupon,...
    remMrktTaus,bondPriceM);
  bondPricePPLS = get(firm.Bond.PredBondPricesPPLS,actualPriceDtNum);
  impliedYieldPPLS = CalcImpliedYield(faceVal,coupon,...
    remMrktTaus,bondPricePPLS);
  bondPriceLS  = get(firm.Bond.PredBondPricesLS, actualPriceDtNum);
  impliedYieldLS = CalcImpliedYield(faceVal,coupon,...
    remMrktTaus,bondPriceLS);
  [bondPriceAct impliedYieldAct] = CalcActualPriceAndYield(...
    actualPriceDtNum,firm);
  bondPriceRF  = get(firm.Bond.PredBondPricesRF, actualPriceDtNum);
  impliedYieldRF = CalcImpliedYield(faceVal,coupon,...
    remMrktTaus,bondPriceRF);

  compNames(end+1) = {firm.CompName};
  bondYieldsPPM(end+1)  = impliedYieldPPM;
  bondYieldsM(end+1)    = impliedYieldM;
  bondYieldsLS(end+1)    = impliedYieldLS;
  bondYieldsPPLS(end+1)  = impliedYieldPPLS;
  bondYieldsAct(end+1)  = impliedYieldAct;
  bondYieldsRF(end+1)    = impliedYieldRF;
end

xVals = [1 : 1 : length(bondYieldsPPM)];

close all;
finalW  = 16;  % Inches?
finalH  = 10;  % Inches?
rect = [0,0,finalW,finalH];
myPlot1 = figure('PaperPosition',rect);

hold on;
minX = xVals(1)-1; maxX = xVals(end)+1;
minY = 0.03; maxY=1.05*max([bondYieldsPPM bondYieldsM bondYieldsPPLS ...
  bondYieldsLS bondYieldsAct bondYieldsRF]);

scatter(xVals,bondYieldsPPM, [const.ColourPPM const.PointPPM]);
scatter(xVals,bondYieldsM, [const.ColourM const.PointM]);
```

```
  scatter(xVals,bondYieldsPPLS, [const.ColourPPLS const.PointPPLS]);
  scatter(xVals,bondYieldsLS, [const.ColourLS const.PointLS]);
  scatter(xVals,bondYieldsAct, [const.ColourAct const.PointAct]);
  scatter(xVals,bondYieldsRF, [const.ColourRF const.PointRF]);
  axis([minX maxX minY maxY]);
  set(gca,'XTick',xVals)
  xlabel('Issuing Firm','FontWeight','Bold');
  ylabel('Yield to Maturity','FontWeight','Bold');
  legend(const.PlotLegendPPM, const.PlotLegendM, ...
    const.PlotLegendPPLS, const.PlotLegendLS, ...
    const.PlotLegendAct, const.PlotLegendRF, 'Location','Best');

  displayW  = 800; % Pixels?
  displayH  = displayW*finalH/finalW;
  set(myPlot1,'Position',[100,100,displayW,displayH]);
  pause;
  destinationFile  = [paths.ThesisImages paths.YieldsAllFirmsScat];
  % Print at eps files for final work
  print('-depsc','-r300', destinationFile);
end

function PlotHistoricBondYields()
  const = Constants(); paths    = PathInfo();
  vasParams    = ParseInterestRateParamsVasicek();

  firms = ParseCompanyList();
  for firm_i = 1 : 1 : length(firms)
    issueDateNums  = [];
    bondYieldsPPM  = [];
    bondYieldsM    = [];
    bondYieldsPPLS  = [];
```

```
bondYieldsLS  = [];
bondYieldsAct  = [];
bondYieldsRF  = [];

% For each firm, we need to get the bond name as it was stored in
% the csv, but then we will load up the ENTIRE firm/bond/financials
% data which was saved by a previous precalculation process.
tmpFirm  = firms(firm_i);
load([paths.PreCalcFirmHistory tmpFirm.Bond.DSBondCode], 'firm');
clear tmpFirm;
disp(['Begin processing firm ' firm.CompName]);
for yrInd = 2002 : 1 : 2008
  for mnthInd = 1 : 1 : 12
    % We wish to loop through several years, pricing each
    % firm's bond as close to the end of each month that we
    % can do so. Calculate the end day of each month:
    eom    = eomday(yrInd, mnthInd);
    tryToPriceDtNum =datenum([num2str(eom) '/' num2str(mnthInd) '/'...
      num2str(yrInd)], const.DateStringAU);

    try
      % Search into the future, one day at a time
      moveDaysBy      = 1;
      maxDist          = 14;
      actualPriceDtNum  = CalcClosestPossibleValuationDate(firm,...
        vasParams, tryToPriceDtNum, moveDaysBy, maxDist);
      actualPriceDtStr=datestr(actualPriceDtNum, const.DateStringAU);

      [remMrktTaus] = CalcRemainingCouponTausInYrs(...
        actualPriceDtNum, firm.Bond.CouponDateNums);
      faceVal    = 1;
    coupon    = firm.Bond.CouponRate;


    bondPricePPM=get(firm.Bond.PredBondPricesPPM,actualPriceDtNum);
    impliedYieldPPM = CalcImpliedYield(faceVal,coupon,...
      remMrktTaus,bondPricePPM);
    bondPriceM  = get(firm.Bond.PredBondPricesM, actualPriceDtNum);
    impliedYieldM = CalcImpliedYield(faceVal,coupon,...
      remMrktTaus,bondPriceM);
    bondPricePPLS = get(firm.Bond.PredBondPricesPPLS, ...
      actualPriceDtNum);
    impliedYieldPPLS = CalcImpliedYield(faceVal,coupon,...
      remMrktTaus,bondPricePPLS);
    bondPriceLS  = get(firm.Bond.PredBondPricesLS,actualPriceDtNum);
    impliedYieldLS = CalcImpliedYield(faceVal,coupon,...
      remMrktTaus,bondPriceLS);
    [bondPriceAct impliedYieldAct] = CalcActualPriceAndYield(...
      actualPriceDtNum,firm);
    bondPriceRF  = get(firm.Bond.PredBondPricesRF,actualPriceDtNum);
    impliedYieldRF = CalcImpliedYield(faceVal,coupon,...
      remMrktTaus,bondPriceRF);


    issueDateNums(end+1)  = actualPriceDtNum;
    bondYieldsPPM(end+1)  = impliedYieldPPM;
    bondYieldsM(end+1)    = impliedYieldM;
    bondYieldsLS(end+1)    = impliedYieldLS;
    bondYieldsPPLS(end+1)  = impliedYieldPPLS;
    bondYieldsAct(end+1)  = impliedYieldAct;
    bondYieldsRF(end+1)    = impliedYieldRF;
  catch
    disp('No price within range, ignore and continue processing');
  end
  end
end


close all;
finalW  = 15;  % Inches?
finalH  = 9;  % Inches?
rect = [0,0,finalW,finalH];
myPlot1 = figure('PaperPosition',rect);
hold on;
plot(issueDateNums,bondYieldsPPM, [const.ColourPPM const.LinePPM]);
plot(issueDateNums,bondYieldsM, [const.ColourM const.LineM]);
plot(issueDateNums,bondYieldsPPLS, [const.ColourPPLS const.LinePPLS]);
plot(issueDateNums,bondYieldsLS, [const.ColourLS const.LineLS]);
plot(issueDateNums,bondYieldsAct, [const.ColourAct const.LineAct]);
plot(issueDateNums,bondYieldsRF, [const.ColourRF const.LineRF]);


minX = min(issueDateNums)
maxX = max(issueDateNums);
minY = 1/2*min([bondYieldsPPM bondYieldsM bondYieldsPPLS ...
  bondYieldsLS bondYieldsAct bondYieldsRF]);
maxY = 1.02*max([bondYieldsPPM bondYieldsM bondYieldsPPLS ...
  bondYieldsLS bondYieldsAct bondYieldsRF]);
axis([minX maxX minY maxY]);
datetick('x','mmmyyyy');
titleText(1) = {'Implied bond yields over time:'};
titleText(2) = {[firm.CompName ' (' firm.Bond.MoodysRating ')']};
title(titleText,'FontWeight','Bold');
xlabel('Year','FontWeight','Bold');
ylabel('Yield to Maturity','FontWeight','Bold');
legend(const.PlotLegendPPM, const.PlotLegendM, ...
  const.PlotLegendPPLS, const.PlotLegendLS, const.PlotLegendAct, ...
```

```
      const.PlotLegendRF, 'Location','Best');
    displayW  = 600; % Pixels?
    displayH  = displayW*finalH/finalW;
    set(myPlot1,'Position',[100,100,displayW,displayH]);
    axis tight
    pause;

    destinationFile  = [paths.ThesisImages paths.YieldHistoryPlotPre ...
      firm.Bond.DSBondCode];
    % Print at eps files for final work
    print('-depsc','-r300', destinationFile);
  end
end

function PlotVasicekCurveFits()
  const = Constants(); paths    = PathInfo();
  vasParams  = ParseInterestRateParamsVasicek();

%   % Code to loop thru day by day over a range and check out how the
%   % yield curve changes
%   yrStartNum  = datenum('19/01/2001', const.DateStringAU);
%   yrEndNum  = datenum('19/01/2001', const.DateStringAU);
%   for day_i = yrStartNum : 1 : yrEndNum
%     PlotDailyVasicekCurveFit(const,paths,day_i,vasParams,'');
%   end

  % Retrieve all firms so we can load their issue dates
  firms = ParseCompanyList();
  for firm_i = 1 : 1 : length(firms)
    tmpFirm  = firms(firm_i);
    load([paths.PreCalcFirmHistory tmpFirm.Bond.DSBondCode], 'firm');
    actualPriceDtNum = CalcClosestPossibleValuationDate(firm, ...
      vasParams, firm.Bond.IssueDateNum, 1, 40);

    PlotDailyVasicekCurveFit(const,paths,actualPriceDtNum,...
      vasParams,firm.CompName, firm.Bond.DSBondCode);
  end

end

function PlotDailyVasicekCurveFit(const, paths, dateOfObsNum, ...
    vasParams, companyName, dsBondCode)
  if has_key(vasParams, dateOfObsNum)
    dailyVasParams  = get(vasParams,dateOfObsNum);
    mrktTaus  = [];
    mrktYields  = [];
    for m_i = 1 : 1 : 10*12
      month_field = ['m' num2str(m_i)];
      if isfield(dailyVasParams, month_field)
        mrktTaus(end+1) = m_i;
        mrktYields(end+1) = dailyVasParams.(month_field);
      end
    end
    mrktTaus = mrktTaus/12;  % Convert to years!

    guessPrices  = UnitDiscBondVasicek(mrktTaus,dailyVasParams);
    guessYields  = CalcDiscountBondYield(mrktTaus, guessPrices);

    close all;
    finalW  = 15;  % Inches?
    finalH  = 6;  % Inches?
    rect = [0,0,finalW,finalH];
    myPlot1 = figure('PaperPosition',rect);
    hold on;
    plot(mrktTaus, mrktYields, ...
      [const.ColourTY const.LineTY const.PointTY]);
    plot(mrktTaus, guessYields, [const.ColourRF const.LineRF]);
    axis([0, max(mrktTaus), 0, 1.05*max([mrktYields guessYields])]);
    legend(const.PlotLegendTY,const.PlotLegendRF,'Location','Best');
    xlabel('Maturity in Years','FontWeight','Bold');
    ylabel('Yield to Maturity','FontWeight','Bold');
%     titleText(1) = {'US Treasury Zero Yield Curve '};
%     titleText(2) = {['on ' datestr(dateOfObsNum)]};
    titleText(1) = {'US Treasury Zero Yield Curve versus Vasicek'};
    titleText(2) = {['Least-Squares fit on ' datestr(dateOfObsNum)]};
    titleText(3) = {['\kappa: ' num2str(dailyVasParams.kappa) ', ' ...
      '\theta: ' num2str(dailyVasParams.theta) ', ' ...
      'r_0: ' num2str(dailyVasParams.r0) ', ' ...
      '\eta: ' num2str(dailyVasParams.eta) ...
      ]};
    titleText(4) = {['for ' companyName]};
    title(titleText,'FontWeight','Bold');
    displayW  = 600; % Pixels?
    displayH  = displayW*finalH/finalW;
    set(myPlot1,'Position',[100,100,displayW,displayH]);
    pause;

    saveDateStr = [num2str(year(dateOfObsNum)) ...
      num2str(month(dateOfObsNum)) num2str(day(dateOfObsNum)) ...
      '_' dsBondCode];
%     destinationFile  = [paths.ThesisImages paths.VasLeastSqrFitPre ...
%       saveDateStr '_mrktObs'];
```

```
    destinationFile  = [paths.ThesisImages paths.VasLeastSqrFitPre ...
      saveDateStr];
    % Print at eps files for final work
    print('-depsc','-r300', destinationFile);
  end
end


function PreviewVasicekPredictions()
  const = Constants(); paths    = PathInfo();
  vasParams  = ParseInterestRateParamsVasicek();


  %% Show the instanteous interest rate as predicted,
  %% throughout time!!!!
  startdate = datenum('06/01/2001','dd/mm/yyyy');
  enddate = datenum('01/12/2008','dd/mm/yyyy');
  plotValues = [];
  for date_i = startdate : 1 : enddate
    if has_key(vasParams, date_i)
      dailyVasObs      = get(vasParams,date_i);
      plotValues(end+1).ObsDateNum  = dailyVasObs.ObsDateNum;
      plotValues(end).r0  = dailyVasObs.r0;
      plotValues(end).m1  = dailyVasObs.m1;
    end
  end
  xvals  = [plotValues(:).ObsDateNum];
  r0vals  = [plotValues(:).r0];
  m1vals  = [plotValues(:).m1];


  finalW  = 15;  % Inches?
  finalH  = 6;  % Inches?
  rect = [0,0,finalW,finalH];
  myPlot1 = figure('PaperPosition',rect);
  hold on;
  plot(xvals, r0vals, [const.ColourRF const.LineRF]);
  plot(xvals, m1vals, [const.ColourTY const.LineTY]);
  titleText(1)  = {'Predicted Instantaneous RF Rate (Vasicek model)'};
  titleText(2)  = {'versus US Treasury 1M Yields'};
  title(titleText);
  displayW  = 600; % Pixels?
  displayH  = displayW*finalH/finalW;
  set(myPlot1,'Position',[100,100,displayW,displayH]);
  datetick('x', 'mmmyy','keepticks');
  axis([min(xvals),max(xvals),0,1.05*max([r0vals m1vals])]);
  legend('Vasicek r_0', 'US Treasury 1Month', 'Location','Best');
  xlabel('Time','FontWeight','Bold');
  ylabel('Yield to Maturity','FontWeight','Bold');

  pause;
  destinationFile  = [paths.ThesisImages paths.VasR0VsTreasuryPlot];
  % Print at eps files for final work
  print('-depsc','-r300', destinationFile);
end

function TabulateAggregateHistoricPerformance()
  const = Constants(); paths    = PathInfo();
  vasParams    = ParseInterestRateParamsVasicek();

  bondPricesPPM  = [];
  bondPricesM    = [];
  bondPricesPPLS  = [];
  bondPricesLS  = [];
  bondPricesAct  = [];
```

```
bondPricesRF  = [];


bondYieldsPPM  = [];
bondYieldsM    = [];
bondYieldsPPLS  = [];
bondYieldsLS  = [];
bondYieldsAct  = [];
bondYieldsRF  = [];


% Retrieve all the preliminary details about all the firms for whom we
% wish to perform bond pricing:
firms = ParseCompanyList();
for firm_i = 1 : 1 : length(firms)
  % For each firm, we need to get the bond name as it was stored in
  % the csv, but then we will load up the ENTIRE firm/bond/financial
  % data which was saved by a previous precalculation process.
  tmpFirm  = firms(firm_i);
  load([paths.PreCalcFirmHistory tmpFirm.Bond.DSBondCode], 'firm');
  clear tmpFirm;
  disp(['Begin processing firm ' firm.CompName]);
  for yrInd = 2002 : 1 : 2008
    for mnthInd = 1 : 1 : 12
      % We wish to loop through several years, pricing each
      % firm's bond as close to the end of each month that we
      % can do so. Calculate the end day of each month:
      eom    = eomday(yrInd, mnthInd);
      tryToPriceDtNum  = datenum([num2str(eom) '/' ...
        num2str(mnthInd) '/' num2str(yrInd)], const.DateStringAU);


      try
        % Search into the future, one day at a time
```

```
moveDaysBy       = 1;
maxDist           = 14;
actualPriceDtNum  = CalcClosestPossibleValuationDate(firm,...
  vasParams, tryToPriceDtNum, moveDaysBy, maxDist);
actualPriceDtStr=datestr(actualPriceDtNum,const.DateStringAU);

[remMrktTaus] = CalcRemainingCouponTausInYrs(...
  actualPriceDtNum, firm.Bond.CouponDateNums);
faceVal         = 1;
coupon         = firm.Bond.CouponRate;

bondPricePPM=get(firm.Bond.PredBondPricesPPM,actualPriceDtNum);
impliedYieldPPM = CalcImpliedYield(faceVal,coupon,...
  remMrktTaus,bondPricePPM);
bondPriceM  = get(firm.Bond.PredBondPricesM, actualPriceDtNum);
impliedYieldM = CalcImpliedYield(faceVal,coupon,...
  remMrktTaus,bondPriceM);
bondPricePPLS = get(firm.Bond.PredBondPricesPPLS, ...
  actualPriceDtNum);
impliedYieldPPLS = CalcImpliedYield(faceVal,coupon,...
  remMrktTaus,bondPricePPLS);
bondPriceLS  = get(firm.Bond.PredBondPricesLS, actualPriceDtNum);
impliedYieldLS = CalcImpliedYield(faceVal,coupon,...
  remMrktTaus,bondPriceLS);
[bondPriceAct impliedYieldAct] = CalcActualPriceAndYield(...
  actualPriceDtNum,firm);
bondPriceRF  = get(firm.Bond.PredBondPricesRF,actualPriceDtNum);
impliedYieldRF = CalcImpliedYield(faceVal,coupon,...
  remMrktTaus,bondPriceRF);

bondPricesPPM(end+1)  = bondPricePPM;
```

```
          bondPricesM(end+1)    = bondPriceM;
          bondPricesPPLS(end+1)  = bondPricePPLS;
          bondPricesLS(end+1)    = bondPriceLS;
          bondPricesAct(end+1)  = bondPriceAct;
          bondPricesRF(end+1)    = bondPriceRF;

          bondYieldsPPM(end+1)  = impliedYieldPPM;
          bondYieldsM(end+1)    = impliedYieldM;
          bondYieldsLS(end+1)    = impliedYieldLS;
          bondYieldsPPLS(end+1)  = impliedYieldPPLS;
          bondYieldsAct(end+1)  = impliedYieldAct;
          bondYieldsRF(end+1)    = impliedYieldRF;
        catch
          disp('No price within range, ignore and continue processing');
        end
      end
    end
  end

  % Calculate pricing errors
  err_price_ppM    = (bondPricesPPM - bondPricesAct)./bondPricesAct;
  err_price_mleM  = (bondPricesM - bondPricesAct)./bondPricesAct;
  err_price_ppLS  = (bondPricesPPLS - bondPricesAct)./bondPricesAct;
  err_price_mleLS = (bondPricesLS - bondPricesAct)./bondPricesAct;
  % Calculate pricing error mean and std dev:
  err_price_ppM_m      = mean(err_price_ppM);
  err_price_ppM_std    = std(err_price_ppM);
  err_price_mleM_m    = mean(err_price_mleM);
  err_price_mleM_std  = std(err_price_mleM);
  err_price_ppLS_m    = mean(err_price_ppLS);
  err_price_ppLS_std  = std(err_price_ppLS);
```

```
err_price_mleLS_m    = mean(err_price_mleLS);
err_price_mleLS_std  = std(err_price_mleLS);

% Calculate yield errors
err_yield_ppM    = (bondYieldsPPM - bondYieldsAct)./bondYieldsAct;
err_yield_mleM  = (bondYieldsM - bondYieldsAct)./bondYieldsAct;
err_yield_ppLS  = (bondYieldsPPLS - bondYieldsAct)./bondYieldsAct;
err_yield_mleLS = (bondYieldsLS - bondYieldsAct)./bondYieldsAct;
% Calculate pricing error mean and std dev:
err_yield_ppM_m      = mean(err_yield_ppM);
err_yield_ppM_std    = std(err_yield_ppM);
err_yield_mleM_m    = mean(err_yield_mleM);
err_yield_mleM_std  = std(err_yield_mleM);
err_yield_ppLS_m    = mean(err_yield_ppLS);
err_yield_ppLS_std  = std(err_yield_ppLS);
err_yield_mleLS_m    = mean(err_yield_mleLS);
err_yield_mleLS_std  = std(err_yield_mleLS);

% Calculate yield diffs
diff_yield_ppM    = (bondYieldsPPM - bondYieldsAct);
diff_yield_mleM    = (bondYieldsM - bondYieldsAct);
diff_yield_ppLS    = (bondYieldsPPLS - bondYieldsAct);
diff_yield_mleLS  = (bondYieldsLS - bondYieldsAct);
% Calculate yield diff mean and std
diff_yield_ppM_m    = mean(diff_yield_ppM);
diff_yield_ppM_std  = std(diff_yield_ppM);
diff_yield_mleM_m    = mean(diff_yield_mleM);
diff_yield_mleM_std  = std(diff_yield_mleM);
diff_yield_ppLS_m      = mean(diff_yield_ppLS);
diff_yield_ppLS_std    = std(diff_yield_ppLS);
diff_yield_mleLS_m    = mean(diff_yield_mleLS);
```

```
  diff_yield_mleLS_std  = std(diff_yield_mleLS);


  displayM(1,1:3) = [err_price_ppM_m err_yield_ppM_m diff_yield_ppM_m];
  displayM(1,4:6) = [err_price_mleM_m err_yield_mleM_m diff_yield_mleM_m];
  displayM(2,1:3) = [err_price_ppM_std err_yield_ppM_std ...
    diff_yield_ppM_std];
  displayM(2,4:6) = [err_price_mleM_std err_yield_mleM_std ...
    diff_yield_mleM_std];

  displayM = displayM*100


  displayLS(1,1:3) =[err_price_ppLS_m err_yield_ppLS_m diff_yield_ppLS_m];
  displayLS(1,4:6) = [err_price_mleLS_m err_yield_mleLS_m ...
    diff_yield_mleLS_m];
  displayLS(2,1:3) = [err_price_ppLS_std err_yield_ppLS_std ...
    diff_yield_ppLS_std];
  displayLS(2,4:6) = [err_price_mleLS_std err_yield_mleLS_std ...
    diff_yield_mleLS_std];

  displayLS = displayLS*100
end

function TabulateAssetDynamics()
  const = Constants(); paths    = PathInfo();
  firms = ParseCompanyList();

  dataOut    = cell(0,0);

  dataCells = cell(0,0);
  dataCells(1,end+1)  = {'Company Name'};
  dataCells(1,end+1)  = {'Moodys Rating'};
dataCells(1,end+1)  = {'Year'};
dataCells(1,end+1)  = {'PP.mu'};
dataCells(1,end+1)  = {'PP.sigma'};
dataCells(1,end+1)  = {'PP.rho'};
dataCells(1,end+1)  = {'M.mu'};
dataCells(1,end+1)  = {'M.sigma'};
dataCells(1,end+1)  = {'M.rho'};
dataCells(1,end+1)  = {'LS.mu'};
dataCells(1,end+1)  = {'LS.sigma'};
dataCells(1,end+1)  = {'LS.rho'};
dataOut(end+1,:)  = dataCells(1,:);

for firm_i = 1 : 1 : length(firms)
  tmpFirm  = firms(firm_i);
  load([paths.PreCalcFirmHistory tmpFirm.Bond.DSBondCode], 'firm');
  clear tmpFirm;
  disp(['Begin processing firm ' firm.CompName]);

  [yrsKeys yrsVals] = dump(firm.Assets.MertonAssetParams);
  yrsKeys  = cell2mat(yrsKeys);

  for yrInd = yrsKeys(1) : 1 : yrsKeys(end)
    yrAssetParamsM  = get(firm.Assets.MertonAssetParams, yrInd);
    yrAssetParamsPP  = get(firm.Assets.PureProxyAssetParams, yrInd);
    yrAssetParamsLS  = get(firm.Assets.LSAssetParams, yrInd);

    dataCells = cell(0,0);

    dataCells(1,end+1)  = {firm.CompName};
    dataCells(1,end+1)  = {firm.Bond.MoodysRating};
    dataCells(1,end+1)  = {yrInd};
```

```
      dataCells(1,end+1)  = {yrAssetParamsPP.mu};
      dataCells(1,end+1)  = {yrAssetParamsPP.sigma};
      dataCells(1,end+1)  = {yrAssetParamsPP.rho};

      dataCells(1,end+1)  = {yrAssetParamsM.mu};
      dataCells(1,end+1)  = {yrAssetParamsM.sigma};
      dataCells(1,end+1)  = {yrAssetParamsM.rho};

      dataCells(1,end+1)  = {yrAssetParamsLS.mu};
      dataCells(1,end+1)  = {yrAssetParamsLS.sigma};
      dataCells(1,end+1)  = {yrAssetParamsLS.rho};

      dataOut(end+1,:)  = dataCells(1,:);
    end
    dataOut(end+1,end)  = {[]};
  end

  % Where do we want to save our results to?
  destination  = paths.TabularAssetDynamicsFile;
  WriteCellToCsv(destination, dataOut);
end

function TabulateBondYieldsAtIssue()
%--------------------------------------------------------------------------
% @description:  TabulateBondYieldsAtIssue
%        For the bond as close as possible to the issue date (within
%        a month, or it fails), perform pricing estimations and
%        save to a file.
%--------------------------------------------------------------------------
  const = Constants(); paths    = PathInfo();
```

```
vasParams  = ParseInterestRateParamsVasicek();
firms = ParseCompanyList();

dataOut    = cell(0,0);
dataCells  = cell(0,0);
dataCells(1,end+1)  = {'Company Name'};
dataCells(1,end+1)  = {'Moodys Rating'};
dataCells(1,end+1)  = {'Issue Date'};
dataCells(1,end+1)  = {'Priced Date'};
dataCells(1,end+1)  = {'r0'};
dataCells(1,end+1)  = {'RF.price'};
dataCells(1,end+1)  = {'RF.yield'};
dataCells(1,end+1)  = {'Act.price'};
dataCells(1,end+1)  = {'Act.yield'};
dataCells(1,end+1)  = {'M.PP.price'};
dataCells(1,end+1)  = {'M.PP.yield'};
dataCells(1,end+1)  = {'M.MLE.price'};
dataCells(1,end+1)  = {'M.MLE.yield'};
dataCells(1,end+1)  = {'LS.PP.price'};
dataCells(1,end+1)  = {'LS.PP.yield'};
dataCells(1,end+1)  = {'LS.MLE.price'};
dataCells(1,end+1)  = {'LS.MLE.yield'};
dataOut(end+1,:)  = dataCells(1,:);

for firm_i = 1 : 1 : length(firms)
  issueDateNums  = [];
  bondYieldsPPM  = [];
  bondYieldsM    = [];
  bondYieldsPPLS  = [];
  bondYieldsLS  = [];
  bondYieldsAct  = [];
```

```
bondYieldsRF  = [];


tmpFirm  = firms(firm_i);
load([paths.PreCalcFirmHistory tmpFirm.Bond.DSBondCode], 'firm');
clear tmpFirm;
disp(['Begin processing firm ' firm.CompName]);


% We wish to price the bond as early as possible on or after date of
% issue. To test whether pricing is possible for a particular date, we
% test that we have:
%  a) Observed equity prices for that date
%  b) Calculated instantaneous interest rate (and additional params)
%  on that date
%  c) An observed bond price on that date
%  d) Asset dynamics parameter estimates for the financial year prior
%  to pricing:
tryToPriceDtNum    = firm.Bond.IssueDateNum;
% Search into the future, one day at a time
moveDaysBy    = 1;
maxDist        = 40;
actualPriceDtNum  = CalcClosestPossibleValuationDate(firm, ...
  vasParams, tryToPriceDtNum, moveDaysBy, maxDist);


[remMrktTaus] = CalcRemainingCouponTausInYrs(actualPriceDtNum, ...
  firm.Bond.CouponDateNums);
faceVal    = 1;
coupon    = firm.Bond.CouponRate;


bondPricePPM = get(firm.Bond.PredBondPricesPPM, actualPriceDtNum);
impliedYieldPPM = CalcImpliedYield(faceVal,coupon,remMrktTaus,...
  bondPricePPM);
```

```
bondPriceM  = get(firm.Bond.PredBondPricesM, actualPriceDtNum);
impliedYieldM = CalcImpliedYield(faceVal,coupon,remMrktTaus,...
  bondPriceM);
bondPricePPLS = get(firm.Bond.PredBondPricesPPLS, actualPriceDtNum);
impliedYieldPPLS = CalcImpliedYield(faceVal,coupon,remMrktTaus,...
  bondPricePPLS);
bondPriceLS  = get(firm.Bond.PredBondPricesLS, actualPriceDtNum);
impliedYieldLS = CalcImpliedYield(faceVal,coupon,remMrktTaus,...
  bondPriceLS);
[bondPriceAct impliedYieldAct] = CalcActualPriceAndYield(...
  actualPriceDtNum,firm);
bondPriceRF  = get(firm.Bond.PredBondPricesRF, actualPriceDtNum);
impliedYieldRF = CalcImpliedYield(faceVal,coupon,remMrktTaus,...
  bondPriceRF);

[ActualBondPrice ActualYield] = CalcActualPriceAndYield(...
  actualPriceDtNum,firm);

issueDateNums(end+1)  = actualPriceDtNum;
bondYieldsPPM(end+1)  = impliedYieldPPM;
bondYieldsM(end+1)    = impliedYieldM;
bondYieldsLS(end+1)    = impliedYieldLS;
bondYieldsPPLS(end+1)  = impliedYieldPPLS;
bondYieldsAct(end+1)  = impliedYieldAct;
bondYieldsRF(end+1)    = impliedYieldRF;

vParamsAtObsDt    = get(vasParams, actualPriceDtNum);
disp(['Priced over rf rate of: ' num2str(vParamsAtObsDt.r0)]);
disp(['The bond was priced on: ' datestr(actualPriceDtNum)]);

%%%
```

```
    % Logic to save the data into a CSV file for analysis... probably
    % supercede this by using the values stored internally in the firm
    % objects once I get that graphing working in Matlab
    %%%%
    dataCells  = cell(0,0);
    dataCells(1,end+1)  = {firm.CompName};
    dataCells(1,end+1)  = {firm.Bond.MoodysRating};
    dataCells(1,end+1)  = {datestr(firm.Bond.IssueDateNum,...
      const.DateStringAU)};
    dataCells(1,end+1)  = {datestr(actualPriceDtNum,const.DateStringAU)};
    dataCells(1,end+1)  = {vParamsAtObsDt.r0};

    dataCells(1,end+1)  = {bondPriceRF};
    dataCells(1,end+1)  = {impliedYieldRF};
    dataCells(1,end+1)  = {ActualBondPrice};
    dataCells(1,end+1)  = {ActualYield};
    dataCells(1,end+1)  = {bondPricePPM};
    dataCells(1,end+1)  = {impliedYieldPPM};
    dataCells(1,end+1)  = {bondPriceM};
    dataCells(1,end+1)  = {impliedYieldM};
    dataCells(1,end+1)  = {bondPricePPLS};
    dataCells(1,end+1)  = {impliedYieldPPLS};
    dataCells(1,end+1)  = {bondPriceLS};
    dataCells(1,end+1)  = {impliedYieldLS};
    dataOut(end+1,:)  = dataCells(1,:);
  end
  % Where do we want to save our results to?
  destination  = paths.TabularBondIssuePricesFile;
  WriteCellToCsv(destination, dataOut);
end
```